INDEX

SESSION - 01

Fundamentals of Testing:

1. **Software Testing**
2. **Quality**
3. **Quality Assurance(SQA)**
4. **Quality Control**
5. **Verification**
6. **Validation**
7. **Static Testing**
8. **Dynamic Testing**
9. **Differences between QA and QC**
10. **Differences between Verification and Validation**
11. **Differences between Static Testing and Dynamic Testing**
12. **Testing principles**
13. **Differences between error, bug and defect**
14. **Differences between product and Project**
15. **Interview questions on Fundamentals of Testing**

What is Software Testing?

Software testing: is a process of evaluation a system to verify that is satisfies the customer requirements or identifies the deference b/w expected and actual results.

Testing is very important activity in product development lifecycle as it measures the quality of product and helps in determining production readiness of an application.

It checks whether all requirements are implemented correctly and defects non-conformances if any, before development.

Testing makes software predictable in nature, improves quality and reliability.

Software testing is process of executing a program or system with intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

Why software testing: It is important as it may cause mission failure, impact on operational performance and reliability if it is not done properly. If done poorly it leads to high maintenance cost and user dissatisfaction.

- ✓ To discover defects
- ✓ To avoid user detecting problems
- ✓ To prove that the software has no faults
- ✓ To ensure that product works as user expects
- ✓ To learn about the reliability of the software
- ✓ To find defects early, which helping in reduce the cost of defect fixing.

Why is Testing Important?

It is the primary duty of a software vendor to ensure that software delivered does not have any defects and that the customer's day to day operations do not get affected. This can be achieved by rigorously testing the software.

To protect an organization from any trouble and in order to address various risks involved during a change to an organization, testing is important. Risks can be related to the ones affecting reputation or resources or could be the ones leading to legal issues.

Objective of Testing: is to find defects in requirements, design and coding phases as early as possible.

Objective of Tester: to find the defects as early as possible and make sure they get fixed.

Testing Basics:

What is Quality?

Quality: Quality means customer satisfaction through bug free, delivered product on time, with in the budget, meets the customer requirements and expectations and is Maintainable.

Why Quality: Quality is the most important factor affecting an organization long-term performance.

Qualities of a great Tester?

- ✓ A great Tester should catch the majority of application defects.
- ✓ Communicate defects clearly to the team to fix it

Tester should have the following characteristics:

- ✓ Good Communication skills
- ✓ Detail orientation
- ✓ Ability to follow directions
- ✓ Ability to understanding of testing concepts
- ✓ Ability to innovate

Quality Assurance:

1. A Planned and Systematic pattern of actions necessary to provide adequate confidence that the product optimally fulfils customer expectations
   - ✓ Prevention of the bug in the process
   - ✓ Process oriented activity

- ✓ Assures quality
- ✓ It relates to process that produce products
- ✓ Identifies weakness in process and improve them

The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented (Or)

An activity which establishes and evaluates the process that produces products

Quality Control:

The process of exercising a product to identify differences between expected and actual behaviour.

- Finding the bugs
- Product oriented activity
- Ensures quality
- Relates to specific product
    - Identifies defects primarily for correcting defects (Or)

An activity which verifies whether or not the product meets standards

Question1: Do we involve in Quality Assurance?

If Answer is Yes what is our Role in Quality Assurance?

Ans:

- Produce the data for the Test Lead or Management to generate Metrics
- Provide accurate data on regular basis to generate accurate metrics

Note: To understand this process you should know about Team architecture and Roles and Responsibilities

Ex: Roles and Responsibilities of Quality Assurance Manager in your project, Test Lead, Tester

Difference between QA & QC:

	Quality Assurance	**Quality Control**
Focus on	QA aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.	QC aims to identify defects in the finished product. Quality control, therefore, is a reactive process.

Goal	The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.	The goal of QC is to identify defects after a product is developed and before it's released.
Statistical Techniques	Statistical Tools & Techniques can be applied in both QA & QC. When they are applied to processes (process inputs & operational parameters), they are called Statistical Process Control (SPC); & it becomes the part of QA.	When statistical tools & techniques are applied to finished products (process outputs), they are called as Statistical Quality Control (SQC) & comes under QC.
What	Prevention of quality problems through planned and systematic activities including documentation.	The activities or techniques used to achieve and maintain the product quality, process and service.
How	Establish a good quality management system and the assessment of its adequacy & conformance audit of the operation system & the review of the system itself.	Finding & eliminating couses of quality problems through tools & equipment so that customer's requirements are continually met.

What is verification?

Verification includes systematic procedures of review, analysis, and testing, employed throughout the software development life cycle, beginning with the software requirements phase and continuing through the coding phase.

Verification can be achieved by

- ✓ Feasibility Review
- ✓ Requirements Review
- ✓ Design Review
- ✓ Code Walkthrough
- ✓ Code Inspection
- ✓ Requirement Tracing

What is Validation?

- ✓ Is the process of checking the end product to assure that it meets requirements and expectations under operating conditions
- ✓ 'Done by executing the system functions through a series of Tests that can be observed and evaluated for compliance with expected Results

What is static Testing?

- ✓ Static Testing is a form of testing (is also known as Dry Run Testing) where the software is not actually used

- Syntax checking and manually reading the code to find errors are methods of static Testing.
- This type of Testing is mostly used by the developers
- Static Testing can be done before compilation
- They do not necessarily involve either manual or automated execution of the product being tested
- Examples include syntax checking, structured walkthroughs, and inspections.
- An inspection of a program occurs against a source code listing in which each code line is read line by line and discussed

What is dynamic Testing?

Dynamic Testing involves working with the software, giving input values and checking the output is expected

Dynamic Testing takes place only after compilation

Dynamic testing techniques are time dependent and involve executing a specific sequence of instructions on paper or by the computer.

Examples include structured walkthroughs, in which the program logic is simulated by walking through the code and verbally describing it.

Boundary testing is a dynamic testing technique that requires the execution of test cases on the computer with a specific focus on the boundary values associated with the inputs or outputs of the program.

Testing principles:

Principle 1: Testing shows presence of defects

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

Principle 2: Exhaustive testing is impossible

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Instead of exhaustive testing, we use risks and priorities to focus testing efforts.

Principle 3: Early Testing

Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.

Principle 4: Defect clustering

A small number of modules contain most of the defects discovered during pre-release testing or show the most operational failures.

Principle 5: Pesticide paradox

If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new bugs. To overcome this 'pesticide paradox', the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

Principle 6: Testing is context dependent

Testing is done differently in different contexts. For example, safety critical software is tested differently from an e-commerce site.

Principle 7: Absence-of-errors fallacy

Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

Introduction of Defect:

A defect is defined as a variance from the desirred product quality (Or)

Any situation where the system does not behave as indicated in the specification

A Program P is considered accurate with respect to a specification S,
If and only if.
For each valid input the output of P is in accordance with the Specification S

Developer Error

Injects Fault within the software

Fault causes software to fail

- **Error – A human action that produces an incorrect result**
- **Fault – A manifestation of an error within the software, also known as a defect or bug**
- **Failure – The departure of operational system behaviour from the user requirements**

Application:

Software developed with respective to requirements of a specific customer is called as application.

Product:

Product is something that is developed based on the company’s specifications and can be used by multiple customers.

Software developed with respective to market needs is called as product.

Usually the company will decide the requirements based on the generic requirements of so many customers in case of products

Project:

Project is something that is developed based on the particular customer's requirements and for his usage only.

Key Points:

- Software testing is a process of evaluating a system by identifying difference between expected and actual results
- Software testing is important to deliver a quality software product that satisfies user's requirements
- If done poorly, defects are found during operation which may lead to high maintance cost and customer dissatisfaction
- The goal of a software tester is to find bugs, find them as early as possible, and make sure that they get fixed.
- Develop master test plan so that resource and responsibilities are uderstood and assigned as early in the project as possible
- Verify that all project deliverables and components are complete.

Interview Questions:

1. Why we need Software Testing?
2. What is Quality?
3. What is Quality Assurance (SQA)?
4. What is Quality Control?
5. What is Verification?
6. What is Validation?
7. What is Static Testing?
8. What is Dynamic Testing?
9. Explain Differences between QA and QC
10. Explain Differences between Verification and Validation
11. Explain Differences between Static Testing and Dynamic Testing
12. What is Defect / Error ?

Testing Classifications:

Testing Classifications Software can be tested by running the programs and then verifying each step of its execution against expected result or by running examining the code against the stated requirements. These 2 distinct methods have led to the popularization of 2 techniques viz. static testing & dynamic testing as given below.

Static Testing:

This is a non-execution based testing technique. It could be done during any phase in the software development life cycle but largley it comes during requirement, design and coding phase. The design, code, testpaln, test cases or any document may be inspected and reviewed against a stated requirement/ standards/ some guidelines/ checklists. Static Testing includes.

- ✓ Informal Reviews → These reviews are generally done by a peer and occur on a need basis.
- ✓ Walk-through → Semiformal review facilitated by the author of the product
- ✓ Inspection → Formal reviews facilitated by a knowledge person who is not the author of the document.

Advantages:

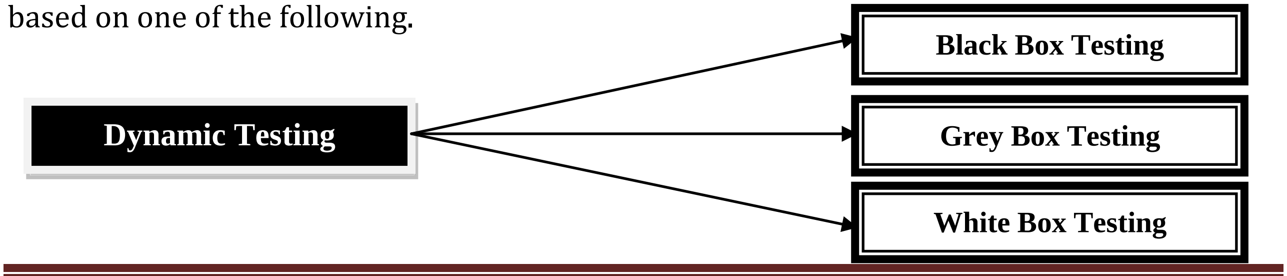
- ✓ Capture defects early, so saves cost.
- ✓ Checklist based approach
- ✓ Highest probability of finding defects
- ✓ Efficient way to education people regading the product
- ✓ Independent of test environment setups

Disadvantages:

- ✓ Time Consuming
- ✓ Can't test data dependences
- ✓ High skill levels required

Dynamic Testing:

This is an execution based testing technique. Here the program, module or the entire system is executed and the output is verified against the expected result. Dynamic execution of test is based on one of the following.



Dynamic Testing

Black Box Testing

Grey Box Testing

White Box Testing

White Box Testing:

This testing technique takes into account the internal structure of system or component. Complete access to the object's source code is needed for white box-testing. This is known as 'white box' testing because tester gets the internal working of the code.

White box testing helps to:

- ✓ Archive high code coverage
- ✓ Test program logic
- ✓ Elimainate redundant code
- ✓ Traverse complicated loop structures
- ✓ Cover control structures and sub-routines
- ✓ Evaluate diferent execution paths

Unit testing and some part of integration testing fall under white box testing category.

White box Testing Techniques:

Statement Coverage: This ensures atleast one execution of each statement in the code

Decision Coverage: This relates to testing of all branches of the code in the condition statements. Decision coverage is more effective than statement coverage. 100% decision coverage ensures 100% statement coverage but not vice versa.

Data Flow Testing: In this approach, each data variable is tracked and its use is verified. This approach will uncover bugs like variables not used intialized, declared and not used, etc. This will also be useful for verifying calculations.

Black Box Testing:

Black box testing is a strategy in which testing is based solely on the requirements and specifications. Unlike its complement, white box testing, black box testing requires no knowledge of the internal paths, structure, or implementation of the software under test (SUT).

The general black box testing process is:

1. The requirements or specifications are analyzed.
2. Valid inputs are chosen based on the specification to determine that the SUT processes them correctly. Invalid inputs must also be chosen to verify that the SUT detects them and handles them properly.
3. Expected outputs for those inputs are determined.
4. Tests are constructed with the selected inputs.
5. The tests are run.

6. Actual outputs are compared with the expected outputs.
7. A determination is made as to the proper functioning of the SUT

A Testing method where the application under test is viewed as a blck box and the internal behaviour of the program is completely ignored. Testing occurs based upon the requirement specifications.

- ✓ Black box testing method where is conducted more htan from user's perspective
- ✓ It focuses on the features and not the implementation
- ✓ Providers a big picture approach

Grey Box Testing:

- ✓ The Grey box testing is a combination of black box and white box testing.
- ✓ It is not fully black box testing because the tester should have the knowledge of internal design of the code.
- ✓ The application is also tested for its behavior and functionality which forms which is white box testing.
- ✓ It is also called as ―translucent testing.

Advantages of Dynamic Testing:

White Box Testing:

- ✓ Logic of the system tested
- ✓ Those parts which could have been omitted in black box testing are also getting covered
- ✓ Redundant code eliminated
- ✓ Cost effective when appropriate techniques are used

Black Box Testing:

- ✓ Simulations actual system usage
- ✓ Makes no assumptions about the system structure

Disadvantages:

White Box Testing:

- ✓ Does not ensure that all user requirements are met
- ✓ May not simulate real time situations
- ✓ Programming knowledge is needed

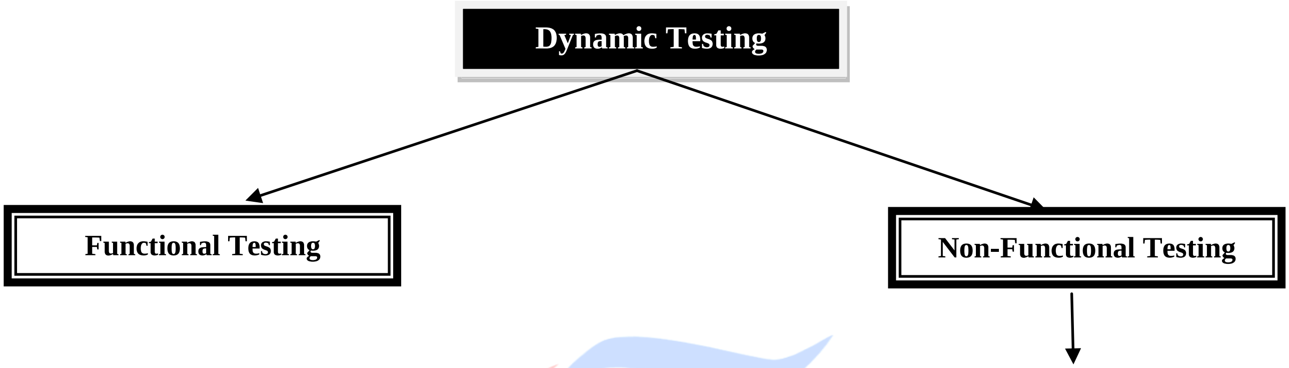
Black Box Testing:

- ✓ May miss out logical errors

- ✓ Chances of redundant testing is there
- ✓ Can't decide which part of code is not retting executed

TYPES OF SOFTWARE TESTING

Testing could be classified at a high level as functional testing and Non-Funcational testing, chart below is a snapshot of the different types of testing.



- ✓ Performance Testing
- ✓ Security Testing
- ✓ Usability Testing
- ✓ Compatibility Testing

Functional Testing:

Functional Testing refers to verifying if the module performs its intended functions in accordance with the specification. The purpose is to ensure that the application's behaviour is as expected. E.g. data entry, navigation, processing, retrivel and display based on requirements.

Determines the extent to which a product meets expected functional requirements through validation of product features. This process can be as simple as a smoke test to ensure primary functional operation, or as detailed as checking a verify of scenarios and validating that all output meets specified expectations

Performance Testing:

- ➢ Why we need performance Testing?
- ➢ What is performance testing?
- ➢ Performance Testing types

Why we need Performance Testing?

Speed: Does the application respond quickly enough for the intended users?
Scalability: Will the application handle the expected user load and beyond?

Stability: Is the application stable under expected and unexpected user loads?
Confidence: Are you sure that users will have a positive experience on go-live day?

Factors influencing the Speed
User Expectations

- Experience
- Psychology
- Usage

System Constraints

- Hardware
- Network
- Software

Costs

- Speed can be expensive

Factors influencing the Scalability
How many users…

- Before it gets "slow"?
- Before it stops working?
- Will it sustain?
- Do I expect today?
- Do I expect before the next upgrade?

How much data can it hold?

- Database capacity
- File Server capacity
- Back-up Server capacity
- Data growth rates

Factors influencing the Stability
What happens if…

- There are more users than we expect?
- All the users do the same thing?
- A user gets disconnected?
- The web server goes down?
- We get too many orders for the same thing

Factors influencing the Confidence
If you know what the performance is…

- You can assess risk.
- You can make informed decisions.
- You can plan for the future.
- You can sleep the night before go-live day.
- The peace of mind that it will work on go-live day alone justifies the cost of performance testing.

What is performance testing?

- Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device.
- Performance testing is a in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Reliability:

- The quality of measurement indicating the degree to which the measure is consistent,
- That is, repeated measurements would give the same results.

Resource Usage:

- Usage of application and system resources like CPU usage, memory.. Etc...

Factors that governs Performance testing

Throughput

- Capability of a product to handle multiple transactions in a given period.
- Throughput represents the number of requests/business transactions processed by the product in a specified time duration.
- Throughput is the measurement of bandwidth consumed during the test. It shows how much data is flowing back and forth from your servers.
- Throughput is measured in units of Kilobytes per Second.

Requests per Second

- RPS is the measurement of how many requests are being sent to the target server.
- It includes requests for HTML pages, CSS style sheets, XML documents, JavaScript libraries, images and Flash/multimedia files.

Concurrent Users

- A concurrent user is the most common way to express the load being applied during a test.
- This metric is measuring how many virtual users are active at any particular point in time.

Concurrent Users

- How many Real/virtual users are active at any particular point in time.

Error Rate

- Error Rate is the mathematical calculation that produces a percentage of problem requests to all requests.
- The percentage reflects how many responses HTTP status codes are indicating an error on the server, as well as any request that never gets a response.
- Normal codes are usually 200
- A common error code is 500, which means the web server knows it has a problem with fulfilling that request.

Response Time

- a generic system or functional unit takes to react to a given input

- Response time is defined as the delay between the point of request and the response from the product.

Performance testing types

- Load testing
- Stress testing
- Spike testing
- Endurance testing

Load testing:

- Verifying the application behavior under customer expected configuration and customer expected load
- The objective is to identify performance bottlenecks before the software application goes live.

Stress testing:

- Verifying the application behavior under customer expected configuration and more than the customer expected load
- Increasing the load sequentially (100, 200, 300 … etc..)
- Involves testing an application under extreme workloads to see how it handles high traffic or data processing .
- The objective is to identify breaking point of an application
- (To estimate the maximum peak load of the system)

Spike testing:

- Verifying the application behavior under customer expected configuration and more than the customer expected load
- Increasing the load randomly (100, 10000, 1000, 5000, 100000 … etc...)

- Involves testing an application under extreme workloads to see how it handles high traffic or data processing.
- The objective is to identify breaking point of an application
- (To estimate the maximum peak load of the system)

Endurance testing:

- Verify the application behavior under customer expected configuration and customer expected load over a long period of time.
- The objective is to identify memory leakages of an application

Bullet Points:

1) What is performance testing?
2) Performance counters
   Through put
   Response time
   Error rate
   Requests per second
   Concurrent users
3) Types of performance testing
   Load testing
   Stress Testing
   Spike Testing
   Endurance testing

Security Testing:

Need of Security Testing:

These days, websites are not only meant for publicity or marketing but these have also evolved into stronger tools to cater complete business needs. Web based Payroll systems, Shopping Malls, Banking, Stock Trade application are not only being used by organizations but are also being sold as products today.

This means that online applications have gained the trust of customers and users regarding their vital feature named as SECURITY. If an online system cannot protect the transaction data, no one will ever think of using it.

General Examples of security flaws in an application:

1) A Student Management System is insecure if ‘Admission’ branch can edit the data of ‘Exam’ branch

2) An ERP system is not secure if DEO (data entry operator) can generate 'Reports'
3) An online Shopping Mall has no security if customer's Credit Card Detail is not encrypted and can be viewed by anyone administering the website.
4) A custom software possess inadequate security if an SQL query retrieves actual passwords of its users

Recent examples:
In recent times, there have been numerous hacking incident causing major damages

- Hackers have stolen data from thousands of Citibank customers in the US, the bank has confirmed.
- Sony reports online security breach on various websites.
- Facebook breaches user privacy.
- EMC's security breach may cost customers $100m.
- US defence firm Lockheed Martin hit by cyber-attack.
- CBI India website hacked.

Wiki Leaks supporters hack Swedish govt website.

Security Testing Definition:

- "*Security means that authorized access is granted to protected data and unauthorized access is restricted*".
- Security testing is a process intended to reveal flaws in the security mechanisms of an information system that protect data and maintain functionality as intended.
- It is a type of non-functional testing.
- Security testing is basically a type of software testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
- It is a process to determine that an information system protects data and maintains functionality as intended.
- Software security is about making software behave in the presence of a malicious attack
- Typical security requirements may include specific elements of confidentiality, integrity, authentication, availability and authorization

1. **Confidentiality:** A security measure which protects against the disclosure of information to parties other than the intended recipient that is by no means the only way of ensuring the security.

2. **Integrity:** A measure intended to allow the receiver to determine that the information provided by a system is correct. Integrity schemes often use some of the same underlying technologies as confidentiality schemes, but they usually involve adding information to a communication, to form the basis of an algorithmic check, rather than the encoding all of the communication.
3. Authentication: This might involve confirming the identity of a person, tracing the origins of an artifact, ensuring that a product is what it's packaging and labeling claims to be, or assuring that a computer program is a trusted one.
4. Authorization: The process of determining that a requester is allowed to receive a service or perform an operation. Access control is an example of authorization.
5. **Availability:** Assuring information and communications services will be ready for use when expected. Information must be kept available to authorized persons when they need it.

Why Security Testing

- For Finding Loopholes
- For Zeroing IN on Vulnerabilities
- For identifying Design Insecurities
- For identifying Implementation Insecurities
- For identifying Dependency Insecurities and Failures
- For Information Security
- For Internet Technology Security
- For Communication Security

Security Testing Techniques:

1) Access to Application:

Whether it is a desktop application or a website, access security is implemented by 'Roles and Rights Management'. It is often done implicitly while covering functionality, E.g. in a Hospital Management System a receptionist is least concerned about the laboratory tests as her job is to just register the patients and schedule their appointments with doctors. So, all the menus, forms and screen related to lab tests will not be available to the Role of 'Receptionist'. Hence, the proper implementation of roles and rights will guarantee the security of access.

How to Test: In order to test this, thorough testing of all roles and rights should be performed. Tester should create several user accounts with different as well as multiple roles. Then he should use the application with the help of these accounts and should verify that every role has access to its own modules, screens, forms and menus only. If the tester finds any conflict, he should log a security issue so it can be fixed.

2. Password cracking:

The security testing on a web application can be started with “password cracking”. In order to log in to protected areas of the application, one can either guess a username/ password or use some password cracking tools for the same. Lists of common usernames and passwords are available in many open-source password cracker tools. If web application does not enforce a complex password it may not take very long to crack the username and password.

If username or password is stored in cookies without encryption, attacker can use different methods to steal the cookie information.

3. Data Protection:

There are further three aspects of data security. First one is that a user should be able to view or utilize only the data, which he is supposed to use. This is also ensured by roles and rights E.g. a TSR (telesales representative) of a company can view the data of available stock, but cannot see how much raw material was purchased for production.

Testing of this aspect is already explained above. The second aspect of data protection is related to how that data is stored in the DB. All the sensitive data must be encrypted to make it secure. Encryption should be strong especially for sensitive data like passwords of user accounts, credit card numbers or other business critical information. Third and last aspect is extension of this second aspect. Proper security measures must be adopted when flow of sensitive or business critical data occurs. Whether this data floats between different modules of same application or is transmitted to different applications, it must be encrypted to make it safe.

How to Test Data Protection:

The tester should query the database for ‘passwords’ of user account, billing information of clients, other business critical and sensitive data and should verify that all such data is saved in encrypted form in the DB. Similarly, (s)he must verify that between different forms or screens, data is transmitted after proper encryption. Moreover, the tester should ensure that the encrypted data is properly decrypted at the destination. Special attention should be paid to different ‘submit’ actions. The tester must verify that when the information is being transmitted between client and server, it is not displayed in the address bar of web browser in understandable format. If any of these verifications fail, the application definitely has security flaws that need to be plugged immediately.

4. URL manipulation through HTTP GET methods:

The tester should check if the application passes important information in the query string. This happens when the application uses the HTTP GET method to pass information between the client and the server. The information is passed via different parameters in the query string. The tester can modify a parameter value in the query string to check if the server accepts it.

Via HTTP GET request user information is passed to server for authentication or fetching data. Attacker can manipulate every input variable passed from this GET request to server in order to get the required information or to corrupt the data. In such conditions, any unusual behavior by application or web server is the doorway for the attacker to get into the application.

5. Brute-Force Attack:

Brute Force Attack is mostly done by some software tools. Using a valid user ID, the software attempts to guess the associated password by trying to login again and again. A simple example of security against such attack is account suspension for a short period of time as all the mailing applications like 'Yahoo' and 'Hotmail' do. If, a specific number of consecutive attempts (mostly 3) fail to login successfully, then that account is blocked for some time (30 minutes to 24 hours).

How to test Brute-Force Attack:

The tester must verify that some mechanism of account suspension is available and is working accurately. (S)He must attempt to login with invalid user IDs and Passwords alternatively to make sure that software application blocks the accounts that continuously attempt login with invalid information. If the application is doing so, it is secure against brute-force attack. Otherwise, this security vulnerability must be reported by the tester.
Above security aspects except URL manipulation should be taken into account for both web and desktop applications while, the following points are related with web based applications only.

6. SQL Injection and XSS (cross site scripting):

If user input data is crafted in SQL queries to query the database, attacker can inject SQL statements or part of SQL statements as user inputs to extract vital information from database. Sometimes, even if attacker is not successful to crash the application, from the SQL query error shown on browser, attacker can get the information they are looking for. Special characters from user inputs should be handled/escaped properly in such cases.

Conceptually speaking, the theme of both of these hacking attempts is similar, so these are discussed together. In this approach, malicious script is used by the hackers in order to manipulate a website. There are several ways to provide security blanket against such attempts. For all input fields of the website, field lengths should be defined small enough to restrict input of any script E.g. Last Name should have field length 30 instead of 255. There may be some input fields where large data input is necessary. For such fields proper validation of input should be performed prior to saving that data in the application. Moreover, in such fields any HTML tags or script tag input must be prohibited. In order to provoke XSS attacks, the application should discard script redirects from unknown or untrusted applications.

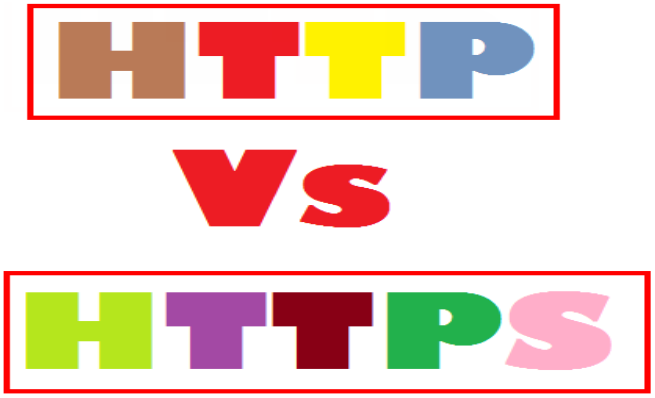
How to test SQL Injection and XSS:

The tester must ensure that maximum lengths of all input fields are defined and implemented. (S)He should also ensure that the defined length of input fields does not accommodate any script input as well as tag input. Both these can be easily tested E.g. if 20 is the maximum length specified for 'Name' field; and input string "<p>thequickbrownfoxjumpsoverthelazydog" can verify both these constraints. It should also be verified by the tester that application does not support anonymous access methods. In case any of these vulnerabilities exists, the application is in danger.

PHISHING:

Phishing is the act of attempting to acquire sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money) by masquerading as a trustworthy entity in an electronic communication. Communications purporting to be from popular social web sites, auction sites, banks, online payment processors or IT administrators are commonly used to lure unsuspecting public. Phishing emails may contain links to websites that are infected with malware. Phishing is typically carried out by email spoofing or instant messaging and it often directs users to enter details at a fake website whose look and feel are almost identical to the legitimate one. Phishing is example of social engineering techniques used to deceive users, and exploits the poor usability of current web security technologies. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and technical security measures.

In order to perform a useful security test of a web application, the security tester should have good knowledge of the HTTP protocol. It is important to have an understanding of how the client (browser) and the server communicate using HTTP. Additionally, the tester should at least know the basics of SQL injection and XSS. Hopefully, the number of security defects present in the web application will not be high. However, being able to accurately describe the security defects with all the required details to all concerned will definitely help



The security testing on a web application can be kicked off by “password cracking”. In order to log in to the private areas of the application, one can either guess a username/ password or use some password cracker tool for the same. Lists of common usernames and passwords are available along with open source password crackers. If the web application does not enforce a complex password (e.g. with alphabets, number and special characters, with at least a required number of characters), it may not take very long to crack the username and password.

If username or password is stored in cookies without encrypting, attacker can use different methods to steal the cookies and then information stored in the cookies like username and password.

Differences between HTTP and HTTPS:

Hypertext Transfer Protocol (**HTTP**) is a protocol used in networking. When you type any web address in your web browser, your browser acts as a client, and the computer having the requested information acts as a server. When client requests for any information from the server, it uses HTTP protocol to do so. The server responds back to the client after the request completes. The response comes in the form of web page which you see just after typing the web address and press “Enter”.

Hypertext Transfer Protocol Secure (**HTTPS**) is a combination of two different protocols. It is more secure way to access the web. It is combination of Hypertext Transfer Protocol (HTTPS) and SSL/TLS protocol. It is more secure way to sending request to server from a client, also the communication is purely encrypted which means no one can know what you are looking for. This kind of communication is used for accessing those websites where security is required. Banking websites, payment gateway, emails (Gmail offers HTTPS by default in Chrome browser), and corporate sector websites are some great examples where HTTPS protocols are used.

For HTTPS connection, public key trusted and signed certificate is required for the server. These certificate comes either free or it costs few dollars depends on the signing authority. There is one other method for distributing certificates. Site admin creates certificates and loads in the browser of users. Now when user requests information to the web server, his identity can be verified easily.

Here are some major differences between HTTP and HTTPS:

HTTP	HTTPS
URL begins with “http://”	URL begins with “https://”
It uses port 80 for communication	It uses port 443 for communication
Unsecured	Secured
Operates at Application Layer	Operates at Transport Layer
No encryption	Encryption is present
No certificates required	Certificates required

SUMMARY

A security test should be avoided on a production system.

The purpose of the security test is to discover the vulnerabilities of the web application so that the developers can then remove these vulnerabilities from the application and make the web application and data safe from unauthorized actions.

Usability Testing:

Usability –Testing the user friendliness of a product to determine the extent to which a product can be used by specified users to achieve specified goals with effectiveness (ability of user to perform work with system.),efficiency(speed and accuracy of use),and satisfaction in a specified context of use. This testing is clearly subjective. User interviews, surveys, video recording of user sessions, and other techniques can be used. Developers and testers are typically not appropriate as usability testers

Compatibility Testing:

Determines well a product works in conjunction with a verity of other products, on certain operating systems, across a board range of hardware and component configurations and when exposed to earlier versions of the product.

Compatibility –Testing the ability of two or more systems to exchange information. In a situation where the developed software replaces in already working program, An investigation should be conducted to assess possible compatibility problems between the new software and other programs or systems

General Testing Methods

Regression Testing:

Regression Testing is retesting the application after the bug fixes just to make sure that the bug has been fixed and its impact on rest of application. The selective retesting of the software system that has been modified to ensure that any defects have been resolved and that no other previously working functions have failed as a result of the reparations.

Regression tests are also used to verify that newly added features have not created problems with previous versions of the software. Regression test packages are made of subsets of the various types of test and takes place multiple times over the life of a system that represents a product. It occurs at all levels during the development life cycle to verify defect resolution and check for new defects. production system may be involved in regression testing when there is a significant chance that the system might be effected by the changes elsewhere, such as to adjacent applications or infrastructure.

Retesting:

Re-testing is the testing for a specific bug after it has been fixed.(one given by your definition).
2. Re-testing can be one which is done for a bug which was raised by QA but could not be found or confirmed by Development and has been rejected. So QA does a re-test to make sure the bug still exists and again assigns it back to them.

Re-testing is simply executing the test plan another time. The client may request a re-test for any reason - most likely is that the testers did not properly execute the scripts, poor documentation of test results, or the client may not be comfortable with the results. I've performed re-tests when the developer inserted unauthorized code changes, or did not document changes.
Regression testing is the execution of test cases "not impacted" by the specific project. I am currently working on testing of a system with poor system documentation (and no user documentation) so our regression testing must be extensive.

QA gets a bug fix, and has to verify that the bug is fixed. You might want to check a few things that are a “gut feel” if you want to and get away by calling it retesting, but not the entire function / module / product. * Development refuses a bug on the basis of it being “Non Reproducible”, then retesting, preferably in the presence of the Developer, is needed

Smoke Testing:

Smoke –Testing of application functionality made immediately after a build or issuance. A smoke test is scripted, shallow and wide test, usually designed to touch every part or the application in a cursory way without bothering with finer details. Pre and post build are issuance test results may be compare to detect build or issuance problem. A smoke test is similar to sanitary test but usually scripted and performed across a broader range rather than A narrower and detailed sanity test.

It is performed only when the build is ready every file is compiled, linked and combined into program is then put through a “smoke test”, a relatively simple check to see whether the product “smokes” when is runs.

Sanity Testing:

Sanity –Testing conducted at the beginning of a level of test (or each iteration or build) to determine if the quality of the product being delivered is performing well enough to start execution of the detailed testing. An example of a sanity testing would be can you get end- to –end through the application (doesn’t mean results are accurat)

Exploratory Testing/ Adhoc Testing:

Exploratory testing is an approach of software testing where there is simultaneous exploring the application, preparing the test design and test execution.

When performance this test there are no exact expected results, the tester decides what will be verified.

It is done when requirements are incomplete or there is a lack of time. Ad-hoc testing is part of Exploratory testing where the test is run once, unless a defect is discovered.

Different Types Of Testing For Quick Reference:

In the testing phase software undergoes various types of testing before it is shipped to the customer.

1. **Automation Testing:**
   Determines how well a product functions through a series of automated tasks, using a variety of tools to simulate complex text data
2. **Acceptance Testing:**
   Formal testing conducted to determine whether or not a system satisfies its acceptance criteria – enables a customer to determine whether to accept the system or not.

3. **Alpha Testing:**

   Testing of software product or system conducted at the developer's site by the customer.

4. **Automated Testing:**

   That part of software testing that is assisted with software tool(s) that does not require operator input, analysis, or evaluation.

5. **Beta Testing:**

   Testing Conducted at one or more customer site by the end user of a delivered software product system.

6. **Black box Testing:**

   Functional Testing based on the requirements with no knowledge of the internal with no knowledge of the internal program structure or data. Also known as closed box testing.

7. **Bottom-Up Testing:**

   An integration testing technique that tests the low level components first using test drivers for those components that have not yet been developed to call the low level components for test.

8. **Clear-Box Testing:**

   Another team for White-Box Testing structural Testing is sometimes reoffered to clear-box testing; since "white boxes" are considered opaque and do not really permit visibility into the code. This is also known as glass-box or open-box Testing

9. **Compatibility Testing:**

   Determines well a product works in conjunction with a verity of other products, on certain operating systems, across a board range of hardware and component configurations and when exposed to earlier versions of the product.

10. **Database Testing:**

    Most web sites f any complexity store and retrieve information from some type of database. Clients often want us to test the connection between their web site and database in order to verify data and display integrity.

11. **Dynamic Testing:**

    Verification or Validation Testing performed which executes the system code.

12. **Functional Testing:**

    Determines the extent to which a product meets expected functional requirements through validation of product features. This process can be as simple as a smoke test to ensure primary functional operation, or as detailed as checking a verify of scenarios and validating that all output meets specified expectations.

13. **Functional Localization Testing:**

    Determines how well a product functions across a range of language, Localized versions are checked to determine whether particular language translations create failures specific to that language versions.

14. **Heuristics Testing:**
    Another term for fault-directed testing.
15. **Hybrid Testing:**
    A combination of top-down testing combined with bottom- up testing of prioritized or available components.
16. **Integration Testing:**
    An orderly progression of testing in which the software components or hardware components, or both are combined and tested until the entire system has been integrated.
17. **Interoperability Testing:**
    Determines to a deeper extent than compatibility testing, how well a product works with a specific cross section of external components such as hardware, device drivers, second –party software and even specific operating systems and factory delivered computer systems.
18. **Intrusive Testing:**
    Testing that collects timing and processing information during program execution that may change the behavior of the software from its behavior in a real environment.
19. **Install Testing:**
    Determines how well and how easily a product installs on a variety of platform configurations.
20. **Load Testing:**
    Determines how well a product functions when it is in competition for system resources. The competition for system resources. The competition most commonly comes from active processes, CPU utilization, I/O activity, and Network traffic or memory allocation.
21. **Manual Testing:**
    That part of software testing that requires operator input, analysis, or evaluation.
22. **Mutation Testing:**
    Amethod to determine test set thoroughness by measuring the extent to which a test set can discriminate the program from slight variants of the program.
23. **Mundane Testing:**
    A test that include many simple and repetitive steps, it can be called as Manual Testing
24. **Operational Testing:**
    Testing performed by the end user on software in its normal operating environment.
25. **Path coverage Testing:**
    A test method satisfying coverage criteria that each logical path through the program is tested. Paths through the program often are grouped into finite set of classes. One path from each class is tested.

26. **Performance testing:**

Determines how quickly a product executes a verify of events. This type of testing sometimes includes reports on response time to a user's command, system throughput or latency. Although the word performance has various meanings, Eg: Speed

27. **Qualification Testing:**

Formal Testing usually conducted by the developer for the customer. To demonstrate that the software meets its specified requirements.

28. **Random Testing:**

An essentially black-box testing approach in which a program is tested by randomly choosing a subset of all possible input values. The distribution may be arbitrary or distribution of inputs in the application environment.

29. **Regression Testing:**

Selective re-testing to detect faults introduced during modification of a system or system component to verify that modifications have not caused unintended adverse effects, or to verify that a modified system or system compent still meets its requirements.

30. **Smoke Testing:**

It is performed only when the build is ready every file is compiled, linked and combined into program is then put through a "smoke test", a relatively simple check to see whether the product "smokes" when is runs.

31. **Statement Coverage Testing:**

A test method satisfying coverage criteria that requires each statement be executed at least once.

32. **Static Testing:**

Verification performed without executing the system's code. Also called static analysis.

33. **Stress Testing:**

Determines, to a deeper extent than load testing, how well a product functions when a load is placed on the system resources that exceeds their capacity, Either stress testing can also determine the capacity of a system by increasing the load placed on the resources until a failure or other unacceptable product behavior occurs. Stress Testing can also involve placing loads on the system for extended periods.

34. **System Testing:**

The process of testing an integrated hardware and software system to verify that the system meets ist specified requirements.

35. **System Integration Testing:**

Determine though isolation, which component of a product is the roadblock in the development process. This testing is beneficial to products that come together though a series of builds where each step in the development process has the potential to intoducea problem. System integration testing is also used in systems composed of hardware and software. In essence, system integration testing is intended to exercise the whole system in real world scenarios and, again through isolation, determine which components is responsible for a certain defect.

36. **Top-Down Testing:**

An integration testing technique that test the high-level components first using stubs for lower-level called components that have not yet been integrated and that stimulate the required actions of those components.

37. **Unit Testing:**

The testing done to show whether a unit (the smallest piece of software that can be independently compiled or assembled, Loaded, and tested) satisfies its functional specification or its implementation structure matches the intended design structure.

38. **White box Testing:**

Testing approaches that examine the program structure and derive test data from the program logic.

39. **Web site Testing:**

Compatibility Testing: Compatibility Testing tests your web site across a wide variety browser/operating system combinations. This testing typically exposes problems with plug-ins. ActiveX controls, java applets

Levels of Testing:

1. Different levels of testing.

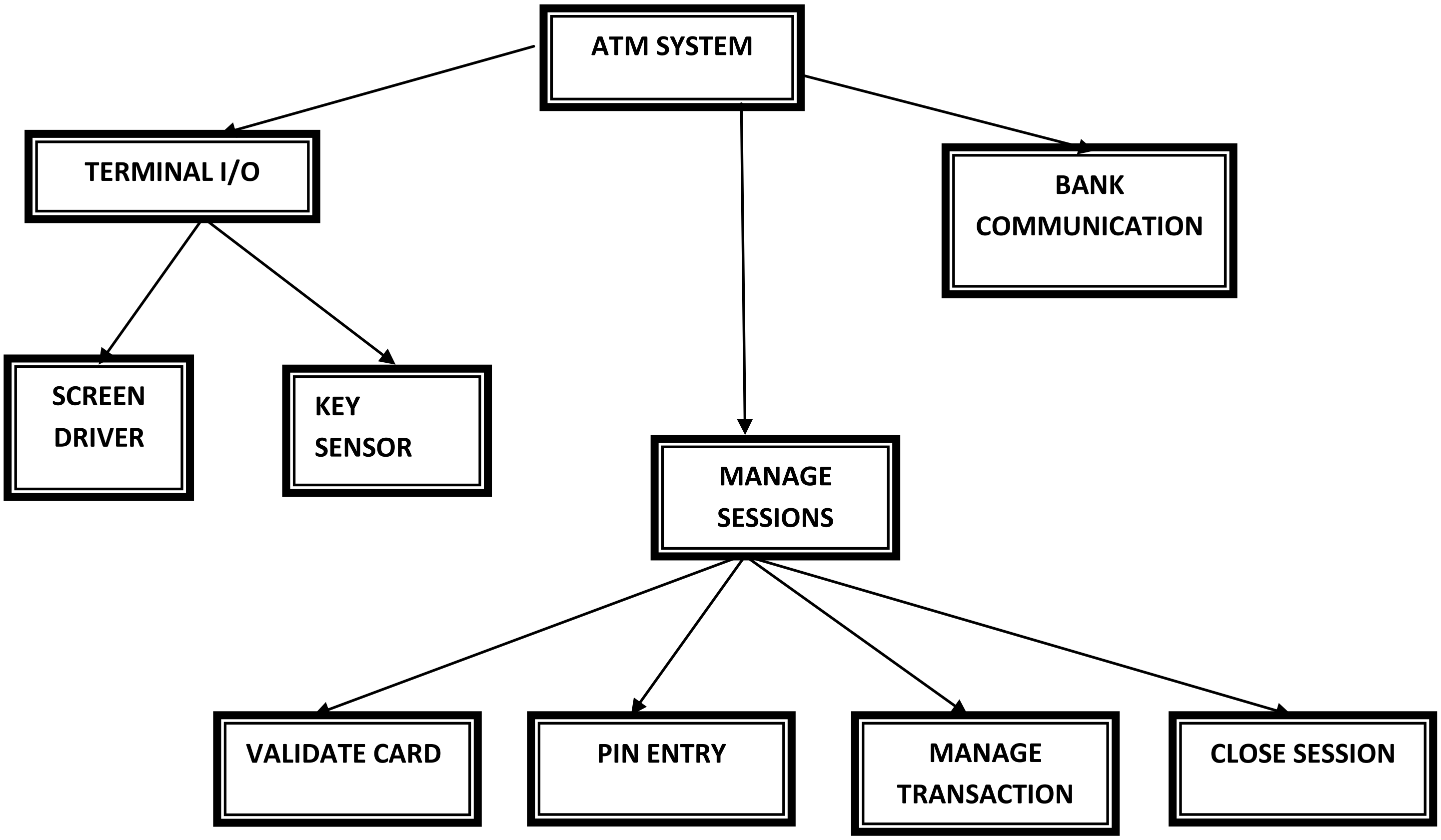
2. Important factors on testing levels.

- ✓ Factors influencing test scope
- ✓ Why test at different levels?

3. Test levels and software life - cycle models.

"V" Model

Decomposition tree of ATM



Different Levels of testing

What is a level of test?

- ✓ Defined by a given environment
- ✓ Environment is a collection of people, hardware, software, interfaces, data etc.

Levels of Testing:

- ✓ Unit Testing
- ✓ Integration Testing
- ✓ System Testing

- ✓ Acceptance Testing
- ✓ Regression testing

Unit Testing

- ➢ A unit is smallest testable piece of software
  - ✓ can be compiled, linked, loaded
  - ✓ e.g functions/procedures, classes, interfaces
  - ✓ normally done by programmer
  - ✓ Test cases written after coding

Disadvantage

  - ✓ Test cases -
    ritten to suit programmer's implementation (not necessarily specification)
- ➢ **Better to use "Buddy Testing"**

Buddy Testing

- ➢ **Team approach to coding and testing**
- ➢ **One programmer codes the other tests and vice versa**
  Test cases - written by tester (before coding starts). Better than single worker approach
  - ✓ **Objectivity**
  - ✓ **Cross – training**
  - ✓ **Models program specification requirement**

Normally in programmers IDE (comfort zone)

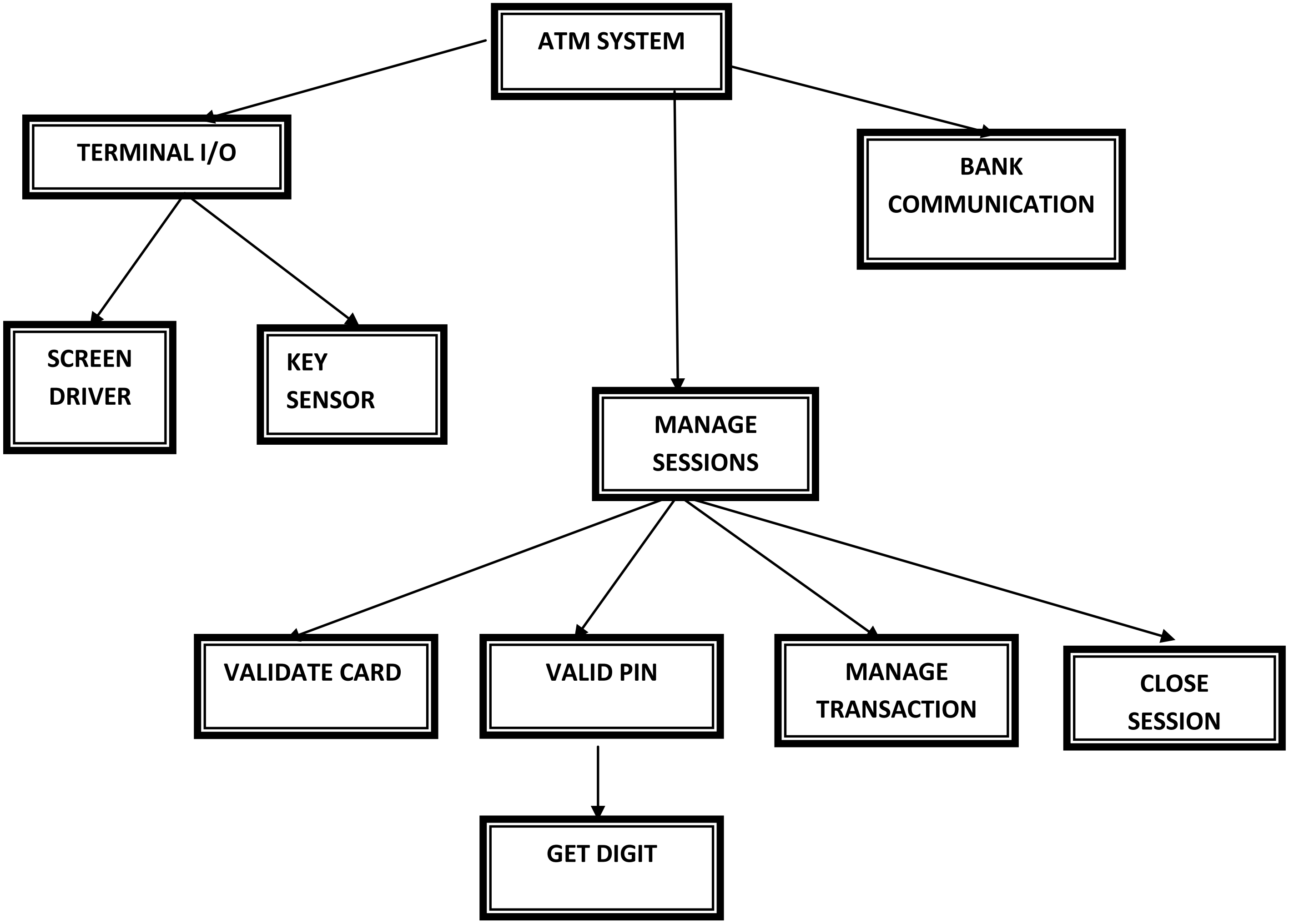
- ➢ **Find unit bugs**
  - ✓ Wrong implementation of functional specs
- ➢ **ATM Example**
  - ✓ Testing function procedures e.g. the ValidatePIN() procedure

Integration Testing

- ➢ Test for correct interaction between system units
- ➢ Systems built by merging existing libraries
- ➢ modules coded by different people
- ➢ Mainly tests the interfaces among units
- ➢ Bottom up integration testing
  - ✓ Use of drivers
- ➢ **Top down integration testing**
  - ✓ Use of stubs
- ➢ **Who does integration testing and when is it done?**
  - ✓ Done by developers/testers
  - ✓ Test cases written when detailed specification is ready
  - ✓ Test continuous throughout project

- **Where is it done?**
  - ✓ done on programmer's workbench
- **Why is it done?**
  - ✓ Discover inconsistencies in the combination of units.

- **Stubs and Drivers**



System Testing

Objective

- The objective of Functional System Test is to validate that the complete application meets business and functional requirements (e.g., testing all the functions of a trade order management application)

Scope

- Focuses on validating that application functionality supports application requirements
- Organize by business event and supporting business processes
- Includes testing different data scenarios that are driven by different business needs or requirements

- Most interfaces between applications are either stubbed-off or simulated to facilitate component based testing
- Opportunities to interfaces with other systems (external system test) prior to the start of Integrated System Test are done on a case by case basis
- Test of overall interaction of components
- Find disparities between implementation and specification
- Usually where most resources go to
- Involves – load, performance, reliability and security testing

- **Who performs system testing and when is it done?**
  - Done by the test team
  - Test cases written when high level design spec is ready
- **Where is it done?**
  - Done on a system test machine
  - Usually in a simulated environment e.g. vmware
- **System vs. Integration testing**
  - What vs. How
  - Requirement spec -> what
  - detailed design spec -> how
  - System testing functional not structural

Example: System - Integration Testing

Objective

- The objectives of Technical Integrated System Test are:
  - To validate that the suite of applications can process targeted average and peak transaction volumes
  - To verify that response times between applications fall within defined target ranges (e.g., information requested from system A by system B is provided in a timely manner)
  - To validate that service level agreements (SLAs) are met in a production-like configuration

Scope

- Focuses on validating that application functionality meets technical performance targets across all systems
- Occurs across the application suite (not individual applications)
- Includes stress / volume and response time tests
- Includes testing on supported system platforms

Validation Source

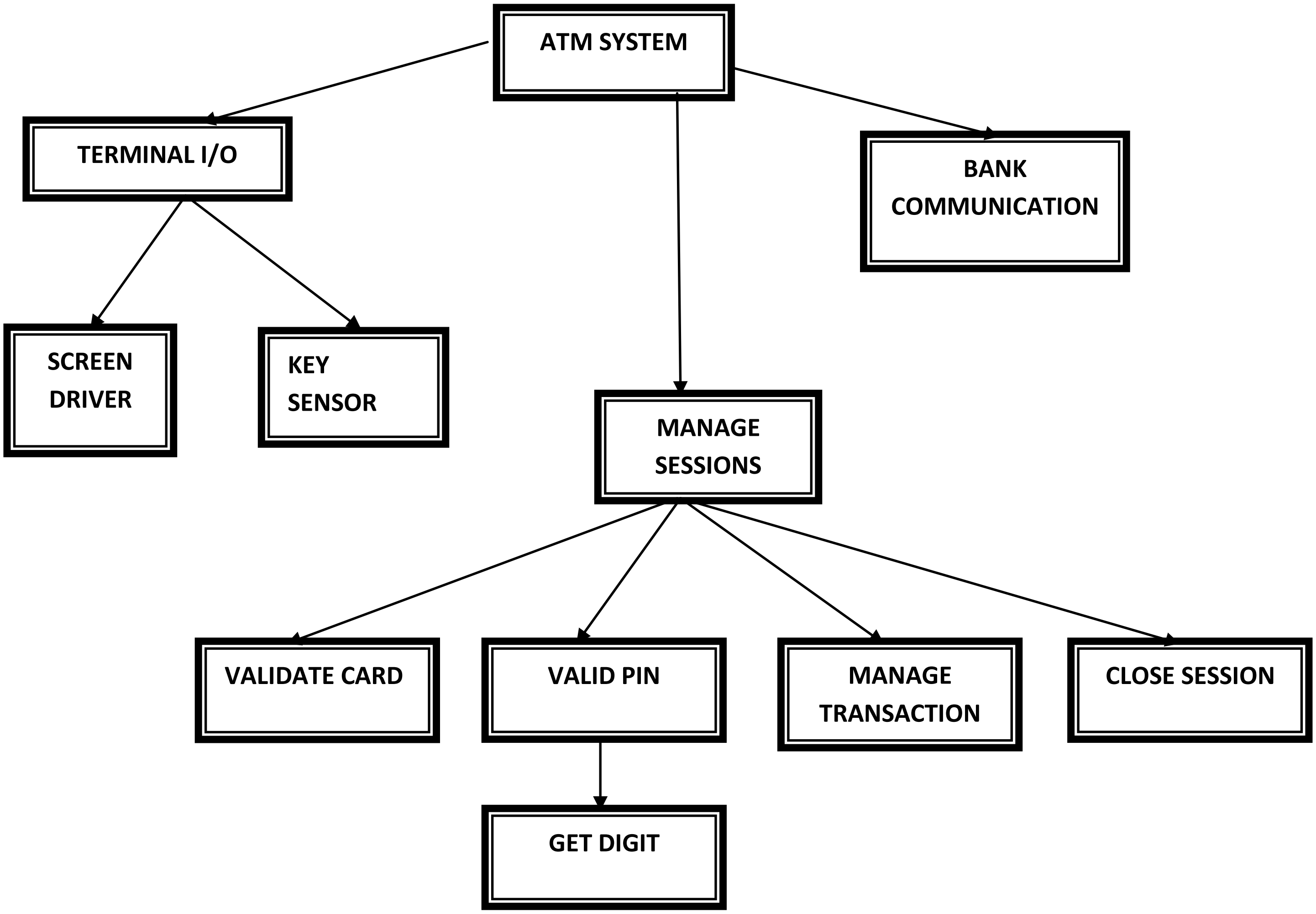
- Interface specifications from Design phase

Resources

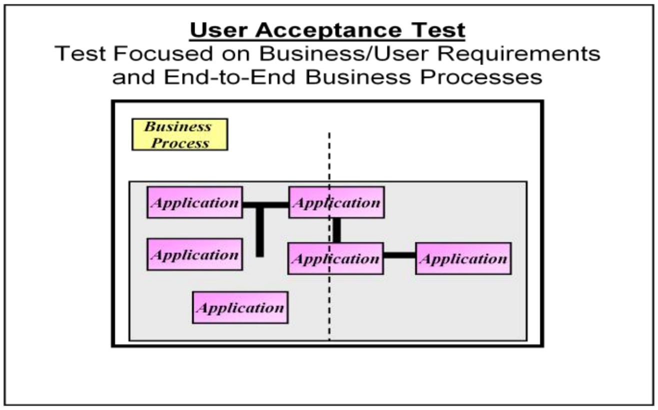
- Test Planning: performed by designers
- Test Execution: performed by Integrated System Test resources

Deliverables

- Integrated System Test Detailed Plan (Technical) – including objectives and scope, regression testing approach, risks, environment requirements, metrics, entry/exit criteria, test resources and work plan
- Integrated System Test Cases, Test Scripts and Test Data
- Configured Integrated System Test Environment
- Executed Integrated System Tests/ Actual Results



User Acceptance Testing



- Demonstrates satisfaction of user
- Users are essential part of process
- Usually merged with System Testing
- Done by test team and customer
- Done in simulated environment/real envi ronment

Objective

- The objective of User Acceptance Test (UAT) is to validate that the integrated suite of applications meets business/user requirements and supports business processes.

Scope

- Focuses on validating the business/user requirements are properly implemented and support business needs and the business processes employed by users
- Users (re)validate results of integrated system test
- Users validate other quality requirements of application are met (e.g., system usability, reference data / report integrity and system performance)

Validation Source

- Business/user requirements from Define phase

Resources

- Test Planning: performed by Business Analysts and users
- Test Execution: performed by User Acceptance Testing resources

Deliverables

- User Acceptance Test Detailed Plan – including objectives and scope, regression testing approach, risks, environment requirements, metrics, entry/exit criteria, test resources and work plan
- User Acceptance Test Cases, Test Scripts and Test Data
- Configured User Acceptance Test Environment
- Executed User Acceptance Tests/ Actual Results
- Updated Defect Log
- User Acceptance Test Team Management & Reports
- Defect Management & Reports
- Defect Fixes
- Sign-off on User Acceptance Test Entry/Exit Criteria

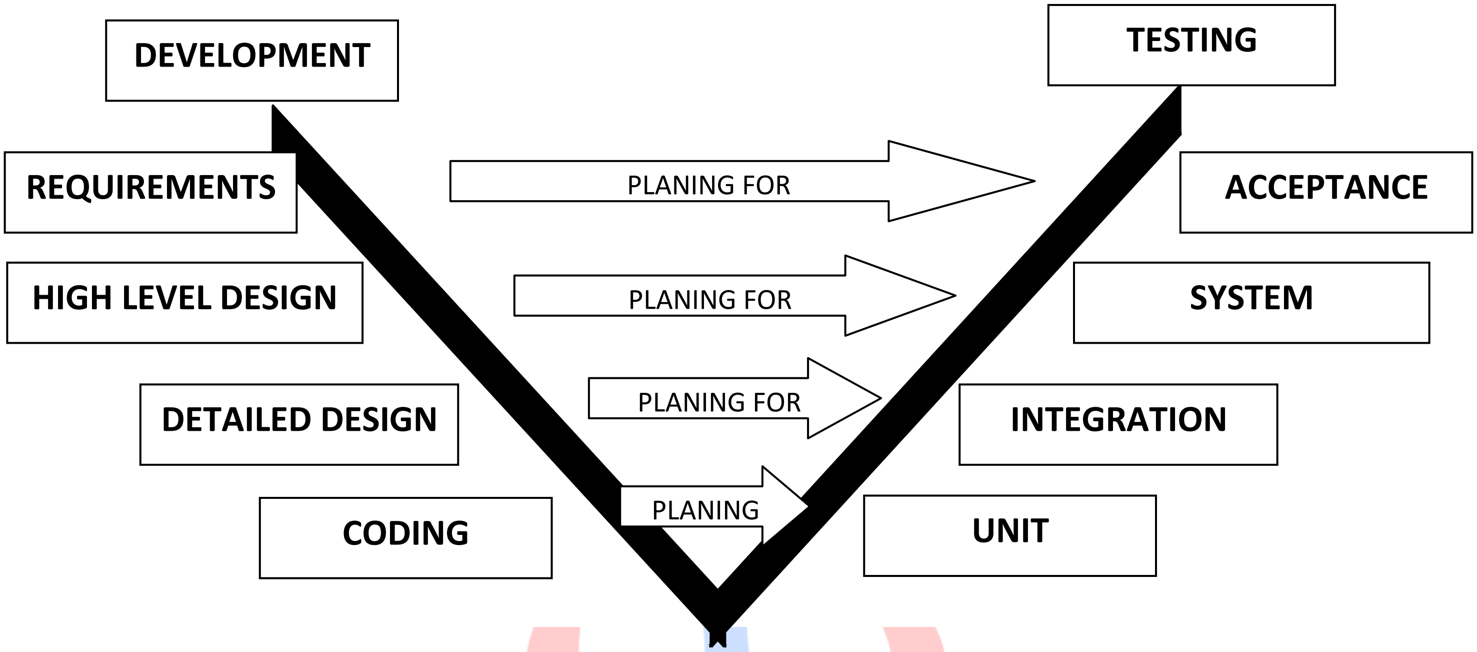
Factors influencing test scope

- Size of project
- Complexity of project
- Budget for project
- Time scope for project
- Number of staff

Why test at different levels

- Software development naturally split to phases
- Easily track bugs
- Ensures a working subsystem/ component/ library
- Software reuse more practical Contents

The “V” model and test levels:



SESSION - 02

SOFTEWARE TESTING LIFE CYCLE:

Fundamentals of Knowledge Transition

Agenda of the Session:

- Knowledge Transition Concepts
- Learning Objectives
- Sources of knowledge transition
- Methodology of Knowledge Transition
- Transition Planning
- Knowledge Acquisition
- Reverse Knowledge Transition

==

Introduction: This module provides a detailed description about the various activities involved in Knowledge Transition and Requirements Analysis Phase.

Introduction to Knowledge Transition and Requirement Analysis

- ✓ Knowledge Transition and Requirements Analysis are major activities in the initial stage of the testing life cycle and are critical to the success of a project.
- ✓ Knowledge Transition (KT) is the process of the project team gaining information on the application, business process and testing processes from the client/SME at the beginning of a project.
- ✓ As testers, it is important that the knowledge on the product/service/application is gathered and assimilated by the tester just after the project kicks off. Requirement Analysis is an activity that commences after the successful completion of the knowledge transition phase. It typically includes two activities:
  - **Eliciting requirements:** The task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering
  - **Analyzing requirements:** Determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues
- ✓ It is important to identify all the stakeholders, take into account all their needs and ensure they understand the implications of the new systems.
- ✓ Testers need a solid understanding of the product or application so that they can devise better and more complete test plans, designs, procedures and cases.
- ✓ Test team involvement at these phases can eliminate ambiguities about the desired behavior of the application later in the project life cycle.

- ✓ Also, early involvement allows the test team to learn over time, which aspects of the application are the most critical to the end user and which are the highest risk elements.
- ✓ This knowledge enables testers to focus on the most important parts of the application first, avoiding over-testing of rarely used areas and under-testing of the more important ones.
- ✓ Testers involved from the beginning of the development life cycle, can help recognize omissions, discrepancies, ambiguities, and other problems that may affect the project requirements, testability, correctness and other qualities.

Knowledge Transition Concepts

Sources of Knowledge Transition

This section details the sources of knowledge transition. The major sources include client/business team and the onsite team deployed by the organization.

KT from Customer / Onsite: As the name suggests KT from Customer/Onsite is transfer of knowledge on the application/product from client or the organization's onsite team to the offshore project team. Since it is an initial session, the existing documents related to the system and business may be provided. The channel of transition varies from project to project.

Generally there are two approaches to this transition. A brief note on each of them is mentioned below.

- ✓ Big Bang Approach - Knowledge Transition for the first time during the start of an engagement – Application, Project / Program.

  Big Bang approach of KT is applicable to a start of a new project. The customer after entering into a business with an IT sector for the first time provides a high level KT on what the application is, to the project team. This helps the developers and testers to understand the objective of the project. It also provides a brief outline about the application and business.
- ✓ Incremental Approach – KT for other applications or incrementally during the future releases / enhancements.

  In general any application/product that is in use may undergo enhancements at later stage due to the changing customer needs. The transition during each enhancement phase includes additional features or modifications to the existing features to be made to an existing application/project. This transition, pertaining to other related applications that are provided incrementally during the future releases / enhancements is classified as incremental knowledge transition.

Internal KT: The knowledge transition that happens within the project team, without the involvement of the client is usually referred to as Internal KT. This can happen within a project when there are new resources, replacing someone or during ramp ups. This includes application/product/domain related sessions.

Methodology of Knowledge Transition

Various methods can be adopted to achieve knowledge transfer effectively. Some of these are outlined below. System information will also be gathered through informal discussions.

- ✓ Conference Calls - A formal presentation/lecture by an SME may be arranged to give a detail view of the application to understand efficiently about the system. The various channels of transfer includes, conference calls (Audio and Video), Web Ex sessions, Live Meeting etc.
- ✓ Query-Responses - While engaged in normal day-to-day work, queries raised by testers will have to be clarified by business/project team. Valuable system related information is shared through this process.
- ✓ Self Study – This involves the study of available documents that can be retrieved, viewed and interpreted by oneself without the aid of others.
- ✓ Shadowing Day to Day activities - A less-experienced performer pairs with a veteran performer on-job to facilitate knowledge transfer. Knowledge Transfer takes place on the actual job site with task accomplishment as a part of the process. It also involves learning skills and applying knowledge hands-on and on-job following or as a part of a defined structured learning process.

Knowledge Transition from Other Internal Systems – Some internal sources like the Community of Practice, Knowledge Repositories, and Content Management System etc also serve as a channel for knowledge transition.

Transition Planning

This is the first phase within Transition. The objective is to ensure readiness of the project team as well as the client to begin the KT – both from a resource and infrastructure perspective. Plans for on-boarding team members with requisite skills, reviews of existing/pre-developed knowledge transition plans, and fine tuning Knowledge Assessment/Knowledge Transition plans follow.

The Knowledge Transfer Plan is a complete, stand-alone document that provides a brief overview of the transition goals, any assumptions that the plan is based on, and any risks that have been identified that could severely limit your ability to complete the transition on schedule.

The following is a synopsis of the information contained in the plan.

- Scope (in-scope and out of scope)
- Transitioning Strategy
- Applications services taking into account each functional area with business line
- Transition Schedule
- KT Calendar
- Receivables and Deliverables during each phase
- Hardware Software requirements
- Escalation Process
- Status Reporting
- Progress Review/Team Meetings
- Team Meetings
- Milestones
- Resource Management Plan
- Training Process
- Risk and Mitigation Strategy

Knowledge Acquisition

During this phase, the testing team (primarily onsite resources) interacts with the client SMEs (subject matter expert) to gain understanding of the following:

- Business process flows
- Application functionality
- QA processes
- QA tools usage

Knowledge Acquisition happens through class room sessions, documentation sharing and self study and hands-on training on the application. Generally there is a daily plan for the class room and other knowledge sharing sessions scheduled between the SMEs and the incumbent testing team. Knowledge is typically acquired by the onsite team from the client SMEs and is passed on to the offshore team on daily basis. KT effort variation, schedule variation and effectiveness are continually monitored during this phase

Reverse Knowledge Transition

Reverse knowledge transition is simply the flow of knowledge in a direction opposite of the standard hierarchical top down approach. It is an important part of the transition phase as it helps in evaluating the effectiveness of knowledge transition. The knowledge acquired by the incumbent team during KT is captured in various application and process understanding documents.

The knowledge level of the transition team is validated in the following ways:

- ✓ Understanding documents – Reviewed by SMEs for completeness and clarity
- ✓ Application Walkthrough – Transition team walks through the application navigation to the client SMEs
- ✓ Feedback from SMEs– The knowledge level of the transition team is rated on several parameters by the client SMEs and the overall score needs to be above the predetermined threshold for the KT to be signed off

Fundamentals of Test Strategizing and Test Planning

Introduction:

About Module: This module will provide an introduction to Test Planning and Test Strategizing.

Module Objectives: After completing this Session, you will be able to:

- ✓ Explain Test Strategizing in Testing cycle
- ✓ Explain why Test Strategizing is required
- ✓ List the key factors to understand before Test Strategizing
- ✓ List the components of Test Strategy
    - Objectives & Critical success factors
    - Scope of testing
    - Testing Approach
    - Types and levels of testing
    - Test Data strategy
    - Test Milestones
    - Test Deliverables
    - Test criteria - Entry/exit criteria
    - Defect Management methodology
    - Roles and Responsibilities
    - Assumptions and Dependencies
    - Risks and Issues Identification and Management
    - Managing exceptional situations - Suspension and Resumption criteria
- ✓ Explain the need for Test Planning
- ✓ List the components of Test planning
    - Description of the system/project
    - Understanding of the Test Requirements
    - Testing Scope

  - Number of iterations for the types and levels of testing
  - Testing Schedule
  - Run plan creation
  - Resource planning – hardware, software, staffing
  - Roles & responsibility
  - Assumptions and Dependencies
  - Risks – Mitigation and contingency plan
  - Metrics and statistics
- ✓ List the benefits of Test Planning

Session 1: Test Strategizing

Test Strategizing in Testing Cycle

Software Projects are of different nature having business requirements focusing on one application, multiple applications, database validation, xml validation etc.

All Projects cannot follow the same approach for testing. Hence it is necessary to decide the appropriate methodology to run the project.

The Testing Strategy defines the overall approach to testing and describes how the testing process will ensure that the solution has the appropriate level of quality and reliability.

Why is Test Strategizing Required?

Test Strategy is required for almost all projects to carry over the testing in a defined and structured approach. The significance of the Testing Strategy is to define the overall context for the entire testing process. The process is different depending on the specific characteristics of the project. In many aspects, this is the most important part of the testing process, since all future testing decisions will be made within the context of the strategy.

Key Factors to Understand before Test Strategizing

- ✓ Business needs
  - The business need for the evolution of the application /system needs to be understood in detail before we define the testing process for it
- ✓ Internal / external interfaces involved
  - The internal / external interfaces of the application and the nature of data flow across these need to be taken into consideration while strategizing for testing
- ✓ Key stake holders of the project and their individual objectives from the project
- ✓ Primary end goal of the system / need for the application

- ✓ Nature of the application – Architecture, special skills required to use and test
  - ▪ The testing strategy will change depending on the architecture of the application. The skill set required for the testers to test certain applications would be unique
- ✓ Development methodology adopted for the application
- ✓ Implementation methodology of the application
- ✓ Infrastructural availability to host the application

Components of Test Strategy

Major components of Test Strategy are:

- ✓ Objectives & Critical success factors
- ✓ Scope of testing
- ✓ Testing Approach
- ✓ Types and levels of testing
- ✓ Test Data strategy
- ✓ Test Milestones
- ✓ Test Deliverables
- ✓ Test criteria - Entry/exit criteria
- ✓ Defect Management methodology
- ✓ Roles and Responsibilities
- ✓ Assumptions and Dependencies
- ✓ Risks and Issues Identification and Management
- ✓ Managing exceptional situations - Suspension and Resumption criteria

Test Strategy and its Components – An Example

In order to explain the Test Strategy and its components, let us consider the below set of requirements of Leave Management System

- ✓ For availing compensatory off, the employee should have worked for a minimum of 6 hours on a holiday, as per the working hours logged in the timesheet system
- ✓ Sick leave cannot be combined with any other type of leave other than loss of pay. Vacation leave can be combined with compensatory off.
- ✓ All types of leave can be approved or rejected by the manager.
- ✓ User can cancel or modify the applied leave before approval by the manager.
- ✓ If employee has availed loss of pay during a month, the salary of the employee for that month will be calculated based on the number of days the employee was on loss of pay.
- ✓ User is not allowed to avail any kind of leave during the serving of notice period.

System requirements:

- ✓ There are 25000 employees in the organization currently and the growth rate is expected to be at 25% over the next two years.
- ✓ System should be able to perform at the peak load of 2000 users with 250 concurrent users
- ✓ System should be accessible to all employees over the company intranet
- ✓ System should be accessible to all employees over internet after sufficient authentication
- ✓ System should be accessible in browsers IE 6 and above, Netscape 7 and above.
- ✓ System should be enabled for multilingual usage - English, German and French
- ✓ Leave accrual should be computed at the end of every month during off business hours
- ✓ Information to payroll system should be processed and ported on the last but one working day of every month
- ✓ Employee information from employee master database can be obtained asynchronously through MSMQ messages
- ✓ Working hours information from timesheet system can be obtained real time from replicated DB.

Below is the test strategy created for the above set of requirements? The example presented here is not to get into the details of all the components of a test strategy document, but to highlight the essential aspects of a test strategy.

Scope: Would include the applications/modules at high level, including the types of testing (manual, automation, performance) and levels of testing (system, system integration) as follows.

- ✓ **System / functional testing of:**
  - ▪ All screens of Leave application and approval
  - ▪ Leave accrual batch process
  - ▪ Reports
- ✓ **System integration testing**
  - ▪ Interfaces to Payroll system and Timesheet system
  - ▪ MSMQ message validation of employee database interface

The above two would address the functional requirements of the system

- ✓ Test automation
  - o Automation of critical test cases of leave application & approval and leave accrual process

  - ○ Test automation is identified to ensure faster, error free verification in the following situations:
    - ▪ Multiple rounds of regression testing will be carried out – after system testing, system integration testing and during UAT.
    - ▪ There are future requirements identified for the system which would result in many enhancements for the system.
    - ▪ For browser compatibility testing, the same set of test cases would be executed for the various browsers in scope
  - ○ The system will be used concurrently by 250 users for which stress testing should be performed
- ✓ Performance testing
  - ○ Load testing and stress testing of leave application
  - ○ Load testing of leave accrual process
    - ▪ The projected no. of users is 50000 (25000 + 50% growth for 2 years), so the system should be load tested for appropriate volume.
    - ▪ The system will be used concurrently by 250 users for which stress testing should be performed
- ✓ Browser compatibility testing
  - ○ For browsers IE 6.0, 7.0, Netscape 7.0, 8.0 and 9.0

Out of scope: This would include items which will not be tested by the testing team, either due to other stakeholders taking responsibility for it or due to request from customer during estimation. Mentioning out of scope aids the project managers to ensure that this work, if necessary for the project, is taken up by other teams.

- ✓ Usability testing of web interface
- ✓ Multilingual testing – Testing of the application for German and French

Test Approach: This would depict the overall approach and the flow of the various types of testing that will be conducted.

This can include the following aspects for the described leave management system:

System Test Approach:

- ✓ Approved use case documents and UI prototypes would be used as the basis for writing system test cases. The test cases would be signed off by the business analysts.
- ✓ As per development plan, application would be delivered in 2 phases: Phase 1 – Leave application and Approval workflow, Phase 2 – Leave accrual and other interfaces
- ✓ System testing would be carried out in phase 1 and 2. System integration testing would be carried out in phase 2, which includes interfaces with external systems

Regression Test Approach:

- ✓ Test cases for regression testing would be identified based on the following aspects:
  - ○ Business critical test cases – identified by business analysts

  - Functionality of phase 1 which will be affected by phase 2

- ✓ In addition, regression testing for related functionality in case of defect fixes will also be carried out
- ✓ The porting routines and MSMQ messaging to connect to the external applications should be available in the system integration testing environment
- ✓ One final round of regression testing would be carried out after system integration testing to avoid any defect leakage
- ✓ Automation scripts for identified critical test cases would be created after round 1 of system testing and these would be used during regression testing
- ✓ Automation scripts of the existing leave accrual process will be analyzed and changes made to fit the current requirements

Non functional Testing Approach:

- ✓ Performance testing will be carried out only after the completion of system testing
- ✓ Response time on the web interface for leave application and approval will be checked with a concurrent load of 50 users (normal scenario) and 250 users (peak usage scenario). Response time target would be < 3 seconds for normal usage and < 5 seconds for peak usage
- ✓ All the system and system integration test cases will be carried out using IE 6.0. The identified critical test cases for regression will be executed with each of the following browsers: IE7.0, NS 7.0. NS 8.0 and NS 9.0

Browser Compatibility Testing Approach:

- ✓ Browser compatibility testing will be carried out in parallel to round 2 of system testing since identifying defects on compatibility should be completed before regression testing.

Test approach as given above, ensures the following:

- ✓ Stakeholders able to validate the approach and correct it if they foresee an issue in the approach
  - System integration testing will be carried out during phase 2 on the SIT environment with interfaces - If the environment management team cannot be ready with the interfaces deployed on the SIT environment for phase 2, they can suggest testing of non-interface related functionality of phase 2 in the ST environment instead of SIT environment
  - Only regression test cases will be executed in all versions of netscape navigator – Development team can suggest additional UI based functionality which they know might behave differently in NS browsers
- ✓ All stakeholders know what is expected from them and the associated dependency

  - Sign off of test cases by business analysts & identification of business critical functionality by business analysts – ensures BAs are aware of the accountability.

- All stakeholders are aware of the risks in the approach and is able to take suitable mitigation at their end
  - Performance testing will be carried out only after system testing – Development team understands that if a serious error is uncovered in performance during the second cycle the project may not be delivered on time. So they ensure adequate architecture and design reviews are done from performance perspective.
- Testing team has clarity on the sequence of the testing activities to create the test plan
  - Testing team understands the sequence of system, system integration, regression, performance, compatibility testing and what needs to be tested in each of these. Test plan schedule and other dependencies like environment can be derived based on this.

Milestones

- Important stages during the delivery of the project are decided and documented, such as Requirement analysis completion, test plan sign off, test case sign off, test completion etc

Test data

- Types of test data, how and who will identify and create the test data
  - For leave management system, test data requirements would be identified as follows:
  - Master data for employee will be loaded from the employee database before test design. This will not be refreshed during the various cycles of testing
  - Transaction data on leave details of employees will be ported from the existing leave management system. This data will be refreshed for every cycle of testing
  - For load testing, data of ~100000 leave transactions (assuming 2 per employee) should be provided to the testing team on the test environment

Environment setup:

- Would contain details on which and how many test environments would be used, what access level is required for testers in each of these environments, what type of test data is required and who will provide it:
  - Environment E1 will be used for system testing and also for developing automation scripts
  - Environment E2 will be used for system integration testing since porting routines and access to payroll system and timesheet system are available in this environment. This environment will also be used for regression testing

  - System integration test cases for interface with employee database will be executed at onsite since interaction with systems support for MSMQ is essential to test this
  - Leave application and leave accrual data from the existing leave management system in production to be provided in the identified system test environment

Test Deliverables:

- Deliverables during each phase of testing such as
  - Test plan, test cases, test data, test summary report etc. for manual testing
  - Feasibility document, proof of concept, test scripts etc. for test automation

Test criteria:

- Each phase of testing (test design, test execution etc.), level of testing (system, system integration etc.) and type of testing (performance, compatibility etc.) requires certain criteria as prerequisite and to determine completion by the various stake holders. The test criteria remove the ambiguity in terms of when a testing activity can start and when it is considered complete.
  - Entry criteria for system testing: All unit test cases and component integration test cases should be executed and logs made available
  - Test completion or exit criteria for system testing: No severity 1 or 2 defects which are open and resolution in place for the rest
  - Suspension criteria: Encountering defects which prevents further testing
  - Resumption criteria: When new version is released after defect fixes and assurance provided
- Critical success factors
  - These are critical factors that determine the success of the project
- Defect Management methodology
  - Definition of a defect
  - Definition of severity and priority of a defect
  - Defect life cycle – definition of status and workflow
- Roles and responsibility:

  Successful testing of application requires many activities to be performed by various stake holders. Stake holders can be internal to testing such as the test manager, test lead etc. There are many external stakeholders who are involved such as:
- Customer IT organization stakeholders:
  - Infrastructure support who satisfy the environment related requirements
  - Database administrators who facilitate data loading in the various environments

- Release management team who deploys the build to be tested in the various environments

- ✓ Customer business stakeholders:
  - Customer SME: Facilitate requirement understanding, clarify business requirement related queries, support UAT
  - Customer business user: Carries out UAT
- ✓ Development organization stakeholders:
  - Development manager: coordinating build deployment, test execution schedule, defect fix schedule etc.
  - Developer: Defect triaging, defect fixing etc.

The roles played by various stakeholders and the activities they would be responsible for is determined.

For instance for leave management system, following roles would be identified:

- ✓ Testing related roles – test analyst, test lead, test manager
- ✓ Development lead – involved in clarification of design, participation in defect triage meetings, root cause analysis for defects etc.
- ✓ Business analyst – involved in reviewing and signing off test cases, identifying test cases for regression testing, co-ordinate for UAT etc.
- ✓ Business users – creating UAT test cases and carrying out UAT
- ✓ Systems support engineer – involved in helping out the testing team with test environment setup.
- ✓ Database administrator – involved in providing test data from the existing production systems, providing adequate data for volume etc.
- ✓ Program manager – involved in resolving issues in timely manner for testing team to proceed with their testing activities, co-ordinate with other development and business stakeholders in testing related activities
- ✓ Other factors influencing Testing
  - Assumptions and Dependencies considered for testing
  - Risks – Identification, Mitigation and contingency plan
  - Handling exceptional situations - Suspension and Resumption criteria

Test Planning

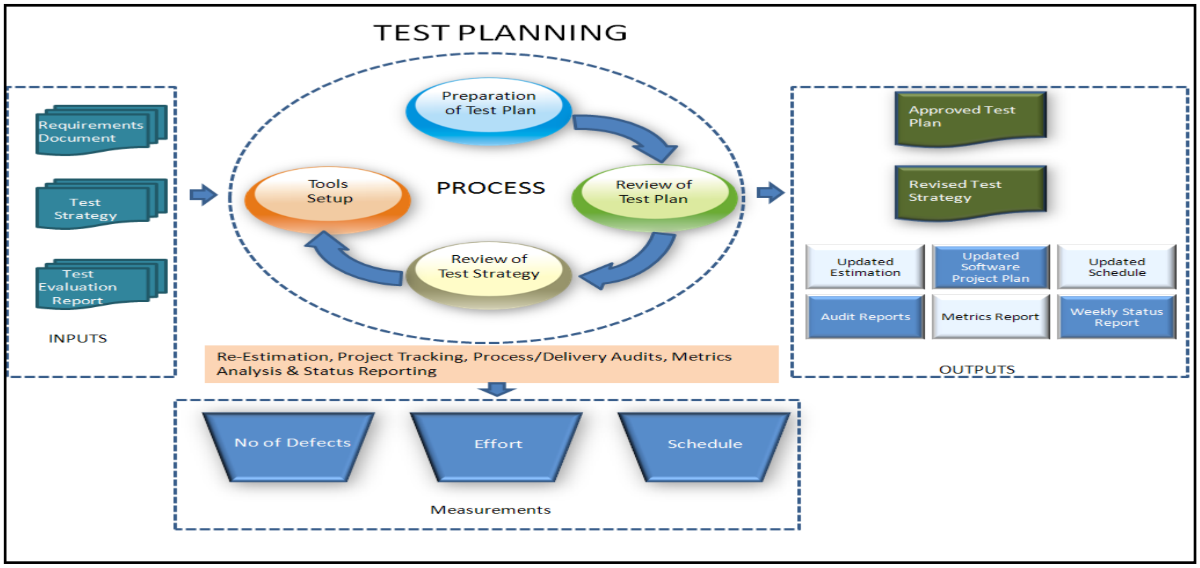
Learning Objectives

- ✓ List the need for Test Planning
- ✓ State the difference between Test strategizing and Test planning
- ✓ List the benefits of Test Planning
- ✓ List the pre-requisites to Test Planning
- ✓ List the components of Test planning
  - o Description of the system/project
  - o Understanding of the Test Requirements
  - o Testing Scope
  - o Number of iterations for the types and levels of testing
  - o Testing Schedule
  - o Run plan creation
  - o Resource planning – hardware, software, staffing
  - o Roles & responsibility
  - o Assumptions and Dependencies
  - o Risks – Mitigation and contingency plan
  - o Communication plan
  - o Metrics and statistics

Need for Test Planning

- ✓ The quality of the testing effort is directly proportional to the quality of the test planning.
- ✓ Test plan contains the critical information used to execute the project for the defined strategy
- ✓ A proper plan on how to execute the test approach determined helps in avoiding confusions and surprises during the execution of the project
- ✓ Once a plan is defined and agreed upon by the stake holders, the project team can refer to this any time during the course of the project

How Test Planning Works?



Components of a Test Plan

- ✓ Major components of a Test Plan include:
- ✓ Description of the system/project
- ✓ Understanding of the Test Requirements
- ✓ Testing Scope
- ✓ Number of iterations for the types and levels of testing
- ✓ Testing Schedule
- ✓ Run plan creation
- ✓ Resource planning – hardware, software, staffing
- ✓ Roles & responsibility
- ✓ Assumptions and Dependencies
- ✓ Risks – Mitigation and contingency plan
- ✓ Communication plan
- ✓ Metrics and statistics

The essential elements of test plan for the leave management system are:

Description of the system/project:

Includes description of the application, functionalities and interfaces

Scope:

Would include the applications/modules along with features at high level, pertaining to the type of testing the plan addresses, as follows:

- Leave application:
  - Login for employees in intranet
  - Login for employees in internet
  - Applying for all types of leave along with necessary verification
  - Modifying leave application before approval
  - Cancelling leave application before approval
  - Leave application submission verifications on leave balance and eligibility depending on the type of leave
  - Viewing of the available leave balance by employee
- Leave approval
  - Approval by manager for submitted leave
  - Rejection by manager for submitted leave
  - Modification of rejected leave by employee
- Leave accrual batch process:
  - Accrual for applicable types of leave
  - Accrual for employees who have joined during the years
  - Leave balance calculation considering accrual and leave transaction records
  - Audit and error notifications
- Interface to timesheet system
  - Availability of working hour information for compensatory leave
  - Eligibility reconciliation based on compensatory leave availed
- Interface to payroll – batch process
  - Porting of loss of pay information to payroll system
  - Audit and error notifications
- Interface to employee database
  - Validation of message received from employee database
  - Date of joining
  - Resignation notice period
- Reports
  - Leave balance report and leave transaction report for employee

- Batch report on leave transactions for finance

Test execution cycles: Plan for the number and scope of test execution cycles

For LMS this plan would appear as follows:

- ✓ **Cycles of testing will be carried out for system testing:**
  - Smoke testing
  - Cycle 1: Execution of all test cases
  - Cycle 2: Defect re-testing for cycle 1 defects and execution of all test cases
  - Cycle 3: Execution of regression testing based on all defect fixes
- ✓ **Cycles of testing will be carried out for system integration testing:**
  - Smoke testing
  - Cycle 1: Execution of all test cases
  - Cycle 2: Defect re-testing for cycle 1 defects and execution of all test cases
  - Cycle 3: Execution of regression testing based on all defect fixes
- ✓ **Cycles of regression testing:**
  - Smoke testing
  - Cycle 1: Execution of all regression test cases
  - Cycle 2: Will be carried out if there were any defects identified in Cycle 1, in which case all regression test cases will be executed again
- ✓ **Schedule:**
  - The overall effort for testing project is estimated using standard estimation techniques like Test Case Point (TCP) estimation.
  - Based on the effort estimation and resource planning, schedule is planned.
  - Detailed schedule of activities, start date and end date. This would also consider dependencies

For Example:

Activity	Start date	End date
Test case preparation	4/2/2011	4/20/2011
Test case review and signoff	4/21/2011	4/25/2011
Test environment setup	4/21/2011	4/25/2011
Cycle 1- test execution	4/26/2011	5/10/2011

- ✓ **Resource Planning:**

  Hardware and software: Includes OS, hardware, applications, testing tools and any connectivity requirements.

  For Example:

Activity	Start date	End date
Test case preparation	4/2/2011	4/20/2011
Test case review and signoff	4/21/2011	4/25/2011
Test environment setup	4/21/2011	4/25/2011
Cycle 1- test execution	4/26/2011	5/10/2011

Staffing:

- ✓ Identification of person performing the identified roles in test strategy

Roles and Responsibilities:

- ✓ The roles of the various stakeholders would be defined in test strategy. In test plan

Assumptions and Dependencies:

- ✓ Assumptions made and dependencies identified during the test planning.

Risks – Identification, Mitigation and Contingency Plan

- ✓ Risks related to test planning such as availability of the right environment, availability of the build with the required functionality etc. along with mitigation and contingency plans.

Communication Plan

- ✓ Plan that determines how, when and what to report to various stake holders of the project during the course of testing

Metrics and Statistics

- ✓ Factors to be measured (measures and metrics) and reported to the stake holders during the course of testing, along with the frequency of reporting

Benefits of Test Planning

- ✓ Helps in better estimation of the project in terms of efforts, size and schedule
- ✓ Clearly defines the items to be tested
- ✓ Estimate the risks involved in the project (technical and procedural) and identify mitigation steps if possible
- ✓ Identify test deliverables and organize the testing efforts
- ✓ Identify the roles and responsibilities of each resource involved in the project to ensure successful testing.

Requirements Gathering

On the successful completion of reverse knowledge transition, the project team starts gathering requirements. Requirements are a description of how a system should behave or a description of system properties or attributes. Requirement gathering involves the task of communicating with customers and users to determine what their requirements are. Testers should interact closely with multiple work-groups, often with conflicting goals, to arrive at a bona fide requirements list. Some of the common issues faced during requirements gathering are mentioned below.

- Ambiguous understanding of business processes
- Inconsistency within a single business process by multiple users
- Insufficient input from stakeholders
- Conflicting stakeholder interests
- Changes in requirements after project has begun

Currently, organizations have started to use tools that are better equipped to handle the complex and multilayered process of requirements gathering. Some of the tools used are:

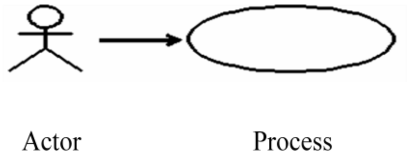
- Prototypes
- Use cases
- Data flow diagrams

Prototypes:

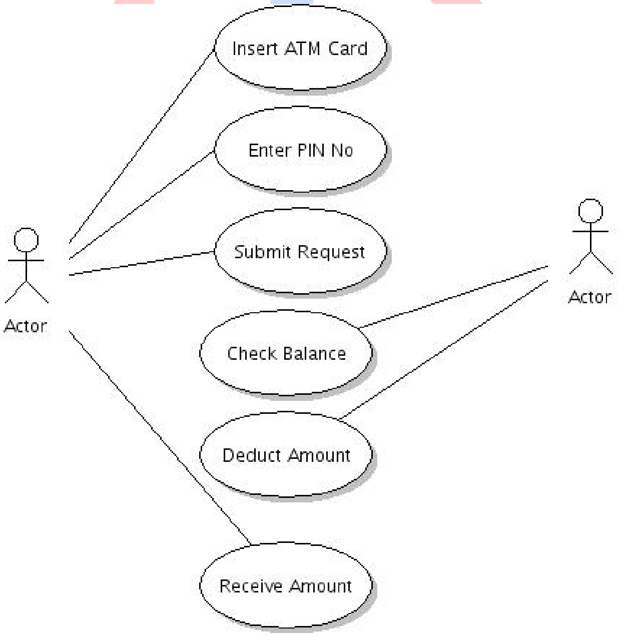
Prototypes are "Mockups" of an application. They help users get an idea of what the system will look like, and make it easier to make design decisions without waiting for the system to be built. Often the end users may not be able to provide a complete set of application objectives, detailed input, processing, or output requirements in the initial stage. After the user evaluation, another prototype will be built based on feedback from users, and again the cycle returns to customer evaluation. The cycle starts by listening to the user, followed by building or revising a mock-up, and letting the user test the mock-up, then back. Major improvements in communication between users and developers were often seen with the introduction of prototypes. Early views of applications lead to fewer changes later and hence reduced overall costs considerably.

Use Cases:

A use case is a set of scenarios that describes an interaction between a user and a system. Use cases capture who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals. A complete set of use cases specifies all the different ways to use the system, and therefore defines all behavior required of the system, bounding the scope of the system.

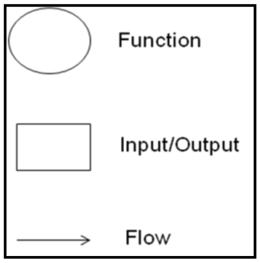


Example:

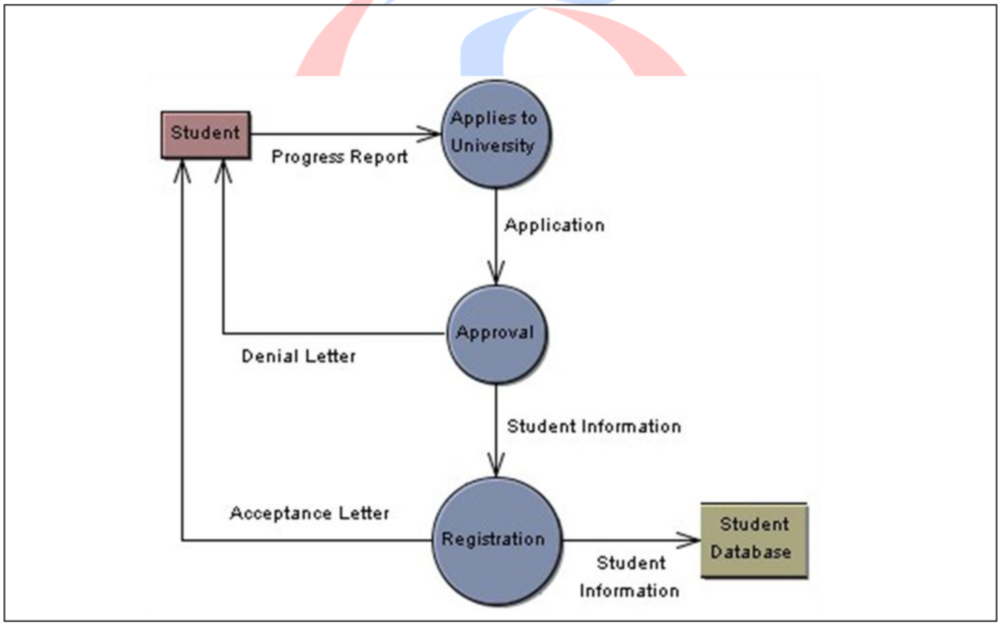


Data flow Diagrams:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing. It shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



Example:



Requirement Analysis

Learning Objectives

After completing this chapter, you will be able to:

- ✓ Explain the types of requirements
- ✓ Analyze the requirement specifications
- ✓ Prioritize the requirements

- ✓ Understand clarification management
- ✓ Analyze requirement defects
- ✓ Understand requirements change management
- ✓ Understand the concept of requirements coverage
- ✓ Understand the difference between Application Development and Maintenance in this phase

Requirement Analysis and Verification

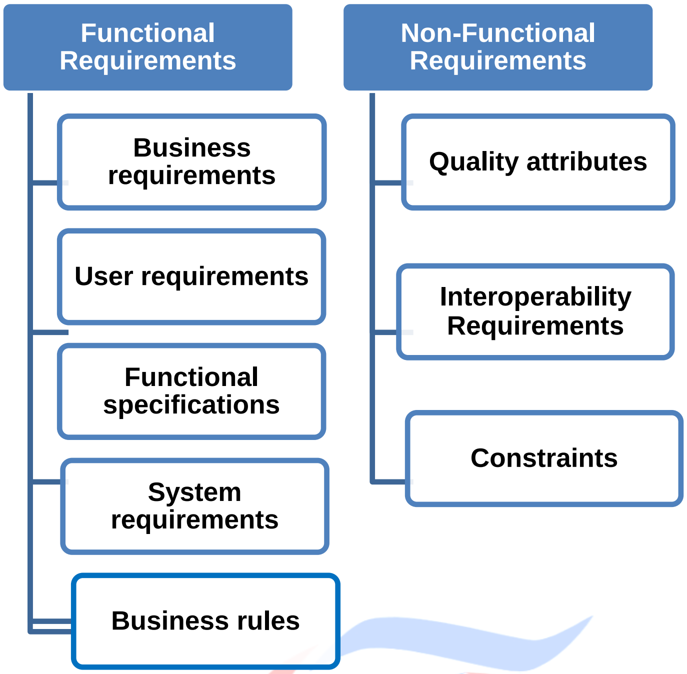
Requirements Analysis is the process of understanding the customer needs and expectations from a proposed system or application and is a well defined stage in the Software Development Life Cycle model. Requirements are a description of how a system should behave or a description of system properties or attributes. It can alternatively be a statement of 'what' an application is expected to do. Given the multiple levels of interaction between users, business processes and devices in global corporations today, there are simultaneous and complex requirements from a single application, from various levels within an organization and outside as well. The Requirements Analysis Process covers the complex task of eliciting and documenting the requirements of all these users, modeling and analyzing these requirements and documenting them as a basis for system design. A streamlined process is a prerequisite to successful projects that align with the client's business goals and meet the project's requirement specifications.

Requirement Types and Documents

A requirement can be defined as a high level statement of the system. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order to have value and utility to a user. It can also be expressed as below:

- Statement of the business need from the client to the developer/tester
- A specification of what has to be built
- Requirements can state either the business need or how the system should behave

Mainly based on the functionality of the system it is classified as Functional and Non-functional requirements.



Functional Requirements:

In general, this describes the functions that the system needs to execute, statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. It depends on the type of software, expected users and the type of system where the software is used. It can be further categorized as Business Requirements, User Requirements, Functional Specifications, System Requirements and Business Rules.

- ✓ **Business Requirements –** Business requirements represent high-level objectives of the organization or customer who requests the system. They describe why the organization is implementing the system—the objectives the organization hopes to achieve, the business objectives, scope of the project, business constraints and current business process. They are usually expressed in terms of broad outcomes the business requires, rather than specific functions the system may perform.

  Example – ATM should allow the withdrawal of given amount from the account with a cap on maximum amount and number of withdrawals.

- ✓ **User Requirements –** User requirements describe the user goals, tasks or activities that the users must be able to perform with the product and the flow of the system. It conveys how the system should interact with the end user or to another system to achieve a specific business goal. Common users are client managers, client QA team and system end users. They are usually represented in the form of tables and diagrams.

Example – The system shall complete a standard withdrawal from a personal account, from login to cash, in less than two minutes for a first-time user.

- ✓ **Functional Specifications –** These specify what has to be done by identifying the necessary task, action or activity that must be accomplished. They describe the existing process, new process, use cases, process flow and reporting requirements. They are also referred to as behavioral requirements.

  Example – The system shall provide users with the ability to select whether or not to produce a hardcopy transaction receipt before completing a transaction.

- ✓ **System Requirements –** System requirements describe the top-level requirements for a product that contains multiple subsystems. It is a structured document having detailed description about the system. It is used by the system end users, software developers and system architects.

  Example:

a. The ATM shall communicate to the Bank via the Internet.

b. The ATM shall issue a printed receipt to the customer at the end of a successful session.

- ✓ **Business Rules –** These are constraints to be considered on the aspect of a business while developing a product. It includes corporate policies, industry standards and computational algorithms. It typically has existence on the outside of the boundaries of any specific software system. It requires that specific functionality be implemented to ensure that the system enforces or complies with those rules. It is also used to control internal system processing based on specific combinations of data values, system states, conditions, or other user-defined criteria.

Example:

a. Limit is $1000 a day (24 hour window)
b. No more than three withdrawals per day (24 hour window)

Non-Functional Requirements:

In general, this refers to the constraints on the services offered by the system such as time constraints, constraints on the development process, standards, etc. It defines the system properties and constraints e.g. reliability, response time, storage requirements, scalability, usability, security. It can be further categorized as Quality Attributes, organizational, external requirements.

- **Quality Attributes –** These describe the system's characteristics in various dimensions that are important either to users or to developers and maintainers. Quality attributes of a system include availability, performance, usability, portability, integrity, efficiency, robustness, and many others. These characteristics are referred to as quality factors or quality of service requirements

  Example – All displays shall be in white 14 point Arial text on black background.
  It must be able to perform in adverse conditions like high/low temperature etc

- **Interoperability Requirements** - These specify the interfaces between the system application and other applications, interface between the system application and other hardware or devices like printers, bar code readers, interface to human users, communication interfaces to exchange information

  Example:

  a. **User Interfaces:** The customer user interface should be intuitive, such that 99.9% of all new ATM users are able to complete their banking transactions without any assistance
  b. **Hardware Interfaces:** The hardware should have following specifications:
     Ability to read the ATM card
     Ability to count the currency notes

Touch screen for convenience

Keypad (in case touchpad fails)

Continuous power supply

Ability to connect to bank's network

Ability to take input from user and validate user

- **Constraints –** These are requirements prescribing stipulations or limitations on how the system can be built. It usually refers to design and implementation constraints, which are restrictions imposed on the choices available to the developer for some legitimate reason. Some of the common constraints on the application are reliability, performance, security, usability and safety. Constraints may be user constraint or system constraint.

  Example - The cabin door will always be locked, which will open only when user swipes his/her ATM card in the slot and is validated as genuine.

Analysis of Requirement Specifications

Testers use requirement documents and interactions with users/clients to develop the initial versions of the system and acceptance tests based on functional requirements, quality requirements, and the specification of the system behavior. A requirement can be considered testable if it is possible to design a procedure in which the functionality being tested can be executed, the expected output is known, and the output can be programmatically or visually verified.

Following is the checklist that can be used by testers during the requirements phase to verify the quality of requirements. Using this checklist is a first step towards trapping requirements -related defects as early as possible, so they don't propagate to subsequent phases, where they would be more difficult and expensive to find and correct. All stakeholders responsible for requirements should verify that requirements possess the following attributes:

- Correctness (Do the requirements reflect the user's needs? Are they stated without error?);

- Completeness (Have all functional and quality requirements described in the problem statement been included?);
- Consistency (Do any requirements contradict with each other?);
- Clarity (It is very important to identify and clarify any ambiguous requirements);
- Relevance (Is the requirement pertinent to the problem area? Requirements should not be superfluous);
- Redundancy (A requirement may be repeated; if it is a duplicate it should be combined with an equivalent one);
- Feasibility (Can the requirements be implemented given the conditions under which the project will progress?)
- Testability (Can each requirement be covered successfully with one or more test cases? Can tests determine if the requirement has been satisfied?);

Prioritization of Requirements

Prioritization allows everyone to understand the relative value to stakeholders of the requirement. When customer expectations are high, timelines are short, and resources are limited, you want to make sure the product contains the most essential functions. Prioritization helps the project team to focus on critical business areas, which need to be tested earlier in the testing cycle. It helps testers to detect defects early in the testing life cycle. It helps conflicts, plan for staged deliveries, and make the necessary trade-off decisions.

Clarification Management

A requirement can be tested and clarified by asking the stakeholders detailed questions. More than just understanding the "inputs and outputs" of the software, testers need deeper knowledge that can come only from understanding the thought process used during the specification of product or application functionality. Such understanding not only increases the quality and depth of the test procedures developed, but also allows testers to provide feedback regarding the requirements. Clarifications that were posted should be tracked and can be used as a knowledge repository for the project.

Testers should list all the clarifications with an "Open" status and should assign to the corresponding stakeholder. The stakeholder should revert back with answers to the clarifications with status as “Closed” or if waiting for more information should change the status as “In Progress”. Tester, if not clarified with the answer should provide additional information and change the status to “Re Open”.

Requirement Defects

The beginning of the software life cycle is critical for ensuring high quality in the software being developed. Defects injected in the early phases can persist and be very difficult to remove in later phases. Since many requirements documents are written using a natural language representation, there are very often occurrences of ambiguous, contradictory, unclear, redundant and imprecise requirements. Some of the common requirements/specification defects are:

- **Functional Description Defects** – The overall description of what the product does and how it should behave is incomplete and or ambiguous.

- **Feature Defects** – Features refer to functional aspects of the software that map to functional requirements as described by the users and clients. Features also map to quality requirements such as performance and reliability. Feature defects are due to feature descriptions that are missing, incorrect, incomplete or superfluous.

- **Feature Interaction Defects** – These are due to an incorrect description of how the features should interact. For example, suppose one feature of a software system supports adding a new customer to a customer database. This feature interacts with another feature that categorizes the new customer. The classification feature impacts on where the storage algorithm places the new customer in the database, and also affects another feature that periodically supports sending advertising information to customers in a specific category. When testing, we certainly want to focus on the interactions between these features.

- **Interface Description Defects** – These are defects that occur in the description of how the target software is to interface with external software, hardware and users.

Requirements Change Management

Requirements generally change with time. Once defined and approved, requirements should fall under change control. For many projects, requirements are altered before the system is complete. This is partly due to the complexity of computer software and the fact that users don't know what they want before they see it. This characteristic of requirements has led to requirements management studies and practices. Process and workflow, authorization and assignment controls who, what, when and where change happens, and communicates the impact of that change across the lifecycle to all involved.

Requirements management is the process of managing changing requirements during the requirements engineering process and through system development and release. Often requirements are incomplete and inconsistent and ofcourse new requirements emerge during the process as the business needs change and a better understanding of the system is developed. Different viewpoints, which also have to be considered as the project evolves, have different requirements and these are sometimes in conflict with those that were stated at the beginning of the project.

Regardless of the type of requirement you are working with, managing change to the set or any of the individual parts can and should be handled in a uniform fashion – one that automatically communicates the intent to change; controls the acceptance or approval of change; enables the analysis of the impact of the change; tracks the actual changes made and communicates the implemented change to all impacted or interested stakeholders.

Requirements Coverage

The coverage of requirements is a fundamental need throughout the software life cycle. Requirements coverage views focus on the localization of the requirements in the rest of the system. These views show if and where a certain requirement is covered in the system. This can be coverage in, for example, the system architecture, in the detailed design, or in the test cases. Be it during design, coding or testing, the ability to ensure that the software meets the expected requirements is something that every customer aspires for. As the software matures and goes into several iterations of enhancements and bug fixes, it becomes more and more daunting to ensure requirement coverage in the software. Some tools offer the capability of mapping requirements to the other project artifacts, such as, use cases, test cases and design documents. Some projects prefer to follow simple spreadsheet based traceability matrix of requirements to various other artifacts.

KT & Requirements Analysis Phase – Application Development Vs Maintenance

In this phase, for application development, the knowledge transition approach adopted is Big Bang. Other activities include Requirements Gathering, Analysis of all the requirement specifications and preparation of system understanding document to evaluate the application/business understanding. Similarly, for application maintenance projects, the incremental approach of knowledge transition is adopted. The activities include performing an impact analysis on the change requests received, analyzing the related requirements, and updating the existing knowledge base\Glossary

Requirement: A high level statement of the system

Stakeholder: Any group or individual who can affect or who is affected directly/indirectly by achievement of a firm's objectives

KT – Knowledge Transition

DFD – Data Flow diagram

RTM: Requirement Traceability Matrix

QC – Quality Center

Testability: Testability can be referred to as the degree to which the characteristics that provide for testing exist, and economically feasible tests can be devised for determining whether the developed software will satisfy the requirements.

Traceability: Defines a relationship between two or more requirements.

Test Case Design Process

1. **Identifying the Test conditions**
   - ✓ High Level Test Conditions -->Test Scenario's
   - ✓ Low Level Test Conditions -->Test Cases
2. **Why we need to create Test Scenario's and Test cases? (Or) What are the Benefits of creating the Test cases?**
   - ✓ If we document test cases effectively then it helps in "Reusability"
     - o While retesting the defects
     - o While doing Regression if any changes happens
   - ✓ Test cases are Reusable and Repeatable
   - ✓ Test Case is a document which clearly describes what to be tested and how to be tested.
     - o What is the purpose of the test?
     - o How to Test it?
     - o What is the expected outcome?
   - ✓ It saves Test Case execution Time
   - ✓ Test Case design is a verification activity which helps to judge quality of requirements at earlier stages of the project to prevent The defects.
   - ✓ It increases collaboration between dev/QA/BA team.

3. **Explain Test Case design process in the project?**

Step 1: We involve in KT process to understand project context, Modules, Business purpose and end user expectations

Step 2: We involve in Requirements Gathering, Requirement Analysis, Query discussions to understand the requirements which are allocated by Test Lead according work Allocation Tracker

Step 3: Once after getting resolution to all the Queries we start identifying Test Scenario's based on the Test Approach mentioned in the "Test Plan"

Test Scenario: A high level Test condition which explains what needs to be tested.

Ex: To Test Search Functionality

Note: Even Dev and BA also proposes Test scenario's in BRD document and FRS document

Test Scenario's provides an high level idea on what we have to test. In our project we are documenting Test Scenario's in "Scenario Selection sheet"

Scenario selection sheet components:

- Scenario ID
- Scenario Desc
- Associated Requirement
- Scenario Priority

Step 4: Send list of Test scenarios to BA and Dev review. Add/modify Test scenario's if they propose any changes. Get signoff on Test Scenario's.

Step 5: Now Breakdown Test scenario's into detailed Test cases

5.1 One Test scenario can be breakdown into N number of Test cases based on the complexity of the requirement

5.2 While writing the test cases cover both positive and Negative Test cases

5.3 Make sure each Test case is properly reviewed

5.4 Make sure we have created Test cases for all the requirements

5.5 We can write test cases in Test case template (Excl) or Test Management tools like HP QC/JIRA/MTM/OTM/IBM RTM/Rally

Test Case: A detailed description of what conditions need to be tested and how to be tested with step by step approach.

What are the important sections of Test case Template?

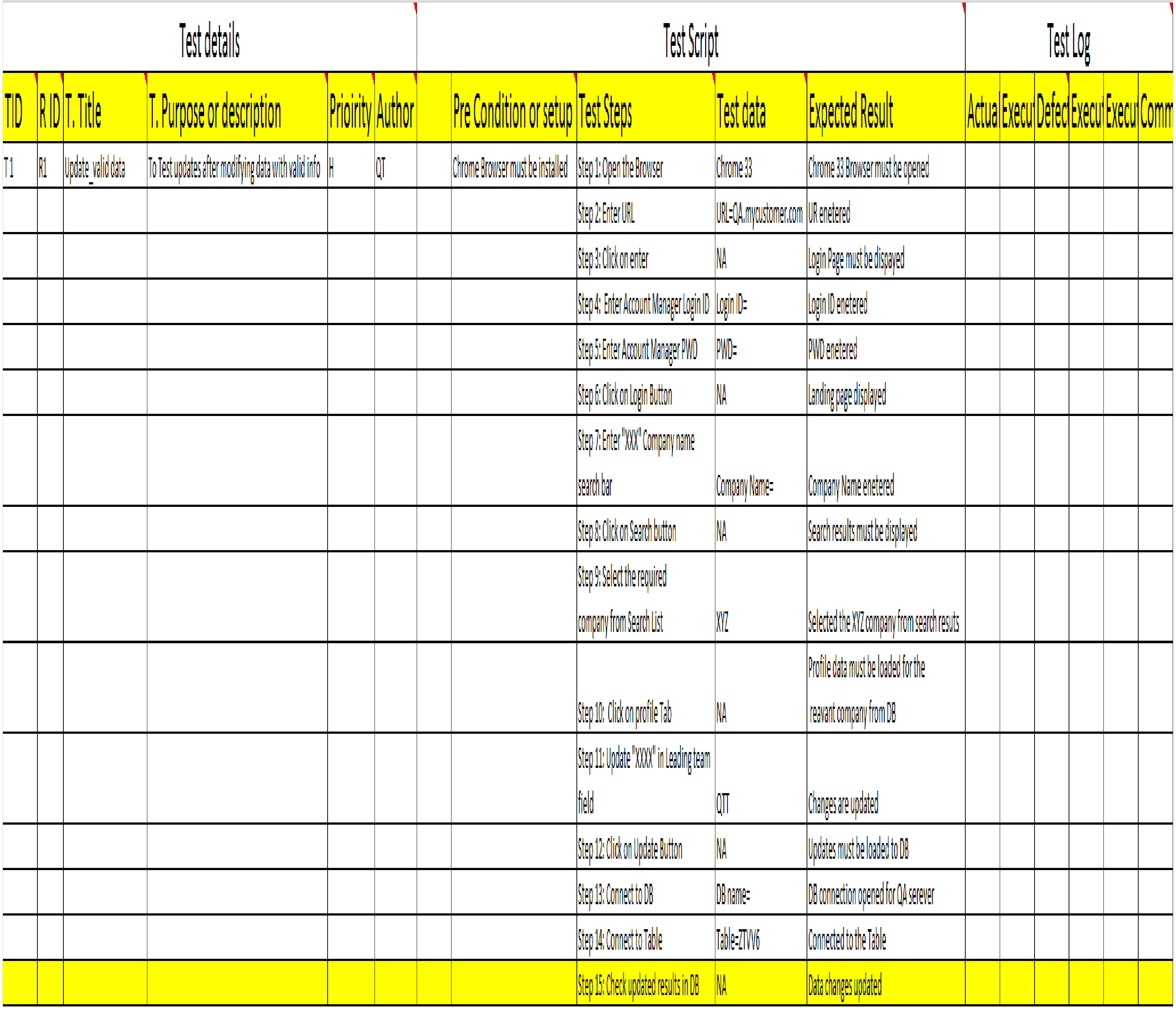
- T ID
- T Description
- Priority of the Test Case
- Pre-condition of the Test case
- Test Steps
- Input Test data
- Expected Result

Step 6: Make sure Test cases are properly reviewed and signed off by BA and Dev

Sample Test Scenario's:

S ID	S Desc	Module/Req	Screen Name	Priority
Scenario 1	To test updates in DB	Profile	Landing Page	H
Scenario 2	To test search functionality	Search	Landing Page	H
Scenario 3	To display of UI components in Landing Page	Profile	Landing Page	H
Scenario 4	To Test Company profile correctly loaded from DB	Profile	Landing Page	H
Scenario 5	To Test updates are correctly working for Account Manager	Profile	Landing Page	H
Scenario 6	To Test updates are correctly working for Marketing Manager	Profile	Landing Page	H
Scenario 7	To Test updates are correctly working for Technical Manager	Profile	Landing Page	H

Sample Test Case:



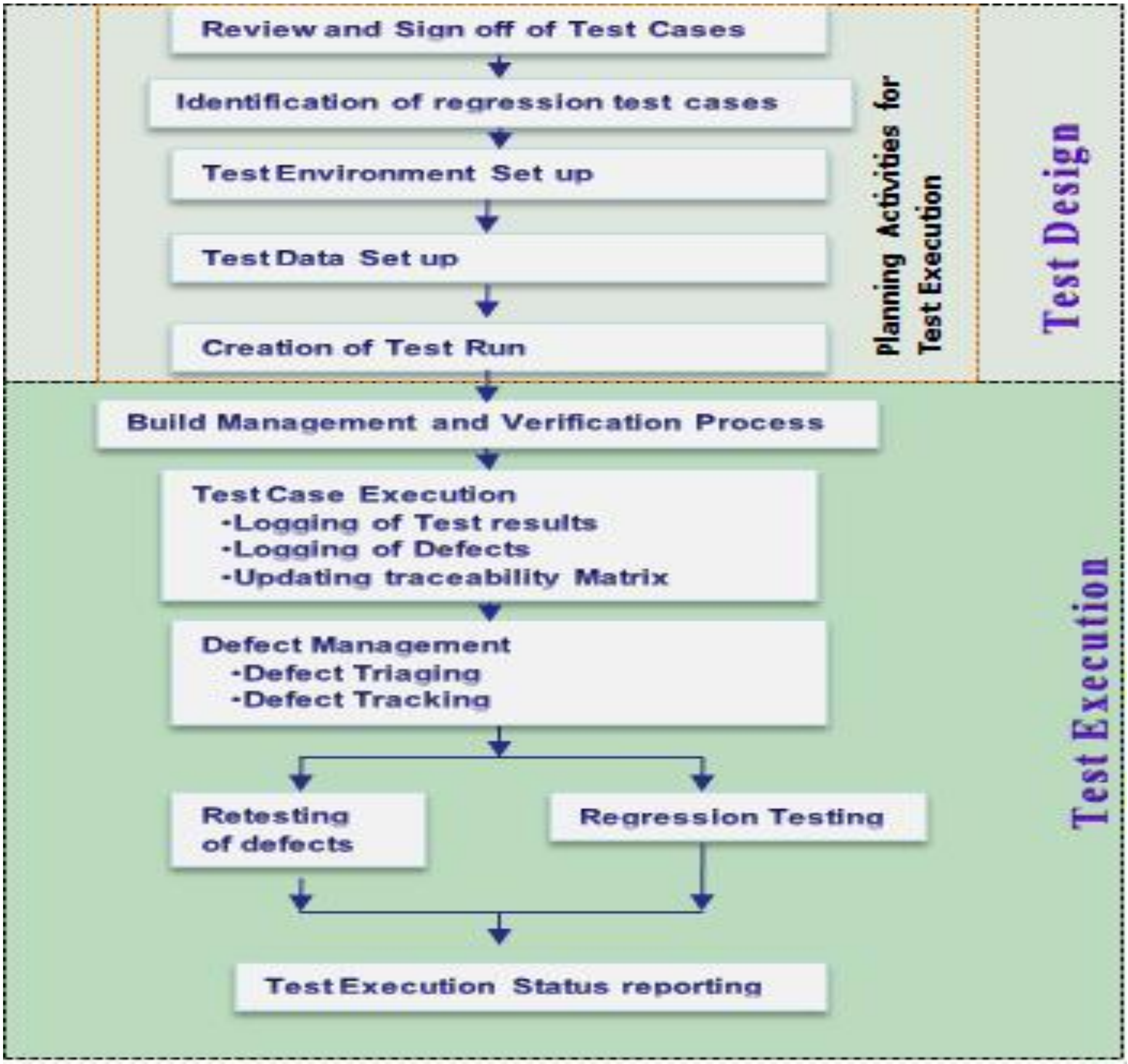
Test details						Test Script					Test Log					
TID	R ID	T. Title	T. Purpose or description	Priority	Author		Pre Condition or setup	Test Steps	Test data	Expected Result	Actual	Execu	Defect	Execu	Execu	Comm
T1	R1	Update_valid data	To Test updates after modifying data with valid info	H	QT		Chrome Browser must be installed	Step 1: Open the Browser	Chrome 33	Chrome 33 Browser must be opened						
								Step 2: Enter URL	URL=QA.mycustomer.com	UR enetered						
								Step 3: Click on enter	NA	Login Page must be dispayed						
								Step 4: Enter Account Manager Login ID	Login ID=	Login ID enetered						
								Step 5: Enter Account Manager PWD	PWD=	PWD enetered						
								Step 6: Click on Login Button	NA	Landing page displayed						
								Step 7: Enter "XXX" Company name search bar	Company Name=	Company Name enetered						
								Step 8: Click on Search button	NA	Search results must be displayed						
								Step 9: Select the required company from Search List	XYZ	Selected the XYZ company from search resuts						
								Step 10: Click on profile Tab	NA	Profile data must be loaded for the reavant company from DB						
								Step 11: Update "XXXX" in Leading team field	QTT	Changes are updated						
								Step 12: Click on Update Button	NA	Updates must be loaded to DB						
								Step 13: Connect to DB	DB name=	DB connection opened for QA serever						
								Step 14: Connect to Table	Table=ZTVV6	Connected to the Table						
								Step 15: Check updated results in DB	NA	Data changes updated						

Test Execution – An Introduction and Planning for Test Execution

Introduction

Test Execution is followed by the Test Design phase in the testing life cycle and verifies if the given system or application behaves as expected.

For Example, in a Car manufacturing company, though every manufactured spare part is tested against its specifications, the car has to be tested as a single unit once all the parts are assembled. Testing the car for functionalities after assembling all the parts is very critical, as that will ensure the reliability and the stability of the product developed.



Test Execution Phase involves:

- ✓ Initial Planning for Test Execution
- ✓ Build Verification Process
- ✓ Test Case Execution
- ✓ Test Log Creation
- ✓ Defect Reporting and Tracking
- ✓ Defect Triaging
- ✓ Updating Traceability Matrix
- ✓ Re Testing of Defects
- ✓ Regression Testing
- ✓ Test Execution Status reporting

Test Execution is the process of executing a series of test cases as defined in the test plan and Test design phases, to validate if the application meets the requirements.

The application is tested and the defects are reported to the development team. The fixes provided by the development team are retested and ensured that they do not recur; the fix does not introduce any new defects.

Testing can be manually performed or it can be automated using software tools. Some of the automation tools are Quick Test Professional (QTP), Win runner, Rational Robot, etc.

Planning for Test Execution

Before testing the application, it is always recommended to ensure that the pre-requisites and the necessary information for Test execution are obtained. The below activities help us to be prepared before executing the test cases.

Test Cases

The Test Cases are prepared by the testing team and signed off by the customer by the end of the Test Design phase. The test cases for each and every level and type of testing are identified and approved by the by the customer.

Test Cycles

The number of rounds the testing will be conducted and the scope of testing at each level need to be identified in the Test Execution plan and agreed upon by all the stake holders.

Test Lab Set Up

The methodology for deploying the build in the respective test environment need to be devised and approved by the development, testing teams and the customer.Test Lab set up also includes creation of a test suite with the test cases which need to be executed in a specific sequence.

The below activities are also taken care during the Test Lab Set up:

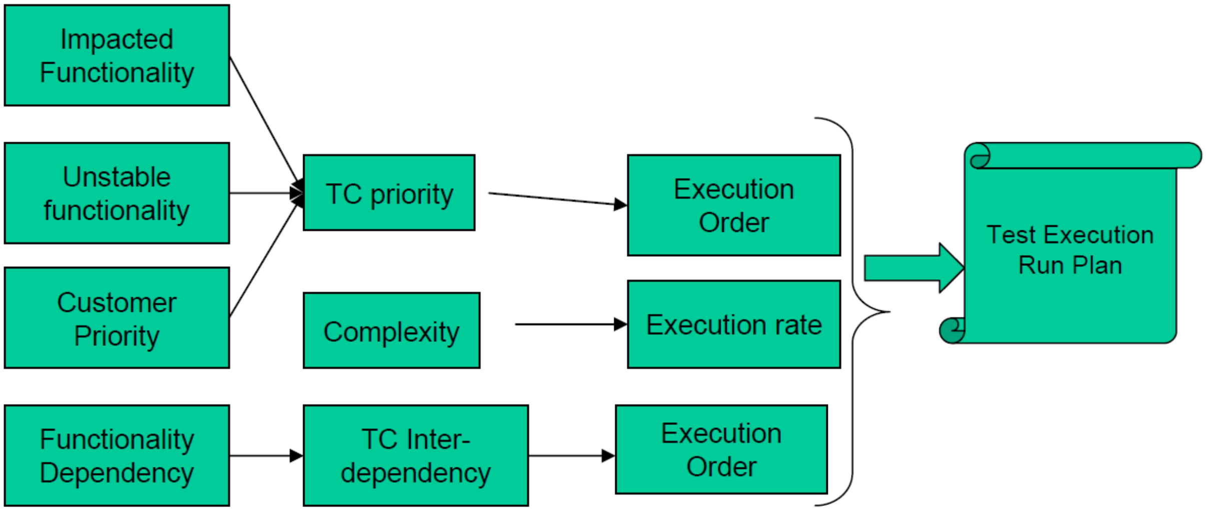
- Creation of cycles for each level of testing
- Identification of Smoke/Build verification test cases
- Identification of Test Suite(collection of test cases) per cycle
- Frequency of Build delivery per cycle

Task Allocation / Run Plan

For effective test execution, allocation of the test cases to be executed need to be done based on the skill set and the availability of the testers.

The following factors need to be considered while creating a run plan and allocating the test cases to testers for execution:

- Interdependency of test cases in terms of functionality and period specificity
- Knowledge levels of the testers - Highly complex test cases are allocated to expert testers to make the execution quick and efficient.



Risks and Mitigation Process

The potential risks that are identified for Test execution during the course of the project are revalidated and the availability of the mitigation and contingency plan for each of these risks need to be ensured.

Escalation Process

The escalation process if an unforeseen event occurs during test execution need to be defined and agreed upon by the stake holders.

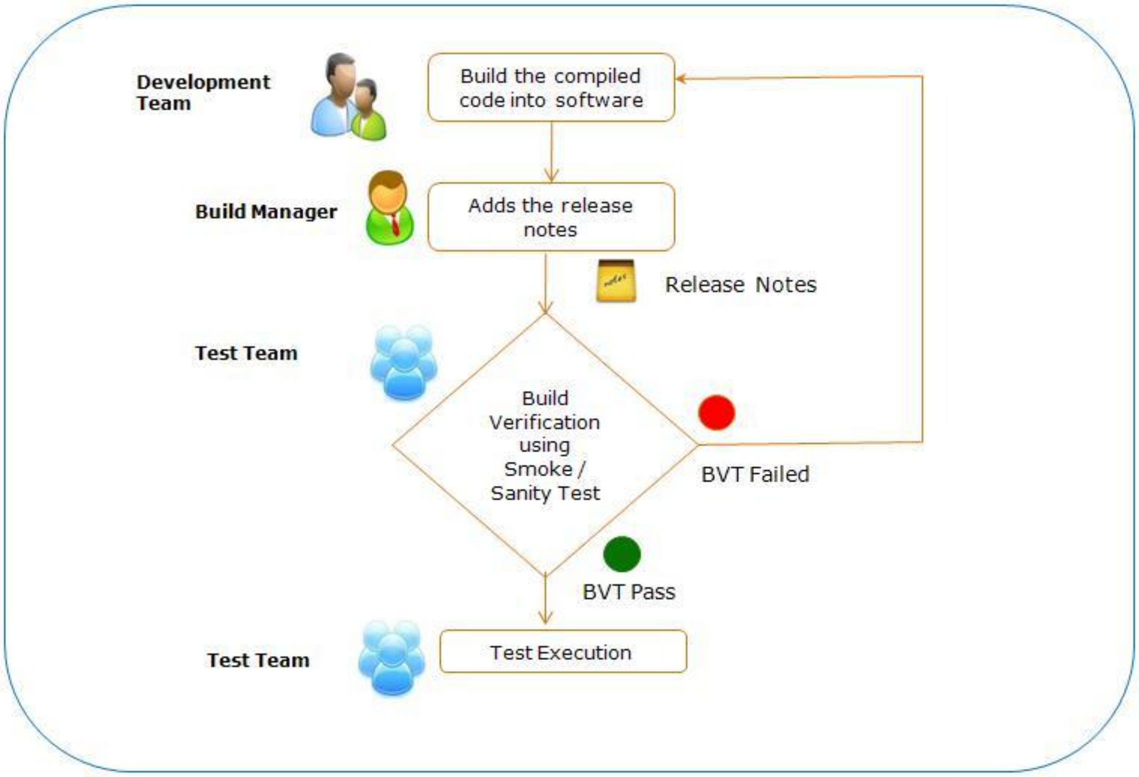
This is to speed up the process if a task is not completed in the specified time frame. Escalation is also done if there is a delay in the fixes of defects which delays the testing process.

Build Management

For every cycle of test execution, the development team shares a controlled version of software (Build) which needs to be deployed in the respective test environment and tested. Each build is associated with a version number and a release notes attached to it.

The release notes of the build primarily include information on the features of the software incorporated in the build and the deployment details. Frequency of build delivery to the project team is as per the Test plan and in consensus with the stake holders of the project.

The files/DB files/Batch Programs used to generate a build are controlled by a configuration tool / team assigned for it. Build Manager is responsible for communicating to the stake holders on the release of the build to test with release notes (In Some Projects, Program Manager). The version of the build is incremented for every change in the build.



The units of software developed is integrated, compiled and built as a single unit by the developers. The Build Manager will add the release notes to this version of the software build and release it for testing.

The Testing team runs the identified Smoke and Sanity test cases on the build, once it is ready. If the smoke/sanity test cases pass with the desired results, build will be accepted and the testing team will proceed with the Test Execution.

If the smoke/sanity test fails with critical defects that the team is not able to proceed with testing, the build will be rejected.

Release notes includes details on

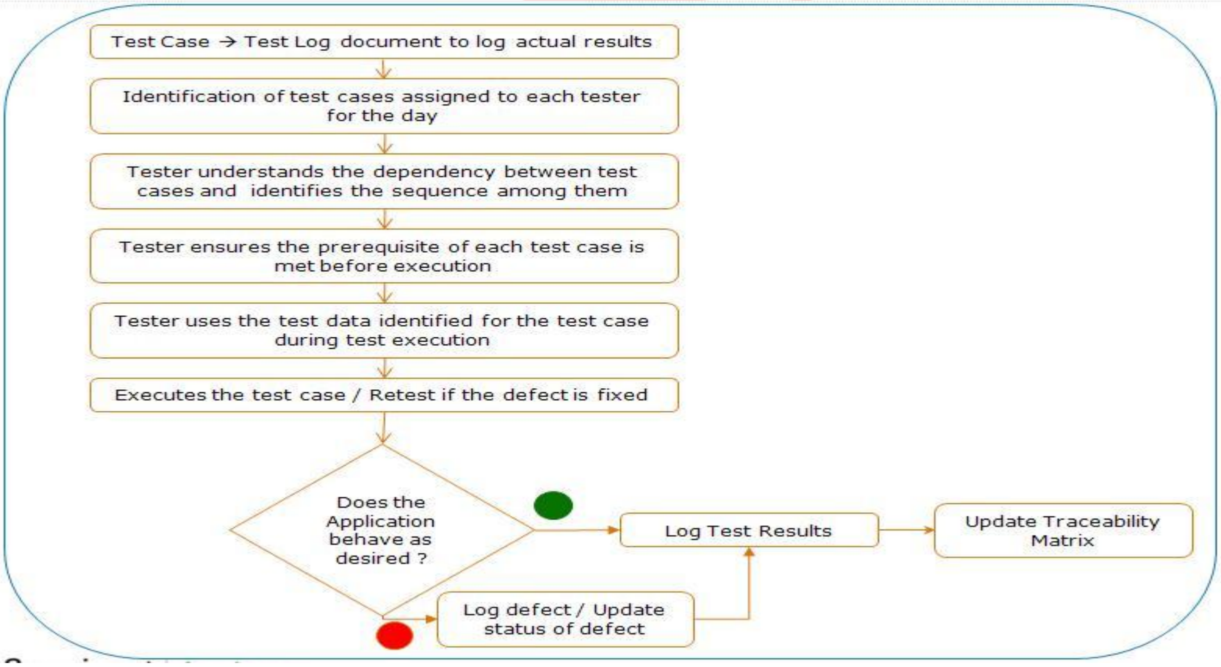
- ✓ Version of the build
- ✓ Features included / not included in the build
- ✓ Defects that are fixed / not addressed in the build

Testers need to test only those features that are mentioned clearly in the release notes.

Test Case Execution and Reporting

Test Case Execution and Logging results

Test Case Execution is process of determining whether the application or system is behaving as desired or not.


Test Case → Test Log document to log actual results
Identification of test cases assigned to each tester for the day
Tester understands the dependency between test cases and identifies the sequence among them
Tester ensures the prerequisite of each test case is met before execution
Tester uses the test data identified for the test case during test execution
Executes the test case / Retest if the defect is fixed
Does the Application behave as desired ?
Log Test Results
Update Traceability Matrix
Log defect / Update status of defect

Before the test execution starts, Test cases are assigned to the individual testers in the Test Execution / Run plan, depending on the dependencies of the test cases and the availability of the test cases. The tester identifies the set of test cases that are assigned to the individual in the run plan and executes one by one in sequence.

The tester looks for the prerequisite in each test case and ensures that the prerequisite is fulfilled before executing the test case. The tester uses the Test data identified for the variances of the test case during the test execution to ensure it works for all possible conditions.

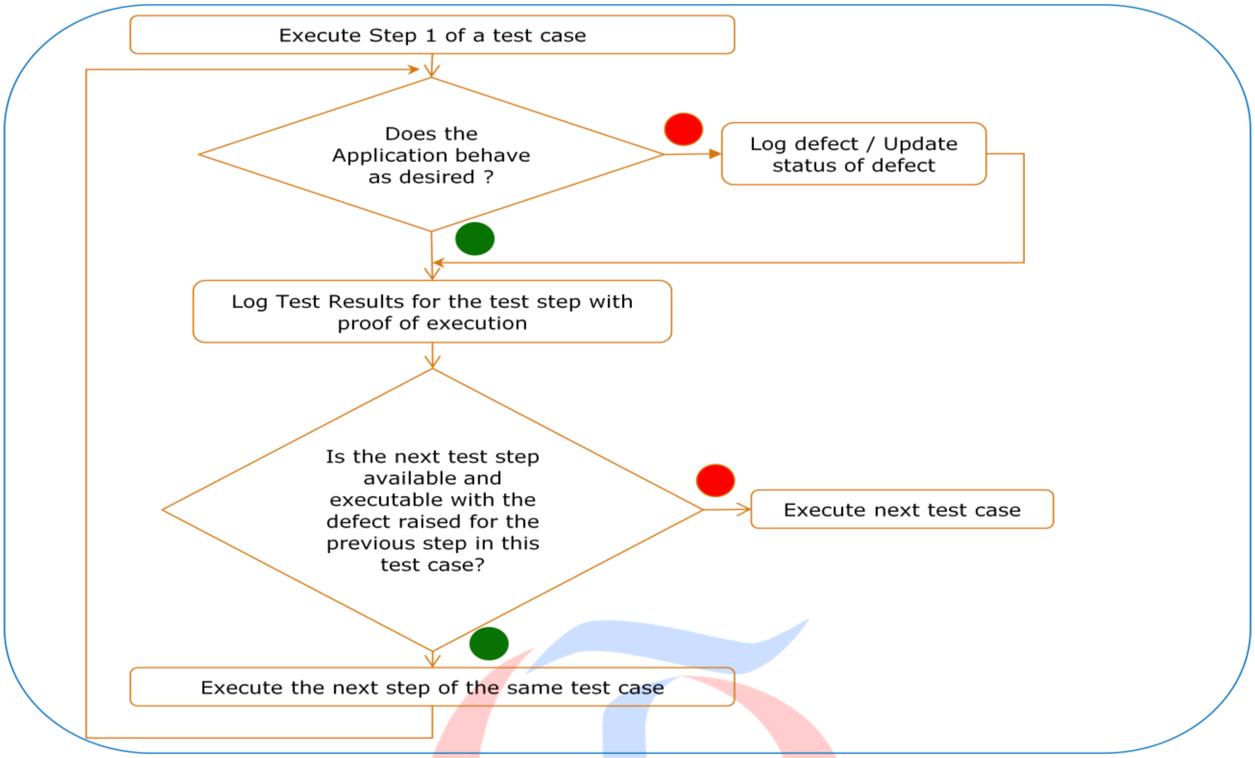
The tester executes the test case step by step as indicated in the Steps To Execute section of a test case. After executing the test case, the actual behavior of the application is logged in the "Actual result" section of the test case

If the actual result is the same as the "Expected result", the application behaves as desired and hence the test case is marked as "Pass"

If the actual result is the not same as the "Expected result", the application is not behaving as desired and hence the test case is marked as "Fail". While logging the "Actual result" the test step in which the test case failed and the behavior of the application, needs to be mentioned clearly. This will help the following

- Other testers in the team to understand the impact of failure of this test case on the test cases that they would execute
- Development team to identify the exact issue and addressing it
- Retrospection of the cause to introduce this defect

Test results are logged with the proof of execution like screenshots with timestamp.


Execute Step 1 of a test case
Does the Application behave as desired ?
Log defect / Update status of defect
Log Test Results for the test step with proof of execution
Is the next test step available and executable with the defect raised for the previous step in this test case?
Execute next test case
Execute the next step of the same test case

Each test case is executed step by step. After executing the test step, the actual behavior of the application is logged in the “Actual result” section of the corresponding test step. If the actual result is the same as the “Expected result”, the application behaves as desired and hence the test step is marked as “Pass”. If the actual result is the not same as the “Expected result”, the application is not behaving as desired and hence the test step is marked as “Fail”.

If a particular test step fails, a defect needs to be logged in the defect log and it is recommended to execute the consecutive test steps if possible.

Example

In the Leave Management system, While applying leave, if the “Type of Leave” combo box does not have the label name as expected, a defect needs to be raised and the testing can be continued for the consecutive test steps of the same test case.

This may not be possible in all cases, and then the tester may proceed to execute the next test case.

Example

In the Leave Management system, while applying leave, if the “From” and “To” dates are not editable.

A test case is considered to be pass only if all the test steps in that particular test case pass.

The actual result of a test case and the details of execution are logged in a document called Test Log/ Test Management tool, irrespective of the behavior of the application. Test log sheet is a copy of test case document with additional details like Actual results, Pass/Fail, Defect ID and the name of the tester who executed it.

Test Log is a chronological record of the all the information about the test case execution

Details of test cases executed

Order of execution

Information on who executed those test cases(Tester)

Status of the each step of the test cases (PASS/FAIL)

This can be created manually or using test management tool

Test Log (Manual process with excel) can be logged by:

Actual Result

After executing the test case, the actual behavior of the application is logged in the "Actual result" section of the test case.

While logging the "Actual result" the test step in which the test case failed and the behavior of the application, needs to be mentioned clearly.

Pass

If the actual result is the same as the "Expected result", the application behaves as desired and hence the test case is marked as "Pass".

Fail

If the actual result is not same as the "Expected result", the application is not behaving as desired and hence the test case is marked as "Fail".

Test log using the Test management tool is possible by clicking the execute button for each and every step of the test case and Tester has to mark fail or pass depends on the result in the system or application.

This will help the following

Other testers in the team to understand the impact of failure of this test case on the test cases that they would execute

Development team to identify the exact issue and addressing it

Retrospection of the cause to introduce this defect

System Testing

During System Testing, the test cases identified to test within the scope of the system are executed. Dependencies on other groups may be very minimal.

System Integration Testing

During System Integration Testing, Test cases identified to validate the data flow across systems are executed. There may be dependencies in executing test cases, associated with various entities like business teams, other vendors, external interfaces, and other application availability. Care needs to be taken to ensure optimum utilization of time and resources during execution – reducing the delays and overlap could be a challenge. In those cases, the responsibilities need to be clearly defined and the test execution run plan needs to be shared with the stake holders.

Test Case Run Status

The run status has to be updated for every test case.

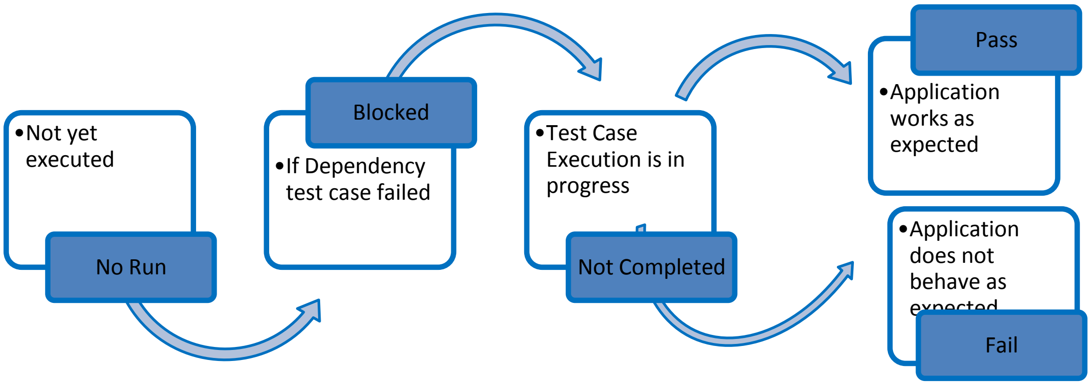
Sample Test Case Run statuses are explained below:

Initially, all the test cases will have "No Run" status. If a test case cannot be executed due to any reason like a dependency test case failed the status is changed to "Blocked". Example: Addition of billing details gets blocked if the login screen of the billing system fails.

The status of the test case is changed to "Not Completed" when a few of the test steps of a test case are executed. Example: If there are 10 steps in test cases and tester is able to test only 5 steps due to some environmental issues, then it is said to be "Not Completed".

If the actual results of all the test steps are the same as the expected results, then the status of the test case is changed to "Pass".

If the actual results of any of the test steps are different from the expected results, then the status of the test case is changed to "Fail".


•Not yet executed
No Run
Blocked
•If Dependency test case failed
•Test Case Execution is in progress
Not Completed
Pass
•Application works as expected
•Application does not behave as expected
Fail

Defect Logging

Testers raise defects when the expected result and the actual behavior of the application do not match. The defect methodology defined for every testing project would be different.

The testers need to follow the one defined in the Test Plan of that particular project. The defect logging can be done in a simple excel document or any defect management / test management tool adopted by the team.

Key Elements of a Defect includes:

Defect Id

A unique Id is assigned to each defect raised during testing. Usually, a naming convention decided by the testing team is followed. Defect ID will be auto generated when the Test management tool is used.

Example: 1001

Defect Description

A textual description provided to describe the defect to the other stakeholders of the project.

Example: Type of leave combo box consists of 3 leaves instead of 5 leave types in 'Apply leave' screen.

Steps to Reproduce the Defect

The list of steps to be followed to reproduce the defect.

1. Login into Leave Management System.
2. Click 'Apply Leave' from Status Menu, Go to Apply Leave screen is displayed.
3. Check the type of Leave combo box.
4. Verify the values of 'type of leave'.
5. Vacation Leave, Sick Leave, Personal Leave are present but maternity leave and loss of pay are not present in the combo box.

Test Case Id

The Id of the test case executed to find the defect is provided for reference. The Test data used to reproduce the defect may also be provided here for reference.

Example: Test Case Id: TC_App_Leave_03

Assigned to

Each defect will be assigned to a developer to fix, the name of the developer is provided here. This is usually assigned during the defect triage meeting.

Example: Enter the name of development module lead for Leave Management System (Say John Peter).

Module Name

The module of the application in which the defect was found is mentioned here.

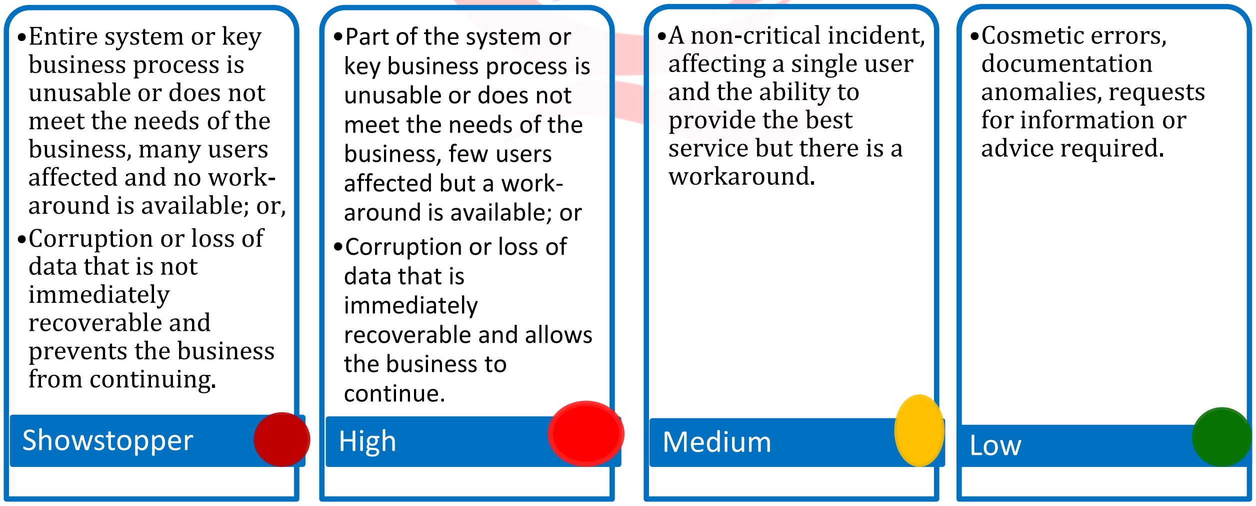
Example: Module Name: Leave Application

Severity

The Impact of the defect in the software is called Severity. Every project will define the severity levels of defects in the Test Plan based on the business need and the nature of business.

Severity of a defect is generally set by the Testing team while logging defects

A few samples of the Severity levels of a defect are



- Entire system or key business process is unusable or does not meet the needs of the business, many users affected and no work-around is available; or,
- Corruption or loss of data that is not immediately recoverable and prevents the business from continuing.

Showstopper ●

- Part of the system or key business process is unusable or does not meet the needs of the business, few users affected but a work-around is available; or
- Corruption or loss of data that is immediately recoverable and allows the business to continue.

High ●

- A non-critical incident, affecting a single user and the ability to provide the best service but there is a workaround.

Medium ●

- Cosmetic errors, documentation anomalies, requests for information or advice required.

Low ●

Example:

a. Showstopper – Login screen of LMS is not working or gets crashed

b. High – Applying of leave in LMS is not working as expected

c. Medium – Applying of leave is happening but only one or two database fields are getting saved in the database

d. Low – Text Box width and length is not present as expected

Priority

- Priority of a defect is generally determined by the delivery manager of the project as defined in the Test Plan based on the following:
    - Severity of the defect
    - Capacity management of the development team
    - Time required to fix a defect
    - Impact of fixing a defect on the other defects – sometimes, fixing one defect may rectify many of the related defects
- Each of these priority levels will have a time associated with it, to enable the developers to fix within that time
    - For Example, Critical defect may have a target resolution time of 2-6 hrs, less critical defects can have the target resolution time anywhere between 8-48 hrs.

Screenshot of the defect

Though the details of the defect are documented, there may be cases where the defect is not repeatable. A clear screenshot of the defect with the relevant area highlighted will help the development team and the business to understand the defect better.

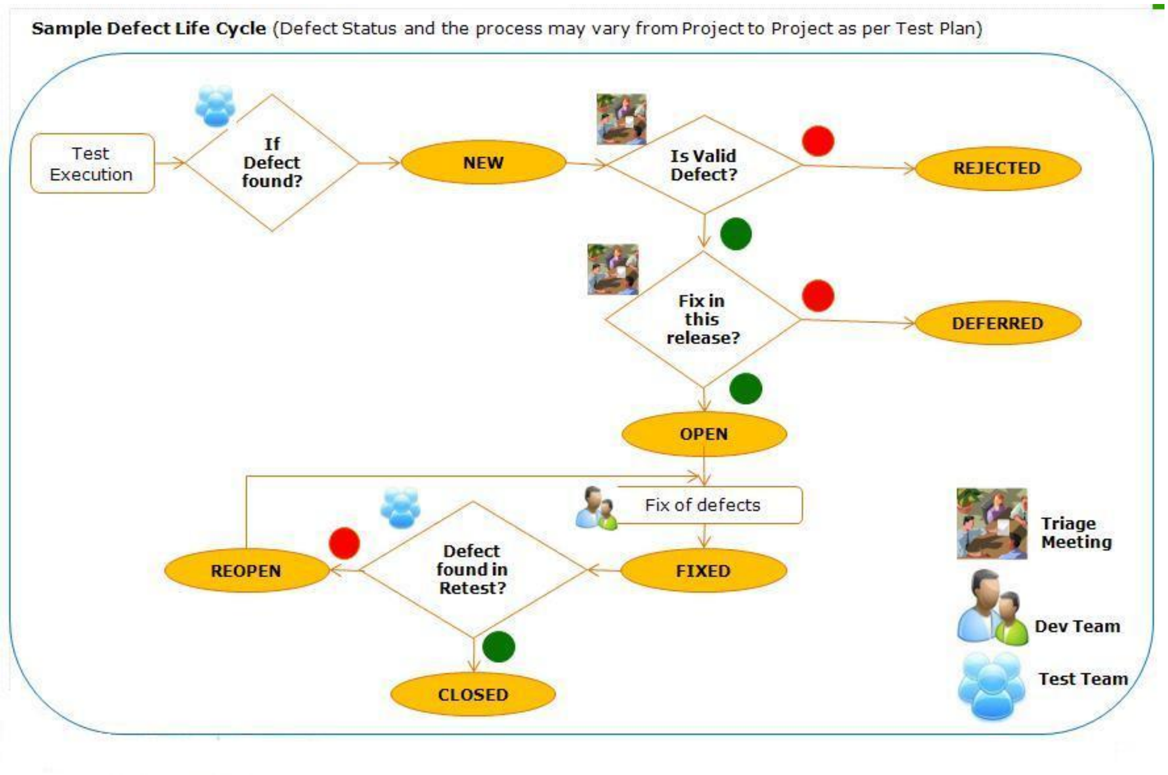
It is always better to highlight the area where the defect had occurred in the screen shot.

Example: jpg file should be attached by highlighting the defect in red color

Defect Status

The status of a defect would change from stage to stage during the defect life cycle.

Note: The defect life cycle is defined in the Test Plan for a project.



Sample Defect Life Cycle (Defect Status and the process may vary from Project to Project as per Test Plan)

Defect Status is New when the tester created a new defect and it will be analyzed by stakeholders during defect triage meeting held every day to discuss whether the defect is valid or invalid, When to Fix, Whom to Fix.

If it is accepted as valid defect, status should be changed to "Open" or it should be "Rejected / Cancelled". If the delivery team decides to fix the defect in the later releases, the status of the defect is changed to "Deferred". The status of the defect is changed to "Fixed" after the development has resolved the same. If fixed defect is again found during retesting, it should be made as "Re-open" or it will be closed.

Test data used to identify the defect

This will help the other stake holders to

- Reproduce the defect
- Identify if the issue is with choosing the test data
- Version of the build in which the defect is identified
- Test Environment details in which the defect occurred

Best Practices while logging a defect

» Clear description of the defect – as is in the application

- Appropriate tone of the language
- Screenshot(s) with
    - Clear marking of the defect area
    - Call outs
    - Timestamp
    - Sequence of screen shots for each step if required
- Highlight if there are any similar / related defects earlier in the remarks
- Mention the other areas/features of testing affected due to this defect in the remarks
- Even if the defect is observed once and is not reproducible, log it and the development team and the business will decide to consider it as a defect or just an observation
- In the above case, log the defect with more details on when the defect occurred and when it is not reproducible

Updating Traceability Matrix

The traceability matrix created during the Requirements Analysis and/ or the Test Design phase is updated with the defect details. The defect id is mapped against the corresponding failed test case id.

This will help in tracking which are the requirements that are misinterpreted / misunderstood by the teams. Only the defect id is mapped here and the details of the defects are obtained from the defect log.

Re-Testing of Defects

Defects that are fixed by the development team are re-tested by the testing team to ensure that the application behaves as desired after the fix. The status of the defect is changed to “Re-Open” and is assigned to the developer if the application does not behave as desired, even after the fix.

Otherwise, the defect status is changed to ”Close” and the test log is updated appropriately. In some cases, fixing of a defect would introduce new defects in the same module or any other related module of the application. Hence the related modules also need to be retested.

Regression Testing

If the application under test is a maintenance application, the enhancement to the application may introduce defects in the existing functionality of the application. Regression testing of the application needs to be performed in this case.

There will generally be a set of regression test cases identified or automated test suite already available.

Best Practices during Test Execution

Below are a few best practices that can be followed during Test Execution.

- Identify the business critical/ error prone areas and focus on those. These can be identified from/using:
  - » Knowledge of the application
  - » Business knowledge
  - » Programming knowledge
  - » End user experiences
- Test the application as an end user
- Try and relate to similar scenarios while testing
- Collaborate with other testers/leads/SMEs in the project to understand
  - » Implications of the defects on other modules
  - » Behavior of the defect if tested with other credentials
  - » Behavior of the defect in various environments – Entire test environment/hardware/software

Test Execution Reports

The status of the Test Execution needs to be communicated to the stakeholders of the project in the agreed frequency as per the Test Plan. Test Execution Reports gives detailed information on the stability of the system after Test Execution and it gives the overview of test case execution status, defect status(Open or closed), Metrics calculation, etc.

The frequency and the details included in the Test Execution reports may vary as below:

- » Daily Execution Report – End of Day
- » Weekly Execution Report – End of every week during Test Execution
- » Test Summary Report (No / No Go Decision) – End of Execution

- These reports will be circulated to the stakeholders including:

  - » Onsite Test Manager
  - » Test Leads
  - » Dev Manager
  - » Program Manager
  - » Onsite Dev Managers
  - » Dev Module Leads
  - » Customer

- ▪ Data for reports can be prepared manually or taken from the tool if a test management tool is used

Below is some of the key information which will be shared with the stakeholders during test execution:

  - » % of test cases executed
  - » No. of test cases passed Vs No. of test cases executed
  - » No. of test cases failed Vs. No. of test cases executed
  - » Defect report
    - By Status
    - By Severity
    - By Priority
    - By Module
    - By age of the defects

End of Test Report

- ▪ At the end of every test cycle, the testing team will share the end of test report with the stake holders
- ▪ This will help the other stake holders of the project to understand
  - » The status of test execution
  - » Summary of defects found grouped by

- Severity
- Status
- Age of defects
- Modules of the application in which the defects occur

» Summary of the modules that are impacted due to the fixes made to the defects

» Changes introduced in the business due to the defects found by the testing team (if any)

» Stability of the application

» Risks involved in moving to the next cycles of testing or UAT with the current state of the application

Data for reports can be prepared manually or taken from the tool if a test management tool is used.

Summary

✓ Test Execution helps to assess the stability of the application or system

✓ The Test Execution involves the below activities:

- Initial Planning for Test Execution
- Build Verification Process
- Test Case Execution
- Test Log Creation
- Defect Reporting and Tracking
- Defect Triaging
- Updating Traceability Matrix
- Re Testing of Defects
- Regression Testing
- Test Execution Status reporting

✓ The following needs to be taken care of before the Test Execution

- Test Cases Review and Sign Off
- Test Cycles
- Test Lab set up
    - Creation of Cycles for each level of testing
    - Identification of Smoke/Build verification test cases
    - Identification of Test Suite(collection of test cases) per cycle
- Frequency of build delivery per cycle
- Task allocation / Run plan
- Risks and Mitigation Process
- Escalation Process

✓ Configuring of Test Environment can include setting up of the following:

- Software required(Ex. Tools like Eclipse, Crystal Reports)
- Hardware required(Ex. Special interfaces like Handheld Device, Touch Screen)
- Server to host the application in the desired configuration
- Database setup/Server Setup to run the application
- Appropriate level of access for test team to access Database, server and application
- Appropriate level of access to external application if the testing scope covers for the same
- Application to be tested
- Test management tool set up
- Access to the remote desktops and the connectivity to them
- Contact Details of the customer's helpdesk should be available during the testing if the environment is down
- Environment availability – All the environments may not be available full time during test execution. There may be exceptions like

    - An application will be running only during US business hours
    - An application will be down for maintenance during US public holidays

- Test Management Tool is a software used to plan and execute testing activities like capturing requirements and test cases, managing defects, collecting metrics and analyzing the reports
- The test environment should be as close to the production environment in terms of
  - Database(DB) set up
  - Levels of users defined and the no. of users defined for each level
  - Volume of data in the database
  - Type of data
- The Appropriate roles identified as test data for testing need to be created and obtained from the customer for all the instances of test environments
- There are two types of test data – Primary and Secondary
- Every build is associated with a version and a release note
- Release notes contains the details of the features of the application in this release and also the details of the defects that are fixed
- Build verification test is done once a new build is received. The build is rejected for testing by the testing team if the build verification fails. Otherwise the testing of the application continues
- Test case steps have to be executed and results should be logged in test log
- A defect is logged against the failed test cases and it has to be retested once it is fixed
- Execution Report have to be prepared with the decision of go/no go to UAT
- Test Management Tool should be finalized and used for execution right from the beginning of test execution cycle till the preparation of execution report

SESSION - 03

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC):

The Software Development life cycle (SDLC) model is:

1. An approach to have a sequence of steps to system or Software product
2. To execute the process from start to finish without revisiting any previous step

Why we need SDLC:

- To Execute projects with proven frame work
- To define and focus roles and responsibilities
- To enforce planning and control
- To have consistency among deliverables
- To Increase productivity by executing the project in systematic manner
- To reduce the rework effort during project

SDLC Activities or Phases:

Project Initiation Phase:

This is 1st Phase in the project life cycle, as it involves starting up a new project.

A project is started by defining its:

- Objectives
- Scope
- Purpose
- Deliverables

Also in this phase we hire the project team, setup the project office and review the project, to gain approval to begin the next phase. The Purpose of the Initiation phase is to start the project.

Concepts Development Phase:

The Concept Development Phase may begin after the approval of the completion of the initiation project status review and the approval to proceed to the Concept Development Phase.

The focus of the phase is two-fold:

1. Evaluate feasibility of alternative and
2. Clearly define and approve project scope, including the system, all deliverables and all required activities.

Planning:

Project Planning – Determines the project's goals and results in a high-level view of the potential project. Proper Comprehensive project planning is essential to a successful IT project, and incomplete project planning and analysis are frequently root causes o project failure.

The purpose o the planning phase is to plan all project processes and activities required to ensure project success and to create a comprehensive set of plans, known as the project Management Plan (PMP), to manage the project from this phase until project termination.

Requirement Analysis:

The Requirement Analysis Phase begins when the previous phase objectives have been achieved. Documentation related to user requirements from the Concept Development Phase and the planning phase shall be used as the basis for further user needs analysis and the development of detailed requirements.

The Purpose of the Requirements Analysis Phase is to transform the needs and high level requirements specified in earlier phases into unambiguous (measurable and lestable), traceable, complete, consistent and stakeholder – approved requirements.

Design:

During the Design Phase the system is designed ti satisfy the requirements identified in the previous phases. The Requirements Identified in the Requirements Analysis Phase are transformation into a system design document that accurately describes the design of the system and that can be used as an input to system development in the next phase.

The purpose of the design phase is to transform the requirements into complete and detailed system design specifications. Once the design is approved, the Development Team begins the Development Phase.

Development:

The Development phase features a key step in the project System construction. The previous phases lay the foundation for system development the following phases ensure that the product functions as required.

To complete the development phase successfully, two elements are required.

1. A complete set of design specifications
2. Proper processes standards and tools.

The purpose of the development phase is to convert the system design prototyped in the design phase into a working information system that addresses all documented system requirements. At the end of this phase, the working system will enter the test phase.

Testing:

The test phase focuses on an empirical investigation in which the results describe the quality of the system testing cannot confirm a system functions property under all conditions but can establish that it fails under specific conditions.

In the test phase, testing of the system proves that the system meets all requirements, including those for performance and security.

The purpose of the test phase is to guarantee that the system successfully built and tested in the development phase meets all requirements and design parameters. After being tested and accepted the system moves to the implementation phase.

Implementation:

The Implementation Phase has one key activity: Developing the new system in its target environment. Supporting actions include training end users and preparing to turn the system over to maintenance personnel. The purpose of the implementation phase is to deploy and enable operations of the new information system in the production environment.

Operation and maintenance

During the operations and maintenance phase, the information system's availability and Performance in executing the work for which it was designed is maintained. System operations continue until the system's termination date, when the next phase, disposition, begins. The purpose of the operations and maintenance phase is to ensure the information system is fully functional and performs optimally until the system reaches its end of life.

Waterfall methodology:

Before we actually dig into the beauty of agile, lets udnerstand what is water fall in reality: Water fall never says we should not go back to second phase, once we are done with it and we are in 3rd phase. If that is so, there is no bugfix in water fall, because development comes before testing and you can not develop if you get a bug.

What is waterfall then?

Water fall says that, one can not visit the 2nd phase unless he / she is actually on second phase. Say, a tester can not start testing unless the project is moved from dev phase to testing phase. Basically one can not feel anything early.

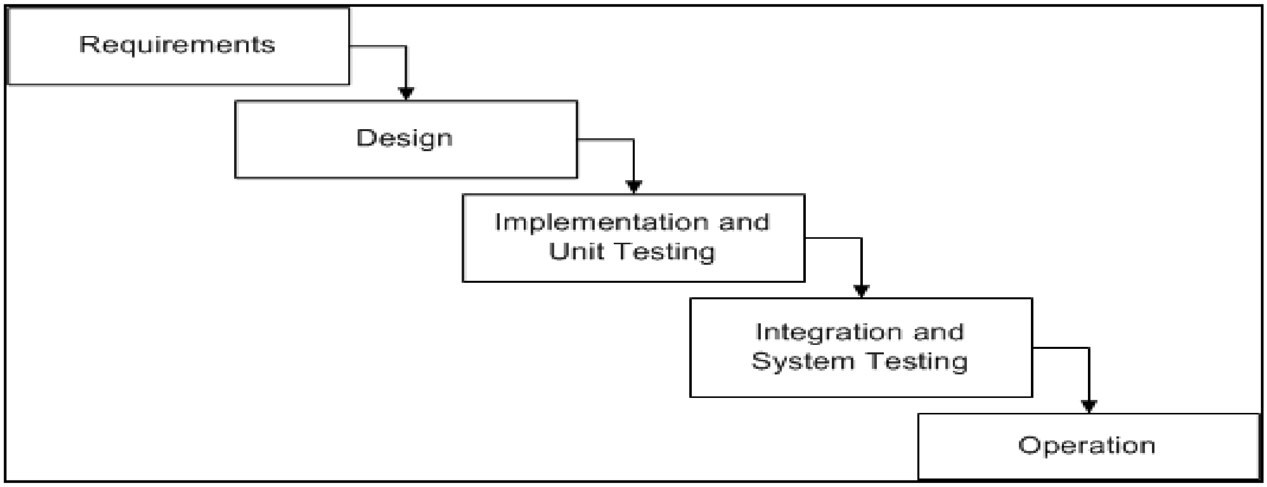
History:

Winston Royce said in 1970 that: sneak peak testing. This means, a tester should feel the testing in a project even before we start with testing phase.

Waterfall Model main features:

- ✓ Whole process of software development is devided into separate phases
- ✓ Derives from its name, giving cascading effect from one phase to another phase
- ✓ Each ohase has well defined starting and ending point with identifiable deliveries to the next phase.
- ✓ Most commonly used model

Block Diagram:


Requirements
Design
Implementation and Unit Testing
Integration and System Testing
Operation

Waterfall Model Disadvantages:

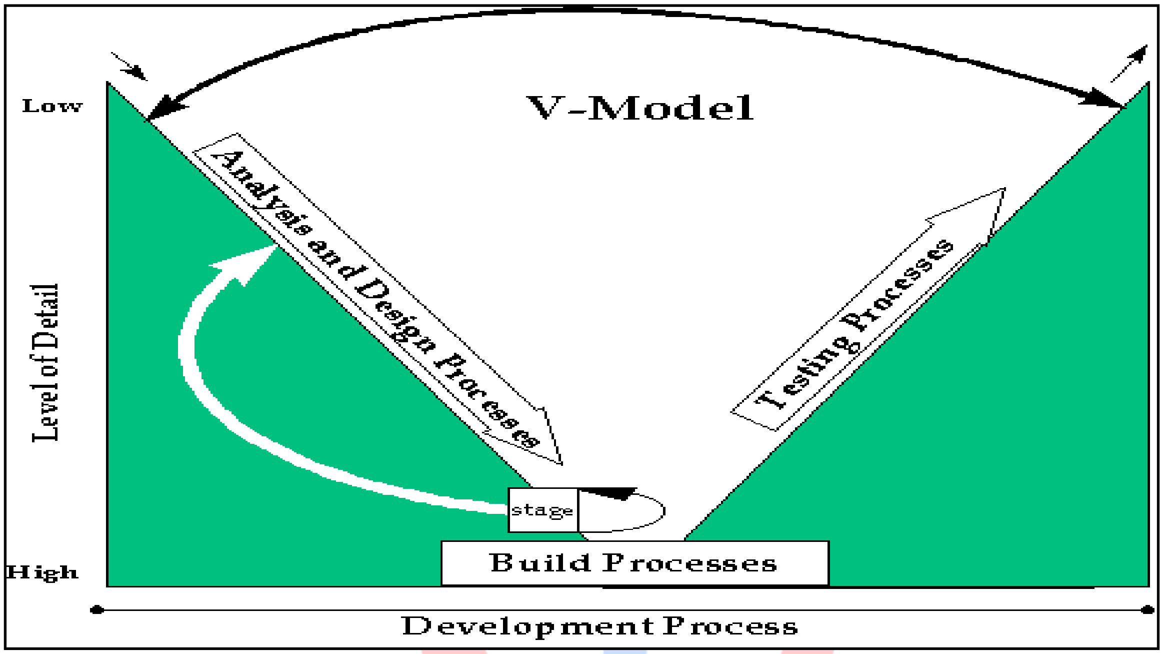
- ✓ No working software is produced until late during the life cycle.
- ✓ High amounts of risk and uncertainty.
- ✓ Poor model for complex and object-oriented projects.
- ✓ Poor model for long and ongoing projects. Works well for smaller projects where requirements are very well understood.
- ✓ Poor model where requirements are at a moderate to high risk of changing.
- ✓ Rework would be more.
- ✓ No early software
- ✓ No early customer feedback
- ✓ Testing is going to start after coding(No paraalel testing)→Defect fixing cost is relatively high

Waterfall vs. Agile:

Waterfall	Agile
Heavy documentation	Effective documentation
Dedicated teams	Multi functional teams
Defined contracts of work	Flexible scope of work
Manged by separate teams	Self manageble
Every phase might not contain a delivery	All phases are going to have live product as end result.

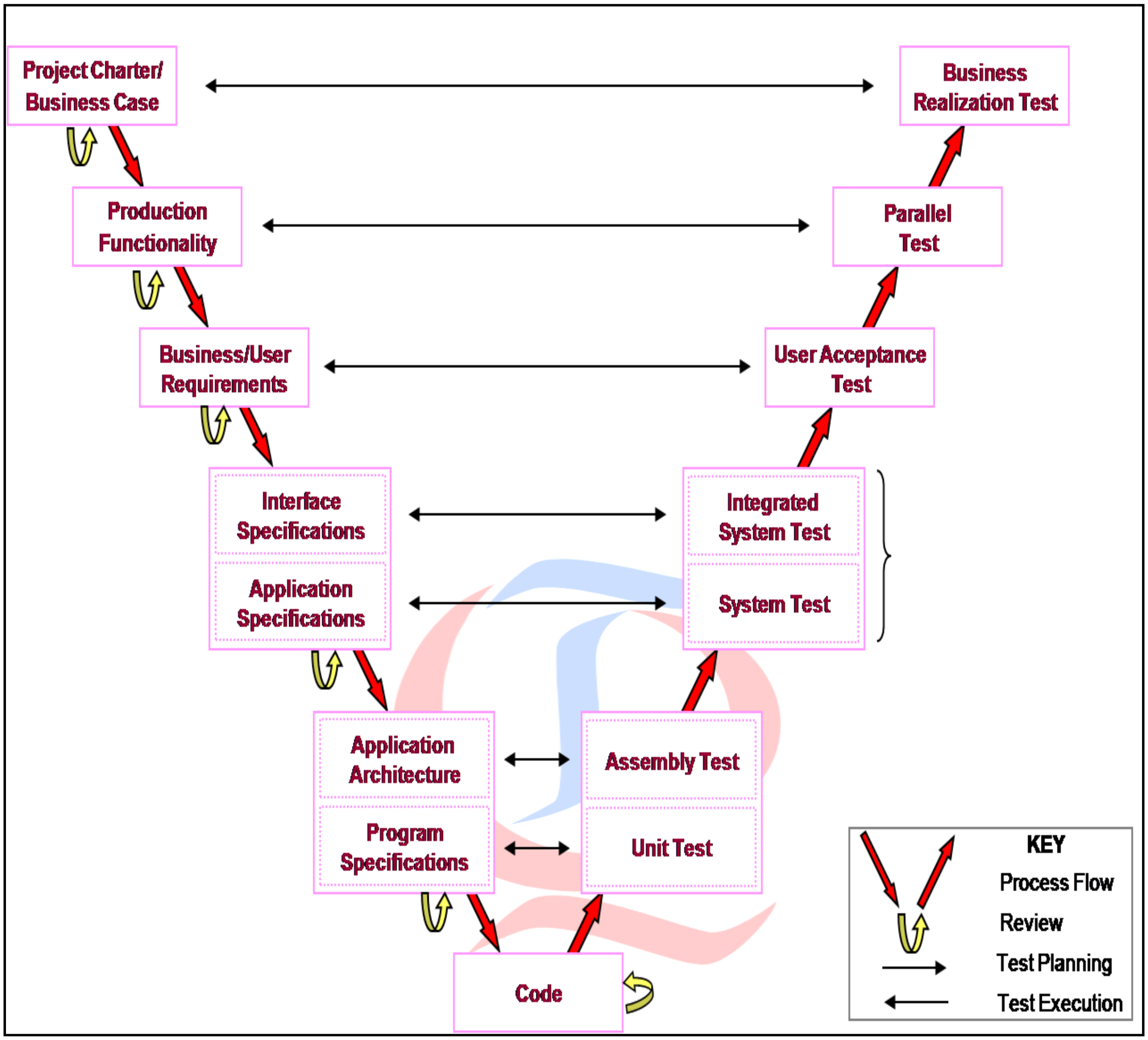
V Model:

The V-Model for Testing provides a structured development framework, emphasizing building quality in from the initial requirements stage through the final testing stage.



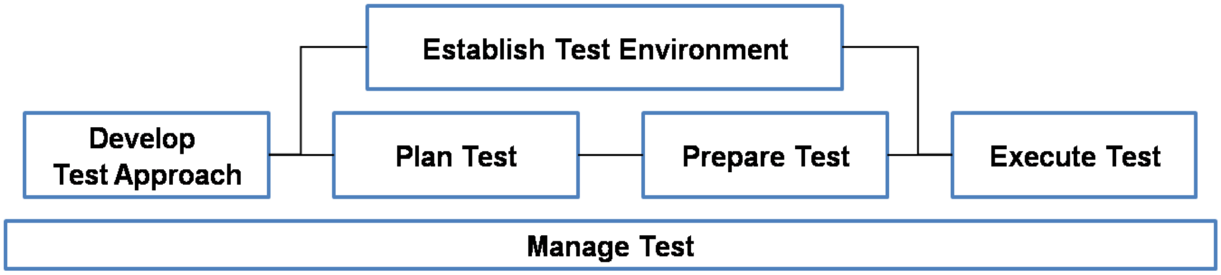
V-Model Desription:

- ✓ Development effort begins on the left-hand side of the V-Model with analysis and design activities
- ✓ Project is specified top down, making decisions and adding more detail at each new specification stage
- ✓ During each analysis or design stage on the left, a test plan is developed, which documents the test cases and expected results required to test the implementation of the specification in the corresponding testing activity on the right side
- ✓ When the design is complete, the build process begins
- ✓ Once development is complete, the application moves through the testing activities on the right hand side of the V
- ✓ Testing done on the right side of then V-Model tests that the related specification on the left side is properly implemented
- ✓ During the earlier stages of testing, the focus is on individual components
- ✓ As testing progresses, the focus in on functionality and achievement of the business case



The V Model demonstrates the relationships between each phase of development life cycle and its associated phase of testing .The V Model Illustrates how testing activities can be integrated into each phase of the software development Life cycle.

During each phase of testing, a structured, repeatable test process is applied and a standard set of deliverables are created.



Test Approach
- Outline Test Scope and Objectives
- Confirm Test Risks
- Update Regression Test Approach
- Determine updated Test Environment Requirements
- Confirm Metrics
- Update Entry / Exit Criteria
- Create Test Work Plans and Test Resource Staffing

Test Plan
- Identify new / updated Test Cases and Expected Results (high level)
- Update Test Case Inventory
- Update Test Script Inventory
- Confirm mapping of Test cases to Requirements
- Confirm mapping of Test Cases to Test Scripts and Components.

Prepare Test
- Develop detailed Test Scripts
- Define and Create Test Data
- Define detailed Expected Results

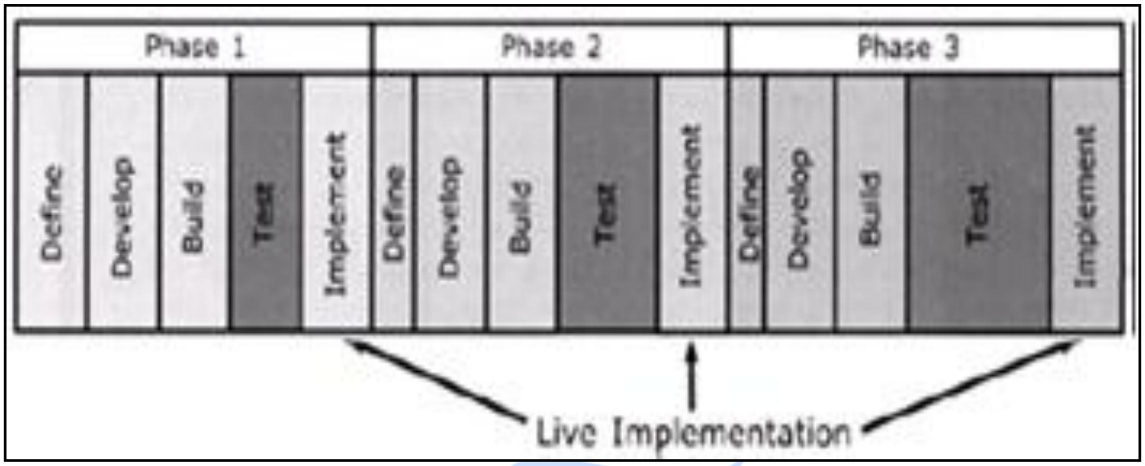
Execute Test
- Determine Actual Test Results
- Log Defects
- Manage Defects

- **Develop Test Approach** provides the objectives, schedule, environment requirements, and entry and exit criteria for the test stage
- **Plan Test** identifies test cases for the test stage
- **Prepare Test** defines test scripts, input data, and expected results
- **Establish Test Environment** ensures that the environment is established and tested before test execution
- **Execute Test** performs the scripts contained in the test model, compares the actual results to expected results, and identifies and resolves discrepancies.

The focus of testing is different across each test phase.

Stage Overview	Step in Delivery Lifecycle	Test Planning Responsibility	Test Execution Responsibility
Validates that the development tasks (individual programs, modules, or components) appropriately implement the functional/technical specifications.	Unit Test	Designers	Development Team
Tests the interaction of related programs to ensure proper function when integrated, as well as testing the program interfaces to validate that the design has been implemented appropriately.	Assembly Test	Designers	Development Team
Focus on validating that the application meets application requirements, workflow and performance targets. System test includes internal system testing, external system testing, and technical testing (performance, volume, stress) at the application level to validate the application meets performance requirements. May include conversion testing.	System Test	Designers	System Testing Team
Validates application interfaces between adjacent systems are functioning correctly. Integration Test ensures business functions are supported across systems. Also includes technical testing (performance, volume, stress) at the application suite level to validate the application suite meets performance requirements. May include conversion testing	Integrated System Test	Designers	Integration System Testing Team
Allows users to review the system in a near-production context in order to validate business requirements and to prove correct operation in a business environment.	User Acceptance Test	Business Analysts and Users	User Acceptance Test Teams/Business Unit(s)
Ensures new application functions properly compared to current production functionality.	Parallel Test	Business Analysts and Users	Parallel Test Team/ Business Unit(s)
Validates that the application can be deployed properly as required for a new application release. Includes installation and data conversion	Conversion Test	Development Team	Development Team

Iterative Development Model



- ✓ Multiple cycles take place here; Cycles are divided up into smaller, more easily managed iterations. Each iteration passes through the requirements, design, coding, testing and Implementation phases.
- ✓ Easier to test and debug during a smaller iteration.
- ✓ More flexible – less costly to change scope and requirements.
- ✓ A working version of software is produced during the first iteration, so you have working software early on during the software life cycle.
- ✓ Iterative or Incremental Development Processes

Prototyping,

Rapid Application Development (RAD),

Rational Unified Process (RUP)

Agile development

Agile Methodology:

Why Agile process is Important:

Agile is a software development process. In today's rapid world, Project stake holders want immediate return on their investment. They don't want to wait longer to get the full features product. As a result, now a day's new software development and testing frameworks are catching momentum Agile approach.

In Agile projects are divided in top small features that are in turn developed and tested in specific time – frames called as sprint (small cycles).The main idea being – features should be developed and tested in specified small time-frames.

Introduction of Agile process:

- ✓ Agile methods break tasks into small increments with minimal planning, and do not directly involve long-term planning.
- ✓ Iterations are short time frames ("time boxes") that typically last from one to four weeks.
- ✓ Each iteration involves a team working through a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders.
- ✓ Team composition in an agile project is usually cross-functional and self-organizing.
- ✓ Team members normally take responsibility for tasks that deliver the functionality an iteration requires.
- ✓ They decide individually how to meet an iteration's requirements..
- ✓ Most agile implementations use a routine and formal daily face-to-face communication among team members.

Principles of agile methods:

- ✓ Customer satisfaction by rapid, continuous delivery of useful software
- ✓ Working software is delivered frequently (weeks rather than months)
- ✓ Working software is the principal measure of progress
- ✓ Even late changes in requirements are welcomed
- ✓ Close, daily cooperation between business people and developers
- ✓ Face-to-face conversation is the best form of communication (co-location)
- ✓ Projects are built around motivated individuals, who should be trusted
- ✓ Continuous attention to technical excellence and good design
- ✓ Simplicity
- ✓ Self-organizing teams

Practical Implementations of Agile Methods:

- ✓ Agile Modeling
- ✓ Agile unified Process
- ✓ Essential Unified Process (EssUP)
- ✓ Extreme Programming(XP)
- ✓ Feature Driven Development (FDD)
- ✓ Open Unified Process Scrum
- ✓ Linear development
- ✓ SCRUM

Scrum Characteristics:

Scrum is a "process skeleton" which contains sets of practices and predefined roles. Scrum is an iterative, incremental framework for projects and product or application development. Scrum has become more and more popular software development and testing framework among organizations in the recent past. Many small to large sized it organizations have started to embrace scrum framework, as this can create excellent quality products in shorter and more quantifiable timeframes than any other traditional software development methodologies. This framework can save companies both time and money.

The main roles in Scrum are:

The name scrum is inspired from rugby game where team work is main key for the success.

An agile project management framework and development process where a team of 5 to 10 people produce increments of products every 14 to 30 days iteration based upon the priority of work set by the customer.

- ✓ Products is designed and built iteratively and incrementally using the product development life cycle in 2 to 4 weeks duration.
- ✓ At the end of iterations, product is delivered based upon the business value of the features as prioritized by the customers.
- ✓ Customer reviews the product and provides feedback to the team.

The "Product Owner", who represents the stakeholders and the business

- ✓ Product owner is the one who decides on priorities of the requirements
- ✓ Product owner is not the person who allocates tasks to team members
- ✓ Product owner may / may not be client-but maintains characteristics
- ✓ Prodcut owner will not give technical suggetions to team members but decides on busuiness

The "Scrum Master", who maintains the processes (typically in lieu of a project manager)

- ✓ Scrum master is the one who acts as bridge between business and team, means acts as bridge between product owner and team (dev, test etc…)
- ✓ Scrum master should be in a position to help any team member to accomplish their tasks by removing impediments in their way.
- ✓ Scrum master should help everyone in the team, from the beginning to the release of every scrum.
- ✓ Scrum master should not allocate tasks but helps in understanding business priorities.
- ✓ Scrum master coordinates with QA and UAT team in getting signoffs for the releases.
- ✓ Organizes the scrum calls / stand-up calls.

The "Team", a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.

- ✓ Here, the cross-functional means, there is no separate tag like tester/developer or business user, rather every one is part of every group.
- ✓ The above statement clarifies that, every one should be having exposure into multiple domains and technology.
- ✓ Team members in agile are self organized and self manageble.

The "Management" folks are someone who is not responsible for any deliveries or someone who is not part of the group but someone who is interested in the project.

- ✓ In Agile team, management doesnot invlove in the allocation of the tasks or priority defination of any activities.
- ✓ Management is not allowed to speak in scrum calls.

Stand-up Calls

Agile beleives that, people can speak hours to gether in meetings but the output is valuable of 15mins worth a day, hence all meetings in agile methodology are stand up meetings, where people should stand and speak, hence they won't speak more than 15mins-which is actually required for the project.

Note: Stand up call generally happens every day at the morning.

Generally, in a call below things are discussed:

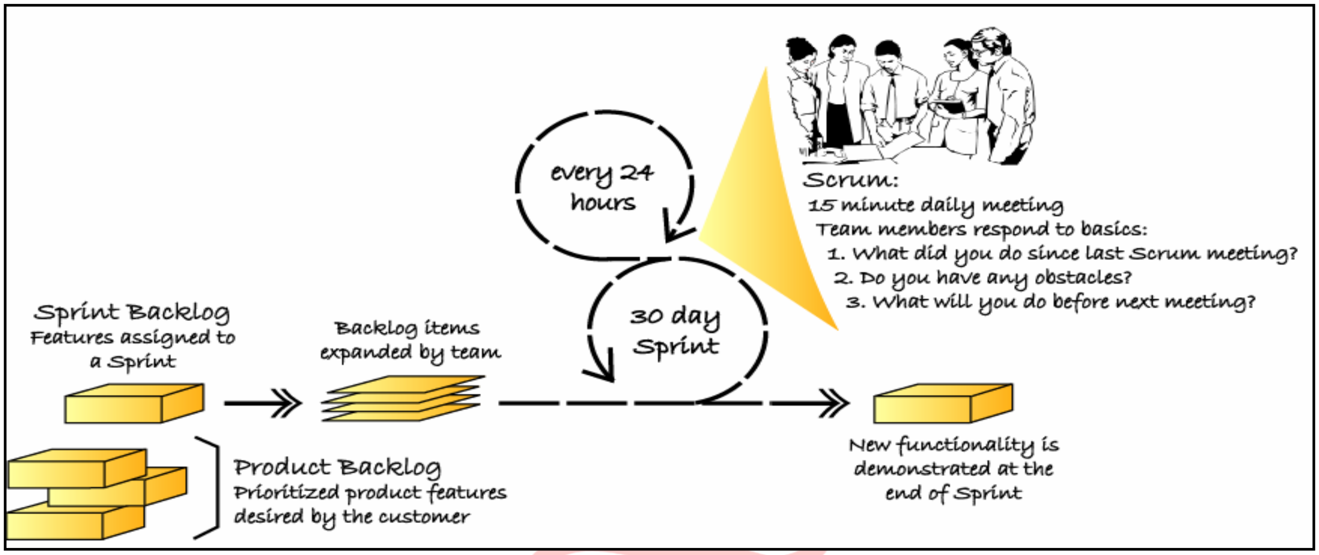
- ✓ What did you do yesterday?
- ✓ What are you going to do today?
- ✓ Are there any obstacles in your way?
- ✓ Strictly, no resolutions in the meetings – resolutions offline.

Terminologies used in Agile

- ✓ Stories

- ✓ Iterations
- ✓ Complexity points
- ✓ Iteration Velocity
- ✓ Product Backlog / Iteration Backlog
- ✓ Scrum / Daily Stand Up Meeting
- ✓ Iteration Planning Meeting (IPM)
- ✓ Iteration Review Meeting
- ✓ Showcase and Retrospective Meeting
- ✓ Burn Up and Burn-Down Chart

The Scrum Framework


every 24 hours
Scrum:
15 minute daily meeting
Team members respond to basics:
1. What did you do since last Scrum meeting?
2. Do you have any obstacles?
3. What will you do before next meeting?
Sprint Backlog
Features assigned to a Sprint
Backlog items expanded by team
30 day Sprint
Product Backlog
Prioritized product features desired by the customer
New functionality is demonstrated at the end of Sprint

SCRUM PROCESS:

- ✓ Stpe1: In Agile methodology all requirements are divided into stories.
- ✓ Step 2: Agile methodology Works in Iterations. Iteration means time line fixed to deliver the Stories without defects. In general Iteration durations are 1 week, 2 weeks or 1month
- ✓ Step 3: Business Analysts will send stories to the team in the starting day of iteration.
- ✓ Step 4: When Iteration starts Developers start working on coding at the same time Testers work on designing of test cases. When the Stories are ready for testing Testers will execute all the test cases.
- ✓ Step 5: Scrum Master Coordinates the Agile project and in Scrum meetings with all teams (BA's, Modelers, Developers, Testing team) discuss about the status on the story
- ✓ Step 5: By the end of iteration all the stories testing should be completed without any defects

- ✓ Step 6: Product Owners i.e. Customers will test the product.
- ✓ Success criteria for Iteration: 1.No critical defects by end of the Iteration, No high defects by end of the iteration

What testers can expect:

- ✓ New features are added to the software in each of the Iteration.
- ✓ Test Teams should expect to run one or more complete test cycle in each iteration of the project, including regression testing previously delivered features as well as the current feature.
- ✓ Testers, Business Analysts, and Developers will work together as an integrated team to define, build, and test software. Testing spans the iteration (time) and spans across the team (roles).
- ✓ Requirements and code continue to change during test cycles.
- ✓ Environments and test data must be ready earlier and more closely managed.

How testers will benefit from an agile approach

- ✓ More features are fully completed earlier in the project, removing the need for a long, high-pressure testing cycle right before the product moves to production.
- ✓ Business Analysts and Testers have more opportunity to provide and receive feedback on the software earlier in the project, leading to fewer major changes late in the effort.
- ✓ More Developer testing reduces the number of "common" defects that the testers need to worry about and reduces the amount of time spent managing large amounts of defects.
- ✓ The defect report/fix cycle is dramatically shortened.
- ✓ An Agile approach can more easily accept changes in customer requests.

Challenges for testers:

- ✓ Tests must be able to run quickly and repeatedly. Iterations last 2-6 weeks. Each iteration should include one to many full testing cycles. This is a huge challenge for efforts that rely on mostly manual testing.
- ✓ Test case creation should begin concurrently with the elaboration of the feature.
- ✓ Test execution must start as soon as features are ready, requiring environments and data to be set up during the first iteration.
- ✓ Performance Testing may need to occur earlier than traditional waterfall efforts.
- ✓ Requirements and code are changed during test cycles, requiring more flexible test schedules, and more regression testing.

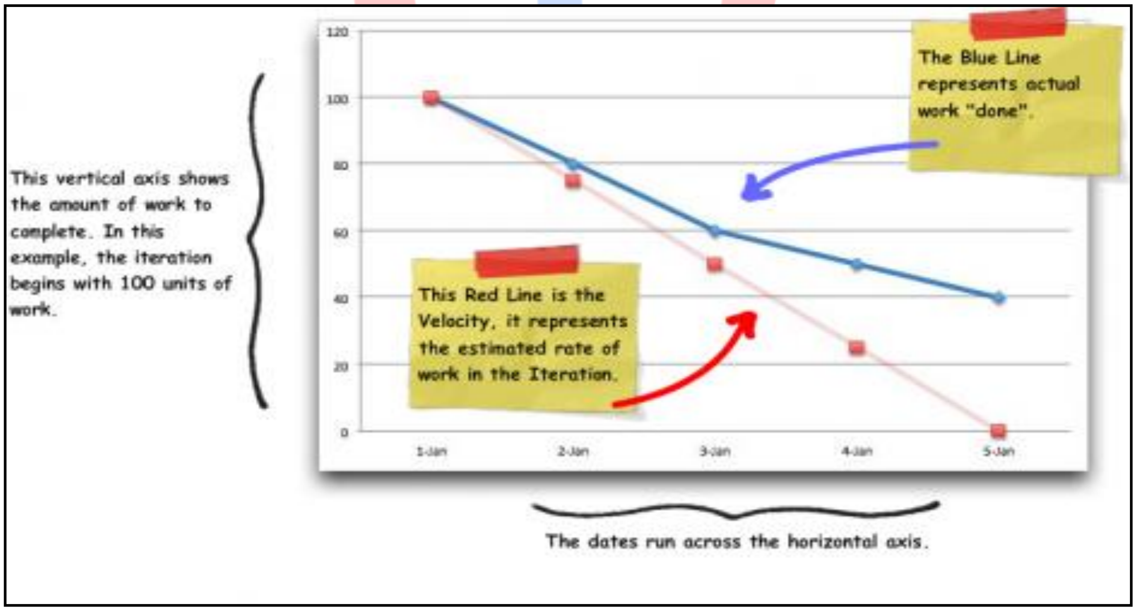
- ✓ Integrating with non-agile projects, can present a timing challenge, as other projects might not have their side of integration points built early enough to do extensive testing.
- ✓ Managing test data and environments on shorter time frames.

Checkpoints

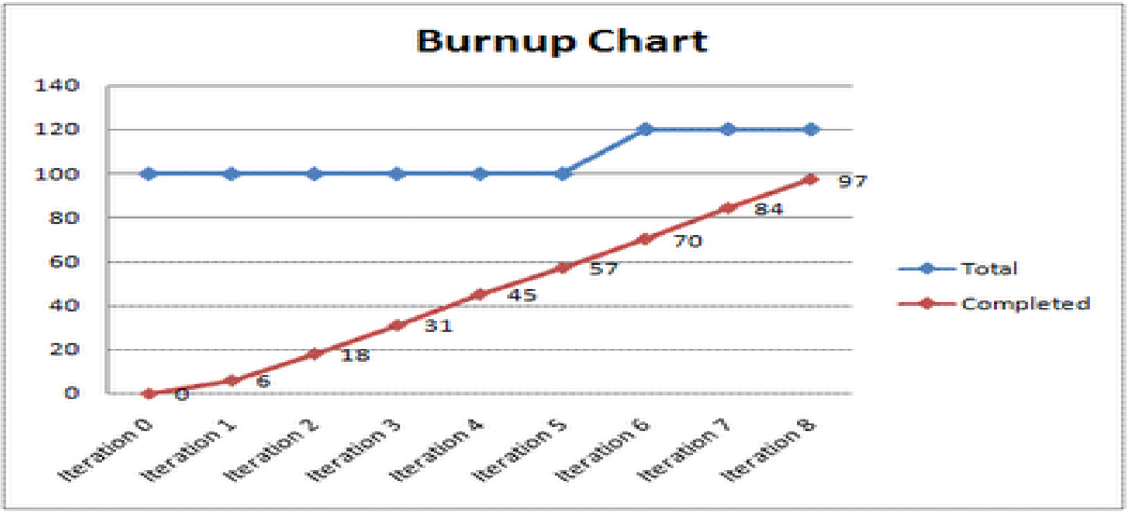
- ✓ Agile Development is a part of which software Development model?
- ✓ What is Velocity?
- ✓ A user story is developed by a?

Burn Down Chart

- ✓ The purpose of the burn down chart is to have a visual representation of the amount of work left to complete in the current iteration.
- ✓ It gives a simple view of the sprint progress.
- ✓ Ideally should burn down to zero to the end of the Sprint
- ✓ The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal



- ✓ The problem is that burn down charts lack two essential pieces of information.
- ✓ First, how much work was actually accomplished during a given iteration (as opposed to how much work remains to be completed)
- ✓ Second how much total work the project contains (or if you prefer how much scope has increased each iteration).



Extreme Programming (XP)

One of the most well-known agile development life cycle models. The methodology claims to be more human friendly than traditional development methods. Some characteristics of XP are:

- ✓ It promotes the generation of business stories to define the functionality.
- ✓ It demands an on-site customer for continual feedback and to define and carry out functional acceptance testing.
- ✓ It promotes pair programming and shared code ownership amongst the developers.
- ✓ It states that component test scripts shall be written before the code is written and that those tests should be automated.
- ✓ It states that integration and testing of the code shall happen several times a day.
- ✓ It states that we always implement the simplest solution to meet today's problems.

Differences b/w XP & Scrum

- ✓ Scrum teams typically work in iterations (called sprints) that are from two weeks to one month long while XP one or two weeks long.
- ✓ Scrum teams do not allow changes into their sprints. XP teams are much more amenable to change within their iterations.
- ✓ Extreme Programming teams work in a strict priority order. Features to be developed are prioritized by the customer and the team is required to work on them in that order. By contrast, the Scrum product owner prioritizes the product backlog but the team determines the sequence in which they will develop the backlog items.

Problems with agile methods

- ✓ Agile says avoid documentation, and it is difficult to have a project which is document free.
- ✓ It can be difficult to keep the interest of Customers who are involved in the process.
- ✓ Team members may be unsuited to the intense involvement that characterizes agile methods.
- ✓ Prioritizing changes can be difficult where there are multiple stakeholders.
- ✓ Maintaining simplicity requires extra work.
- ✓ Requirement will not be freeze, which causes more rework.

How to overcome the challenges

- ✓ Agile allows documentation but one should try to avoid it, meas – one should not have lengthy documentation for any requirements.
- ✓ Follow better documentation approaches such as-specification by example, which says there is no separate document for requirements and separate for testing / uat. Rather have one document which can be used by all.
- ✓ Requirements should be priorotized always in a way that, one should get output always for every release as a live product.
- ✓ Say, there is a coffee vending machine project, one should not prepare a draft plan and say that this is the output of my first sprint as iam in planning phase.
- ✓ But, the expectation from agile is, one should come up with a coffee vending machine itself, may be out of 10, only 2 options are in working condition which gives some use to client.

Best Practices / Lesson Learnt

- ✓ Iteration Tracker
- ✓ Stand Up meetings
- ✓ Defect Query Log
- ✓ Retrospective
- ✓ Defect Removal Efficiency Sheet(DRE)
- ✓ Withdrawn Defects Analysis Report
- ✓ Maintain proper backup plan for every resource in the team
- ✓ Automate whatever and where ever
- ✓ Continuous integration – there are tools available in market
- ✓ Script less automation – for the use of business users
- ✓ Specification by example – best documentation approach

Differences you felt in V-model and Agile Model?

V-Model	Agile Model
1.Verification and validation	Quick moving process
2.Requirements are freezed	Requirements change dynamically so they are not freezed.
3. It's very rigid and least flexible any changes happen mid way, not only the requirements documents but also the test documentation needs to be updated.	Business requirements are designed in the form of iterations, which makes the requirements update process to be flexible.
4.The execution of operation ,maintenance repair and disposal of the system are not covered by the v-Model	Ex3ecution, Repair and Maintenance are covered in Agile model.
5. Involve more documentation. Document involves details of operations performed in before stage.	Less documentation because there is frequent communication between team members of different stage

When to go for agile methodology?

- ✓ When the requirements change dynamically.
- ✓ Continues monitoring of project by client.

When not to go for agile methodology?

- ✓ When there are dependencies.
- ✓ When there is communication barrier between team members or client this makes no quick response from them.
- ✓ For projects where requirements doesn't change dynamically.

Highlights:

- ✓ Several surveys say, agile methodology will be used in 90+% of the projects by all software industries.
- ✓ The purest form of agile will yeild the better results, means:
    - o Agile + best practices + Scrum -> Scragile
    - o Agile - best practices + Scrum -> Fragile

Interview Questions

1. What is SDLC? Can you explain me in detail? Why do we need SDLC?

2. Define various SDLC phases?

3. Describe below models with life cycle, Advantageous, disadvantageous?

- ✓ Waterfall Model
- ✓ V Model
- ✓ Prototype Model
- ✓ Iterative Model
- ✓ Spiral Model
- ✓ RAD Model

4. Can you explain waterfall model in Manual Testing?

5. Can you explain V Model in Manual Testing with your real Time experience?

6. Explain the difference between Waterfall and V Model?

7. Explain differences between V Model and Agile Methodology?

8. Explain the SCRUM Framework process in your Project?

9. Explain Verification Validation Model with each level of Testing?

10. What are the different agile methodologies Implementations? Explain the difference between agile methodology and Scrum Frame work?

11. For which type of projects SCRUM Framework is suitable?

12. What are the agile methodologies Principles?

13. How testers will benefit from an agile approach?

14. What are the challenges for testers in agile methodology?

15. For which type of Projects Agile Methodology is not suitable?

16. How you decide Iteration Backlog in your Project?

17. What is Iteration Planning Meeting, Retrospective Document in Agile Methodology?

18. List few differences between Agile, V model in terms of

    a. Documents preparation

    b. Testing Life Cycle

19. How you are able to complete all the activities in Testing in short life cycle?

20. What are the roles and responsibilities of SCRUM Master?

SESSION - 04

APPLICATION LIFECYCLE MANAGEMENT:

HP (Hewlett Packard) Application Life cycle Management formerly known as Quality Center is a Test Management tool to manage entire Quality Assurance and testing process for an organization. Quality Center is a upgraded version of Test Director built by the same vendor Mercury (Now acquired by HP in 2007).Test Director Version 8.2 onwards is known as Quality Center. The current version of HP ALM is 11.5.

Application Life Cycle Management versions:

1) Test Director
2) 8.0 - First Version QC
3) 8.2 - SP1
4) 9.0 - In 2006
5) 9.2 - In 2007
6) 9.5 - In 2008
7) 10.0 - In 2009
8) 11.0 - In 2010
9) 11.5 - ALM

Quality Centre is a web-based test management tool. It gives us a centralized control over the entire testing life cycle. It gives an easy interface to manage and organize activities like Requirements coverage, Test Case Management, Test Execution Reporting, Defect Management, and Test Automation. All these activities are provided from a single tool, which is web-based and can be accessed from anywhere. Hence, making the task of the testers and managers easy.

Application life Cycle management can be divided into two parts:
- Site Administrator Bin
- Application lifecycle Management Bin

Site Administration Bin: Site administration bin is used for all the administrative activities. Password for site admin is defined during the installation so make sure that remember the password during installation.

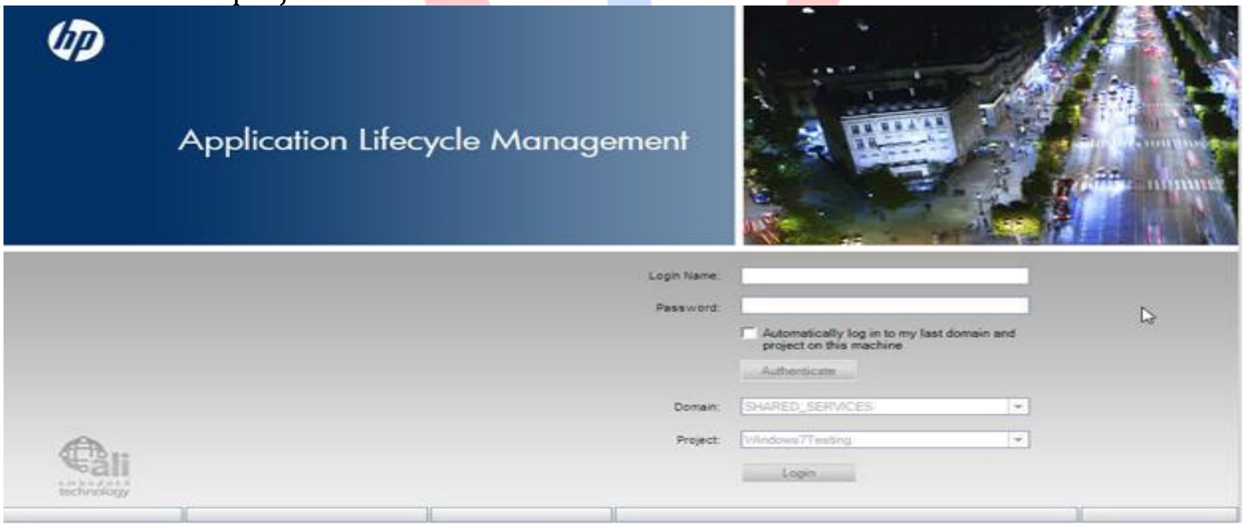
We generally do the following activities in Site Administration Bin:
- Creating the projects
- Assigning users to the projects
- Creating specific roles
- Configuring the mail servers
- Verifying licensing information
- Information about database

Application Life Cycle Management Bin: This part of Quality Centre gives functionality of everything that as a tester or test manager need to do in our day to day activity apart from execution. This is the most common interface used by the customers or users.

We generally do the following activities in ALM BIN:

Creating Releases and Cycles
Defining requirements
Creating test cases
Executing test cases
Raising defects
Traceability,
Integration with functional and Regression test tools (QTP and load runner)

Application Life Cycle Management Login process:
How to start Quality Center:
Step 1: To start Application Life Cycle Management type the address http://<QC [server Name] [local host]>[<:port number >]/qcbin in Qc explorer.
Ex: http://localhost:8080/qcbin
Step 2: Click "Application life cycle management" icon
Step 3: Enter the user name and password then Authenticate button gets activated. click on it. The Domain and project fields get activated. Depending on your login credentials you have access to certain projects.


hp
Application Lifecycle Management
Login Name:
Password:
□ Automatically log in to my last domain and project on this machine
Authenticate
Login
ali technology

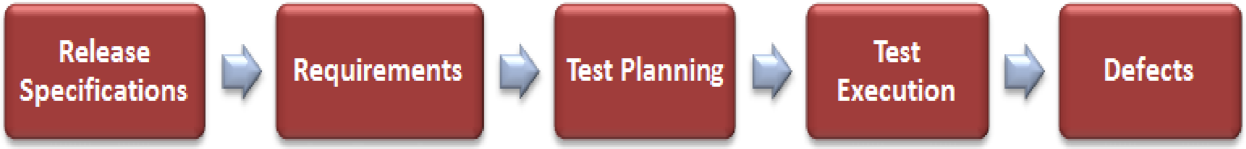
Domain: SHARED_SERVICES
Project: Windows7Testing

Step 4: Choose the Domain and project as required and click "Login" button. Once logged into Application Life Cycle Management window it displays the modules in which we are working.
Step 5: The user domain, project and user information is displayed on the upper
Right hand corner .Also notice the side bar it contains the below components.

- ✓ Dashboard
- ✓ Management
- ✓ Requirements
- ✓ Testing
- ✓ Defects

Application Life Cycle Management testing process:

Application Life Cycle Management flow

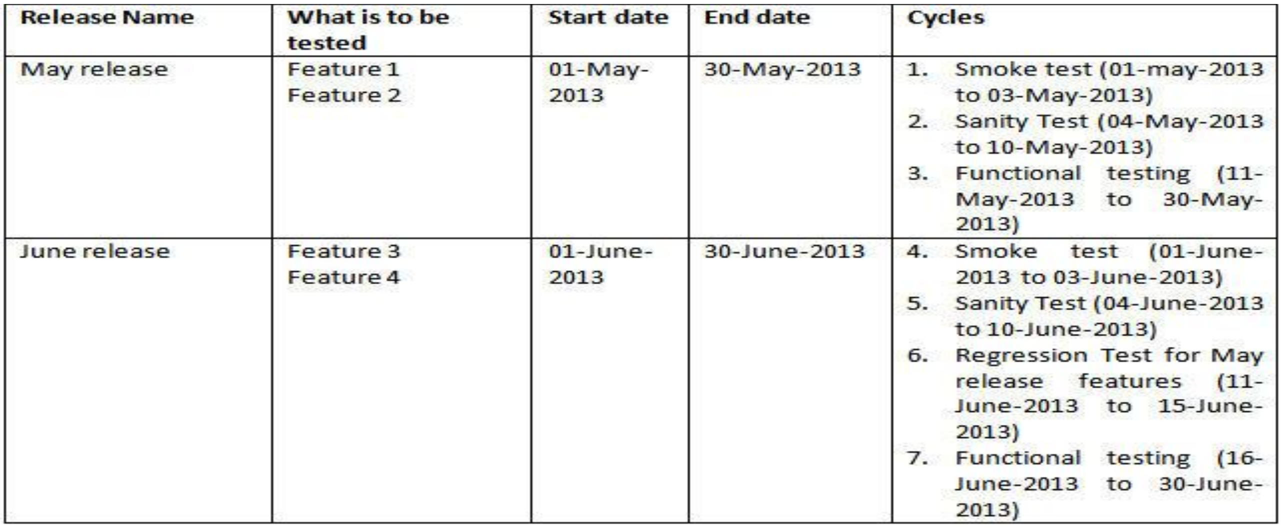


Release Management Module:

Releases: The Releases module enables us to define releases and cycles for managing the testing process.

Release: Release is nothing but a significant change or changes that are going live at a given time.

Cycles: Cycles are the different sets of testing efforts that are performed to facilitate the release.



Release Name	What is to be tested	Start date	End date	Cycles
May release	Feature 1 Feature 2	01-May-2013	30-May-2013	1. Smoke test (01-may-2013 to 03-May-2013) 2. Sanity Test (04-May-2013 to 10-May-2013) 3. Functional testing (11-May-2013 to 30-May-2013)
June release	Feature 3 Feature 4	01-June-2013	30-June-2013	4. Smoke test (01-June-2013 to 03-June-2013) 5. Sanity Test (04-June-2013 to 10-June-2013) 6. Regression Test for May release features (11-June-2013 to 15-June-2013) 7. Functional testing (16-June-2013 to 30-June-2013)

Observations:

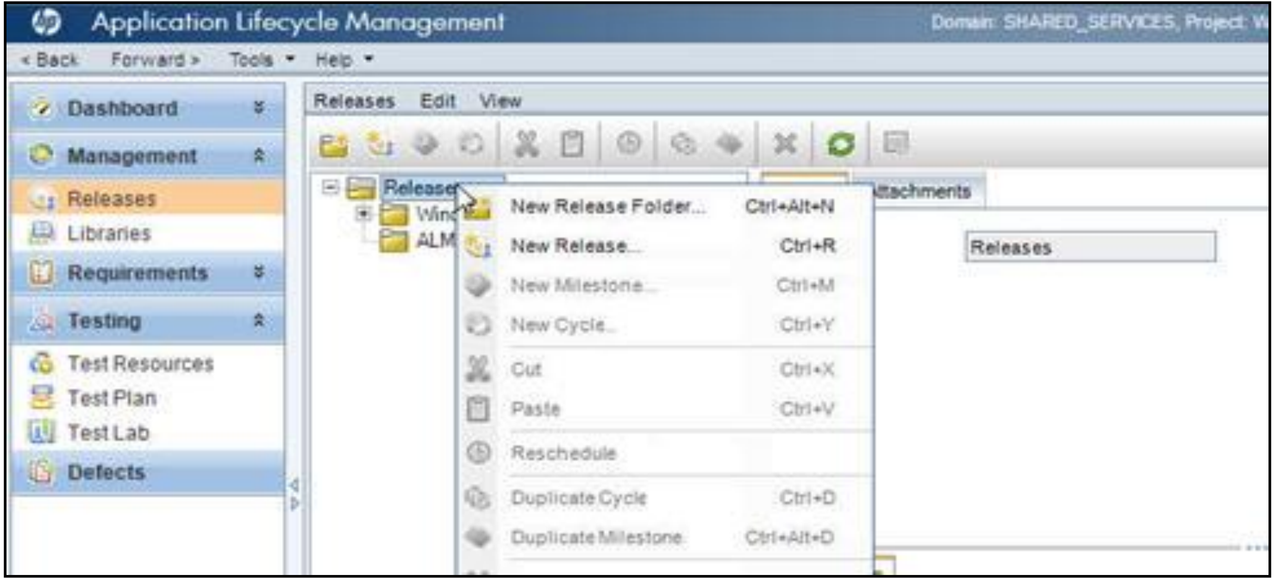
- ➢ Each release comes with a predefined set of requirements that are expected to be done by a certain time.
- ➢ The phases are nothing but cycles. Each release will have different cycles within it.
- ➢ Both cycles and Releases have start and end dates defined.

Steps to create a new release/cycle in ALM:

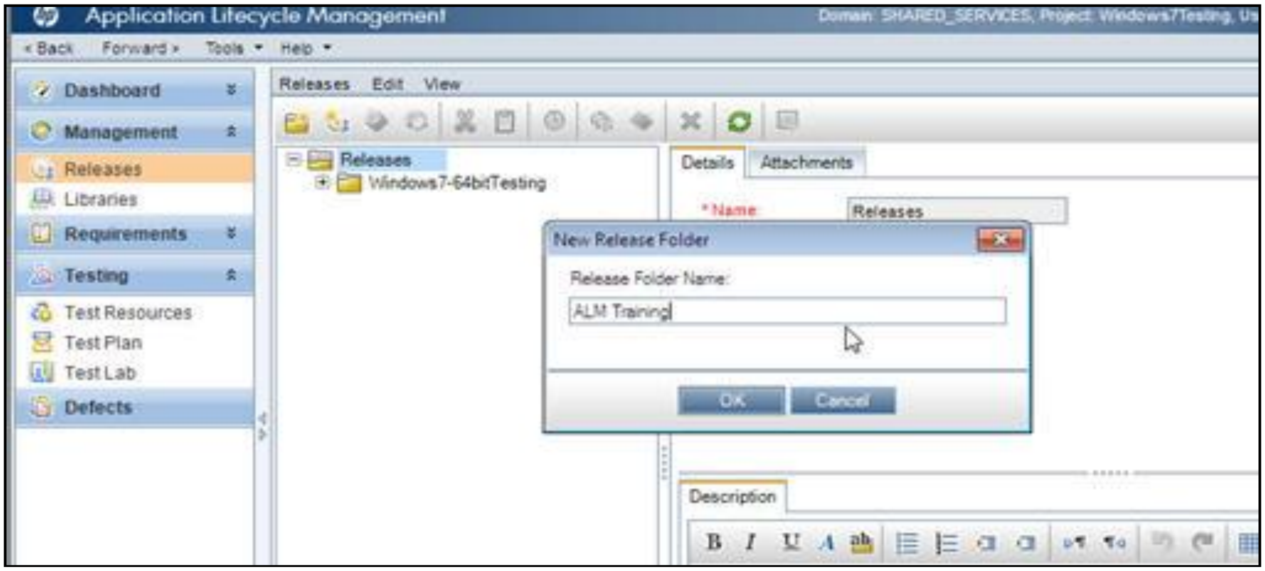
Step #1: Login to Application Life Cycle Management into with valid user name and password, then select the domain and project which we required.

Step #2: Go to the side bar options “Management->Releases”.

Step #3: Create Release folder: We can create a new release folder or directly create a release here. To create a new folder, choose the “New Release Folder” from the menu or right click on the “Releases” root folder and choose “New Release Folder”. Enter the name and Click OK. The folder gets added to the hierarchical structure under Releases.



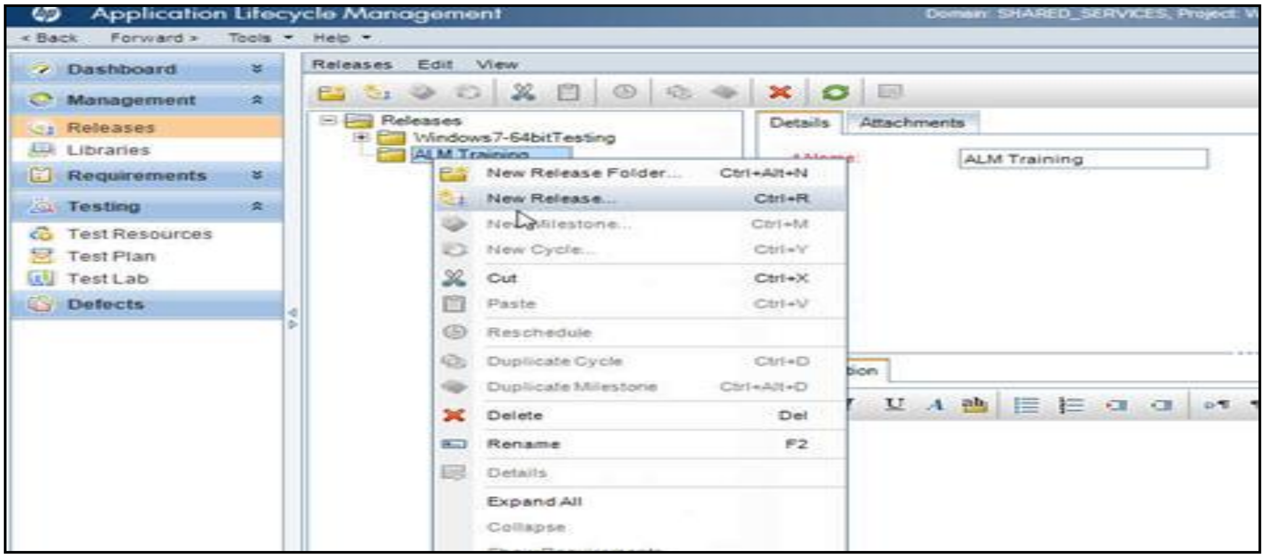
Enter the folder name:



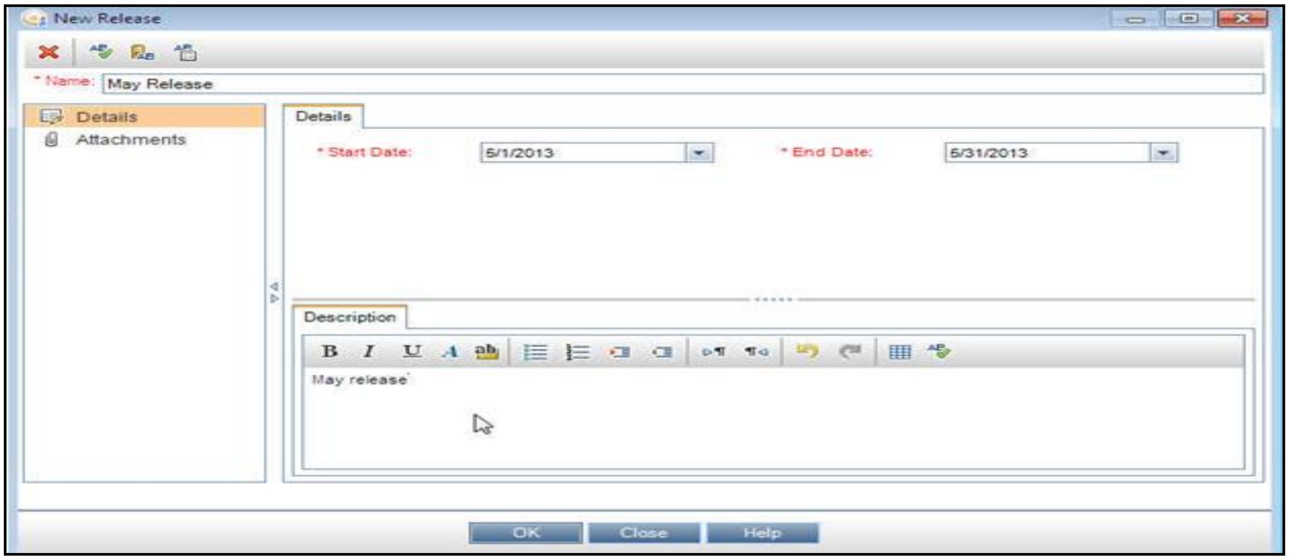
* Name: Releases

Release Folder Name: ALM Training

Step #4: Create Release: Under the newly created folder, we can add a ‘New Release” and we can choose the “New Release” from the menu or right click on the root folder and choose “New Release” or choose Ctrl+R. Enter the Name. As we can see, a start date and end date are the required fields. There is a description box where we can enter any text and then we can upload any supporting documents under attachments. Enter the information as required and click OK.



Enter release details



* Name: May Release

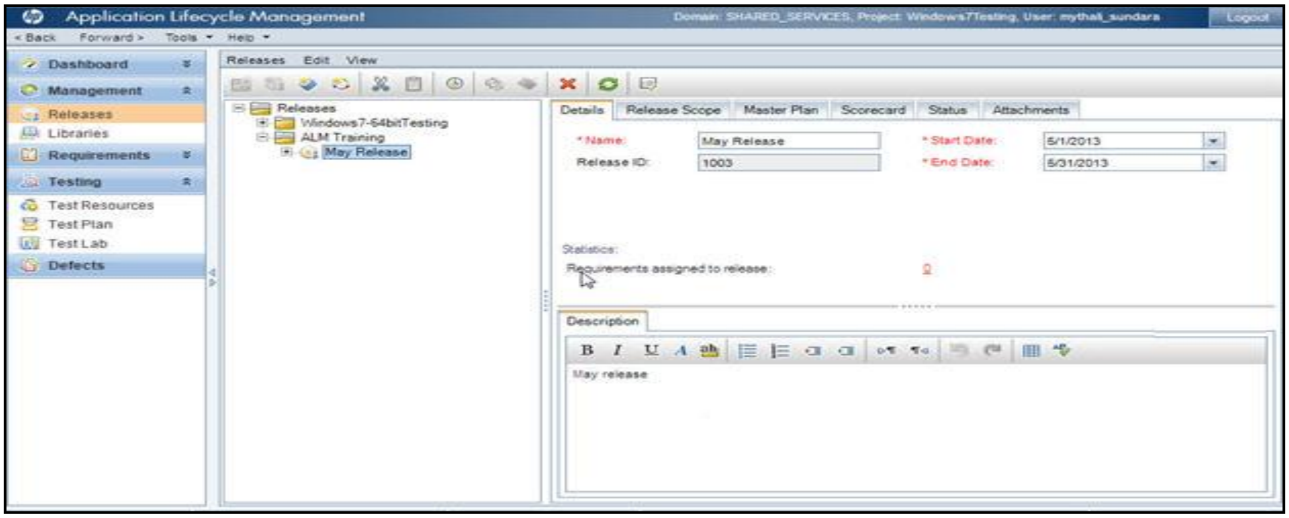
Details

* Start Date: 5/1/2013

* End Date: 5/31/2013

Description: May release

Step #5: The release gets added.



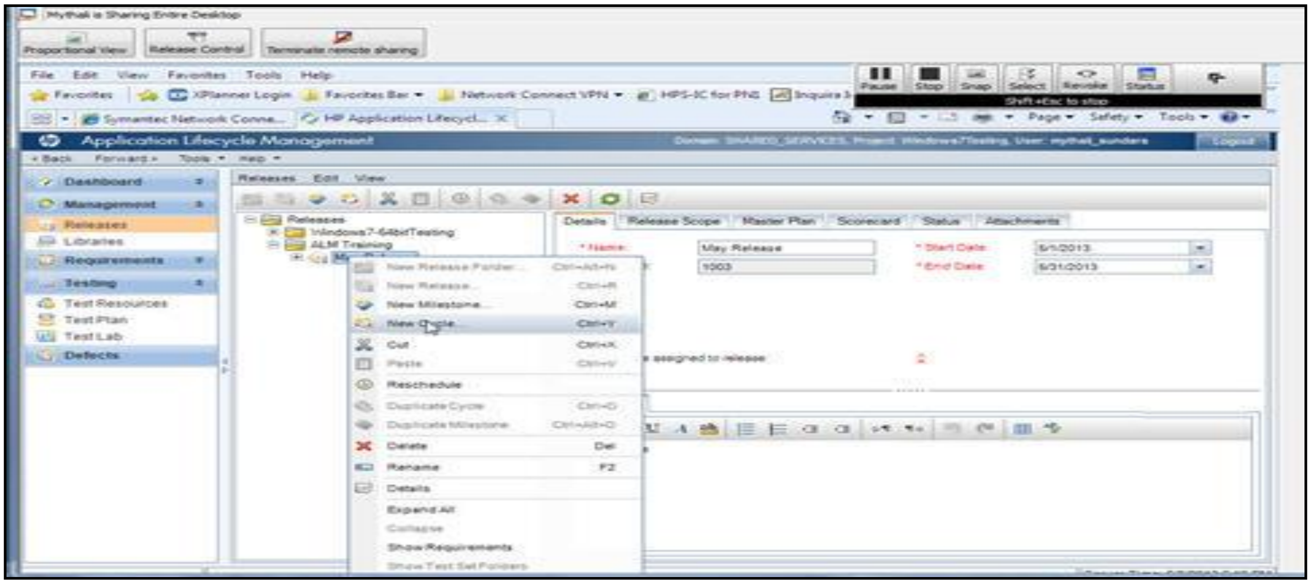
* Name: May Release
Release ID: 1003
* Start Date: 5/1/2013
* End Date: 5/31/2013

Statistics:
Requirements assigned to release: 0

Description

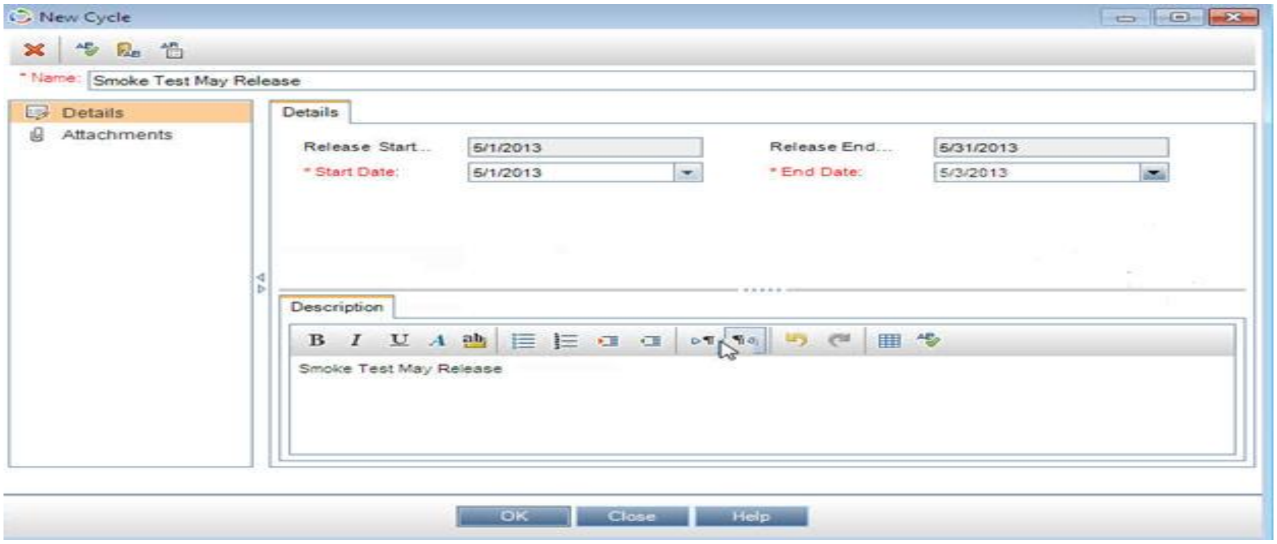
May release

New Release folder name → Release name -→Cycles
Project Name -→ Release version →Cycles
Policy Magic -→ May Release → Cycles(1,2&3)
Step #6: Adding cycles: The next step is to add cycles. Under the newly added release, the options to add a new release folder or release are disabled. We have options in the menu to add new cycle activated or the right click menu has the option or Ctrl+Y keys can be used. Use one of these and choose to add a new cycle. Again name, start date and end date are mandatory items. Enter the same and click OK.



* Name: May Release
1003
* Start Date: 5/1/2013
* End Date: 5/31/2013
s assigned to release: 0

Enter cycle details:



* Name: Smoke Test May Release

Release Start...: 5/1/2013

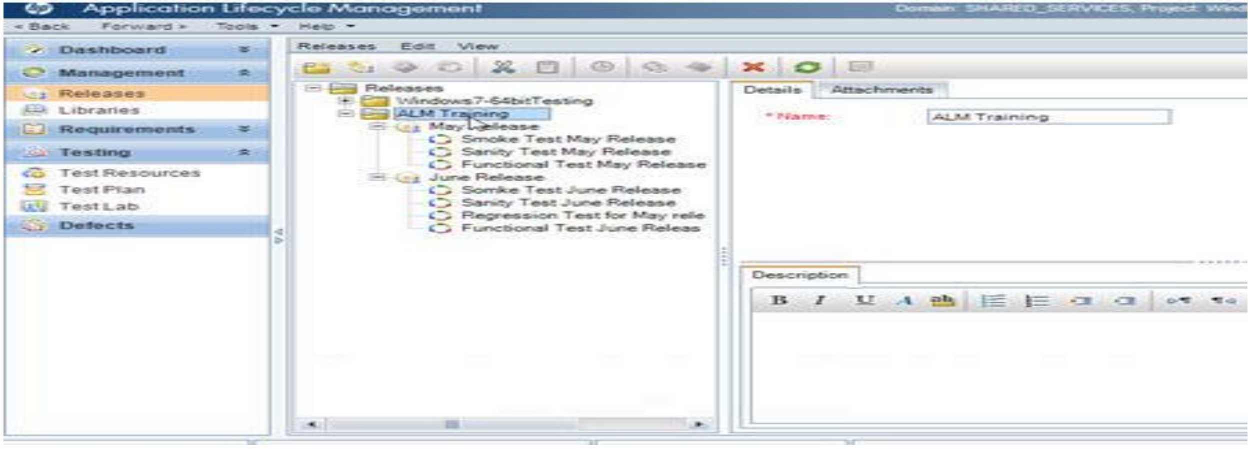
* Start Date: 5/1/2013

Release End...: 5/31/2013

* End Date: 5/3/2013

Description: Smoke Test May Release

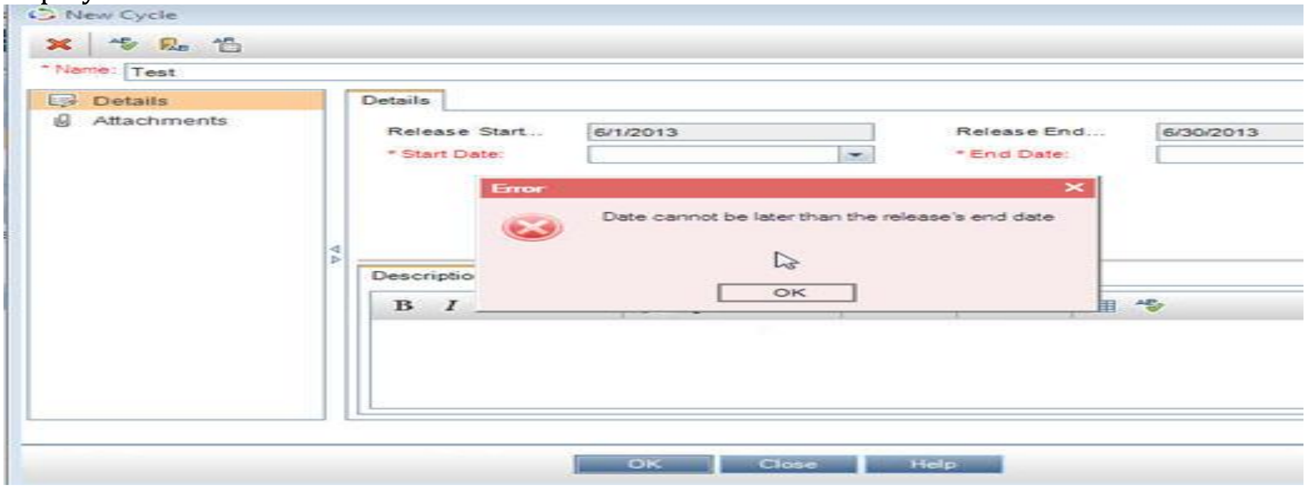
Step #7: The cycle will now get added under the cycle. This is how the May and June release information from our table is going to look once it is all added to ALM.



- Releases
  - Windows7-64bitTesting
  - ALM Training
    - May Release
      - Smoke Test May Release
      - Sanity Test May Release
      - Functional Test May Release
    - June Release
      - Somke Test June Release
      - Sanity Test June Release
      - Regression Test for May rele
      - Functional Test June Releas

* Name: ALM Training

Step #8: There are some validations for entering the cycle dates. They have to fall within the range of the release dates under which the cycle is created. If not, a validation message gets displayed.



* Name: Test

Release Start...: 6/1/2013

* Start Date:

Release End...: 6/30/2013

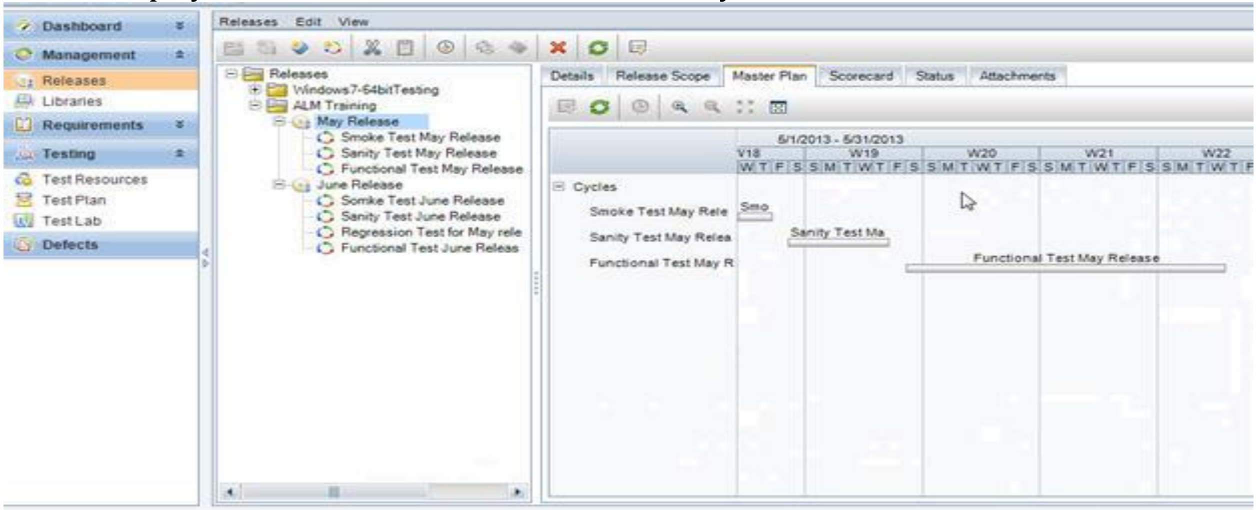
* End Date:

Error

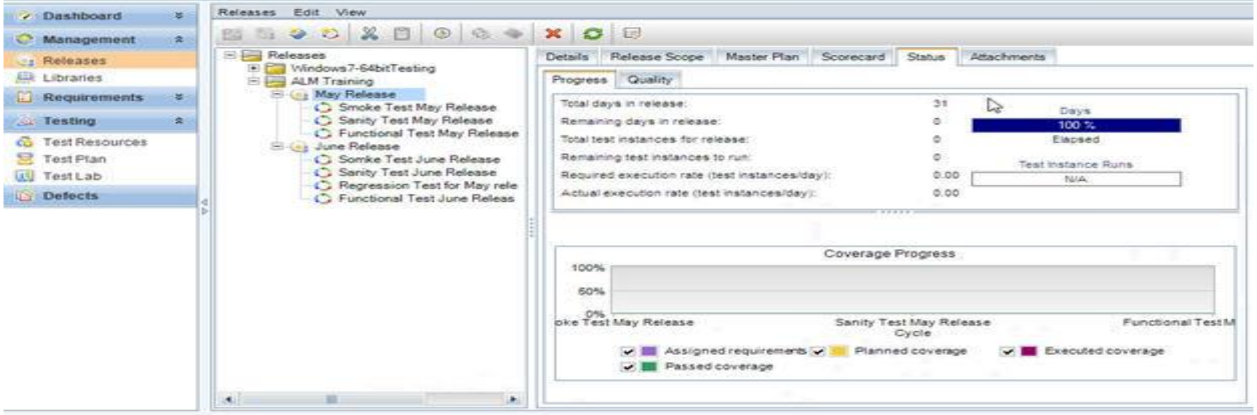
Date cannot be later than the release's end date

OK

Step #9: Once it is created. Choose the “May release” and observe its data. The master plan tab will display the time line for the release with its cycle information:



Step #10: The Status attachment displays the progress of the release:

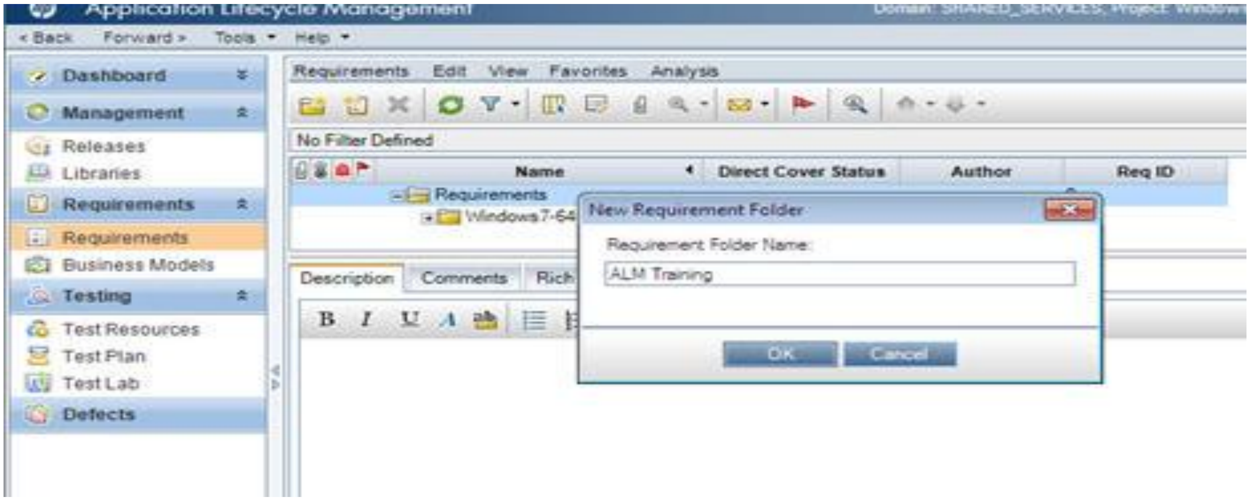


Total days in release: 31
Remaining days in release: 0
Total test instances for release: 0
Remaining test instances to run: 0
Required execution rate (test instances/day): 0.00
Actual execution rate (test instances/day): 0.00

Requirements: The Requirements module enables us to specify our testing requirements. This includes Defining Requirements, Creating Requirement tree and requirement Coverage.

Steps to Create requirements:

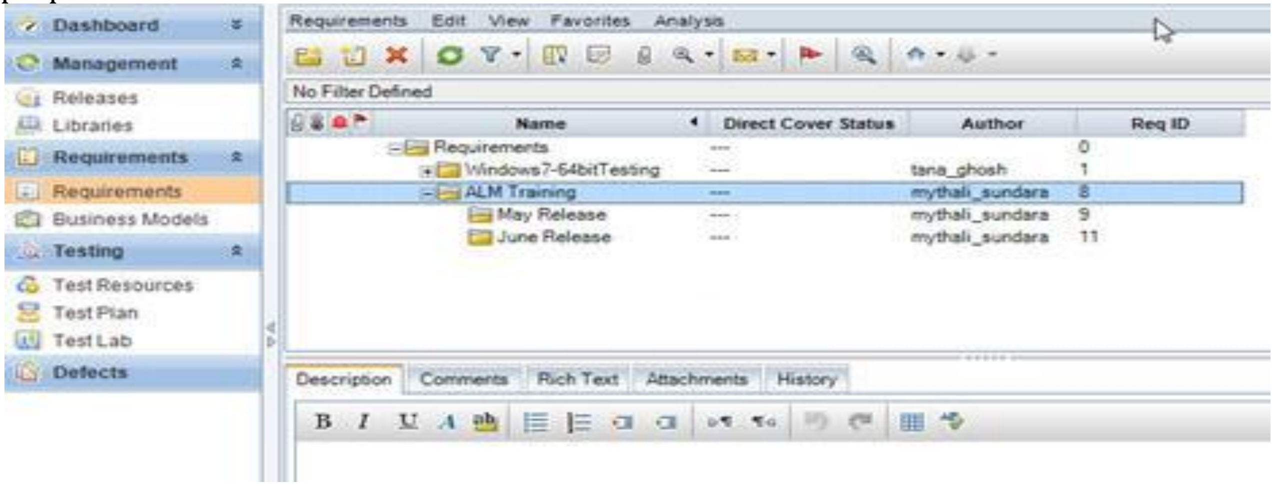
Step #1: Go to “Requirements->Requirements” option from the ALM sidebar.
Step #2: Create new requirement folder. We can create a new requirement folder in 3 ways.1.By clicking + icon directly or choose the “New requirement folder” from the menu or Right click on the “Requirements“root folder and choose “New requirement folder”. Enter the name and click ok. The folder gets added to the hierarchical structure under Requirements.
In this way we can add folder structure as per release module.
New requirement folder → Release name → new requirement name.



New Requirement Folder

Requirement Folder Name: ALM Training

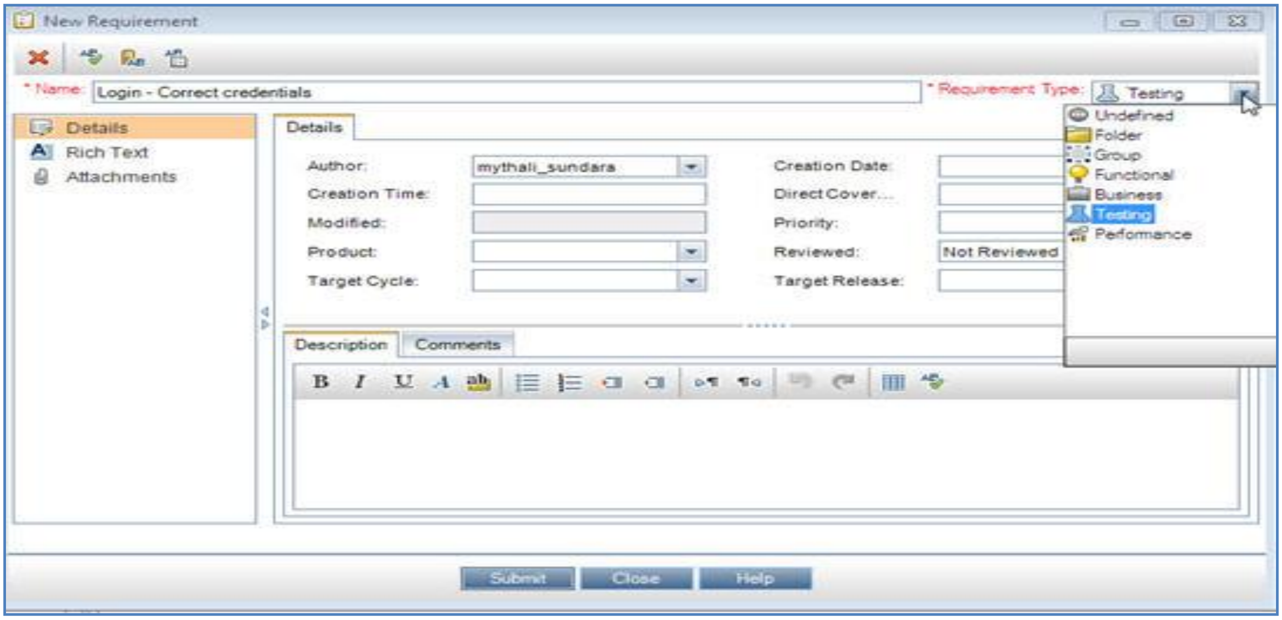
Step #3: Once created, we can choose in the tree folder structure and add additional properties.



No Filter Defined

Name	Direct Cover Status	Author	Req ID
Requirements	---		0
Windows7-64bitTesting	---	tana_ghosh	1
ALM Training	---	mythali_sundara	8
May Release	---	mythali_sundara	9
June Release	---	mythali_sundara	11

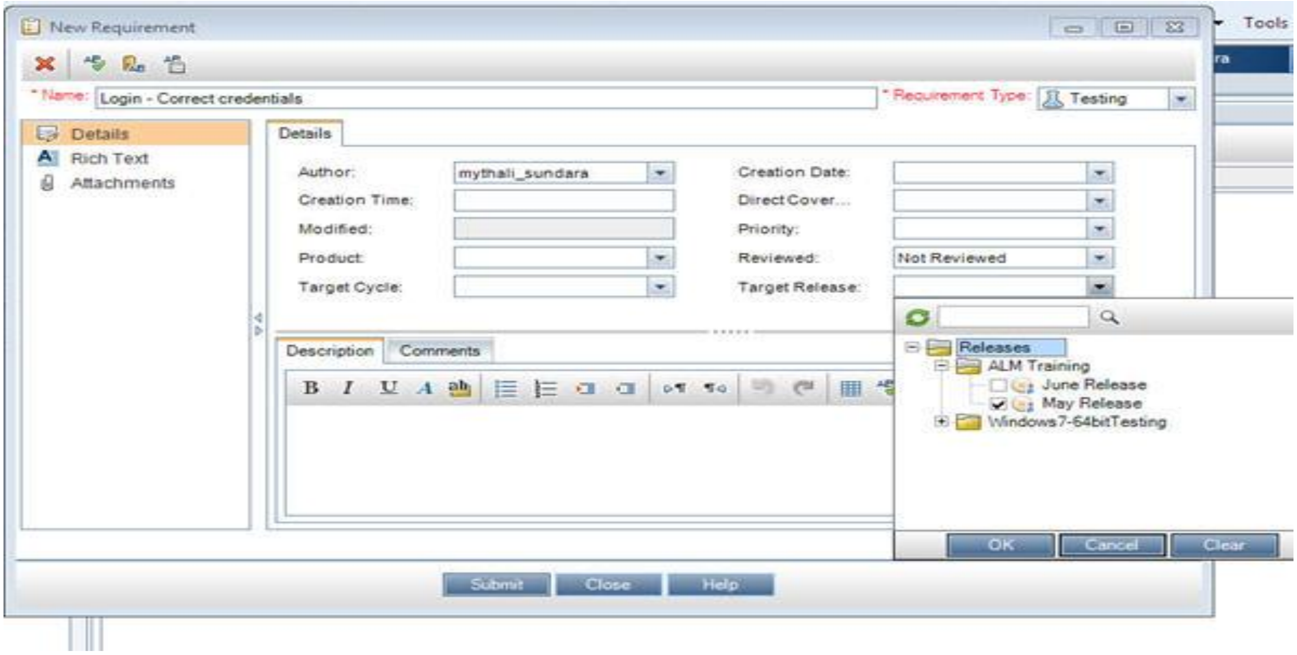
Step #4: Adding Requirements: Now we can add one new requirement, to click on "New Requirement" icon from the menu under folder which we want to add the requirement or Right click on the folder which we want to add then choose "New Requirement" The author name gets auto-populated. Enter the name and choose the relevant requirement type from the drop down. I am going to choose "Testing".



* Name: Login - Correct credentials
* Requirement Type: Testing
Author: mythali_sundara
Reviewed: Not Reviewed

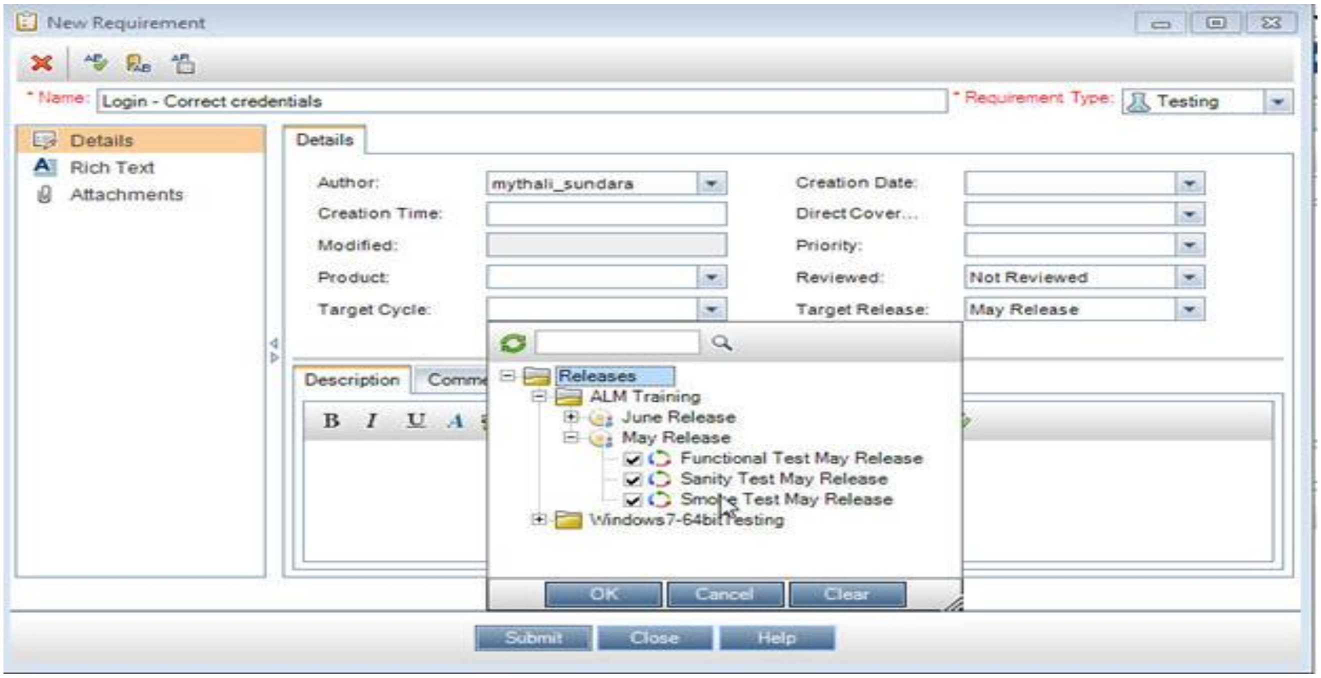
Step #5: Link the requirement to a cycle and release: Once you have added the requirement name and other details, we can now link it to any release and cycles as desired. We can do that by clicking on the "Target Release" drop down and selecting the release information. Since in our example this belong to the may release I am going to choose the same from the dropdown.

Note: You can associate a requirement with more than one release.



* Name: Login - Correct credentials
* Requirement Type: Testing
Author: mythali_sundara
Reviewed: Not Reviewed
☐ June Release
☑ May Release

Step #6: To choose the Cycle. Click on "Target Cycle" drop down and choose the required Cycle. Since our login is a crucial function, I would like to perform it in every cycle. So I am going to map it to all 3 cycles under the May release. Here is how I do it.

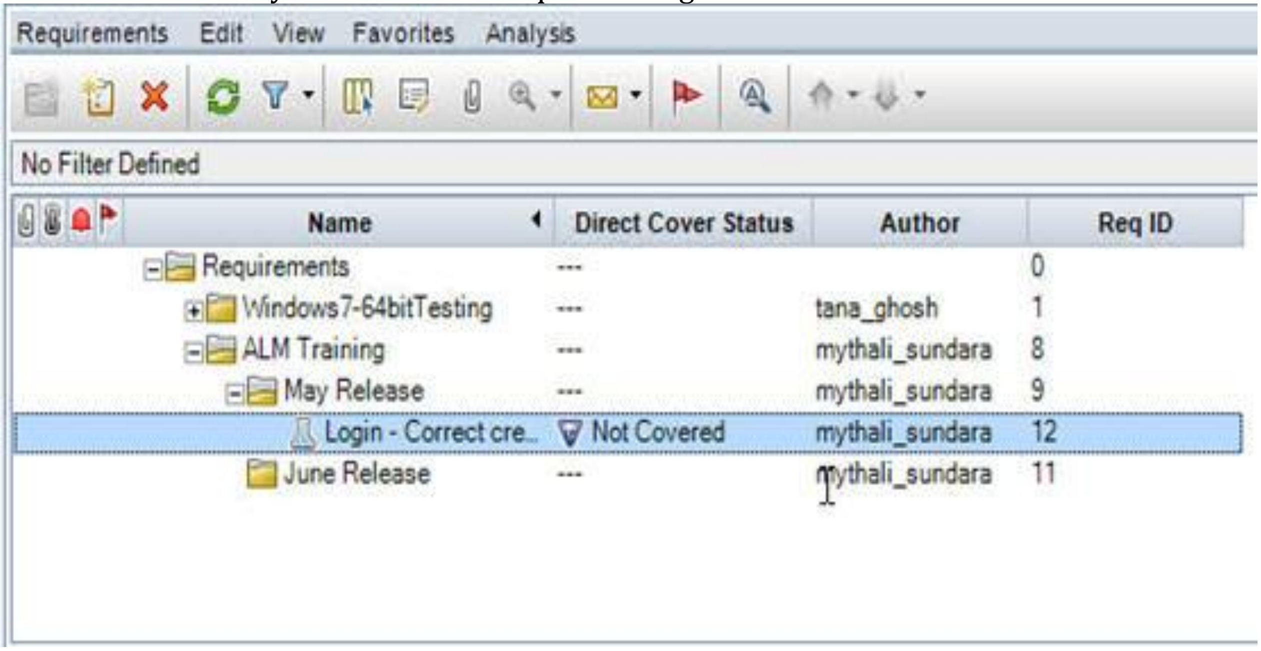


Name: Login - Correct credentials
Requirement Type: Testing

Author:	mythali_sundara	Creation Date:	
Creation Time:		Direct Cover...	
Modified:		Priority:	
Product:		Reviewed:	Not Reviewed
Target Cycle:		Target Release:	May Release

- Releases
  - ALM Training
    - June Release
    - May Release
      - ☑ Functional Test May Release
      - ☑ Sanity Test May Release
      - ☑ Smoke Test May Release
  - Windows7-64bitTesting

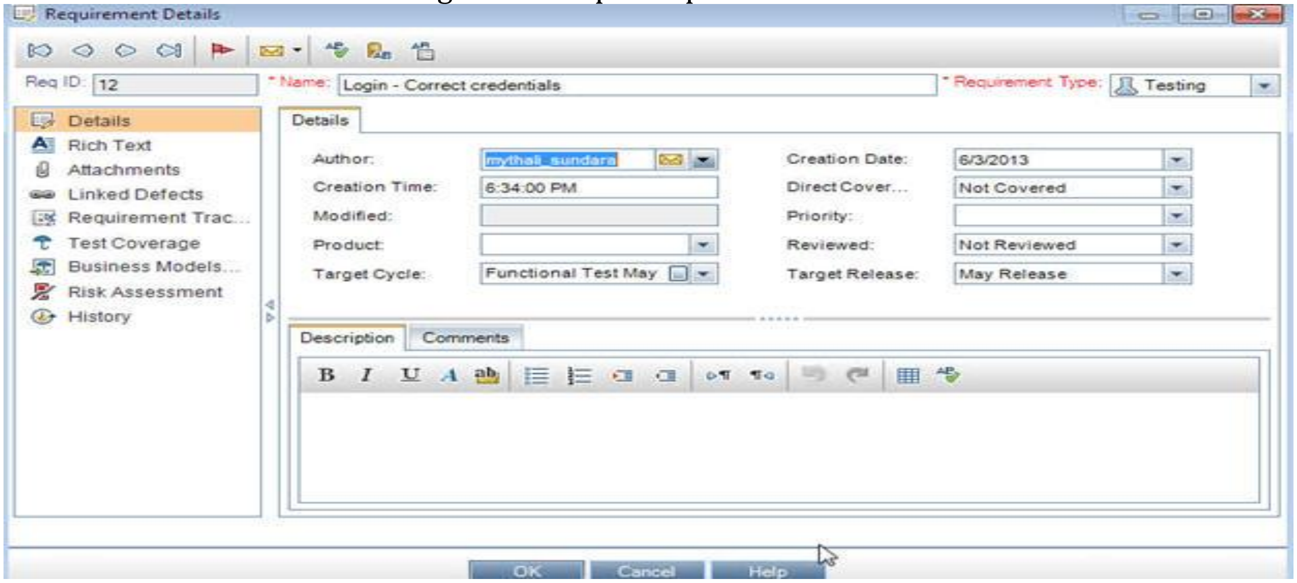
Step #7: we can then assign additional details like the priority, provide attachments and other details before you Submit. The requirement gets added.



No Filter Defined

Name	Direct Cover Status	Author	Req ID
Requirements	---		0
Windows7-64bitTesting	---	tana_ghosh	1
ALM Training	---	mythali_sundara	8
May Release	---	mythali_sundara	9
Login - Correct cre...	Not Covered	mythali_sundara	12
June Release	---	mythali_sundara	11

Step #8: If we need to modify or update the requirement we can do so, by double clicking from the tree view. The following window opens up:



Req ID: 12
* Name: Login - Correct credentials
* Requirement Type: Testing

Author:	mythali_sundara	Creation Date:	6/3/2013
Creation Time:	6:34:00 PM	Direct Cover...	Not Covered
Modified:		Priority:	
Product:		Reviewed:	Not Reviewed
Target Cycle:	Functional Test May	Target Release:	May Release

Step #9: For every requirement a ‘Req. ID” gets auto generated. In this case, it is “7”.

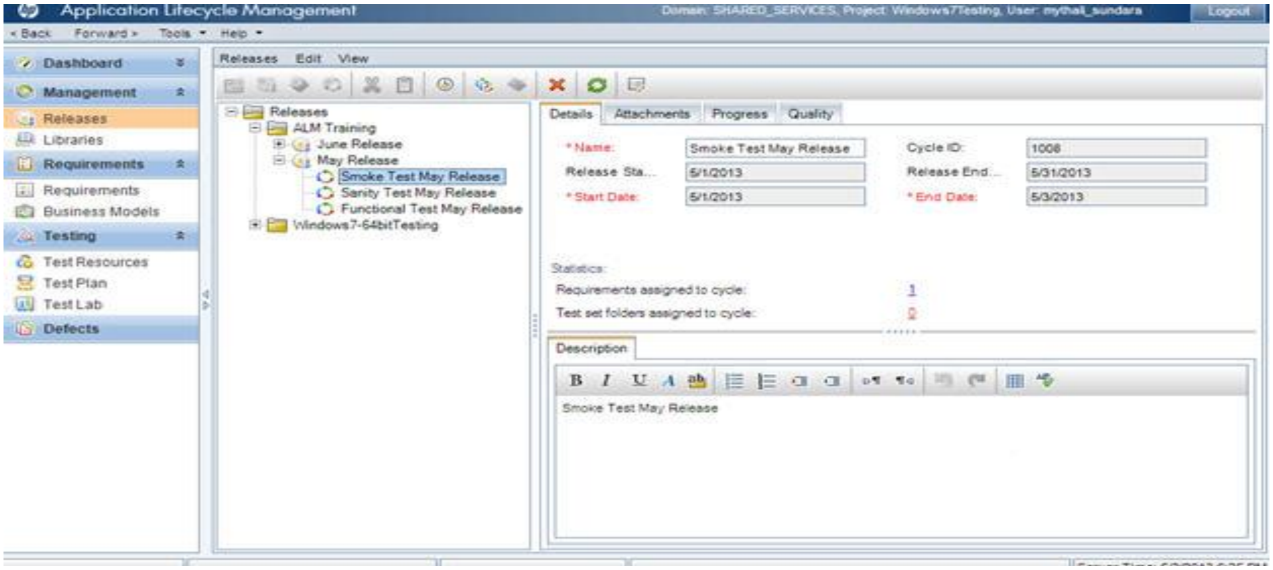
Step #10: Also there are other features like requirement traceability, test coverage, etc in the side bar of the “Requirement Details” dialog. We will discuss all that in details once we finish understanding the end to end flow from releases/cycles-requirements-tests-test sets-defects.

Requirements and Release Statistics:

Now that we created one requirement and linked it with a Release and Cycle, let us see how this linking effect the “Release” and “Cycle” properties for the ones we created earlier.

1) Navigate back to “Management->Releases” from the sidebar.

2) Select one of the cycles we added earlier. Notice the properties:



* Name:	Smoke Test May Release	Cycle ID:	1008
Release Sta...	5/1/2013	Release End...	5/31/2013
* Start Date:	5/1/2013	* End Date:	5/3/2013

Statistics:

Requirements assigned to cycle: 1

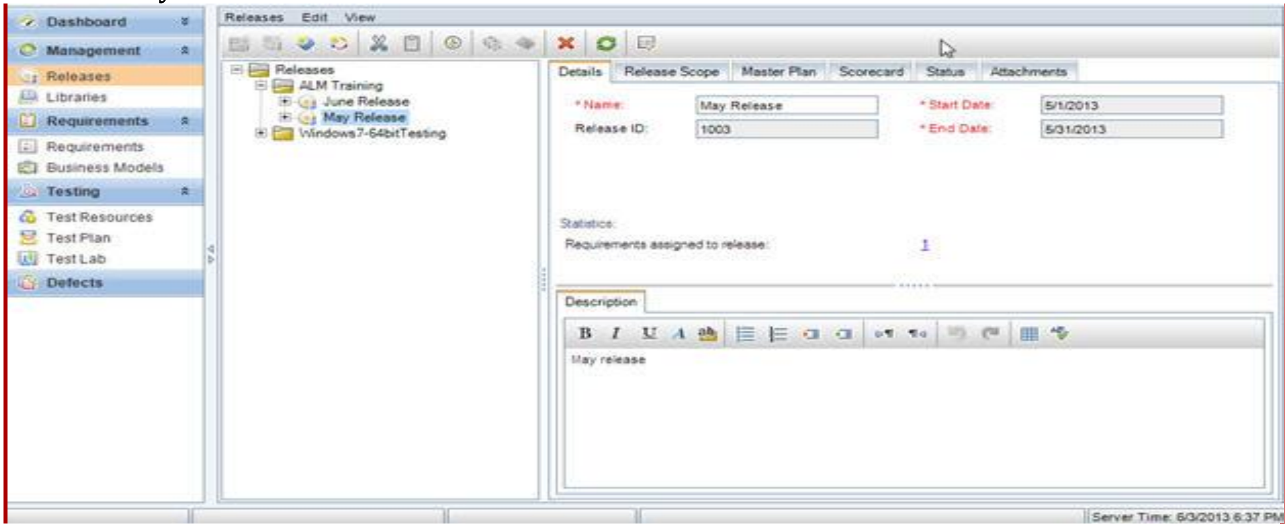
Test set folders assigned to cycle: 0

Description: Smoke Test May Release

Tip: If the statistics don't reflect your changes as soon as you access this page, hit the refresh button in the menu.

1) Notice the statistics section; it shows "1" for the "Requirements assigned to cycle". It now clearly indicates how many requirements are associated to this cycle.

4) Same thing with the Release. Hit the refresh button if the statistics don't get updated immediately.



* Name: May Release
Release ID: 1003
* Start Date: 5/1/2013
* End Date: 5/31/2013

Statistics:
Requirements assigned to release: 1

Description: May release

Server Time: 6/3/2013 6:37 PM

Remember, that ALM does not mandate that a requirement created be mapped back to a release or cycle. It is merely a best practice to do so and when done, you can completely leverage the ALM features to your benefit. The same thing applies with Tests. Once you create the tests you don't necessarily have to link them back to requirement and/or release. But then, you will not be utilizing the tool its 100% if you don't.

Observations:

1) In real time projects, all the data i.e. the requirements and tests are created in either excel sheets or word documents instead of directly creating them in ALM. This is not a tool imposed pre-requisite but testing teams across different companies find this more comfortable. One of the reasons for this is that you can keep local copies of the data on your machine thus reducing the network overhead.

2) You can import the data (requirements or tests) into ALM from Microsoft Word or Excel. To do so, you will need HP ALM Microsoft Word Add-in or HP ALM Microsoft Excel Add-in respectively.

3) You can also integrate ALM with an email server to make sure automatic email alerts can be sent to concerned parties based on certain conditions. For example, when a new defect is logged in or there has been a status change of the defect etc.

4) While creating folders, releases and cycles make sure that you choose the right parent node.

5) ALM organizes and stores data in a Grid or tree form.

Test Plan: The Test Plan module enables us to develop test cases based on testing requirements.

- How to create test cases in Quality Center
- How to link test cases to requirements
- Creating test suites in Quality Center

Creating Test Cases in HP ALM/Quality Center

Gmail features in the May release:

1) Login – with correct credentials

a) Launch Gmail, Enter correct user name, enter correct password and click login
b) Launch Gmail, Enter correct user name, enter correct password, select "Stay signed in" and click login

2) Login – incorrect credentials
a) Launch Gmail, Enter correct username, incorrect password and click Login
b) Launch Gmail, Enter incorrect username, correct password and click Login
c) Launch Gmail, enter incorrect username, incorrect password and click Login.

Example for writing test case for login with correct credentials.

Test Case #	Test Case Description	Step #	Steps to perform	Test Data	Expected Result	Actual Result	Comments
1.a	1.Loing with valid credentials	1	Open browser and enter valid url	url: www.gmail.com	Gmail home page should be displayed		
		2	Enter valid User Name	Username : krishna	The username should be entered into the username filed.		
		3	Enter valid password	Password : qtp@1234	The password should be entered into the password field and characters are hidden as dots		

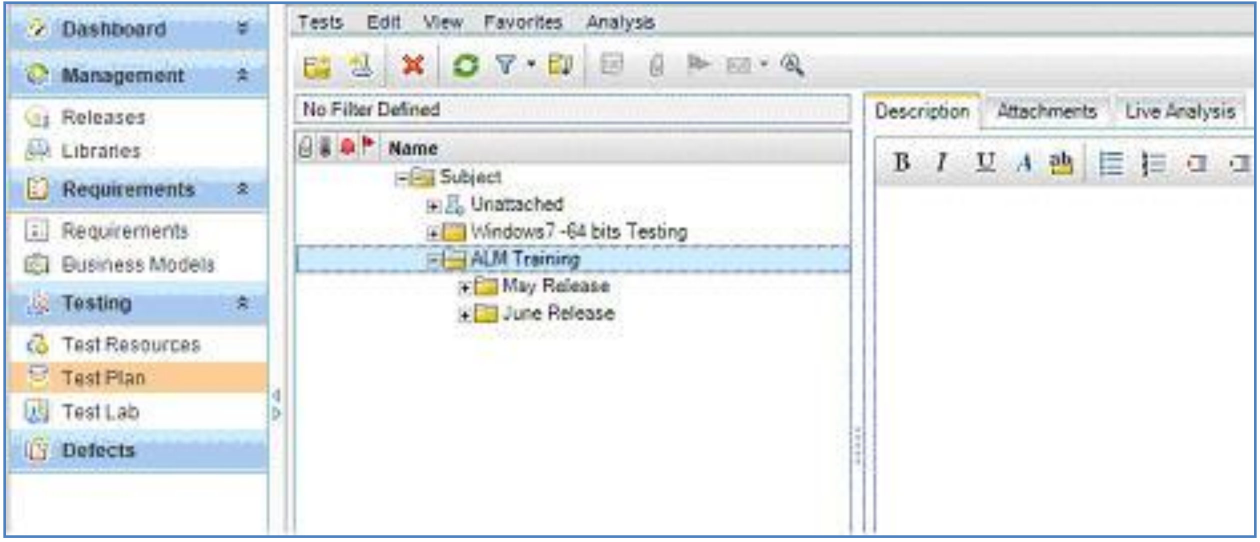
		4	Click sign in button		The user logged into gmail.com page and inbox page should be displayed		

Steps to create test cases under Test Plan tab:

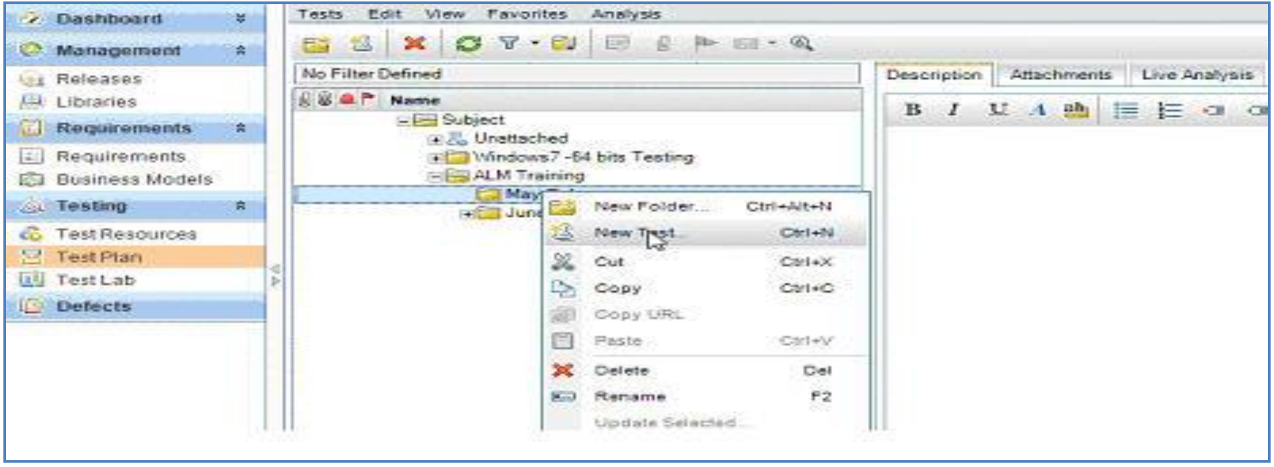
Step #1: Login to ALM into the right project. Create the release, cycles and requirements as described in the previous tutorials.

Step #2: Go to the Test Plan tab by choosing “Testing->Test Plan” from the side bar.

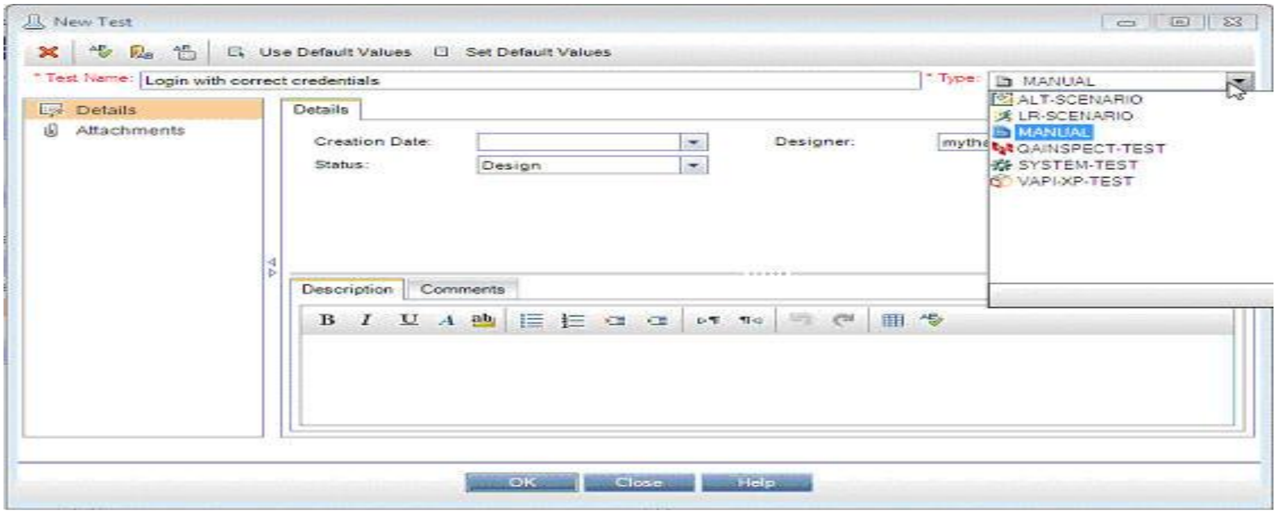
Step #3: Choose “Subject” as your home folder and create a sub folder “ALM training” under it. I am going to create “May Release” and “June Release” Sub Folders under it.



Step #4: Go to May Release folder and choose the option to create a new test.



Step #5: Enter name and then choose the “type”. Choose “manual” for our tutorial.



* Test Name: Login with correct credentials

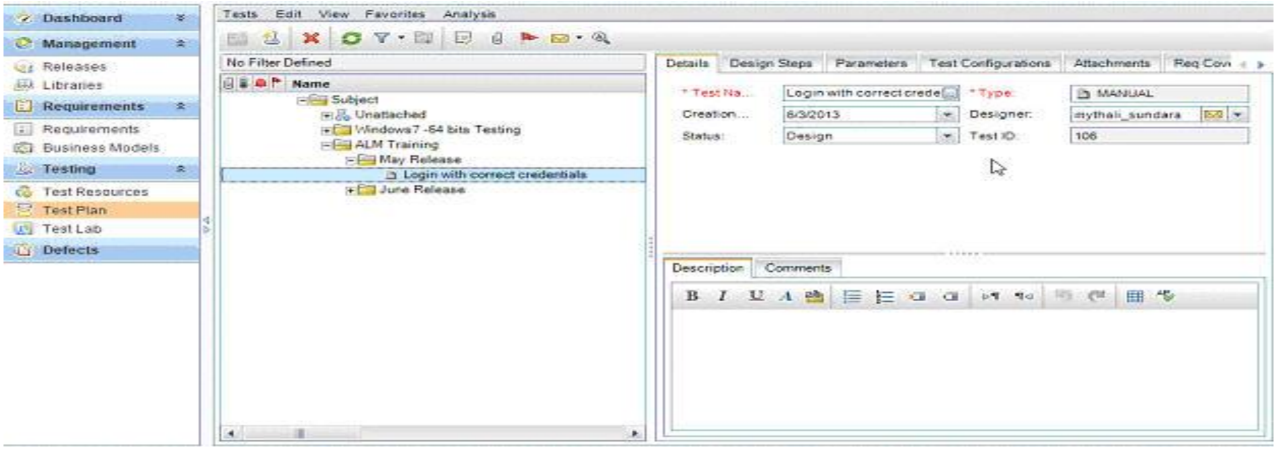
* Type: MANUAL

Creation Date:

Status: Design

Designer: mythe

Step #6: Enter the other details. The designer name will be auto populated based on your login credentials. Click OK. The test gets added.



* Test Na...: Login with correct crede

* Type: MANUAL

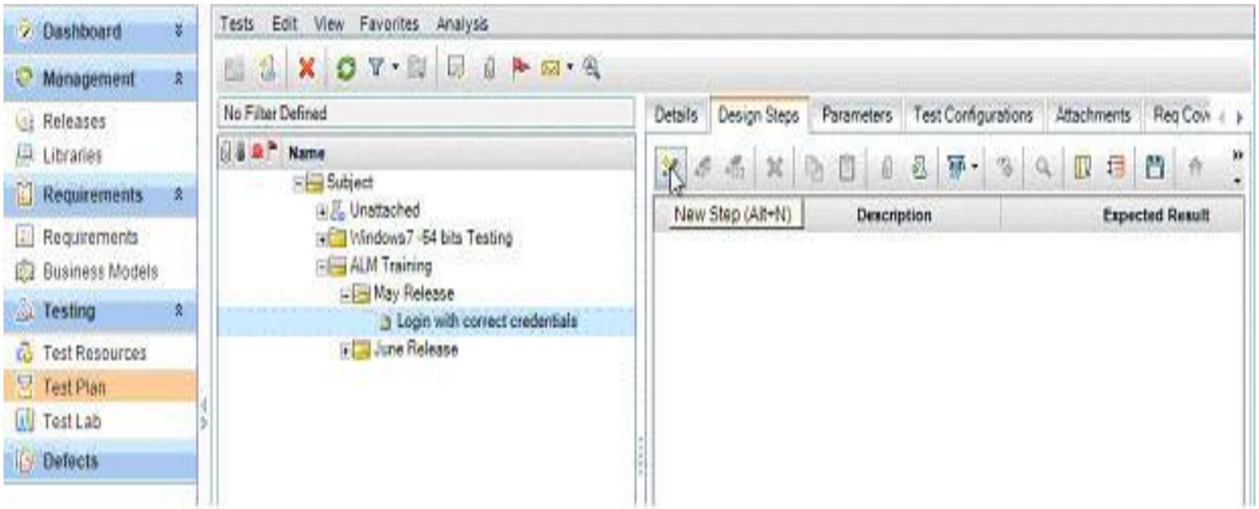
Creation...: 6/3/2013

Designer: mythali_sundara

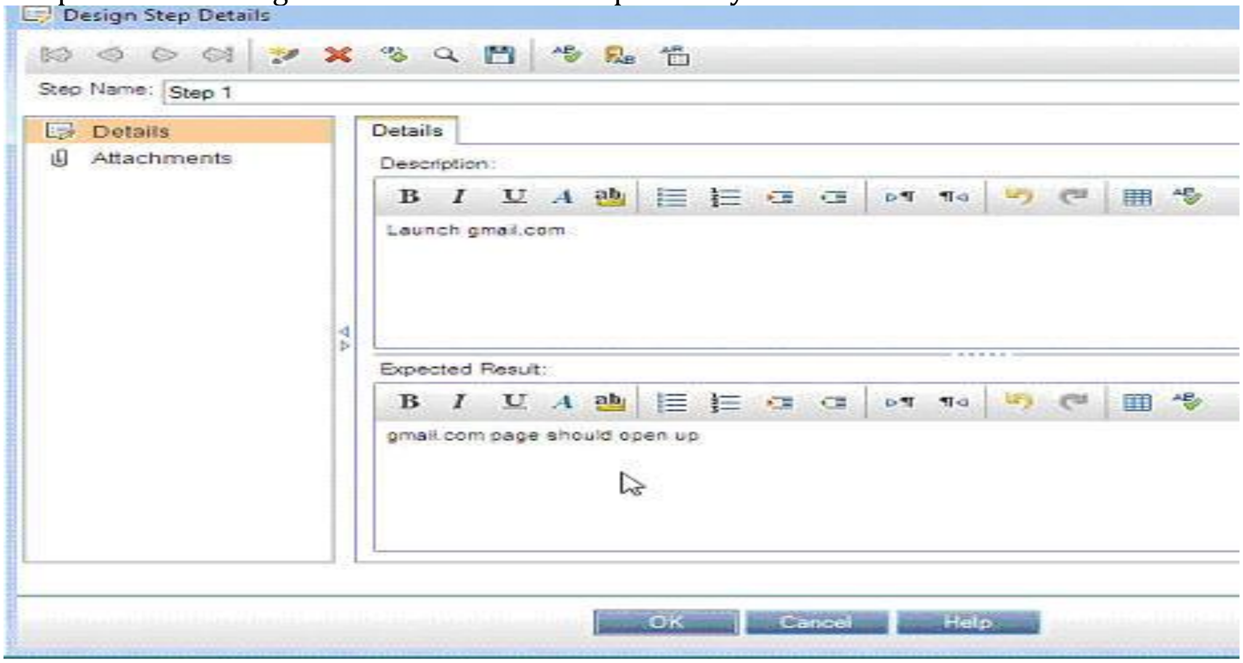
Status: Design

Test ID: 106

Step #7: Now you can add your steps. Click on the “Design Steps” tab. Click on “New Step” icon.



Step #8: Enter the step details. The description and Expected Results fields come up complete text editing features that are self explanatory.

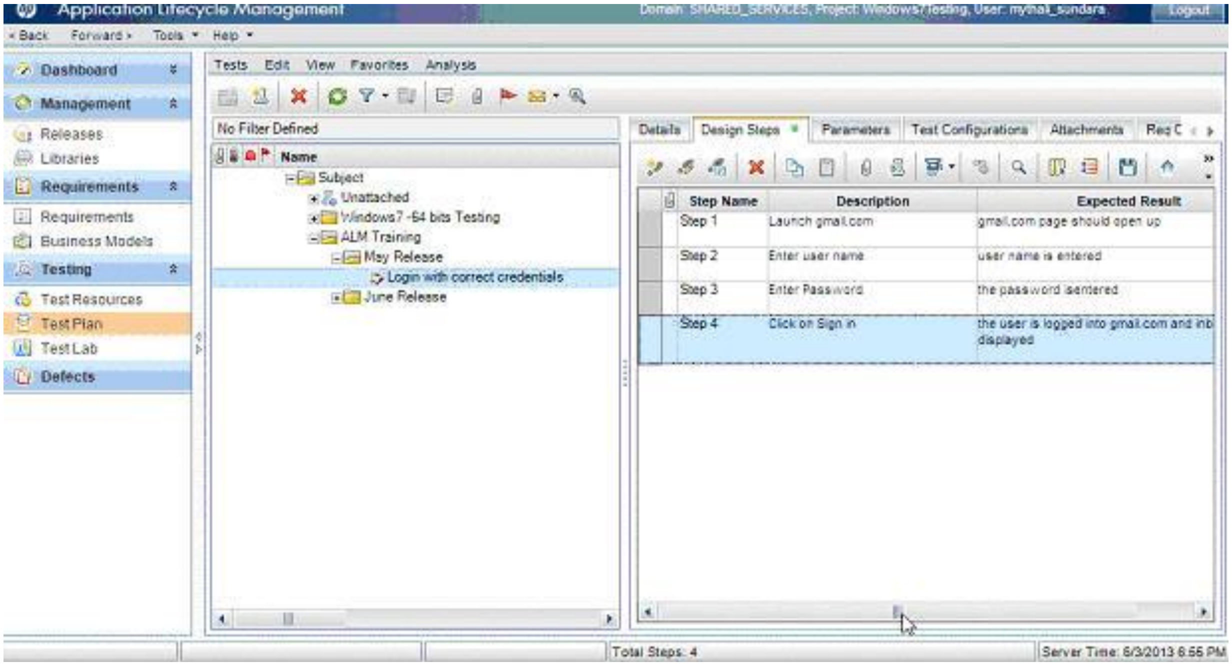


Step Name: Step 1

Description: Launch gmail.com

Expected Result: gmail.com page should open up

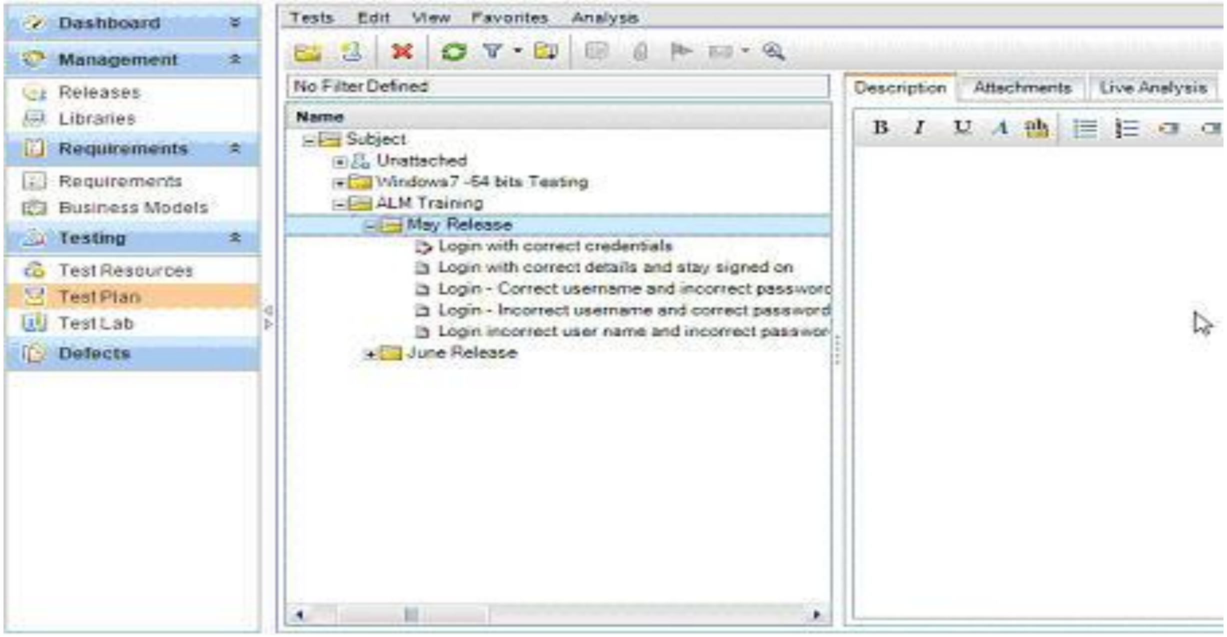
Step #9: I am going to create all the steps as shows above. This is how the completed test case looks like:



Step Name	Description	Expected Result
Step 1	Launch gmail.com	gmail.com page should open up
Step 2	Enter user name	user name is entered
Step 3	Enter Password	the password isentered
Step 4	Click on Sign in	the user is logged into gmail.com and inb displayed

This completes the process of adding test cases and steps to them.

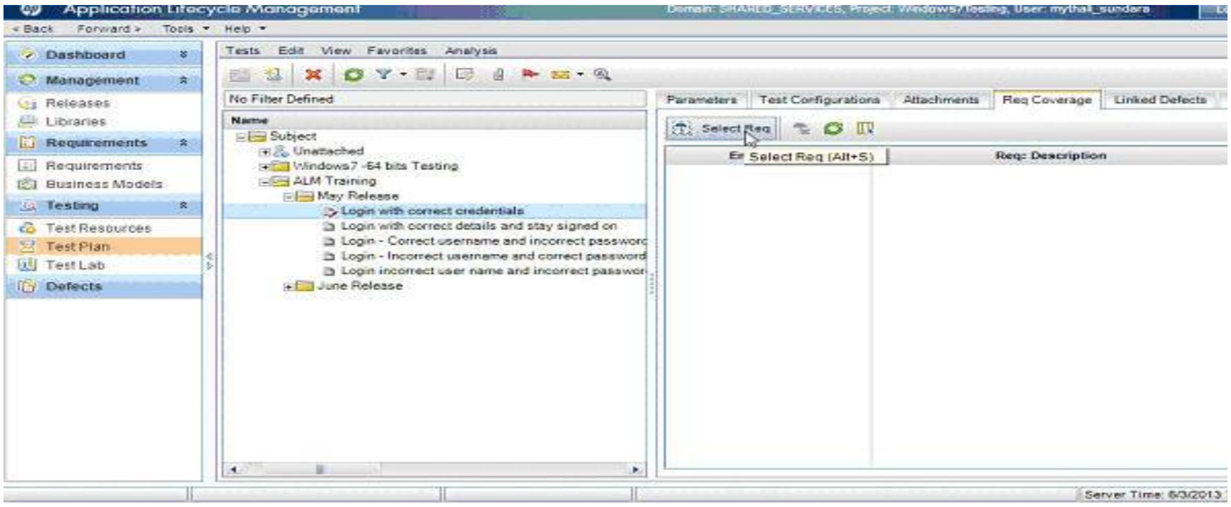
Step #10: Under May release I am going to add some more test cases.



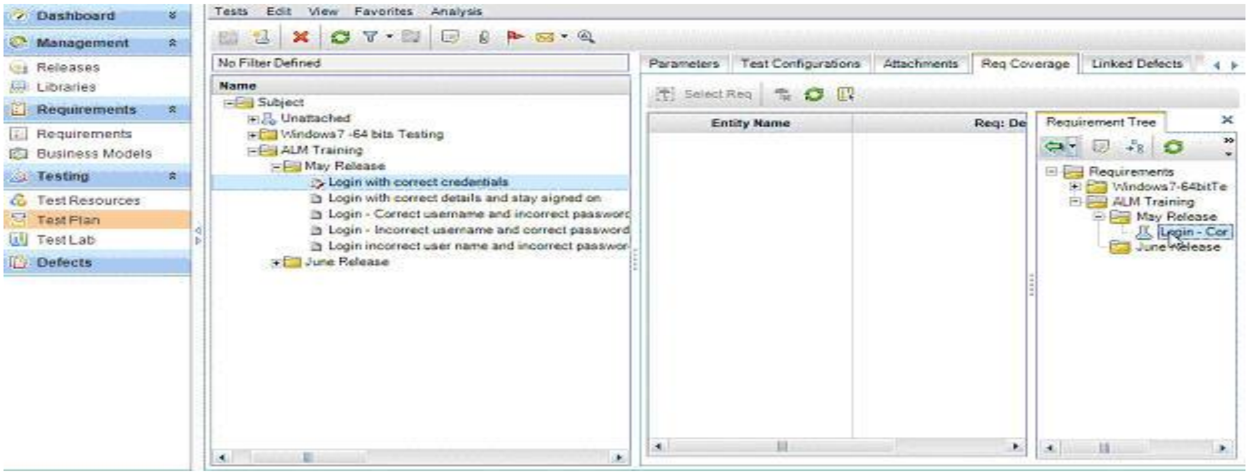
Linking Test Cases to Requirements:

Steps to link test cases and requirements:

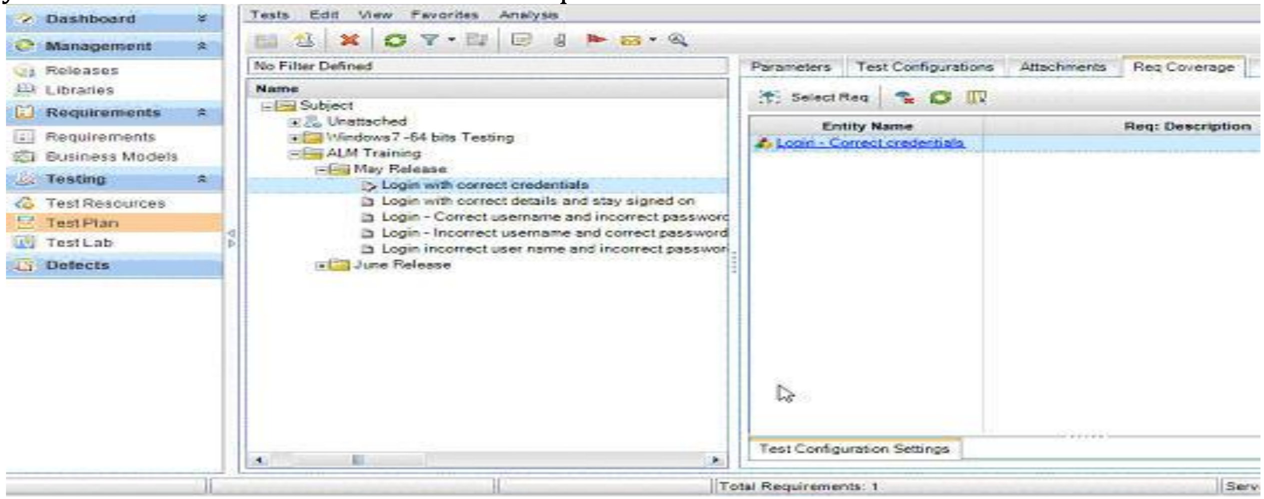
1) Select a test case created and click on it. All the properties get displayed in the right hand side tab. Go to “Req coverage” tab and click on “Select Req”



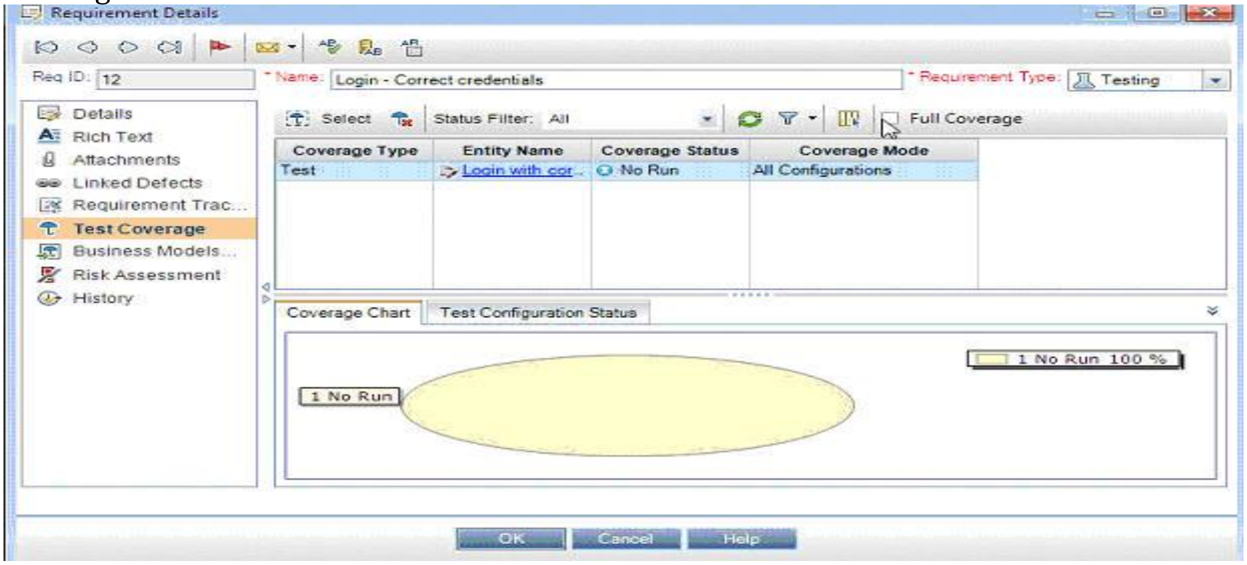
2) The requirements tree gets displayed on the side. Expand the tree and select the needed requirements.



3) Once done, close the requirement tree. You can link a test case to as many requirements as you would like. This is how the added requirement looks.


Dashboard
Management
Releases
Libraries
Requirements
Requirements
Business Models
Testing
Test Resources
Test Plan
Test Lab
Defects
Tests Edit View Favorites Analysis
No Filter Defined
Name
Subject
Unattached
Windows7 -64 bits Testing
ALM Training
May Release
Login with correct credentials
Login with correct details and stay signed on
Login - Correct username and incorrect password
Login - Incorrect username and correct password
Login incorrect user name and incorrect password
June Release
Parameters
Test Configurations
Attachments
Req Coverage
Select Req
Entity Name
Req: Description
Login - Correct credentials
Test Configuration Settings
Total Requirements: 1

4) Let us now check, how this mapping effects the requirement. Go to Requirements tab from the side bar. Double click on the requirement that you just mapped and notice the “Test Coverage” details:



Req ID: 12

* Name: Login - Correct credentials

* Requirement Type: Testing

Status Filter: All

Coverage Type	Entity Name	Coverage Status	Coverage Mode
Test	Login with cor...	No Run	All Configurations

You see how the test details and it’s the test’s status is displayed. Since this test was just created and never run, the coverage status shows as “No Run”. Let us now move on in our tutorial and learn how to run a test.

“Test Lab” tab in ALM:

Test Lab: The Test Lab module enables us To run tests on the application and analyze the results.

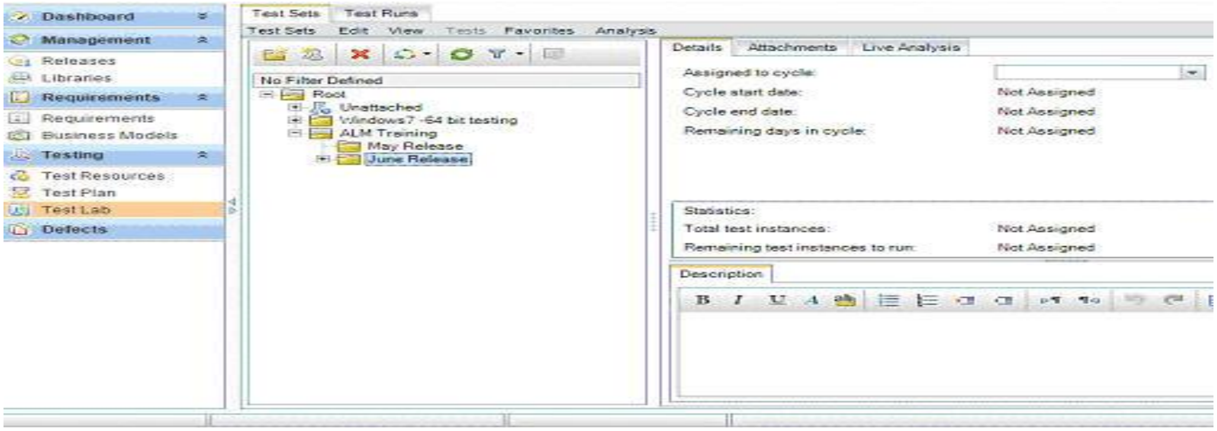
Imagine we have just begun the May release testing phase. The first cycle is Smoke test. We are not going to execute all the test cases we have.

- Briefly, a smoke test is a high level test performed by the QA team on an AUT as soon as the code is deployed to make sure the application did not break. So we are only going to execute test cases 1-a and 2-c.
- Similarly for sanity testing, which is checking the key functionality of the AUT? We will execute 1-a, 1-b and 2-e
- Functional testing is everything.

Test lab tab will help us create test sets that contain the test cases that we need to execute in each phase. This is where the tester can execute the tests and record the test results. Let’s see how.

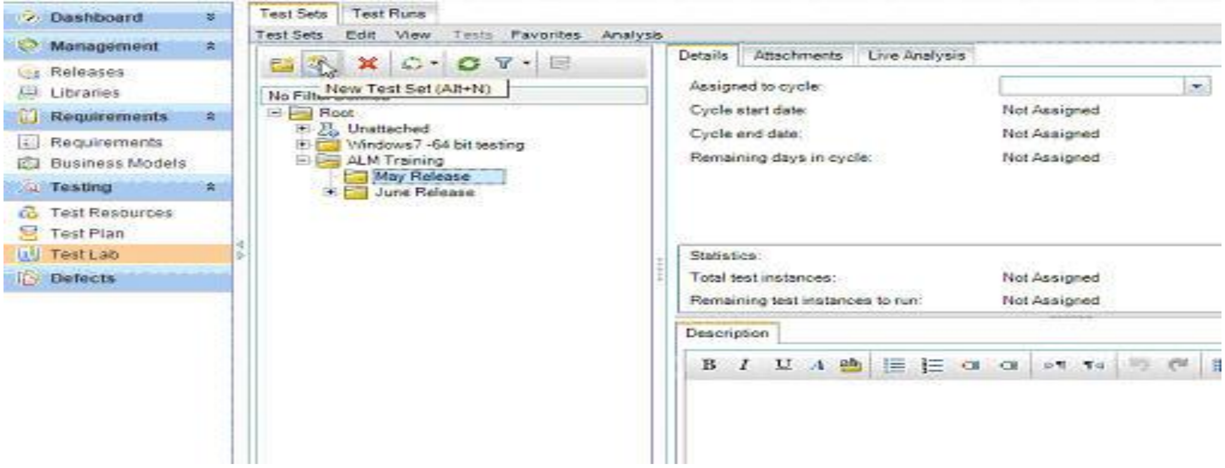
Steps to create test suites in Quality Center:

Step #1: Go to Test lab tab by navigating from the side bar. Create the folders as shown below:



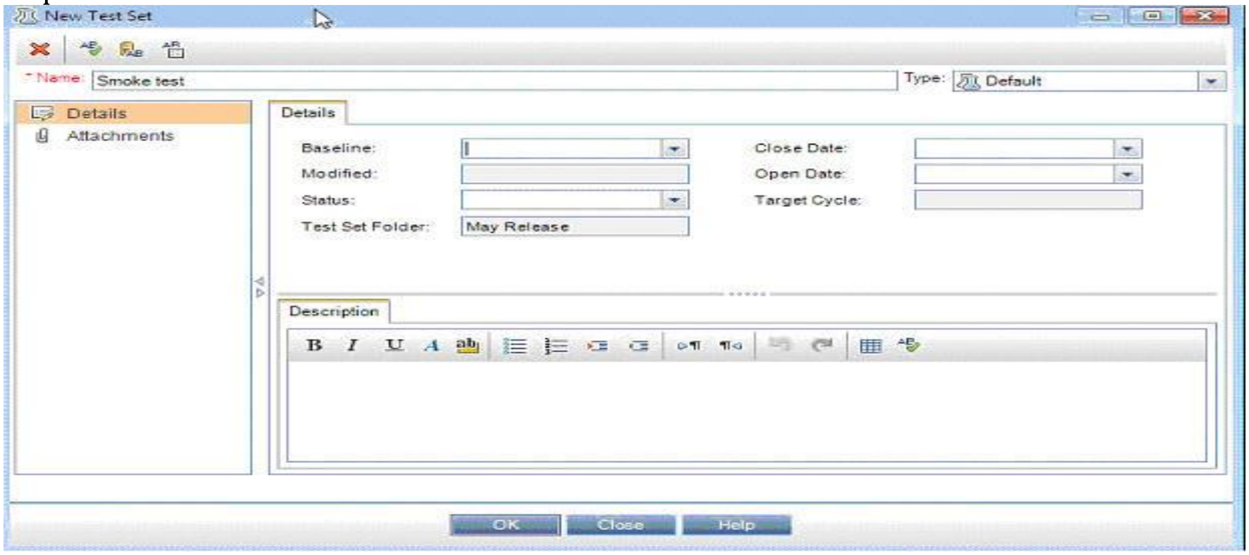
Assigned to cycle:
Cycle start date: Not Assigned
Cycle end date: Not Assigned
Remaining days in cycle: Not Assigned
Statistics:
Total test instances: Not Assigned
Remaining test instances to run: Not Assigned

Step #2: Under May release, choose the option to create a new test set



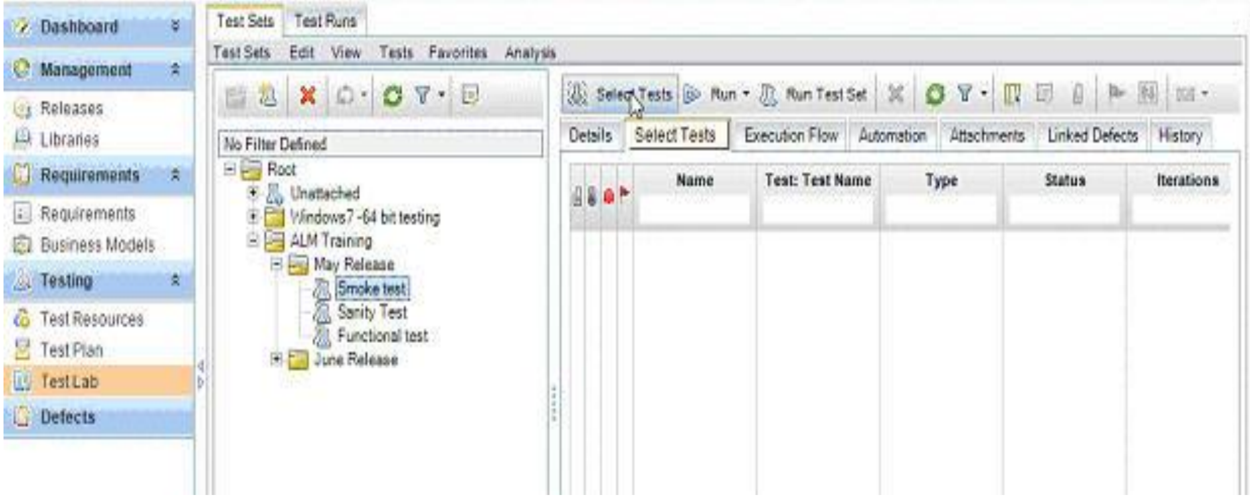
Assigned to cycle:
Cycle start date: Not Assigned
Cycle end date: Not Assigned
Remaining days in cycle: Not Assigned
Statistics:
Total test instances: Not Assigned
Remaining test instances to run: Not Assigned

Step #3: Enter the test set name. Click OK.

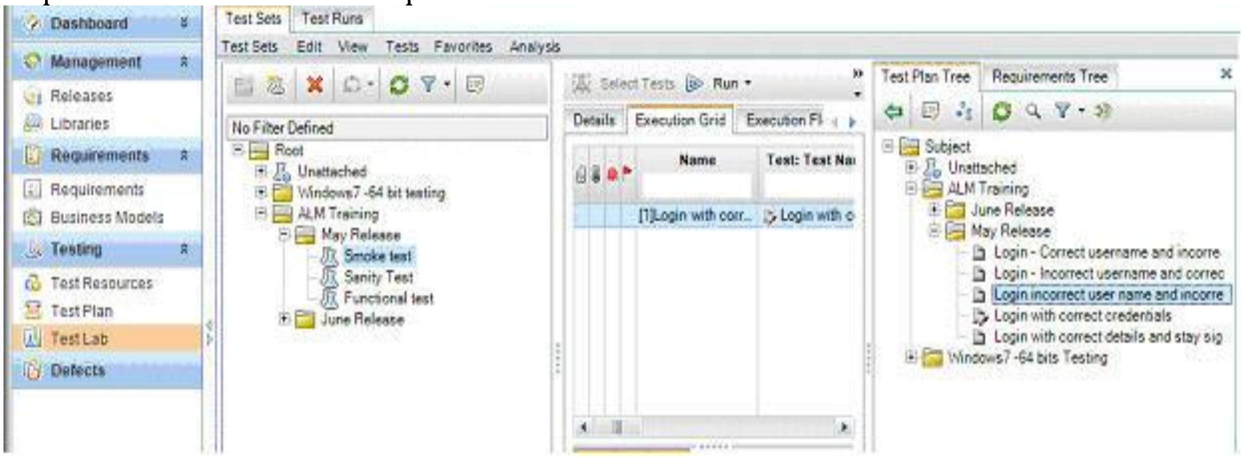


* Name: Smoke test
Type: Default
Baseline:
Modified:
Status:
Test Set Folder: May Release
Close Date:
Open Date:
Target Cycle:

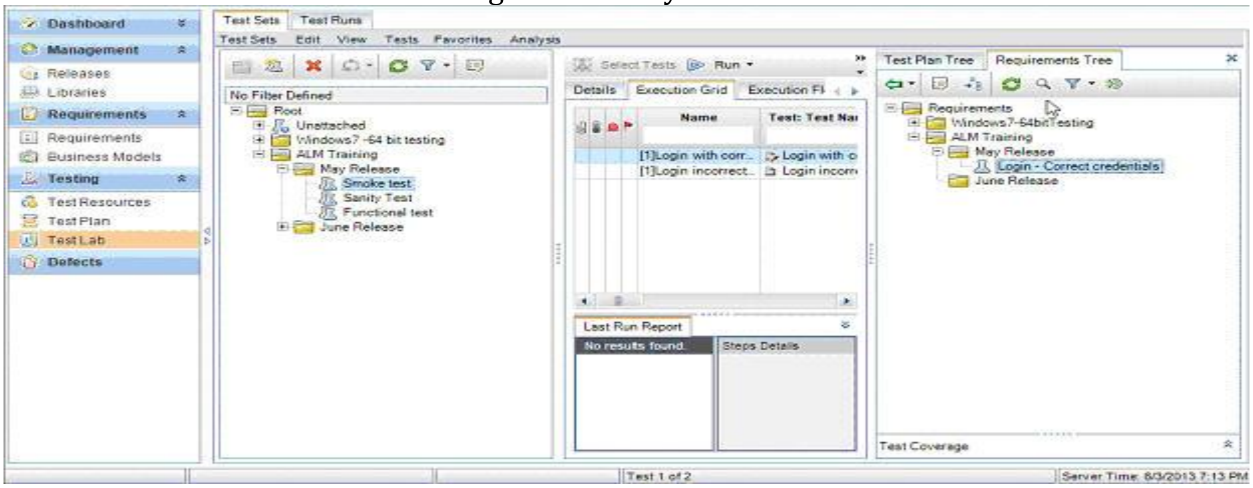
Step #4: Once it is created. Click on “Select Tests” from the menu



Step #5: Select the tests as required



Step #6: Alternately if you choose the “Requirements Tree” tab, you can choose requirement and all the tests that are linked to it get added to your test set.



Step #7: Go ahead and create all the test sets. As you can see from the above screen shot I have created a test set each for each cycle.
Step #8: Tip: You can choose to add the same test multiple times in a test case. In that case, the second instance of the test will appear with a prefix [2].

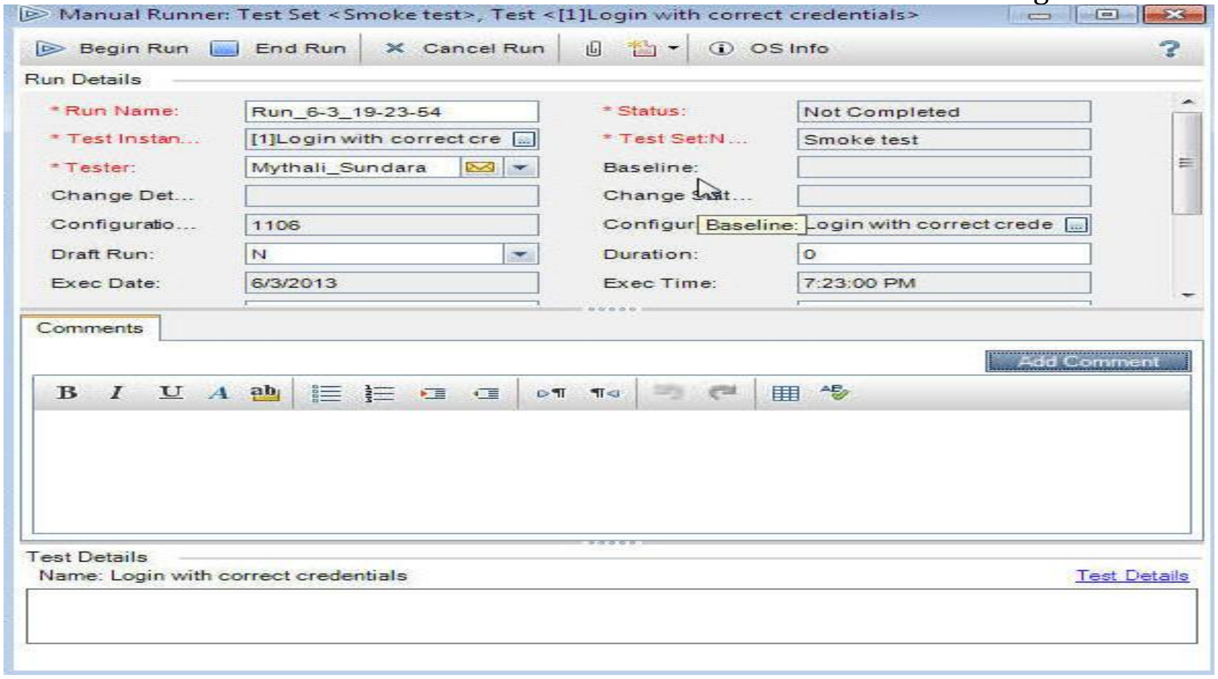
Points to Note:

1) If you observe, we have created our releases, requirements, tests etc all under the parent folder "ALM training". There is no rule that you should do that. You can choose a different name for your folder in each section. But as you have seen, it makes so much sense to use the same folder name. For a real time project that you are working on and trying to use ALM for, try to come up with a name that you would want to consistently use across as the first step for your test management process.

2) The columns in the test plan tab or any other tabs can be customized by your project's ALM admin.

How to Execute Test Cases Using ALM/QC

Step #1: In the Test lab tab, choose the test set that you would like to run and click on "Run Test" or "Run Test Set". Run Test- will execute the test set selected and the "Run test set" will run the entire set one test after the other until the end. Click on "Being Run".



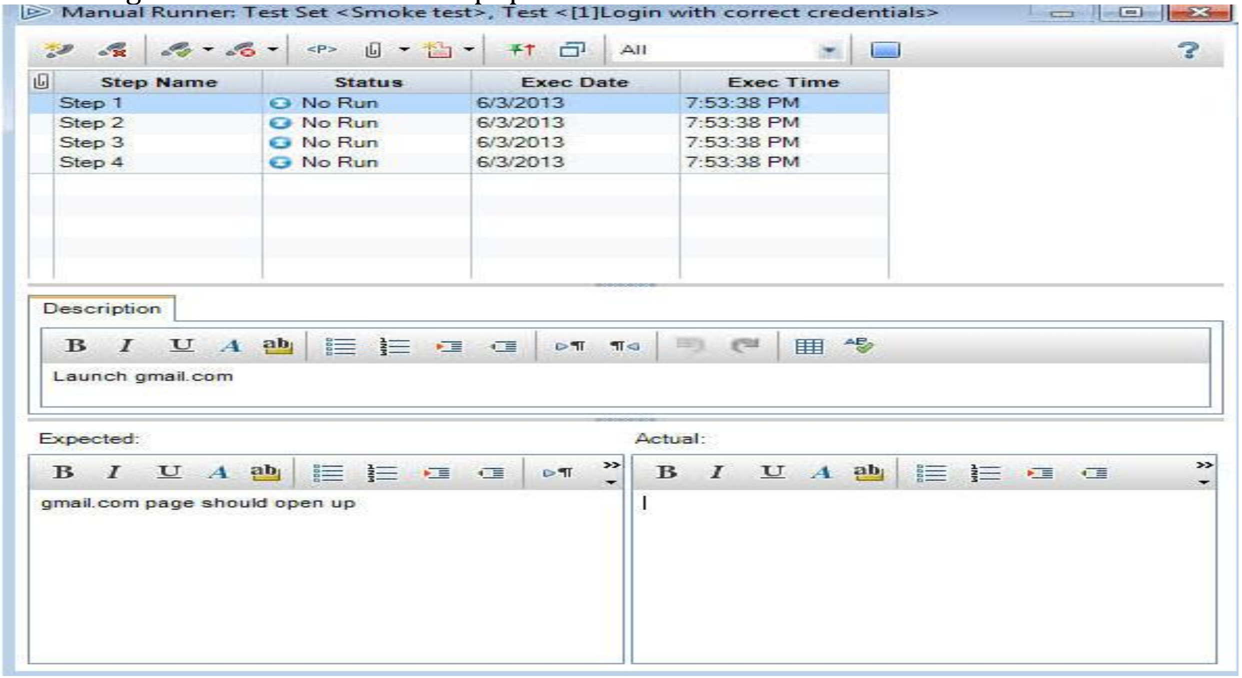
Manual Runner: Test Set <Smoke test>, Test <[1]Login with correct credentials>

Run Details

* Run Name:	Run_6-3_19-23-54	* Status:	Not Completed
* Test Instan...	[1]Login with correct cre	* Test Set:N...	Smoke test
* Tester:	Mythali_Sundara	Baseline:	
Change Det...		Change Set...	
Configuratio...	1106	Configur Baseline:	Login with correct crede
Draft Run:	N	Duration:	0
Exec Date:	6/3/2013	Exec Time:	7:23:00 PM

Test Details
Name: Login with correct credentials

Step #2: If it is an automation test and the tool is integrated, then it launches and runs it in the testing tool. In that case it auto populates the test result.



Manual Runner: Test Set <Smoke test>, Test <[1]Login with correct credentials>

Step Name	Status	Exec Date	Exec Time
Step 1	No Run	6/3/2013	7:53:38 PM
Step 2	No Run	6/3/2013	7:53:38 PM
Step 3	No Run	6/3/2013	7:53:38 PM
Step 4	No Run	6/3/2013	7:53:38 PM

Description

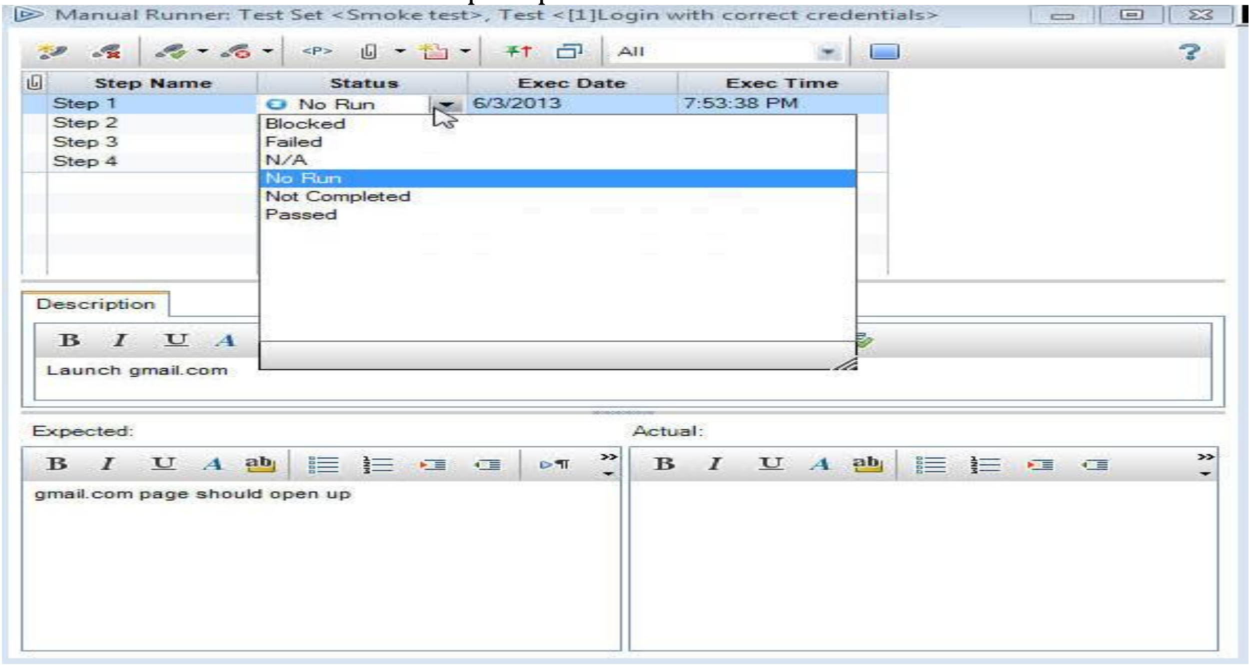
Launch gmail.com

Expected:

gmail.com page should open up

Actual:

Step #3: Since ours is a manual test, we will have to execute the steps manually on our AUT and set the results. Go to the test status field and click on it to set it to a certain value. You can also enter the actual result in the space provided.



Manual Runner: Test Set <Smoke test>, Test <[1]Login with correct credentials>

Step Name	Status	Exec Date	Exec Time
Step 1	No Run	6/3/2013	7:53:38 PM
Step 2			
Step 3			
Step 4			

Description

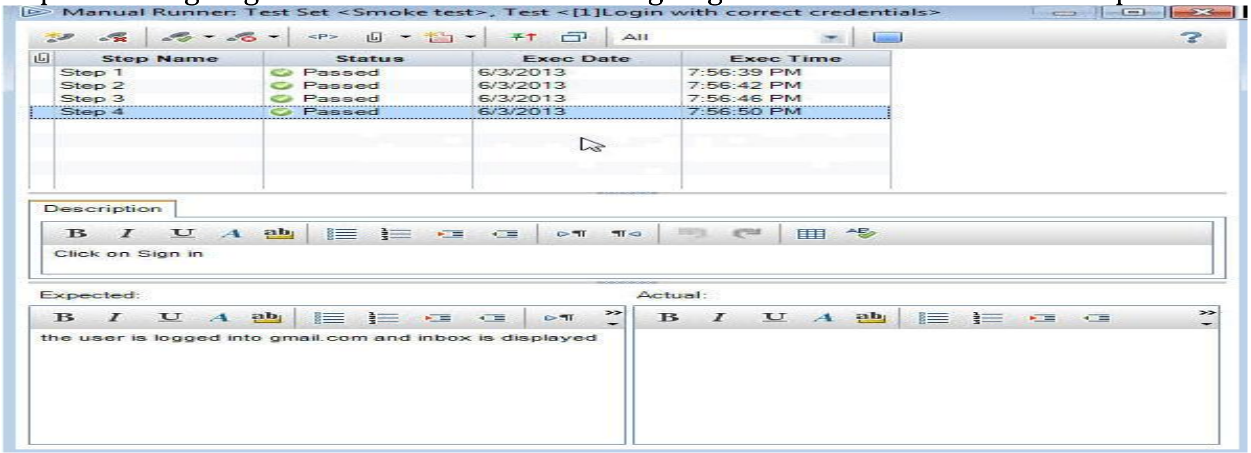
Launch gmail.com

Expected:

gmail.com page should open up

Actual:

Step #4: I am going to choose “Passed”. And am going to do the same for all the steps.



Manual Runner: Test Set <Smoke test>, Test <[1]Login with correct credentials>

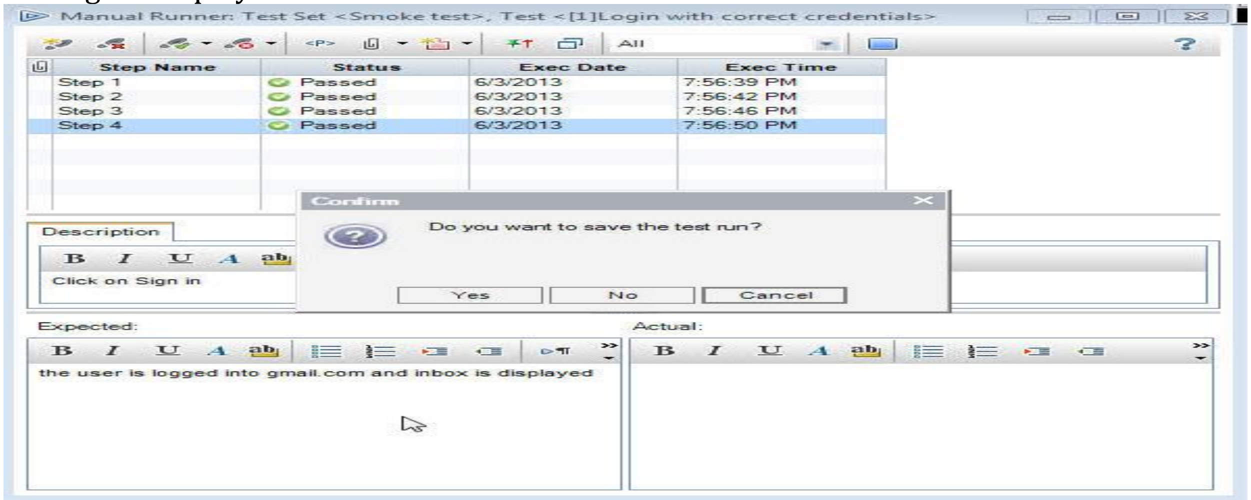
Step Name	Status	Exec Date	Exec Time
Step 1	Passed	6/3/2013	7:56:39 PM
Step 2	Passed	6/3/2013	7:56:42 PM
Step 3	Passed	6/3/2013	7:56:46 PM
Step 4	Passed	6/3/2013	7:56:50 PM

Description:
Click on Sign in

Expected:
the user is logged into gmail.com and inbox is displayed

Actual:

Step #5: When done, click on the cross button to the dialog. The following confirmation message is displayed. Click Yes



Manual Runner: Test Set <Smoke test>, Test <[1]Login with correct credentials>

Step Name	Status	Exec Date	Exec Time
Step 1	Passed	6/3/2013	7:56:39 PM
Step 2	Passed	6/3/2013	7:56:42 PM
Step 3	Passed	6/3/2013	7:56:46 PM
Step 4	Passed	6/3/2013	7:56:50 PM

Confirm

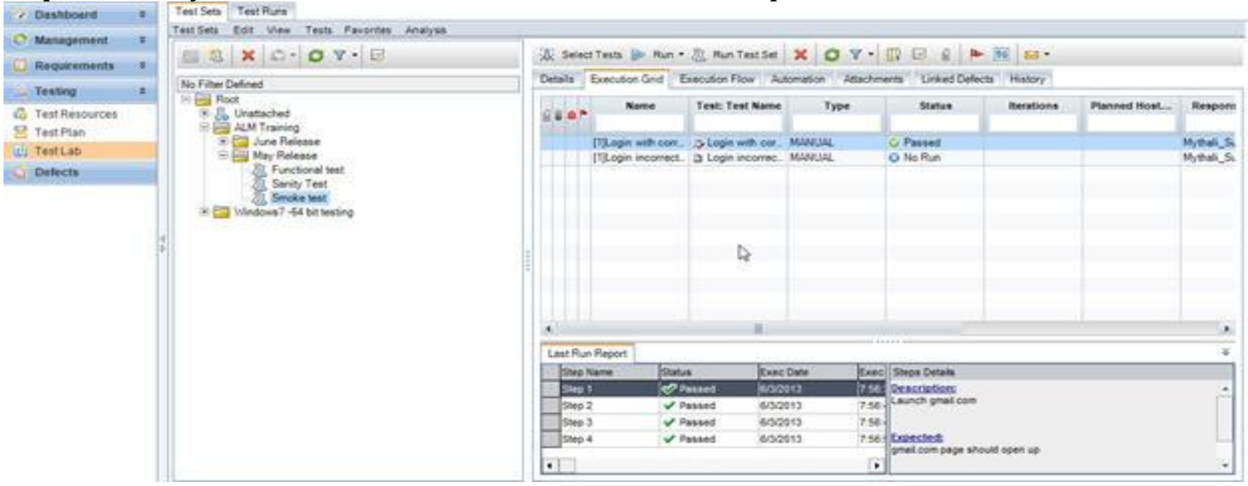
Do you want to save the test run?

Description:
Click on Sign in

Expected:
the user is logged into gmail.com and inbox is displayed

Actual:

Step #6: Now you see, the status of the test is marked as passed.



Name	Test: Test Name	Type	Status	Iterations	Planned Host...	Respon:
[1]Login with corr...	Login with cor...	MANUAL	Passed			Mythali_S.
[1]Login incorrect...	Login incorrec...	MANUAL	No Run			Mythali_S.

Last Run Report

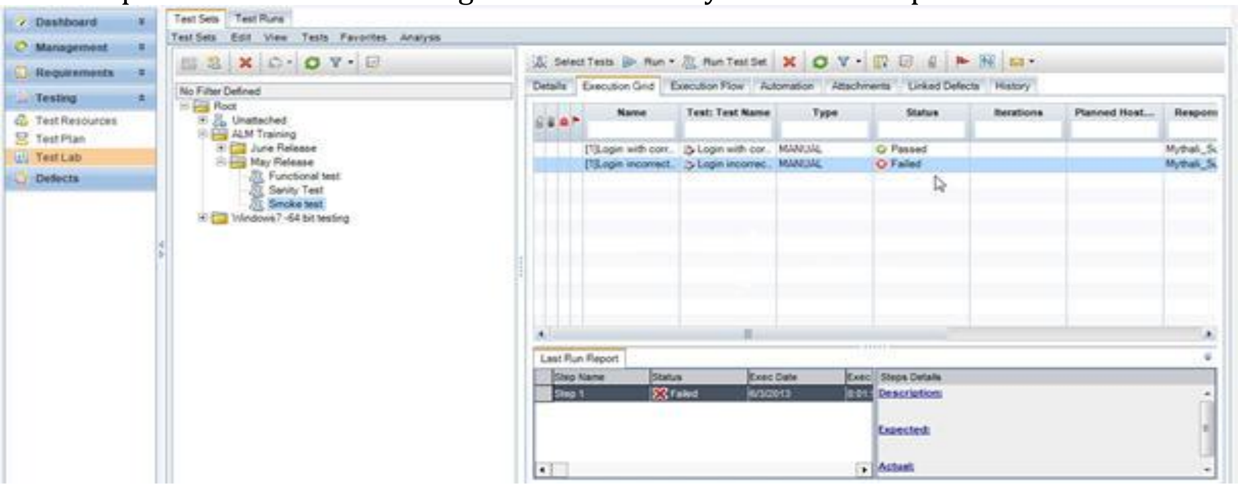
Step Name	Status	Exec Date	Exec
Step 1	Passed	6/3/2013	7:56:
Step 2	Passed	6/3/2013	7:56:
Step 3	Passed	6/3/2013	7:56:
Step 4	Passed	6/3/2013	7:56:

Steps Details

Description:
Launch gmail.com

Expected:
gmail.com page should open up

Step #7: Let us execute the next test in the list. I will follow the same steps and fail just one of the steps. The status of the test goes to failed if any one of the steps fail.



Name	Test: Test Name	Type	Status	Iterations	Planned Host...	Respons
[1]Login with corr...	Login with cor...	MANUAL	Passed			Mythali_S
[1]Login incorrect...	Login incorrec...	MANUAL	Failed			Mythali_S

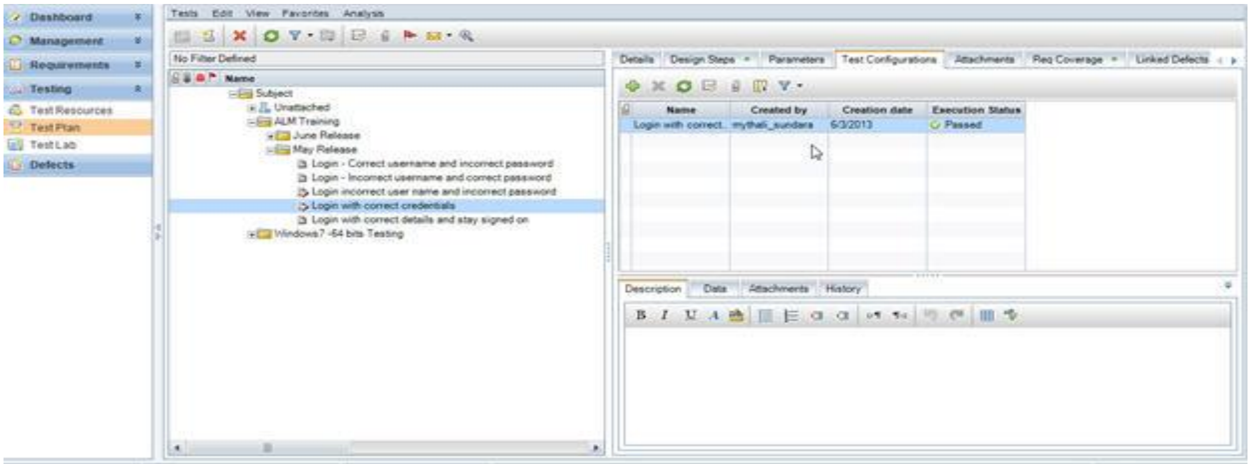
Step Name	Status	Exec Date	Exec
Step 1	Failed	6/3/2013	8:01

Steps Details
Description:
Expected:
Actual:

Tip: You can change this status any time you wish by clicking on the status column and setting the desired value.

Check the execution status in the Test Plan tab:

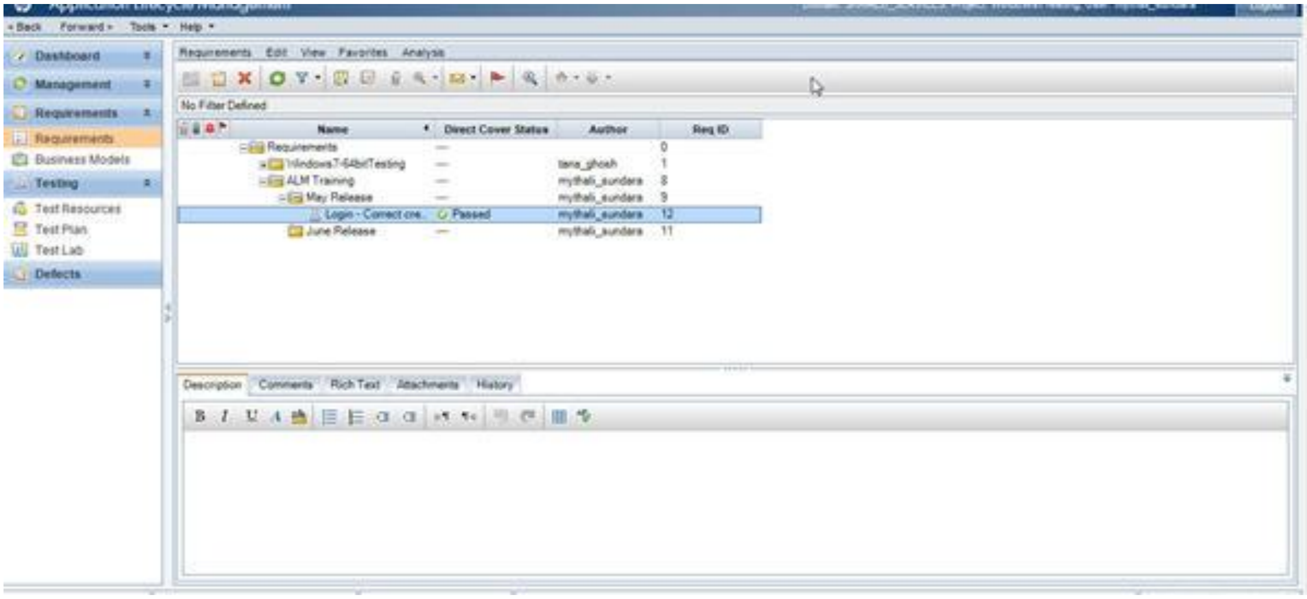
Once you have executed the test case, go back to the test plan tab and open the corresponding test case. Go to Test Configurations tab and the following is how the execution status gets reflected.



Name	Created by	Creation date	Execution Status
Login with correct...	mythali_sundara	6/3/2013	Passed

Check the execution status in the Requirements tab:

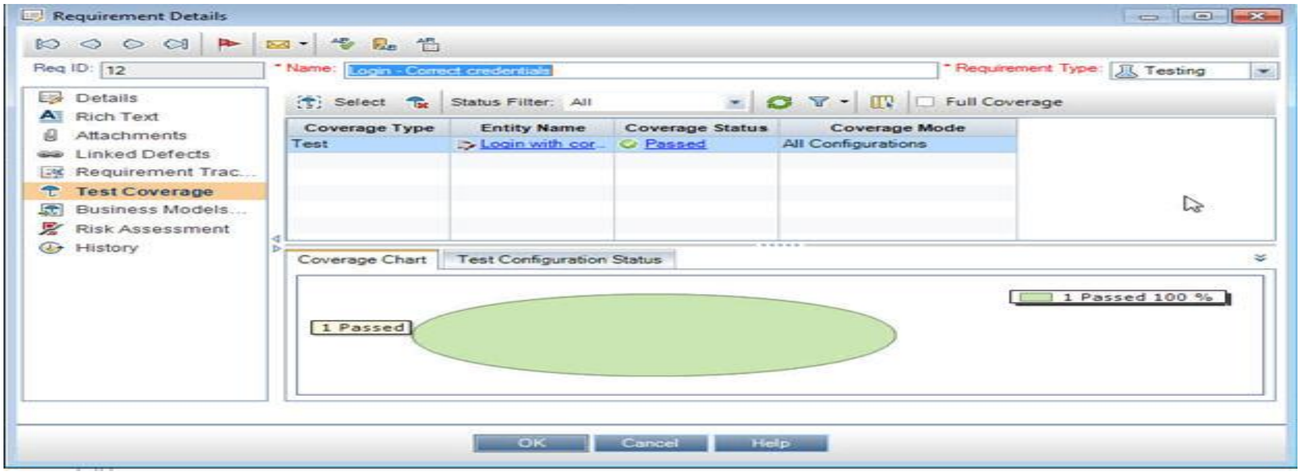
As you can see, the corresponding requirement's direct cover status has changed to "Passed".



No Filter Defined

Name	Direct Cover Status	Author	Req ID
Requirements	—		0
Windows7-64bitTesting	—	tana_ghosh	1
ALM Training	—	mythali_sundara	8
May Release	—	mythali_sundara	9
Login - Correct cre...	Passed	mythali_sundara	12
June Release	—	mythali_sundara	11

Double click on the requirement and go to “Test Coverage Status”



Req ID: 12

* Name: Login - Correct credentials

* Requirement Type: Testing

Status Filter: All

□ Full Coverage

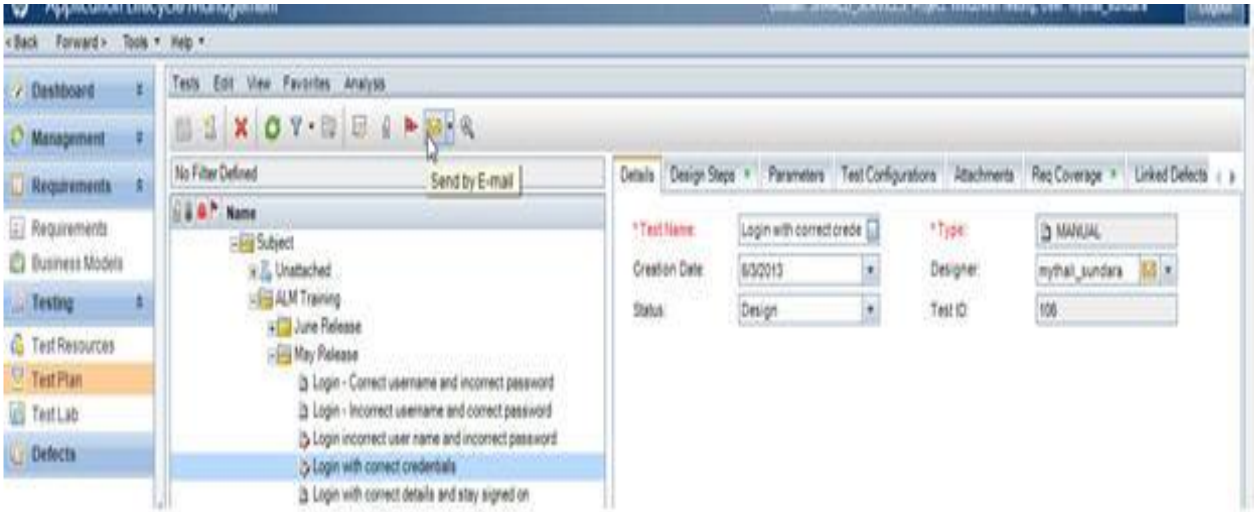
Coverage Type	Entity Name	Coverage Status	Coverage Mode
Test	Login with cor...	Passed	All Configurations

Points to note:

1) The different states of a test case execution can also be customized by the Admin.

2) You can choose to parameterize a manual test by adding parameters to it. When the test is run, the actual test data can be entered.

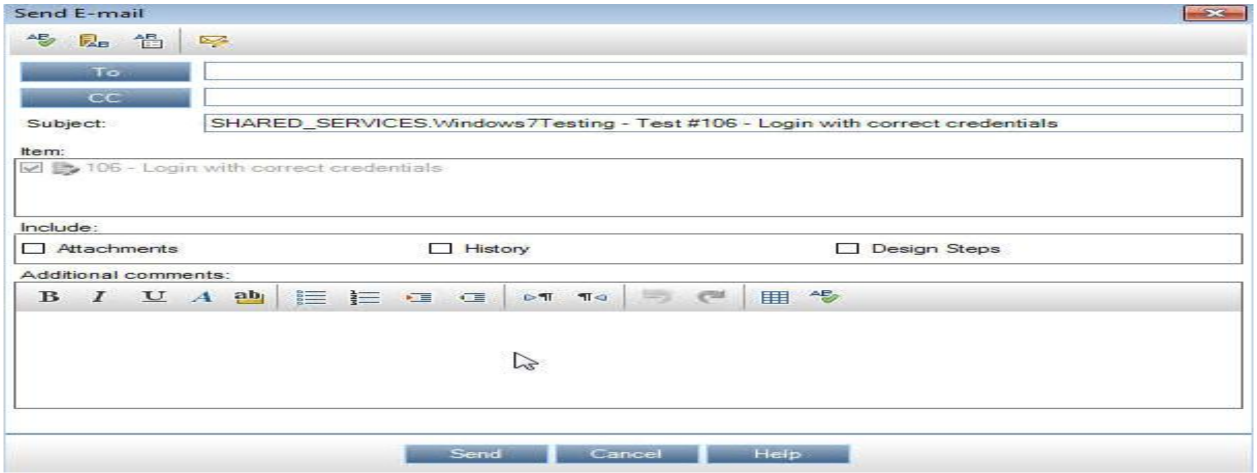
3) At every step in your ALM you can send email through it. To do so, find this icon in your ALM window and click on it. For example, in the test plan tab I am clicking on it.



Login - Correct username and incorrect password
Login - Incorrect username and correct password
Login incorrect user name and incorrect password
Login with correct credentials
Login with correct details and stay signed on

*Test Name: Login with correct crede
*Type: MANUAL
Creation Date: 6/3/2013
Designer: mythali_sundara
Status: Design
Test ID: 106

Once you click on it, the following dialog opens. You can enter the required details and send an email.



Subject: SHARED_SERVICES.Windows7Testing - Test #106 - Login with correct credentials

Item:
☑ 106 - Login with correct credentials

Include:
□ Attachments □ History □ Design Steps

Additional comments:

Conclusion:

In previous and this tutorials we learned – how we create tests, add steps to the tests, link the tests to requirements, create test sets, adding tests to test sets, Executing the tests and finally back tracking our way to see how each activity effects the statistics each time.

Defects: The Defects module enables us to add defects, determine repair priorities, repair open defects, and analyze the Data

Defects can be:

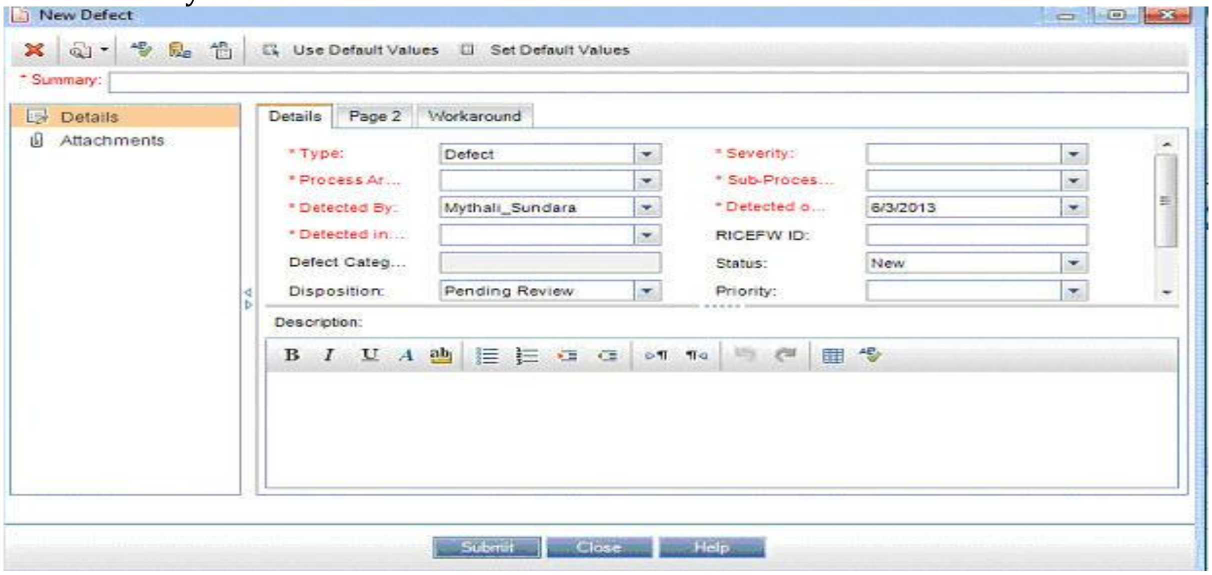
1. Variation in expected and actual results
2. Documentation errors
3. Un-testable requirements that you want to report and track

4. Environment failures that prevent you from testing

QC has a wonderful mechanism that lets we create and track any kind of defects. In the following steps, we will see how to manage defects through ALM.

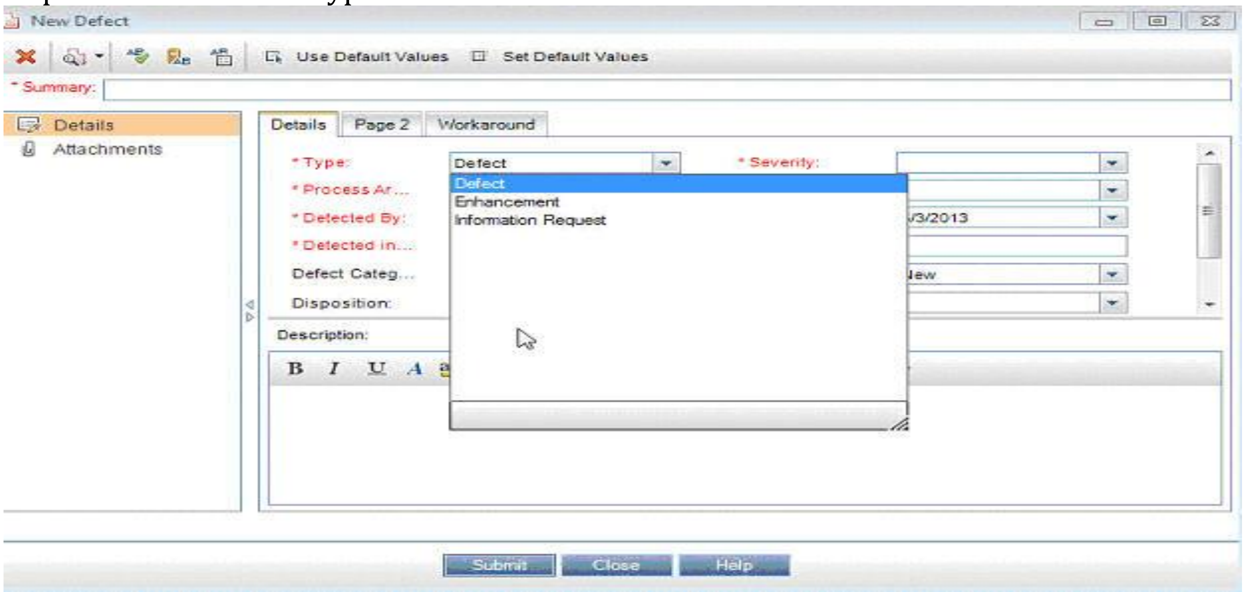
How to add a defect to ALM:

Step #1: Login to ALM to the right project and go to "Defects" tab by navigating from the sidebar. The lists of defects under the project are displayed in a list here.
Step #2: Click on "New Defect". Enter the required details. As you can see all the fields in red are mandatory to enter.



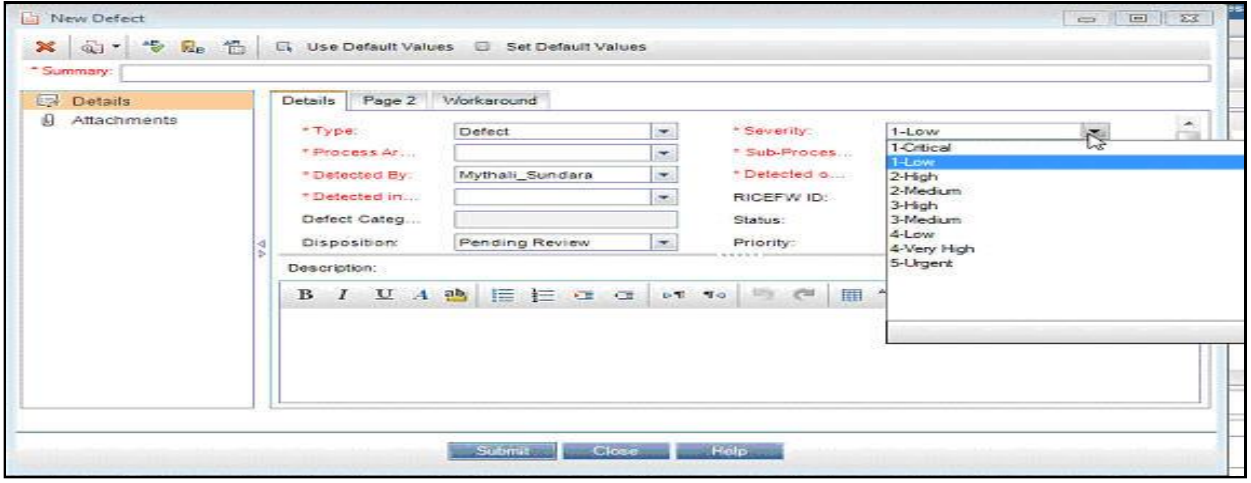
* Summary:
* Type: Defect
* Process Ar...:
* Detected By: Mythali_Sundara
* Detected in...:
Defect Categ...:
Disposition: Pending Review
* Severity:
* Sub-Proces...:
* Detected o...: 6/3/2013
RICEFW ID:
Status: New
Priority:

Step #3: Choose defect type



* Summary:
* Type: Defect
* Process Ar...
* Detected By:
* Detected in...
Defect Categ...
Disposition:
* Severity:

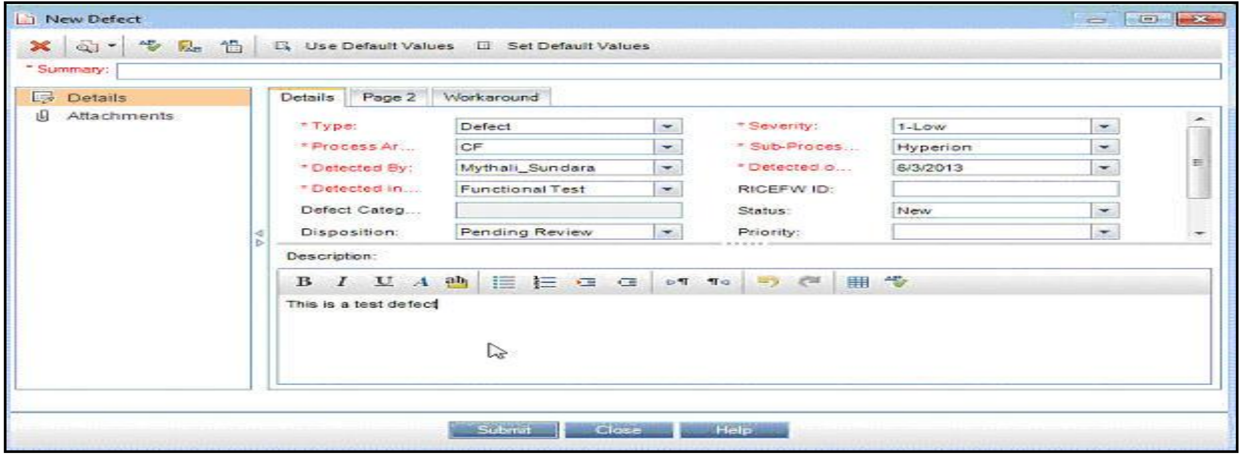
Step #4: Choose severity



* Summary:			
* Type:	Defect	* Severity:	1-Low
* Process Ar...		* Sub-Proces...	
* Detected By:	Mythali_Sundara	* Detected o...	
* Detected in...		RICEFW ID:	
Defect Categ...		Status:	
Disposition:	Pending Review	Priority:	

1-Critical
1-Low
2-High
2-Medium
3-High
3-Medium
4-Low
4-Very High
5-Urgent

Step #5: Enter other details and describe the defect in the "Description" box. You can provide attachments. The other tabs in this dialog are for additional details. Again, project specific.

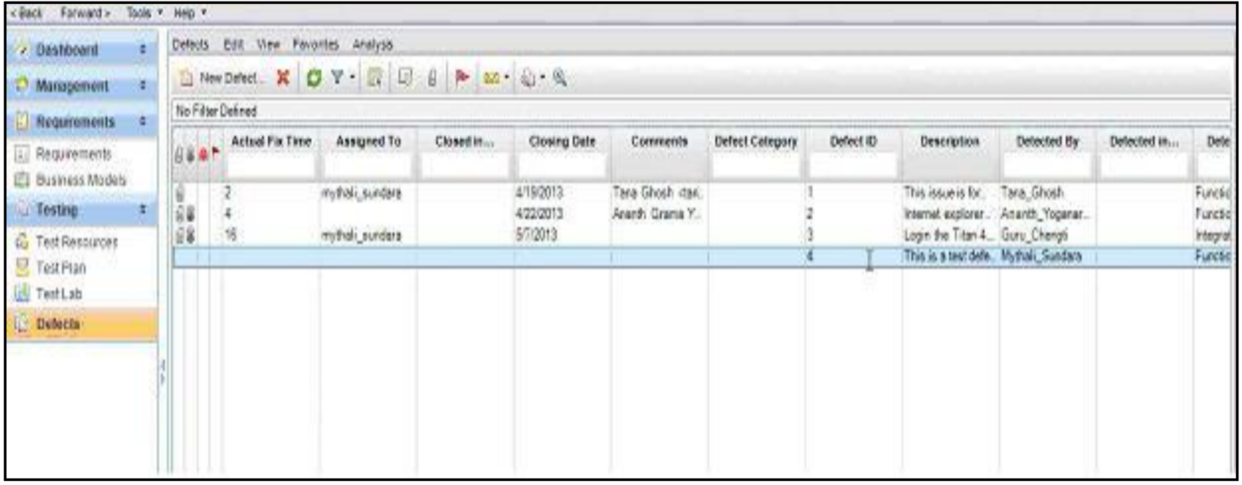


* Summary:			
* Type:	Defect	* Severity:	1-Low
* Process Ar...	CF	* Sub-Proces...	Hyperion
* Detected By:	Mythali_Sundara	* Detected o...	6/3/2013
* Detected in...	Functional Test	RICEFW ID:	
Defect Categ...		Status:	New
Disposition:	Pending Review	Priority:	

Description:

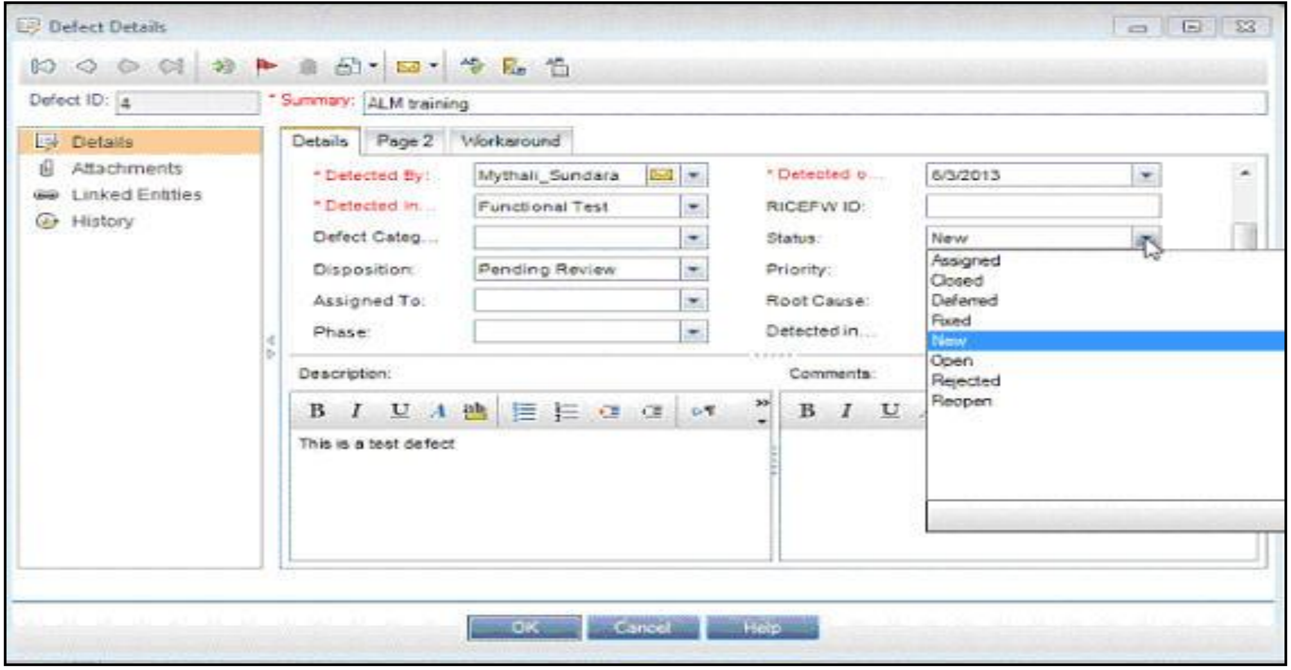
This is a test defect

Step #6: This defect will now appear in the list. It has a unique ID to be identified with.



Actual Fix Time	Assigned To	Closed in...	Closing Date	Comments	Defect Category	Defect ID	Description	Detected By	Detected in...	Dete
2	mythali_sundara		4/19/2013	Tara Ghosh <tar...		1	This issue is for...	Tara_Ghosh		Functio
4			4/22/2013	Ananth Grama Y...		2	Internet explorer...	Ananth_Yoganar...		Functio
16	mythali_sundara		5/7/2013			3	Login the Titan 4...	Guru_Chengti		Integrat
						4	This is a test defe...	Mythali_Sundara		Functio

Step #7: You can now change its status.



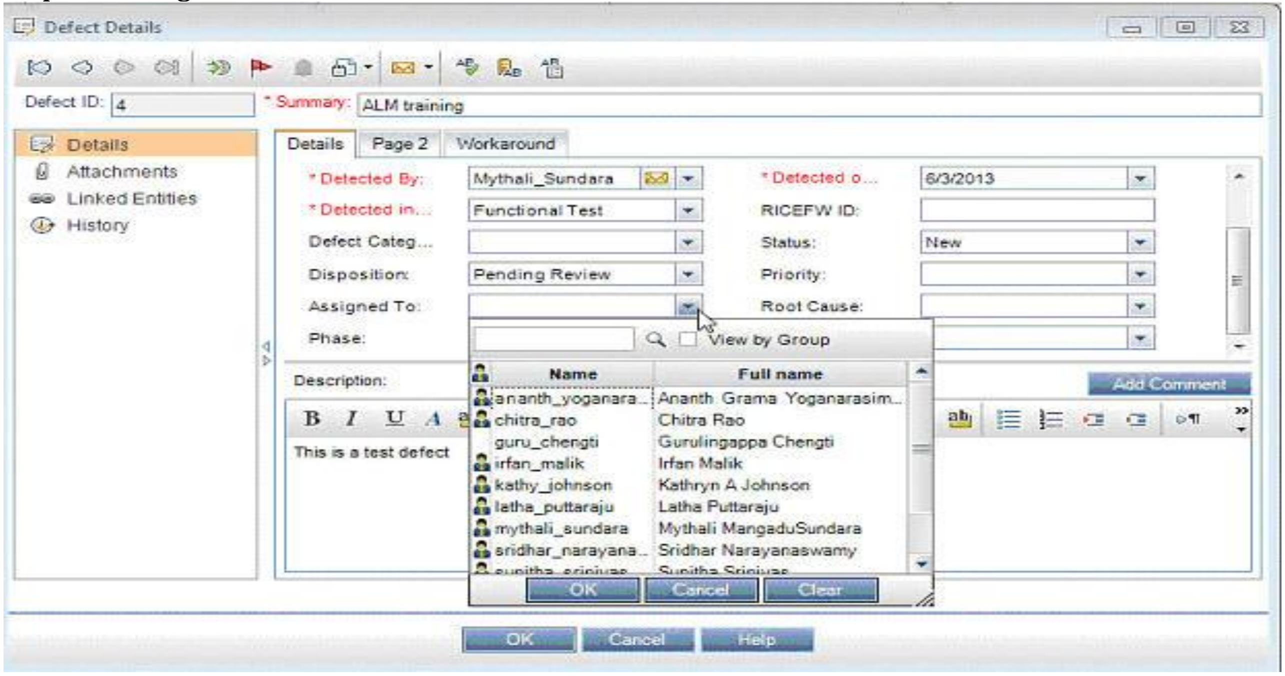
Defect ID: 4
* Summary: ALM training

* Detected By:	Mythali_Sundara	* Detected o...	6/3/2013
* Detected in...	Functional Test	RICEFW ID:	
Defect Categ...		Status:	New
Disposition:	Pending Review	Priority:	
Assigned To:		Root Cause:	
Phase:		Detected in...	

Description: This is a test defect

Comments:

Step #8: Assign it to another user:



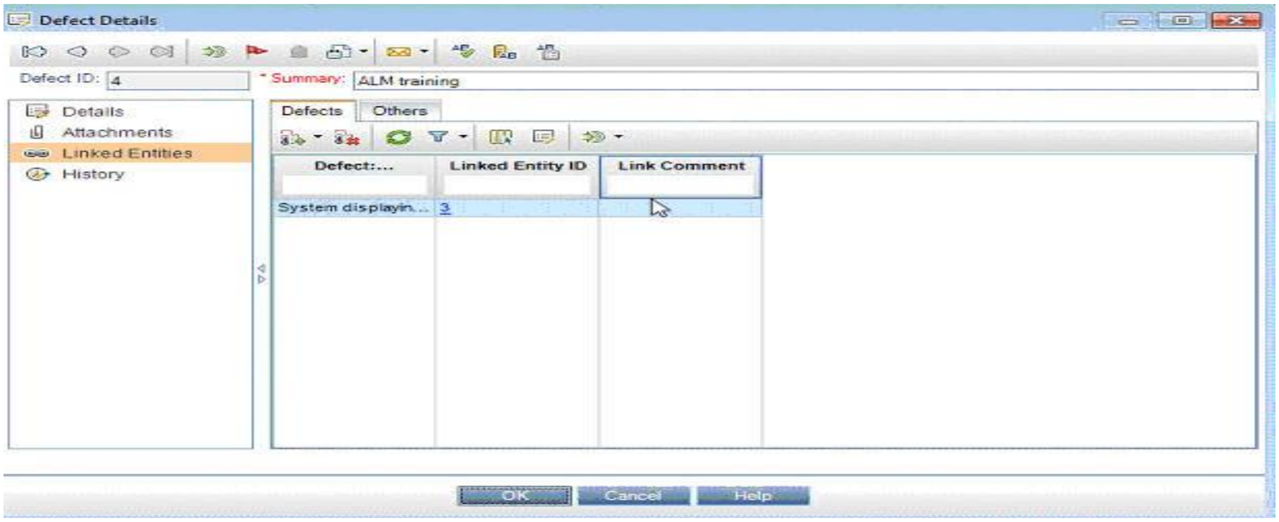
Defect ID: 4
* Summary: ALM training

* Detected By:	Mythali_Sundara	* Detected o...	6/3/2013
* Detected in...	Functional Test	RICEFW ID:	
Defect Categ...		Status:	New
Disposition:	Pending Review	Priority:	
Assigned To:		Root Cause:	
Phase:			

Description: This is a test defect

Name	Full name
ananth_yoganara...	Ananth Grama Yoganarasim...
chitra_rao	Chitra Rao
guru_chengti	Gurulingappa Chengti
irfan_malik	Irfan Malik
kathy_johnson	Kathryn A Johnson
latha_puttaraju	Latha Puttaraju
mythali_sundara	Mythali MangaduSundara
sridhar_narayana...	Sridhar Narayanaswamy
sunitha_srinivas	Sunitha Srinivas

Step #9: Or if this defect is linked to any other defect or you can do so, by choosing "Linked Entities" from the left side bar and choose the other defect that is causing or effecting this defect.



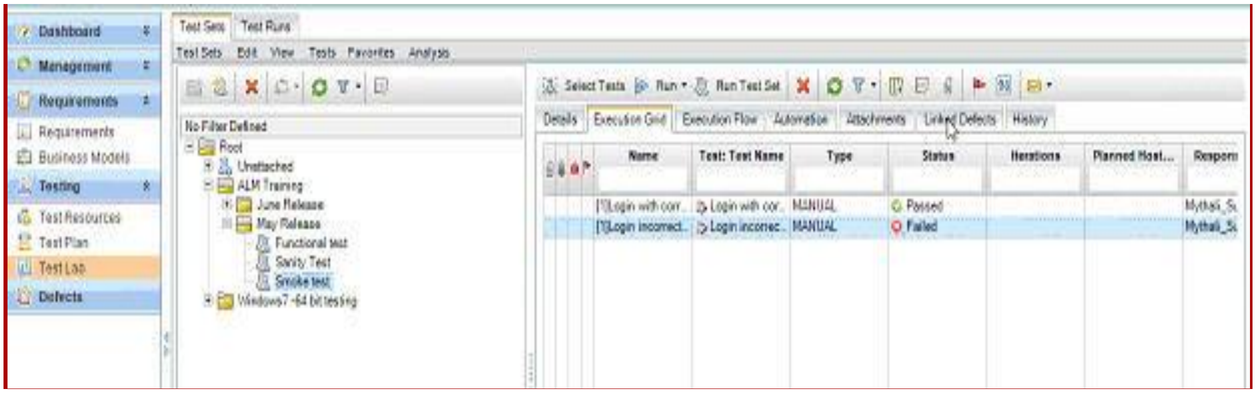
Defect ID: 4

* Summary: ALM training

Defect:...	Linked Entity ID	Link Comment
System displayin...	3	

How to link the defect to a test case:

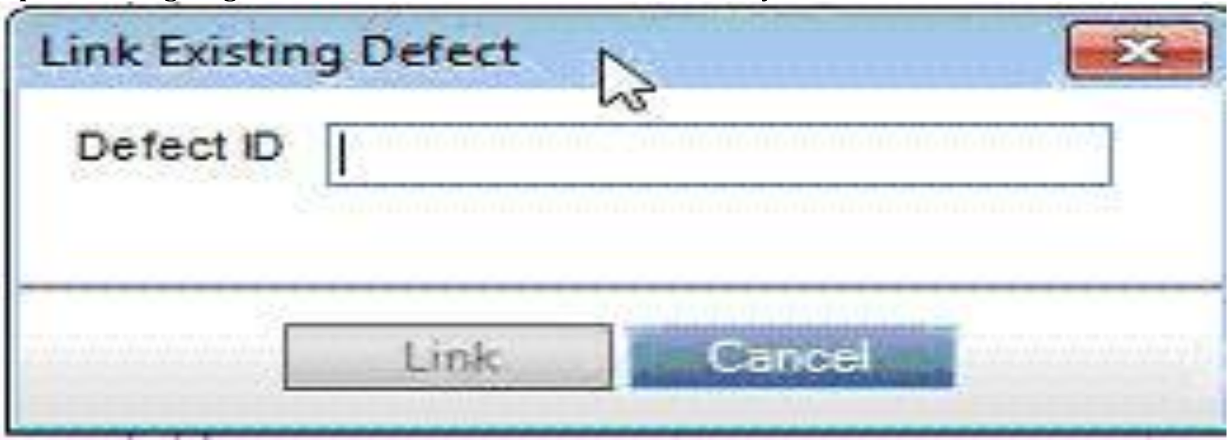
Step #1: Go to Test lab tab and choose the test that failed and the defect is related to that failure.



Name	Test: Test Name	Type	Status	Iterations	Planned Host...	Respon
[1]Login with corr...	Login with cor...	MANUAL	Passed			Mythali_S...
[1]Login incorrect...	Login incorrec...	MANUAL	Failed			Mythali_S...

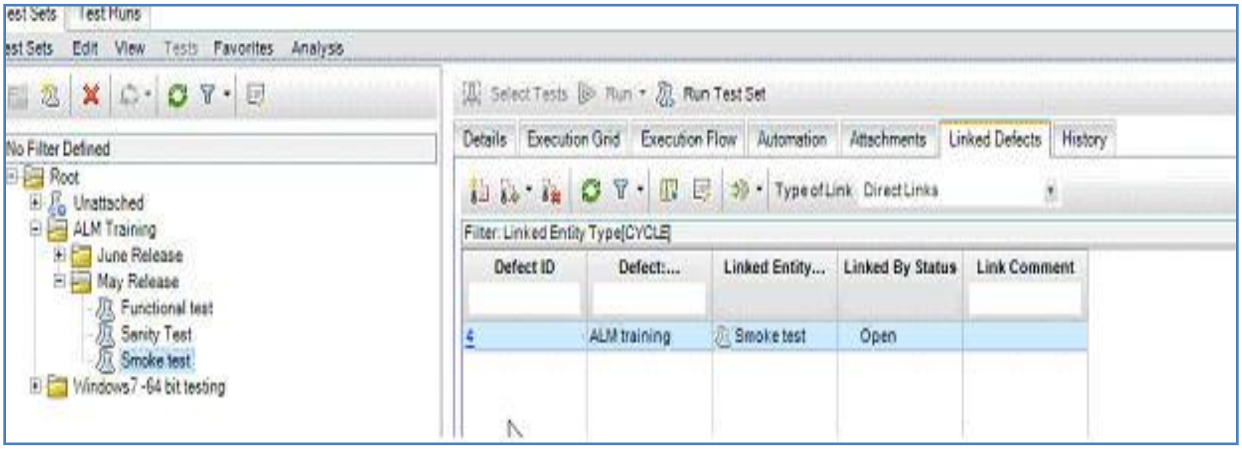
Step #2: Click on "Linked Defects". Here you can either add a new defect and link it by click on this icon: . If you do so, follow the steps that you did in the above section and the defect created will get linked.

Step #3: I am going to choose the icon: To link already created defect



Defect ID

Step #4: Enter the ID and Click “Link”. The defect gets linked.

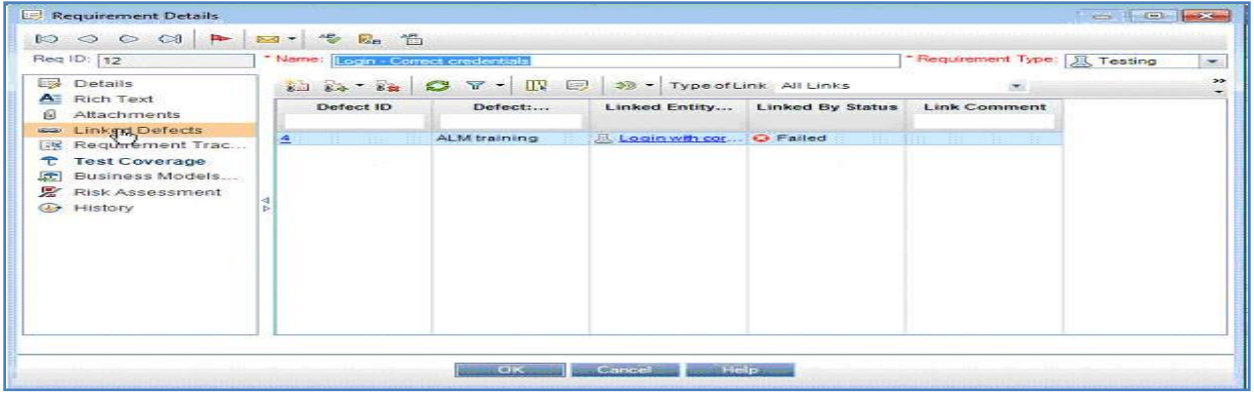


Type of Link: Direct Links

Filter: Linked Entity Type[CYCLE]

Defect ID	Defect:...	Linked Entity...	Linked By Status	Link Comment
4	ALM training	Smoke test	Open	

Step #5: The linked defect can be seen in the test plan as well as requirements.



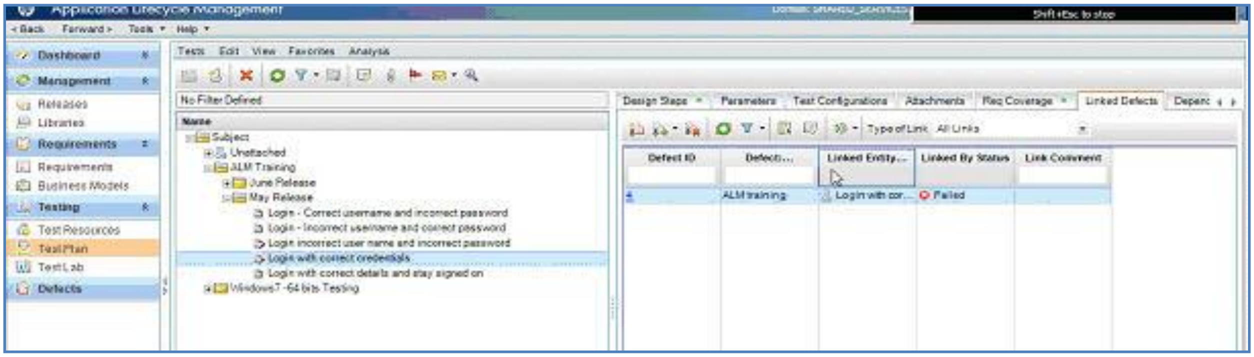
Req ID: 12

* Name: Login - Correct credentials

* Requirement Type: Testing

Type of Link: All Links

Defect ID	Defect:...	Linked Entity...	Linked By Status	Link Comment
4	ALM training	Login with cor...	Failed	



Application Lifecycle Management
Back
Forward
Tools
Help
Dashboard
Management
Releases
Libraries
Requirements
Requirements
Business Models
Testing
Test Resources
Test Plan
Test Lab
Defects
Tests Edit View Favorites Analysis
No Filter Defined
Name
Subject
Unattached
ALM Training
June Release
May Release
Login - Correct username and incorrect password
Login - Incorrect username and correct password
Login incorrect user name and incorrect password
Login with correct credentials
Login with correct details and stay signed on
Windows7 -64 bits Testing
Design Steps
Parameters
Test Configurations
Attachments
Req Coverage
Linked Defects
Depend

Type of Link: All Links

Defect ID	Defect:...	Linked Entity...	Linked By Status	Link Comment
4	ALM training	Login with cor...	Failed	

That finishes our end to end flow from Release to defects.

Tip: Please take a moment and try to see all the field values in the defect details. This will help you get excellent insights into how QC has a field for everything you can ever think of to add to the description to make it completely complete. I specially emphasize on the

“Status” field. This will show how QC is not just a tool to place all your defects at one place but it actually is at the crux of the defect management process. All the states that you would find in a defect life cycle can be set through the “Status” field. I find that it’s most wonderful feature.

Points to note:

1) We have worked with adding a new cycle under a release in the tutorial number 3 and it is mainly a testing related activity. Similarly, you can add a “New Milestone” signifying a certain step in your Project Management Activities.

2) Even though ALM is test management tool, the development and other support teams also have access to it. One of the reasons is to update the defect status.

3) The attachments for a defect are not mandatory but always provide a screenshot of the error in the attachment whenever applicable and possible.

4) The mandatory fields to be entered during the creation of a defect are defined by your Admin and may differ from project to project.

5) The other drop-down values are also defined by your admin.

Conclusion:

Now you are equipped with **everything you needed to know on how to use QC for all your test management activities.** All we need to know now, is how to use its analysis features to make the test reporting and metric collection activities an integral part of your QA process.

JIRA TOOL:

Introduction:

JIRA is an issue tracking software product developed by Atlassian for bug tracking, development tasks, test case management and project management. Project teams choose JIRA to manage and organize tasks, assign work to team members, and follow team activities.

This Material helps to understand JIRA Agile Project Management, test case management activities, and defect management activities.

JIRA as test management tool:

- ✓ Create, view, edit and clone tests
- ✓ Link to stories, tasks, requirements etc.
- ✓ New Agile integration with Test Boards
- ✓ Plan test execution cycles
- ✓ Execute tests
- ✓ Link defects
- ✓ Project-centric navigation
- ✓ Execution Navigator with basic/advanced search and pre-defined/saved filters
- ✓ Two-way end-to-end traceability reports
- ✓ Track quality metrics
- ✓ Zephyr Query Language (ZQL) for advanced searching

Agenda of the Session:

- ✓ Understanding JIRA-Test Process

- Explore the User Interface
- Writing Tests
  - Create Tests
  - Import
  - View and Modify Tests
  - Organize Tests
  - Search Tests
  - Delete Tests
- Planning Test Cycles
  - Creating and cloning Test cycles
  - Adding Tests
  - Exporting Tests
- Executing Tests
  - Adhoc Executions
  - Executing in Test Cycles
  - Quick Execute
- Tracking Test Progress
  - Test Metrics
  - Test Metrics Gadgets
  - Trecability

Test Process

This Tool supports Test Process followed by well-established project and even It supports test process followed in Startup companies (Simple Test Process)

Simple Test Process:

Starting with a Simple Test Process allows you to create or clone an existing test, immediately execute it, file a bug and check out the metrics gadget to keep track of your testing. This is the simplest way of using this application and no prior planning is needed.

Test Process with Basic Execution Planning

A Test Process with Basic Execution Planning essentially lets you group your test executions into Test Cycles so that you can keep track of what has been tested, what is passing/failing, how much is left etc. It involves very basic planning in setting up Test Cycle(s) in a Version and then making sure that every test execution is recorded against those Test Cycle(s).

Test Process with Structured Execution Planning

The next level is a Test Process with Structured Execution Planning. This involves setting up Test Cycle(s) and also getting into a process of creating/modifying a large number of tests, adding them to the appropriate Test Cycle(s) for future execution and then executing them during the dedicated execution stages of your project.

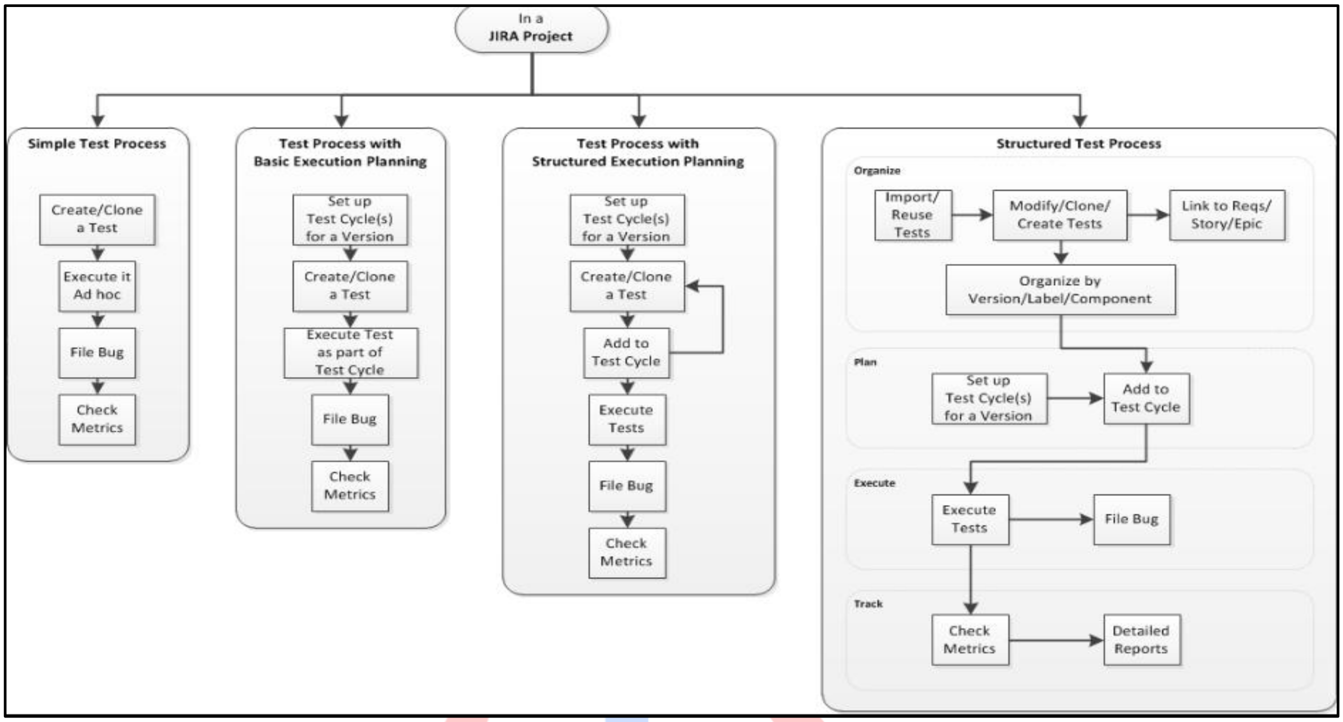
Structured Test Process

A Structured Test Process, as the name implies, allows for distinct phases of organization, planning, execution and tracking of all your testing efforts.

During the organization phase, tests are imported, reused, modified, cloned or created. They are linked to requirements, stories, epics or other issue-types. They are organized by Versions, Components and Labels. During the planning phase, various Test Cycles are set up and tests that need to be executed are added to these cycles. This kind of grouping allows for easy progress tracking.

The execution phase is all business, with users methodically executing tests within the various Test Cycles, linking to existing bugs or filing new ones, tracking progress etc.

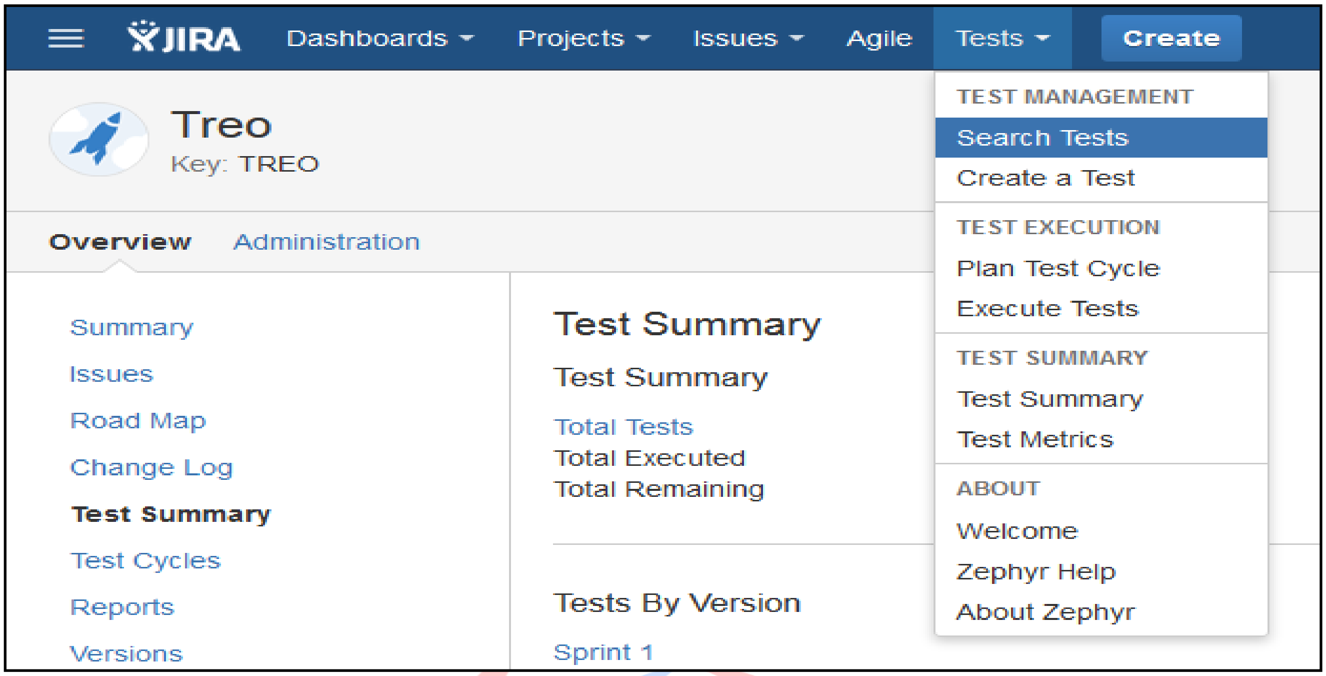
Tracking happens continually with metrics showing in the Test Metrics keeping track of the organized tests, Test Cycle summaries showing progress of the execution, execution metrics keeping track of the quality of the software and detailed reports in various formats available for consumption by a varied audience.



Exploring User Interface in JIRA TOOL:

Main Navigation Bar

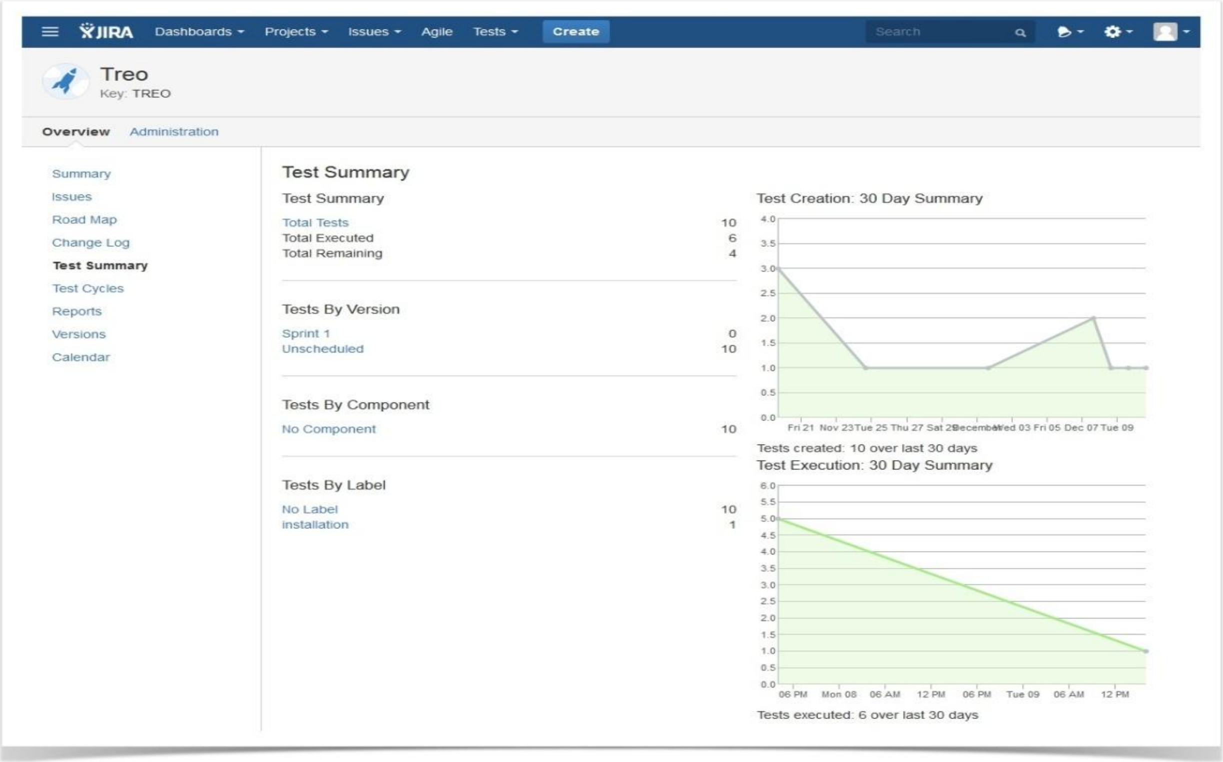
On logging into JIRA, you will see a top-level navigation bar with "Dashboard", "Projects" and "Issues" among other items. A "Tests" item is also present if you have **Zephyr for JIRA Cloud**, and it looks like this:



- ✓ **Search Tests:** This opens the Issue Navigator to allow searching for tests
- ✓ **Create a Test**: Allows you to create a brand new test under the current project. The issue type is "Test" and is indicated by this icon:
- ✓ **Plan Test Cycle**: This takes you to the "Test Cycles" tab on the left, to let you create and view various test cycles based on version
- ✓ **Execute Tests**: Allows the tester to access any test cycle for that project and start executing tests. Note: a test can also be executed directly while viewing it
- ✓ **Test Metrics**: Opens the Test Dashboard with testing metrics gadgets displayed
- ✓ **Welcome**: This page educates and provides access to a variety of resources such as Support, Buying/Upgrading etc.
- ✓ **Zephyr Help**: Online Help documentation (probably what you are reading right now!)
- ✓ **About Zephyr**: Version and license information

Test Summary

The Test Summary page is accessed either via clicking on the top-level "Tests" in the navigation bar or by selecting the "Test Summary" tab on the left side.



Test Summary

Test Summary

Total Tests	10
Total Executed	6
Total Remaining	4

Tests By Version

Sprint 1	0
Unscheduled	10

Tests By Component

No Component	10

Tests By Label

No Label	10
installation	1

Test Creation: 30 Day Summary

Tests created: 10 over last 30 days

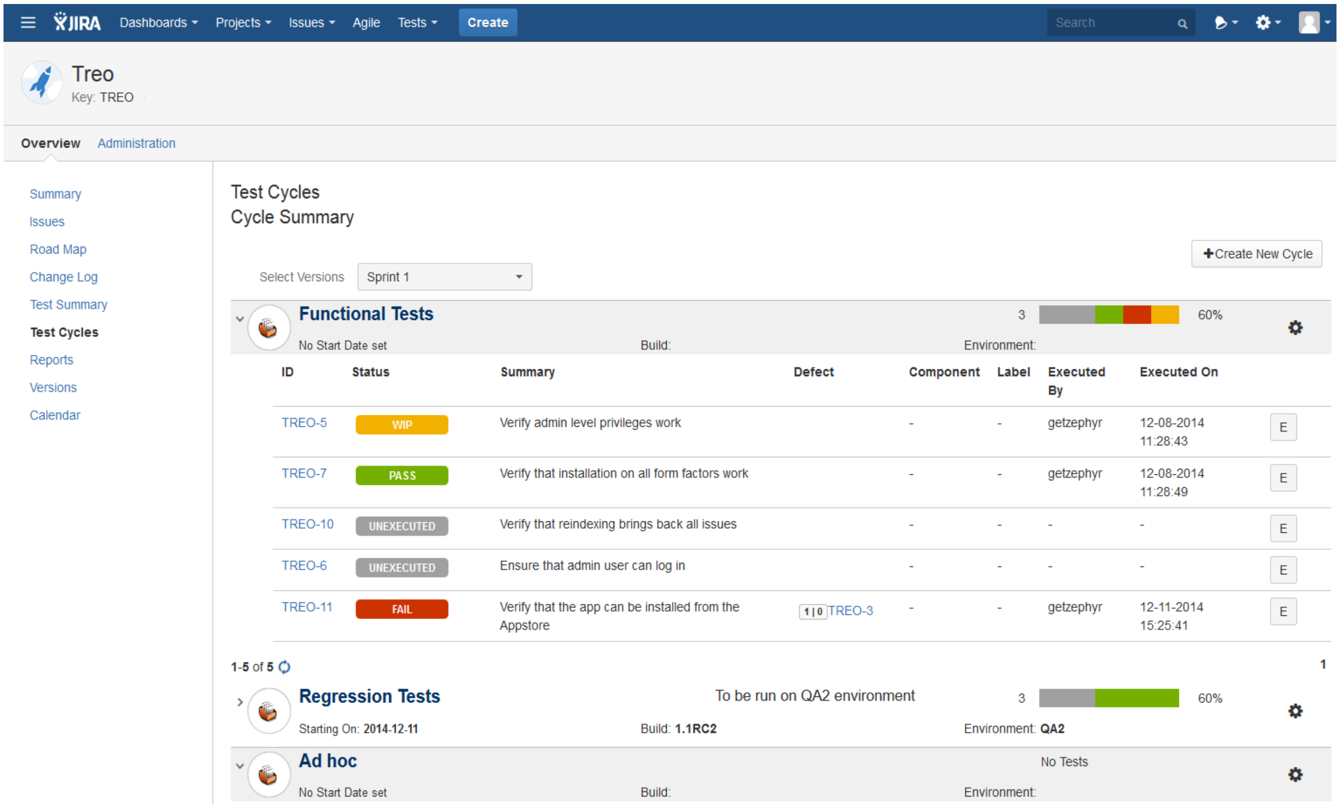
Test Execution: 30 Day Summary

Tests executed: 6 over last 30 days

This page provides a comprehensive view of all the tests that belong to a particular project, summarized based on overall totals, Versions, Components and Labels. Clicking on any of these links will take you to the Issue Navigator and display a list of those tests.

Test Cycles

The Test Cycles page is accessed either via clicking on "Plan Test Cycle" or "Execute Tests" in the top-level "Tests". This can also be selected by clicking on the "Test Cycles" tab on the left side.



Test Cycles
Cycle Summary

Functional Tests 3 60%
No Start Date set Build: Environment:

ID	Status	Summary	Defect	Component	Label	Executed By	Executed On	
TREO-5	WIP	Verify admin level privileges work		-	-	getzephyr	12-08-2014 11:28:43	E
TREO-7	PASS	Verify that installation on all form factors work		-	-	getzephyr	12-08-2014 11:28:49	E
TREO-10	UNEXECUTED	Verify that reindexing brings back all issues		-	-	-	-	E
TREO-6	UNEXECUTED	Ensure that admin user can log in		-	-	-	-	E
TREO-11	FAIL	Verify that the app can be installed from the Appstore	1 | 0 TREO-3	-	-	getzephyr	12-11-2014 15:25:41	E

1-5 of 5 1

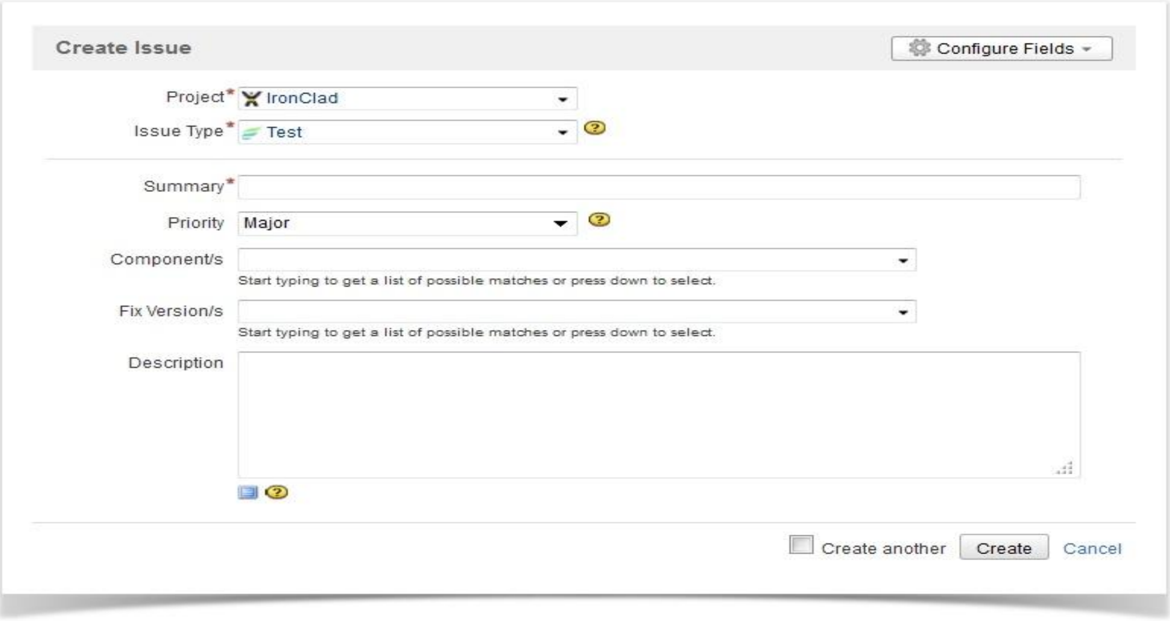
Regression Tests To be run on QA2 environment 3 60%
Starting On: 2014-12-11 Build: 1.1RC2 Environment: QA2

Ad hoc No Tests
No Start Date set Build: Environment:

This page allows you to view test cycles that have been created for a particular version of a project, create new ones, and drill-down into the list of tests for execution. Selecting a different version will change the view and display the test cycles for that selected version. A summary of the execution status is also displayed on the right and hovering over it provides additional stats.

Create and View a Test

A test can be created in any one of the ways that any issue type can be created in JIRA. For example, clicking on the "Create Issue" link or "Create Test" link will bring up the following screen:



Create Issue

Project* IronClad

Issue Type* Test

Summary*

Priority Major

Component/s

Start typing to get a list of possible matches or press down to select.

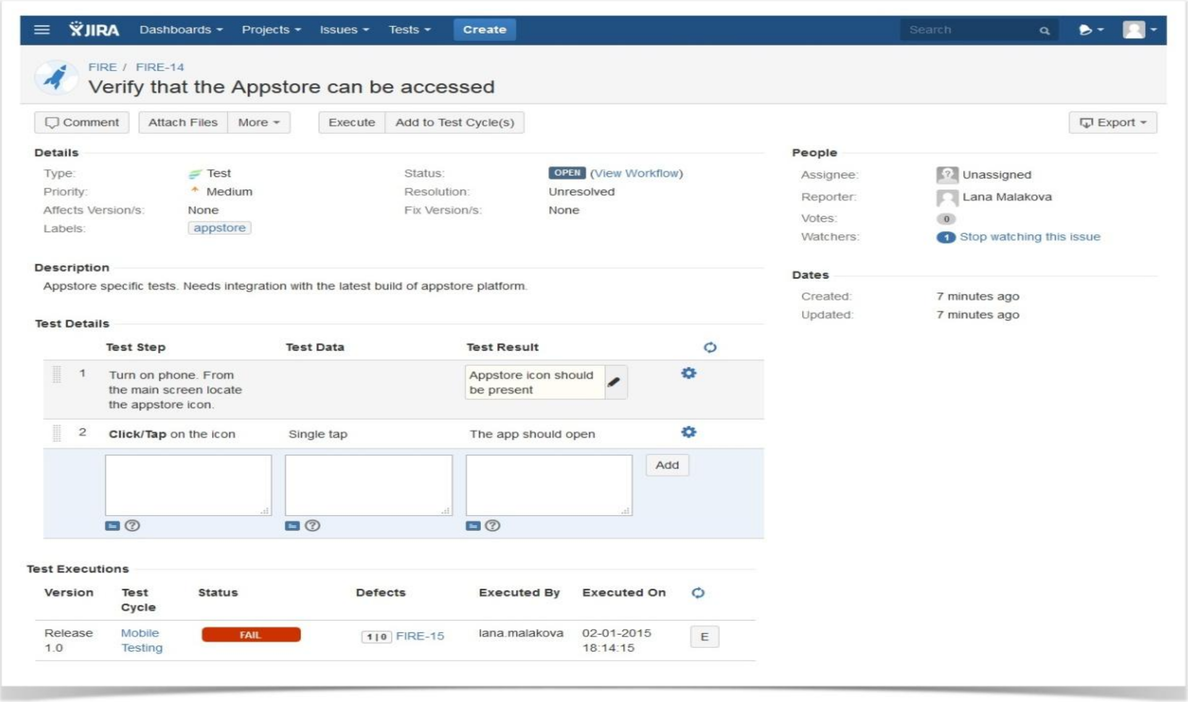
Fix Version/s

Start typing to get a list of possible matches or press down to select.

Description

☐ Create another

A test can be viewed in its entirety in the following "View Issue" page (click to enlarge):



FIRE / FIRE-14

Verify that the Appstore can be accessed

Details

Type:	Test	Status:	OPEN (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	appstore		

Description

Appstore specific tests. Needs integration with the latest build of appstore platform.

Test Details

	Test Step	Test Data	Test Result
1	Turn on phone. From the main screen locate the appstore icon.		Appstore icon should be present
2	**Click/Tap** on the icon	Single tap	The app should open

Test Executions

Version	Test Cycle	Status	Defects	Executed By	Executed On
Release 1.0	Mobile Testing	FAIL	1 | 0 FIRE-15	lana.malakova	02-01-2015 18:14:15

People

Assignee:	Unassigned
Reporter:	Lana Malakova
Votes:	0
Watchers:	1 Stop watching this issue

Dates

Created:	7 minutes ago
Updated:	7 minutes ago

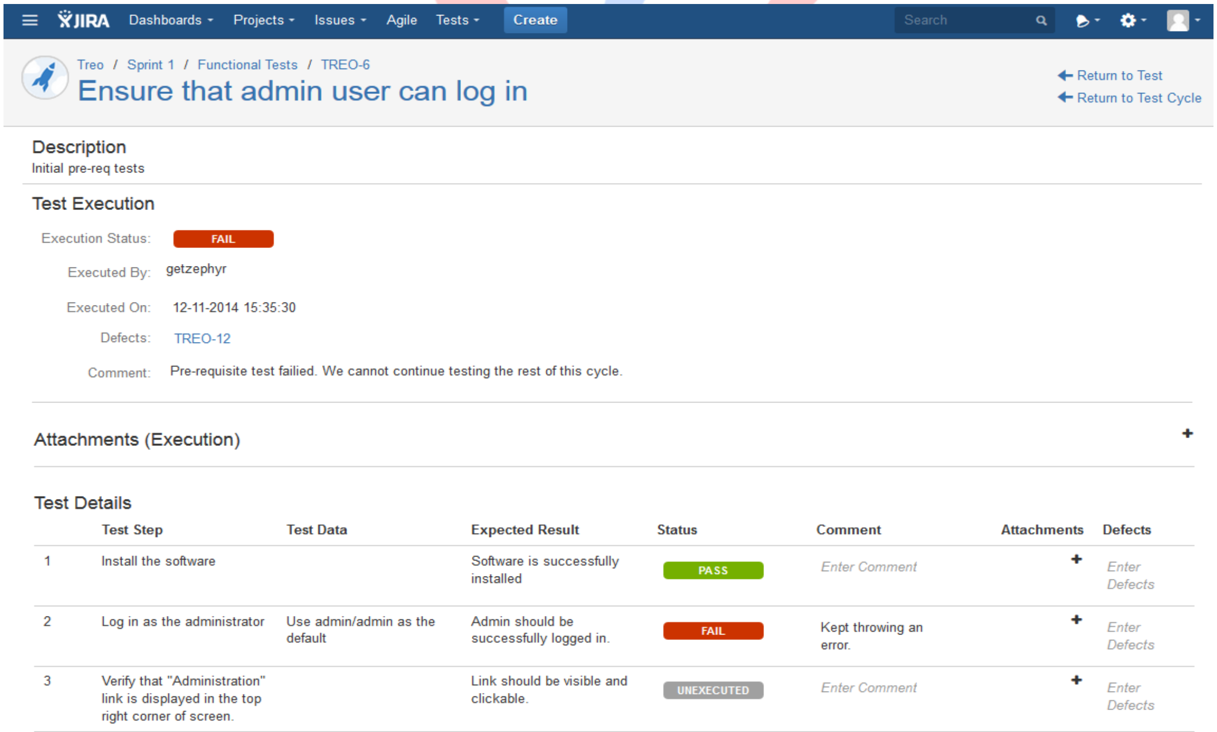
On this screen, you'll find a toolbar that will allow you to access various features associated with editing and executing a test:

- ✓ **Comment:** This allows users to add comments to the test

- ✓ **Add to Test Cycle(s)**: Allows this test to be added to a specific test cycle of a version for test execution
- ✓ **Attach Files**: Allows files to be attached to this test
- ✓ **More Actions**:
  - ○ **Link:** This brings up the "Link Issue" dialog that allows this test to be linked to any other issue (Task, Improvement, Story, Epic etc.)
  - ○ **Clone**: A test can be very easily and quickly cloned
  - ○ **Attach Screenshot**: Allows screenshots to be attached to this test
- ✓ **Execute**: This allows for this test to be executed - either in an ad hoc manner (i.e. not associated with any particular test cycle and/or version) or as part of a test cycle

Execute Test

This screen allows you to execute a test by changing its status, filing a new bug or associating an existing one, adding attachments and comments. You can then save and return to the test or the test cycle from where you had come.



Treo / Sprint 1 / Functional Tests / TREO-6

Ensure that admin user can log in

← Return to Test
← Return to Test Cycle

Description
Initial pre-req tests

Test Execution

Execution Status: FAIL
Executed By: getzephyr
Executed On: 12-11-2014 15:35:30
Defects: TREO-12
Comment: Pre-requisite test failied. We cannot continue testing the rest of this cycle.

Attachments (Execution) +

Test Details

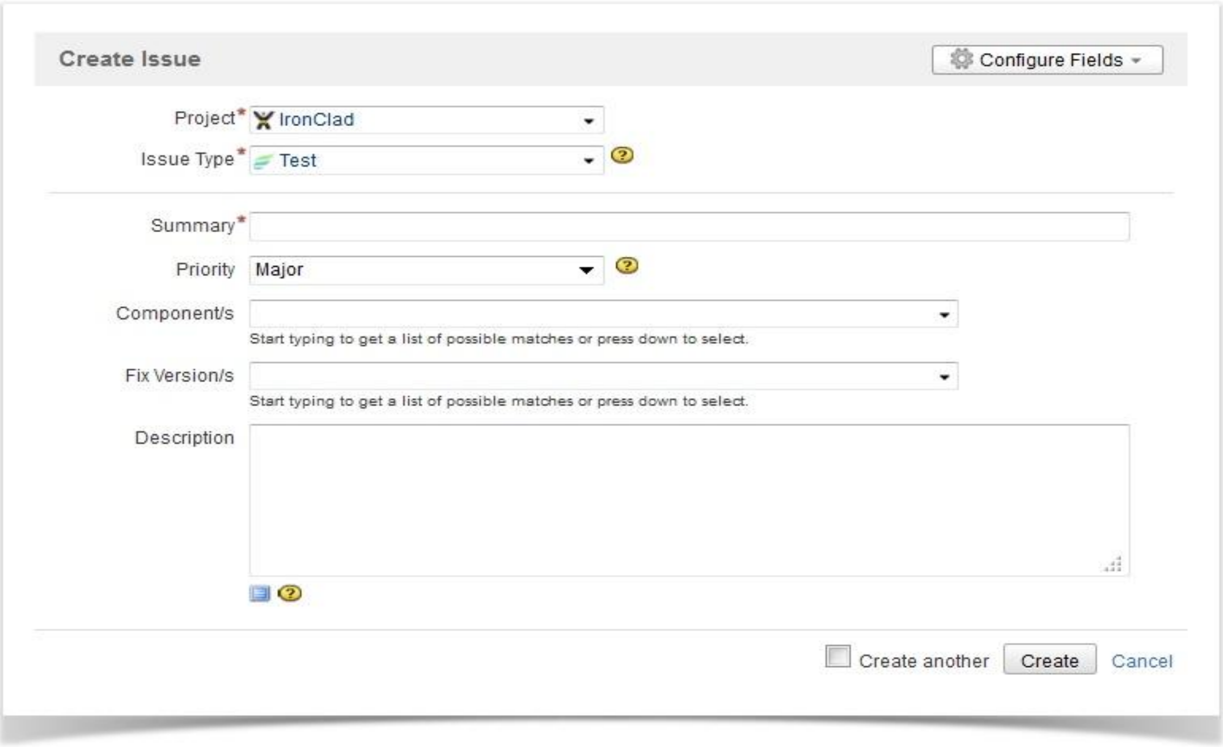
	Test Step	Test Data	Expected Result	Status	Comment	Attachments	Defects
1	Install the software		Software is successfully installed	PASS	Enter Comment	+	Enter Defects
2	Log in as the administrator	Use admin/admin as the default	Admin should be successfully logged in.	FAIL	Kept throwing an error.	+	Enter Defects
3	Verify that "Administration" link is displayed in the top right corner of screen.		Link should be visible and clickable.	UNEXECUTED	Enter Comment	+	Enter Defects

Create Tests

You need to have the "Create Issue" permission in the relevant project to be able to create a test. If you do not have this permission, please contact your JIRA administrator.

To create a new test:

1. Do one of the following to open the **Create Issue** dialog.
   - ✓ Keyboard shortcut: **'c'**.
   - ✓ In the top navigation bar, click the **Create** button.
   - ✓ From the "Tests" menu, click on "Create a Test"
2. In the displayed **Create Issue** dialog, enter the details for the test.
3. Optional: If you want to create a series of test with similar details, you can select the **Create another** check box.
4. When you select this option, a new issue dialog will appear after your test is created, automatically pre-populated with your previous test details, while leaving the 'Summary' field blank.
5. Click the **Create** button to create the test.



Create Issue

Project* IronClad

Issue Type* Test

Summary*

Priority Major

Component/s

Start typing to get a list of possible matches or press down to select.

Fix Version/s

Start typing to get a list of possible matches or press down to select.

Description

□ Create another

Once the test has been created, a confirmation is displayed with the new test ID. You can then go to that test to enter next level details such as Test Details and Attachments.

⚠ Note that only the Fix Version/s field is used for tests. All tests that have a Fix Version populated will be organized accordingly. If that field is left blank, then they will belong to an "Unscheduled" version. See Organize Tests for more details.

How to Import Test Cases to JIRA Tool from excl?

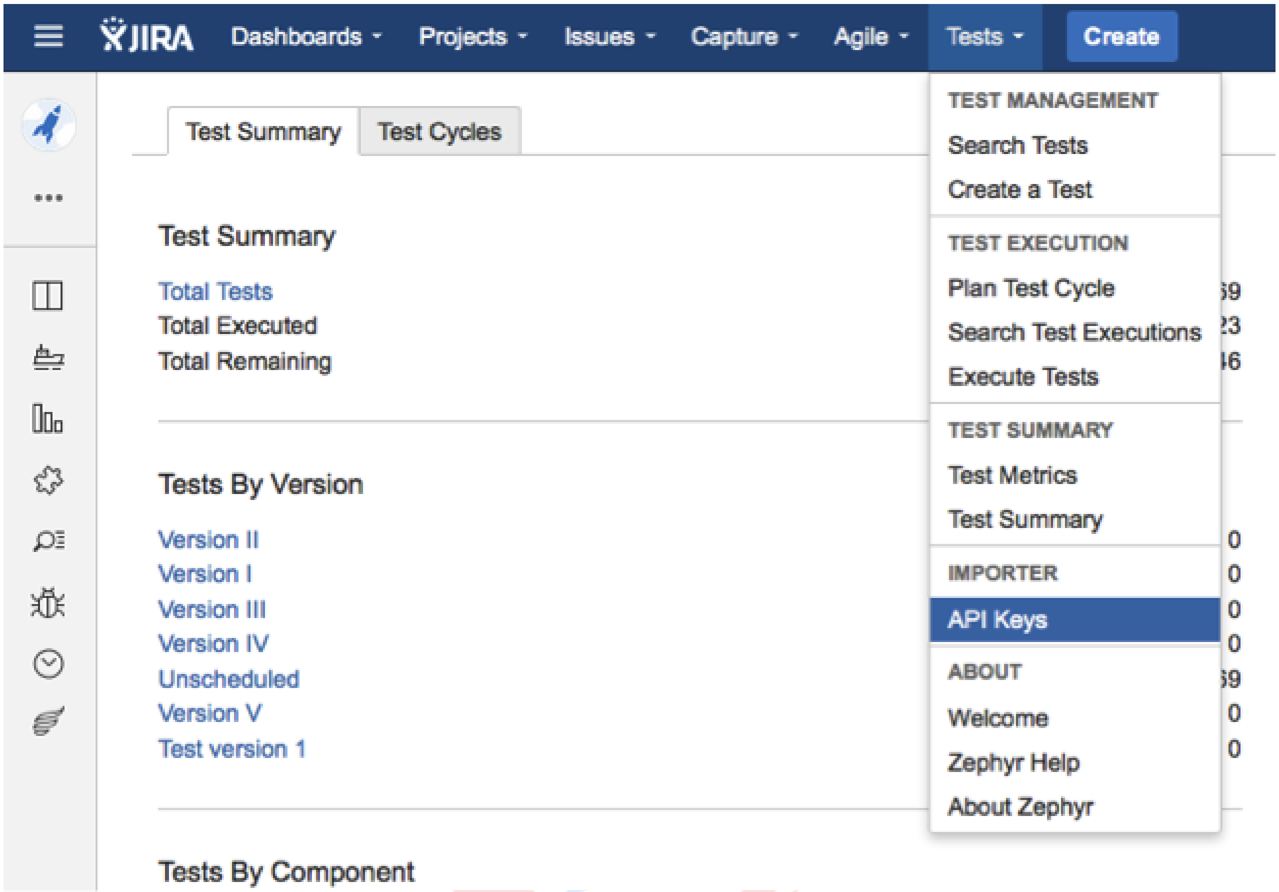
Zephyr for JIRA Importer Utility

This guide will cover how to import tests into Zephyr for JIRA (both Server/DataCenter and Cloud versions) using the Importer Utility; this includes:

1. Downloading the Importer Utility
2. Launching the Importer Utility
3. Configuring fields necessary for importing
4. JIRA results of an import

Pre Requisite:

Java – Please make sure you have the latest version of Java installed. You can acquire it from http://java.com if you do not have it already. **API Access & Secret Keys** (only for **Zephyr for JIRA Cloud**) – you'll need your Zephyr for JIRA API access and secret key in order to make a connection. You can find them by logging into your JIRA Cloud instance, browse to Tests (top menu bar) > Importer > API Keys.



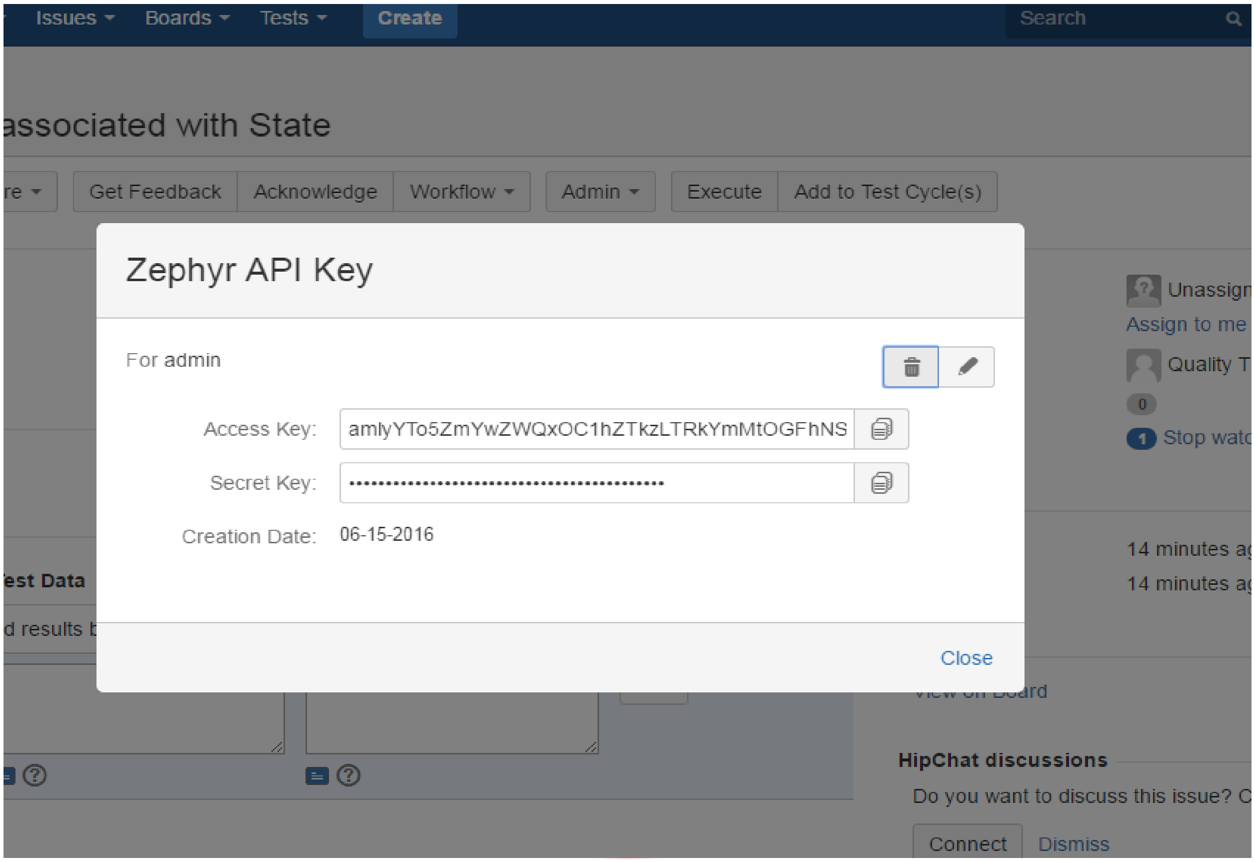
Selecting "API Keys", the Zephyr API Key view will be displayed which contains an Access Key and Secret Key unique to your JIRA user. Also on this view are three options in the top right hand corner:

- ✓ Copy to clipboard which copies both Keys to your system clipboard
- ✓ Delete which removes the current generated Key pair
- ✓ Regenerate which regenerates the Key pair. Note that once the Keys are regenerated the previous Keys can no longer be used with the Importer Utility.

Selecting "API Keys", the Zephyr API Key view will be displayed which contains an Access Key and Secret Key unique to your JIRA user. Also on this view are three options in the top right hand corner:

- ✓ Copy to clipboard which copies both Keys to your system clipboard
- ✓ Delete which removes the current generated Key pair

- ✓ Regenerate which regenerates the Key pair. Note that once the Keys are regenerated the previous Keys can no longer be used with the Importer Utility.



Zephyr API Key

For admin

Access Key: amlyYTo5ZmYwZWQxOC1hZTkzLTRkYmMtOGFhNS

Secret Key: ••

Creation Date: 06-15-2016

se the Copy to Clipboard option to save this Key pair to your system clipboard. You will require it before starting the import process in the Importer Utility.

Note the following:

1. The use of the Access and Secret Keys is required only for Zephyr for JIRA Cloud users. This option in the Test menu is not available in Zephyr for JIRA Server/DataCenter.
2. For security purposes, do not share this Key pair. Each user that needs to use the Importer Utility should use the steps above to generate their unique Key pair.
3. This Key pair does not expire so users can use their unique Key pair every time they use the Importer Utility.
4. Copying to the clipboard copies both Access Key and Secret Key in below format. Paste it to a notepad. Copy each key individually and paste it in its appropriate box in the importer.

5. “XXXXXXX” in Access Key (First box) and “YYYYYYY” in Secret Key (Second box) of the importer.

{"accessKey":"XXXXXXXX","secretKey":"YYYYYYYYY"}

Downloading the Importer

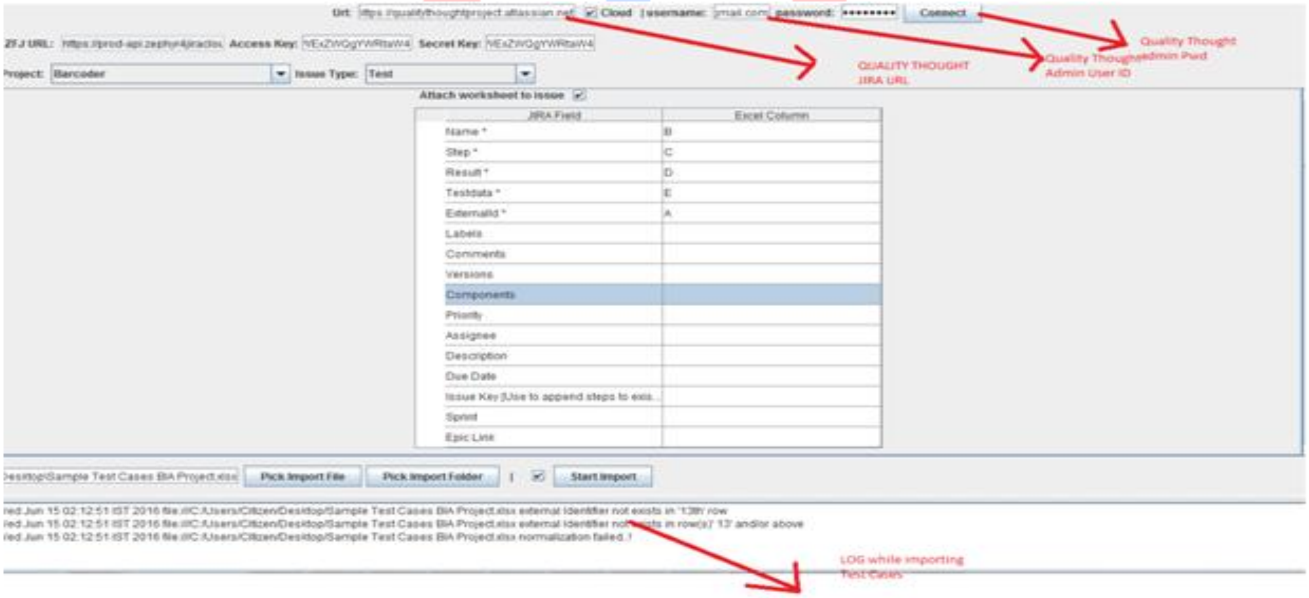
The Zephyr for JIRA Importer Utility is available on BitBucket. The direct link for it is https://bitbucket.org/zfjdeveloper/zfj-importer/downloads. This location provides download of the Importer Utility software. Download the Utility software to the system from which imports to Zephyr for JIRA will be run.

Using the Importer

Launching the Importer

To launch the utility double click on the downloaded jar file or run through the command prompt as: java -jar <importer file which will result in opening a window as shown below.

Importing from Excel



Url: https://qualitythoughtproject.atlassian.net ☑ Cloud | username: gmail.com password: •••••••••

ZFJ URL: https://prod-api.zephyr4jiraclou Access Key: VExZWQgYWRtaW4 Secret Key: VExZWQgYWRtaW4

Project: Barcoder Issue Type: Test

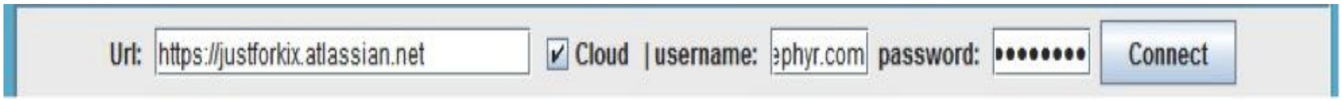
Attach worksheet to issue ☑

JIRA Field	Excel Column
Name *	B
Step *	C
Result *	D
Testdata *	E
ExternalId *	A
Labels	
Comments	
Versions	
Components	
Priority	
Assignee	
Description	
Due Date	
Issue Key [Use to append steps to exis...	
Sprint	
Epic Link	

Desktop\Sample Test Cases BIA Project.xlsx | ☑

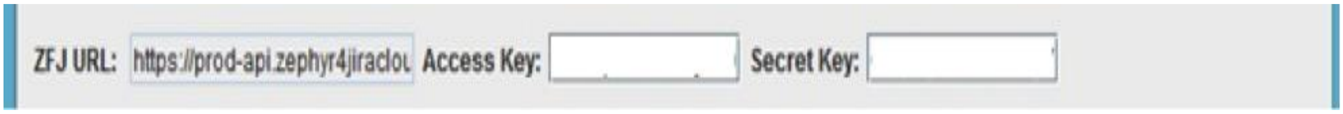
```
ed Jun 15 02:12:51 IST 2016 file:///C:/Users/Citizen/Desktop/Sample Test Cases BIA Project.xlsx external Identifier not exists in '13th' row
ed Jun 15 02:12:51 IST 2016 file:///C:/Users/Citizen/Desktop/Sample Test Cases BIA Project.xlsx external Identifier not exists in row(s)' 13' and/or above
ed Jun 15 02:12:51 IST 2016 file:///C:/Users/Citizen/Desktop/Sample Test Cases BIA Project.xlsx normalization failed..!
```

- ✓ Enter the URL for your JIRA Server or Cloud instance. If importing to Zephyr for JIRA Cloud, make sure the Cloud checkbox is checked. Enter your JIRA username and password. Zephyr for JIRA Server version users, proceed to Step 3.



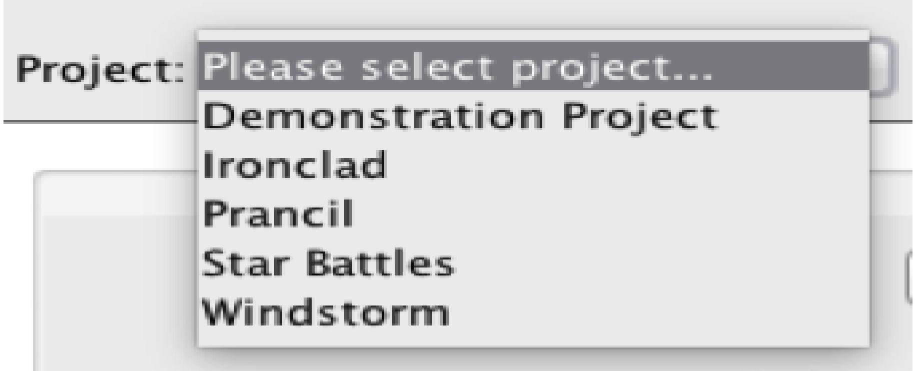
Url: https://justforkix.atlassian.net ☑ Cloud | username: ephyr.com password: ●●●●●●●●

- ✓ **For Zephyr for JIRA Cloud users only:** The "ZFJ URL" field is a read-only field and is pre-entered. Enter your API Access and Secret keys (view the "Requirements" section of this document to see how to access this Key pair).

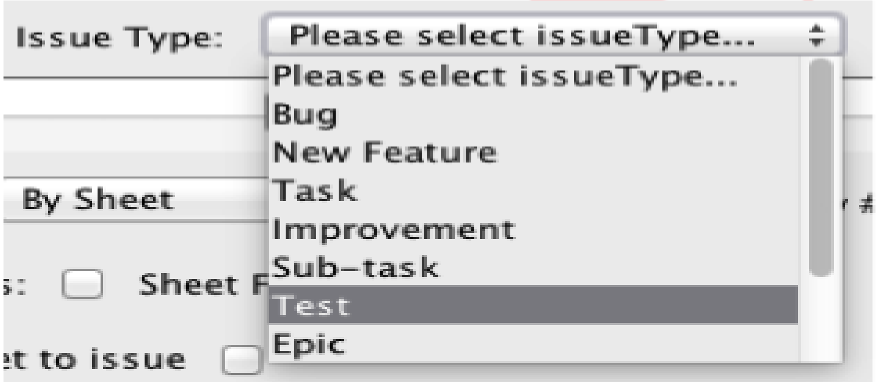


ZFJ URL: https://prod-api.zephyr4jiraclou Access Key: Secret Key:

- ✓ Click Connect.
- ✓ Select the project and issue type desired (supported: Test, Bug, Improvement, Task, New Feature)

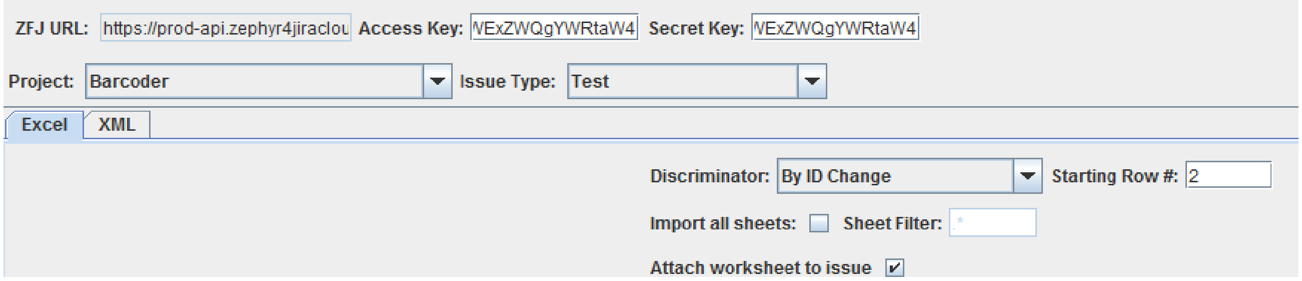


Project: Please select project...



Issue Type: Please select issueType...

- ✓ Select a Discriminator and enter Starting Row value.



ZFJ URL: https://prod-api.zephyr4jiraclou Access Key: WExZWQgYWRtaW4 Secret Key: WExZWQgYWRtaW4

Project: Barcoder Issue Type: Test

Excel XML

Discriminator: By ID Change Starting Row #: 2

Import all sheets: □ Sheet Filter: *

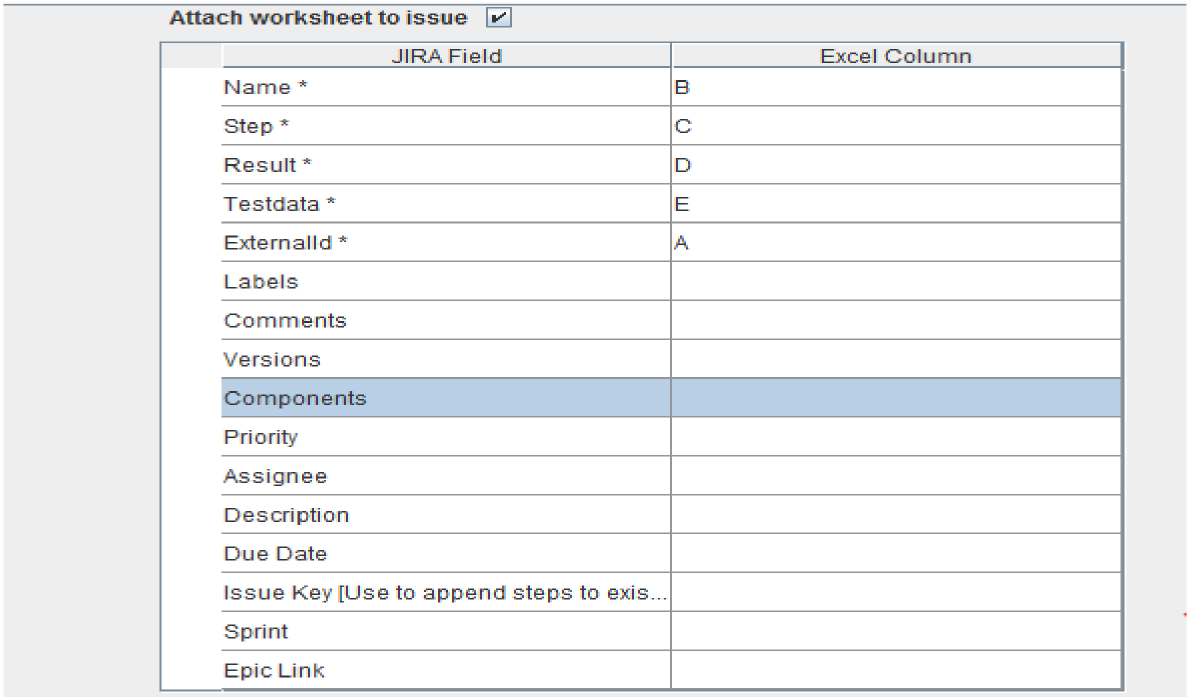
Attach worksheet to issue ☑

The Discriminator field is used by the Importer Utility to determine the end of one test case and the beginning of the next test case. It allows one of four options for selection:

- ✓ By Sheet: use this when a test case is listed per excel sheet
- ✓ By Empty Row: use this when an empty row separates consecutive test cases
- ✓ By ID change: use this when a unique test ID exists for each test case to be imported
- ✓ By Test Case Name change: use this when each test case has a unique test case name

Note: to properly determine the Starting row value, reference the excel file to be imported. The Starting row will be the first row in the excel spreadsheet containing a test case. For example, if the first two rows of the excel sheet are headers and the third row is where the first test case is listed, then the Starting row value should be set to 3.

- ✓ Map your Excel spreadsheet columns to your JIRA and Zephyr for JIRA fields. Use the column letters instead of any headers you might have to reference the data. For example: if test case name is in Column B of the excel sheet, enter "B" against the JIRA field “Name”

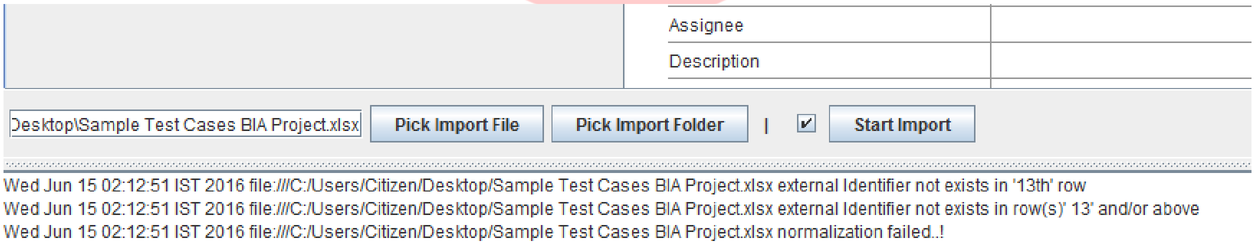
- 

Attach worksheet to issue ☑

JIRA Field	Excel Column
Name *	B
Step *	C
Result *	D
Testdata *	E
ExternalId *	A
Labels	
Comments	
Versions	
Components	
Priority	
Assignee	
Description	
Due Date	
Issue Key [Use to append steps to exis...	
Sprint	
Epic Link	

If your Excel file has multiple sheets and you wish to import them all, check **Import All Sheets**.

- ✓ Select the Excel file to be imported by choosing either **Pick Import File** or **Pick Import Folder**.



Desktop\Sample Test Cases BIA Project.xlsx

```
Wed Jun 15 02:12:51 IST 2016 file:///C:/Users/Citizen/Desktop/Sample Test Cases BIA Project.xlsx external Identifier not exists in '13th' row
Wed Jun 15 02:12:51 IST 2016 file:///C:/Users/Citizen/Desktop/Sample Test Cases BIA Project.xlsx external Identifier not exists in row(s)' 13' and/or above
Wed Jun 15 02:12:51 IST 2016 file:///C:/Users/Citizen/Desktop/Sample Test Cases BIA Project.xlsx normalization failed..!
```

Click **Start Import**. The results of the import will appear in the log window below. Any errors that occur will be displayed here.

Viewing and Modifying Tests

A test can be viewed and modified just like any other issue in JIRA. If you know the issue key of the test, you can enter that in any search field or search through the Issue Navigator.

The view test screen shows the following:

- ✓ **Header**: This displays the project to which this test belongs, the issue key of the test and its one-line Summary

- ✓ **Toolbar**: This toolbar allows you to access often used functionality

- ✓ Edit: This button takes you to the edit mode of a test where many of the fields can be edited. This is very similar to creating an issue. Please note that Test Details have to be edited from this View Issue screen.

- ✓ Comment: Adding a comment to a test is a useful way to record additional details about a test, and collaborate with team members. Comments are shown in the Comments tab of the **Activity** section.

- ✓ Attach Files: You can attach one or more files to a test (Note: Your JIRA Administrator must have enabled file attachments. You will also need "Create Attachments" permission in the appropriate projects).

- ✓ Attach Screenshot: You can attach a screenshot to a test if your Administrator has configured JIRA and your permissions correctly.

- ✓ Clone: 'Cloning' (copying) a test allows you to quickly create a duplicate of the test within the same project. The clone test is a replica of the original test, containing the same information stored in the original test— e.g. Summary, Labels, Components, Test Details, etc. except forTest Execution details. A clone issue is a separate entity from the original issue. Operations on the original issue have no effect on the clone issue and vice versa.

- ✓ Link: Allows you to link another issue or a web link to this test, typically a requirement or a story or an epic.

- ✓ Execute: This button allows you to immediately execute this test, either in an ad hoc manner or as part of a Test Cycle.

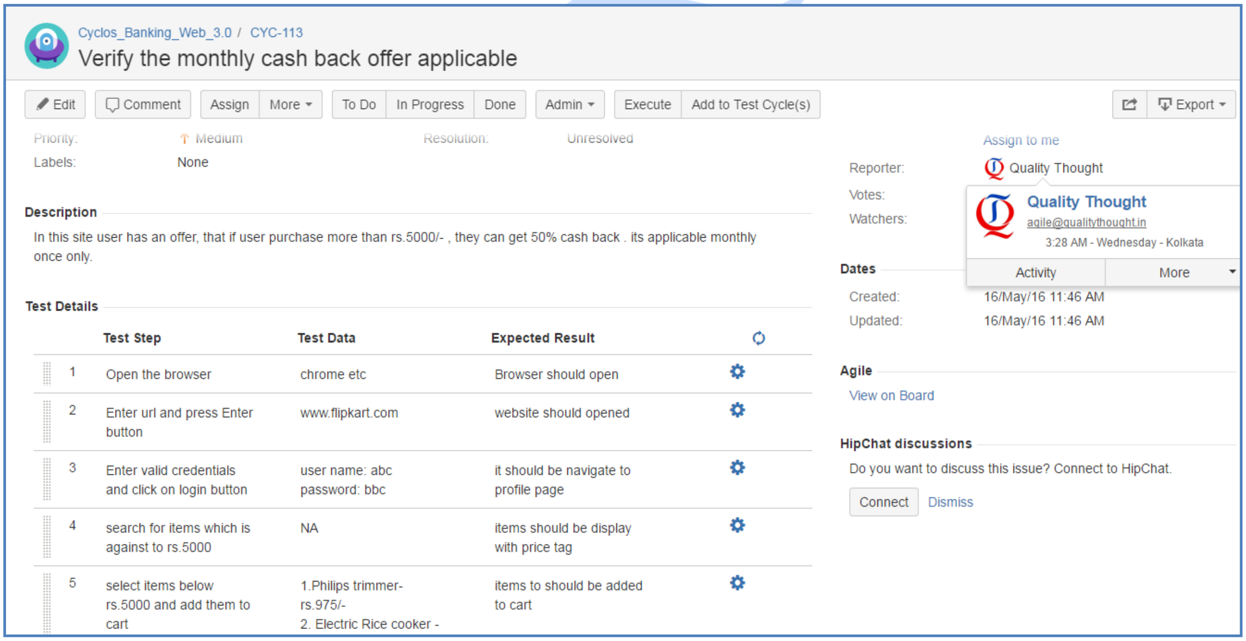
- ✓ Add to Test Cycle(s): This allows you to add this test to an existing Test Cycle. That then also shows in the **Test Executions** section below.

- ✓ **Details**: This section is almost exactly like the one for any other issue-type but note the following:

- ✓ Fix Version/s: This field is the one that is used to organize tests i.e. make it belong to a Version.

- ✓ Status: This field is not used.

1. Resolution: This field is not used.
2. **Description**: This section is exactly like the one for any other issue-type and can be useful for adding a high level description or pre-conditions for the test.
3. **Test Details**: This section allows you to enter detailed test steps, any accompanying test data and expected results. Test steps can be reordered, cloned or deleted.
4. **Test Executions**: This section shows the details of all scheduled and previous executions of this test. It also allows you to execute a test.
5. **Attachments**: This section is exactly like the one for any other issue-type
6. **Issue Links**: This section is exactly like the one for any other issue-type



Verify the monthly cash back offer applicable

Priority: Medium
Resolution: Unresolved
Labels: None

Reporter: Quality Thought
Votes:
Watchers:

Quality Thought
agile@qualitythought.in
3:28 AM - Wednesday - Kolkata

Description

In this site user has an offer, that if user purchase more than rs.5000/- , they can get 50% cash back . its applicable monthly once only.

Dates

Created: 16/May/16 11:46 AM
Updated: 16/May/16 11:46 AM

Agile

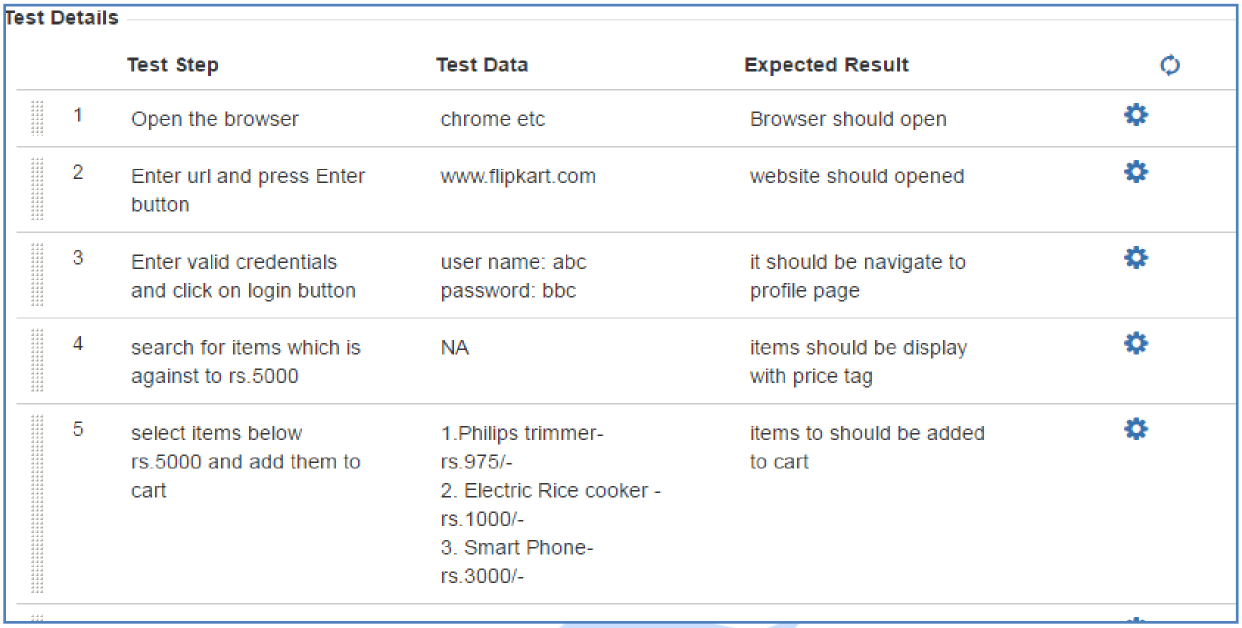
Test Details

	Test Step	Test Data	Expected Result
1	Open the browser	chrome etc	Browser should open
2	Enter url and press Enter button	www.flipkart.com	website should opened
3	Enter valid credentials and click on login button	user name: abc password: bbc	it should be navigate to profile page
4	search for items which is against to rs.5000	NA	items should be display with price tag
5	select items below rs.5000 and add them to cart	1.Philips trimmer- rs.975/- 2. Electric Rice cooker -	items to should be added to cart

HipChat discussions

Do you want to discuss this issue? Connect to HipChat.

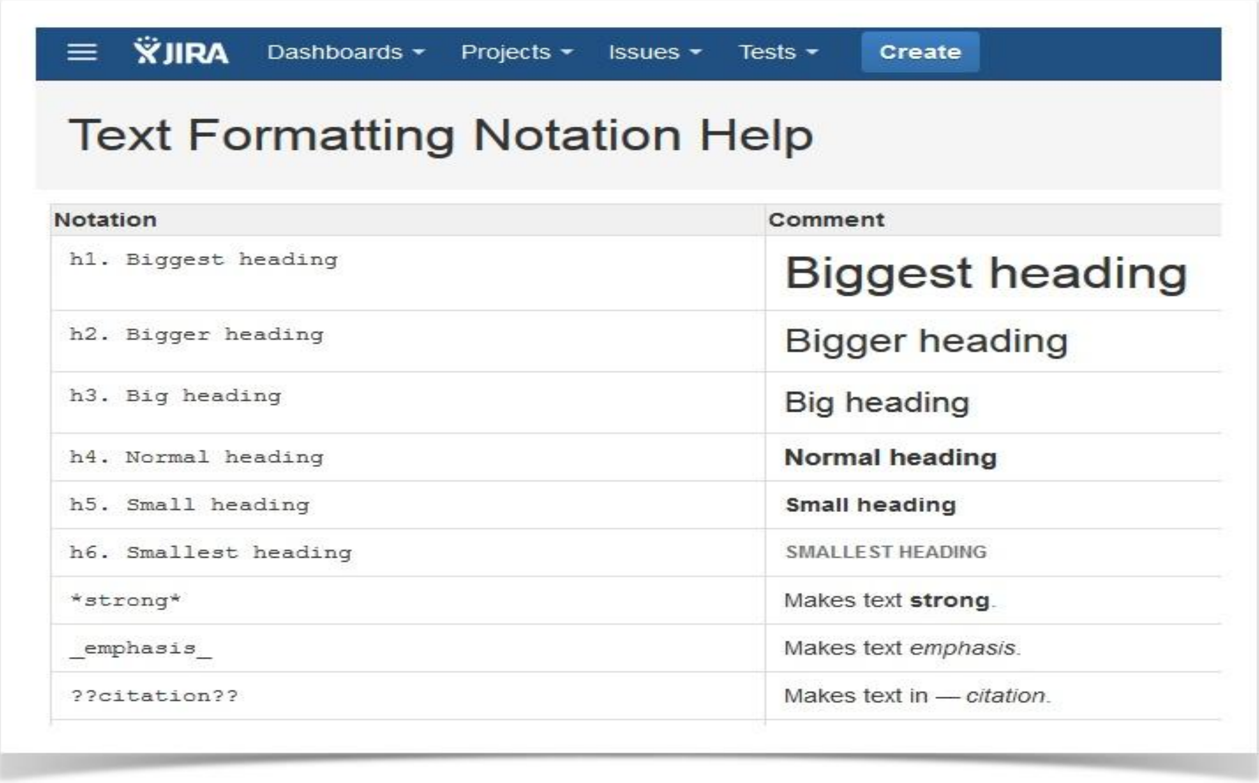
Clicking on the preview icon shows this:



Test Details

	Test Step	Test Data	Expected Result
1	Open the browser	chrome etc	Browser should open
2	Enter url and press Enter button	www.flipkart.com	website should opened
3	Enter valid credentials and click on login button	user name: abc password: bbc	it should be navigate to profile page
4	search for items which is against to rs.5000	NA	items should be display with price tag
5	select items below rs.5000 and add them to cart	1.Philips trimmer-rs.975/- 2. Electric Rice cooker - rs.1000/- 3. Smart Phone- rs.3000/-	items to should be added to cart

Click on the "?" icon to get a list of available markup options.

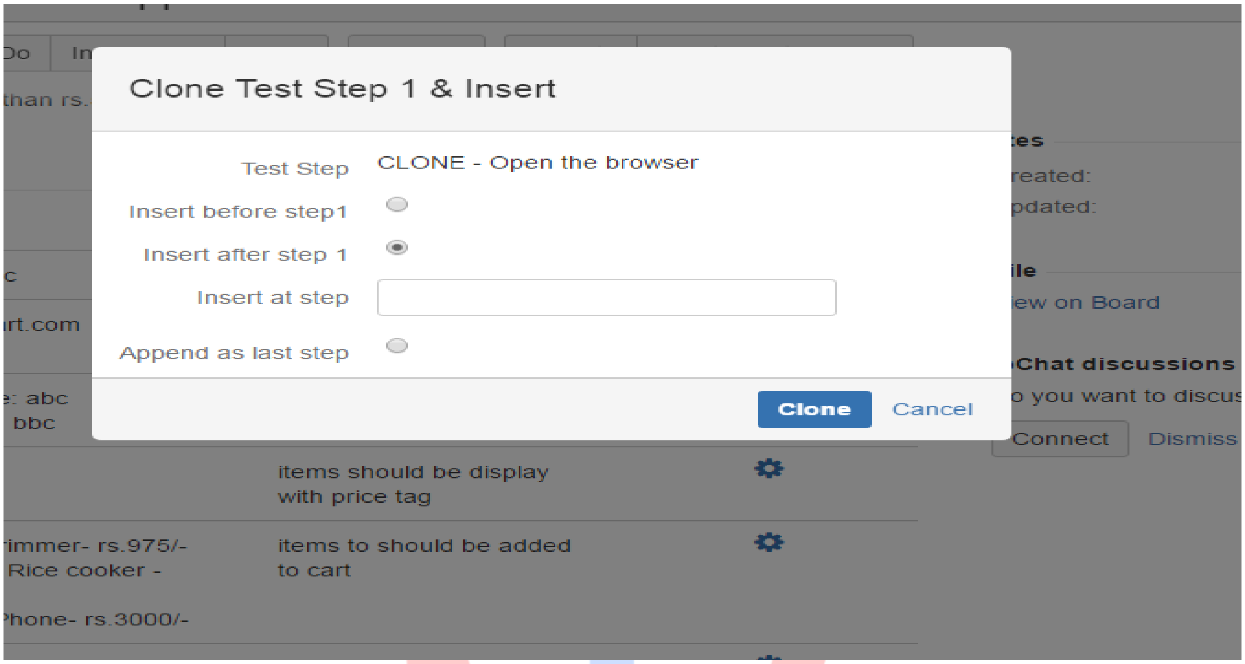


Text Formatting Notation Help

Notation	Comment
h1. Biggest heading	Biggest heading
h2. Bigger heading	Bigger heading
h3. Big heading	Big heading
h4. Normal heading	Normal heading
h5. Small heading	Small heading
h6. Smallest heading	SMALLEST HEADING
strong	Makes text **strong**.
emphasis	Makes text *emphasis*.
??citation??	Makes text in — *citation*.

Cloning Test Steps

To speed up the test creation process, test steps can be easily cloned. Selecting the step to be cloned and clicking on the gear icon on the right of it, shows a drop-down with "Clone" and "Delete" options.



Clone Test Step 1 & Insert

Test Step CLONE - Open the browser

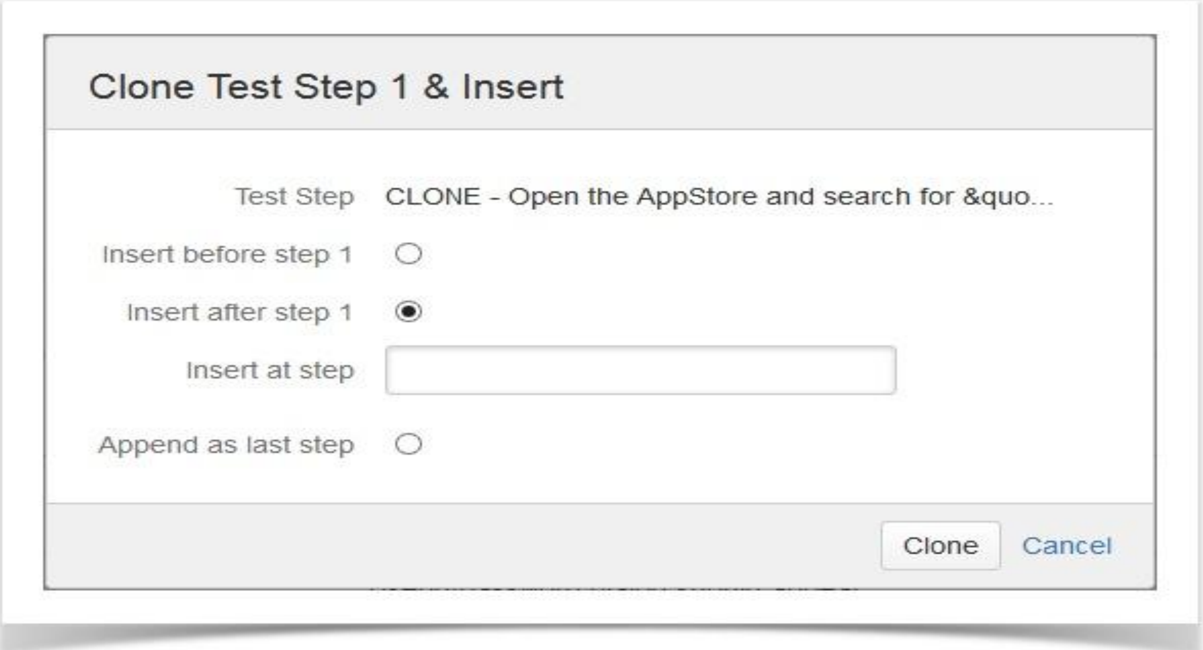
Insert before step1 ○

Insert after step 1 ●

Insert at step

Append as last step ○

Clone
Cancel
items should be display with price tag
items to should be added to cart
Chat discussions
Connect
Dismiss



Clone Test Step 1 & Insert

Test Step CLONE - Open the AppStore and search for &quo...

Insert before step 1 ○

Insert after step 1 ●

Insert at step

Append as last step ○

Search Tests

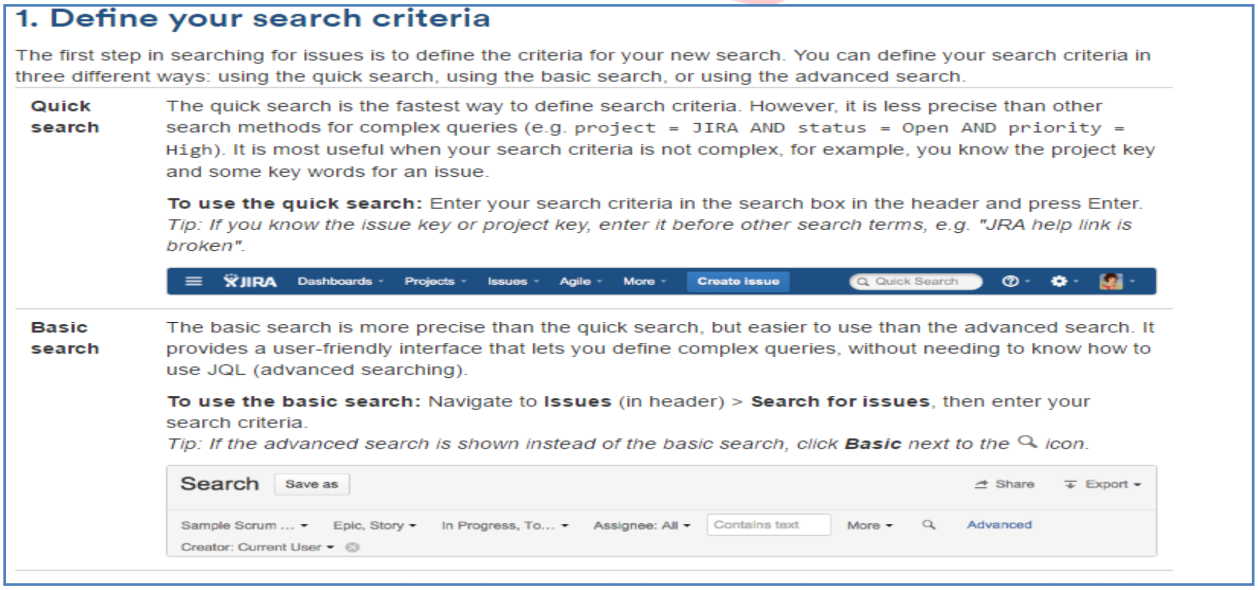
JIRA has a very powerful issue search facility and since a test is an issue-type, all the different mechanisms that you can use to search for issues, apply to tests too. On selecting the "Search Tests" menu option, you are taken to JIRA's Issue Navigator with the "Test" issue-type pre-selected thereby allowing you to search for tests

If you frequently run the same search, you can save the search criteria as a filter. This saves you from having to manually redefine the search criteria every time. JIRA applications also include a number of predefined system filters for common queries, such as 'My Open Issues', 'Reported by Me', 'Recently Viewed', and 'All Issues'.

To save your search as a filter: On the search results page, click **Save as** and enter a name for the filter. Your new filter will be shown in the left panel with your other favorite filters, filters shared with you, and the system filters. To run a filter, just click it.

Read the following related topics:

- ✓ Quick searching
- ✓ Basic searching
- ✓ Advanced searching
- ✓ Saving your search as a filter
- ✓ Working with search results



1. Define your search criteria

The first step in searching for issues is to define the criteria for your new search. You can define your search criteria in three different ways: using the quick search, using the basic search, or using the advanced search.

Quick search

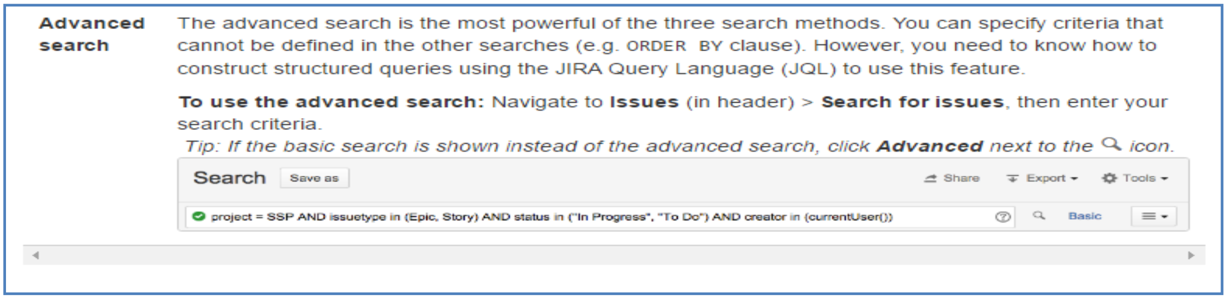
The quick search is the fastest way to define search criteria. However, it is less precise than other search methods for complex queries (e.g. `project = JIRA AND status = Open AND priority = High`). It is most useful when your search criteria is not complex, for example, you know the project key and some key words for an issue.

To use the quick search: Enter your search criteria in the search box in the header and press Enter.
Tip: If you know the issue key or project key, enter it before other search terms, e.g. "JRA help link is broken".

Basic search

The basic search is more precise than the quick search, but easier to use than the advanced search. It provides a user-friendly interface that lets you define complex queries, without needing to know how to use JQL (advanced searching).

To use the basic search: Navigate to **Issues** (in header) > **Search for issues**, then enter your search criteria.
*Tip: If the advanced search is shown instead of the basic search, click **Basic** next to the 🔍 icon.*



Advanced search

The advanced search is the most powerful of the three search methods. You can specify criteria that cannot be defined in the other searches (e.g. ORDER BY clause). However, you need to know how to construct structured queries using the JIRA Query Language (JQL) to use this feature.

To use the advanced search: Navigate to **Issues** (in header) > **Search for issues**, then enter your search criteria.

Tip: If the basic search is shown instead of the advanced search, click ***Advanced*** *next to the* 🔍 *icon.*

project = SSP AND issuetype in (Epic, Story) AND status in ("In Progress", "To Do") AND creator in (currentUser())

Working with the search results

You've got the search results displaying the way that you want. Now you can work with the actual issues in the search results. The issue navigator lets you action individual issues, as well as the entire set of issues returned by your search.

Individual issues:

- ✓ **View the issue:** Click the issue key or name.
- ✓ **Action individual issues**: Click the cog icon next to the issue row and select an option.
- ✓ All issues in the search results:
- ✓ **Export the search results to different formats, like Excel and XML:** Click **Export** and select the desired format.
- ✓ **Share the search results:** Click **Share**, then enter the recipient's details.
- ✓ **Create an RSS feed:** Click **Export > RSS (Issues)** or **RSS (Comments)**.
- ✓ **Bulk modify issues in search results**: Click **Tools** and select **all <n> issue(s)** under **Bulk Change**.

Save your search

If you frequently run the same search, you can save the search criteria as a filter. This saves you from having to manually redefine the search criteria every time. JIRA applications also include a number of predefined system filters for common queries, such as 'My Open Issues', 'Reported by Me', 'Recently Viewed', and 'All Issues'.

To save your search as a filter: On the search results page, click **Save as** and enter a name for the filter. Your new filter will be shown in the left panel with your other favorite filters, filters shared with you, and the system filters. To run a filter, just click it.

Planning Test Cycles

Overview

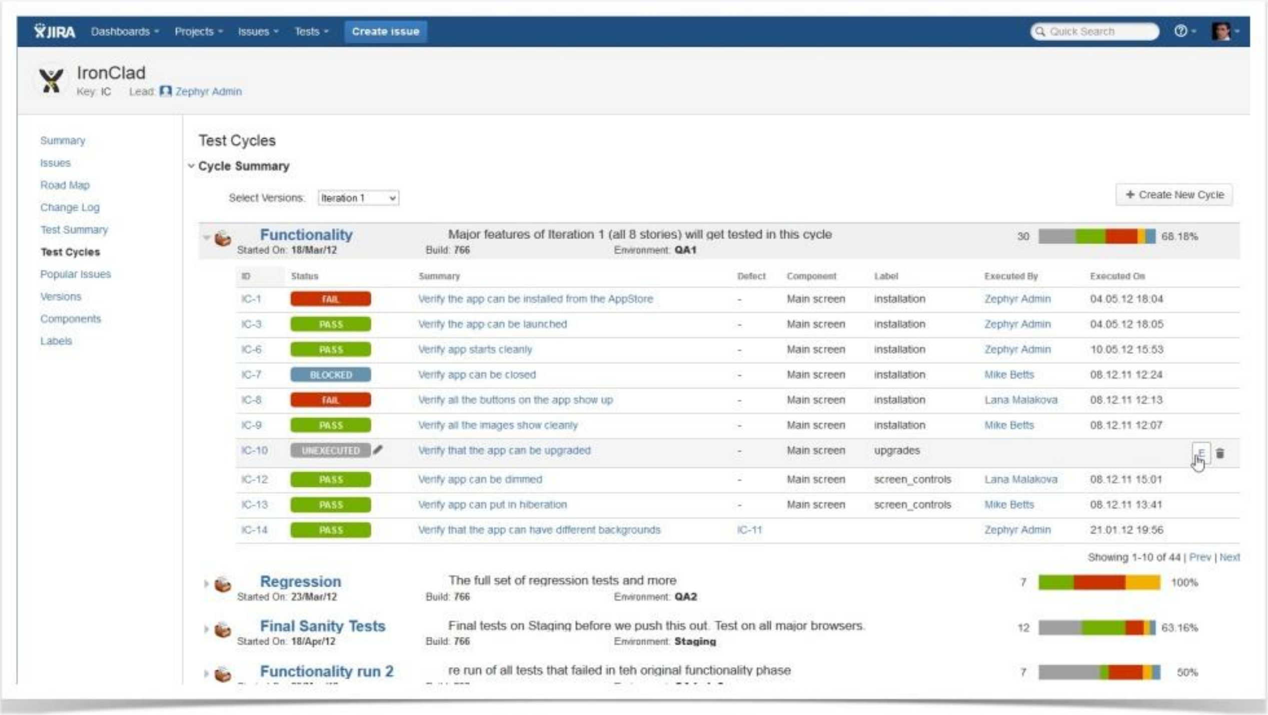
Planning Test Cycles and executing tests as part of those cycles is very powerful yet easy to set up.

Based on your Test Process, you could have a very simple flow where you could create and execute tests in an ad hoc manner, but if you have any basic execution planning in place or have a more elaborate structured test process, then planning your test execution ahead of time is important. Test Cycles are used to set up and execute tests in a structured manner. This allows tests to be grouped logically and executed in a structured fashion. It also allows for the tracking of progress and reporting of quality metrics.

For a given project, the Test Cycles page can be accessed from the navigation tab on the left or from the "Plan Test Cycles" menu item in the top-level "Tests" menu. For each selected version, a list of scheduled test cycles are shown. The "Ad hoc" cycle is default in every version. If no version exists for the project, then it is listed as "Unscheduled", and will also have an "Ad hoc" cycle.

The progress tracking status bar provides a quick view on how many tests have been executed, what % of the overall testing has been completed for a test cycle and on hovering over the different status colors, the # and % of tests in that status (e.g. PASSED, FAILED etc.)

Clicking on any test cycle displays a list of tests for that cycle, which can then be executed or viewed.



IronClad
Key: IC Lead: Zephyr Admin

Test Cycles

Cycle Summary

Select Versions: Iteration 1

Functionality Major features of Iteration 1 (all 8 stories) will get tested in this cycle 30 68.18%
Started On: 18/Mar/12 Build: 766 Environment: QA1

ID	Status	Summary	Defect	Component	Label	Executed By	Executed On
IC-1	FAIL	Verify the app can be installed from the AppStore	-	Main screen	installation	Zephyr Admin	04.05.12 18:04
IC-3	PASS	Verify the app can be launched	-	Main screen	installation	Zephyr Admin	04.05.12 18:05
IC-6	PASS	Verify app starts cleanly	-	Main screen	installation	Zephyr Admin	10.05.12 15:53
IC-7	BLOCKED	Verify app can be closed	-	Main screen	installation	Mike Betts	08.12.11 12:24
IC-8	FAIL	Verify all the buttons on the app show up	-	Main screen	installation	Lana Malakova	08.12.11 12:13
IC-9	PASS	Verify all the images show cleanly	-	Main screen	installation	Mike Betts	08.12.11 12:07
IC-10	UNEXECUTED	Verify that the app can be upgraded	-	Main screen	upgrades		
IC-12	PASS	Verify app can be dimmed	-	Main screen	screen_controls	Lana Malakova	08.12.11 15:01
IC-13	PASS	Verify app can put in hiberation	-	Main screen	screen_controls	Mike Betts	08.12.11 13:41
IC-14	PASS	Verify that the app can have different backgrounds	IC-11			Zephyr Admin	21.01.12 19:56

Showing 1-10 of 44 | Prev | Next

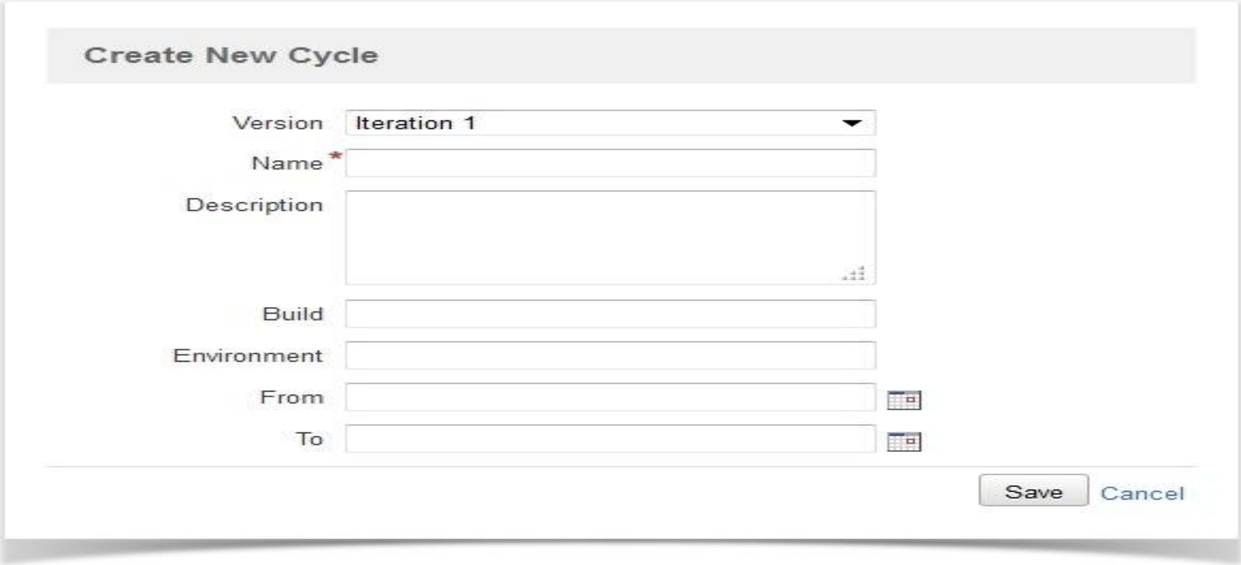
Regression The full set of regression tests and more 7 100%
Started On: 23/Mar/12 Build: 766 Environment: QA2

Final Sanity Tests Final tests on Staging before we push this out. Test on all major browsers. 12 63.16%
Started On: 18/Apr/12 Build: 766 Environment: Staging

Functionality run 2 re run of all tests that failed in teh original functionality phase 7 50%

Creating a Test Cycle

A new test cycle can be created for a selected version by clicking on the "Create New Cycle" button that displays the following dialog:



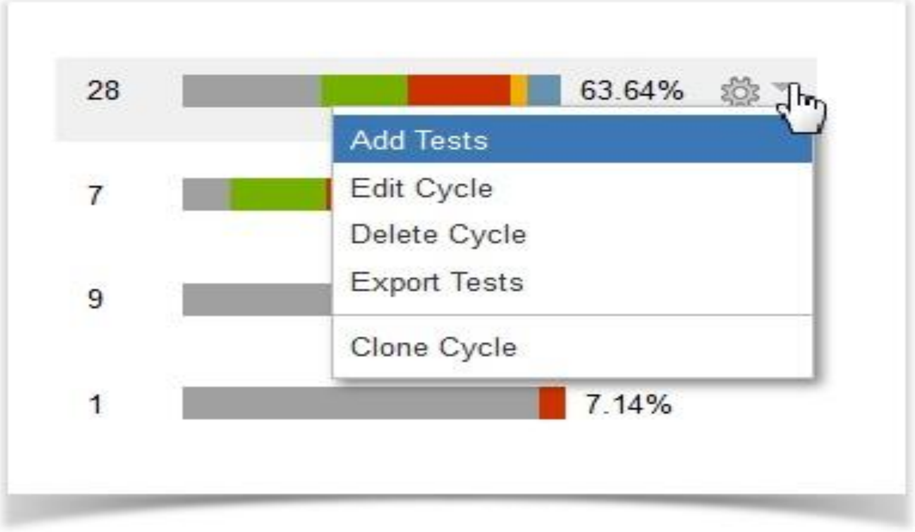
Create New Cycle

Version	Iteration 1
Name*	
Description	
Build	
Environment	
From	
To	

- ✓ **Version**: Shows the list of versions that exist for this project
- ✓ **Name**: Name of the test cycle - this is mandatory

- ✓ **Description**: A description of the cycle
- ✓ **Build**: This is informational only
- ✓ **Environment**: This is informational only
- ✓ **From**: The start date of this test cycle
- ✓ **To**: The end date of this test cycle. The end date is not enforced (i.e. you can continue adding tests to the test cycle and executing them beyond this date)

Once a test cycle has been created, it can be edited or deleted via menu items in the drop-down list at the right of the progress bar for that cycle, as below. You can also add tests to it.



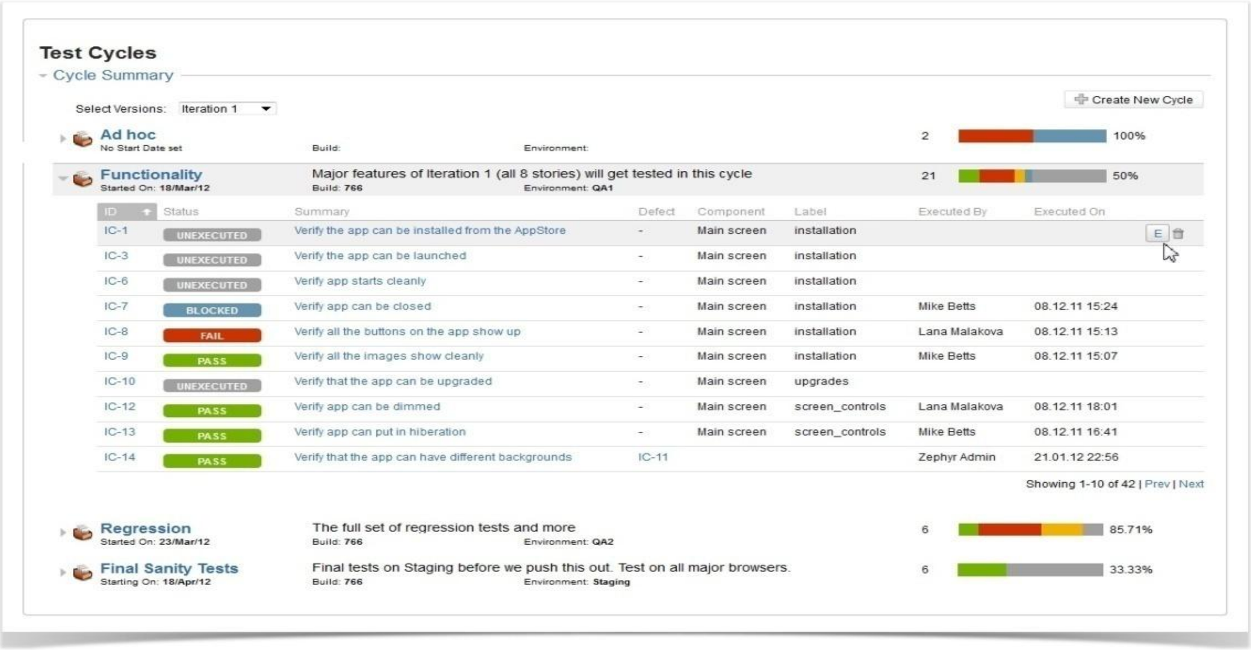
Warning

Use the "Delete Cycle" option carefully. Once deleted, all the test executions, comments, attachments and bug linkages are deleted. Test Execution metrics are affected too. This does not delete the actual tests.

Viewing a Test Cycle and adding Tests

Clicking on any test cycle displays a list of tests for that cycle that can then be executed or viewed. The Fixed Version/s field of a test does not automatically bring it into a test cycle for that version. It still has to be explicitly added to a test cycle for execution.

Once a test has been added to a cycle - either for future execution or as part of the execution - it is displayed under the cycle. It can be executed from there by clicking on the "E" button.



Test Cycles

Cycle Summary

Select Versions: Iteration 1

Create New Cycle

Ad hoc
No Start Date set
Build:
Environment:
2 100%

Functionality
Started On: 18/Mar/12
Major features of Iteration 1 (all 8 stories) will get tested in this cycle
Build: 766
Environment: QA1
21 50%

ID	Status	Summary	Defect	Component	Label	Executed By	Executed On
IC-1	UNEXECUTED	Verify the app can be installed from the AppStore	-	Main screen	installation		
IC-3	UNEXECUTED	Verify the app can be launched	-	Main screen	installation		
IC-6	UNEXECUTED	Verify app starts cleanly	-	Main screen	installation		
IC-7	BLOCKED	Verify app can be closed	-	Main screen	installation	Mike Betts	08.12.11 15:24
IC-8	FAIL	Verify all the buttons on the app show up	-	Main screen	installation	Lana Malakova	08.12.11 15:13
IC-9	PASS	Verify all the images show cleanly	-	Main screen	installation	Mike Betts	08.12.11 15:07
IC-10	UNEXECUTED	Verify that the app can be upgraded	-	Main screen	upgrades		
IC-12	PASS	Verify app can be dimmed	-	Main screen	screen_controls	Lana Malakova	08.12.11 18:01
IC-13	PASS	Verify app can put in hiberation	-	Main screen	screen_controls	Mike Betts	08.12.11 16:41
IC-14	PASS	Verify that the app can have different backgrounds	IC-11			Zephyr Admin	21.01.12 22:56

Showing 1-10 of 42 | Prev | Next

Regression
Started On: 23/Mar/12
The full set of regression tests and more
Build: 766
Environment: QA2
6 85.71%

Final Sanity Tests
Starting On: 18/Apr/12
Final tests on Staging before we push this out. Test on all major browsers.
Build: 766
Environment: Staging
6 33.33%

Cloning Test Cycles

As the name implies, you can clone an existing test cycle and do so very easily. Cloning essentially copies an existing test cycle and all the tests that it contains. It zeroes out all execution status, comments, attachments and defects filed. This is a great way to very quickly set up a test cycle based on planning that was previously done and is a great time saver.

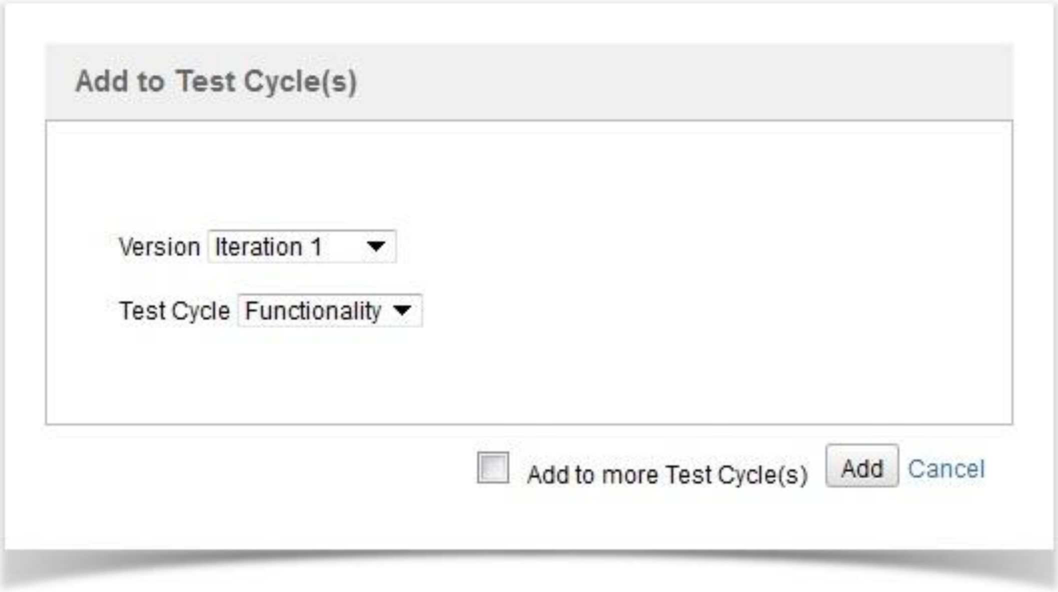
From the drop-down menu at the right of the progress bar for that cycle, as above, select "Clone Cycle". This clones that entire test cycle for the current version you are in and gives you an opportunity to modify any of its details.

Adding a Test(s) to a Test Cycle

There are many ways to add a test to a Test Cycle.

1. When viewing a Test

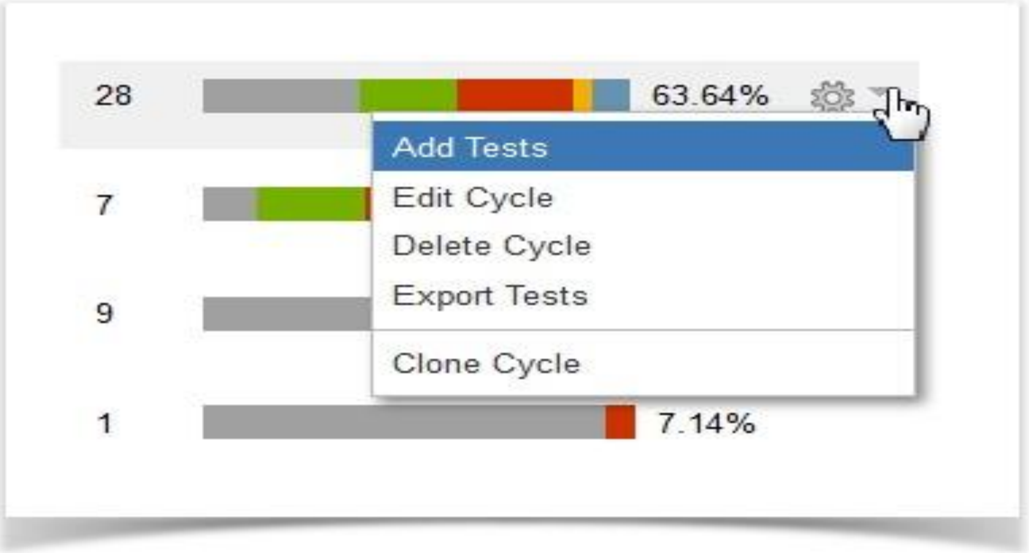
When you view a test, you can immediately add it to any Test Cycle. The Fixed Version/s field of a test does not automatically bring it into a test cycle for that version. It has to be explicitly added to an existing test cycle for that version to be executed. This can be done when viewing a test by selecting the "Add to Test Cycle(s)" item in the menu which then displays this dialog:



The same test can be added to more than one test cycle by checking the box "Add to more Test Cycle(s)". These additions then show up in the Test Executions section of the page as well as in the Test Cycles page and it can then be executed from either place.

2. When viewing a Test Cycle

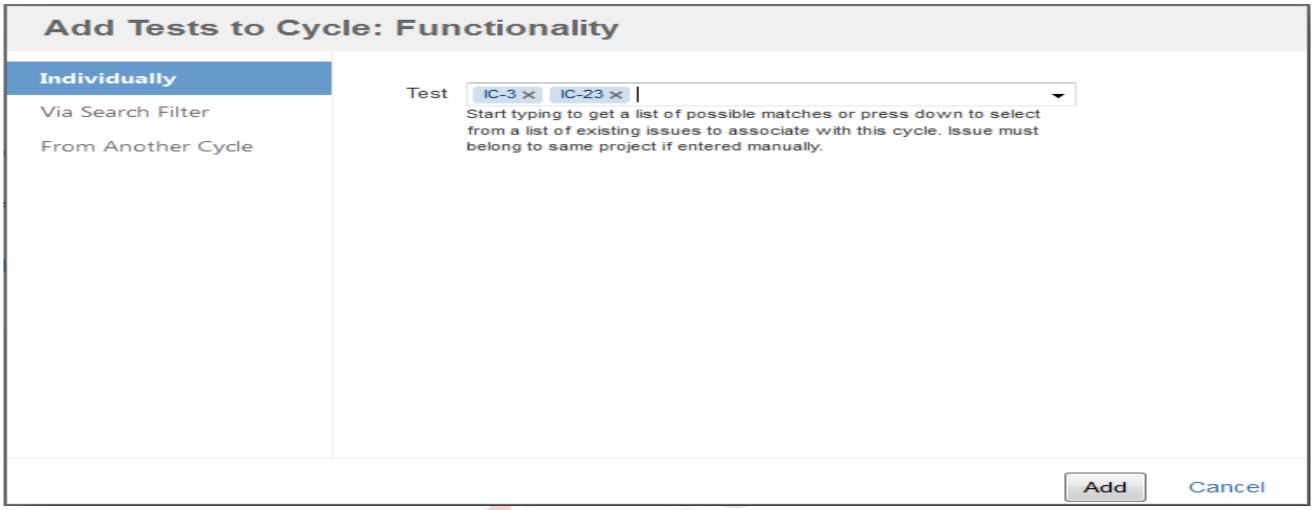
You can also add tests to a Test Cycle from the drop-down menu next to the cycle as below.



There are 3 ways you can add tests to that cycle:

A. Individually

Here you can add the ID of the test(s) and these then get added to the Test Cycle. If you do not know the issue ID, you can start typing keywords in that box and a list of possible matches will be shown, from which you can select your test(s).



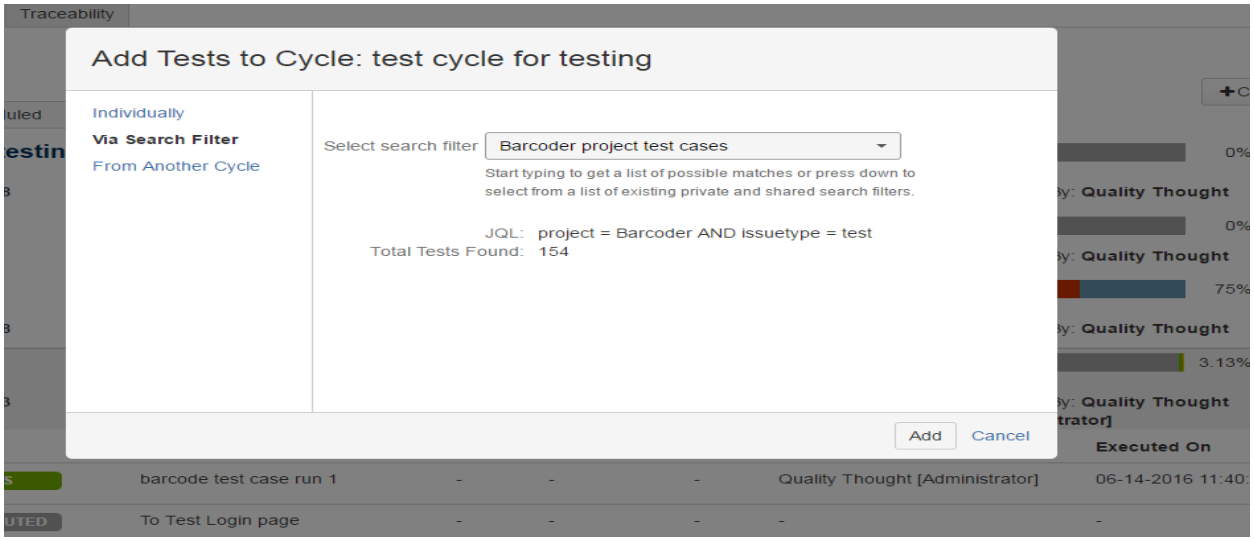
Add Tests to Cycle: Functionality

Test: IC-3 × IC-23 ×

Start typing to get a list of possible matches or press down to select from a list of existing issues to associate with this cycle. Issue must belong to same project if entered manually.

B. Via a saved search filter

A quicker, easier and more powerful way to add tests to any Test Cycle is by using JIRA's native search filters. In the Issue Navigator, you can create and save your search filters. This list of favorite filters is now shown in a drop-down and on selecting that filter, all the tests found by running that search can now be easily added to this Test Cycle with just one click. Adding a particular filter again or adding multiple filters that reference the same test will not do any damage as a particular test can be only added once to a Test Cycle. Note that irrespective of all the issue-types that show up in your search result, only issues of type "test" will be added to the test cycle.



Add Tests to Cycle: test cycle for testing

Select search filter: Barcoder project test cases

Start typing to get a list of possible matches or press down to select from a list of existing private and shared search filters.

JQL: project = Barcoder AND issuetype = test
Total Tests Found: 154

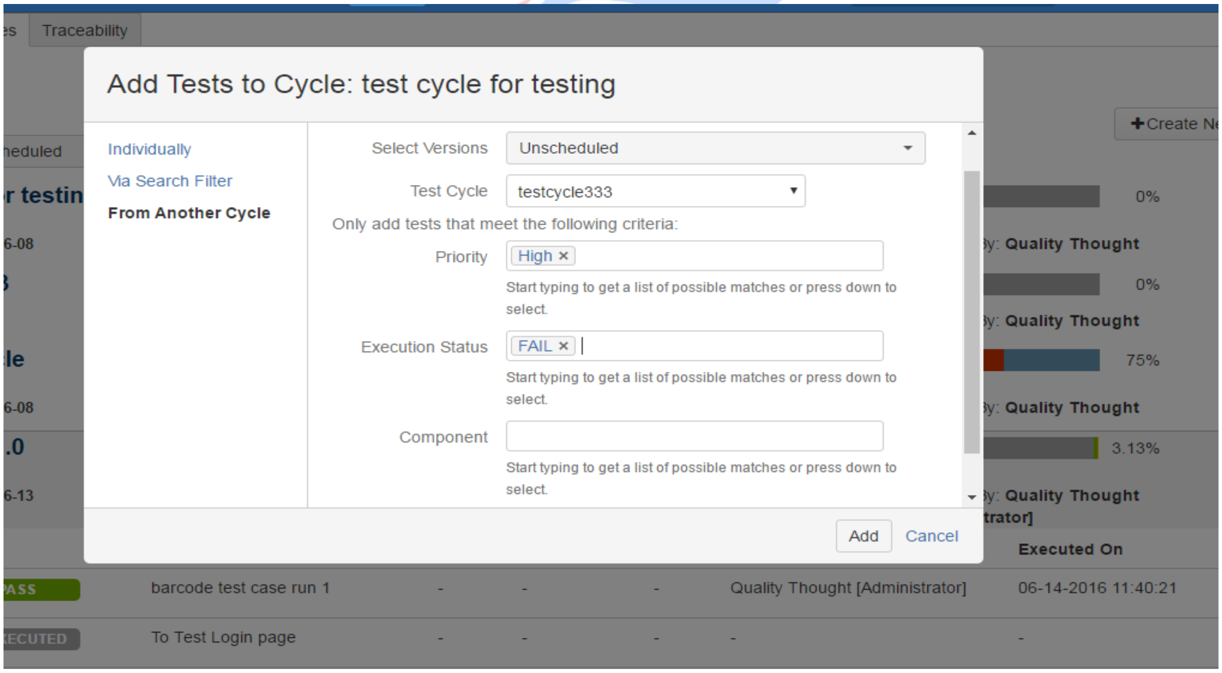
barcode test case run 1 - - - Quality Thought [Administrator] 06-14-2016 11:40:

To Test Login page - - - - -

C. From another Test Cycle

You can also add tests from another cycle. If you want to add ALL the tests from a previous cycle, you might be better off cloning it. If you want to copy a group of tests from a previous test cycle that meet certain criteria, then this dialog provides the options to allow you to do that. You can pick a test cycle from any version and add tests that meet the criteria you specify for Priority, Execution Status, Component and/or Label. You can add more than one value too.This makes it really easy to build test cycles off of previous ones, allowing you to refine your grouping, set up cycles for re-test, reuse certain portions of a cycle, re-run your failed tests etc.

On top of that you can specify to bring over only the tests from a previous cycle that have linked defects to it and furthermore, specify what their resolution statuses are. For example, you can create a new cycle for regressing/re-testing just the tests that had failed in a previous cycle and had a defects filed against them but have now been fixed.



Add Tests to Cycle: test cycle for testing

Select Versions: Unscheduled

Test Cycle: testcycle333

Only add tests that meet the following criteria:

Priority: High ×

Start typing to get a list of possible matches or press down to select.

Execution Status: FAIL ×

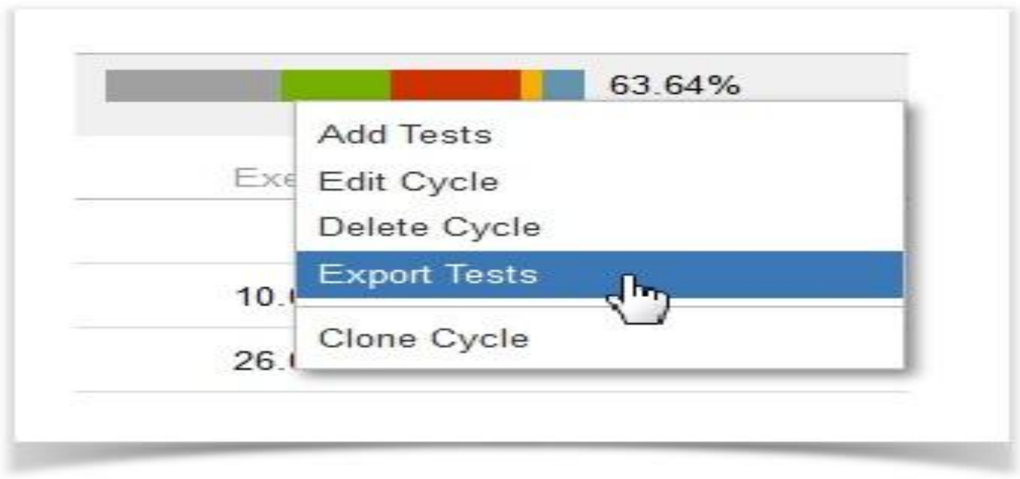
Start typing to get a list of possible matches or press down to select.

Component:

Start typing to get a list of possible matches or press down to select.

Exporting

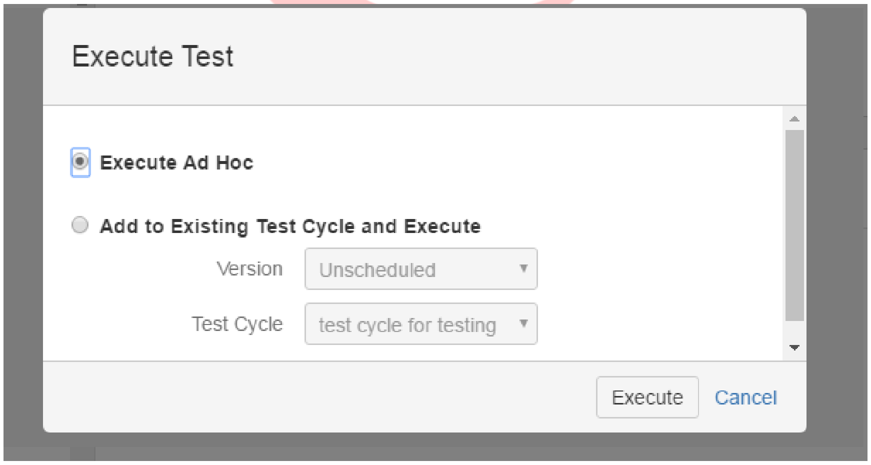
Test Cycles can be exported to a CSV, HTML or XML file very easily. Select a Test Cycle by clicking on its name and from the gear icon on the right, pick the "Export Tests" item from the drop-down menu and the format you want the export in.



The resulting CSV file can be opened in Microsoft Excel, HTML file in any browser and the XML file in any XML editor.

Ad hoc Executions

If a test is being executed outside of a Test Cycle, then it is termed to be executed in an "Ad hoc" fashion. By default, every Version (or if one doesn't exist, "Unscheduled") will have an Ad hoc cycle. All tests executed in an Ad hoc fashion will have the results summary reported in the test cycle.


Execute Test

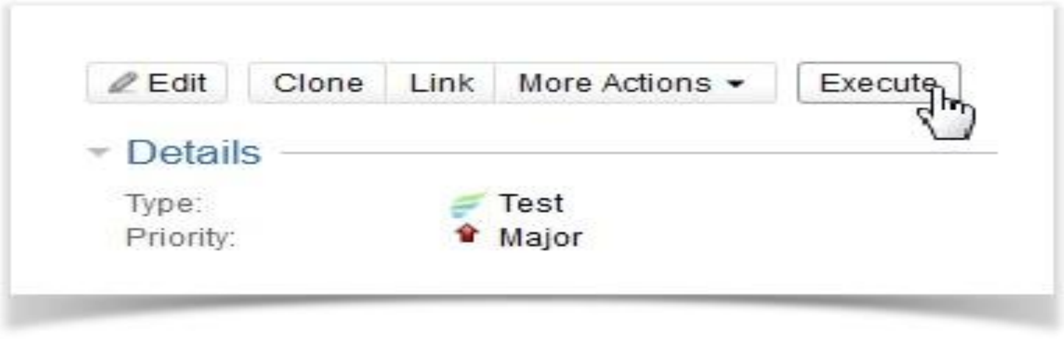
● Execute Ad Hoc

○ Add to Existing Test Cycle and Execute

Version: Unscheduled

Test Cycle: test cycle for testing

Tests can be executed in an Ad hoc manner from the "Execute" button found in the toolbar of every issue.



This opens the "Execute Test" dialog as below where the "Execute Ad Hoc" option is selected by default. You could also select the "Add to Existing Test Cycle and Execute" option, pick a Version and pick the "Ad hoc" option - ensuring that the execution status gets recorded against that particular version albeit in an ad hoc manner

Tests can be executed in a wide variety of ways:

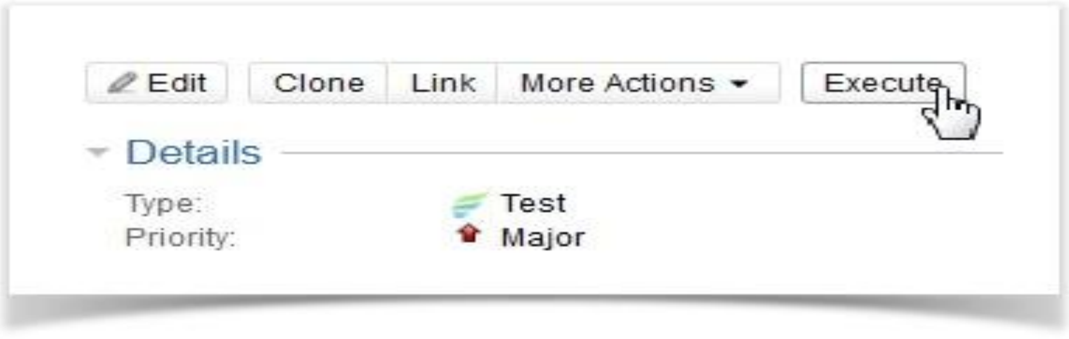
1. Using the "Execute" button while viewing a test
2. From the "Test Execution" section of a test
3. From the "Test Cycle Summary" page

You can also view Execution Details for a test.

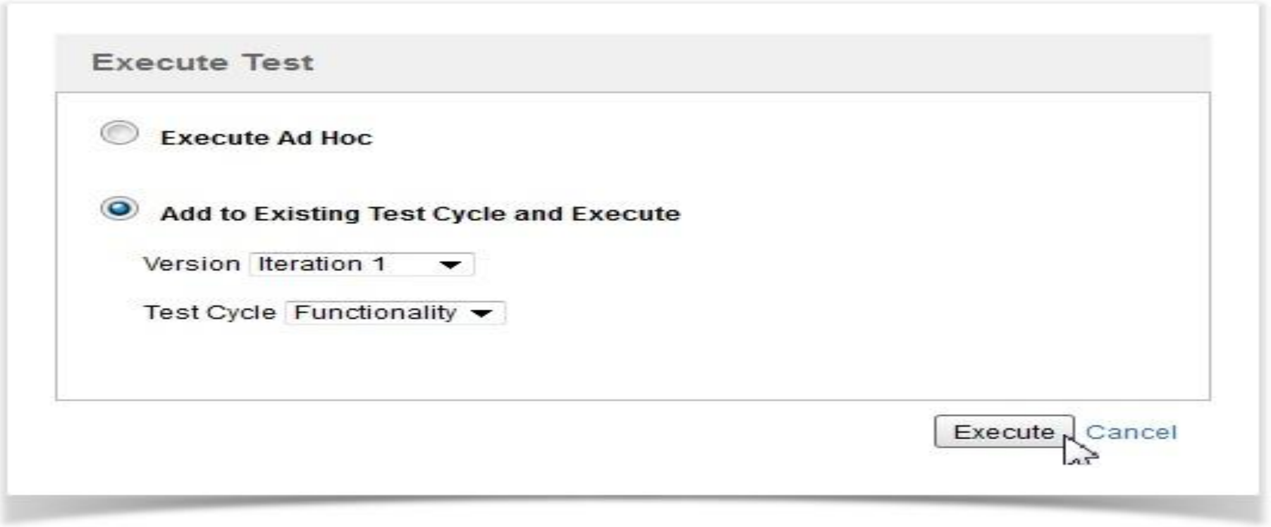
There are 3 ways to execute tests that belong to a test cycle.

1. Using the "Execute" button while viewing a test

You can immediately execute a test while viewing it by clicking on the "Execute" button found in the toolbar.



This opens the "Execute Test" dialog as below, where - just before you actually execute the test - you will want to add it to an existing test cycle. On selecting this option, you can pick the Version (this is the Fix Version/s) and the Test Cycle against which you want the execution results to count. If you cannot find the test cycle against which you want this execution to count, you may have to create a test cycle first.



Execute Test

○ Execute Ad Hoc

● Add to Existing Test Cycle and Execute

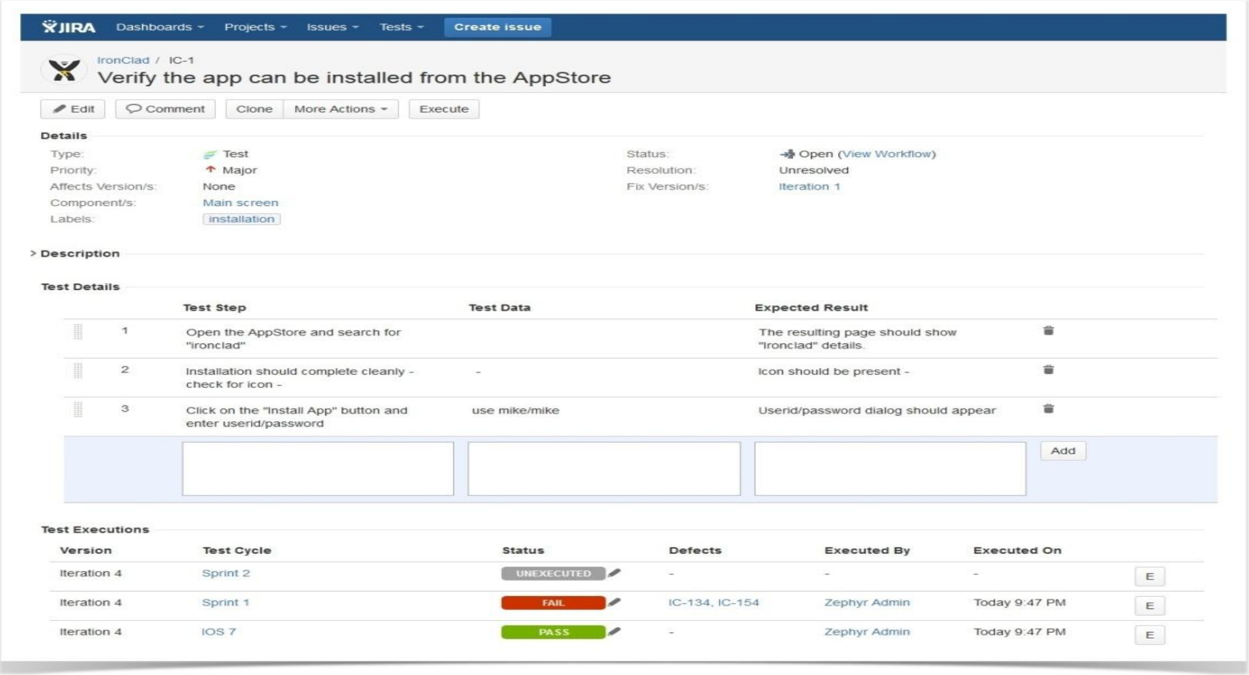
Version: Iteration 1

Test Cycle: Functionality

On clicking the "Execute" button above you can now enter execution details.

2. From the "Test Execution" section of a test

When a test has been added to a Test Cycle, it shows up in the Test Executions section of the test as you are viewing it. See an example below. You can click on the "E" button for any of these tests and enter execution details. You can also re-execute a test from here. Note that if you re-execute a test, the previous status is overwritten.



IronClad / IC-1

Verify the app can be installed from the AppStore

Details

Type:	Test	Status:	Open (View Workflow)
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	Iteration 1
Component/s:	Main screen		
Labels:	installation		

> Description

Test Details

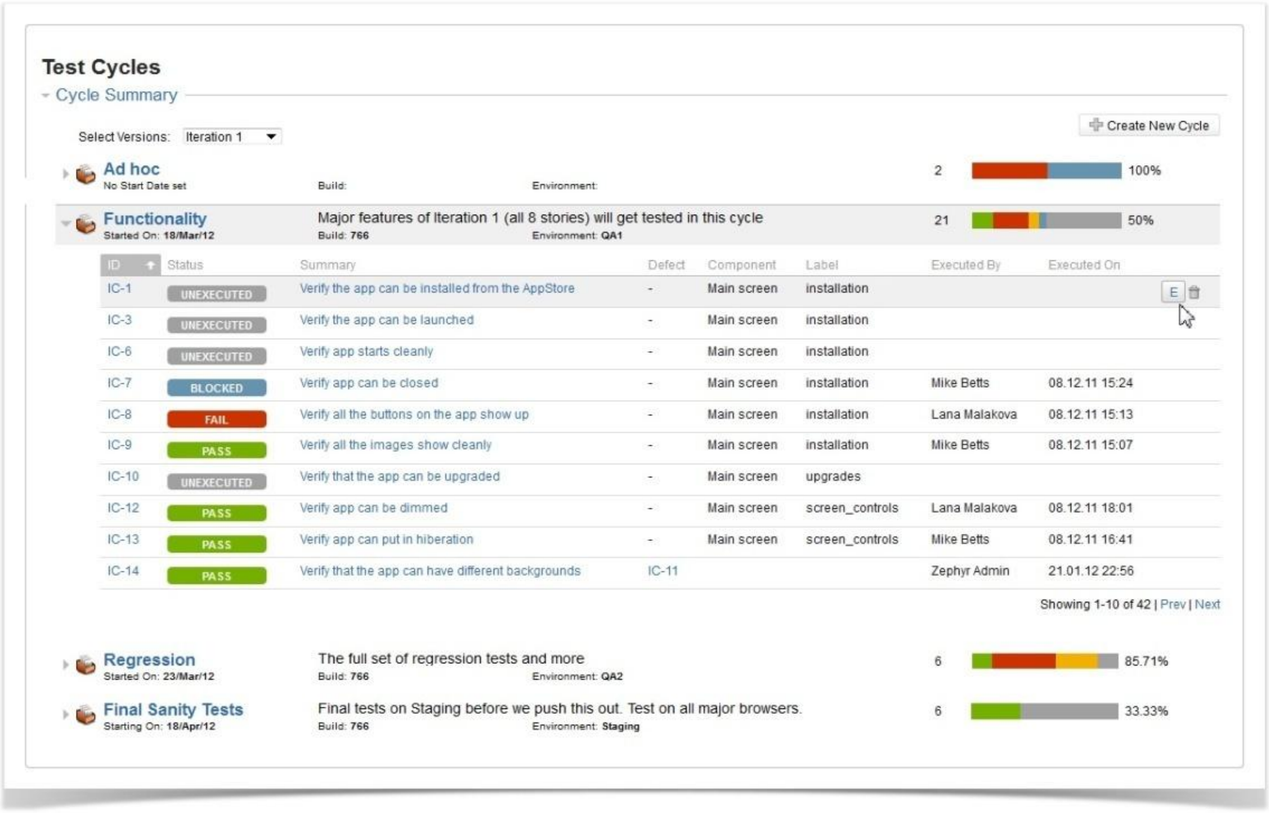
	Test Step	Test Data	Expected Result
1	Open the AppStore and search for "ironclad"		The resulting page should show "Ironclad" details.
2	Installation should complete cleanly - check for icon -	-	Icon should be present -
3	Click on the "Install App" button and enter userid/password	use mike/mike	Userid/password dialog should appear

Test Executions

Version	Test Cycle	Status	Defects	Executed By	Executed On
Iteration 4	Sprint 2	UNEXECUTED	-	-	-
Iteration 4	Sprint 1	FAIL	IC-134, IC-154	Zephyr Admin	Today 9:47 PM
Iteration 4	IOS 7	PASS	-	Zephyr Admin	Today 9:47 PM

3. From the "Test Cycle Summary" page

For the more structured testing process the Test Cycles Summary page is the starting point for executions. This page shows a list of all tests that have been grouped together by Test Cycle for a particular Version. Selecting the appropriate Version displays a list of Test Cycles that have been created for it. Clicking on any one of those Test Cycles shows the list of tests that are part of that cycle. A progress tracking bar on the right provides a quick breakdown of the status of that test cycle.



Test Cycles

Cycle Summary

Select Versions: Iteration 1

Cycle	Description	Count	Progress
Ad hoc No Start Date set	Build: Environment:	2	100%
Functionality Started On: 18/Mar/12	Major features of Iteration 1 (all 8 stories) will get tested in this cycle Build: 766 Environment: QA1	21	50%

ID	Status	Summary	Defect	Component	Label	Executed By	Executed On
IC-1	UNEXECUTED	Verify the app can be installed from the AppStore	-	Main screen	installation		
IC-3	UNEXECUTED	Verify the app can be launched	-	Main screen	installation		
IC-6	UNEXECUTED	Verify app starts cleanly	-	Main screen	installation		
IC-7	BLOCKED	Verify app can be closed	-	Main screen	installation	Mike Betts	08.12.11 15:24
IC-8	FAIL	Verify all the buttons on the app show up	-	Main screen	installation	Lana Malakova	08.12.11 15:13
IC-9	PASS	Verify all the images show cleanly	-	Main screen	installation	Mike Betts	08.12.11 15:07
IC-10	UNEXECUTED	Verify that the app can be upgraded	-	Main screen	upgrades		
IC-12	PASS	Verify app can be dimmed	-	Main screen	screen_controls	Lana Malakova	08.12.11 18:01
IC-13	PASS	Verify app can put in hiberation	-	Main screen	screen_controls	Mike Betts	08.12.11 16:41
IC-14	PASS	Verify that the app can have different backgrounds	IC-11			Zephyr Admin	21.01.12 22:56

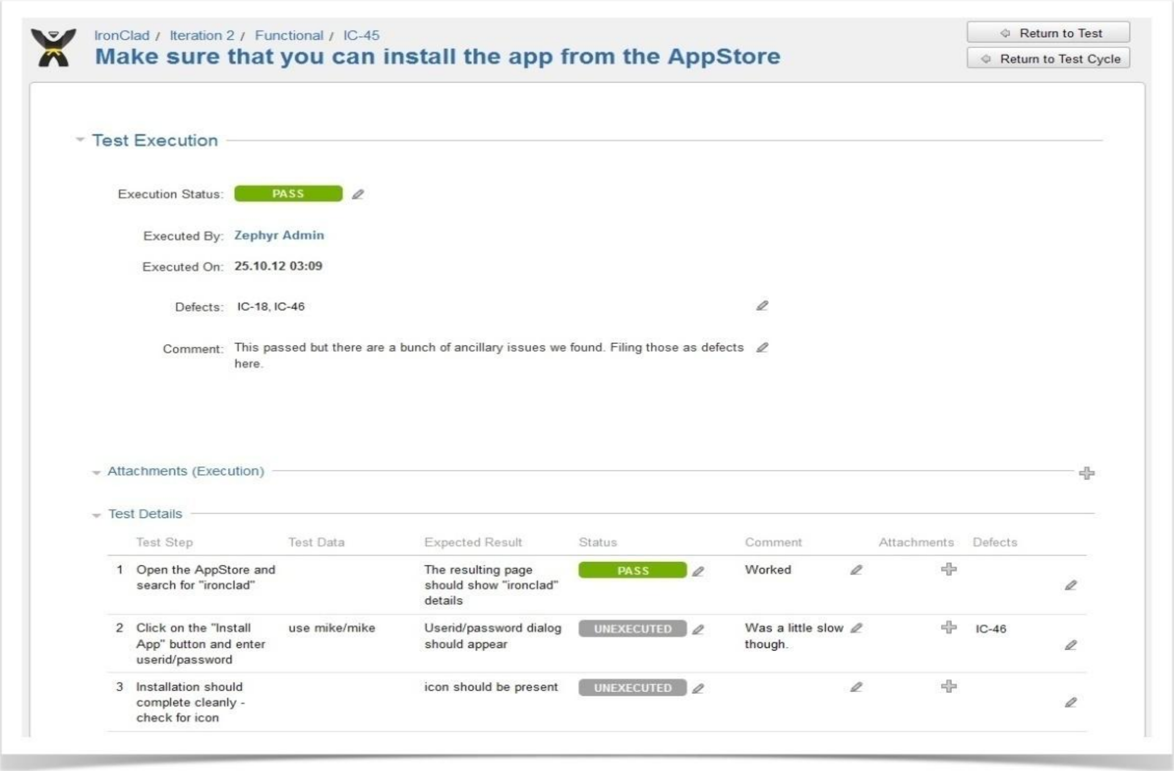
Showing 1-10 of 42 | Prev | Next

Cycle	Description	Count	Progress
Regression Started On: 23/Mar/12	The full set of regression tests and more Build: 766 Environment: QA2	6	85.71%
Final Sanity Tests Starting On: 18/Apr/12	Final tests on Staging before we push this out. Test on all major browsers. Build: 766 Environment: Staging	6	33.33%

Tests are shown in groups of 10 and a navigation bar allows you to jump to the next 10 in the list. The following fields are sortable: ID, Status, Summary, Executed On and Executed By. The "E" button on each row allows for the test to be executed where you can enter execution details.

Execution Details

In all of the above scenarios, clicking on the "Execute" button will open the Execution page where details about the execution can be entered.



IronClad / Iteration 2 / Functional / IC-45

Make sure that you can install the app from the AppStore

Test Execution

Execution Status: PASS

Executed By: Zephyr Admin

Executed On: 25.10.12 03:09

Defects: IC-18, IC-46

Comment: This passed but there are a bunch of ancillary issues we found. Filing those as defects here.

Attachments (Execution)

Test Details

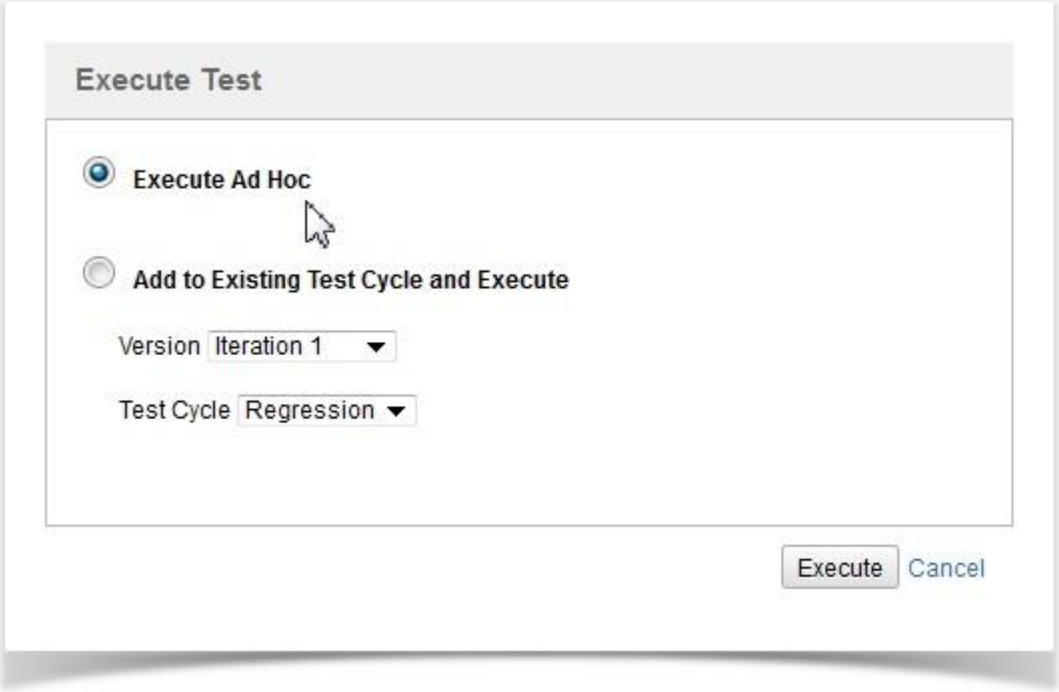
	Test Step	Test Data	Expected Result	Status	Comment	Attachments	Defects
1	Open the AppStore and search for "ironclad"		The resulting page should show "ironclad" details	PASS	Worked		
2	Click on the "Install App" button and enter userid/password	use mike/mike	Userid/password dialog should appear	UNEXECUTED	Was a little slow though.		IC-46
3	Installation should complete cleanly - check for icon		icon should be present	UNEXECUTED			

- ✓ **Execution Status**: Selecting this field, a drop-down list of options are shown. The default statuses are UNEXECUTED, PASS, FAIL, WIP (Work in Progress) and BLOCKED. These statuses can be customized by your JIRA Administrator.
- ✓ **Defects**: A new defect can be created at the Test level and the Test step level by selecting the edit icon for the Defects field and selecting "Create New Issue". A new defect can also be created by clicking on the "Create" button in the main toolbar or by directly typing in text to get a list of possible matches. The drop down also lists existing issues that could be selected to associate with this execution.
- ✓ **Comment**: Details about the test execution can be entered here.
- ✓ **Attachments**: Files can be attached to this execution

You can also do test step-level executions and change the status. You can also provide step level information like Comments, Attachments and link existing Defects at that level.

⚠ Note: The **Execution Status** of the overall test is NOT automatically modified based on the status of the individual test steps. You will still have to set the **Execution Status** of overall test manually.

Once done entering the execution details, click anywhere else to automatically save your information. You can then decide (based on how you got to this execution screen) whether you want to return to the test or return to the test cycle.



Execute Test

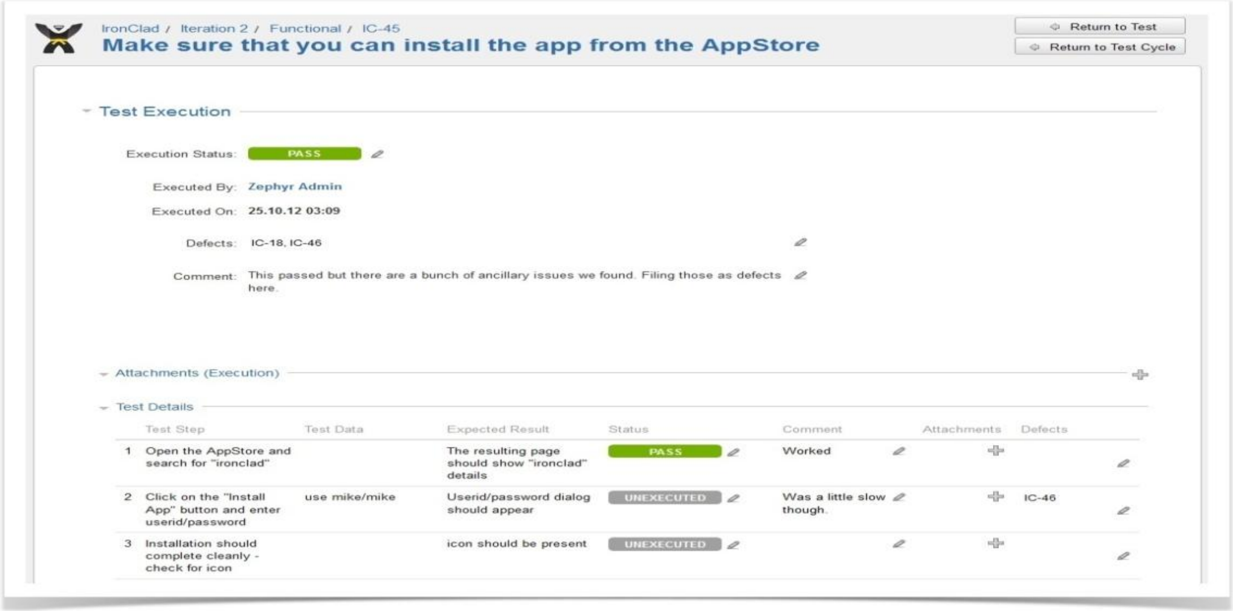
● Execute Ad Hoc

○ Add to Existing Test Cycle and Execute

Version: Iteration 1

Test Cycle: Regression

Clicking on the "Execute" button will open the Execution page where details about the execution can be entered.



IronClad / Iteration 2 / Functional / IC-45

Make sure that you can install the app from the AppStore

Test Execution

Execution Status: PASS

Executed By: Zephyr Admin

Executed On: 25.10.12 03:09

Defects: IC-18, IC-46

Comment: This passed but there are a bunch of ancillary issues we found. Filing those as defects here.

Attachments (Execution)

Test Details

	Test Step	Test Data	Expected Result	Status	Comment	Attachments	Defects
1	Open the AppStore and search for "ironclad"		The resulting page should show "ironclad" details	PASS	Worked		
2	Click on the "Install App" button and enter userid/password	use mike/mike	Userid/password dialog should appear	UNEXECUTED	Was a little slow though.		IC-46
3	Installation should complete cleanly - check for icon		icon should be present	UNEXECUTED			

- ✓ **Execution Status**: Selecting this field, a drop-down list of options are shown. The default statuses are UNEXECUTED, PASS, FAIL, WIP (Work in Progress) and BLOCKED. These statuses can be customized by your JIRA Administrator.
- ✓ **Defects**: A new defect can be created at the Test level and the Test step level by selecting the edit icon for the Defects field and selecting "Create New Issue". A new defect can also be created by clicking on the "Create" button in the main toolbar or by directly typing in text to get a list of possible matches. The drop down also lists existing issues that could be selected to associate with this execution.
- ✓ **Comment**: Details about the test execution can be entered here.
- ✓ **Attachments**: Files can be attached to this execution

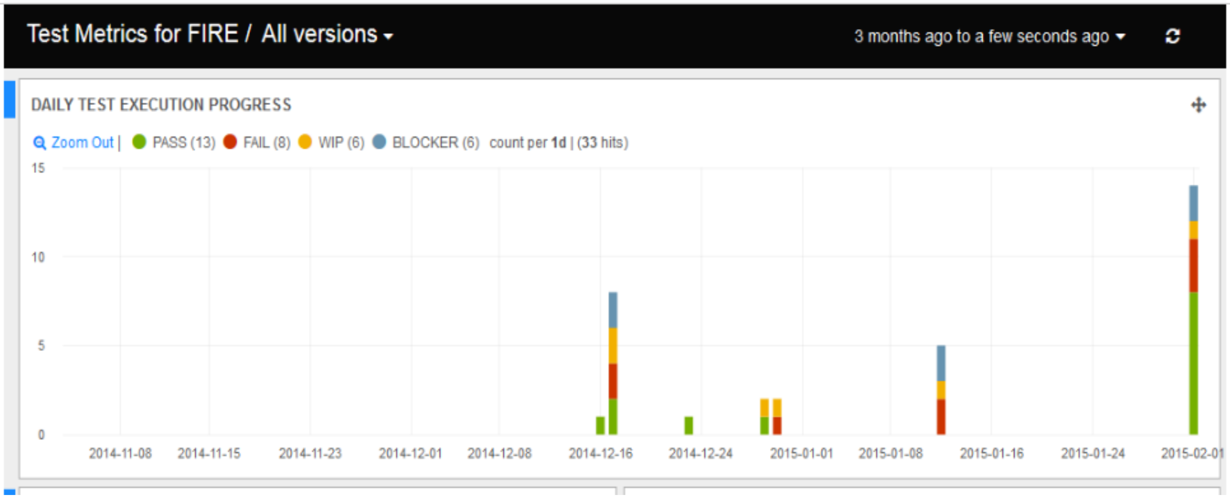
You can also do test step-level executions and change the status. You can also provide step level information like Comments, Attachments and link Defects at that level.

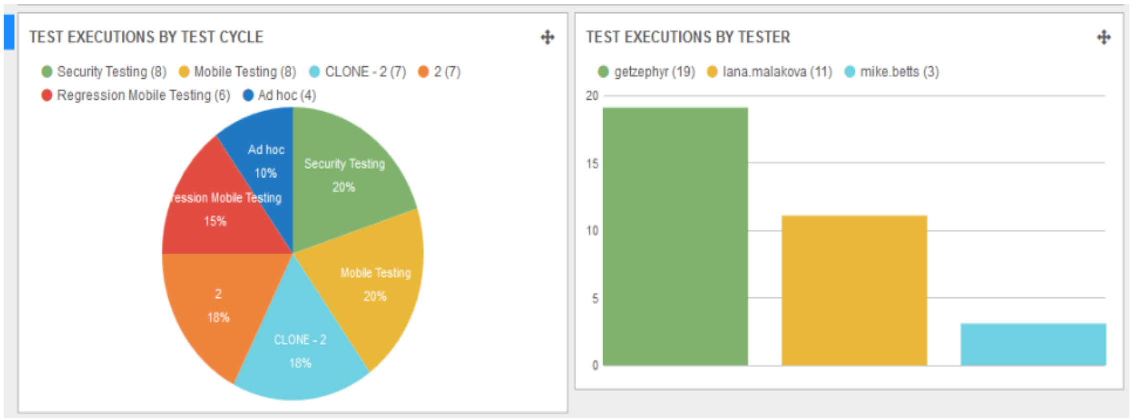
⚠ Note: The **Execution Status** of the overall test is NOT automatically modified based on the status of the individual test steps. You will still have to set the **Execution Status** of overall test manually.

Test Metrics

Testing specific metrics can be viewed via a test dashboard that is displayed for a particular project. This is accessed by clicking on the "Test Metrics" item in the "Tests" menu from the main toolbar.

Multiple test metrics are automatically tracked for every project:





- ✓ Daily Test Execution Progress, by status
- ✓ Test Executions by Test Cycle
- ✓ Test Executions by Tester
- ✓ Number of executions per day
- ✓ And a List of executions with their details

Options

The data displayed in the dashboard can be across an entire project (i.e. including all Versions) or can be specific to a particular Version.

You can also change the data range for which the data is displayed. In the above example, the range is set to "3 months to a few seconds ago". Click on that to pick another option.

A tiny refresh icon allows the data to be refreshed on demand. Auto-refresh options also exist. Pick one of the various available refresh rates by clicking on the range and selecting the "Auto-Refresh" option.

Each panel of this dashboard can be moved around to a different location, during a single session.

Changes made to this dashboard are only remembered for the current session. All defaults are restored when the metrics page is refreshed.

Customizable Test Metrics (Gadgets) and Dashboards

5 different types of metrics can be customized and added to standard JIRA Dashboards. Click here to learn how.

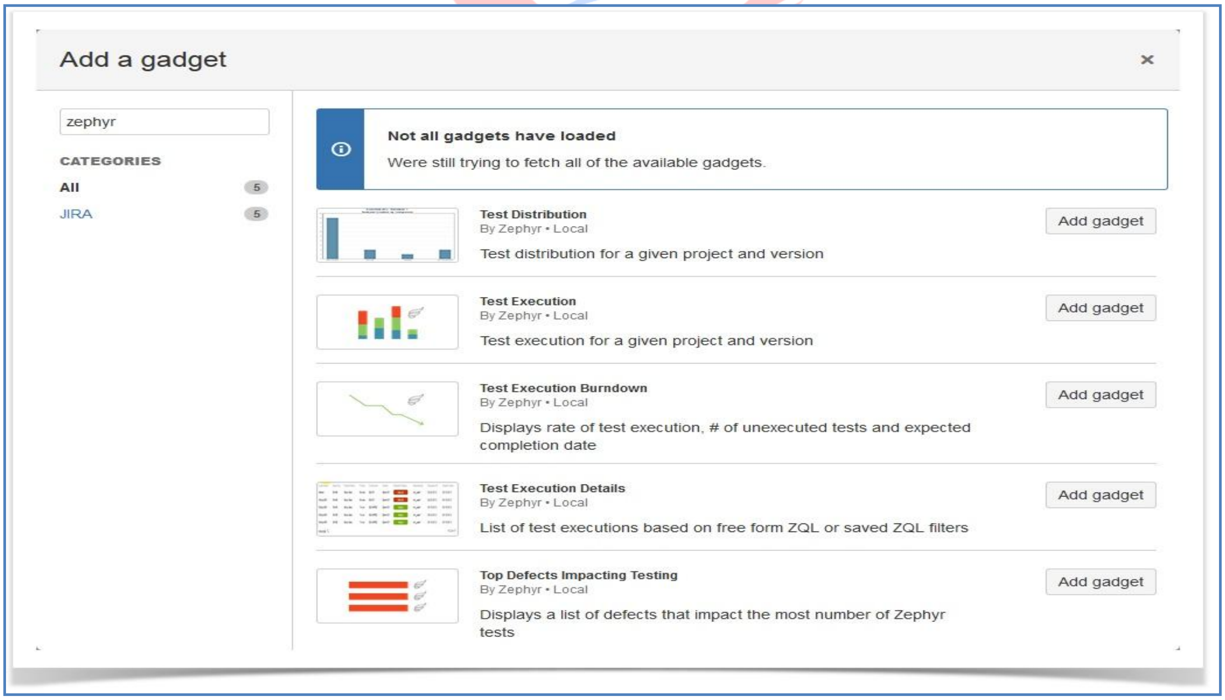
Test Metrics (Gadgets)

Introduction

Testing specific metrics can be viewed via a variety of gadgets that are found in the Gadget Directory. Since **Zephyr for JIRA** uses a standard JIRA issue-type, many out-of-the-box gadgets can be used. The following testing-specific gadgets are available:

- ✓ Test Distribution
- ✓ Test Execution
- ✓ Top Defects Impacting Testing
- ✓ Test Execution Burndown
- ✓ Test Execution Details

Open a dashboard, click on "Add Gadget" and search for "Zephyr" or click on "Other" in the list on the left to find all the testing related gadgets. Once added to a dashboard, they can be customized to provide detailed test metrics for any project and version.



Not all gadgets have loaded
Were still trying to fetch all of the available gadgets.

Test Distribution
By Zephyr • Local
Test distribution for a given project and version

Test Execution
By Zephyr • Local
Test execution for a given project and version

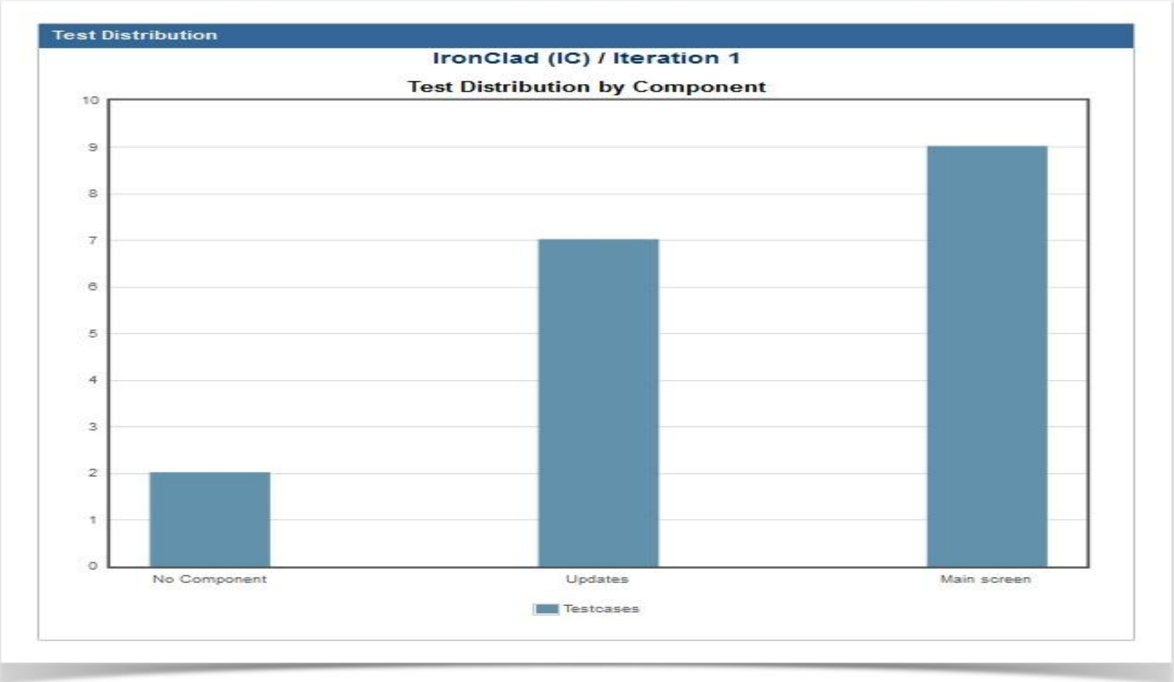
Test Execution Burndown
By Zephyr • Local
Displays rate of test execution, # of unexecuted tests and expected completion date

Test Execution Details
By Zephyr • Local
List of test executions based on free form ZQL or saved ZQL filters

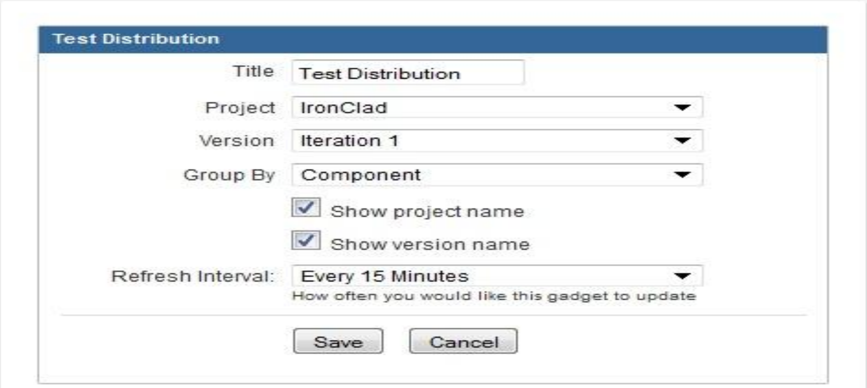
Top Defects Impacting Testing
By Zephyr • Local
Displays a list of defects that impact the most number of Zephyr tests

Test Distribution

The Test Distribution gadget displays totals of tests that have been created for a particular Project-Version combination. The Version here is the "Fixed Version".



The gadget can be customized by selecting the "Edit" option in the top right corner of the gadget. The Title, Project and Version can be changed. Test Distribution data can be grouped either by Component or User. The Refresh Interval can also be modified.


Test Distribution

Title: Test Distribution

Project: IronClad

Version: Iteration 1

Group By: Component

☑ Show project name

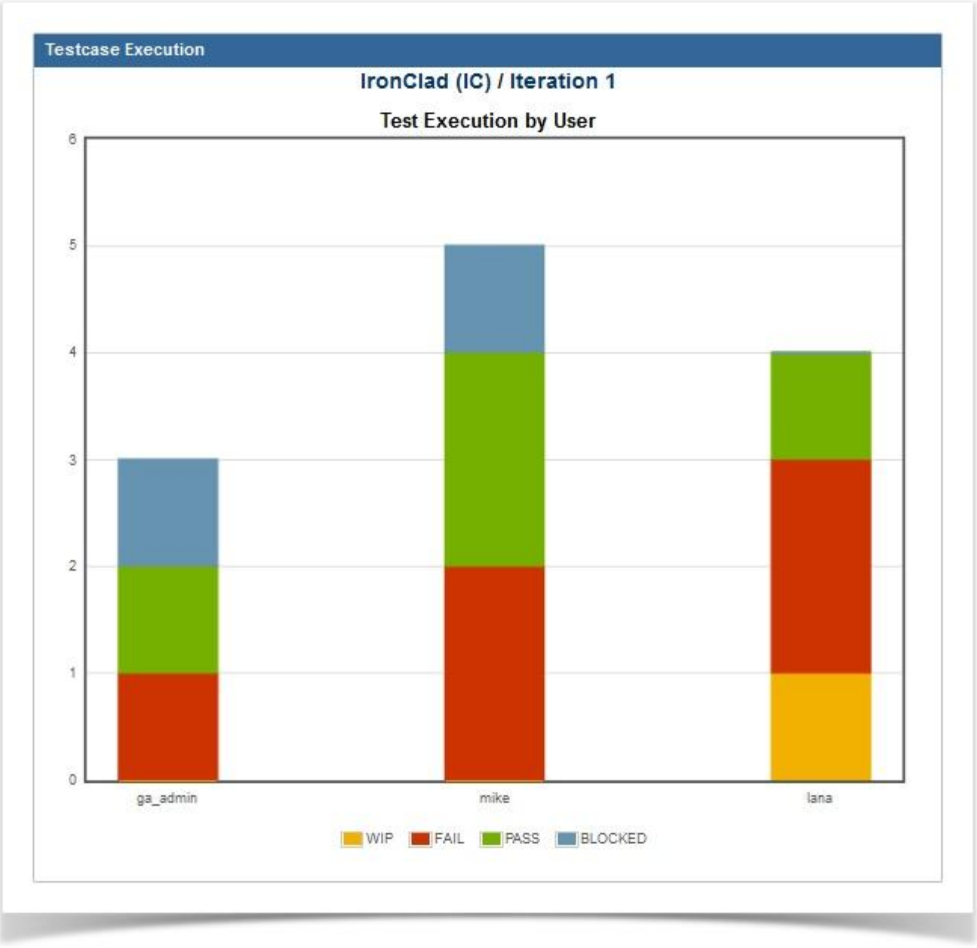
☑ Show version name

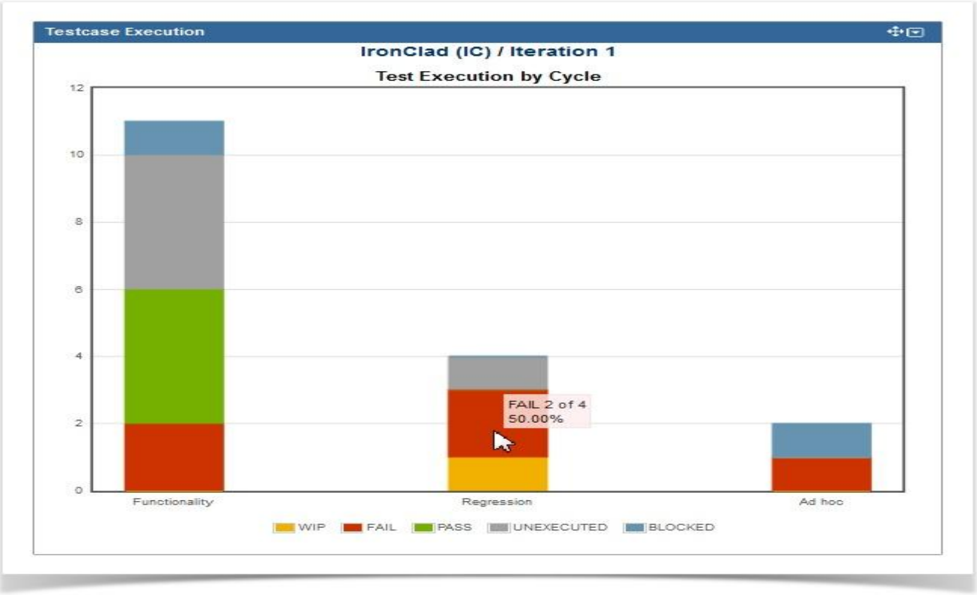
Refresh Interval: Every 15 Minutes

How often you would like this gadget to update

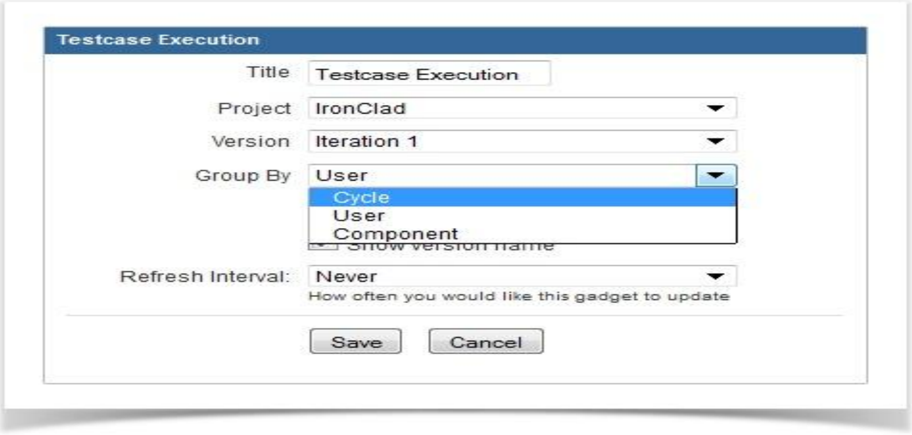
Test Execution

The Test Execution gadget displays totals of tests that have been executed for a particular Project-Version combination. The Version here is the "Fixed Version". This can be customized to show the data grouped either by Test Cycles, Components or Users.


Testcase Execution
IronClad (IC) / Iteration 1
Test Execution by User
6
5
4
3
2
1
0
ga_admin
mike
lana
WIP
FAIL
PASS
BLOCKED


Testcase Execution
IronClad (IC) / Iteration 1
Test Execution by Cycle
12
10
8
6
4
2
0
FAIL 2 of 4
50.00%
Functionality
Regression
Ad hoc
WIP
FAIL
PASS
UNEXECUTED
BLOCKED

The gadget can be customized by selecting the "Edit" option in the top right corner of the gadget. The Title, Project and Version can be changed. Test Execution data can be grouped by Test Cycle, Component or User. The Refresh Interval can also be modified.



Testcase Execution

Title: Testcase Execution

Project: IronClad

Version: Iteration 1

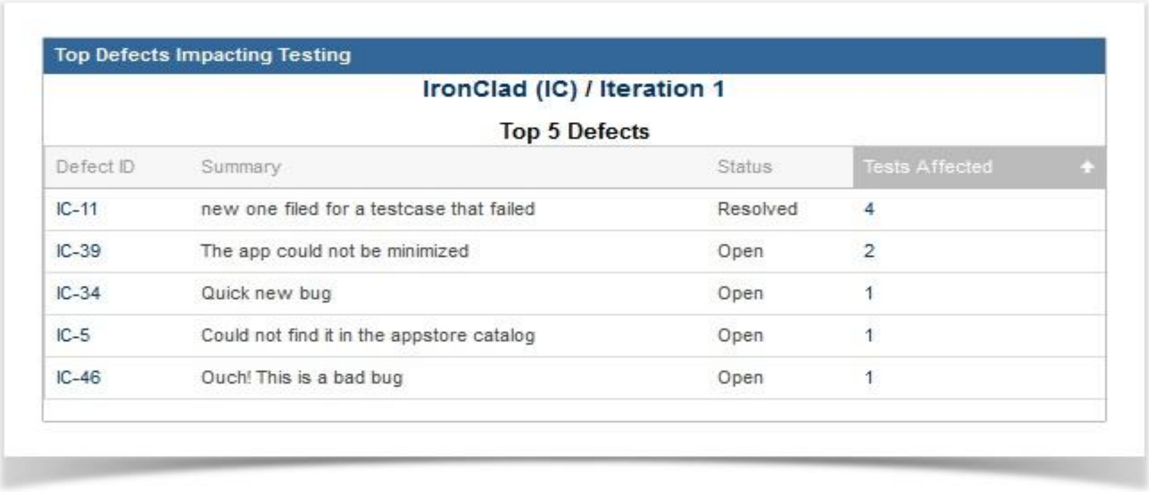
Group By: User

Refresh Interval: Never

How often you would like this gadget to update

Top Defects Impacting Testing

This is a very useful metric that provides critical and impactful information about which defects are currently holding up the maximum number of tests from passing. This gives project teams a clear indication of which issues they should be focusing their energies on in order to move testing forward. This can also provide a view into how resolving defects have impacted testing.



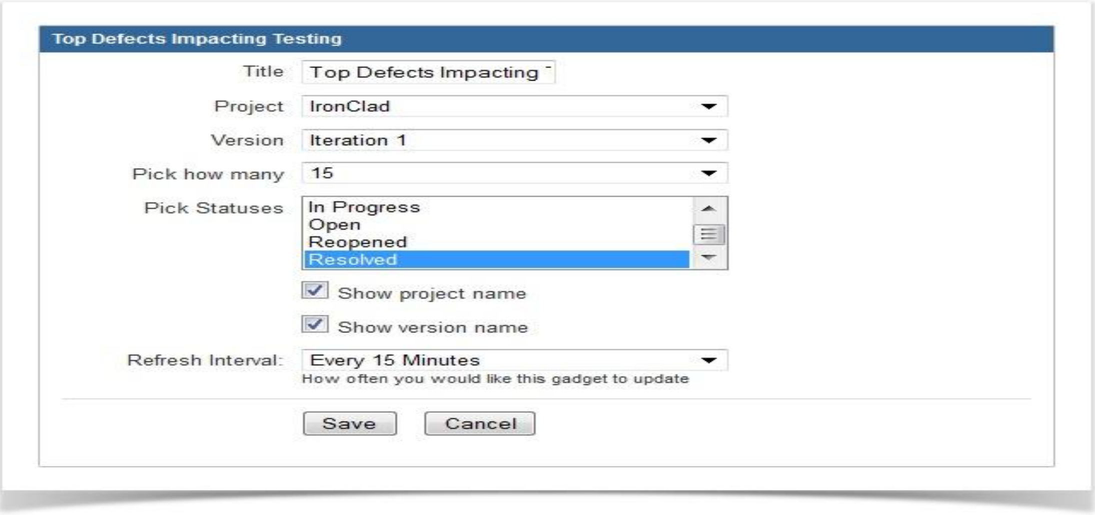
Top Defects Impacting Testing

IronClad (IC) / Iteration 1

Top 5 Defects

Defect ID	Summary	Status	Tests Affected
IC-11	new one filed for a testcase that failed	Resolved	4
IC-39	The app could not be minimized	Open	2
IC-34	Quick new bug	Open	1
IC-5	Could not find it in the appstore catalog	Open	1
IC-46	Ouch! This is a bad bug	Open	1

This gadget can be customized to show 5/10/15 top defects for any project-version combination as well as picking one or more statuses of the defect.

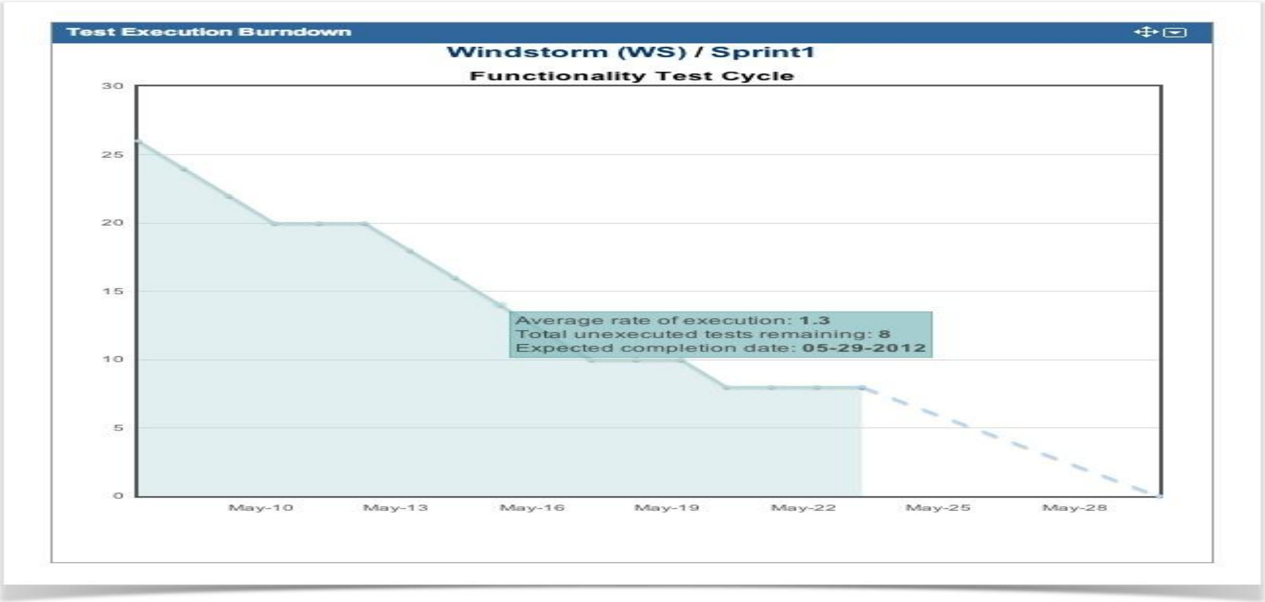


Top Defects Impacting Testing

Title	Top Defects Impacting `
Project	IronClad
Version	Iteration 1
Pick how many	15
Pick Statuses	In Progress Open Reopened Resolved
	☑ Show project name
	☑ Show version name
Refresh Interval:	Every 15 Minutes How often you would like this gadget to update

Test Execution Burndown

This metric tracks the progress of a test cycle, providing valuable information regarding how many tests are still unexecuted, what the current rate of execution is (= number of tests executed / number of days since the test cycle commenced) and extrapolating to when the projected date of completion would be based on the current rate.



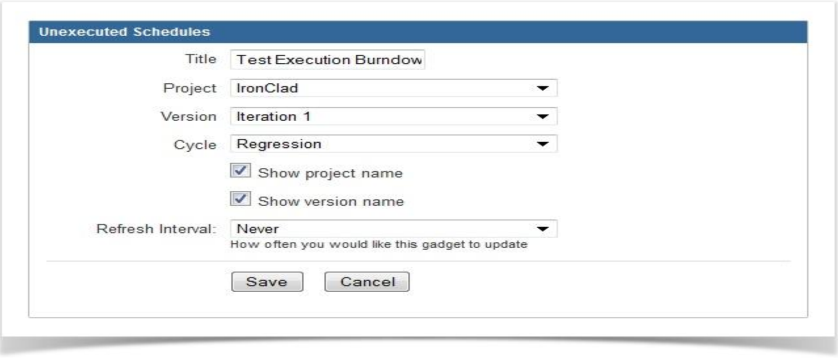
Test Execution Burndown

Windstorm (WS) / Sprint1

Functionality Test Cycle

Average rate of execution: 1.3
Total unexecuted tests remaining: 8
Expected completion date: 05-29-2012

The metric can be customized via the “Edit” button.

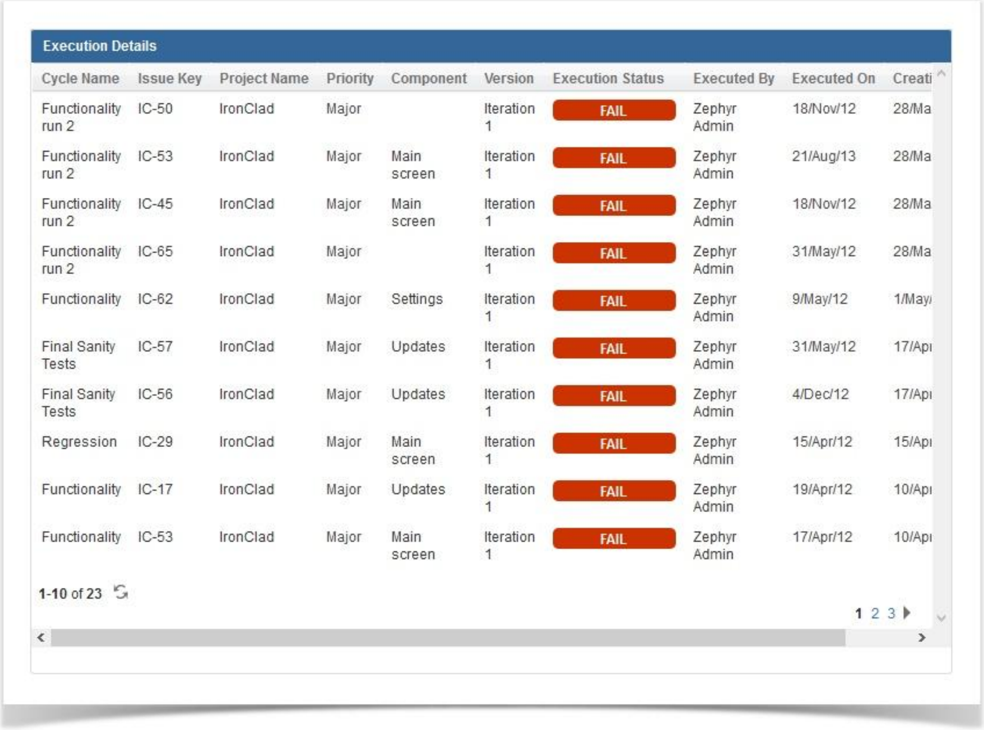


Unexecuted Schedules

Title	Test Execution Burndow
Project	IronClad
Version	Iteration 1
Cycle	Regression
	☑ Show project name
	☑ Show version name
Refresh Interval:	Never How often you would like this gadget to update

Test Execution Details

This gadget fetches data based on a saved ZQL search (i.e. a test execution filter) or a free-form ZQL query.

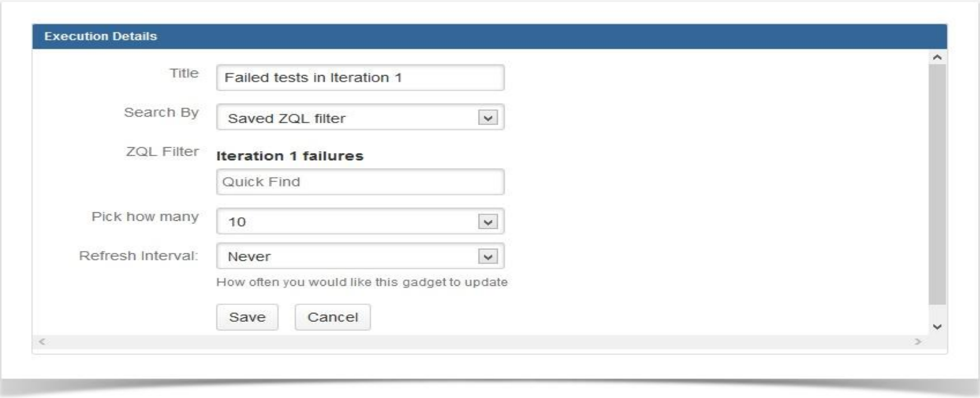


Execution Details

Cycle Name	Issue Key	Project Name	Priority	Component	Version	Execution Status	Executed By	Executed On	Creati
Functionality run 2	IC-50	IronClad	Major		Iteration 1	FAIL	Zephyr Admin	18/Nov/12	28/Ma
Functionality run 2	IC-53	IronClad	Major	Main screen	Iteration 1	FAIL	Zephyr Admin	21/Aug/13	28/Ma
Functionality run 2	IC-45	IronClad	Major	Main screen	Iteration 1	FAIL	Zephyr Admin	18/Nov/12	28/Ma
Functionality run 2	IC-65	IronClad	Major		Iteration 1	FAIL	Zephyr Admin	31/May/12	28/Ma
Functionality	IC-62	IronClad	Major	Settings	Iteration 1	FAIL	Zephyr Admin	9/May/12	1/May/
Final Sanity Tests	IC-57	IronClad	Major	Updates	Iteration 1	FAIL	Zephyr Admin	31/May/12	17/Apı
Final Sanity Tests	IC-56	IronClad	Major	Updates	Iteration 1	FAIL	Zephyr Admin	4/Dec/12	17/Apı
Regression	IC-29	IronClad	Major	Main screen	Iteration 1	FAIL	Zephyr Admin	15/Apr/12	15/Apı
Functionality	IC-17	IronClad	Major	Updates	Iteration 1	FAIL	Zephyr Admin	19/Apr/12	10/Apı
Functionality	IC-53	IronClad	Major	Main screen	Iteration 1	FAIL	Zephyr Admin	17/Apr/12	10/Apı

1-10 of 23

The metrics can be customized via the "Edit" button where you can search for and select a saved filter or type in the ZQL query right there.



Execution Details

Title: Failed tests in Iteration 1

Search By: Saved ZQL filter

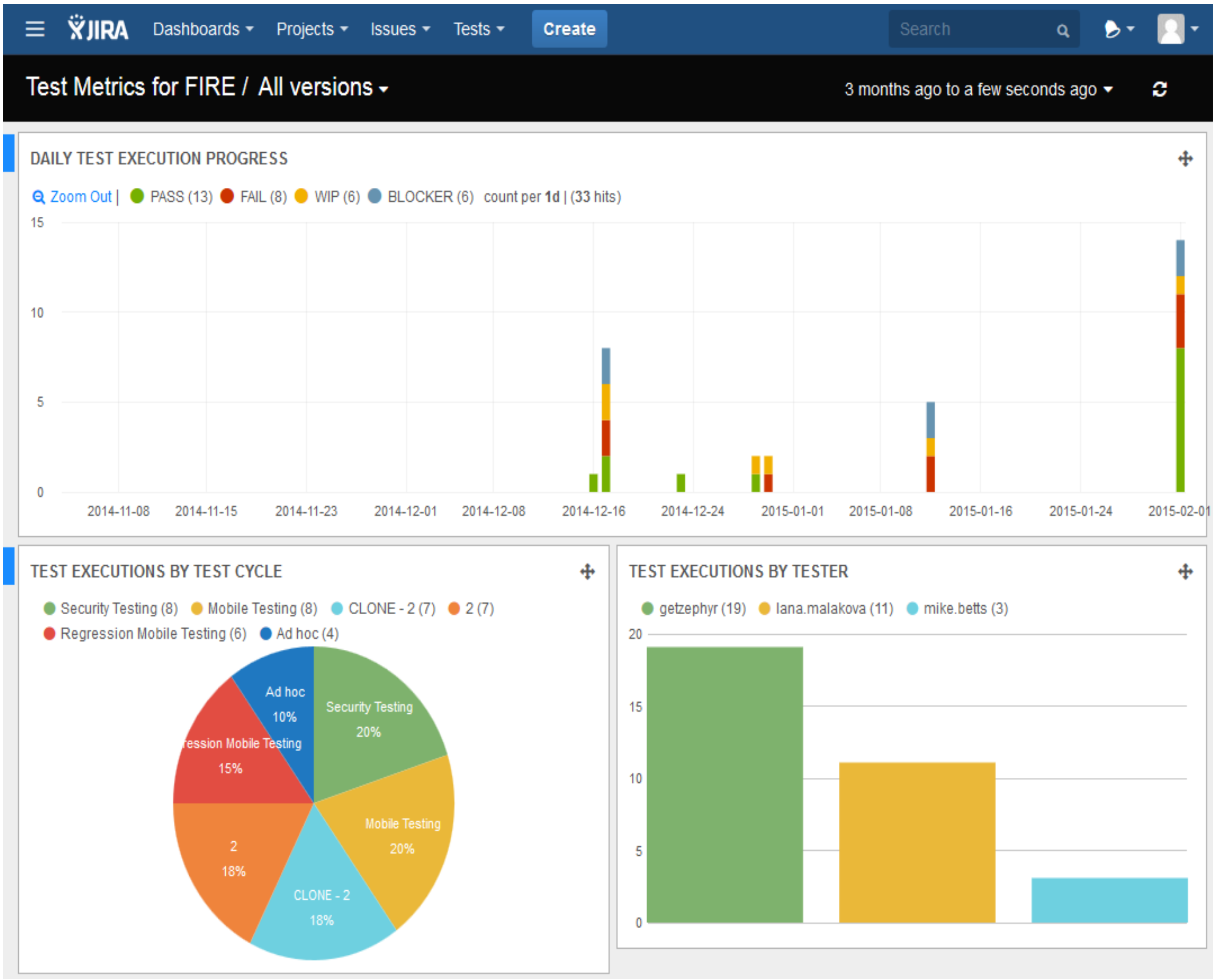
ZQL Filter: **Iteration 1 failures**

Quick Find

Pick how many: 10

Refresh Interval: Never

How often you would like this gadget to update



JIRA
Dashboards
Projects
Issues
Tests
Create
Search
Test Metrics for FIRE / All versions
3 months ago to a few seconds ago
DAILY TEST EXECUTION PROGRESS
Zoom Out | ● PASS (13) ● FAIL (8) ● WIP (6) ● BLOCKER (6) count per 1d | (33 hits)
15
10
5
0
2014-11-08
2014-11-15
2014-11-23
2014-12-01
2014-12-08
2014-12-16
2014-12-24
2015-01-01
2015-01-08
2015-01-16
2015-01-24
2015-02-01
TEST EXECUTIONS BY TEST CYCLE
● Security Testing (8) ● Mobile Testing (8) ● CLONE - 2 (7) ● 2 (7)
● Regression Mobile Testing (6) ● Ad hoc (4)
Ad hoc 10%
Security Testing 20%
ession Mobile Testing 15%
Mobile Testing 20%
2 18%
CLONE - 2 18%
TEST EXECUTIONS BY TESTER
● getzephyr (19) ● lana.malakova (11) ● mike.betts (3)
20
15
10
5
0

Multiple test metrics are automatically tracked for every project:

- Daily Test Execution Progress, by status

- Test Executions by Test Cycle
- Test Executions by Tester
- Number of executions per day
- And a List of executions with their details

Options

The data displayed in the dashboard can be across an entire project (i.e. including all Versions) or can be specific to a particular Version.

You can also change the data range for which the data is displayed. In the above example, the range is set to "3 months to a few seconds ago". Click on that to pick another option.

A tiny refresh icon allows the data to be refreshed on demand. Auto-refresh options also exist. Pick one of the various available refresh rates by clicking on the range and selecting the "Auto-Refresh" option.

Each panel of this dashboard can be moved around to a different location, during a single session.

Changes made to this dashboard are only remembered for the current session. All defaults are restored when the metrics page is refreshed.

Customizable Test Metrics (Gadgets) and Dashboards

5 different types of metrics can be customized and added to standard JIRA Dashboards. Click here to learn how.

Traceability

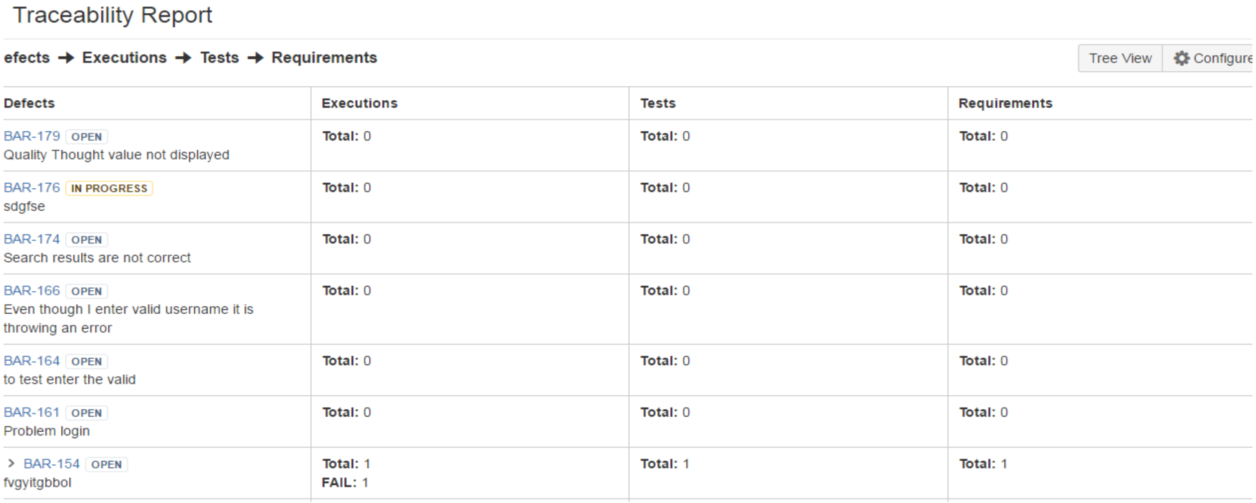
Introduction

The ability to trace linkages from requirements all the way to defects or vice versa is particularly useful in a software release cycle. In fact, it can have special meaning at various phases in that release cycle. For example, starting with requirements, knowing how many of them have tests written for them is useful in the early stages to ensure appropriate test coverage. Once the software has been built, keeping track of which test executions have passed for a particular requirement allows the team to make a quality statement about these requirements. And then keeping track of how many open defects exist for requirements help make a Go/No-Go decision regarding the readiness of the software to be shipped.

Traceability reports can also be tremendously useful in producing end-of-release audit reports for compliance and regulatory reasons. They can also be used as customer delivery reports highlighting how every one of their requirements has been met, tested and is defect-free.

Another very important reason to run Defect to Requirement traceability report is to get a better sense of how many defects are holding up requirements and more importantly, which defect(s) is impacting the most number of requirements – thereby allowing for better bug-fixing prioritization.

Access

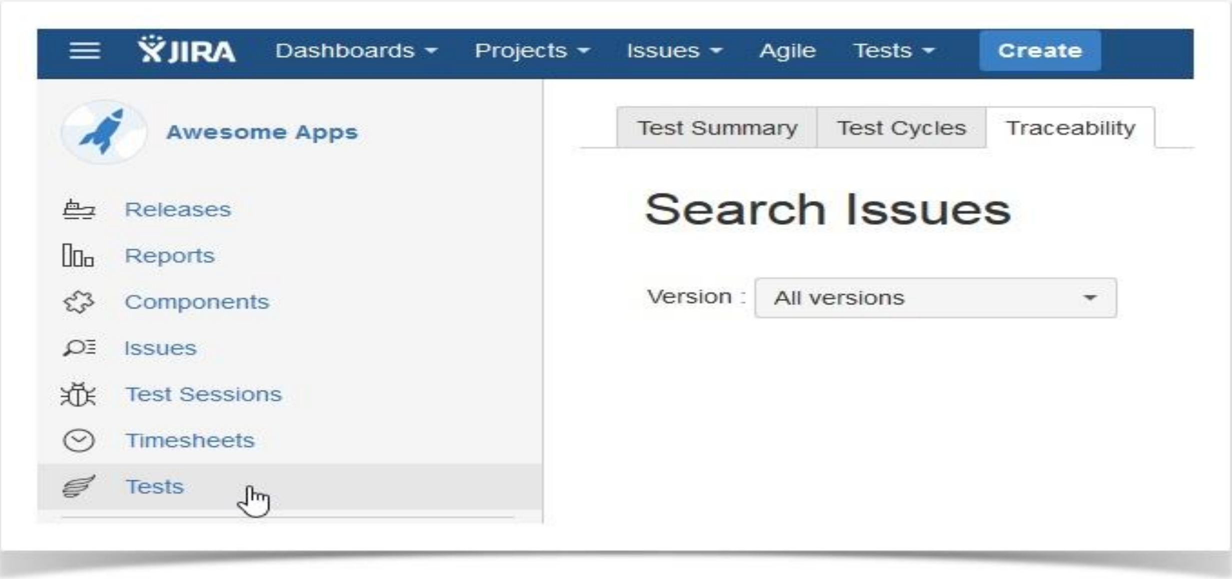


Traceability Report

efects → Executions → Tests → Requirements

Defects	Executions	Tests	Requirements
BAR-179 OPEN Quality Thought value not displayed	Total: 0	Total: 0	Total: 0
BAR-176 IN PROGRESS sdgfse	Total: 0	Total: 0	Total: 0
BAR-174 OPEN Search results are not correct	Total: 0	Total: 0	Total: 0
BAR-166 OPEN Even though I enter valid username it is throwing an error	Total: 0	Total: 0	Total: 0
BAR-164 OPEN to test enter the valid	Total: 0	Total: 0	Total: 0
BAR-161 OPEN Problem login	Total: 0	Total: 0	Total: 0
› BAR-154 OPEN fvgyitgbbol	Total: 1 FAIL: 1	Total: 1	Total: 1

Traceability in Zephyr for JIRA Cloud can be accessed from the top menu or from the "Tests" menu in the Project sidebar



Search Issues

Version : All versions

IMPORTANT

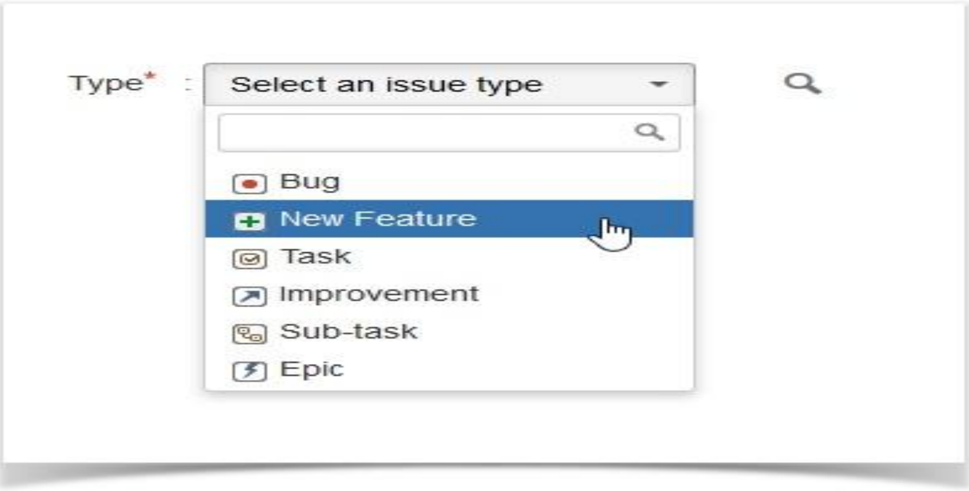
: In order to get useful traceability reports, ensure the following during test creation and execution:

- ✓ Your tests are linked to requirements
- ✓ Defects that are filed are linked to the test executions in which they were found

Types of Traceability Reports

Requirements to Defects

Select a Version and a particular Issue Type that can be the starting point of your traceability report. That is usually a “New Feature” issue-type or a “Story” or an “Epic”. The resulting list of issues can then be narrowed down to the ones you want a report on. Selecting the “Requirement to Defect” option and clicking on “Generate Traceability Report” will give you a full traceability report from Requirements --> Tests --> Test Executions --> Defects.

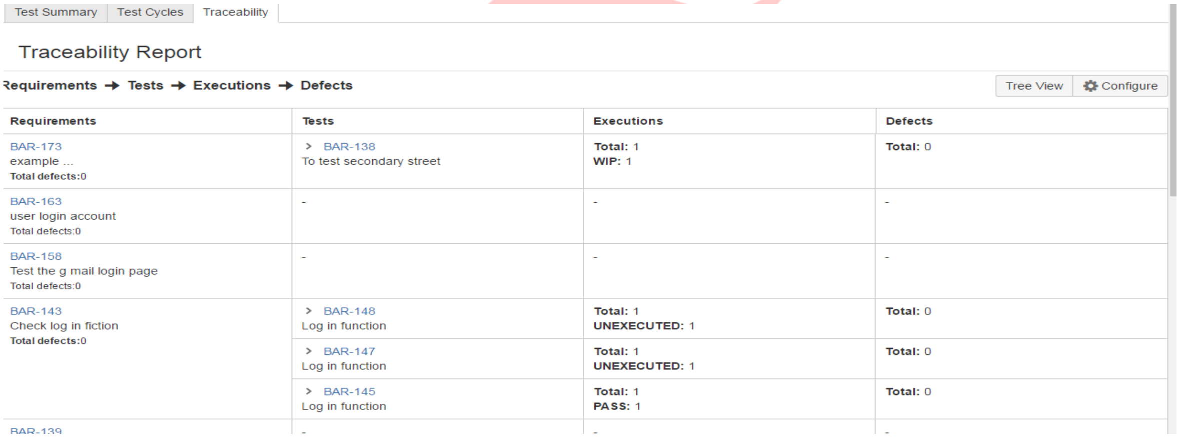


This can then be exported as an HTML file or an Excel file for further manipulation.

Useful for:

1. Keeping track of progress at various stages of the software release cycle
2. Audit and Compliance reports
3. Customer Delivery Reports

Here's an example of such a report:



Traceability Report

Requirements → Tests → Executions → Defects

Requirements	Tests	Executions	Defects
BAR-173 example ... **Total defects:**0	> BAR-138 To test secondary street	**Total:** 1 **WIP:** 1	**Total:** 0
BAR-163 user login account Total defects:0	-	-	-
BAR-158 Test the g mail login page Total defects:0	-	-	-
BAR-143 Check log in fiction **Total defects:**0	> BAR-148 Log in function	**Total:** 1 **UNEXECUTED:** 1	**Total:** 0
	> BAR-147 Log in function	**Total:** 1 **UNEXECUTED:** 1	**Total:** 0
	> BAR-145 Log in function	**Total:** 1 **PASS:** 1	**Total:** 0
BAR-139	-	-	-

Defects to Requirements

Select a Version and a particular Issue Type that can be the starting point of your traceability report. That is usually a "Bug". The resulting list of issues can then be narrowed down to the ones you want a report on. Selecting the "Defect to Requirement" option and clicking on

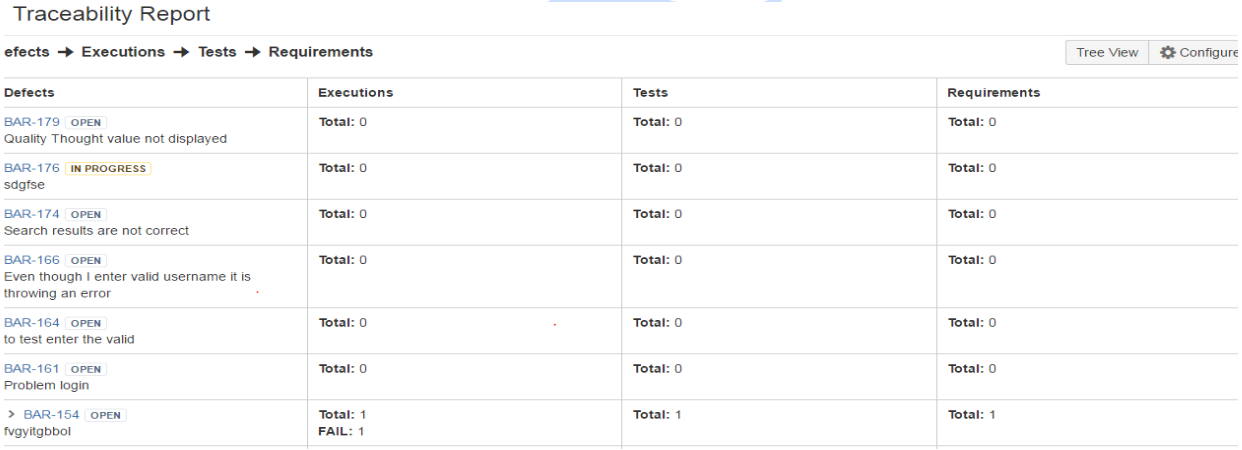
“Generate Traceability Report” will you give you a full traceability report from Defects --> Test Executions --> Tests --> Requirements.

This can then be exported as an HTML file or an Excel file for further manipulation.

Useful for:

1. Keeping track of how many open defects are impacting requirements
2. Prioritizing defect fixing based on which defect(s) is impacting the most number of requirements
3. Creating defect-regression test cycles for individual requirements based on number and status of defects

Here's an example of such a report:



Traceability Report

efects → Executions → Tests → Requirements

Defects	Executions	Tests	Requirements
BAR-179 OPEN Quality Thought value not displayed	**Total:** 0	**Total:** 0	**Total:** 0
BAR-176 IN PROGRESS sdgfse	**Total:** 0	**Total:** 0	**Total:** 0
BAR-174 OPEN Search results are not correct	**Total:** 0	**Total:** 0	**Total:** 0
BAR-166 OPEN Even though I enter valid username it is throwing an error	**Total:** 0	**Total:** 0	**Total:** 0
BAR-164 OPEN to test enter the valid	**Total:** 0	**Total:** 0	**Total:** 0
BAR-161 OPEN Problem login	**Total:** 0	**Total:** 0	**Total:** 0
> BAR-154 OPEN fvgyitgbbol	**Total:** 1 **FAIL:** 1	**Total:** 1	**Total:** 1

Activity 1:

How to create test case in JIRA Tool?

Activity 2:

How to link requirement to test case in JIRA Tool?

Activity 3:

How to search Test Cases in JIRA Tool?

Activity 4:

How to clone Test Cases in JIRA Tool?

Activity 5:

How to do bulk updates to Test cases in JIRA Tool?

Activity 6:

How to do Adhoc execution in JIRA Tool?

Activity 7:

How to do structured execution in JIRA Tool?

Activity 8:

How to create Test Metrics?

Activity 9:

How to create Traceability Report?

Activity 10:

How to create defect in JIRA Tool?

SESSION - 05

ADVANCED TESTING:

1. Entry and Exit criteria
2. Defect Reports
3. Testing Metrics
4. Best Practices in Testing
5. Achievements and Additional contribution to project
6. Risks ,Issues, challenges

7. Test Reports
8. Testing Estimations
9. Risk Based Testing(RBT)
10. Test Plan and Test strategy
11. Auditing process in Testing

Interview Questions:

1. Do you have any knowledge on Entry criteria and Exit criteria?
2. Do you have any knowledge on Testing Metrics?
3. Do you have any knowledge on defect reports?
4. Do you have any knowledge on below reports

    Weekly status report
    Release status report
    Sign off report
    Release notes
    Burn up chart
    Burn down chart
    Sprint Retrospective report
    Test Summary Report
    MOM

5. Do you have any knowledge on Testing Estimations?
6. Do you have any knowledge on Risk management? Explain risk you have identified in current project?
7. Explain few issues and challenges from your project?
8. Explain how many test cases are enough for a requirement?
9. Explain about below deliverables?

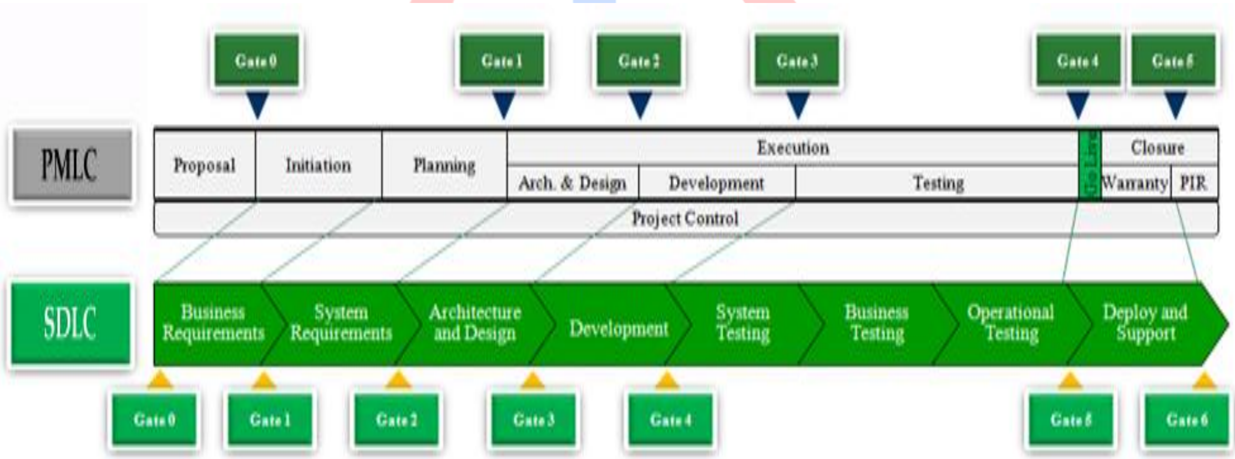
    Test Strategy
    Test Plan

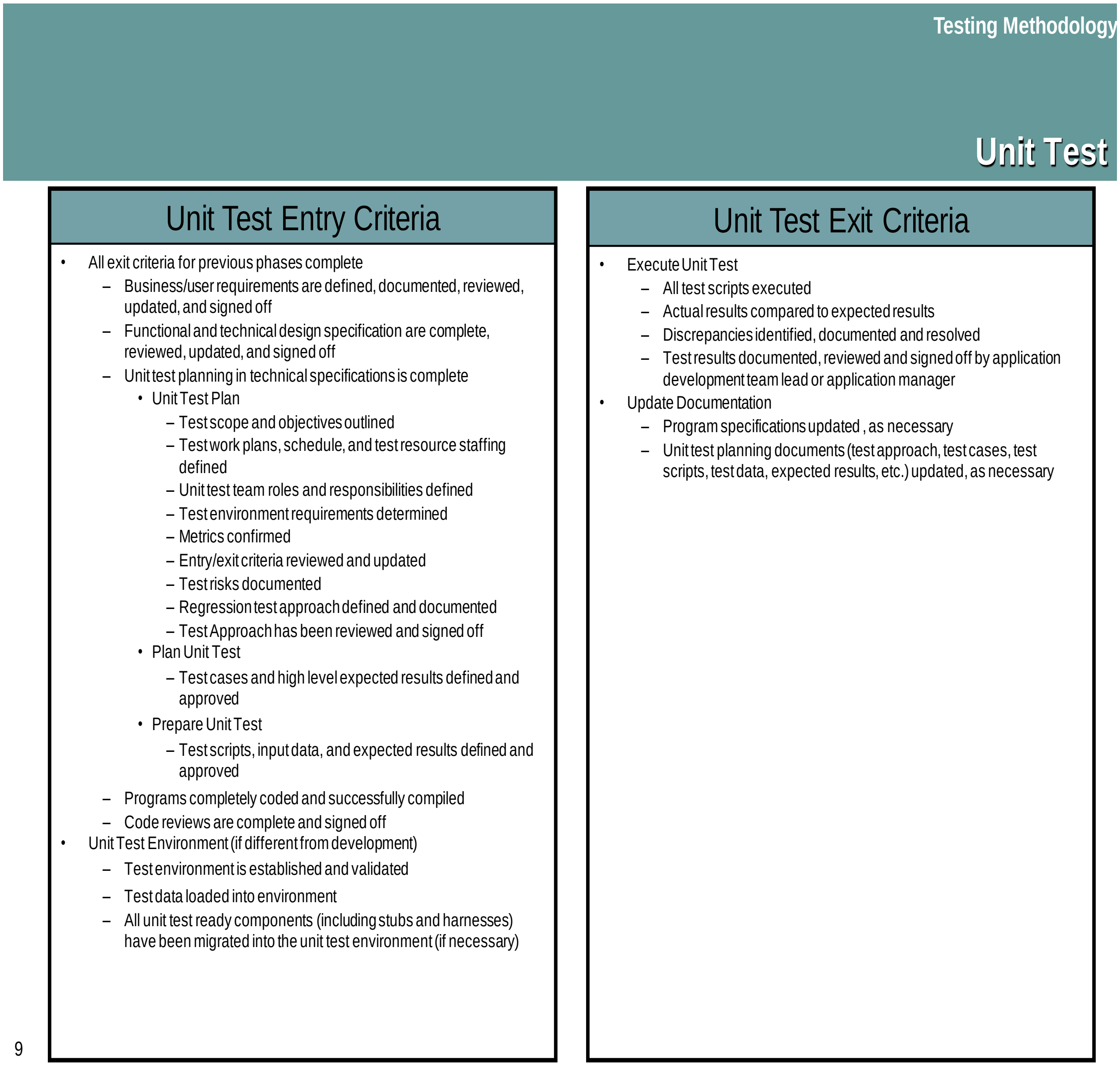
10. Explain your additional contribution to the project in addition to general activities like Test case design, review and execution?

Topic: 1

Entry criteria and Exit criteria

- ✓ Test Gating helps to confirm that the product (i.e. IT business applications) is of sufficient quality and that the necessary conditions are in place to successfully move to the next execution step in a project.
- ✓ Entry and exit criteria are used to "gate" the process.
- ✓ A "gate" is the point at which a decision is made to allow the project to enter and exit an execution step. The recommended entry criteria for an test phase on a project should be fulfilled before test execution can commence, and the recommended exit criteria should be achieved before leaving the test phase. Ultimately, the decision as to whether a project moves forward beyond a gate is an informed decision that is based on risk, with the stakeholders being in agreement that the risks are manageable, and also weigh issues such as schedule delays and cost. The two types of gates in effect are 'Hard' and 'Soft' gates.
- ✓ Hard Gate: Requires a gate meeting to review the gate assessment, provide clarification, and reach a gating decision.
- ✓ Soft Gate: Does not require a gate meeting, as sharing the results of the gate assessment and reaching a gating decision can be done entirely through email.





Testing Methodology

Unit Test

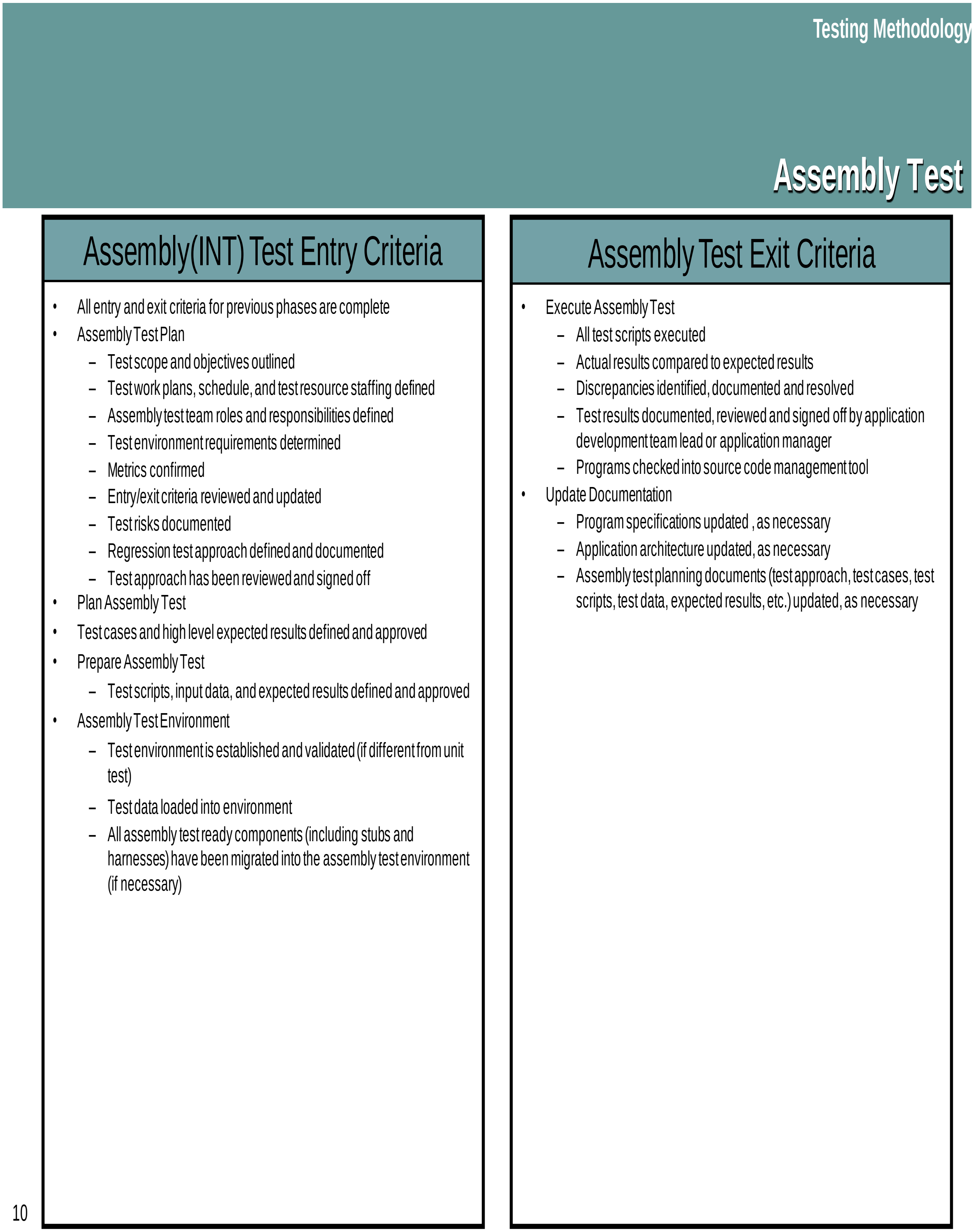
Unit Test Entry Criteria

- All exit criteria for previous phases complete
  - Business/user requirements are defined, documented, reviewed, updated, and signed off
  - Functional and technical design specification are complete, reviewed, updated, and signed off
  - Unit test planning in technical specifications is complete
    - Unit Test Plan
      - Test scope and objectives outlined
      - Test work plans, schedule, and test resource staffing defined
      - Unit test team roles and responsibilities defined
      - Test environment requirements determined
      - Metrics confirmed
      - Entry/exit criteria reviewed and updated
      - Test risks documented
      - Regression test approach defined and documented
      - Test Approach has been reviewed and signed off
    - Plan Unit Test
      - Test cases and high level expected results defined and approved
    - Prepare Unit Test
      - Test scripts, input data, and expected results defined and approved
  - Programs completely coded and successfully compiled
  - Code reviews are complete and signed off
- Unit Test Environment (if different from development)
  - Test environment is established and validated
  - Test data loaded into environment
  - All unit test ready components (including stubs and harnesses) have been migrated into the unit test environment (if necessary)

Unit Test Exit Criteria

- Execute Unit Test
  - All test scripts executed
  - Actual results compared to expected results
  - Discrepancies identified, documented and resolved
  - Test results documented, reviewed and signed off by application development team lead or application manager
- Update Documentation
  - Program specifications updated , as necessary
  - Unit test planning documents (test approach, test cases, test scripts, test data, expected results, etc.) updated, as necessary

9



Testing Methodology

Assembly Test

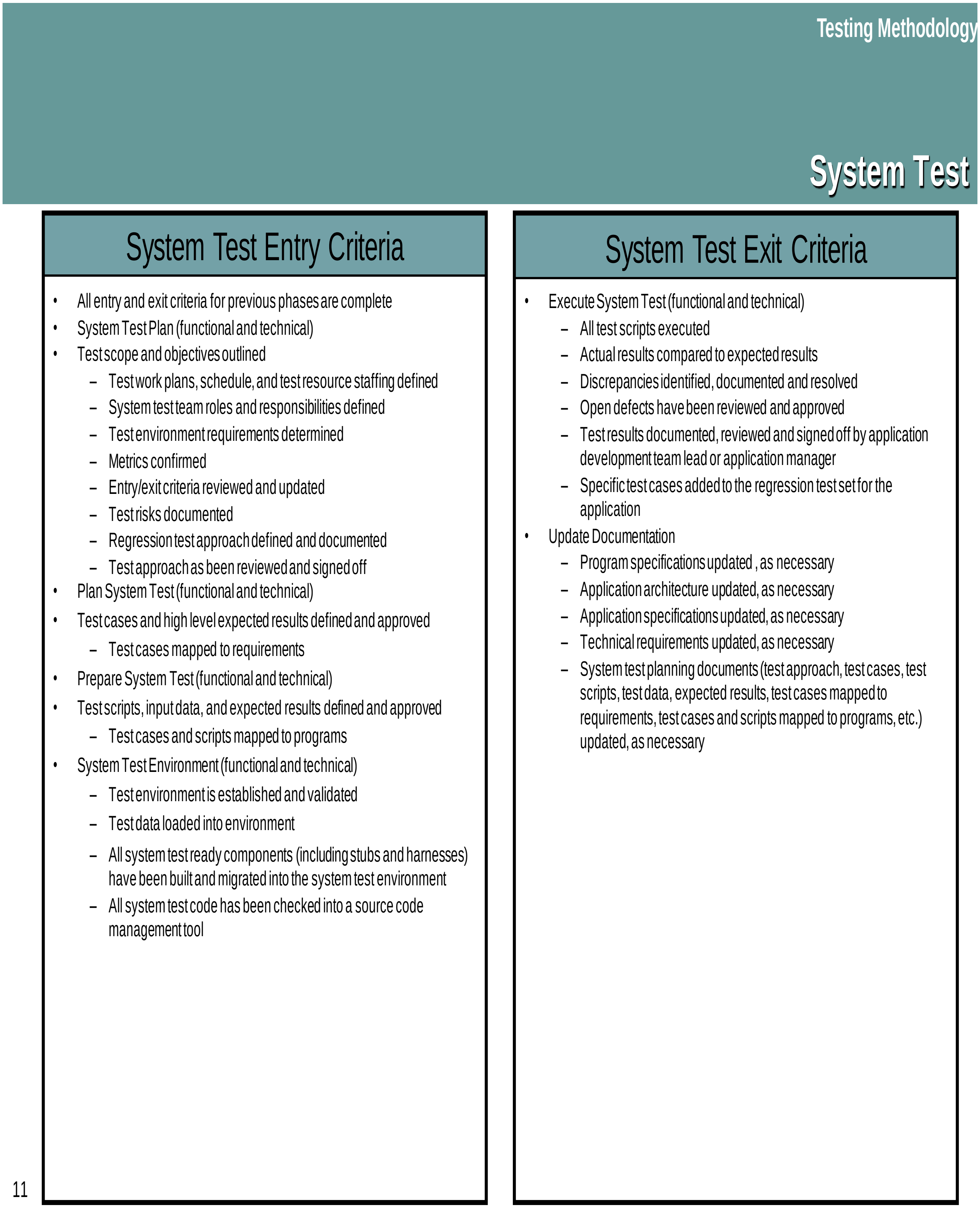
Assembly(INT) Test Entry Criteria

- All entry and exit criteria for previous phases are complete
- Assembly Test Plan
  - Test scope and objectives outlined
  - Test work plans, schedule, and test resource staffing defined
  - Assembly test team roles and responsibilities defined
  - Test environment requirements determined
  - Metrics confirmed
  - Entry/exit criteria reviewed and updated
  - Test risks documented
  - Regression test approach defined and documented
  - Test approach has been reviewed and signed off
- Plan Assembly Test
- Test cases and high level expected results defined and approved
- Prepare Assembly Test
  - Test scripts, input data, and expected results defined and approved
- Assembly Test Environment
  - Test environment is established and validated (if different from unit test)
  - Test data loaded into environment
  - All assembly test ready components (including stubs and harnesses) have been migrated into the assembly test environment (if necessary)

Assembly Test Exit Criteria

- Execute Assembly Test
  - All test scripts executed
  - Actual results compared to expected results
  - Discrepancies identified, documented and resolved
  - Test results documented, reviewed and signed off by application development team lead or application manager
  - Programs checked into source code management tool
- Update Documentation
  - Program specifications updated , as necessary
  - Application architecture updated, as necessary
  - Assembly test planning documents (test approach, test cases, test scripts, test data, expected results, etc.) updated, as necessary

10



Testing Methodology

System Test

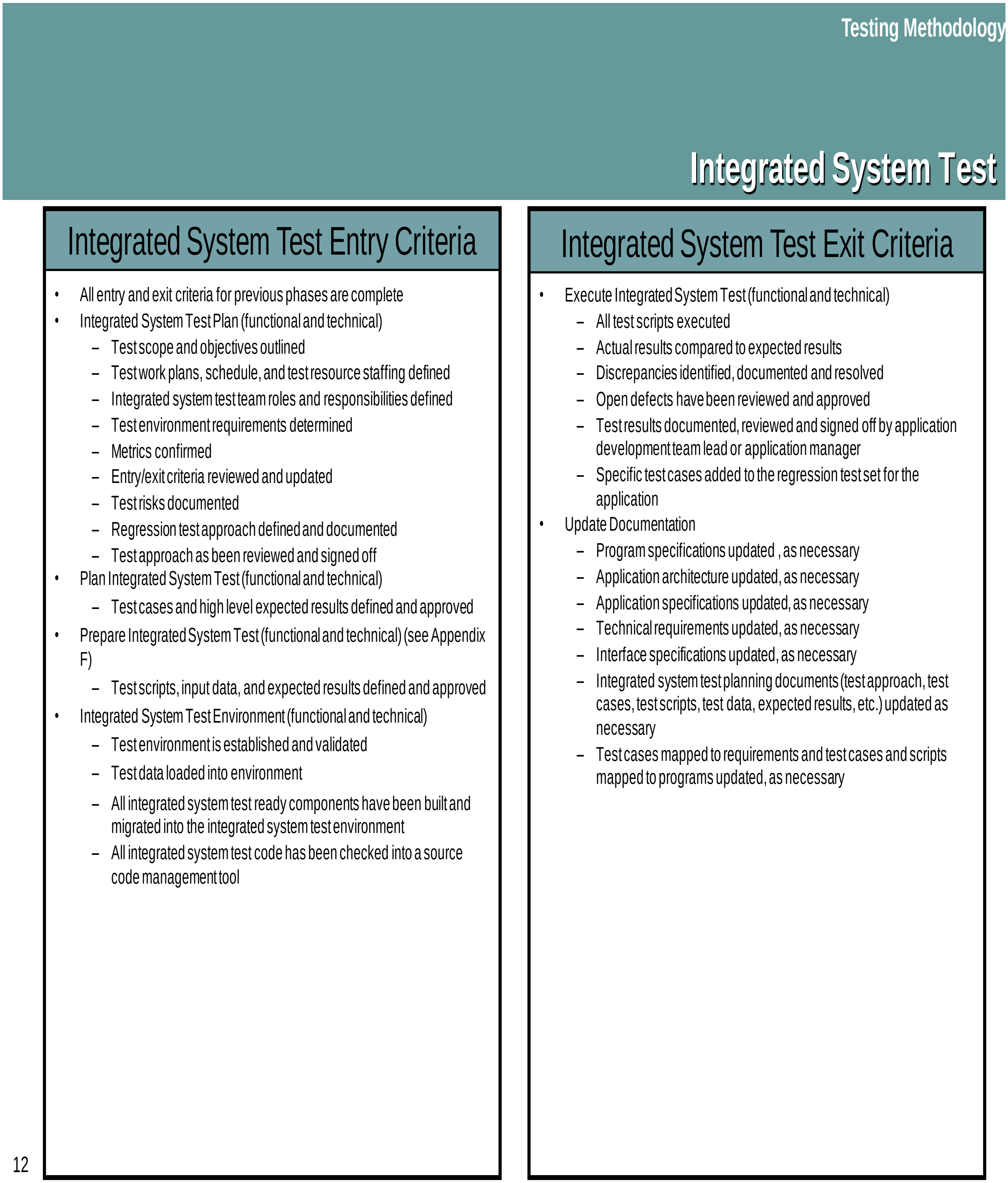
System Test Entry Criteria

- All entry and exit criteria for previous phases are complete
- System Test Plan (functional and technical)
- Test scope and objectives outlined
  - Test work plans, schedule, and test resource staffing defined
  - System test team roles and responsibilities defined
  - Test environment requirements determined
  - Metrics confirmed
  - Entry/exit criteria reviewed and updated
  - Test risks documented
  - Regression test approach defined and documented
  - Test approach as been reviewed and signed off
- Plan System Test (functional and technical)
- Test cases and high level expected results defined and approved
  - Test cases mapped to requirements
- Prepare System Test (functional and technical)
- Test scripts, input data, and expected results defined and approved
  - Test cases and scripts mapped to programs
- System Test Environment (functional and technical)
  - Test environment is established and validated
  - Test data loaded into environment
  - All system test ready components (including stubs and harnesses) have been built and migrated into the system test environment
  - All system test code has been checked into a source code management tool

System Test Exit Criteria

- Execute System Test (functional and technical)
  - All test scripts executed
  - Actual results compared to expected results
  - Discrepancies identified, documented and resolved
  - Open defects have been reviewed and approved
  - Test results documented, reviewed and signed off by application development team lead or application manager
  - Specific test cases added to the regression test set for the application
- Update Documentation
  - Program specifications updated , as necessary
  - Application architecture updated, as necessary
  - Application specifications updated, as necessary
  - Technical requirements updated, as necessary
  - System test planning documents (test approach, test cases, test scripts, test data, expected results, test cases mapped to requirements, test cases and scripts mapped to programs, etc.) updated, as necessary

11



Testing Methodology

Integrated System Test

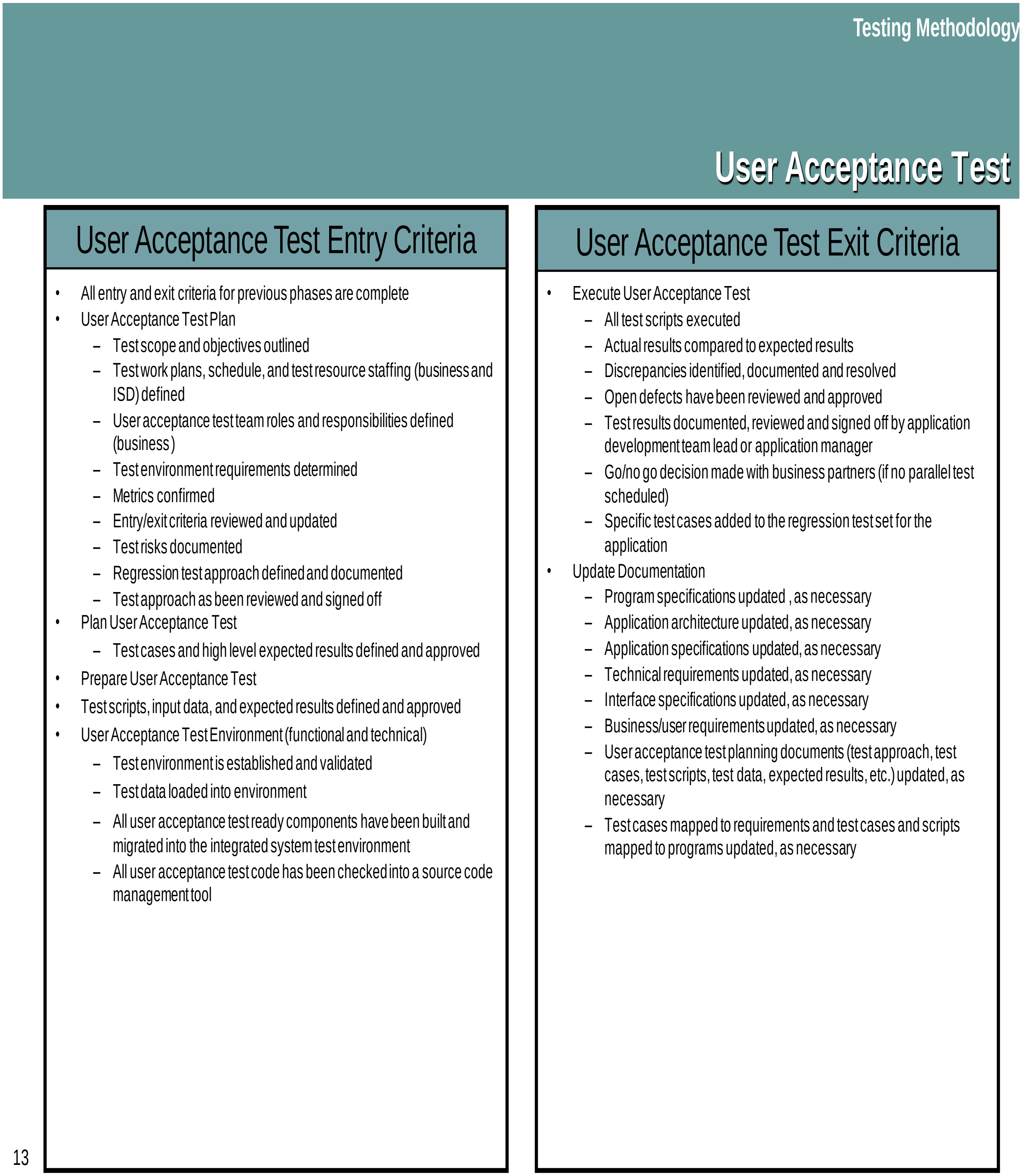
Integrated System Test Entry Criteria

- All entry and exit criteria for previous phases are complete
- Integrated System Test Plan (functional and technical)
  - Test scope and objectives outlined
  - Test work plans, schedule, and test resource staffing defined
  - Integrated system test team roles and responsibilities defined
  - Test environment requirements determined
  - Metrics confirmed
  - Entry/exit criteria reviewed and updated
  - Test risks documented
  - Regression test approach defined and documented
  - Test approach as been reviewed and signed off
- Plan Integrated System Test (functional and technical)
  - Test cases and high level expected results defined and approved
- Prepare Integrated System Test (functional and technical) (see Appendix F)
  - Test scripts, input data, and expected results defined and approved
- Integrated System Test Environment (functional and technical)
  - Test environment is established and validated
  - Test data loaded into environment
  - All integrated system test ready components have been built and migrated into the integrated system test environment
  - All integrated system test code has been checked into a source code management tool

12

Integrated System Test Exit Criteria

- Execute Integrated System Test (functional and technical)
  - All test scripts executed
  - Actual results compared to expected results
  - Discrepancies identified, documented and resolved
  - Open defects have been reviewed and approved
  - Test results documented, reviewed and signed off by application development team lead or application manager
  - Specific test cases added to the regression test set for the application
- Update Documentation
  - Program specifications updated , as necessary
  - Application architecture updated, as necessary
  - Application specifications updated, as necessary
  - Technical requirements updated, as necessary
  - Interface specifications updated, as necessary
  - Integrated system test planning documents (test approach, test cases, test scripts, test data, expected results, etc.) updated as necessary
  - Test cases mapped to requirements and test cases and scripts mapped to programs updated, as necessary



Testing Methodology

User Acceptance Test

User Acceptance Test Entry Criteria

- All entry and exit criteria for previous phases are complete
- User Acceptance Test Plan
  - Test scope and objectives outlined
  - Test work plans, schedule, and test resource staffing (business and ISD) defined
  - User acceptance test team roles and responsibilities defined (business)
  - Test environment requirements determined
  - Metrics confirmed
  - Entry/exit criteria reviewed and updated
  - Test risks documented
  - Regression test approach defined and documented
  - Test approach as been reviewed and signed off
- Plan User Acceptance Test
  - Test cases and high level expected results defined and approved
- Prepare User Acceptance Test
- Test scripts, input data, and expected results defined and approved
- User Acceptance Test Environment (functional and technical)
  - Test environment is established and validated
  - Test data loaded into environment
  - All user acceptance test ready components have been built and migrated into the integrated system test environment
  - All user acceptance test code has been checked into a source code management tool

User Acceptance Test Exit Criteria

- Execute User Acceptance Test
  - All test scripts executed
  - Actual results compared to expected results
  - Discrepancies identified, documented and resolved
  - Open defects have been reviewed and approved
  - Test results documented, reviewed and signed off by application development team lead or application manager
  - Go/no go decision made with business partners (if no parallel test scheduled)
  - Specific test cases added to the regression test set for the application
- Update Documentation
  - Program specifications updated , as necessary
  - Application architecture updated, as necessary
  - Application specifications updated, as necessary
  - Technical requirements updated, as necessary
  - Interface specifications updated, as necessary
  - Business/user requirements updated, as necessary
  - User acceptance test planning documents (test approach, test cases, test scripts, test data, expected results, etc.) updated, as necessary
  - Test cases mapped to requirements and test cases and scripts mapped to programs updated, as necessary

13

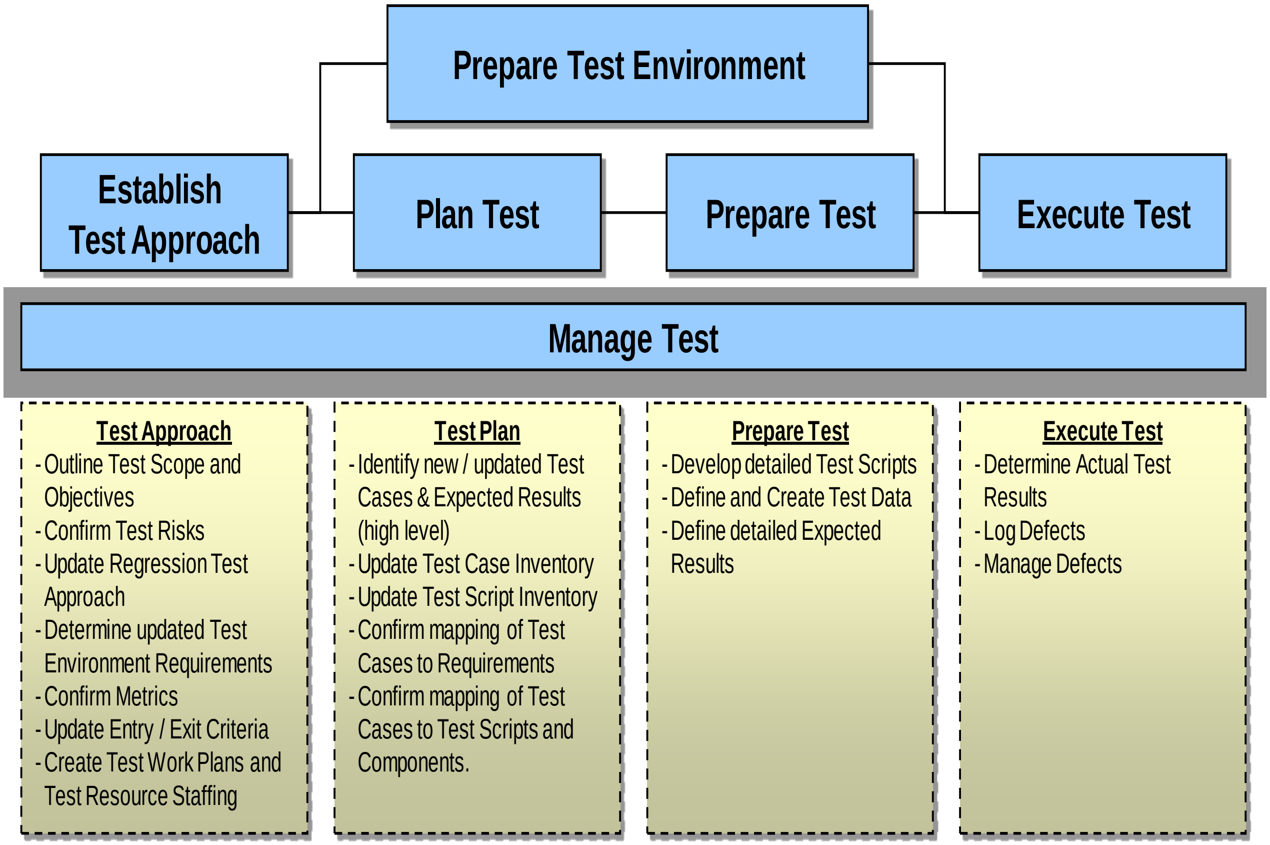
Defect Reporting:

Test Management - Metrics and Reporting

Test Process Flow

Metrics and reporting addresses the Manage Test process in the test process flow. Metrics and reporting can provide objective data to support the management of the testing process.

Test Process Flow



Prepare Test Environment

Establish Test Approach

Plan Test

Prepare Test

Execute Test

Manage Test

Test Approach

- Outline Test Scope and Objectives
- Confirm Test Risks
- Update Regression Test Approach
- Determine updated Test Environment Requirements
- Confirm Metrics
- Update Entry / Exit Criteria
- Create Test Work Plans and Test Resource Staffing

Test Plan

- Identify new / updated Test Cases & Expected Results (high level)
- Update Test Case Inventory
- Update Test Script Inventory
- Confirm mapping of Test Cases to Requirements
- Confirm mapping of Test Cases to Test Scripts and Components.

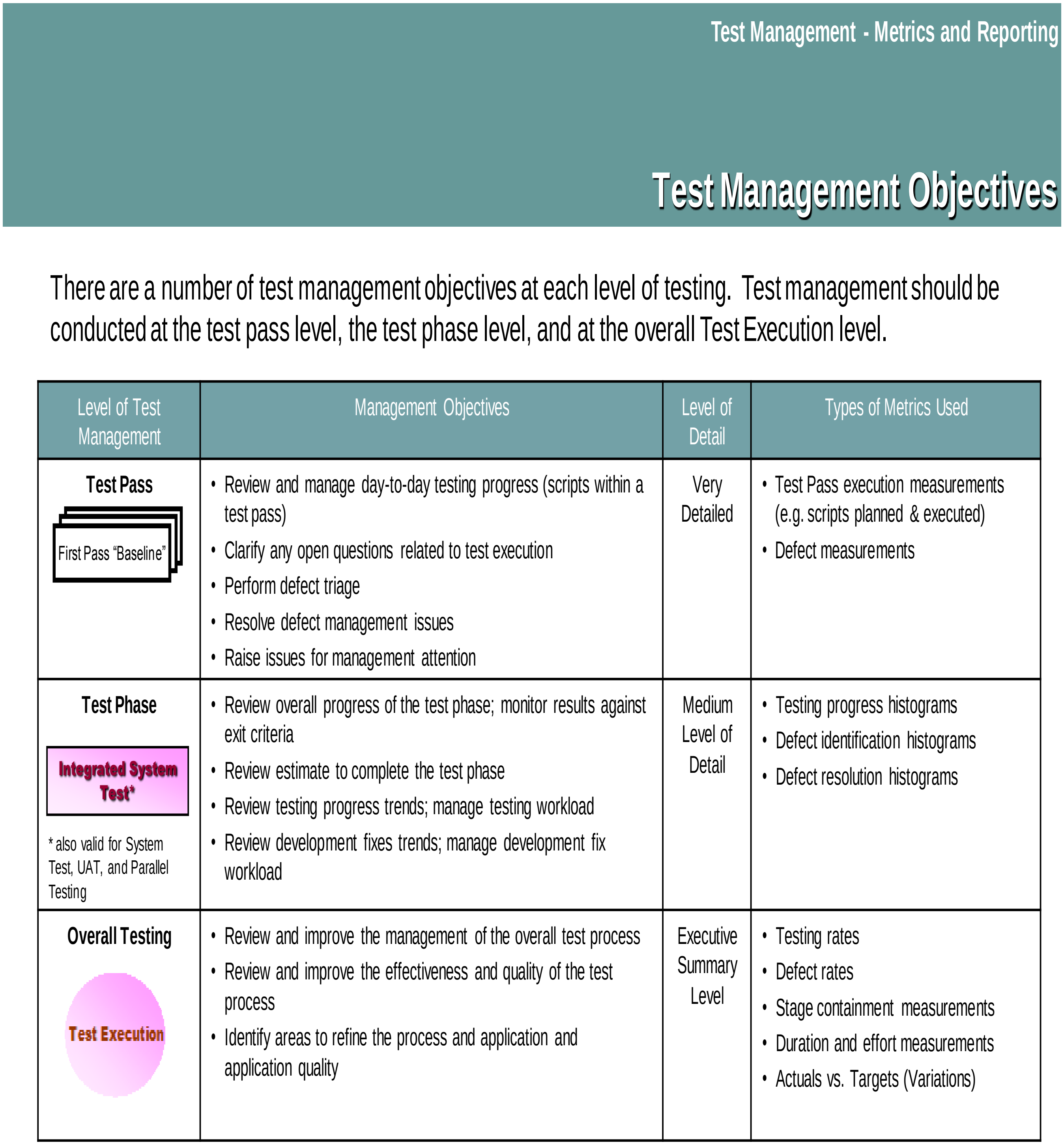
Prepare Test

- Develop detailed Test Scripts
- Define and Create Test Data
- Define detailed Expected Results

Execute Test

- Determine Actual Test Results
- Log Defects
- Manage Defects

15



Test Management - Metrics and Reporting

Test Management Objectives

There are a number of test management objectives at each level of testing. Test management should be conducted at the test pass level, the test phase level, and at the overall Test Execution level.

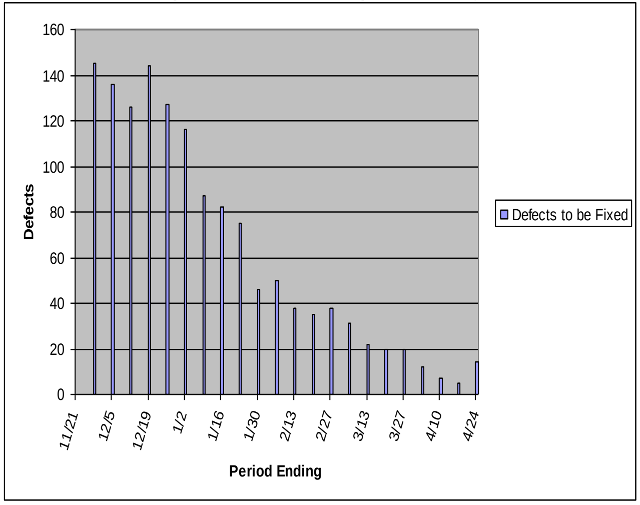
Level of Test Management	Management Objectives	Level of Detail	Types of Metrics Used
Test Pass First Pass "Baseline"	• Review and manage day-to-day testing progress (scripts within a test pass) • Clarify any open questions related to test execution • Perform defect triage • Resolve defect management issues • Raise issues for management attention	Very Detailed	• Test Pass execution measurements (e.g. scripts planned & executed) • Defect measurements
Test Phase Integrated System Test* * also valid for System Test, UAT, and Parallel Testing	• Review overall progress of the test phase; monitor results against exit criteria • Review estimate to complete the test phase • Review testing progress trends; manage testing workload • Review development fixes trends; manage development fix workload	Medium Level of Detail	• Testing progress histograms • Defect identification histograms • Defect resolution histograms
Overall Testing Test Execution	• Review and improve the management of the overall test process • Review and improve the effectiveness and quality of the test process • Identify areas to refine the process and application and application quality	Executive Summary Level	• Testing rates • Defect rates • Stage containment measurements • Duration and effort measurements • Actuals vs. Targets (Variations)

16

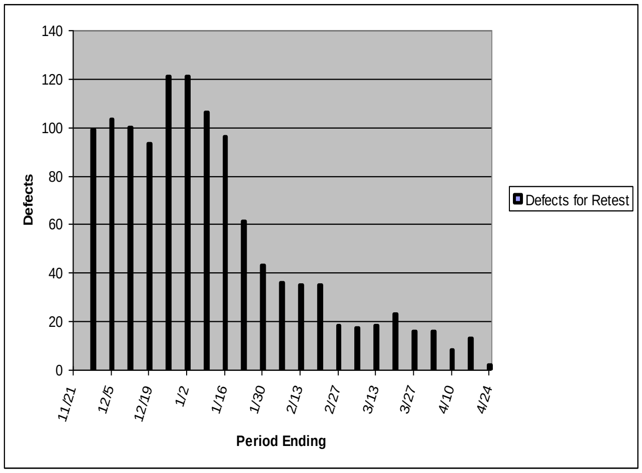
Test Phase Management Sample Reports

There are several reports that can be used for test phase management. The samples below facilitate the management the development and testing workloads.

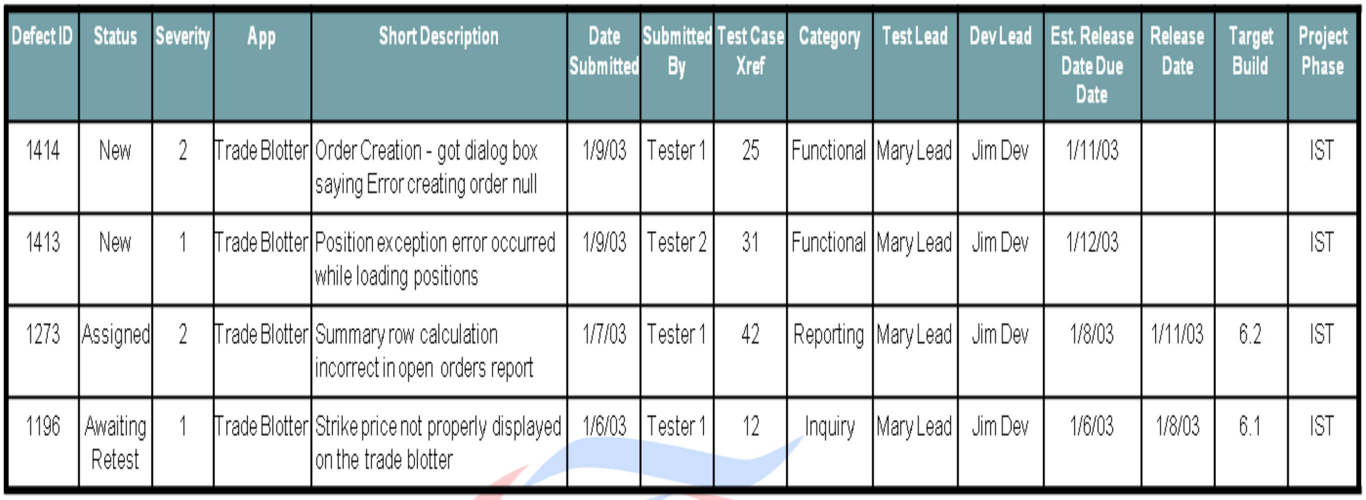
Sample Defects in Development Histogram



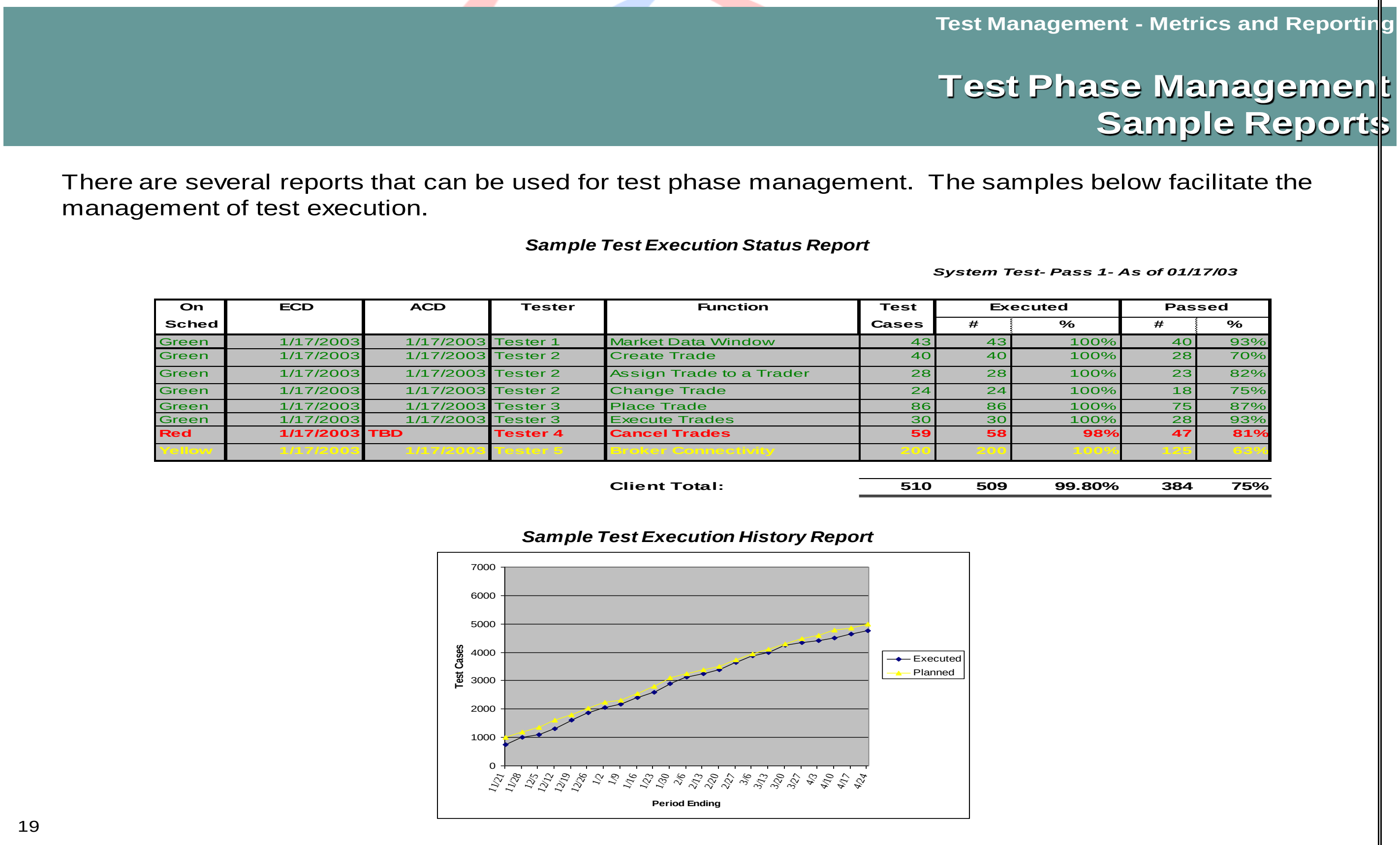
Sample Defects Awaiting Retest Histogram



One sample report for test pass management is the defect report. This view/report includes new defects that have been entered, as well as other open defects that are in the process of being fixed or are awaiting retesting.



Defect ID	Status	Severity	App	Short Description	Date Submitted	Submitted By	Test Case Xref	Category	Test Lead	Dev Lead	Est. Release Date Due Date	Release Date	Target Build	Project Phase
1414	New	2	Trade Blotter	Order Creation - got dialog box saying Error creating order null	1/9/03	Tester 1	25	Functional	Mary Lead	Jim Dev	1/11/03			IST
1413	New	1	Trade Blotter	Position exception error occurred while loading positions	1/9/03	Tester 2	31	Functional	Mary Lead	Jim Dev	1/12/03			IST
1273	Assigned	2	Trade Blotter	Summary row calculation incorrect in open orders report	1/7/03	Tester 1	42	Reporting	Mary Lead	Jim Dev	1/8/03	1/11/03	6.2	IST
1196	Awaiting Retest	1	Trade Blotter	Strike price not properly displayed on the trade blotter	1/6/03	Tester 1	12	Inquiry	Mary Lead	Jim Dev	1/6/03	1/8/03	6.1	IST



Test Management - Metrics and Reporting

Test Phase Management Sample Reports

There are several reports that can be used for test phase management. The samples below facilitate the management of test execution.

Sample Test Execution Status Report

System Test- Pass 1- As of 01/17/03

On Sched	ECD	ACD	Tester	Function	Test Cases	Executed		Passed	
						#	%	#	%
Green	1/17/2003	1/17/2003	Tester 1	Market Data Window	43	43	100%	40	93%
Green	1/17/2003	1/17/2003	Tester 2	Create Trade	40	40	100%	28	70%
Green	1/17/2003	1/17/2003	Tester 2	Assign Trade to a Trader	28	28	100%	23	82%
Green	1/17/2003	1/17/2003	Tester 2	Change Trade	24	24	100%	18	75%
Green	1/17/2003	1/17/2003	Tester 3	Place Trade	86	86	100%	75	87%
Green	1/17/2003	1/17/2003	Tester 3	Execute Trades	30	30	100%	28	93%
Red	1/17/2003	TBD	Tester 4	Cancel Trades	59	58	98%	47	81%
Yellow	1/17/2003	1/17/2003	Tester 5	Broker Connectivity	200	200	100%	125	63%
				Client Total:	510	509	99.80%	384	75%

Sample Test Execution History Report

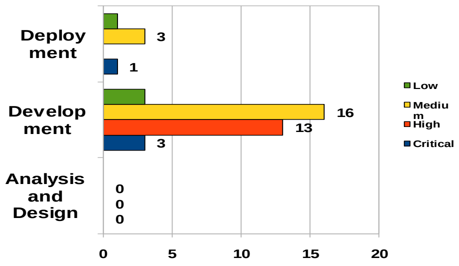
19

Test Management - Metrics and Reporting

Test Phase Management Sample Reports

Defect by Phase and Severity report

Defects by Phase & Severity



Injected in Phase	Critical	High	Medium	Low	Percentage of Total(%)
Analysis and Design	0	0	0	0	0
Development	3	13	16	3	87.5
Deployment	1		3	1	12.5
Percentage of Total(%)	10	32.5	47.5	10	

More than 50% of the defects were injected during the Development phase. This is due to

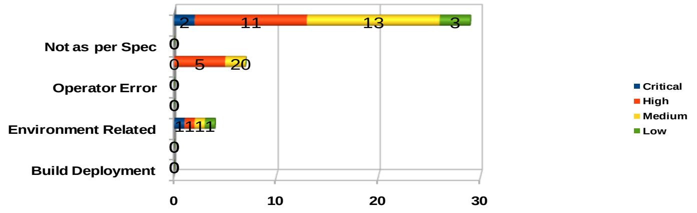
- In Development phase unit testing coverage seems to be poor. Due to the time constraints all the scenarios are not covered in unit testing.
- During Code Review code logic had not been reviewed thoroughly
- To avoid this in future, ensure more unit testing coverage and thorough code reviews

20

Test Management - Metrics and Reporting

Test Phase Management Sample Reports

Defects by Cause/Type



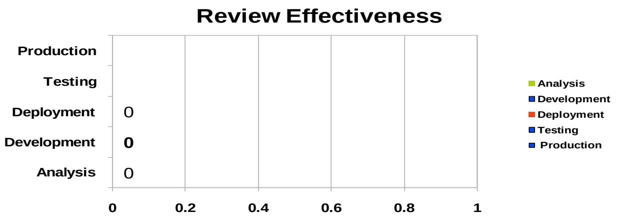
Cause/Type	Critical	High	Medium	Low	% of Total
Build Deployment	0	0	0	0	0
Coding Standards	0	0	0	0	0
Environment Related	1	1	1	1	10
Incomplete Analysis	0	0	0	0	0
Operator Error	0	0	0	0	0
User interface related	0	5	2	0	17.5
Not as per Spec	0	0	0	0	0
Process Logic	2	11	13	3	72.5

The cause of majority of defects are due to Process Logic(code logic) of development.

21

Test Management - Metrics and Reporting

Test Phase Management Sample Reports



Code Review Effectiveness

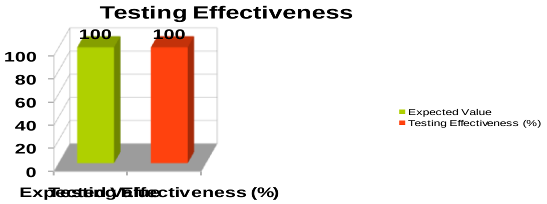
Detected in which Phase	Injected In Analysis	Injected Development	Injected In Deployment	Total	Review Effectiveness (%)
Analysis	0	0	0	**0**	**0**
Development	0	0	0	**0**	**0**
Deployment	0	0	0	**0**	**0**
Testing	0	25	9	**34**	
Production	0	0	0	**0**	
Total	**0**	**25**	**9**		

Review Effectiveness during the Design and Development was 0% which is not meeting the goal of minimum 50% of review effectiveness for the project. Review effectiveness can be ensured by covering more no of unit tests ,frequency of reviews in Design and Development phase to be improved and during the code review code logic should also be reviewed

22

Test Management - Metrics and Reporting

Test Phase Management Sample Reports



Testing Effectiveness

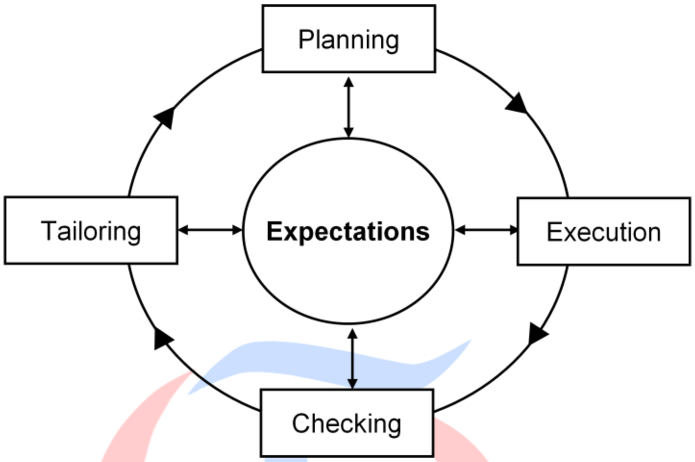
Iteration	Defects identified in Testing	Defects identified in Production	Blockers identified in Production	Breakages identified in Production	Percentage(%) of defects	Expected Value	Testing Effectiveness (%)
Iteration	40	0	0	0	0	100	100

Testing Effectiveness had met the goal set for the project

23

Testing metrics

Testing metrics can be used to facilitate continuous improvement in the testing process by identifying what went well and what can be improved.



- ✓ Planning: Use metrics to estimate the effort required for testing, and generate the planned test schedule. The metrics used to control the process are defined up front in the test approach.
- ✓ Execution: Collect and monitor metrics throughout the process to ensure that deliverables meet exit criteria, ensure that modules are testable, etc.
- ✓ Checking: Use metrics to ensure that the test process is being followed. Use metrics to evaluate where the project is in relation to the plan. Throughout the process, collect all of the raw data that you need to support the metrics defined in planning.
- ✓ Tailoring: As a result of evaluation, actions should be taken either to update the process or revise the estimating guidelines. The focus should be to continually improve within the development stage, between development stages, between releases, between projects, etc.

It is important to consider the purpose for collecting a metric. Each metric should provide useful information, making it possible to make decisions and take actions in order to achieve organizational goals.

Metrics Objectives:

Objective 1: Improve Management of the Test Process

1.1 What is the rate of test planning?

1.2 What is the rate of test preparation?
1.3 What is the rate of test execution?
1.4 What is the rate of incoming defects?
1.5 What is the rate of fixing defects?

Objective 2: Improve the Quality of the Test Process

2.1 How effective are the design, development, and test stages at containing errors in the originating stage?
2.2 How effective is the defect repair process?

Objective 3: Further Refine/Improve the Process and Application Quality

3.1 How effective is the testing process at discovering defects?
3.2 Which programs are error prone?
3.3 How many defects are in each program?
3.4 Was the testing process adhered to, from planning through execution?
3.5 Did the actual effort or duration vary from the plan? By how much?

Each metric has a specific calculation and a target. Targets should be tailored to meet the needs of the specific application and/or package/release.

No.	Metric	Calculation	Target	Source	Comments
1	Percentage of test plans documented	(Total number of test plans documented/total number of potential test plans)*100	Ideally 100% indicating all test plans were documented	Test Plan Management System	
2	Percentage of test plan signoffs	(Total number of signoffs on test plan/total number of potential signoffs)*100	Ideally 100%, indicating all test plans were signed off	Test Plan Management System	
3	Total number of documented requirements (received as input)	Total number of documented requirements	Reasonable number should be determined base on size and scope of effort	Requirements Repository	

4	Total number of test cases documented	Total number of test cases documented	Measure baseline and compare to total number over time. The target is to demonstrate improvement over time.	Test Plan Management System	
5	Test Planning Rate	Total number of hours spent creating test cases/total number of test cases documented and signed off	To meet or exceed the planned productivity in order to perform as well as or better than the budget and schedule	Test Plan Management System Time Tracking System	
6	Total number of test scripts created	Total number of test scripts created	Reasonable number should be determined based on number of cases and size and scope of effort	Test Plan Management System	
7	Test preparation rate	Total number of hours spent creating test scripts/total number of scripts created and signed off	Measure baseline and compare to total number over time. The target is to demonstrate improvement	Test Plan Management System Time Tracking System	
8	Percentage of test scripts entered into repository	(Total number of test scripts entered into repository/total number of test scripts created) *100	Ideally 100%, indicating that all test scripts were entered into the repository	Test Plan Management System	
9	Percentage of test environments established on time	(Total number of test environments established on time/total number of test environments)*100	Ideally 100%, indicating that all test environments were established on time	Project Plan	
10	Percentage of test environments	(total number of test environments validated on	Ideally 100%, indicating that all test environments were	Project Plan	

	validated on time	time/total num of envronme	validated on time		
11	Percentage of test scripts executed	(total number of test scripts executed/total number of test scripts planned)*100	Ideally 100%, indicating that all planned test scripts were executed	Test Plan Management System	
12	Test execution rate	Total number of hours spent executing the test scripts/total number of test scripts executed	To meet or exceed the planned productivity in order to perform as well as or better than the budget and schedule	Test Plan Management System Time Tracking System	
13	Total number of defects	Total number of defects	Reasonable number should be estimated base on size and scope of effort	Defect Tracking System	
14	Actual defects by program	Actual number of defects by program above a target threshold of defects.	Reasonable target number should be estimated base on size and scope	Defect Tracking System	
15	Defect rate	Total number of defects/test execution period (e.g., days(s) or week(s)	Below or at the planned defect rate	Defect Tracking System	
16	Defect fix rate	Number of defects fixed/hours spent fixing defects	Above or at the planned defect fix rate	Defect Tracking System Time Tracking System	
17	Percentage of test scripts (results) signed off	Total number of test scripts (results) signed off by stakeholders/total number of test scripts executed	Ideally 100%. Reasonable percentage should be determined by release or phase	Test Plan Management System	
18	Testing effectiveness percentage	(Number of defects found during testing for a given phase/Hours of	A reasonable target should be determined for each effort	Defect Tracking System/Time Tracking System	

		testing for a given stage) * 100			
19	Repair Effectiveness Percentage	(Total number of defects fixed correctly the first time/Total number of defects attempted to be fixed) * 100	Achieving 100 percent is ideal; this would indicate that all defects are being thoroughly fixed and regression tested.	Defect Tracking System	
20	Repair Effort Percentage (stage containment)	(total number of hours spent fixing defects from a given phase/Original number of hours to develop the phase) * 100	Ideally 0, indicating that there are no defects. Reasonable percentage should be determined by release or phase	Time Tracking System	
21	Duration variation percentage per phase	([Actual duration/planned duration]*100)-100	Ideally 0%, indicating that the phase was planned accurately and executed according to plan	Project Plan	
22	Effort variation percentage per phase	([Actual effort/planned effort]*100)-100	Ideally 0%, indicating that the phase was planned accurately and executed according to plan	Time Tracking System	
23	Percentage of personnel entering time by phase in time tracking tool	(Total number of personnel entering time by phase into time tracking tool/total number of personnel)*100	Ideally 100%, indicating all resources are tracking time by phase	Time Tracking System	

Below are sample metrics for the first three steps in the test process flow (establish test approach, plan test, and prepare test).

Test Activity	Category	Primary Indicator	Metric	Calculation
Approach	Productivity	Test Plan	Total numbers of hours spent establishing test approach	Total hours dedicated to test planning, by phase
Approach	Productivity	Test Plan	Total number of signoffs on test approach	Total number of signoffs on test approach
Plan	Productivity	Cases	Total numbers of hours spent on planning test (test cases)	Total hours dedicated to test planning, by phase
Plan	Productivity	Cases	Total number of test conditions documented and signed off	Total number of test conditions that have been signed-off
Plan	Productivity	Cases	Test Cases/Function	Test Cases/Function
Plan	Productivity	Cases	Test Cases/Release or Package	Test Cases/Release or Package
Plan	Productivity	Cases	Test Planning Rate	Total number of hours spent planning the test phase/ Total number of conditions documented and signed off for the phase
Prep	Productivity	Scripts	Total # of test scripts created	Total # of test scripts created
Prep	Productivity	Scripts	Total % of test scripts created	Total # of test scripts created
Prep	Productivity	Scripts	Total number of hours spent preparing scripts	Total hours dedicated to test planning, by phase
Prep	Productivity	Scripts	Total number of hours spent preparing scripts by phase	Total hours dedicated to creating test scripts, by phase
Prep	Productivity	Scripts	Number of scripts entered into repository	Number of scripts entered into repository
Prep	Productivity	Scripts	% of scripts entered into repository	% of scripts entered into repository
Prep	Productivity	Scripts	Test Preparation Rate	Total number of hours spent scripting the test phase/ Total number of cycles w hich have been scripted and signed off this test phase
Environment	Productivity	Environments	Test environments established on time	Test environments established on time
Environment	Productivity	Environments	Test environments vaidated on time	Test environments vaidated on time

Below are sample metrics for productivity measures of test execution using test scripts?

Test Activity	Category	Primary Indicator	Metric	Calculation
Execution	Productivity	Scripts	Total hours spent executing test scripts	Total hours of testing
Execution	Productivity	Scripts	Total hours spent executing each test phase	Total hours of testing per phase
Execution	Productivity	Scripts	Test Hrs/Function	Test Hrs/Function
Execution	Productivity	Scripts	Test Scripts/Hour	Test Scripts/Hour
Execution	Productivity	Scripts	Test Execution Rate	Total number of hours spent executing the test phase/ Total number of test scripts executed in the phase
Execution	Productivity	Scripts	Total # of test scripts executed	Total # of test scripts executed
Execution	Productivity	Scripts	Total # of test scripts In Progress	Total # of test scripts In Progress
Execution	Productivity	Scripts	Total # of test scripts NOT Started	Total # of test scripts NOT Started
Execution	Productivity	Scripts	Total # of test scripts Scheduled	Total # of test scripts Scheduled
Execution	Productivity	Scripts	Total % of test scripts executed	Total # of test scripts executed
Execution	Productivity	Scripts	Total % of test scripts In Progress	Total # of test scripts In Progress/ Total number of scripts
Execution	Productivity	Scripts	Total % of test scripts NOT Started	Total # of test scripts NOT Started/ Total number of scripts
Execution	Productivity	Scripts	Total % of test scripts Scheduled	Total # of test scripts Scheduled/ Total number of scripts
Execution	Productivity	Scripts	Total # of test scripts added	Total # of test scripts added
Execution	Productivity	Scripts	Total # of test scripts removed	Total # of test scripts removed
Execution	Productivity	Scripts	Total # of test scripts (results) signed off	Total # of test scripts (results) signed off
Execution	Productivity	Scripts	Total % of test scripts (results) signed off	Total % of test scripts (results) signed off

Below are sampled metrics for software quality measures for test execution and a general productivity category that applies to all steps in the test process.

Test Activity	Category	Primary Indicator	Metric	Calculation
Execution	Quality	Defects	Total # of defects found	Total # of defects found
Execution	Quality	Defects	Total # of defects fixed	Total # of defects fixed
Execution	Quality	Defects	Total # of outstanding defects	Total # of outstanding defects
Execution	Quality	Defects	Number of defects by module	# of defects per module
Execution	Quality	Defects	Total defects by phase	# of defects by phase
Execution	Quality	Defects	Total defect by severity	# of total defects by severity
Execution	Quality	Defects	Total defect by severity by phase	# of total defects by severity by phase
Execution	Quality	Defects	# of defects by type (application, data, enviroment, script)	# of defects created for each category
Execution	Quality	Defects	# of defects by component (online, batch, report, etc)	# of defects created for each component
Execution	Quality	Defects	# of defects by component complexity	# of defects for each component, by component complexity
Execution	Quality	Defects	Defect rate	number of defects / days or weeks of execution
Execution	Quality	Defects	Defect ratio	# of defects /software size
Execution	Quality	Defects	Defect density	# Defects/KSLOC (thousand lines of code)
Execution	Quality	Scripts	Total # of test scripts Passed	Total # of test scripts Passed
Execution	Quality	Scripts	Total % of test scripts Passed	Total # of test scripts Passed/ Total number of scripts
Execution	Quality	Scripts	Total # of test scripts Failed	Total # of test scripts Failed
Execution	Quality	Scripts	Total % of test scripts Failed	Total # of test scripts Failed/ Total number of scripts
All/General	Productivity	time/schedule	Duration Variance Percentage	[(actual duration/planned duration) *100]-100
All/General	Productivity	work/effort	Effort Variance Percentage	[(actual effort/planned effort) *100]-100
All/General	Productivity	work/effort	Number of Personnel entering time into time management tool	Number of Personnel entering time into time management tool
All/General	Productivity	work/effort	% of Personnel entering time into time management tool	% of Personnel entering time into time management tool

Below are sample metrics for productivity measures of test execution using defects.

Test Activity	Category	Primary Indicator	Metric	Calculation
Execution	Productivity	Defects	Average time to fix a defect for a given phase	Total # of defects/Total time to fix defects
Execution	Productivity	Defects	Number of hours spent fixing defects at a given phase	Total hours to fix defects by phase
Execution	Productivity	Defects	Number of defects fixed correctly for a given phase	# of defects fixed per phase
Execution	Productivity	Defects	(Stage Containment) Repair Effort Percentage	(# of hours spent repairing defects from a given stage/original number of hours to build the stage)*100
Execution	Productivity	Defects	Repair Effectiveness Percentage	(# defects fixed correctly the first time for a given phase/total # attempted fix for the phase)*100
Execution	Productivity	Defects	Average Turnaround Days of Defects by priority	Sum of all Turnaround Days of Defects of Priority P resolved during the week/Number of Defects of Priority <P> resolved during the week
Execution	Productivity	Defects	Best Defect Turnaround by priority	Minimum among the Turnaround Days of All Defects of Priority <P>
Execution	Productivity	Defects	Worst Defect Turnaround by priority	Maximum among the Turnaround Days of All Defects of Priority <P>
Execution	Productivity	Defects	Defect Fix Rate	number of defects fixed / hours spent fixing defects
Execution	Productivity	Defects	Defect Closure Rate by priority	Defects of Priority <P> whose Date Closed falls within the start and end of reporting week
Execution	Productivity	Defects	Defect Emergence Rate by priority	Defects of Priority <P> whose Date Open falls within the start and end of reporting week
Execution	Productivity	Defects	Testing Effectiveness Percentage	(Number of defects found during testing for a given phase/ Hours of testing for a given phase) * 100

TESTING PROCESS- BEST PRACTISES

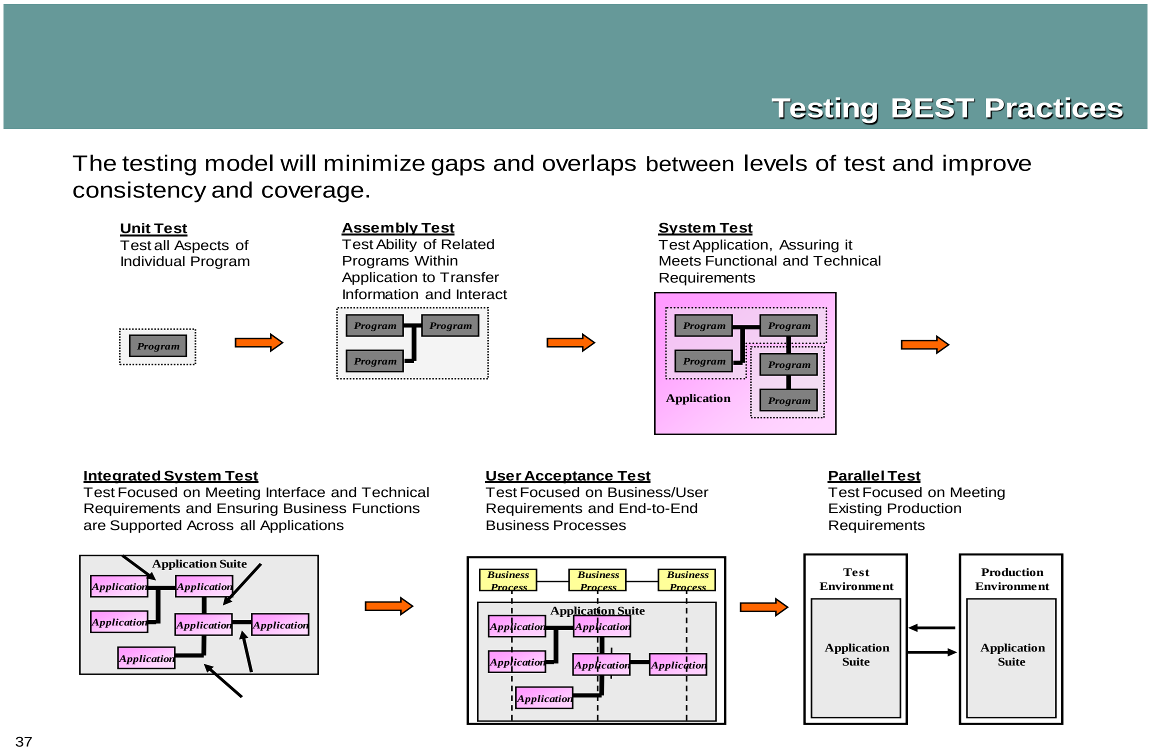
The testing methodology is based on the following principles:

- ✓ Plan Early. This facilitates starting the test on time and starting early. It includes developing an overall testing approach at the onset of the project and/or program, and developing a test approach and plan for each test stage concurrently with the corresponding specification stage.
- ✓ Test the most important things first. The testing effort should be sequenced based on customer values. Understand what the customer's values are, and then prioritize the work for all specification and testing stages accordingly
- ✓ Minimize gaps and overlaps in testing by clearly defining the objectives of each test stage and establishing entry and exit criteria to ensure that the objectives are met
- ✓ Define test cases and cycles as part of specification development in order to ensure that the specification is complete and can be tested
- ✓ Develop well-documented, repeatable test models to facilitate analysis of problems and regression testing in the current release, as well as testing future releases
- ✓ Automate testing. Tools currently exist for documenting test models, issue tracking, script recording and playback, data generation and manipulation, comparison of actual to expected results, and configuration management. Using these tools simplifies the testing process, and can result in significant cost and schedule savings.
- ✓ Implement validation and verification techniques for each specification and test stage to facilitate early detection of problems, making the problems less costly to correct.
- ✓ Stage Containment is an approach used to identify problems in the application before they pass to the next stage, with the goal being to minimize the number of problems being passed to the next stage
- ✓ For the purpose of stage containment, problems can be sorted into categories: errors are problems found in the stage were they were created; defects are problems found in a stage successive to the stage where they were created; faults are problems found after implementation (i.e., production problems)
- ✓ The process of determining the stage that was the origin of the defect is called root cause analysis
- ✓ Entry and exit criteria state what is required from previous processes to support a given stage (entry criteria) and what is required of a given process to determine completeness (exit criteria)
- ✓ Entry and exit criteria are defined for each stage to assure quality deliverables from one stage to the next

- ✓ The testing model specifies that activity in one stage must be complete before moving on to the next stage
- ✓ It is key that exit criteria defined for that stage have been met
- ✓ Entry and exit criteria should be defined for each specification stage as well as for each test stage
- ✓ Exit criteria from requirements analysis and design will include documenting a test planning approach and cases.

Stage containment, entry and exit criteria, and root cause analysis are key concepts in the test process.

- ✓ Stage Containment is an approach used to identify problems in the application before they pass to the next stage, with the goal being to minimize the number of problems being passed to the next stage
- ✓ For the purpose of stage containment, problems can be sorted into categories: errors are problems found in the stage were they were created; defects are problems found in a stage successive to the stage where they were created; faults are problems found after implementation (i.e., production problems)
- ✓ The process of determining the stage that was the origin of the defect is called root cause analysis
- ✓ Entry and exit criteria state what is required from previous processes to support a given stage (entry criteria) and what is required of a given process to determine completeness (exit criteria)
- ✓ Entry and exit criteria are defined for each stage to assure quality deliverables from one stage to the next
- ✓ The testing model specifies that activity in one stage must be complete before moving on to the next stage
- ✓ It is key that exit criteria defined for that stage have been met
- ✓ Entry and exit criteria should be defined for each specification stage as well as for each test stage
- ✓ Exit criteria from requirements analysis and design will include documenting a test planning approach and cases.



Testing BEST Practices

The testing model will minimize gaps and overlaps between levels of test and improve consistency and coverage.

Unit Test
Test all Aspects of Individual Program

Assembly Test
Test Ability of Related Programs Within Application to Transfer Information and Interact

System Test
Test Application, Assuring it Meets Functional and Technical Requirements

Integrated System Test
Test Focused on Meeting Interface and Technical Requirements and Ensuring Business Functions are Supported Across all Applications

User Acceptance Test
Test Focused on Business/User Requirements and End-to-End Business Processes

Parallel Test
Test Focused on Meeting Existing Production Requirements

37

Best practices

1. Create test case based on requirements – Business requirements are the input to test planning. Define test cases based on detailed requirements. Develop multiple cases to test individual business requirements when necessary.
2. Validate test cases – Validate that the test cases are defined correctly to test requirements and that cases provide adequate coverage of requirements. Review with business/functional experts.
3. Create scripts based on test cases – The scripts for each test run are created from the test cases defined during test planning. This includes creating input data and detailed expected results.
4. Validate test scripts – Once the scripts are complete, they should be verified. This will ensure that all the cases were accurately scripted, the proper development process was followed, and testing standards were followed. The developer of the scripts should desk check the scripts for standards and accuracy. In addition, a walkthrough or formal inspection with the development cell or team and subject matter experts may be required in areas of high complexity or risk.
5. Scripts should be thorough and comprehensive – Provide a systematic check-out of all of the functions of the system, all classes of valid input must be accepted, all classes of invalid input must be rejected, and all functions must be exercised. Scripts may include environment setup (restore, restart, backup) and data conversion jobs.
6. Target high risk functions – It is not always possible to test everything. Scripts should target high risk functions first for more extensive testing.

7. Validate all processing types – Scripts should cover online as well as batch processing. Reports should also be tested and validated.
8. Maintain traceability – Map test cases to requirements, components and scripts. This ensures that the test documents are maintainable. Traceability is very important when the requirements or test cases change and it becomes necessary to update the test scripts.
9. Repeatable – Manual scripts need to be detailed enough for anyone to follow (see slide 9 - Test Script Template and Definitions).
10. Manual and automated – Test scripts may be manual or automated. Leverage automated testing based on a test automation ROI. Manual test scripts should be defined with an eye toward automation by providing the appropriate level of detail to allow scripts to be easily automated (e.g., outlining the detailed steps to execute a test case rather than simple stating to execute the case)
11. Include audit information – Include audit information as part of a test case or test script document, particularly if the document is stored in Microsoft Word or Microsoft Excel. This helps to facilitate maintenance.

Version #:		<Enter the version number of the test case>
Change Description		<Enter the description of the change>
Prepared By:		<Name of business analyst>
Date:		<Date prepared>
Reviewed By:		<Name of reviewer>
Date:		<Date reviewed>

Test Execution Best practices

1. Define Test Execution Entry and Exit Criteria - It is critical to define specific entry and exit criteria, and to communicate these criteria to the associated design phase and the corresponding test phase. This will ensure that the entire test process results in a quality solution that runs smoothly. Entry and exit criteria also establish clear boundaries for the test phases. By knowing when one test is ending and the next is beginning, duplication of testing can be avoided.
2. Prioritize Test Execution Based on Risk - Assess the the impact of system changes made with respect to how critically they affect the business. High-risk changes should have increased testing focus and possibly additional resources allocated. For mission critical systems, for example, the testing effort may be increased whereas testing in a

system which is not critical can be reduced. The risk assessment allows management to determine the effort and prioritization of what needs to be tested.

3. Group and Prioritize Test Scripts - To assist with test execution, test cases and test scripts can be prioritized and possibly grouped together. For example, scripts can be grouped according to application functionality, and key functions can be executed first. This also helps corresponding defects to be grouped and prioritized for fixing and retesting.
4. Leverage Automated Execution - Automated execution is typically used to create playback scripts or to automate data comparison. Automation can reduce the time spent on redundant testing, especially when common tools are used for test planning, preparation, and execution.
5. Implement a Defect Tracking Tool and Defect Management Process - Defect tracking and management are critical communication points between the development and test teams. The defect process and the use of shared defect tracking tools will facilitate the identification of defects, the prioritization of defects, the anticipated timing for fixes, the re-test of fixes, and help improve stage containment. Defect management is an integral part of test execution, and details will be addressed further in the Defect Management topic.
6. Recognize That Defects Are Not Always Limited to Program Problems - There are many sources of problems that will be identified during testing besides application code problems. For example, problems with procedures, reports, forms, training materials, documentation, the application design, or systems software all could be identified during tests. Alternatively, the application may be operating correctly and the test scripts may be incorrect. In this case, the defect should still be documented to facilitate the necessary corrections.
7. Facilitate Communication Between Teams - Good communication between the development team and the testing team is critical. Effective testing and defect resolution requires the testing team to have clear communication and expectations with the development team.
8. Maintain a Test Bed of Data - Common test data describes the test data that can be used across multiple test stages and be maintained for multiple projects. The use of common data reduces the amount of test preparation required before execution. A baseline database can be used to create a facilitate test execution and provides the ability to rerun a given test case or scenario whenever required. When defining a test bed of data, formal processes should also be in place for maintaining the data.
9. Implement Testing Progress and Quality Reviews - During the testing phase, progress and quality reviews can be held at project checkpoints to ensure the project is proceeding on schedule. The quality reviews should be conducted to discuss status, issues and risks. The team should raise any issues, identify action to be taken and

follow up with key stakeholders. This will be addressed in further detail in the Reporting & Metrics topics.

Defect Management Best Practices

1. Maintain a defect log - Consider using a single defect tracking tool, or defect log for all defects from System Test through Production. The benefit of a defect log is that it facilitates phase containment. The defect log is also the source for metrics and reports which can be used to determine the quality of the process and product. This information is used to manage the fix effort, and to communicate the status of problems to the teams.
2. Manage Defects – Defect management and tracking facilitate communication between the development and test teams. The process of defect tracking will determine the turnaround time for a fix, the escalation procedures for fixes, and a common definition of the severity of a defect. Defects should be managed to ensure that all defects are logged (for future analysis), and that only approved problems are addressed. Fixes should be coordinated with test execution schedule requirements. Therefore, it is best to have the team lead or management assign target completion dates or target releases for each defect.
3. Identify Source and Phase of Defects - Key concepts are to properly identify the source of the defect and the phase in which the defect was identified. The problem could be from incorrect design, incorrect code, or incorrect testing. By properly identifying the source, better metrics can be kept to investigate the effectiveness of stage containment.
4. Use consistent definitions – Defects should be logged and managed using a consistent method to define a defect and assign defect severity.

5. Manage with facts - Metrics provide objective data upon which decisions can be made, actions can be taken, and goals can be
   achieved.
6. Focus on the problem, not the symptom - Metrics facilitate a better understanding of problems. For example,if application product test is behind schedule, the assumption can be made that the estimates were bad. If the metrics are good, they will initiate an investigation which may point to a number of reasons for being behind schedule: vague requirements, code that was not component tested, unstable environment, etc.
7. Facilitate predictability - If appropriate metrics are collected and used accurately and consistently, one can predict the quality and productivity of the remaining work, the next development stage, the next release, etc.

8. Facilitate continuous improvement - Throughout the testing process, one can review the metrics, determine the problems, and improve the process or the estimating guidelines. This process should be continuous. A project can learn from previous mistakes and benefit by eliminating the recurrence of those difficulties. An example would be a project implementing scripting standards to allow personnel developing scripts more time to concentrate on the content, as opposed to the format, of the scripts. Improvement can occur from development stage to development stage, release to release, and project to project.

Additional Responsibilities includes:

1. Thinking beyond BRD
2. Maintaining ENV downtime tracker
3. Regression optimization
4. Exploratory Testing
5. Automation expertise knowledge even for manual Testers
6. Maintaining guidelines for each activity
7. Updating below trackers on timely basis
8. Query Tracker
9. Review Tracker
10. Env down time tracker
11. Generating accurate metrics
12. Defect Root cause Analysis and Trend Analysis
13. More collaborative and eary conversations with BA/Dev team
14. More Business related training and effective KT documentation

Risks and Issues in the Project

1. QA coverage for Regression Testing is 80%
2. Limited Regression Coverage
3. Last minute change in scope and Release
4. No documented requirements:
5. Requested to Test security but no documented requirements
6. High defect fail rate and open defects
7. Delay in fixing defects impacts schedule
8. Less time given for execution and Regression
9. Test Scripts writing is delayed
10. Some requirements are missing and not updated
11. The build is delayed

12. Environment issues
13. Resource issues
14. Lack of Business and design documentation
15. Requirement keep changing affecting Test schedule
16. Requirement freeze not defined even after QA testing phase
17. Risks and Issues in Functional Testing Project:
18. QA coverage for Regression Testing is 80%
19. Limited Regression Coverage
20. Last minute change in scope and Release
21. No documented requirements:
22. Requested to Test security but no documented requirements
23. High defect fail rate and open defects
24. Delay in fixing defects impacts schedule
25. Less time given for execution and Regression
26. Test Scripts writing is delayed
27. Some requirements are missing and not updated
28. The build is delayed
29. Environment issues
30. Resource issues
31. Lack of Business and design documentation
32. Requirement keep changing affecting Test schedule
33. Requirement freeze not defined even after QA testing phase

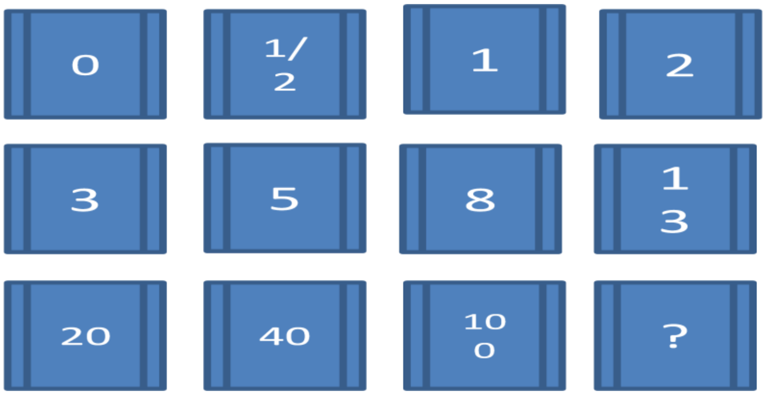
Agile Software Project Estimations
Introduction:

- Necessity to Estimate
  - To understand the complexity of a given requirement.
  - To prioritize the requirements based on complexity in-order to deliver them within a stipulated deadline.
  - To identify the complexity and distribute them equally among the testers.
- Who does the estimation?
  - In most other methodologies, the estimation is done only by the business, client or the manager.
  - However, in Agile the estimation is done with the team members.

- When team members estimate any requirement based on their capacity, they will be able to meet the assured deadline without breaching the SLA as the estimation is directly acquired from them.
- There must always be a Scrum Master during this estimation who acts as a mediator between the team members and the product-owner.
- Scrum Master will not estimate but will be a host for the show.

- Effective software project estimation is one of the most challenging and important activities in software development.
- Proper project planning and control is not possible without a sound and reliable estimate
- There are many ways of estimate a project and for Agile project the most common used is Planning Poker
- Planning Poker – widely used technique for Agile Estimation.
    - Estimating Steps

- The Scrum Master, Product-Owner and the team members sit for estimation together.
- Each team member possesses a set of cards for estimating the complexity of user stories.
- The cards follow Fibonacci Sequence and are numbered in the following manner;



Planning Poker cards

How to estimate using planning poker?

- A specific user story is picked up from the product backlog for estimation.
- The requirement is discussed by the product-owner and the team members share their knowledge with respect to the requirement.
- Now the team members are completely aware of the complexity of the requirement.
- Opportunities are given for the team members to clarify their doubts with respect to the user story.
- Once every aspect of the user story is clarified, the team members are now asked to begin the estimation.
- From the set of cards which each team member possesses, each pick up a number to decide the complexity of the user story.
- Once the card is picked they will place it upside down on the table so that no one is able to see the flip side of it.
- Once everybody would pick up, they place their card on the table with the number hidden.
- The Scrum Master then confirms with everyone if estimation is done and he requests all of them to open the cards.
- Each team member would have picked up a specific number, not all will be the same.
- So each of them support with reasons for choosing the specific complexity and the discussion continues until they all agree to a particular number from the Poker cards.
- If required, estimations are repeated for the same user-story until every team member agrees to the complexity.

Estimation Technique 2: Agile Estimation Project Based on Objective Criteria:

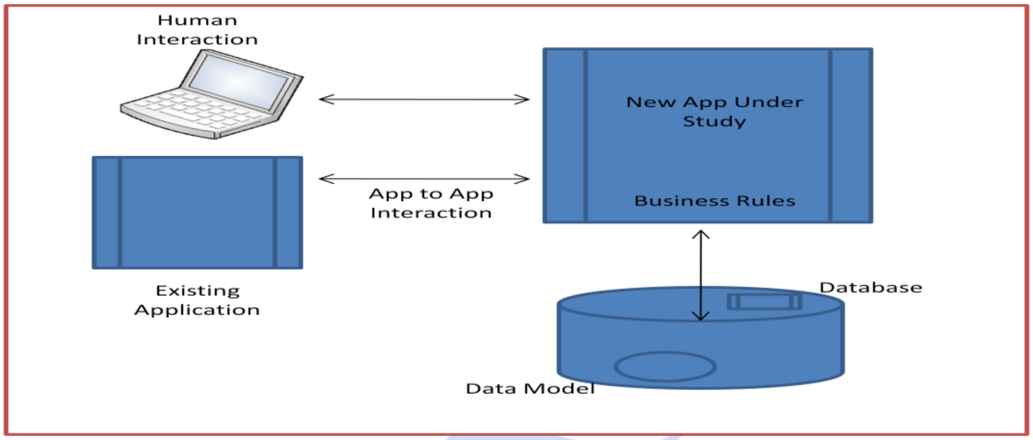
As you can see on this diagram an application is nothing more than :

(1) Some business users trying to interact with some working code that implements

(2) some business rules running against

(3) a model containing some business entities, whose values are stored in the physical database

(4) which it is to create, read, update, or delete



So then, we'll estimate the user story or Product Backlog Item, one type at a time:

1. Interaction type
2. Business rules
3. Number of entities manipulated
4. Data to be created, read, updated, and deleted (CRUD)

For Interaction Type use the following table to calculate the value

Interaction Type	Description	Value
Simple	**Well Defined Interface**	**1**
Average	**Dynamic Interface**	**2**
Complex	**Human Interaction**	**3**

If the story you are looking at requires a human interaction, you should give it a value of 3. If it requires, however, only an interaction with another application, according to a well defined protocol, then that user story should get a value of 1 for the interaction

Use the following table to calculate the complexity based on the number of business rules to be applied

Business Rules	Description	Value

Simple	1 Rule	1
Average	1 – 3 Rules	2
Complex	> 3 Rules	3

If there is only one business rule, then you should give the story a value of 1. If there is more than one rule but less than three, then that story should get a value of 2. If there are more than three rules, give the story a value of 3

Use the following table to calculate the complexity based on the number of data entities needed to execute this user story

Entities	Description	Value
Simple	**1 Entity**	**1**
Average	**1 – 3 Entities**	**2**
Complex	**> 3 Entities**	**3**

The number of entities manipulated means that if the number of data entities is only one, then you should give that story a value of 1, but if it is between two and three, then that story should get a value of 2, and so on.

Use the following table to calculate the complexity based on the data manipulation (CRUD) factor

Entities	Description	Value
Simple	Read, Delete	1
Average	Create	2
Complex	Update	3

Once we have calculated the complexity for each type at a time then we need to calculate the Unadjusted Points (UP)

Imagine that our results were as follows:

A. Interaction Type = 3 points

B. Business rules = 1 point

C. Number of entities manipulated = 1 point

D. Create, Read, Update, Delete (CRUD) = 2 points

To calculate the UP we just need to add all the points together

On this example the UP = 7

For each of these dimensions, a higher value indicates higher team ability or capability, whereas a lower or minus value indicates a lower team ability or capability. A zero will mean the lowest score, while a positive value indicates a high level of ability or capability with 2 being the maximum.

Organization dimension	
Factor	**Value Range (0/2)**
Have different departments worked successfully together on a Scrum project previously?	
Does some strong resistance exist within the organization with regard to Scrum?	
Does a great support for Scrum exist between different departments within the company?	
Development infrastructure dimension	
Factor	**Value Range (0/2)**
Is automatic testing already in place and a common practice?	
Is continuous integration testing already in place and a common practice?	
Is daily build environment already in place and a common practice?	
Team dimension	
Factor	**Value Range (0/2)**
Is the team completely new to Scrum?	

Have the team members successfully worked together before?	
Do team members know well and appreciate one another?	
Technology dimension	
Factor	**Value Range (0/2)**
Is the development team very experienced in the programming language?	
Are development team members very experienced in the technology to be employed?	
Is a Scrum production environment already ready?	
Process dimension	
Factor	**Value Range (0/2)**
Is Scrum the company's adopted process framework?	
Is there a good support for Scrum within the company?	
Is there strong resistance against Scrum within the company?	
Business dimension	
Factor	**Value Range (0/2)**
Is there a Product Owner fully available and completely engaged with the team?	
Is the product owner familiar with Scrum but has no practical experience?	

Has the Product Owner successfully used Scrum before?	

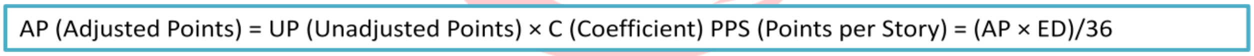
Depending on this total value, three scenarios will be possible:

If the ED value is between 0 and 11, then the multiplication coefficient C will be 2. This implies that the environment dimensions are such that the team will not be able deliver as many stories during the Sprint than if the ED score had been higher.

If the ED value is between 12 and 23, then the multiplication coefficient C will be 1. This implies that the environment makes the team job neither difficult nor easy.

If the ED value is between 24 and 36, then the multiplication coefficient C will be ½. This implies that the environment dimensions are such that the team should be able to deliver more stories during the Sprint.

To calculate the total value in points for a single story, simply use the following formula:



AP (Adjusted Points) = UP (Unadjusted Points) × C (Coefficient) PPS (Points per Story) = (AP × ED)/36

Imagine that the values of our environment dimensions (ED) are equal to the following coefficient for every dimension listed below:

1. Organization = 3 2.
2. Infrastructure = 2 3.
3. Team = 4 4.
4. Technology = 3 5.
5. Process = 2 6.
6. Business = 4

Adding up all of these values gives us an ED that is equal to 18

Since ED is equal to 18, this would mean, as previously mentioned, that the coefficient of multiplication to be used will be equal to 1

AP = UP × C

AP = 7 × 1 = 7

(With UP equal to 7 points, as was calculated previously).

Then,

PPS = (AP × ED)/36

PPS = (7 × 18)/36 = 126/36 = 3.5 points

Using the same formula for every other story, you should have a matrix like below example that provides the overall estimate for the entire product to be built

	Characteristics				**Total Up** **(Unadjusted Points)**	**Coefficient**	**AP** **Adjusted Points**	**ED** **Env** **Dimensions**	**PPS** **(=AP’ED)/36)**
PBIs (Story)	Interaction Type	Business Rules	Entities	Data Manipulation Type					
Sprint1									
Sign In to app	3	1	1	2	7	1	7	12	3.5
Add data	3	1	1	2	7	1	7	12	3.5
Delete Data	3	1	1	3	8	1	8	12	4
Sprint 2									
Edit Data	3	1	1	3	8	1	8	12	4

Cancel Updates	3	1	1	3	8	1	8	12	4
Browse Records	3	1	1	3	8	1	8	12	4
Total (Sprint 1 + Sprint 2)								108	23

The advantage of this type of calculation is that it is based on objective criteria; therefore, it is more appropriate for comparison between different teams and even between different members of the same project team As a consequence of Scrum success, more and more companies are contemplating rolling out Scrum to their entire IT department

One of the impediments to this is the fact that the story point value is unfortunately not comparable between teams. With the weight of the velocity so different from one team to another, you can see why it has become a problem for many Agile PMOs to plan an enterprise-wide deployment of Scrum. With this technique based on an objective criteria-based estimating process in the form of a series of relatively straightforward tables to guide the team in their effort to estimate the different stories As simple as it is, this method allows us to make objective comparisons among teams as well as among different members of the same Scrum team.

SESSION - 06

INSURANCE DOMAIN

Insurance Concepts

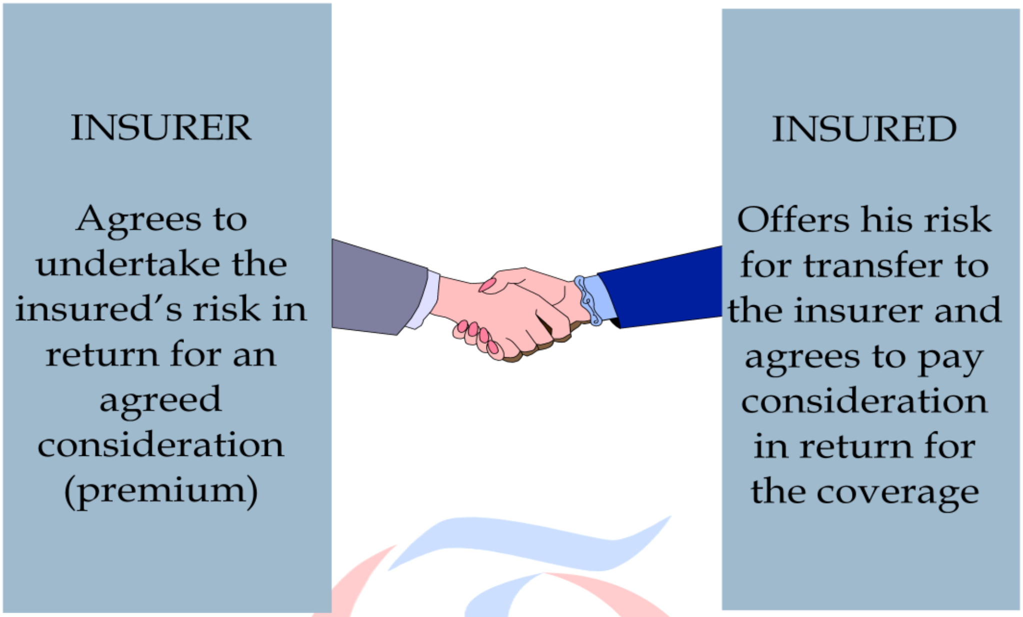
Risk is uncertainty about the future. Insurance is Risk Management Technique

Insurance: A transfer system, in which one party – the insured – transfers the chance of financial loss to another party – the insurer.

In insurance parlance, risk is the

- uncertainty about potential losses
- which could cause financial setbacks
- Only Pure Risks are insurable
- Insurance results in the risk being transferred to the insurance company
- Sharing of risk among large groups is the basis for insurance

Insurance is a contract ..



INSURER

Agrees to undertake the insured's risk in return for an agreed consideration (premium)

INSURED

Offers his risk for transfer to the insurer and agrees to pay consideration in return for the coverage

Functions of Insurance

Primary functions

- *Collective bearing of risk*
- *Provide certainty*
- *Provide protection*

Secondary functions

- *Prevention of losses*
- *Small capital to cover large risks*
- *Contribution towards development*
- *Source of savings, therefore investments*
- *Foreign exchange earnings*
- *Risk free trading*

- **Law of Large Numbers** - A Mathematical principle which enables the insurers to make predictions about losses.
- It states that as the number of similar but independent exposure units increases, the relative accuracy of predictions about future outcomes (losses) based on these exposures also increases.
- An exposure unit is a measure of loss potential and is used in pricing insurance.

E.g.: In a HO-W Insurance, each home is an exposure unit. The insurer insures thousands of home owners who face the same uncertainty.

- **Premium**: Is a small periodic payment the insured pays for long term insurance coverage.

Calculation of Premium:

Premium = Rate x Number of exposure units

Ideally Insurable Loss Exposures:

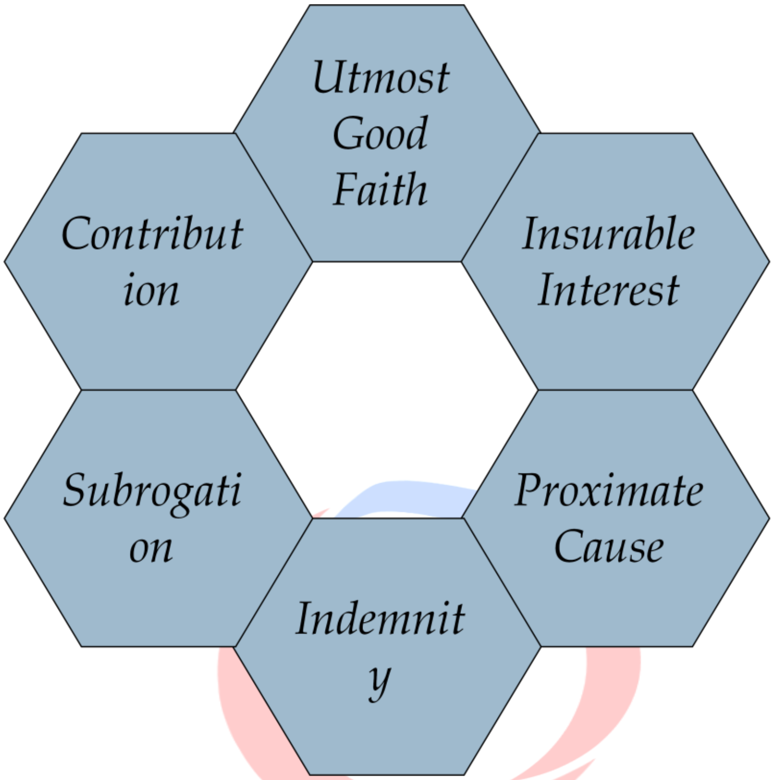
- Insurance companies generally prefer to provide insurance for the loss exposures that have the following characteristics:

a. Large number of similar exposure units.
b. Losses that is accidental
c. Losses that definite and measurable.
d. Losses that is not catastrophic.
e. Losses that is economically feasible to insure.

Check Point:

- ✓ Voice of a singer…is it an ideal loss exposure???
- ✓ Gambling is it an ideal loss exposure??
- ✓ Robbery \Theft is it an ideal loss exposure???

Principles of Insurance



Utmost Good Faith

Contribution

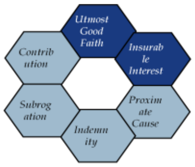
Insurable Interest

Subrogation

Proximate Cause

Indemnity

Principles in detail



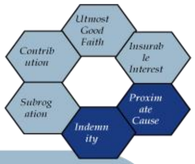
Utmost good faith

- *The Applicant (Policyholder) is bound to make full disclosure of all material facts*
- *The Insurer is bound to make full disclosure of all provisions in the contract*
- *(Material Facts - Facts that influence the insurer's judgment and rate of premium to be charged)*

Insurable interest

- *Insured should have a pecuniary (financial interest) in the subject matter of insurance and he should suffer a financial loss on the occurrence of the insured event*
- *Helps to separate insurance from gambling*
- *Implies legal right to insure*
- *Does not arise from relationship alone*

Principles in detail



Proximate cause

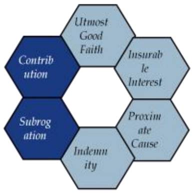
- The admissibility of a claim under a policy is determined by confirming whether the proximate cause leading to the event is covered
- It is defined as:

~ the active efficient cause
~ that sets in motion a train of events
~ which brings about a result
~ without the intervention of any forces from a new/independent source

Indemnity

- An insured can recover a loss under a policy only if he has insurable interest
- He can recover the loss only to the extent of his insurable interest
- Insurance of persons are usually not policies of strict indemnity, because the value of life is not exactly ascertainable

Principles in detail



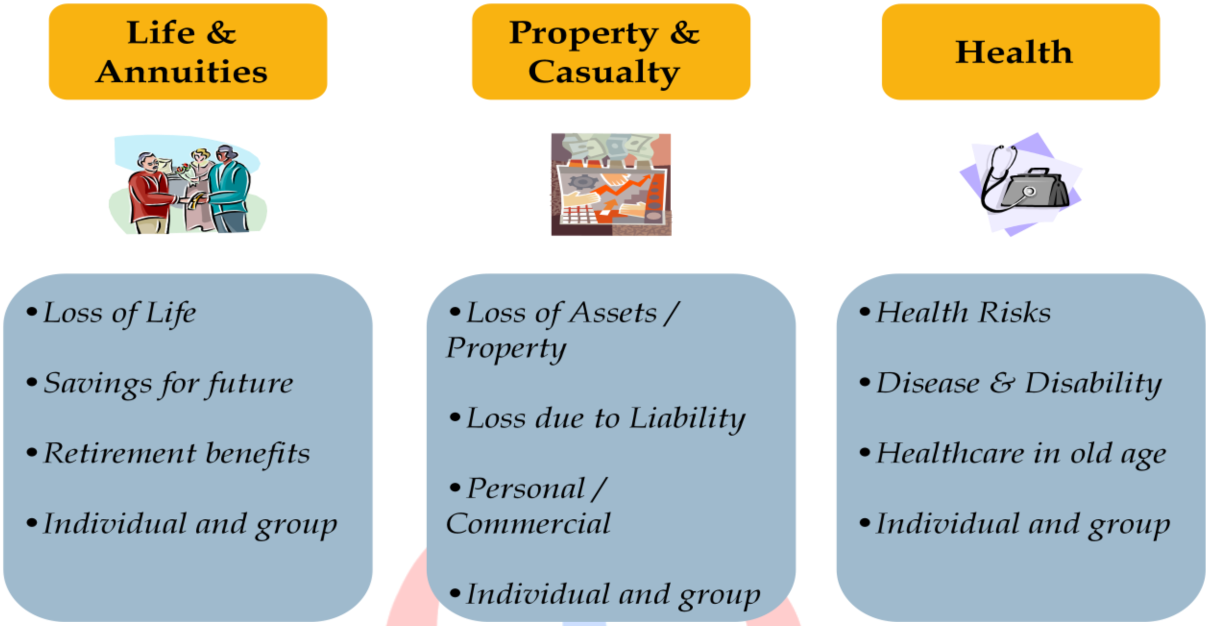
Subrogation

- Subrogation is the transfer of rights of rights and remedies of the insured to the insurer who has paid the loss
- This arises out of the indemnity principle that the insured can not recover more than his insurable interest
- Obviously not applicable to Life Insurance

Contribution

- Contribution is the right of an insurer who settles a claim to recover from other insurers who are liable for the same loss
- Like subrogation, not applicable to Life Insurance

Insurance industry segments



Life & Annuities

- *Loss of Life*
- *Savings for future*
- *Retirement benefits*
- *Individual and group*

Property & Casualty

- *Loss of Assets / Property*
- *Loss due to Liability*
- *Personal / Commercial*
- *Individual and group*

Health

- *Health Risks*
- *Disease & Disability*
- *Healthcare in old age*
- *Individual and group*

Types of Insurance

- **Property Insurance**: Property Insurance covers the costs of accidental losses to an insured's property. The insured could be a person insuring his house and personal property or a business insuring its building, inventory and equipment.
- Provides Insurance Cover for property against damages
- Generally classified as Marine, Fire, Motor, & Miscellaneous insurance

Line of Insurance Business:

- Line of insurance is just another way of saying type of insurance.
- **Personal Lines**: is any type of insurance purchased by individuals and families to cover non business loss exposures.

- **Commercial Lines**: Insurance is any type of insurance that covers loss exposures for business and organizations.

Check Point….

- ✓ Can you name some Policy types which come under Personal Lines??
- ✓ Can you name some Policy types which come under Commercial Lines?

E.g.: Fire & Allied lines, Crime, Ocean & inland marine insurance come under Property insurance.

- **Liability Insurance**: Liability insurance also called as third-party insurance as, three parties are involved: the insured, the insurance company and the party who is injured or whose property is damaged by the insured.

E.g.: Auto Liability & Personal Liability come under Liability Insurance.

- **Life Insurance**: Life Insurance generally reduces the adverse consequences of premature death of a family member, by providing funds to replace the lost income and to pay expenses associated with final illness.

E.g.: Whole Life Insurance, Term Life Insurance & Universal life Insurance are the types of Life Insurances.

- Provides Insurance Cover against death (Early Death)
- Provides annuity benefits when grow older (Long Living)
- Provides range of products for risk coverage & saving

Health Insurance: Health Insurance is designed to protect individuals and families from financial losses caused by accidents and sickness.

- Health Insurance provides coverage against
- Hospital expenses
- Surgical expenses
- Physicians expenses
- Supplemental Coverage provides coverage against
- Dread disease

- Critical illness
- Long-term care
- Dental & vision care

Important Insurance Terminologies:

A. Insured

B. Insurer / Insurance Company

C. Liability

D. Insurable Interest

E. Insurable Risk

F. Agent

G. Application

H. Broker

I. Peril

J. Physical hazard

K. Moral Hazard

L. Policy

M. Cover Note

N. Coverage

O. Endorsement

P. Indemnity

Q. Exclusion

E.g.: Medical Insurance & Disability income insurance come under Health Insurance.

Government Insurance Programs:

- Some federal government insurance programs exist because huge amount of financial resources are needed to provide insurance to its citizens.
- Federal Government programs provides insurance for Catastrophic losses.

  E.g.: Social Security, National Flood Insurance Program.

Insurance Operations:

Main Operations of Insurance Companies are:

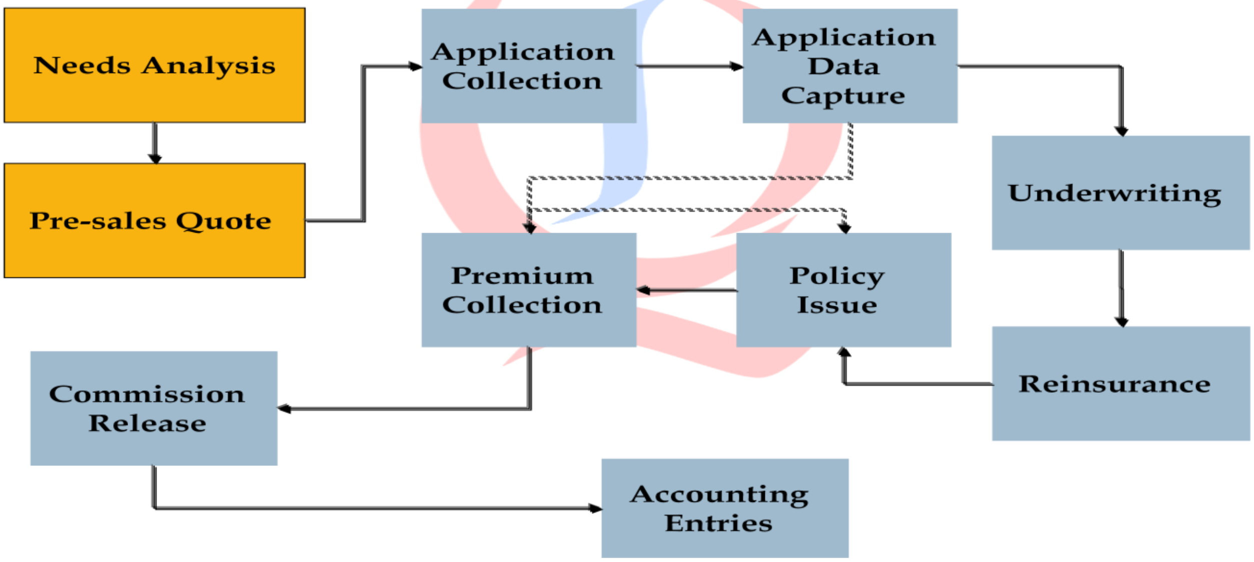
1. Marketing
2. Underwriting
3. Claim Handling
4. Ratemaking

- **Marketing**: Insurance Marketing is the process of identifying customers, selling and delivering a product or service. Other important aspect of marketing are advertising and marketing management.

Agents & Brokers:

- **Agents:** Legal representatives of the insurance company for which they have contractual agreements to sell insurance.
- **Brokers:** An independent business owner or firm that sells insurance by representing customers rather than insurer.
- The authority of the Agent and Brokers are generally stated in a written document called a **Agency agreement** or **Agency contract.**

Sell Business / Write New Business process flow



- **Underwriting**: Underwriting is the process by which insurance companies decide which potential customers to insure and what coverage to offer. Underwriters are insurance company employees responsible for selecting insured's, pricing coverage's, and determining policy terms and conditions.
- **Underwriting is a heart of a Insurance Business.** To a large extent a company's goals depends on the effectiveness of its underwriting.

Underwriting process involves:

- **Selecting Insured**: Selecting those applicants who meet the company's underwriting guidelines.
- **Pricing Coverage**: Pricing the coverage to charge the premium commensurate with the exposure.

Determining policy terms and conditions

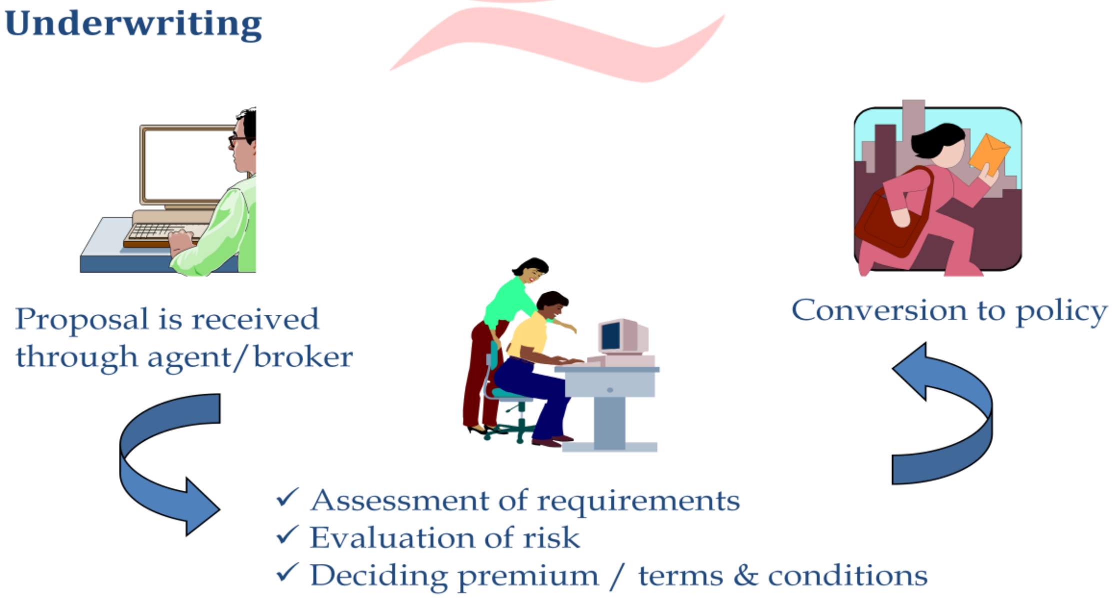
- **Monitoring underwriting decisions** to see whether they have desired effect or not.
- Underwriting management sets the company's guidelines in order to make optimal use of resources and avoid adverse selection.

Reinsurance:

- One of the main important aspect of the Underwriter is to arrange for Reinsurance.
- Types of Reinsurance are:

  a. **Treaty Reinsurance:** Is an arrangement whereby a reinsure agrees to reinsure automatically a portion of all eligible insurance of the primary insurer.

  b. **Facultative reinsurance:** Involves separate transaction for each reinsured policy. That is, the reinsure evaluates individually each policy it is asked to reinsure.


Underwriting
Proposal is received through agent/broker
✓ Assessment of requirements
✓ Evaluation of risk
✓ Deciding premium / terms & conditions
Conversion to policy

Claim handling

Claim handling enables insurance companies to determine whether a covered loss has occurred and, if so, the amount to be paid for loss. Claims are generally handled by Claim Representatives.

- **Claim**: Demand by a person or business seeking to recover from an insurance company for a loss that might be covered in the insurance policy.
- The employees of the insurance company who handle the claims are called as **Claim representative** or **Adjuster.**
- The responsibilities of a Claim representative are:

  a. Respond promptly to the submitted claim.
  b. Obtain adequate information
  c. Properly evaluate the claim
  d. To treat all parties fairly.

- The person who submits the claim to an insurance company is called a claimant.
- In liability insurance the claimant is the third party. In all other cases the claimant is the insured (primary or first – party).
- Independent Adjusters are independent claim representatives who offer claim handling services to insurance companies for a fee.

Claim Handling Process:

- The Claim Handling process generally involves three steps:

  a. Investigation
  b. Valuation
  c. Negotiation and Settlement

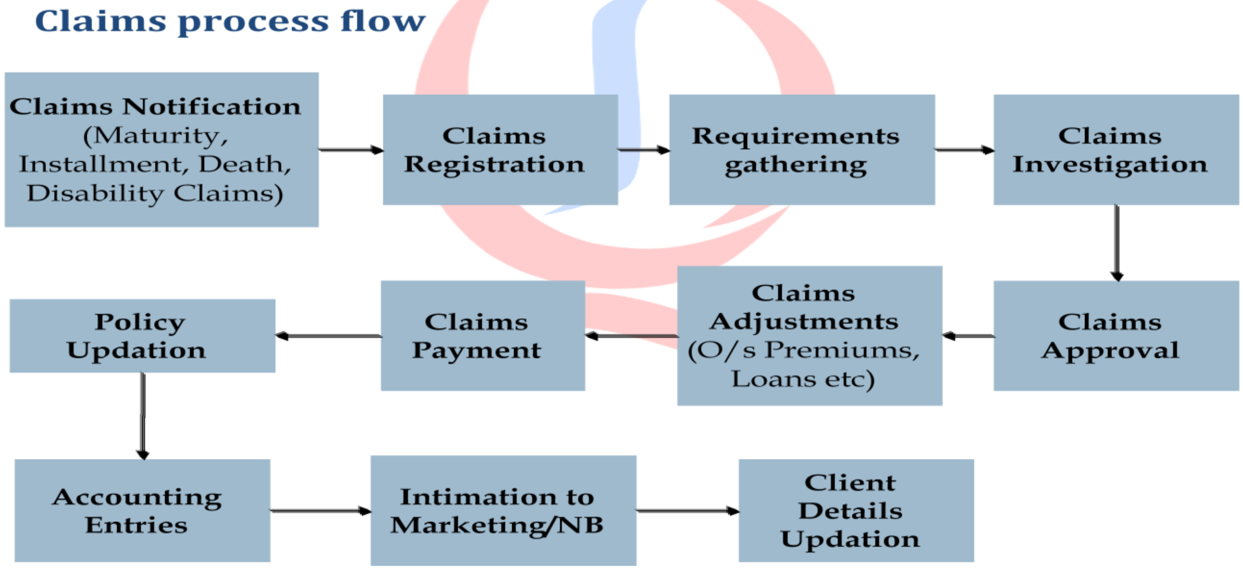
Valuation:

- Common Property Valuation Methods:
- **Actual Cash Value(ACV):** The cost to replace the property minus an allowance for the property's depreciation.
- **Cost to Replace**: Is calculated on the basis of like kind & Quality.

- **Depreciation:** Allowance for physical wear and tear.
- **Replacement Cost Analysis**: In this case, deduction for depreciation is not a part of the valuation.
- **Agreed Value**: Agreed value is a method of valuating property in which the insurer and the insured agree on the value of the property at the time the policy is written and that amount is stated in the policy declarations.

Negotiate and Settle:

- After the claim representative and the insured agree on the amount of the settlement, Other factor that can affect the insurers cost for property claims is: "Subrogation"
- **Subrogation**: Subrogation is the insurers right to recover payment from a negligent third party. When an insurer pays an insured for a loss, the insurer assumes the insured's right to collect damages from a third party responsible for loss.



Rate Making: Rate making is the process by which insurers determine the rates to charge the thousands of similar but independent insured's.These services are also called as Actuarial services.

Actuarial

Conduct research on

- trends in mortality
- policy lapse

- company expenses

Develop products based on

- market feedback
- financial feasibility
- Calculate Legal reserve, Liabilities & Premium rates
- Liaise with regulators, Prepare and submit all relevant reports to Regulators
- Review product /company performance
- Suggest course correction, if needed

Check Point….

✓ Identifying Customers …comes under

A. Marketing
B. Underwriting

✓ Pricing Coverage comes under

A. Underwriting
B. Rate Making

Solvency of Insurance Company:

- The ability to pay expenses and still make a reasonable profit is a measure of an insurance company's solvency, that is, its long – term financial strength.
- **Income**: Insurance companies receive income from two major sources. The first is the sale of insurance and the second is from investments it makes.
- **Written Premium:** Total Premium on all policies put into effect, or "written" during a given period. Even if the premium is not collect
- **Earned Premium**: Is the portion of the written premium that applies to the part of the policy period that has already occurred.
- **Unearned Premium**: The portion of the written premium that applies to the part of the policy period that has not yet occurred.

- **Investment Income**: As insurance company handles large amount of money , it invests available funds in the stock market or purchase bonds to generate additional income.
- Insurance companies select high-quality investments that are relatively secure and that can be readily converted to cash. E.g.: Stocks & Bonds.

Check Point….

✓ Sam paid a premium of $1200 on January 1st 2010 for a Annual policy.

Q1. As on Jan 1st 2010. How much is the Written Premium ???

Q2. As on May 1st 2010. How much is the Earned Premium??

Q3. As of May 1st 2010. How much is Unearned Premium.??

Q4. If Sam cancels the policy on September 1st 2010. How much is the Written Premium.??

EXPENSES:

- **EXPENSES**: The major expenses incurred by an insurance company are claim payments for insured's who have suffered losses and the costs associated with handling those claims.
- Other expenses are General expenses that relate to Marketing, day to day Operations, staffing, accounting and maintenance.
- For an insurer to be **Profitable**,

  Premium + Investment Income > Total loss payments and other expenses.

Profitability Ratio:

- Profitability ratios are generally used to analyze the financial performance of the insurance company.
- **Loss ratio** = Incurred loss expenses/Earned Premium
- **Expense ratio** = Incurred Underwriting expenses / Written Premium
  - Combined Ratio : Loss ratio + Expense Ratio

- Investment income ratio: Net investment Income/Earned Premium
- Overall Operating ratio: Combined ratio – Investment ratio

Overall Operating Ratio:

- An insurer with an overall operating ratio of 100% breaks even.
- If the Overall Operating ratio is greater than 100%, it indicates that an operating loss has occurred because expenses are greater than revenues.
- If the Overall Operating ratio is less than 100%, it indicates an overall operating gain because revenues are greater than expenses.
- Even monitoring financial results from past years helps to determine the accuracy of the insurance company's loss reserve estimates.

CORE BANKING

- A core banking system is the software used to support a bank's most common transactions.
- Core banking functions differ depending on the specific type of bank. Retail banking, for example, is geared towards individual customers; wholesale banking is business conducted between banks; and securities trading involves the buying and selling of stocks, shares and so on.
- Products that are designed to deal with multiple types of core banking functions are sometimes referred to as universal banking systems.

Core Banking Solution - is a simple solution that maintains -

1. Account - Balance in real time

2. Transaction History

3. Various parameters and rules for secured operating on these accounts and their parameters and rules.

4. Various Reports, Listings, sums on required groups for regulatory and informative purposes.

5. Interfaces for various internal, external systems for communication and exchange and invoking events and processes following a strict security policy.

Elements of core banking

- Making and servicing loans.
- Opening new accounts.
- Processing cash deposits and withdrawals.
- Processing payments and cheques.
- Calculating interest.
- Customer relationship management (CRM) activities.
- Managing customer accounts.
- Establishing criteria for minimum balances, interest rates, number of withdrawals allowed and so on.
- Establishing interest rates.
- Maintaining records for all the bank's transactions.

Features of a Core banking solution

- Increasingly accepted multi-tiered web paradigm
- Fully deployable in 365*24*7 mode not only across delivery channels but also for all the traditional branches
- Unified and integrated delivery channel strategy
- Time-to-market advantage through the extensibility tool-kit
- Seamless integration with various other business applications both online and in batch mode. Open to external Interfaces and systems and new delivery channels
- Well thought architecture and security framework
- The solution is a highly parameterizable solution and has been designed to provide parameters at different levels for the bank to add new products by changing parameters, adding new business rules, modifying them and extending the application.

Single Sign On

- Single sign on framework enables the application users to access multiple applications through a single login id and password. All the login related validations happen in SSO.
- Logging in with your user id created by Admin gives you access to Savings, Loans, CRM (used to maintain CIF details of users)

Creating USER IDS

- The sequence of processes in SSO are :
- Role Definition –level of access is defined here
- Password policy configuration
- User Creation
- Assigning of Role (admin, clerk, manager) as to the users
- SSO profile creation
- Assigning Access rights to SSO administration and applications
- Creating user profile for the application
- Password changes for a user
- User id management(Resetting login attempts, login attempts and password changes)
- Report generation(Audit, User and Role based reports)

Objective

- Types of Loan
- Phases of Loan
- Loan life cycle process
- Loans Terminology

Terminologies

Service Outlet/Branch. Any location from where the operations can be carried out or a logical location created for reporting purpose is called as Service Outlet

Work Class: This decides the powers vested for the user to have access to Menu, over riding of exceptions, Passing/Posting powers etc.

Temporary Overdraft (TOD): A Temporary Overdraft (TOD) is a limit/facility granted by the bank to its customers for a short period

- Clean Overdraft: An overdraft which is not backed by any security is called clean overdraft.
- Secured Overdraft: Secured Overdraft is generally back by securities
- Collaterals: This is the security given by the customer for the limits availed by the customer. The securities given by the customer has to be in the approved list of collaterals. Collaterals can be : Lands/Property/Jewellery/Gold/Animal Husbandry
- General ledger is a process of consolidation of the balances of the various accounts maintained in a bank. All the accounts maintained in the Bank/Branch are classified into various categories depending upon the nature, type and behaviour of the account. Such classified accounts are grouped/consolidated daily during the batch process to arrive at a position which reflects the total turnover/business of the Bank/Branch. There are no accounting entries at GL level

Parameter Set up

- One is at Bank Level which would be applicable for the entire bank and the other is at Branch
- At Bank level: There are certain parameters with regard to General Details, Term Deposits, Loans, Transaction A/c, Exchange Rate, Fees, Collateral Module, Connect 24, FAB, Trade Finance, and Exception Handling which can be set at the Bank Level

At Branch level MICR Centre, MICR Centre/Branch/Bank Code, Branch Open Date, Type of Cash allowed, License no, Tax Circle No etc are captured

Terminologies

- Office Accounts are accounts opened at the instance of the bank.
- When Interest is booked, interest goes to Interest Receivable Account, which is an office account.
- It is an account opened, without reference to a customer Id or involving a customer.
- Features like issuance of cheque book, charge calculation, minimum balance check, and interest calculations are not applicable to Office Accounts like Savings or loan accounts.
- All operations on the office account are initiated and done by the Bank.
- Office accounts are required by the banks to record transactions relating to Assets and Liability Accounts, Income and Expenditure Accounts, Inter Branch Accounts etc.

- Inventory: Inventory means stock of items that a bank holds. From the banking perspective, inventory can be classified into secured and non-secured items.

- Secured items are Demand Drafts, Chequebooks, Term deposit receipts, Travellers cheques & Gift cheques of different denominations etc, where tracking of each single unit of inventory is required.

- Non secured items are items like furniture, fixtures, stationary items etc

- Exceptions: Alerts can be generated either as a Warning, Exception or an Error. If a warning is generated it notifies the enterer, If an exception is generated it has to be authorized by a higher work class. If an error is encountered then it is against the bank policy and no user can override the same.

- Demand Drafts are payable by any other branch other than the issuing branch.

- Banker's Cheque is payable only by issuing branch.

- Demand Drafts/Banker's cheque is an important mode of remittance of money from one centre to another for the public in general. This is one of the key services offered by the Bank which generates non-fund income to the Bank.

Apart from issue and payment of Draft, other activities involved in this line of business are cancellation, re-validation, reversal, noting caution as a part of DD handling process

- **Savings Account:** This is a customer account wherein the customer maintains a credit balance. This is a liability type of account. The Product type used for such accounts is SBA. Customer is entitled to get interest on the credit balances maintained by him.

- **Current Account:** This is a customer account wherein the customer maintains either a credit or debit balance. If the customer maintains a credit balance this would be a liability account and if the customer maintains a debit balance this would be an asset account.

- **CIF ID:** Customer Information File. A customer must have a CIF for him to open an account with the bank.

- **Inactive Account:** When there are no customer induced transactions in the account for the specified period which is specified at the Product level, the status is changed to Inactive account.

Dormant Account: when there are no customer induced transactions in the account for a specified period the status is changed to Inactive account. After changing the status to inactive account if there are still no transactions for a further period which is specified at the Product level, then the status is changed to dormant account

- **Revolving OD:** Revolving overdraft is a facility offered to retail clients with credit card features such as billing date, minimum payment, and pay by date, penal interest and late fee for late payment.

- **Drawing Power:** The limit which the customer would be allowed to withdraw based on the sanctioned limit.

- **Sanctioned Limit:** This is the limit sanctioned by the bank to the customer based on the eligibility of the customer.

- **Sweeps:** It is possible that a customer can have more than one account of the same type or different types like Savings Bank, Current account, Overdraft facility, Term deposit accounts. There may be a situation that one of the SB accounts on which a cheque has been issued does not have enough funds for passing of the same but substantial amount is available in any of the other account. In such a situation the Bank may not return the cheque and would like to allow the debit to go through. In order to facilitate such a feature, SWEEPS helps the Bank.

- **Frozen Accounts:** The accounts may be restricted from either debit, credit or both operations due to various reasons. When any account has to be restricted from operations it can be frozen.

- **Lien:** Holding a part or full amount on the account so that the same is not available for the customer. There could be various reasons for which lien can be marked on the account.

- **Multi Currency accounts .** This is an umbrella account wherein the customer can have multiple savings/current accounts in different currencies account linked to a main account called the MultiCurrency Account.

The multicurrency account is represented by a consolidation currency and no financial transactions are performed on them. It is just used as an umbrella account to view the balances of all the accounts in a consolidated currency

- **Online Transaction:** A transaction which is put for an account with updating to account immediately. The user enters the transaction details.

- **Batch transaction:** The transaction is created by initiation of a process where in user intervention is not required.

- **Backdated transaction:** A transaction that is put for any account where the transaction date is a prior date.

- **Post dated transaction:** A transaction for an account where the transaction date is beyond the current date (system date/BOD date)

- **Value dated transaction:** A transaction with transaction date as BOD date. All accounting entries will be for BOD date, but the effective date of the transaction is not BOD/System date.

- **Posting of transaction:** The process of updating the balances of an account based on an entry is posting of transaction.

- **Proxy Posting:** When the user tries to post a transaction either online or through batch if the transaction

- Standing Instructions (SI) is a facility provided by the banks to its account holders who want to make payments or remittances of a recurring nature like payment of insurance premiums, subscriptions, transfer of funds, instalments to recurring deposits or loan accounts.

Instructions are created such that they run on a specific date to debit money from customer specified account of same bank

- **Proxy Posting:** When the user tries to post a transaction either online or through batch if the transaction is not going through for posting then the user can enable proxy posting which would post the transaction to a common proxy account defined. This would enable the data centre to smoothly run the End of Day process since the EOD will not go through if there are any pending transactions in the entered status.

- **Transaction Types:** Transactions have been classified into three types. They are Cash, Clearing and Transfer. The user cannot add the type of transactions. It is pre-defined.

- **Cash Transactions:** The user will be able to put through either a debit or credit transaction to the account depending upon whether he is doing a cash payment (cash withdrawal) or cash receipt (cash deposit) for the account.

- Clearing Transactions: Clearing transactions can be either inward or outward clearing.

- **Inward Clearing:** Inward clearing is a process whereby the Bank receives various types of negotiable instruments that are drawn on it by its customers and parts the resultant proceeds to the presenter of the instruments.

- **Outward Clearing:** In case of Outward clearing, the Bank gets the proceeds since it acts as a collecting agent on behalf of its customers who have tendered the instruments for credit of their accounts.

Transfer Transactions:

Transfer transaction is initiated when there is a need of transfer of funds from one account to the other. In Transfer transaction, the user has to enter both debit and credit transactions and also has to ensure that the set is balanced – The sum of both debit and credit transactions should match/agree

- **Transaction Sub Types:** Each of the Cash, Clearing and Transfer transaction has sub types.

- **For cash transactions:** The sub types supported are Normal Payment, Normal Receipt, Cross Currency Receipt, Cross Currency Payment, Cash transfer, ECS outward transaction, ECS inward transaction.

- **Normal Payment and Normal Receipt:** Normal Payment and Normal Receipts are cash transactions involving one currency.

- **Cross Currency Receipt and Cross Currency Payment:** Cross Currency Receipt and Cross Currency Payment transactions happens between two different currencies.

- **Cash Transfer:** Cash Transfer transactions are basically for internal use of the Bank and is put through for cash movement between the Cash account of the Bank and the Cash Account of individual teller accounts

- **For clearing transactions:** The sub types supported are Inward clearing, outward clearing.

- **For Transfer transactions:** The sub types supported are Inward clearing, Outward clearing, Bank induced, Customer induced, Interest Collection, Interest paid, Standing instruction, Bank induced standing instruction, Account revaluation, Service Charges, Back office transaction, ECS outward transaction, ECS inward transaction

- **ECS:** Electronic Clearing System. This is a system where in the clearing process happens without physical movement of instrument.

- **Transaction ID:** A financial transaction put through into the system will have a unique identification number. These are referred to as Tran id and Part Tran serial number in . A set or a bunch of transactions (credit and debit transactions) put through in one instance is referred to as a Tran id. A Tran id can have multiple credits and multiple debits where the sum of all debit and credit entries matches. Under a Tran id the individual credit or debit transactions are referred as a Part transaction and will have a part Tran serial number. A set or a Tran id will be treated as either posted/verified when all the transactions, either debit or credit are individually posted/verified to the concerned accounts. Posting of a transaction is not possible until the debit amount and credit amounts are matched (transaction is balanced).

- **Transaction Status:** There are three statuses which can be associated with a transaction. They are Entered, Posted and Verified.

- **Entered Status:** When a transaction is entered system generates a Tran ID and the transaction record is saved. But the account balance will not be updated till the transaction is posted.

- **Posted Status:** The process of updating the balances of an account based on an entry is posting of transaction.
- **Verified Status:** Verification process does not have any financial implication.
- **Deleted Status:** A transaction can be deleted prior to posting

LOANS

- Types of Loan
- Phases of Loan
- Loan life cycle process
- Loans Terminology

What is loan

- Bank lends Funds to Individuals/Industries from the deposits made by customers
- Bank earn profit by interest charged on loans
- Hence loans are assets to bank

Types of Loan

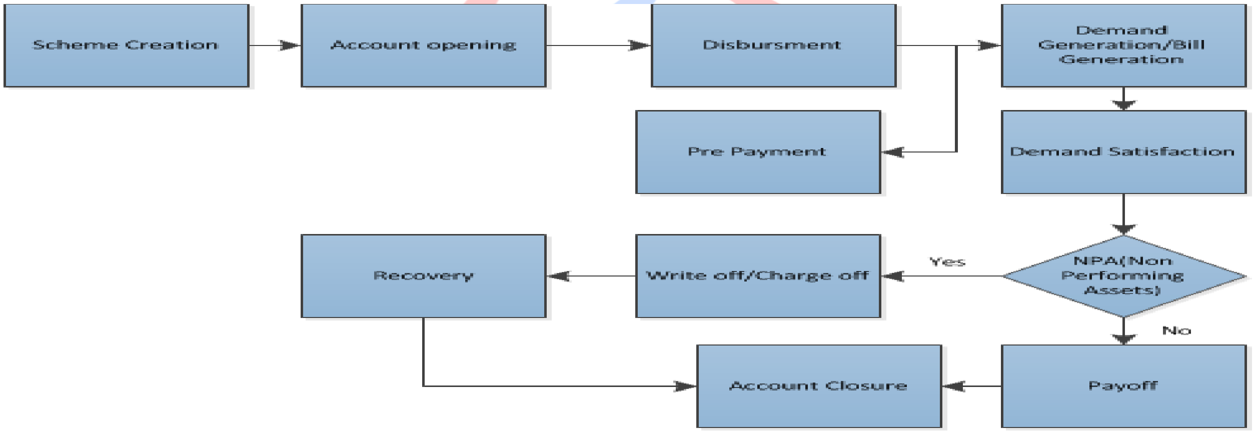
- Retail Loan(Product type LAA)
    - ✓ Home Loan
    - ✓ Vehicle Loan
    - ✓ Personal Loan
    - ✓ Student Loan
    - ✓ Salary Loan
    - ✓ Agriculture loan
    - ✓ Commercial Loan(Product type CLA)

Phases of Loan

- Pre Sanction
    - ✓ Application & relevant documents will be collected from the customer.
    - ✓ Credit appraisal of the customer
    - ✓ Credit report preparation and submission to the concerned authority

- ➢ Sanction
  - ✓ Loan is sanctioned to the customer
- ➢ Post Sanction
  - ✓ Execution of loan papers
  - ✓ Opening of loan account
  - ✓ Disbursement of loan
  - ✓ Follow up and Recovery of Loan
  - ✓ Asset Classification and Non Performing Assets
  - ✓ Writing off loan accounts
  - ✓ Handling of recoveries subsequent to writing off

Loan Life Cycle



Product creation

Main parameters at Product level are:

- ➢ Interest
- ➢ Currencies
- ➢ Fees
- ➢ GL code
- ➢ EI/Non EI

- Interest calculation method
- Repayment mechanism

Terminologies

- FLOWS
- Flow is a code.
- Every Transaction either a Debit or a credit to a Loan account is associated with a flow code/ID.
- Eg : Loan account is opened and disbursement has been made . Say transaction ID is TXN1223. TXN1223, would be mapped to DISB Flow code created and stored in table backend which records all the activities on loan account. Similarly, EIDEM, for demand generation, INDEM, when interest is collected. Amount collected would be stored against flow id in table.

Based on the flow, the system has to do some internal storing of information, processes, validations etc.

- This is required to arrive at the Demand, Collection and Balance position of an account thus reflecting the overdue position of an account. This is also useful for classification of assets into performing and non-performing assets.

Flows…..

- The flows are categorised into following categories.
- Disbursement
- Collection
- Demand
- Equated instalments
- Transfer

Flows –Disbursement

- DISBURSEMENT:
- Represents loan amount released to the account holder based on the sanction limit. Normally disbursement flows will happen while opening.
- Single or multiple disbursements in a loan account. In case of multiple flows disbursement will happen subsequent to account opening.

- Disbursement flow is associated with debit transactions except in case of reversals of disbursement.

FLOW – DEMAND/BILL

- DEMAND (also known as bill):
- Demand is made on the customer for repayment of the loan amount due.
- The demand can be Principal demand, Interest demand, Bank charges demand and Other charges demand.
- Demands are raised periodically during the life cycle of the loan account.
- As and when an installment becomes due, system automatically raises a demand for the principal amount.

Flows –Collection

- This is a transaction for recovery of amount to a loan account.
- Collection flow‟s can happen during the lifecycle of the loan account, when the customer makes a repayment.
- For collection flows, it is possible to specify the offset sequence (the sequence in which the demands raised are to be offset or adjusted (such as principal first, interest next, charges etc.)).
- Collection is associated with credit transactions to loan a/c.

Flows –Transfer

- **TRANSFER :**
- A loan account can also go into credit balance in some cases.
- When the loan account goes to credit balance, it needs to be debited to transfer the credit balance.
- This debit cannot be treated as a disbursement, Interest or any other Charge. In such cases, the transaction will be Classified/Identified as "Transfer flow". "Transfer flow" is possible only if the liability of the account is nil and the interest calculation is up-to-date
- Late fee

When a loan is sanctioned, it is associated with a repayment. If the repayment is on equated installments and interest calculation is based on schedule balance method, then there may be chances that the Bank will not go in for charging penal interest on the balance outstanding.

Instead they may choose to collect penalty on the defaulted amount (monthly installment amount due). This penalty is known as late fee collection.

Account Opening

- EQUATED INSTALMENT
- The repayment to a loan account could be done in equated installments. The equated installments can be monthly, quarterly or any other specified period. If the installments are of equated in nature, then the amount repaid will be adjusted towards both principal and interest.
- Interest is compounded with PMT formula, Rule of 78.
- While account opening, CIF created for customer, Product code for Retail Loans (eg Personal loan) are entered. Account opening date can be backdated or same as today. Loan Amount, Loan Repayment date every month (due date day), loan tenor has to be specified.

Disbursement

- Disbursements can be done in any of the following modes,
- - Can be one time
- - Can be multiple times
- - Can be based on disbursement schedule
- - Can be through ACH/SWIFT/ECS
- - Can be through cash
- - Can be through transfer
- - Can be through DD/Pay order
- - Recovery of charges as a part of disbursement
- Accounting entry
- Dr. Loan A/c
- Cr. SBA A/c/ DDA A/c /ACH A/c (depending on the disbursement type)

Interest booking

Interest accrued/accumulated/recognized from previous payment date to current due date but not yet paid

Demand Generation

- Both principal and interest demand can be generated
- For billing type of accounts, instead of demand a bill is generated
- Bill shows the EMI to be paid by the customer, the previous emi paid if any, Late fee if any set up and applicable along with the date and month for which bill has been raised.
- For demand model the transaction is
    - Dr. Loan Int Acct

Cr. Int receivable

Repayment Schedule/Amortization schedule

- The important components of the repayment schedule are
- Instalment flow IDs (EIDEM)
- Instalment start dates (both Principal and Interest)
- Instalment Frequency (Monthly generally, Quarterly)
- Number of instalments
- Instalment amount

Demand Satisfaction

- Customer can repay the loan in multiple methods
- ECS
- Cheque
- Cash Payment
- Debit from operative account
- HLASPAY is the online menu for demand satisfaction

Asset Classification

- Bank classifies the loan accounts based on their past due period. This is called asset classification.
- Based on days past due(DPD), accounts are classified into different delinquency cycle.
- DPD will be calculated from the last pending installment date.

Delinquency –DPD

- This is with reference to the overdue.
- There will be cycles of user-defined days. While moving a demand to a given cycle the date of demand raised and the date of repayment will be skipped.
- There is a maximum of fifteen cycles that user can define. Let us take a example for ten cycles. We take that each cycle period is of 15 days. Then the set up would be as under:
- Cycle 1 15 days
- Cycle 2 30 days
- Cycle 3 45 days
- Cycle 4 60 days
- Cycle 5 75 days
- Cycle 6 90 days
- Cycle 7 105 days
- Cycle 8 120 days
- Cycle 9 135 days
- Cycle 10 150 days

Charge off and recovery

- Some loans are not repaid, in spite of banks best efforts . A process called write off or Charge off is applied on such loans by the bank.
- Charge off is done only for accounts which are marked as Past due.
- Demand generation does not happen for an account that is charged off
- When account is charged off it is a loss to bank.
- Recovery after charge off is done when some amount is recovered from an already charged off account.

Payoff

- Payoff means collecting entire dues in the account to ensure that the balance becomes zero.

- Collection of entire dues in the account can be on maturity of the loan, subsequent to maturity or earlier to maturity.

Account Closure

- Once the balance in the account is zero the account can be closed.

Restructuring

- The restructuring or rephasement is the revision of the repayment schedule which may be necessitated because of changes in interest rates, irregular payment of installments and so on.

Deferment/Forbearance

- Applicable for student loan
- Deferment is a period during which the repayment of the principal and interest of your loan is temporarily delayed.
- The account has to be in repayment period to mark the account for deferment.
- For deferment period interest is not applicable. But for forbearance period interest will be applicable.

Exceptions

- Change in account name
- Value dated transaction - When transaction date differs from value date of the transaction this exception is raised.
- Back dated transaction - When transaction date is lesser than the BOD Date this exception is raised.
- Cash transaction
- Transfer transaction
- Clearing transaction
- Referred account closure
- Account in credit balance

Grace Period

- Grace Period is the holiday period for a loan after the loan disbursement.
- The repayment starts only after the grace period.

- Interest may or may not be applicable during grace period.

Interest Capitalization

- The interest during grace period can be added up to the principal.
- This process is called interest capitalization.

SESSION - 07

ORACLE:

What is DB Testing

Computer applications (front ends) are more complex these days. The more complex the front ends, the back ends are even more complicated. So, it is all the more important to learn about DB testing and be able to validate the databases effectively to ensure **'secure and quality database'**.

- ✓ Backend testing mainly includes testing the integration between the application and the database.
- ✓ It is more like checking whether the changes made in the database gets reflected in the front end application.

For example: consider a new column is been added in the table. We can test this by providing values in the front end application and check whether they are stored in the table (backend database).

- ✓ It's basically testing data while travelling from **front to back end** or **back end to front end** or **back end to back end** only.

  Type 1 DB Testing: Application to DB

  Type 2 DB Testing: DB to DB

  Type 3 Testing: DB to Application

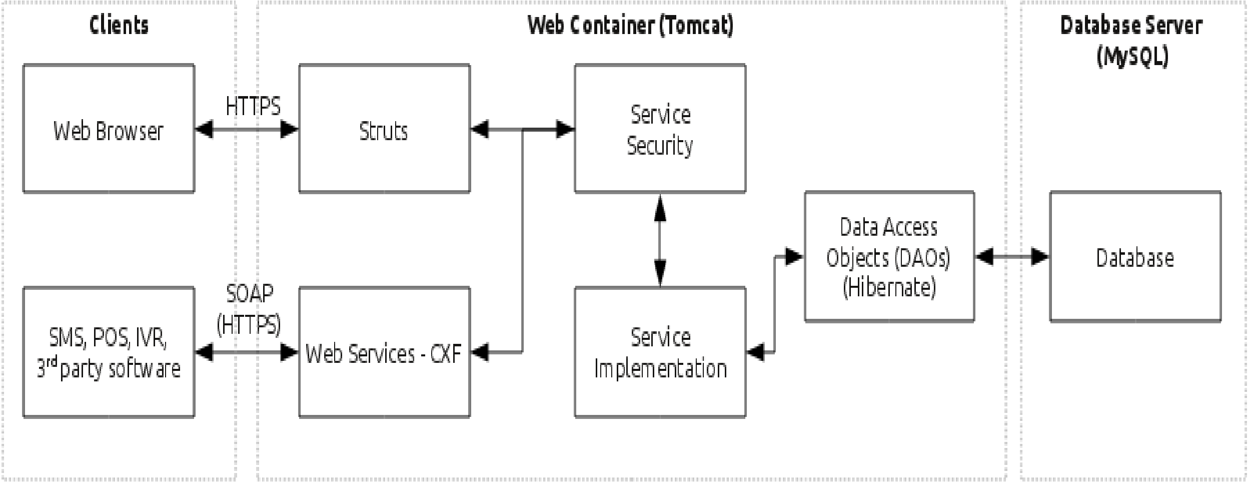
- ✓ It also involves testing the application from the logical storage of the data. Validating the storage data with the UI.
- ✓ Front end testing mainly focuses on testing the application from the user perspective, it consists of functionality, usability, and GUI .It is very likely that many tests in a front end only hit a small portion of a backend.
- ✓ A backend is the engine of any client/server system. A bug in a backend may raise a serious impact on the entire system. This includes deadlock or data corruption or data loss and bad performance. Too many bugs in a backend will cost tremendous resources to find and fix bugs and delay the system developments.
- ✓ Many bugs can be effectively found and fixed in the early development stage

Database schema: A database schema is a way to logically group objects such as tables, views, stored procedures etc. schema as a container of objects.

Explain Importance of Data Layer Testing in Web Application Architecture?

Web Application Architecture/3 Tier Architecture:

- ✓ Typically comprise a presentation layer, a business or data access layer, and a data layer. Three layers in the three tier architecture are as follows.
  1. Presentation / Client Layer
  2. Business Logic Layer
  3. Data Layer
- ✓ **UI/Presentation layer** :
  - ➢ Represents UI part of Application.
  - ➢ Where the data is presented to the user or input is taken from user.
  - ➢ Ex: Registration form, labels, Buttons etc.
- ✓ **Business Logic layer:**
  - Validation of Data
  - Logic calculations and implementation
  - Acts as interface b/w Presentation and Data layer
- ✓ **Data layer:**
  - Database connection
  - Store and retrieve data.



How to understand data before performing database Testing?

- Applications are used by end users and they enter a group of raw data
- This data is later collated (collect and combine) and used by management to arrive at meaningful information
- Before we first understand the technical aspects of database, we must understand the business data clearly
- Rule 1: In any application, first identify raw data
- Rule 2: Group related data and associate data type and size (summary and detail)

- Rule 3: Create a set of samples for each of these groups for better clarity
- Rule 4: Identify the relationship between the data

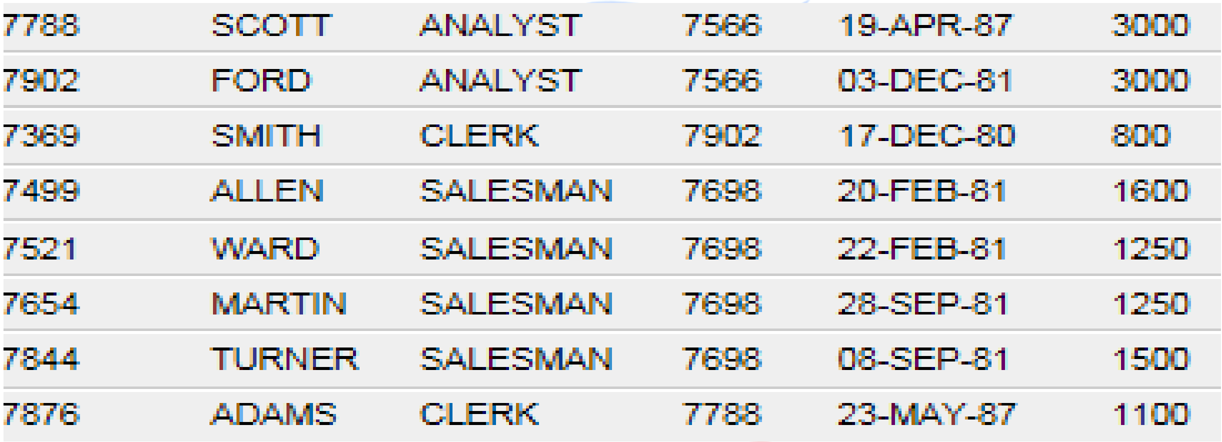
What is Data Actually?

Data: It is Stored Representation of OBECTS and EVENTS That Have Meaning and importance in the, User's Environment.
Data can be Structure OR Unstructured.

Structured -> Student Name, address, Phone number, mail id

Unstructured -> Stud Photo, address map, log files.

Representation of Data:



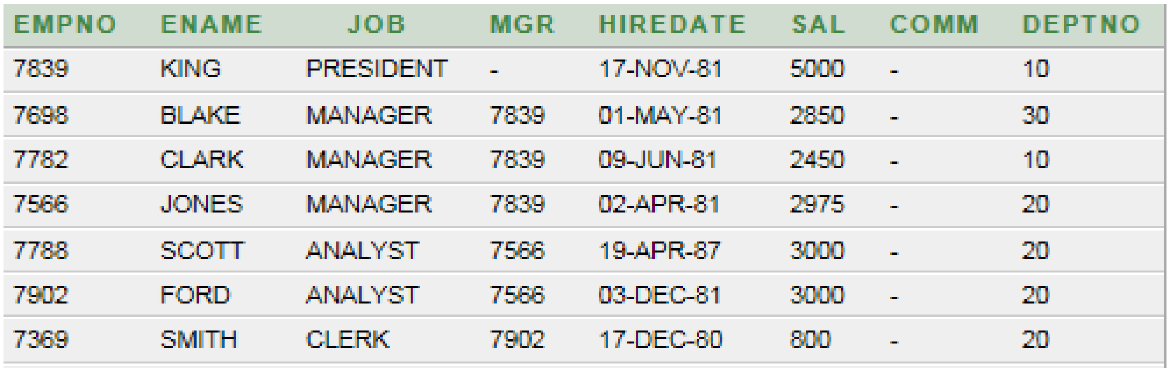
7788	SCOTT	ANALYST	7566	19-APR-87	3000
7902	FORD	ANALYST	7566	03-DEC-81	3000
7369	SMITH	CLERK	7902	17-DEC-80	800
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600
7521	WARD	SALESMAN	7698	22-FEB-81	1250
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250
7844	TURNER	SALESMAN	7698	08-SEP-81	1500
7876	ADAMS	CLERK	7788	23-MAY-87	1100

What is Information actually?
Information:

It is Data that is in Processed Form, Such That It Increases the Knowledge of the Person Who Uses the Data.

Representation of information:

Employee information:



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	-	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20

What is Metadata Actually?

Metadata: It is the Data Which Describes the Properties OR Characteristics of end Users Data and the Context of the Data.

Metadata Properties Can Include Information Such as:
Data Name, Definitions, Length OR Size, Values Allowed, Source of Data, Ownership.

Database Management Systems:

Database Management Systems is Software that is used to Create, Maintain, and Provide Controlled Access to User Databases. Database Management Systems Should Provide Systematic Method of

- ✓ Creating the Database
- ✓ Updating the Database.
- ✓ Storing the Database.
- ✓ Retrieving of Data from Database.

How to Communicate With RDBMS (Relational Database Management Systems)?

The Structured Query Language (SQL) is used to communicate with RDBMS.

Relational Database Terminology

Row or Tuple:

It Represents All Data Required for a Particular Instance in an Entity. Each Row in an Entity is uniquely identified by declaring it As PRIMARY KEY OR UNIQUE. The Order of the Rows is Not Significant, While Retrieving the Data.

Column OR Attribute:

It Represents One Kind of Data in a Table Vertically Collected. Structured Query Language (SQL) Statements in Oracle

The Different Categories Into Which the SQL Statements Fall Are As Follows.

Data Retrieval Statement (DRL) SELECT Statement

Data Manipulation Language Statements (DML) INSERT Statement, UPDATE, Statement, and DELETE Statement.
Data Definition Language Statements (DDL) CREATE Statement, ALTER Statement, DROP Statement, RENAME Statement and TRUNCATE statement.

Transaction Control Language Statements (TCL) COMMIT Statement, ROLLBACK Statement and SAVEPOINT Statement.

Data Control Language Statements (DCL) GRANT Statement, REVOKE Statement

Creating and Managing Tables.

Table ->Used to Store Data.

Rules for Create A Table:

The User Should Have Permission on CREATE TABLE command, and Storage Area Should be Allocated.

The Table Name Should Begin with a Letter and can be 1 -30 Characters Long. Table Names can Contain Combination of A -Z (OR) a -z (OR) 0 -9 (OR) -, $, #. Names cannot be duplicated for Another Object Name in the Same ORACLE Server. Names cannot be Oracle Servers Reserved Words. Table Names Are Not Case Sensitive in Oracle. Table is a Collection of Attribute names (Column Names) and Oracle Data Types and Required Width along With Required Constraints.

Create Table Statement

Syntax

```
SQL> CREATE TABLE <table_Name>

(

Column Name1 <DataType>(Width),

Column Name2 <DataType>(Width),

Column NameN <DataType>(Width)

)
```

Note

All Data Types May Not Have The Width Property. No Two Columns in The Same Table Can Have the Same Name.

Building Blocks of SQL Statements

Data Types, Literals and NULLS.

Data Types in Oracle: Each Value in ORACLE is manipulated by a Data Type.

The Data Type Associates a Fixed Set of Properties With that Value Stored. The Values of One Data Type are Different from another Data Type.

NUMBER data type:
It Stores zero, positive, and negative fixed and floating-point numbers

The general declaration is: NUMBER (P,S)

P: it specifies the precision, i.e the total no of digits (1 to 38).

S:it specifies the scale, the number of digits to the right of decimal point, can range from -84 to 127

VARCHAR2 data type:

- ✓ Specifies the Variable length Character string.
- ✓ Minimum size is 1 byte maximum size is 4000 bytes.
- ✓ It occupies only that space for which the data is supplied

CHAR data type

It specifies fixed length Character string.

The size should be specified.

If the data is less than the original specified size, blank pads are applied.The default length is 1 byte and maximum is 2000 bytes

DATE data type:
It is used to store date and time information. The information revealed by data is :

Century, year, month, date, hour, minute, second.

Default Date format in oracle is **DD-MON-YY.** The date Range provided by oracle is January 1, 4712 BC to December 31, 9999 AD.

Illustrative Example To Create A Table:

CREATE TABLE Students

(
StudIO NUMBER(6), Fname VARCHAR2(30), Lname VARCHAR2(30), DOB DATE, DOJ DATE,

Fees NUMBER(7,2), Gender VARCHAR2(1)
);

Populating the Data into Tables:
INSERT Statement.
Inserting Data into All Columns of a Table

INSERT INTO Students VALUES (1234, 'SAMPATH', 'KUMAR', '29-JAN-80', '30-MAR-95', 25000, 'M');

In This Case the Values Should be provided to All the Columns That Exist Inside the Table.

The Order of Values Declared in the VALUES Clause Should Follow the Original Order of the Columns in Table.

The CHAR, VARCHAR and DATE Type Data should be declared in **Single Quotes**.

Numerical Information can be applied normally.

Inserting Data into Required Columns

INSERT INTO Students (Studno, Fname, Lname, DOJ, Gender) VALUES (1235, 'Raj', 'Ramana', '20-Feb-85', 'M');

In This Case the Order of Columns Declared in INSERT Need Not Be the Same As That of the Original Table Order.
The Data Values in the VALUES Clause Should Match With that of INSERT List.
The Column(s) Not Supplied with Data is Filled With NULL Values, Until the NOT NULL Constraint is declared.
Inserting Special Values into Table
SYSDATE Function:
It is a PSEUDO COLUMN Provided by The Oracle.
The Function Returns the CURRENT DATE and TIME from the System Clock.

USER Function:
It is Special Function, Which Records the Current USER Name.

INSERT INTO Students (StudNo, Fname, DOJ, Fees, Gender, Insel1By) VALUES (1234, 'Pavan', SYSDATE, 25000, 'M', USER);
Data Retrieval Standards Using SELECT Statement:

Querying the Data From Tables

It is an Operation That Retrieves Data From One OR More TABLES OR VIEWS.

SELECT Statement
The SELECT Statement is used to Retrieve Data From One OR More TABLES.
Prerequisites:

The User Must Have the SELECT Privileges on the Specified Object.

Capabilities of SQL SELECT Statement:

The SELECT Statement Can be Used to Select OR Retrieve Data From the Object Using Any One of the Following Criteria.

SELECTION, PROJECTION, JOIN.

SELECTION
It Chooses the Rows in a Table that are Expected to be returned by a Query.
PROJECTIQN
It Chooses the Columns in a Table that are Expected to Return by a Query.
JOIN
It Chooses the Data in From One OR More Number of Tables by Creating a Link Between Them.
Basic SELECT Syntax

SELECT [DISTINCT] [*] {Column1 [Alias] , ... } FROM Table_Name;

SELECT Key word Identifies Columns.

FROM Clause Identifies Tables.

SELECT -> Specifies a List of Columns.

DISTINCT -> Suppresses Duplicates.

* -> Projection Operator to Select All Columns from the Table.

COLUMN -> Selects the Named Column.

Alias -> Gives Selected Columns Alternate Column Name.

FROM Table_Name -> Specifies the Table Containing the Columns.

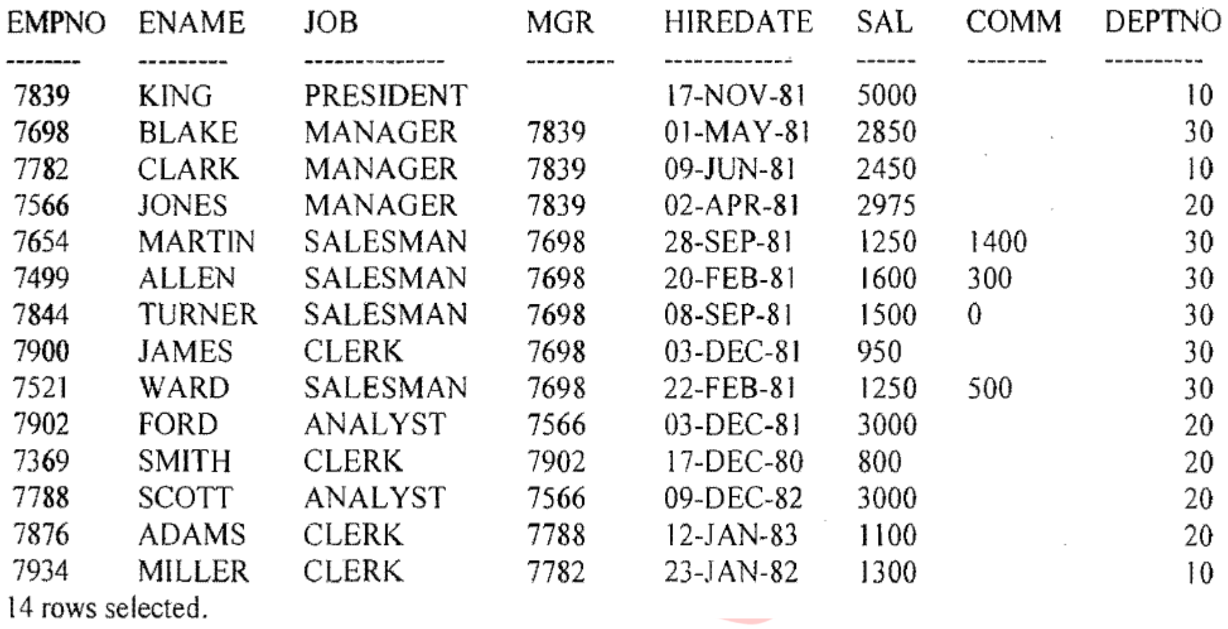
Retrieving Data from All Columns of a Table

FOR This Purpose the Projection Operator '*' is Used.

The Operator Projects Data from All The Columns Existing in The Table With All Records.

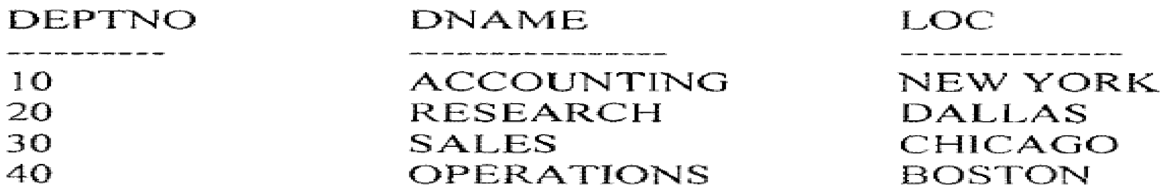
The Data is displayed in a Table Format.

SELECT * FROM Emp;



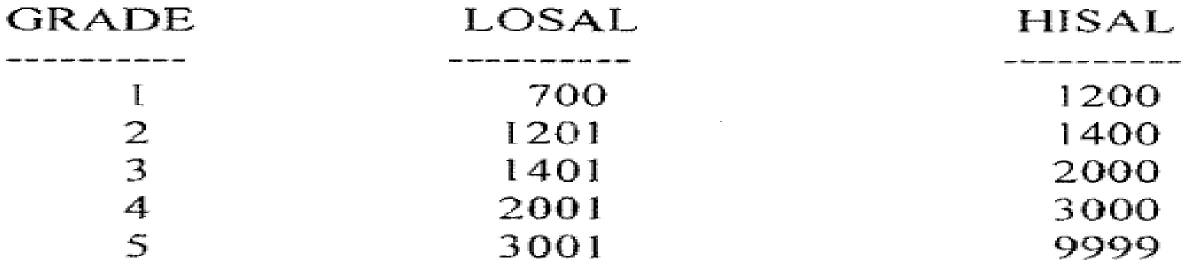
```
EMPNO  ENAME   JOB        MGR   HIREDATE   SAL   COMM  DEPTNO
-----  ------  ---------  ----  ---------  ----  ----  ------
 7839  KING    PRESIDENT        17-NOV-81  5000            10
 7698  BLAKE   MANAGER    7839  01-MAY-81  2850            30
 7782  CLARK   MANAGER    7839  09-JUN-81  2450            10
 7566  JONES   MANAGER    7839  02-APR-81  2975            20
 7654  MARTIN  SALESMAN   7698  28-SEP-81  1250  1400      30
 7499  ALLEN   SALESMAN   7698  20-FEB-81  1600  300       30
 7844  TURNER  SALESMAN   7698  08-SEP-81  1500  0         30
 7900  JAMES   CLERK      7698  03-DEC-81  950             30
 7521  WARD    SALESMAN   7698  22-FEB-81  1250  500       30
 7902  FORD    ANALYST    7566  03-DEC-81  3000            20
 7369  SMITH   CLERK      7902  17-DEC-80  800             20
 7788  SCOTT   ANALYST    7566  09-DEC-82  3000            20
 7876  ADAMS   CLERK      7788  12-JAN-83  1100            20
 7934  MILLER  CLERK      7782  23-JAN-82  1300            10
14 rows selected.
```

SQL> SELECT * FROM Dept;



```
DEPTNO      DNAME          LOC
------      ----------     ---------
10          ACCOUNTING     NEW YORK
20          RESEARCH       DALLAS
30          SALES          CHICAGO
40          OPERATIONS     BOSTON
```

SQL> SELECT *FROM SalGrade;



```
GRADE          LOSAL          HISAL
----------     ----------     ----------
    1              700           1200
    2             1201           1400
    3             1401           2000
    4             2001           3000
    5             3001           9999
```

Retrieving Data From Specific Columns

SQL> SELECT Empno, Ename, Sal FROM EmP;

SQL> SELECT Ename, Job, Sal, Deptno FORM Emp;

SQL> SELECT Deptno, Dname, Loc FROM Dept;

SQL> SELECT HiSal, LoSal, Grade FROM SalGrade;

SQL> SELECT Empno, Ename, Sal, HireDate FROM Emp;

The Column Names Need Not Be in The Same Order as Table. The Columns Should be Separated. Using Comma. The Column Names Can be Separated Onto Different Lines Within the SQL BUFFER.

The Casing of Column Names is Not Important

We can change the data type if the column contains NULL's

Drop:

It removes the definition of the oracle table (including data) and associated INDEXES.

Syntax: drop table <table name>

Any views and synonyms will remain but are kept in invalid state.

Drop table statement once executed is irreversible.

Changing the name of an object:

The rename command can be used to change the name of a Table, View

Syntax:Rename <oldname> to <newname>

Truncation a table:

It is used to remove all rows from a table and to release the storage space Used by the specific table.

Syntax:

Truncate table <table_name>

UPDATE Command:

The UPDATE statement is used to change the existing values in a table or in the Base table of view.

Syntax:

UPDATE <table_name> SET <specification>WHERE clause

DELETE Statement:

It is used to remove rows from table.

Syntax: DELETE [from] <table_name> [where condition];

Applying Arithmetical Operations

Arithmetic Expressions Can be Implemented Through SELECT Statement to perform calculations.

Arithmetic Operators

The Arithmetic Operators Can be Used to Create Expressions on NUMBER and DATE Data Type Columns.

The Arithmetic Operators Supported Are

Addition:+

Subtraction →-

Multiply:*

Divide:/

The Arithmetic Operators Can be Used in Any Clause of a SQL Statement, Except the FROM Clause.

```
SELECT Empno, Ename, Sal, Sal + 500 FROM Emp;

SELECT Empno, Ename, Sal, Sal-1000 FROM Emp;
```

Operator Precedence

Multiplication and Division Take Priority over Addition and Subtraction (*, /, +, -).

Operators of the Same Priority are Evaluated from Left to Right.

To Prioritize Evaluation and to Increase Clarity Parenthesis Can be implemented.

```
SELECT Empno, Ename, Sal, (12 *Sal) + 100 FROM Emp;

SELECT Empno, Ename, Sal, 12 * (Sal + 500) FROM Emp;
```

Handling NULL values:

NULL: It is a Value which is Unavailable, Unassigned, Unknown and Inapplicable. A NULL is Not Same as Zero OR Blank Space. If a Row Lacks the Data for a Particular Column, Than That Value is Said to Be NULL OR to Containing NULL.

SELECT Ename, Job, Sal, Comm FROM Emp;
If Any Column Value in an Arithmetic Expression is NULL, The Overall Result is Also NULL. The Above Situation is named as NULL PROPAGATION and Has to be Handled Very Carefully.

```
SELECT Ename, Job, Sal, Comm, Sal + Comm FROM Emp;
SELECT Ename, Job, Sal, Comm, 12 *(Sal + Comm) FROM Emp;
```

NVL () Function:

The NVL Function is Used to Convert a NULL Value to an Actual Value.

Syntax NVL(Exprl, Expr2).

Expr1: It is the Source Value OR Expression That May Contain NULL.

Expr2: It is the Target Value for converting NULL.

NVL Function Can be Used to Convert Any Data Type, The Return Value is Always The Same as the Data Type of Expr1.

The Data Types of The Source And Destination Must Match.

NVL(Comm, 0)

NVL(Hiredate, '01-JUN-99')

NVL(Job, 'Not Assigned')

SELECT Ename, Sal, Comm, Sal + NVL(Comm, 0) FROM Emp;

SELECT Ename, Sal, Comm, (Sal * 12) + NVL(Comm, 0)FROM Emp;

SELECT Ename, Sal, Comm, (Sal + 500) + NVL (Comm, 0) FROM Emp;

Working with Aliases:

An ALIAS is an Alternate Name Given for any ORACLE OBJECT.

Aliases in Oracle are of two types: Column Alias

Column Alias Renames a Column Heading in a Query. The Column Alias is specified in The SELECT List by Declaring the Alias after the Column Name by Using the Space Separator. ALIAS Heading Appears in UPPER Casing by Default. The Alias Should be Declared in Double Quotes if it is Against the Specifications of Naming Conversions of Oracle. The AS Keyword Can be Used Between The Column Name and Alias. An Alias Effectively Renames the SELECT List Item for the Duration of that Query only.

An Alias Cannot be Used, Any Where in THE SELECT List for Operational Purpose.

Table Alias

Table Alias Renames the Original Name of the Table in a SQL Statement. Table Aliases are Very Important When Working with Self Joins. The Table Alias is applied for the Current SQL Statement Only.

SELECT Empno Numbers, Ename Name, Sal "Basic Salary", Job Designation FROM Emp;

SELECT Deptno AS "Department no", Dname AS "Department Name", Loc AS Place FROM Dept;

SELECT Hisal As "Maximum Range", Losal As "Minimum Range", Grade FROM Salgrade ;

Literals in ORACLE:

A LITERAL and a CONSTANT Value are Synonyms to One Another and Refer to a Fixed Data Value.

The Types of LITERALS Recognized by ORACLE are TEXT Literals, INTEGER Literals, NUMBER Literals & INTERVAL Literals.

It Specifies a TEXT OR CHARACTER literal. TEXT Literal Should be Enclosed in Single Quotes. They have Properties of Both CHAR and VARCHAR2 Data Types. A TEXT Literal Can Have a Maximum Length of 4000 Bytes. A Literal that is Declared in a SELECT List Can be a CHARACTER, a NUMBER, OR a DATE. A Literal is Not a Column Name OR a Column Alias.

A LITERAL is Printed for Each Row, That is Retrieved by the SELECT Statement. DATE and CHARACTER Literals Must be Enclosed Within the Single Quotation Marks. LITERALS Increase the Readability of The Output.

```
SELECT Ename||' : '||' Month Salary = '|| Sal AS Salaries FROM Emp;
SELECT 'The Designation of '|| Ename||' is '||Job As Designation FROM Emp;
SELECT 'The Annual Salary of' ||Ename||' is '||Sal * 12 AS Annual_Salary FROM Emp;
SELECT Dname||' Department is Located At '||Loc Locations FROM Dept;
SELECT Ename|| ' Joined The Organization on '||Hiredate FROM Emp;
SELECT Ename||' Works in Department Number '||Deptno ||' as '||job FROM Emp;
```

Applying Concatenation Operator:

The Concatenation Operator Links Columns to Other Columns, Arithmetic Expressions, or Constant Values. Columns on Either Side of the Operator are Combined to Make a Single Output Column. The Resultant Column is treated as a CHARACTER EXPRESSION. The Concatenation Operator is represented in ORACLE by Double Pipe Symbol (||).

```
SELECT Empno||' '||Ename||', Designation is ' ||Job "Employees Information" FROM Emp;
```

Suppressing Duplicate Rows in Output

Until it is Instructed SQL*Plus Displays the Results of a Query Without Eliminating Duplicate Rows. To Eliminate the Duplicate Rows in the Result, the DISTINCT Keyword is used. Multiple Columns can be declared after the DISTINCT Qualifier. The DISTINCT Qualifier Affects all the Selected Columns, and Represents a DISTINCT Combination of the Columns.

```
SQL> SELECT DISTINCT Deptno FROM Emp;

SQL> SELECT DISTINCt Deptno FROM Emp;

SQL> SELECT DISTINCT MGR FROM Emp;

SQL> SELECT DISTINCT Job, Deptno FROM Emp;

SQL> SELECT DISTINCT Deptno, Job FROM Emp;
```

Filtering of Records

The Number of Rows Returned by a QUERY can be Limited Using the WHERE Clause.

A WHERE Clause Contains a Condition That Must be Met and Should Directly Follow the FROM Clause.

Syntax

SQL> SELECT [DISTINCT] [*] {ColumnI [Alias], ... }

FROM Table Name [WHERE Condition(s)];

The WHERE Clause Can Compare Values in Columns, Literal Values

- ✓ Arithmetic Expressions
- ✓ Functions

The Components of WHERE Clause are Column Name, Comparison Operator.

Column Name (OR) Constant (OR) List of Values.

The CHARACTER Strings and DATES should be enclosed in Single Quotation Marks.

CHARACTER Values are Case Sensitive and DATE Values are Format Sensitive (DD-MON-YY)

The Comparison Operators are used in Such Conditions That Compare One Expression to Another.

The Different Comparison Operators are

Equality Operator:=

Not Equality Operator: <> or != or ^=

- ✓ Greater Than Operator : >
- ✓ Less Than Operator :<
- ✓ Greater Than or Equal to Operator : >=
- ✓ Less Than or Equal to Operator : <=

The Format of The WHERE Clause is

WHERE Expr OPERATOR VALUE.

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job = 'MANAGER';

SQL> SELECT Ename, Hiredate, Deptno, Sal FROM Emp WHERE Deptno = 10;

SQL> SELECT Empno, Ename, Sal FROM Emp WHERE Sal >= 3000;

SQL> SELECT Ename||' Joined on '||Hiredate "Employees Joining Dates" FROM Emp

```
WHERE Hiredate =  'OJ-JAN-95';

SQL> SELECT Ename||' Works in Department'||Deptno "Employees and Deaprtments"

FROM Emp WHERE Deptno <> 20;

SQL> SELECT Ename, Sal, Deptno, Job FROM Emp WHERE Job <> 'CLERK';

SQL> SELECT Ename Name, Sal Basic, Sal * 12 Annual FROM Emp WHERE Sal * 12> 6000;
```

Applying Logical Operators to Filters

The LOGICAL OPERATORS Combine the results of Two COMPONENT Conditions to Produce a Single Result.

The LOGICAL OPERA TORS Provided by ORACLE are

- ✓ Logical Conjunction Operator AND
- ✓ Logical Disjunction Operator OR
- ✓ Logical Negation operator NOT

AND Qperator

It Returns TRUE if Both or All Component Conditions are TRUE.

It Returns FALSE if Either is FALSE, Else Returns Unknown.

```
SQL> SELECT Ename, Sal, Deptno, Job FROM Emp WHERE Deptno=20 AND Job='MANAGER';

SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE Sal >= 1100 AND Job = 'CLERK';

SQL> SELECT Empno, Ename , Job, Sal FROM Emp WHERE Deptno = 10 AND Job = 'CLERK';

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Sal >= 1500 AND Sal> 5000;

SQL> SELECT Ename, Sal, Job FROM Emp WHERE (Sal >=  500 AND Sal <= 5000) AND Job = 'MANAGER';
```

OR Operator

- ✓ It Returns TRUE if Either of the Component Condition is TRUE.
- ✓ It Returns FALSE jfBoth are FALSE, Else Returns Unknown.

SQL> SELECT Ename, Sal, Deptno, Job FROM Emp WHERE Deptno = 20 OR Job= 'MANAGER';

SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE Sal >= 1100 OR Job = 'CLERK';

SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE Deptno=10 OR Job ='MANAGER';

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Sal >= 1500 OR Sal >= 5000;

SQL> SELECT Ename, Sal, Job, Deptno FROM Emp WHERE Deptno = 10 OR Deptno= 20;

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job 'CLERK' OR Job= 'MANAGER';

SQL> SELECT Ename, Sal, Job FROM Emp WHERE (Sal <= 2500 OR Sal >= 5000) OR

Job = 'MANAGER';

NOT Operator

- ✓ It Returns TRUE if the Following Condition is FALSE.
- ✓ It Returns FALSE ifthe Following Condition is TRUE.

SQL> SELECT Ename, Sal, Job FROM Emp WHERE NOT Job = 'MANAGER';

SQL> SELECT Ename, Sal, Job FROM Emp WHERE NOT Sal> 5000;

SQL> SELECT Ename, Sal, Job FROM Emp WHERE NOT Sal < 5000 ;

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE NOT Hiredate = '20-FEB-81 ';

SQL> SELECT Ename, Job, Sal, Deptno FROM Emp WHERE NOT Job = 'SALESMAN' AND Deptno= 30;

Combination of AND and OR Operators

SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE (Deptno = 10 AND Job='MANAGER') OR Sal= 3000;

SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE (Deptno = 10 AND Job ='MANAGER') OR (Deptno = 20 AND Sal >= 3000);

SOL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE (Sal> 1100 OR Job = 'CLERK') AND Deptno=20;

Some Things to Note

SOL> SELECT Ename, Sal, Job FROM Emp WHERE Job> 'MANAGER';

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job < 'MANAGER';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE Hiredate > '20-FEB-1981';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE Hiredate < '20-FEB-1981';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE Hiredate <> '20-FEB-1981';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE Job <> 'CLERK';

SOL> SELECT Ename, Sal, Comm FROM Emp WHERE Comm IS NULL;

SQL> SELECT Ename, Sal, Comm FROM Emp WHERE Comm IS NOT NULL;

SQL> SELECT Ename, Sal, Job FROM Emp WHERE NOT Job> 'MANAGER';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE NOT Hiredate = '17-DEC-1980';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE NOT Hiredate > '17-DEC-1980';

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE NOT Hiredate >' 17-DECEMBER-1980';

Rules of Operator Precedence

- ✓ The Default Precedence Order is ...
- ✓ All Comparison Operators
- ✓ NOT Operator
- ✓ AND Operator
- ✓ OR Operator

The Precedence Can be Controlled Using Parenthesis.

SQL> SELECT Ename, Deptno, Job, Sal FROM Emp WHERE Deptno=10 OR Deptno = 20 AND Job= 'SALESMAN' AND Sal> 2500 OR Sal < 1500;

SQL> SELECT Ename, Deptno, Job, Sal FROM Emp WHERE Deptno=10 OR (Deptno =20 AND Job ='SALESMAN') AND (Sal> 2500 OR Sal < 1500);

SQL*Plus Operators

BETWEEN ... AND... Operator

- ✓ This Operator is Used to Display Rows Based on a Range ofVaJues.
- ✓ The Declared Range is Inclusive.
- ✓ The Lower Limit Should be Declared First.
- ✓ The Negation of this Operator is NOT BETWEEN ... AND...

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Sal BETWEEN 1000 AND 1500;
SQL> SELECT Ename, Sal, Job FROM Emp WHERE Sal NOT BETWEEN 1000 AND 1500;
SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job BETWEEN 'MANAGER' AND 'SALESMAN';
SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job NOT BETWEEN 'MANAGER' AND 'SALESMAN';
SQL> SELECT Ename, Sal, Job, Hiredate FROM Emp WHERE Hiredate BETWEEN '17-FEB-1981' AND '20-JUN-1983';
SQL> SELECT Ename, Sal, Job, Hiredate FROM Emp WHERE Hiredate NOT BETWEEN '17-FEB-1981' AND '20-JUN-I983';

IN Operator

- ✓ The Operator is Used to Test for Values in a Specified List.
- ✓ The Operator Can be Used Upon Any Data type.
- ✓ The Negation of the Operator is NOT IN.

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Ename IN('FORD', 'ALLEN');

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Ename NOT IN('FORD', 'ALLEN');

SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Deptno IN(10,30);

SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Deptno NOT IN(10, 30);

SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE Hiredate IN('20-FEB-1981’, '09-JUN-1981 ');

```
SQL> SELECT Ename, Sal, Hiredate FROM Emp WHERE Hiredate NOT IN('20-FEB-1981',
'09-JUN-1981 ');
```

IS NULL Operator

- ✓ The Operator Tests for NULL, Values.
- ✓ It is the Only Operator That can be Used to Test for NULL's.
- ✓ The Negation is IS NOT NULL.

```
SQL> SELECT Ename, Deptno, Comm FROM Emp WHERE Comm IS NULL;

SQL> SELECT Ename, Deptno, Job, MGR FROM Emp WHERE MGR IS NULL;

SQL> SELECT Ename, Deptno, Comm FROM Emp WHERE Comm IS NOT NULL;

SQL> SELECT Ename, Deptno, Comm, MGR FROM Emp WHERE MGR IS NOT NULL;
```

LIKE Operator

- ✓ The LIKE Operator is Used to Search for a Matching Character Patterns.
- ✓ The Character Pattern Matching Operation is Referred as a WILD CARD SEARCH.
- ✓ The Available WILD CARDS in Oracle are

% Used to Represent Any Sequence of Zero or More Characters

_ Represents Any Single Character, Only At That Position.

- ✓ The WILD CARD Symbols Can be Used in Any Combination With Literal Characters.
- ✓ For Finding Exact Match For '%' and '_' the ESCAPE Option Has to be Used along with '\' Symbol.

```
SQL> SELECT Ename, Job FROM Emp WHERE Ename LIKE 'S%';

SQL> SELECT Ename, Job FROM Emp WHERE Ename NOT LIKE 'S%';

SQL> SELECT Ename, Job FROM Emp WHERE Ename LIKE '_A%';

 SQL> SELECT Ename Job FROM Emp WHERE Ename NOT LIKE '_A%';

SQL> SELECT Ename ,Sal FROM Emp WHERE Ename like'SM%';

SQL> SELECT Ename, Hiredate FROM Emp WHERE Hiredate LIKE '%-FEB-1981';

SQL> SELECT Ename, Hiredate FROM Emp WHERE Hiredate LIKE '03-%-1981 ';

SQL> SELECT * FROM Dept WHERE Dname LIKE '%\_%' ESCAPE '\';
```

Ordering Information

- ✓ The Order of Rows Returned in the Result of a Query Undefined.
- ✓ The ORDER BY Clause Can be Used to SORT The Rows in the Required Order.
- ✓ The ORDER BY Clause Should be the Last Clause in the Order of All Clauses in The SELECT Statement.
- ✓ An Expression or an Alias Can be Specified to ORDER BY Clause for Sorting.
- ✓ Default Ordering of Data is Ascending
- ✓ Numbers ~ 0 -9
- ✓ Dates ~ Earl iest -Latest.
- ✓ Strings ~ A Z.
- ✓ NULLS ~ Last.

Syntax

```
SQL> SELECT [DISTINCT] [*] {Column I [Alias] , .... } FROM Table Name

[WHERE Condition(s)] [ORDER BY {Column, Expr}[ASCIDESC]];
```

The Default Ordering Upon a Column is ASCENDING, to Change the Default Ordering DESC Should be Used After the Column Name.

Sorting Can be Implemented on Column Aliases, and Can Also be Implemented Upon Multiple Columns.

```
SQL> SELECT Ename, Job, Deptno, Hiredate FROM Emp ORDER BY Hiredate;

SQL> SELECT Ename, Job, Deptno, Hiredate FROM Emp ORDER BY Hiredate DESC;

SQL> SELECT Ename, Job, Sal FROM Emp WHERE Job = 'MANAGER' ORDER BY Sal;

SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal >= 2500 ORDER BY Job, Ename DESC;

SQL> SELECT Empno, Ename, Sal, Sal * 12 Annsal FROM Emp ORDER BY Annsal;

SQL> SELECT Empno, Ename, Sal FROM Emp ORDER BY Deptno, Sal, Hiredate;

SQL> SELECT Empno, Ename, Sal FROM Emp WHERE Sal >= 2000 ORDER BY Hiredate,Sal DESC;
```

SQL Functions:

SQL Functions are built into ORACLE and are available for use in Various Appropriate SQL Statements.

The SQL Functions Can be Used to...

- Perform Calculations on Data.
- Modify Individual Data Items.
- Manipulate Output for Groups of Rows.
- Format DATES
- Convert Column Data Types.
- SQL Functions May Accept Arguments and Always Return a Value, and Can be Nested.
- If an SQL Function is Called with a NULL Argument, then a NULL is Returned.

SQL Function Types:

SQL Identifies Two Types of Functions

- SINGLE Row FUllctions.
- MULTIPLE Row Functions.

SINGLE Row Functions:

These Functions Return a Single Result for Every Row of a Queried Table or View.

MULTIPLE Row Functions:

- These Functions Manipulate GROUPS of Rows and Return One Result Per Group of Rows.

The Single Row Functions Can Appear in

- SELECT List.
- WHERE Clause and ORDER BY Clause.

They Can Accept One or More Arguments and Return One Value For Each Row Returned By the Query.

syntax

FunctionName(Column_name/Expr, [Arg1, Arg2, ... n).

SINGLE Row Functions:

- LOWER Function.
- UPPER Function.

- ✓ INITCAP Function.

Lower Function

- ✓ It Converts Alpha Character Values to Lower Case.
- ✓ The Return Value Has the Same Data Type as Argument CHAR Type (CHAR or VARCHAR2) Syntax: LOWER(CoJumn/Expression)

SQL> SELECT 'ORACLE CORPORATION' String, LOWER('ORACLE CORPORATION') Lower FROM DUAL;

SQL> SELECT Ename, LOWER('MY INFORMATION') Lower FROM Emp;

SQL> SELECT Ename, LOWER(Ename) Lower FROM Emp WHERE Job = 'MANAGER';

Upper Function

- ✓ It Converts the Alpha Character Values to Upper Case.
- ✓ The Return Value Has the Same Data Type as the Argument CHAR.

Syntax: UPPER(ColumnlExpression)

SQL> SELECT 'oracle corporation' String, UPPER('oracle corporation') Upper FROM DUAL;

SQL> SELECT Ename, UPPER('my information') Upper FROM emp;

SQL> SELECT Ename,LOWER(Ename),UPPER(Ename) FROM Emp WHERE Job='MANAGER';

SQL> SELECT Ename, Job FROM Emp WHERE Job= UPPER('Manager');

INITCAP Function

It Converts the Alpha Character Values il1to Uppercase for the First Letter of Each Word, Keeping all Other Letters in Lower Case.

Words are delimited by White Spaces or Characters That are Not Alphanumeric.

Syntax: INITCAP(Column/Expression)

SQL> SELECT 'oracle corporation' String, INITCAP('oracle corporation') InitCap FROM DUAL;

SQl> SELECT 'The Job Title for '||INITCAP(Ename)||' is '||LOWER(Job) FROM Emp;

SELECT Ename, UPPER(Ename), LOWER(Ename), INitcap(Ename) FROM Emp;

SQL> SELECT Empno, iNiTCAP(Ename), Deptno FROM Emp WHERE Ename = UPPER('blake');

CONCAT Function

- It Concatenates the First Characters Value to the Second Character Value.
- It Accepts Only Two Parameters Accept.
- (t Return The Character Data Type.

Syntax: CONCAT(Column/Expr1, Column2/Expr2)

SQL> SELECT 'Oracle' String1, 'Corporation' String2, CONCAT('Oracle' , 'Corporation ') Concat FROM DUAL;

SQL> SELECT Ename, Job, CONCAT(Ename, Job) Concat FROM Emp WHERE Deptno = 10;

SQL> SELECT CONCAT ('The Employee Name is ' , INITCAP(Ename)) Info FROM Emp

WHERE Deptno IN(10,30);

SQL> SELECT CONCAT(CONCAT(iNITCAP(Ename), ' is a '), Job) Job FROM Emp WHERE Deptno IN(10, 20);

SUB STRING Function

Returns Specified Characters Form Character Value, Starting From a Specified Position 'm' to 'n' Characters Long.

Syntax: SUBSTR(Col/Expr, m, n) Points to Remember

- If"m" is 0 it is Treated as 1.
- If"m" is Positive, Oracle Counts From the Beginning of String to Find the First Character.
- If"m" is Negative, Oracle Counts Backwards From the End of The String.
- If"n" is Omitted, Oracle Returns All Characters to theEnd of String.
- If"n" is Less Than I or 0, A NULL is Returned.

Floating Point Numbers Passed as Arguments to SUBSTR are Automatically Convet1ed to Integers.

SQL> SELECT 'Quality Thought' String, SUBSTR('Quality Thought' ,1,7) SubString FROM DUAL;

```
SQL> SELECT 'Quality Thought' String, SUBSTR('Quality Thought' ,-5,4) SubString FROM DUAL;
```

```
SQL> SELECT 'Quality Thought' String, SUBSTR('Quality Thought' ,0,4) SubString FROM DUAL;
```

```
SQL> SELECT 'Quality Thought' String, SUBSTR('Quality Thought' ,4) SubString FROM DUAL;
```

```
SQL> SELECT 'Quality Thought' String, SUBSTR('Quality Thought', 4, 0) SubString FROM DUAL;
```

```
SQL> SELECT' Quality Thought' String, SUBSTR(' Quality Thought', 4 ,-2) SubString FROM DUAL;
```

```
SQL> SELECT' Quality Thought' String, SUBSTR(' Quality Thought', 1) SubString FROM DUAL;
```

LENGTH Function

- ✓ Returns the Number of Characters in a Value.
- ✓ If the String has Data Type CHAR, The Length Includes All Trailing Blanks.
- ✓ If The String is NULL, It Returns NULL.

Syntax: LENGTH(ColumnlExpression)

```
SQL> SELECT 'ORACLE' String, LENGTH('ORACLE') Length FROM DUAL;
```

```
SQL> SELECT LENGTH (Ename)||' Characters exit in '||INITCAP(Ename) ||'''s Name' AS "Names and Lengths" FROM Emp;
```

INSTRING Function

- ✓ It Returns The Numeric Position of a Named Character.

Syntax: INSTR(ColumnlExpression, Char, m, n)

The INSTR Functions Search String for Substring that is Supplied.

The Function Returns an Integer Indicating the Position of the Character in String That is The First Character of This Occurrence.

Searches for Column OR Expression Beginning With its ‘mth’ Character For The 'nth’ Occurrence.

'm' Can be Positive or Negative, ifNegative Searches Backward From The End of Column OR Expression.

- ✓ The Value of 'n' Should be Positive.
- ✓ The Default Values of Both 'm' and On' Are 1.
- ✓ The Return Value is Relative to The Beginning of CharI Regardless of The Value of 'm', and is Expressed in Characters.

If the Search is Unsuccessful, the Return Value is Zero.

SQL> SELECT 'STRING' String, INSTR('STRING', 'R') InString FROM DUAL;

SQL> SELECT 'CORPORATE FOOR' String, iNSTR('CORPORATE FOOR', 'OR', 3, 2) InString

FROM DUAL;

SQL> SELECT 'CORPORATE FOOR' String, INSTR('CORPORATE FLOOR', 'OR', -3, 2)

InString FROM DUAL;

SQL> SELECT Job, INSTR(Job, 'A', 1, 2) Position FROM Emp WHERE Job = 'MANAGER';

SQL> SELECT Job, INSTR(Job, 'A', 2, 2) Position FROM Emp WHERE Job ='MANAGER';

SQL> SELECT Job, INSTR(Job, 'A', 3, 2) Position FROM Emp WHERE Job= 'MANAGER';

SQL> SELECT Job, INSTR(Job, 'A', 2) Position FROM Emp WHERE Job = 'MANAGER';

LPAD Function

- ✓ Pads The Character Value Right Justified to a Total Width of On' Character Positions.
- ✓ The Default Padding Character is Space.
- ✓ Syntax: LPAD(Char1, n, 'Char2')

SQL> SELECT 'Page1' String, LPAD('Page1',15, '*.') LPadded FROM DUAL;

SQL> SELECT 'Page1' String, LPAD('Page1',15,'@') LPadded FROM DUAL;

SQL> SELECT Ename, LPAD(Ename, 10, '-') LPadded FROM Emp WHERE Sal >= 2500;

RPAD Function

- ✓ Pads the Character Value Left Justified to a Total Width of'n' Character Positions.
- ✓ The Default Padding Character is Space.

Syntax: RPAD(Char1, n, 'Char2')

SQL> SELECT 'Page1' String, RPAD('Page1' , 15, '*.') RPadded FROM DUAL;

SQL> SELECT 'Page I' String, RPAD('Page1' , 15) RPadded FROM DUAL;

SQL> SELECT Ename, Rpad(Ename, 10, '-') RPadded FROM Emp WHERE Sal> = 2500;

SQL> SELECT Ename, LPAD(Ename, 10, '-') LPadded, RPAD(Ename, 10, '-') RPadded FROM Emp;

SQL> SELECT Ename, LPAD(RPAD(Ename, 10, '-'),15, '-') Centered FROM Emp;

LTRIM Function

- ✓ It Enables to TRIM Heading Characters From a Character String.
- ✓ All The Leftmost Characters That Appear in The SET are Removed.

Syntax: LTRIM(Char, SET)

SQL> SELECT 'xyzXxyLAST WORD' String, LTRIM('xyzXxyLAST WORD', 'xy') LTrimmed FROM DUAL;

SQL> SELECT Job, LTRIM(Job, 'MAN') LTrimmed FROM Emp WHERE Job LIKE 'MANAGER';

RTRIM Function

- ✓ It Enables the Trimming of Trailing Characters From a Character STRING.
- ✓ All the Right Most Characters That Appear in The SET are Removed.

Syntax: RTRIM(Char, SET)

SQL> SELECT 'BROWNINGyxXxy' String, RTRIM(,BROWNINGyxXxy', 'xy') RTrimmed FROM DUAL; SQL> SELECT RTRIM(Job, 'ER'), Job FROM Emp WHERE LTRIM(Job, 'MAN') LIKE 'GER';

REPLACE Function

- ✓ It Returns the Every Occurrence of Search String Replaced by The Replacement String.
- ✓ If the Replacement String is Omitted or NULL, All Occurrences of Search String are Removed.
- ✓ It Substitutes One String for Another as Well as Removes Character Strings.

Syntax: REPLACE(Char, Search_String, Replace_Str)

SQL> SELECT 'JACK AND JUE' String, REPLACE('JACK AND JUE' , 'J', 'BL') Replaced FROM DUAL;

SQL> SELECT Ename, REPLACE(JOB, 'MAN', 'DAM') Replaced FROM Emp WHERE Job= 'MANAGER';

SQL> SELECT Job, REPLACE (Job, 'P') FROM Emp WHERE Job= 'PRESIDENT';

SQL> SELECT Job, REPLACE (Job, 'MAN', 'BOY') FROM Emp WHERE Job = 'SALESMAN';

TRANSLATE Function

- ✓ Used to Translate Character by Character in a String.

Syntax: TRANSLATE(char, From, To)

- ✓ It Returns a CHAR With All Occurrences of Each Character in 'From' Replaced By lts Corresponding Character in 'To'.
- ✓ The Argument FROM Can Contain More Characters Than TO.
- ✓ If The Extra Characters Appear in CHAR, They are Removed From the Return Value.

SQL> SELECT Job, TRANSLATE(Job, 'MN', 'OM') FROM Emp WHERE Job = 'MANAGER';

SQL> SELECT Job, TRANSLATE(Job, 'A', 'O') FROM Emp WHERE Job= 'SALESMAN';

CHR Function

- ✓ It Returns a Character Having the ASCII Equivalent to 'n'.

Syntax: CHR(n)

SQL> SELECT CHR(65) Sample FROM DUAL;

ASCII Function

It Returns The ASCII Representation in the Character Database Set of The First Characters of the CHAR.

Syntax: ASCII(Char)

SQL> SELECT ASCIT('A'), ASCIJ('APPLE') FROM DUAL;

SELECT ASCIi('Q'),ASCIi('U'),ASCIi('A'),ASCIi('L'),ASCIi('I'),ASCIi('T'),ASCIi('Y'), ASCIi('APPLE') FROM DUAL;

Working with Dates:

- ✓ Oracle Stores Dates in an Internal Numeric Format.
- ✓ The Dates in Oracle Range From JANUARY 1,4712 BC to DECEMBER 31, 9999 AD.
- ✓ The Default Display and Input Format for any Date is DD-MON-YY.
- ✓ The Internal Date Format Represents
- ✓ Century ~ Year ~Month ~Day ~Hours ~Minutes ~Seconds SYSDATE:
- ✓ It is a Date Function That Returns Current DATE and TIME.
- ✓ SYSDATE is Generally Selected Upon a DUMMY Table.

SQL> SELECT SYSDATE FROM DUAL;

Date Arithmetic

- ✓ As Database Stores Dates as Numbers, Arithmetic Operations can be Implemented.
- ✓ Number Constants Can be Added or Subtracted Upon Dates.
- ✓ The Operations That Can be Applied are
- ✓ Date + Num ber ~ Returns Date
- ✓ Adds Number of Days to a Date.
- ✓ Date -Number ~ Returns Date.
- ✓ Subtracts Number of Days From a Date.
- ✓ Date -Date ~ Returns Number of Days.
- ✓ Subtracts One Date from Another Datc.
- ✓ Date + Number/24 -7 Retunis Date.
- ✓ Adds Number of Hours to a Date.

SQL> SELECT SYSDA TE, SYSDATE + 3 FROM DUAL;

SQl> SELECT SYSDATE, SYSDATE -3, SYSDATE + 72/24 FROM DUAL;

SQL> SELECT Ename, Hiredate, Hiredate + 3 FROM Emp;

SQl> SELECT Ename, Hiredate, Hiredate -3 FROM Emp;

SQl> SELECT Ename, Hiredate, SYSDATE Hiredate FROM Emp;

SQL> SELECT Ename, (SYSDATE -Hiredate) / 7 Weeks FROM Emp WHERE Deptno = 10;

DATE Functions

ADD_MONTHS Function

Syntax: ADD_MONTHS(D, n)

- ✓ The Argument 'n' Can be Any Positive OR Negative Integer.

```
SQl> SELECT SYSDATE, ADD_MONTHS(SYSDATE, 2) FROM DUAL;
```

```
SQL> SELECT Sal, Hiredate, ADD_MONTHS(Hiredate, 2) FROM Emp WHERE Deptno =20;
```

MONTHS_BETWEEN Function

Syntax: Months_Between(D1, D2)

- ✓ It Returns Number of Months Between Dates 'd1' and 'd2'.
- ✓ If 'd1' is Later Than 'd2', The Result is Positive, else Negative.

If'd1' and 'd2' are Either the Same Days of The Months or Both Last Days of The Months, The Result is Always An Integer.

```
SQL> SELECT Ename, HireDate, SYSDATE, MONTHS_BETWEEN(SYSDATE, Hiredate) FROM Emp;
```

SELECT Empno, Hiredate, MONTHS_BETWEEN(SYSDATE, Hiredate) FROM Emp**Next Day_Function:**

Syntax: NEXT_DAY(d, Char)

- ✓ It Returns The Date of The First Week Day Named By CHAR, That is Later Than the Date'd'.
- ✓ The CHAR Must be a Day of The Week in the Sessions Date Language.
- ✓ The Day ofThe Week Can Be Full Name or The Abbreviation.

```
SQL> SELECT SYSDATE, NEXT_DAY(SYSDATE, 'WED') FROM DUAL;
```

```
SQL> SELECT Sal, Hiredate, NEXT_DAY(Hiredate, 'MONDAY')FROM Emp;
```

LAST DAY Function

Syntax: LAST_DAY (D)

- ✓ It Returns The Date of The Last Day of The Month That Contains' D' .
- ✓ Mostly Used to Determine How Many Days Are Left in the Current Month.

```
SQL> SELECT SYSDATE, LAST DAY(SYSDATE) LastDay FROM DUAL;
```

```
SQL> SELECT LAST_DAY(SYSDATE) Last, SYSDATE, LAST_DAY(SYSDATE) -SYSDATE Daysleft FROM DUAL;
```

Conversion Functions

- ✓ The Conversion Functions Convert a Value From One Data Type to Another.

TO CHAR(Date Conversion)

Syntax: TO_CHAR(DATE, fmt, 'nlsparams')

- ✓ Converts Date of DATE Data Type to a Value ofVARCHAR2 Data Type in The Format Specified.
- ✓ 'fmt' is the Optional Date Format, That Can be Used.
- ✓ The 'nlsparams' Specifies the Language in Which Month and Day Names And Abbreviations are Returned.

Date Format Models:

- ✓ The Date Format Models Can be Used in The TO CHAR Function to Translate a DATE Value From Original Format to User Format.
- ✓ The Total Length of a Date Format Model Cannot Exceed 22 Characters.

Date Format Elements

- ✓ A Date Format Model is Composed of One or More Date Format Elements.
- ✓ For Input Format Models, Format Items Cannot Appear Twice, and Format Items That Represent Similar Information Cannot be Combined.
- ✓ Capitalization in a Spelled Word, Abbreviation, or Roman Numeral Follows Capitalization in the Corresponding Format Element.
- ✓ Punctuation Such as Hyphens, Slashes, Commas, Periods and Colons.

AD or A.D.lBC or B.c. Indicator

- ✓ Indicates ADIBC With OR Without Periods.

SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'AD') FROM DUAL;

SQL> SELECT TO_CHAR(SYSDATE, 'B.C.'), TO_CHAR(SYSDATE, 'A.D.') FROM DUAL;

SQL> SELECT Ename, Sal, Hiredate, TO_CHAR(Hiredate, 'A.D.') FROM Emp;

Meridian Indicator: AM OR A.M.lPM OR P.M.

- ✓ It Indicates Meridian Indicator With or Without Periods.

SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'A.M. '), TO_CHAR(SYSDATE, 'PM') FROM DUAL;

SQL> SELECT Ename, Sal, Hiredate, TO_CHAR(Hiredate, 'AM') FROM Emp;

Century Indicator: CC Indicates The Century.

SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'CC') FROM DUAL;

Four Digit Year Indicator: YYYY

SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'YYYY') Four, TO_CHAR(SYSDATE, 'YYY') Three FROM DUAL;

SQL> SELECT Ename, HireDate, TO_CHAR(HireDate, 'YYYY') FROM Emp WHERE Deptno =20 ;

Spelled Year Indicator: YEAR OR SYEAR

- ✓ Returns the Numerical Year in Spelling.
  SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'YEAR') FROM DUAL;
  SQL> SELECT Ename, HireDate, TO_CHAR(HireDate, 'YEAR') FROM Emp;

Quarter of the Year Indicator: Q

- ✓ Returns the Quarter of The Year.
- ✓ Quarter Starting With the Month of January and Ending With Every Three Months.

SQL> SELECT SYSDATE, TO CHAR(SYSDATE, 'Q') FROM DUAL;

SQL> SELECT Ename, HireDate, TO_CHAR(HireDate, 'Q') FROM Emp WHERE TO_ CHAR(HireDate, 'Q') = 4;

Numeric Month Indicator: MM

- ✓ Returns the Numeric Abbreviation of the Month.

SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'MM-YYYY’) FROM DUAL;

SQL> SELECT Ename, HireDate, TO_CHAR(HireDate, 'DD-MM-YYYY') FROM Emp WHERE TO_CHAR(HireDate, 'MM')=12

Abbreviated Month Indicator: MON

Returns the Abbreviated Name of The Month.

SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'MM-MON') FROM DUAL;

Month Spelling Indicator: MONTH

Spells the Name of the Month, Padded to a Length of 9 Characters.

```
SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'MON-MONTH') FROM DUAL;
```

```
SQL> SELECT Ename, HireDate,TO_CHAR(HireDate, 'MONTH, YYYY') FROM Emp;
```

Twelve Hour Clock Mode: HH OR HH 12

Returns the Hour ofThe Day in Twelve Hour Clock Mode.

```
SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'HH'), TO_CHAR(SYSDATE, 'HH12, AM') FROM DUAL;
```

```
SQL> SELECT Ename, HireDate, TO_CHAR(HireDate, 'HH12: AM') FROM Emp;
```

Twenty Hour Clock Mode: HH24

Returns the Hour of the Day in Twenty Four Hour Clock Mode.(0-23)

```
SQL> SELECT SYSDATE,TO_CHAR(SYSDATE, 'HH24') FROM DUAL;
```

Minutes Indicator: MI

Returns the Minutes From The Given Date(0-53).

```
SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'MI'), TO_CHAR(SYSDATE, 'HH:MI') FROM DUAL;
```

```
SQL> SELECT Ename, Sal, TO_CHAR(HireDate, 'HH:MI') FROM Emp WHERE Job 'CLERK';
```

Seconds Indicator: SS

Returns Seconds From the Given Date(0-59).

```
SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'SS'), TO_CHAR(SYSDATE, 'HH:MI:SS') FROM DUAL;
```

```
SQL> SELECT SYSDATE, TO_CHAR(SYSDATE, 'DD-MONTH-YYYY, HH:MI:SS A.M.') FROM DUAL;
```

```
SQL> SELECT Ename, Sal, HireDate, TO_CHAR(HireDate, 'HH24:MI:SS') FROM Emp WHERE Deptno IN(10, 30);
```

TO_DATE function:

Syntax: TO_DATE(Char, fmt, 'nlsparam')

Converts Given Char of CHAR or VARCHAR2 Data Type to a Value of DATE Data Type,

The 'fmt' is an Optional Date Format Specifying the Format of CHAR.

```
SQL> SELECT Ename, HireDate, ADD_MONTHS(TO_DATE ('17-DEC-1980','DD-MON-YY'),3)
FROM Emp WHERE HireDate= '17-DEC-1980';
```

Aggregate or Group Functions:

- These Functions Return a Single Row Based on Groups of Rows.
- These Functions Can Appear in SELECT Lists and HAVING Clauses Only.
- These Functions Operate on Sets of Rows to Give One Result Per Group.
- The Sets May Be The Whole Table or The Table Split Into Groups.

Guidelines to use Group Functions:

- DISTINCT Makes The Function to Consider Only Non Duplicate Values.
- Syntax: GroupFul1ctionName(Distinct/ALL Col])
- The Data Types For Arguments May Be CHAR, VARCHAR2, NUMBER OR DATE.
- All Group Functions Except COUNT(*) Ignore NULL Values. To Substitute a Value For NULL Value, Use the NVL Function.
- When a Group FUllction is Declared in a SELECT List, no Single Row Columns Should be Declared.
- When a Group Function is Declared in a SELECT List, Other Columns Can Be Declared, But They Should Be Grouped Columns, And All The Non Functional Columns Should Be Declared Into a GROUP BY Clause.

Average Function Syntax: AVG(DISTINCT/ALL Col)

- It Returns the AVERAGE Value of Column.
- It Ignores NULL Values.

```
SQL> SELECT AVG(Sal), AVG(DISTINCT Sal) FROM Emp;
```

```
SQL> SELECT AVG(Comm) FROM Emp;
```

SUM Function Syntax: SUM(DISTINCT/ALL Col)

- It Returns the SUM Value of Column.
- It ignores NULL Values.

```
SQL> SELECT SUM(Sal), SUM(DISTINCT Sal) FROM Emp;
```

SQL> SELECT SUM(Comm), SUM(DISTINCT Comm) FROM Emp;

MAXIMUM Function Syntax: MAX(DISTINCT/ALL Col)

- ✓ It Returns the Maximum Value of Column.
- ✓ It Ignores NULL Values.. ;

SQL> SELECT MAX(Sal), MAX(DISTINCT Sal) FROM Emp;

SQL> SELECT MAX(Comm) , MAX(DISTINCT Comm)FROM Emp;

MINIMUM Function Syntax: MIN(DISTINCT ALL Col)

- ✓ It Returns the Minimum Value of The Column.
- ✓ It Ignores NULL values.

SQL> SELECT MIN(Sal), MIN(DISTINCT Sal)FROM Emp;

SQL> SELECT MIN(Comm), MIN(DISTINCT Comm) FROM Emp;

COUNT Function Syntax: COUNT(*/DlSTlNCT/ALL Col)

- ✓ It Returns the Number of Rows in The Query.
- ✓ If'*' is Used Returns All Rows, Including Duplicated And NULLs.
- ✓ It Can Be Used to Specify The Count of All Rows or Only Distinct Values of Col.

SQL> SELECT COUNT(*) FROM Emp;

SQL> SELECT COUNT(Job), COUNT(DISTINCT Job) FROM Emp;

SQL> SELECT COUNT(Sal), COUNT(Comm) FROM Emp;

SQL> SELECT COUNT(Empno), COUNT(DlSTINCT MGR) FROM Emp;

Creating Groups of Data:

The **Group By** Clause is used to decide the rows in a table into groups.

Syntax1: SELECT ColumnName 1, ColumnName2, ... FROM TableName WHERE Condition(s) GROUP BY ColumnName(s) ORDER BY Column(s);

Syntax2: SELECT ColumnName, GRP_FUN(Column) FROM TableName WHERE Condition(s) GROUP BY ColumnName(s) ORDER BY Column(s);

Guidelines to Use GROUP BY Clause: If The GROUP Function is Included in a SELECT Clause, We Should Not Use Individual Result Columns.

- ✓ The Extra Non Group Functional Columns Should Be Declared in The GROUP BY Clause.
- ✓ Using WHERE Clause, Rows Can Be Pre Excluded Before Dividing Them Into Groups.
- ✓ Column Aliases Cannot Be Used in GROUP BY CLAUSE.
- ✓ By Default, Rows are Sorted by Ascending Order of The Columns Included in The GROUP BY LIST.
- ✓ The Column Applied Upon GROUP BY Clause Need Not be Part of SELECT list.

```
SQL> SELECT Deptno FROM Emp GROUP BY Deptno;

SQL> SELECT Job FROM Emp GROUP BY Job;

SQL> SELECT MGR FROM Emp GROUP BY MGR;

SQL> SELECT TO_CHAR(HireDate, 'YYYY') YearGroup FROM Emp GROUP BY

TO_CHAR(HireDate, 'YYYY');

SQL> SELECT TO_CHAR(HireDate, 'Month') MonthGroup FROM Emp GROUP BY
TO_HAR(HireDate, 'Month');

SQL> SELECT TO_CHAR(HireDate, 'Month') MonthGroup FROM Emp WHERE
TO_CHAR(HireDate, 'Month') <> 'September'
GROUP BY TO_CHAR(HireDate, 'Month');
```

Creating Group Wise Summaries

```
SQL> SELECT Deptno, AVG(Sal) FROM Emp GROUP BY Deptno;

SQL> SELECT Deptno, AVG (Sal) FROM Emp GROUP BY Deptno ORDER By AVG (Sal) ;

SQL> SELECT Deptno, MIN(Sal), MAX(Sal) FROM Emp GROUP BY Deptno;

SQL> SELECT Deptno, Job, SUM(Sal) FROM Emp GROUP BY Deptno, Job;

SQL> SELECT Deptno, MIN(Sal), MAX(SaJ) FROM Emp WHERE Job = 'CLERK' GROUP
BY Deptno;

SQL> SELECT Deptno, SUM(Sal), AVG(Sal) FROM Emp WHERE Job ='CLERK' GROUP
BY Deptno;
```

Excluding Groups of Results Having Clause

- ✓ It is Used to Specify Which Groups Are to be Displayed.
- ✓ The Clause is Used to Filter Data That is Associated With Group Functions.

Syntax: SELECT Column, Group_Function FROM Table [WHERE Condition(s)] [GROUP BY Group_By_Expr] [HAVING Group_ Condition(s)] [ORDER BY Column_Name/Alias];

Steps Performs By Having Clause:

- ✓ First The Rows are Grouped.
- ✓ Second The Group Function is Applied to The Identified Groups.
- ✓ Third The Groups That Match The Criteria in The HAVING Clause are Displayed.
- ✓ The HAVING Clause can Precede GROUP BY Clause, But it is More Logical to Declare it After GROUP BY Clause.
- ✓ GROUP BY Clause Can Be Used, Without a Group Function in The SELECT list.
- ✓ If Rows are Restricted Based on the Result of a Group Function, We Must Have a GROUP BY Clause as well as the HAVING Clause.

Existence of GROUP BY Clause Does Not Guarantee The Existence of HAVING Clause, But The Existence of HAVING Clause demands the Existence of GROUP BY Clause.

SQL> SELECT Deptno, AVG(Sal) FROM Emp GROUP BY Deptno HAVING MAX(Sal) > 2900;

SQL> SELECT Job, SUM(Sal) Payroll FROM Emp WHERE Job NOT LIKE 'SALES%'

GROUP BY Job HAVING SUM(Sal) > 5000 ORDER BY SUM (Sal);

SQL> SELECT Deptno, MIN(Sal), MAX(Sal) FROM Emp WHERE Job = 'CLERK' GROUP BY Deptno HAVING MIN (Sal) < 1000;

SQL> SELECT Deptno,SUM(Sal) FROM Emp GROUP BY Deptno HAVING COUNT(Deptno) > 3;

SQL> SELECT Deptno, AVG(Sal), SUM(Sal), MAX(Sal), MIN(Sal) FROM Emp GROUP BY Deptno HAVING COUNT(*) > 3;

SQL> SELECTDeptno, AVG(Sal), SUM(Sal) FROM Emp GROUP BY Deptno HAVING AVG (Sal)> 2500;

SQL> SELECT Deptno, Job, SUM(Sal), AVG(Sal) FROM Emp GROUP BY Deptno, Job HAVING AVG(Sal) > 2500;

Nesting of Group Functions:

Group Functions Can be Nested To a Depth of Two Levels.

SQl> SELECT MAX(AVG(Sal)) FROM Emp GROUP BY peptno;

SQl> SELECT MAX(SUM(Sal)), MIN(SUM(SAL)) FROM Emp GROUP BY Deptno;

SQL> SELECT MAX(SUM(Sal)), MIN(AVG(Sal)) FROM Emp GROUP BY Job;

Miscellaneous Functions

USER Function Syntax: USER

- ✓ It Returns the Current Oracle Users Name Within The VARCHAR2 Data Type.

The Function Cannot Be Used in The Condition of The CHECK Constraint.

SQL> SELECT USER FROM DUAL;

Increase the Integrity and Quality of Our Database

Data Integrity in data Bases

Data Integrity

- ✓ It is a State in Which All the Data Values Stored in The Data Base Are Correct.
- ✓ Enforcing Data Integrity Ensures The Quality of The Data in The Data Base.

Constraints in Oracle

Constraints in Data Bases Are Used to Define An Integrity Constraint, As a Rule That Restricts The Values in a Data Base.

- ✓ NOT NULL Constraint.
- ✓ UNIQUE Constraint.
- ✓ PRIMARY KEY Constraint.
- ✓ FOREIGN KEY Constraint.
- ✓ CHECK Constraint.

Declaration Style

- ✓ Column Level (OR) IN LINE Style.
- ✓ Table Level (OR) OUT OF LINE Style. Column Level:
- ✓ They Are Declared As Part of The Definition of An Individual Column or Attribute.

Table Level:

They are declared as Part of the Table Definition.
Definitely Applied When the Constraint is applied on Combination of Columns Together.

Note: NOT NULL Constraint is The Only Constraint Which Should Be Declared As INLINE Only.

- ✓ Every Constraint is managed by Oracle with a Constraint Name in The Meta Data.
- ✓ Hence When We Declare a Constraint if We do not Provide a Constraint Name Oracle Associates the Constraint With Name.
- ✓ Rather Than Depending on the Oracle Supplied Constraint Name, it is Better to Define Our Own Name for all Constraints.
- ✓ When Constraints are Named We Should Use 'CONSTRAINT' Clause. The CONSTRAINT Clause Can Appear in
- ✓ CREATE And ALTER Table Statement.

NOT NULL Constraint:

- ✓ A NOT NULL Constraint Prohibits a Column From Containing NULL Values. ,
- ✓ NOT NULL Should Be Defined Only At COLUMN Level.

Syntax

SQL> CREATE Table <table_Name>

(Column_Name1 <Data Type>(Width) NOT NULL, Column_Name2 <Data Type>(Width) CONSTRAINT ConsName NOT NULL, Column_NameN <Data Type>(Width));

Illustration

SQL> CREATE TABLE Students (StudNo NUMBER(6) CONSTRAINT StudnoNN NOT NULL, StudName VARCHAR2(25) CONSTRAINT StudNameNN NOT NULL, CourseName VARCHAR2(25) CONSTRAINT CourseNameNN NOT NULL, JoinDate DATE NOT NULL);

UNIQUE Constraint

- ✓ The UNIQUE Constraint Designates a Column As a UNIQUE Key.
- ✓ A Composite UNIQUE Key Designates a Combination of Columns As The UNIQUE Key.
- ✓ A Composite UNIQUE Key is Always Declared At The Table Level.
- ✓ To Satisfy a UNIQUE Constraint, No Two Rows in The Table Can Have The Same Value For The UNIQUE Key.

Oracle creates an index Implicitly on The UNIQUE Key Column.

Restrictions:

A Table or View Can Have Only One UNIQUE Key Column.

A Composite UNIQUE Key Cannot Have More Than 32 Columns.

- ✓ Same Column or Combination of Columns Cannot Be Designated As Both PRIMARY KEY and UNIQUE KEY.

Svntax SQL> CREATE Table <Table_Name>

(Column_Name I <Data Type>(Width) UNIQUE, Column_Name2 <Data Type>(Width)

CONSTRAINT ConsName UNIQUE,

Column_NameN <Data Type>(Width));

Illustration: 1

Column Level Syntax

SQL> CREATE Table Promotions (promo ID NUMBER(fi) CONSTRAINT PromoIDUNQ UNIQUE, PromoName VARCHAR2(20), PromoCategory VARCHAR2(15), PromoCost NUMBER(lO, 2), PromoBegDate DATE, PromoEndDate DATE);

Illustration: 2 Composite UNIQUE Constraint Syntax

SQL> CREATE Table WareHouse (WareHouseID NUMBER(6), WareHouseName VARCHAR2(30), Area NUMBER(4), DockType VARCHAR2(50), WaterAccess VARCHAR2(\O), RailAccess VARCHAR2(IO), Parking VARCHAR2(10), Vclearance NUMBER(4), CONSTRAINT WareHouseUNQ UNIQUE(WareHouselD,WareHouseName));

PRIMARY KEY Constraint:

- ✓ A PRIMARY KEY Constraint Designates a Column As The PRIMARY KEY of a TABLE or VIEW.
- ✓ A COMPOSITE PRIMARY KEY Designates a Combination of Columns As The PRIMARY KEY.
- ✓ When The Constraint is Declared At Column Level Only PRIMARY KEY Keyword is Enough.
- ✓ A Composite PRIMARY KEY is Always Defined At Table Level Only.
- ✓ A PRIMARY KEY Constraint Combines a NOT NULL and UNIQUE Constraint in One declaration.

Restrictions:

- ✓ A TABLE or VIEW Can Have Only One PRIMARY KEY.

A Composite PRIMARY KEY Cannot Have More Than 32 Columns.

- ✓ The Same Column or Combination of Columns Cannot Be Designated Both As PRIMARY KEY and UNIQUE KEY.

Syntax

SQL> CREATE Table <Table_Name>

(Column_Name1 <Data Type>(Width) CONSTRAINT ColNamePK PRIMARY KEY, Column_Name2 <Data Type>(Width), Column NameN <Data Type>(Width));

Illustration: I Column level Syntax:

SQL> CREATE TABLE Locations (LocationlD NUMBER(4) CONSTRAINT LoclDPK PRIMARY KEY, StAddress VARCHAR2(40) NOT NULL, PostalCode VARCHAR2(6) CONSTRAJNT PCNN NOT NULL, City VARCHAR2(30) CONSTRAINT CityNN NOTNULL);

Illustration: 2 Table Level Syntax

SQL> CREATE TABLE Locations (LocationlD NUMBER(4), StAddress VARCHAR2(40) NOT NULL, PostalCode V ARCHAR2(6) CONSTRAINT PCNN NOT NULL, City VARCHAR2(30) CONSTRAINT CityNN NOT NULL,CONSTRAINT LocIDPK PRIMARY KEY(LocationlD));

Illustration: 3

SQL> CREATE TABLE Saleslnfo (SalelD NUMBER(6), CustTD NUMBER(6), ProdlD NUMBER(6), Quantity NUMBER(6) NOT NULL, SaleDate DATE NOT NULL, SaleDesc LONG NOT NULL, CONSTRAINT ProdCustlDPK PRIMARY KEY(SalelD, ProdlD, custID));

PRIMARY KEY Constraint With Composite Key Constraint Style.

FOREIGN KEY Constraint

- ✓ It is Also Called As REFERENTIAL INTEGRITY CONSTRAINT,
- ✓ It Designates a Column as FOREIGN KEY And Establishes a RELATION Between The FOREIGN KEY And a Specified PRIMARY or UNiQUE KEY.
- ✓ The TABLE or VIEW Containing The FOREIGN KEY is Called the Child Object.
- ✓ The TABLE or VIEW Containing The REFERENCED KEY is Called the Parent Object.
- ✓ The FOREIGN KEY And The REFERENCED KEY Can Be in The Same TABLE or VIEW.

The Corresponding Column or Columns of the FOREIGN KEY And The REFERENCED KEY Must Match DATA TYPE.

A COMPOSITE FOREIGN KEY CONSTRAINT, Must Refer To a COMPOSITE UNIQUE

KEY or a COMPOSITE PRIMARY KEY in the PARENT TABLE or VIEW.

CHECK Constrain

- ✓ It Defines a Condition That Each Row Must Satisfy.

Restrictions

The Condition of a CHECK Constraint Can Refer To Any Column in The Same Table, But It Cannot Refer to Columns of Other Tables.

The CHECK Constraints Can Be Defined At The Column Level or TABLE Level.

Adding Constraints to a Table

A Constraint Can Be Added To a Table At Any Time After The Table Was Created By Using

ALTER TABLE Statement, Using ADD Clause.

Syntax

SQl> ALTER TABLE <tableName> ADD [CONSTRAINT <ConstraintName>]

CONS_TYPE(Column_Name);

Guidelines

- ✓ We Can ADD, DROP, ENABLE, or DISABLE a Constraint, but Cannot Modify The Physical Structure of The Table.
- ✓ A NOTNULL Can Be Added to Existing Column By Using The MODIFY Clause ofthe ALTER TABLE Statement.
- ✓ NOT NULL Can Be Defined Only When The Table Contains No Rows.

Example

SQl> ALTER TABLE Emp ADD CONSTRAINT Emp_Mgr_FK FOREIGN KEY(Mgr) REFERENCES Emp(Empno);

DROPPING Constraints

Syntax

SQL> ALTER TABLE <table_Name> DROP PRIMARY KEY/UNlQUE(Column) CONSTRAINT ConstraintName[CASCADE];

- ✓ The CASCADE Option of the DROP Clause Causes Any Dependent Constraints Also To Be Dropped.

Example

SQL> ALTER TABLE Dept DROP PRIMARY KEY CASCADE;

SQL> ALTER TABLE Emp DROP CONSTRAINT Emp_Mgr_FK;

VIEWING Constraints

- ✓ To View All Constraints On a Table Query Upon the Data Dictionary USER CONSTRAINTS.
- ✓ The Codes That Are Revealed Are...

C -: CHECK

P -: PRIMARY KEY

R -: REFERENTIAL INTEGR1TY

U -: UNIQUE KEY

SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION FROM USER_CONSTRAINTS WHERE TABLE_NAME = 'EMP';

JOINS

- ✓ A Join is a Query That Combines Rows from Two or More Tables, Views, or Materialized Views.
- ✓ A Join is Performed Whenever Multiple Tables Appear in The Queries FROM Clause.
- ✓ The Queries SELECT List Can Select Any Columns From Any of These Tables.
- ✓ The Common Column Names Within The Tables Should Qualify All References To These Columns.

SQl> SELECT Empno, Ename, Dname, Loc FROM Emp, Dept;

SQL> SELECT Empno, Ename, Sal, Grade FROM Emp, SalGrade;

SQl> SELECT Empno, Ename, Dname, Loc, SalGrade FROM Emp, Dept, SalGrade;

SQL> SELECT Empno, Ename, Dept.Deptno, Dname, Loc FROM Emp, Dept;

JOIN Condition

- Many Join Queries Contain WHERE Clause, Which Compares Two Columns, Each From a Different Table.
- The Applied Condition is Called a JOIN CONDITION.
- To Execute a Join ...
- Oracle Combines Pairs of Rows, Each Containing One Row From Each Table, For Which The JOIN Condition Evaluates to TRUE.

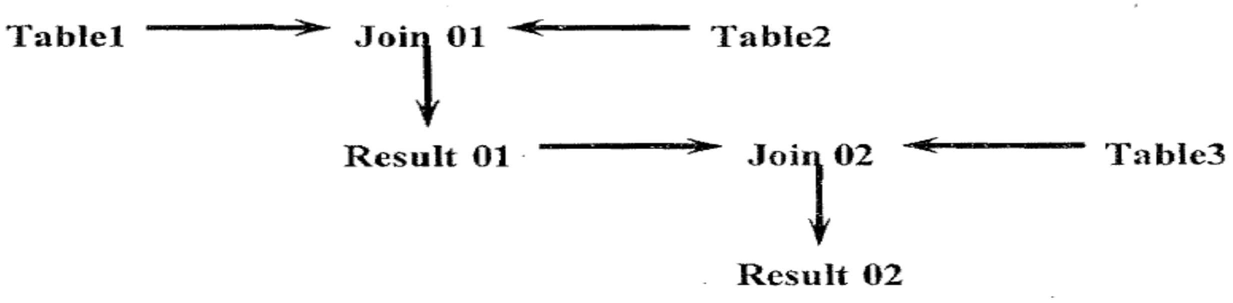
The Columns in The Join Conditions Need Not Be Part of The SELECT List.

- To Execute a Join of Three or More Tables
- Oracle First Joins Two ofThe Tables Based on The Join Conditions Comparing These Columns And Then Join's The Result To Another Join.

The Oracle Optimizer Determines The Order in Which ORACLE Should Join The Tables Based on ...

- Given JOIN Condition(s).
- INDEXES upon the Tables.
- STATISTICS for the Tables.

Syntax WHERE Table1.Column1 = Table2.Column2



Guidelines

- When Writing a SELECT Statement That JOIN'S Tables, Precede the Column Name with the Table Name for Clarity and Enhance Database ACCESS.
- If The Same Column Name Appears in More Than One Table, The Column Name Must Be Prefixed With The Table Name.
- To Join 'n' Tables Together, We Need a Minimum of 'n-1' Join Conditions.

Types of joins:

Equi Joins OR Simple Joins OR Inner Joins

- An EQUI JOIN is a Join with a Join Condition Containing an Equality Operator.
- It Combines Rows That Have Equivalent Values For The Specified Columns.
- Qualifying Ambiguous Column Names
- The Names ofThe Column Names Should Be Qualified in The WHERE Clause, With The Table . Name To Avoid Ambiguity.
- If ThereAre no Common Column Names Between The Two Tables, The Qualification is Not Necessary But It is Better.

```
SQL> SELECT Emp.Empno Empno, Emp.Ename Ename, Emp.Deptno Deptno, Dept.Deptno Deptno, DepLDname Dname, Dept.Loc Loc FROM Emp, Dept WHERE Emp.Deptno Dept. Deptno;
```

```
SQL> SELECT Empno, Ename, Emp.Deptno, Loc FROM Emp, Dept WHERE Emp.Deptno = Dept.Deptno AND Job= UPPER('manager');
```

```
SQL> SELECT Empno, Ename, Sal * 12 AnnSal, Emp.Deptno, Loc FROM Emp, Dept WHERE Emp.Deptno = Dept.Deptno;
```

Using Table Aliases:

- Tables Aliases Can Be Used Instead of Original Table Names.
- A Table Alias Gives an Alternate Name For The Existing Queried Table.
- Table Aliases Help in Keeping The SQL Code Smaller, Hence Using Less Memory.
- The Table Alias is Specified in the FROM Clause.

Guidelines:

- A Table Alias Can Be Up To 30 Characters in Length.
- If a Table Alias is Used For a Particular Table Name in The FROM Clause. Then That Table Alias Must be Substituted For The Table Name Through Out The SELECT Statement.
- A Table Alias Should Be Meaningful and Should Be Maintained as Short as Possible.
- A Table Alias is Valid Only for the Current SELECT Statement Only.

```
SQL> SELECT E.Empno, E.Ename, D.Deptno, D.Dname FROM Emp E, Dept D WHERE E.Deptno=D.Deptno;
```

```
SQL> SELECT E.Ename, E.Job, D.Deptno, D.Dname, D.Loc FROM Emp E, Dept D WHERE E.Deptno= D.Deptno AND E.Job IN(‘ANALYST', 'MANAGER' );
```

```
SQL> SELECT E.Ename, E.Job, D.Dname. D.Loc FROM Emp E, Dept D WHERE E.Deptno = D.Deptno AND D.Dname <> 'SALES';
```

Self Joins

- ✓ It is a Join of a Table to Itself.
- ✓ The Same Table Appears Twice in the FROM Clause And is Followed By Table Aliases.
- ✓ The Table Aliases Must Qualify the Column Names ill The Join Condition.
- ✓ To Perform a Self Join, Oracle Combines And Returns Rows of The Table That Satisfy The Join Condition.

Syntax:

```
SQL> SELECT Columns FROM Table T1, Table T2 WHERE T1.Column1 = T2.Column2
```

Illustrations

```
SQL> SELECT EI.Ename "Employee Name", E2. Ename "Managers Name" FROM Emp EI, Emp E2 WHERE E1.Mgr = E2.Empno;
```

```
SQL> SELECT E1.Ename||'''s Managers is'|| E2.Ename "Employees And Managers" FROM Emp E1, Emp E2 WHERE E1.Mgr = E2.Empno;
```

```
SQL> SELECT E1.Ename||'Works For'|| E2.Ename "Employees And Managers" FROM Emp E1 , Emp E2 WHERE(E1.Mgr =E2.Empno) AND E1.Job ='CLERK';
```

Cartesian Products

- ✓ The CARTESIAN PRODUCT is a Join Query, that Does Not Contains a Join Condition.
- ✓ During Caltesian Product Oracle Combines Each Row of One Table With Each Row of The Other.
- ✓ It Tends to Generate a Large Number of Rows And The Result is Rarely Useful.

```
SQL> SELECT Ename, Job, Dname FROM Emp, Dept;
```

Joining Data from More Than Two Table

- ✓ JOINS Can Be Established on More Than Two Tables.
- ✓ The Join is First Executed Upon The Two Most Relevant Tables And Then The Result is Applied Upon the Third Table.

```
SQL> SELECT Ename, Sal, Grade, Dept.Deptno, Dname FROM Emp JOIN Dept ON Emp.Deptno=Dept.Deptno JOIN SalGrade ON Emp.Sal BETWEEN LoSal AND hiSal;
```

Right Outer Join: Return all rows from the right table, and the matched rows from the left table(Oracle Returns NULL for un matched records)

SQL> SELECT Ename, Dept.Deptno, Dname, Loc FROM Emp RIGHT JOIN Dept ON Emp.Deptno = Dept.Deptno;

Left Outer Join: Return all rows from the left table, and the matched rows from the right table (Oracle Returns NULL for un matched records).

SQL> SELECT Enanie, Dept.Deptno, Dname, Loc FROM Emp LEFT JOIN Dept ON Emp.Deptllo = Dept.Deptno;

FULL Join: Combination of left outer join and right outer join. Return all rows when there is a match in ONE of the tables.

SQL> SELECT Ename, Dept.Deptno, Dname, Loc FROM Emp FULL JOIN Dept ON Emp.Deptno = Dept.Deptno

Sub Queries OR Nested Select OR Sub Select OR Inner Select

- ✓ A Sub Query Answers Multiple-Part Questions.
- ✓ A Sub Query in The WHERE Clause of a SELECT Statement is called as NESTED SUBQUERY.
- ✓ A Sub Query in the FROM Clause of a SELECT Statement is Called as INILINE VIEW.
- ✓ A Sub Query Can Be Part of a Column, in The SELECT List.
- ✓ A Sub Query Can Contain Another Sub Query.
- ✓ Oracle Imposes no Limit on the Number of Sub Query Levels in the FROM Clause of The Top-level Query.
- ✓ Within The WHERE Clause Up to 255 Sub Queries Can be nested.
- ✓ To Make The Statements Easier For Readability, Qualify The Columns in a Sub Query With The Table Name or Table Alias.

Purpose of A Sub Query

- ✓ To Define The Set of Rows To Be Inserted Into The Target Table of An INSERT or CREATE TABLE Statement.
- ✓ To Define The Set of Rows To Be Included in a View OR a Materialized View in a CREATE VIEW or CREATE MATERIALIZED VIEW Statement.
- ✓ To Define One or More Values To Be Assigned To Existing Rows in An UPDATE Statement.

Sub Query Principle:

- ✓ Solve a Problem By Combining The Two Queries, Placing One Query Inside The Other Query.

- The Inner Query or The Sub Query Returns a Value that is used by the Outer Query upon the Main Query.

Sub Query Usage:

They Are Practically Very Useful When We Need to SELECT ROWS From a Table With a Condition That Depends on The Data iil The Table Itself.

Syntax:

```
SQL> SELECT SelectList FROM TableName WHERE ColumnName Operator ( SELECT
SelectList FROM TableName );
```

- The Expressional Operators in Sub Queries Can Be Categorized into
- Single Row Operators -> =, <>, >, >=, <=
- Multiple Row Operators -> IN

Types of Sub Queries:

Single Row Sub Query:

- These Queries Return Only One Row From The Inner SELECT Statement.

Multiple Row Sub Query:

- These Queries Return More Than One Row From The Inner SELECT Statement.

Multiple Column Sub Query:

- These Queries Return More Than One Column From The Inner SELECT Statement

Guidelines to Follow:

- A Sub Query Must be Enclosed in Parenthesis.
- A Sub Query Must Appear on The Right Side of The Comparison Operator Only.
- Sub Queries Should Not Contain An ORDER BY CLAUSE.
- Only One ORDER BY Clause Can Be Implemented For The Total SELECT Statement.
- Two Classes of Comparison Operators Can be Used in Sub Queries They Are ...
- Single Row Operators.
- Multiple Row Operators.

Let Us Start With Single Row Sub queries:

```
SQL> SELECT Ename, Sal, Job FROM Emp WHERE Sal >(SELECT Sal FROM Emp WHERE Empno= 7566);
```

```
SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job (SELECT Job FROM Emp WHERE Ename = UPPER('smith')) ORDER BY Sal;
```

```
SQL> SELECT Empno, Ename, Hiredate, Sal FROM Emp WHERE Hiredate>(SELECT Hiredate FROM Emp WHERE Ename = 'TURNER') ORDERBY Sal;
```

```
SQL> SELECT Empno, Ename, Sal, Job FROM Emp WHERE Deptno (SELECT Deptno FROM Dept WHERE Dname 'SALES');
```

```
SQL> SELECT Empno, Ename, Sal, Comm, Sa) +NVL( Comm, 0 ) FROM Emp WHERE Deptno = (SELECT Deptno FROM Dept WHERE Loc = 'DALLAS');
```

Applying Group Functions in Sub Queries

- The Data From The Main Query Can Be Displayed By Using a Group Function in a Sub Query.
- As a Group Function Returns a Single Row, The Query Passes Through The Success State.
- The Inner Sub Query Should Not Have a GROUP BY Clause in This Scenario.

```
SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal =(SELECT MAX(Sal) FROM Emp;
```

```
SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal (SELECT MIN(Sal)FROM Emp);
```

```
SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal> (SELECT AVG(Sal)FROM Emp);
```

Applying HAVING Clause with Sub Queries

- A Sub Query Can Be Also Applied in HAVING Clause.
- The Oracle Server Executes The Sub Query, And The Results Are Returned Into The HAVING Clause of the Main Query.
- The Inner Query Need Not Use Any GROUP Functions in This Scenario.
- The Outer Queries HAYING Clause Contains GROUP Function.

```
SQL> SELECT Deptno, MIN(Sal) FROM Emp GROUP BY Deptno HAVING MIN(Sal) > (SELECT MIN(Sal) FROM Emp WHERE Deptno=20);
```

```
SQL> SELECT Job, AVG(Sal) FROM Emp GROUP BY Job HAVING AVG(Sal) (SELECT MIN(AVG(Sal) FROM Emp GROUP BY Job);
```

SQL> SELECT Job, A VG(Sal) FROM Emp GROUP BY Job HAVING AVG(Sal) < (SELECT MAX(AVG(Sal)) FROM Emp GROUP BY Job);

Sub Queries Returning More Than One Row

- ✓ The Sub Queries That Return More Than One RO\v Are Called as MULTIPLE ROW SUB QUERIES.
- ✓ In This Case a Multiple Row Operator Should Be Used.
- ✓ Tile Multiple Row Operators Expect One or More Values as Arguments.

SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal IN(SELECT MIN(Sal) FROM Emp GROUP BY Deptno);

SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal IN(SELECT MAX(Sal) FROM Emp GROUP BY Deptno);

SQL> SELECT Ename, Sal, Deptno , Job FROM Emp WHERE Sal IN(SELECT MAX(Sal) FROM Emp GROUP BY Job);

Sub Queries Returning Multiple Columns

In Sub Queries Multiple Columns Can Be Compared in The WHERE Clause, By Writing a Compound WHERE Clause Using Logical Operators.

- ✓ Multiple Column Sub Queries enable us to combine the Duplicate WHERE Condition into a Single WHERE CLAUSE.

Syntax

SQL> SELECT Column I, Column2, ... FROM TableName WHERE (Column a, Column b ,...) IN(SELECT Column a, Column b,.." FROM TableName , WHERE Condition).

- ✓ The Column Comparisons in a Multiple Column Sub Query Can Be
- ✓ Pair & Non Pair Wise Comparison.
- ✓ In Pair Wise Comparisons Each Candidate Row in The SELECT Statement Must Have Both The Same Values Associated With Each Column in The Group.

Pairwise Comparison OR Compound WHERE Clause Based SubQuery

SQL> SELECT OrdID, ProdlD, Qty FROM Item WHERE (ProdlD, Qty) IN(SELECT Prodld, Qty FROM Item WHERE OrdID=605) AND OrdlD < > 605;

Non Pairwise Comparision or Component WHERE Clause Based SubQuy

SQL> SELECT OrdlD, ProdID, Qty FROM Item WHERE ProdlD IN (SELECT ProdlD FROM Item WHERE OrdID=605) AND Qty IN (SELECT Qty FROM Item WHERE OrdID =605) AND OrdlD <> 605;

Handling NULL Values in Sub Queries:

- If One of The Values Returned By The Inner Query' is NULL Value, Then The Entire Query Returns NO ROWS.
- All CONDITIONS That Compare a NULL Value Result in a NULL
  SQL> SELECT E.Ename FROM Emp E WHERE E.Empno IN (SELECT M.Mgr FROM Emp M);

Applying Sub Query in From Clause

- A Sub Query in The From Clause is Equivalent To a View.
- The Sub Query in The From Clause Defines a Data Source For That Particular SELECT Statement And Only That SELECT Statement.

SQL> SELECT E.Ename, E.Sal , E.Deptno, El.SalAvg FROM Emp E, (SELECT Deptno, AVG(Sal) SalAvg FROM Emp GROUP' BY Deptno) E1 WHERE E.Deptno =E1.Deptno AND E.Sal > E1.Sal Avg;

SQL> SELECT E.EmpCount, D.DeptCount FROM (SELECT COUNT(*) EmpCount FROM Emp) E, (SELECT COUNT(*) DeptCount FROM Dept) D;

SQL> SELECT E.EmpCount, D.DeptCount, S.GradeCnt, E.EmpCount + D.DeptCount + S.GradeCnt TotalRecCnt FROM (SELECT COUNT(*) EmpCount FROM Emp) E, (SELECT COUNT(*) DeptCount FROM Dept) D, (SELECT COUNT(*) GradeCnt FROM SalGrade) S

Sub Select Statements

- These Are SELECT Statements Declared as Part of The SELECT List.

  SQL> SELECT Ename, Sal,(SELECT AVG(Sal) FROM Emp) "Organization Average" FROM Emp;

  SQL> SELECT Ename, Sal, (SELECT MAX(Sal) FROM Emp) "Organization Maximum", (SELECT MIN(Sal) FROM Emp) "Organization Minimum" FROM Emp;

Correlated Sub Queries

- It is Another Way of Performing Queries upon the Data with a Simulation of Joins.
- In This The Information From the Outer SELECT Statement Participates As a Condition in The

INNER SELECT Statement.

Syntax

SQL> SELECT SelectList FROM Table1 Alias1 WHERE Expr Operator (SELECT SelectList FROM Table2 Alias2 WHERE Alias1.Column OPERATOR Alias2.Column2);

Steps Performed

- ✓ First the Outer Query is executed.
- ✓ Passes The Qualified Column Value to the Inner Queries WHERE Clause.
- ✓ Then The Inner Query or Candidate Query is Executed, And The Result is Passed To The Outer Queries WHERE Clause.
- ✓ Depending on The Supplied Value The Condition is Qualified For The Specific Record.
- ✓ Successful Presented Else Suppressed From Display

SQL> SELECT Deptno, Dname FROM Dept D WHERE EXISTS (SELECT *FROM Emp E WHERE D.Deptno = E.Deptno);

SQL> SELECT Deptno, Dname FROM Dept D WHERE NOT EXISTS (SELECT * FROM Emp E WHERE D.Deptno = E.Deptno);

SQL> SELECT E.Ename FROM Emp E WHERE EXISTS (SELECT * FROM Emp E1 WHERE E1.Empno= E.MGR);

SQL> SELECT E.Ename FROM Emp E WHERE NOT EXISTS (SELECT * FROM Emp E1 WHERE E1.Empno = E.MGR);

EXISTS Condition

The SQL EXISTS condition is used in combination with a sub query and is considered to be met, if the sub query returns at least one row. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

WHERE EXISTS (subquery);

The subquery is a SELECT statement. If the subquery returns at least one record in its result set, the EXISTS clause will evaluate to true and the EXISTS condition will be met. If the subquery does not return any records, the EXISTS clause will evaluate to false and the EXISTS condition will not be met.

Note

SQL statements that use the EXISTS condition are very inefficient since the sub-query is rerun for EVERY row in the outer query's table. There are more efficient ways to write most queries, that do not use the EXISTS condition.

Pseudo Column

- ✓ Pseudo Columns Behave Like a Table Column, But is Not Actually Stored in a Table.
- ✓ Upon Pseudo Columns Only SELECT Statements Can Be Implemented, But INSERT, UPDATE or DELETE Cannot be Implemented.
- ✓ ROWID
- ✓ ROWNUM

ROWNUM Pseudo Column

For Each Row Returned By a Query, The ROWNUM Pseudo Column Returns a Number Indicating The Order in Which Oracle Selects The Rows

The First Row Selected Has a ROWNUM of 1, The Second Has 2. And So On..

Conditions Testing For ROWNUM Values Greater Than a Positive Integer Are Always FALSE.

```
SQL> SELECT LPAD(‘’ , ROWNUM, '*' ) FROM Emp;

SQL> SELECT ROWNUM, Ename, Sal FROM Emp;

SQL> SELECT * FROM (SELECT * FROM Emp ORDER BY Sal DESC) WHERE ROWNUM < 6;
```

ROWID Pseudo Column

- ✓ This Pseudo Column Returns a ROW's Address For Each Row Stored in The Database.
- ✓ ROWID Values Contain Information Necessary To Locate a The Physical Area of The Data Base Row.
- ✓ The Row Belongs To Which Data Block in the Data File.
- ✓ The Row Belongs To Which Row in The Data Block (First Row is 0)
- ✓ The Row Belongs To Which Data File (First File is 1)

Uses of ROWID Values

- ✓ ROWID is The Fastest Means of Accessing a Single Row From Data Base.
- ✓ ROWID Can Show How a Tables Row's Are Physically Stored.
- ✓ ROWID's Are UNIQUE Identifiers For a Row in a Table.

- A RowID Can Never Change During The Life Time of Its Row.

When a Row is DELETED, ORACLE May Reassign Its ROWID To a New Row That is Inserted.

- The ROWID Can Never Be INSERTED, UPDATED and DELETED Manually.
- The ROWID Pseudo Column Can Be Used in SELECT and WHERE Clauses.

```
SQL> SELECT ROWID, Ename, Job FROM EmpWHER deptno=20;
```

SET Operators

- These Operators Are Used to Combine Information of Similar DATA Type From One or More Than One Table.
- DATA Type of The Corresponding Columns in All The SELECT Statements Should Be Same.

The Different Types of SET Operators Are

- UNION Operator.
- INTERSECT Operator.
- UNION ALL Operator.
- MINUS Operator.
- SET Operators Can Combine Two or More Queries Into One Result.
- The Result of Each SELECT Statement Can Be Treated As a Set, and SQL Set Operations Can Be Applied on Those Sets To Arrive At a Final Result.
- SQL Statements Containing SET Operators Are Referred To As Compound Queries, And Each SELECT Statement in a Compound Query is Referred To As a Component Query.
- Set Operators Are Often Called Vertical Joins, As The Result Combines Data From Two or More SELECTS Based on' Columns Instead of Rows.

The Generic Syntax

```
<component query> {UNION IUNION ALL IMINUS I INTERSECT} <component query>;
```

UNION

- Combines The Results of Two SELECT Statements Into One Result Set, And Then Eliminates Any Duplicate Rows From That Result Set.

UNION ALL

- Combines The Results of Two SELECT Statements Into One Result Set Including The Duplicates. INTERSECT " Returns Only Those Rows That Are Returned By Each of Two SELECT Statements.

MINUS

- Takes The Result Set of One SELECT Statement, And Removes Those Rows That Are Also Returned By a Second SELECT Statement point of concentration
- The Queries Are All Executed Independently But Their Output is merged,
- Only Final Query Ends With a Semicolon.

Rules and Restrictions

- The Result Sets of Both The Queries Must Have the Same Number of Columns.
- The Data Type of Each Column in The Second Result Set Must Match The Data Type of Its Corresponding Column in The First Result Set.
- The Two SELECT Statements May Not Contain An ORDER BY Clause, The Final Result of The Entire SET Operation Can Be Ordered.
- The Columns used For Ordering Must Be Defined Through The Column Number.

Illustrations

```
SQL> SELECT Empno, Enarne FROM Emp WHERE Deplno=10 UNION SELECT Empno,
Ename FROM Emp WHERE Deptno=30 ORDER BY 1;
```

```
SQL> SELECT Empno,Ename, Job FROM Emp WHERE Deptno= (SELECT Deptno FROM
Dept WHERE Dname='SALES')
```

UNION

```
SELECT Empno,Ename, Job FROM Emp WHERE Deptno (SELECT Deptno FROM Dept WHERE
Dname= 'ACCOUNTING') ORDER BY 1;
```

```
SQL> SELECT Empno, Ename FROM Emp WHERE Deptno=10 UNION ALL SELECT
Empno, Ename FROM Emp WHERE Deptno =30 ORDER BY 1;
```

INTERSECT

```
SELECT Empno, Ename FROM Emp WHERE Deptno= 30 intersect SELECT Empno,
Ename FROM Emp WHERE Deptno = 10
```

MINUS

SELECT Empno, Ename FROM Emp WHERE Deptno= 30 minus SELECT Job FROMEmp WHERE Deptno =20

UNION

SELECT Job FROM Emp WHERE Deptno= 30 union SELECT Job FROM Emp WHERE Deptno =20

UNION ALL

SELECT Job FROM Emp WHERE Deptno= 30 union all SELECT Job FROM Emp WHERE Deptno=20

INTERSECT

SELECT Job FROM Emp WHERE Deptno =30 intersect SELECT Job FROM Emp WHERE Deptno =20

MINUS

SELECT Job FROM Emp WHERE Deptno =10 SELECT ROWNUM, Ename FROM Emp WHERE ROWNUM < 7 minus SELECT ROWNUM, Ename FROM Emp WHERE ROWNUM < 6;

Views

- ✓ It is A Logical Table Based on One OR More Tables OR Views.
- ✓ A View in Practicality Contains No Data By Itself.
- ✓ The Tables Upon Which A View is Based Are Called As BASE TABLES.

Simple Views

SQL> CREATE VIEW Employees AS SELECT Empno "ID Number", Ename Name, Sal "Basic Salary", Job Designation FROM Emp;

Selecting Data from A View

SQL> SELECT Name, Designation FROM Employees;

SQL> SELECT "ID Number", Name, "Basic Salary" * 12 FROM Employees;

DECODE Function

- ✓ It Is A Single Row Function.
- ✓ The Function Works On The Same Principle As The If -Then -Else.
- ✓ We Can Pass A Variable Number Of Values Into The Call Of The DECODEO Function.
- ✓ The First Item is Always the Name of the Column That Need to Be Decoded.
- ✓ Once All Value-Substitute Pairs Have Been Defined, We Can Optionally Specify A Default Value.

Syntax

SQL> SELECT DECODE(ColumnName, Value 1, Substitutel, Value 2, Substitute2, ... Return Defau]t FROM TableName;

- ✓ The Function Has No Restriction on The INPUT And OUTPUT Data Type.
- ✓ It is The Most Power full Function in Oracle.
- ✓ The Function Can Work For only an Analysis That Considers an Equality Operator in The Logical Comparison.

SQL> SELECT Ename, Job, Sal, DECODE(Deptno, 10, 'ACCOUNTING', 20, 'RESEARCH', 30, 'SALES', 40, 'OPERATIONS', 'OTHER') Departments FROM Emp ORDER BY Departments;

Working with CASE expressions

- ✓ The CASE Expression Can Be Used To Perform If-Then-Else Logic in SQL.
- ✓ CASE is Similar To DECODE But It is ANSI-Compliant.
- ✓ It Can be Used Even For Executing Conditions on range Based Comparison.
- ✓ Case Expressions Are of Two Types
- ✓ SIMPLE CASE Expressions
- ✓ SEARCHED CASE Expressions.

Simple CASE Expressions

- ✓ These Expressions Are Used To Determine The Returned Value.
- ✓ They Work With Equality Comparison Only, Almost All Similar To DECODE.
- ✓ It Has A Selector Which Associates To The Compared Value Either From The Column or Constant.
- ✓ The Value in The Selector is Used For Comparison With The Expressions Used in The WHEN Clause.

Syntax

SQL> CASE Search_Expr WHEN Expr1 THEN Result.l WHEN Expr2 THEN Result2 ELSE Default Result END

Illustration

SQL> SELECT Ename, Deptno, CASE Deptno WHEN lO THEN' ACCOUNTS' WHEN 20 THEN 'RESEARCH' WHEN 30 THEN 'SALES' WHEN 40 THEN 'OPERATIONS' ELSE 'NOT FOUND' EN D FROM Emp;

Searched CASE Expressions

- ✓ The Statement Uses Conditions To Determine The Returned Value.
- ✓ It Helps in Writing Multiple Conditions For Evaluation.
- ✓ Helps in Range Analysis of Values Also.

Syntax

SQL> CASE WHEN Condition 1 THEN Result 1 WHEN Condition 2 THEN Result 2 WHEN Condition n THEN Resultn ELSE DefaultResult END

iIlustration

SQL> SELECT Ename, Deptno, CASE WHEN Deptno = 10 THEN' ACCOUNTING' WHEN Deptno = 20 THEN 'RESEARCH' WHEN Deptno=30 THEN 'SALES' WHEN Deptno=40 THEN 'OPERATIONS' ELSE 'Not Specified' END FROM Emp;

SQL> SELECT Ename, Sal, CASE WHEN Sal >= 800 AND Sal <= 2000 THEN 'LOWEST PAY' WHEN Sal >= 2001 AND Sal <= 4000 THEN 'MODERATE PAY' ELSE 'HIGH PAY' END FROM Emp;

Analytic Functions:

Ranking Function:

- ✓ They Enable US To Calculate Ranks, Percentiles

Normal Ranking

SQL> SELECT EName,Deptno, Sal, RANK() OvER(ORDER BY Sal) EmpRank FROM Emp GROUP BY Deptno, EName, Sal ORDER By Emprank;

SQL> SELECT ename,Deptno, Sal, DENSE_RANK() OYER(ORDER BY Sal DESC) EmpRank FROm Emp GROUP BY Deptno, EName, Sal ORDER BY EmpRank Ranking With Partition.

SQL> SELECT ENall1', Deptno, Sal, RAN() OvER(PARTITION BY DeptNo ORDER BY Sal DESC) "TOP Sal" FROM Emp ORDER BY Deptno, Sal DESC;

```
SQL> SELECT ename, DeptNo, Sal, DENSE_RANKO OYER(PARTITION BY Deptno ORDER BY Sal DESC) "TOP Sal" FROM Emp ORDER BY DeptNo, Sal DESC;
```

Ranking With Partition and Filters

```
SQL> SELECT * FROm (SELECT Ename, Deptno, Sal, RANK() OvER(PARTlTION BY DeptNo ORDER BY Sal DESC) "TOP Sal" FROM Emp) WHERE "TOP Sal" <=3 ORDER BY DeptNo, Sal DESC;
```

Data updating And Deletion

Updating The Data in A Table .

- ✓ The UPDATE Statement is Used To Change The Existing Values in A Table OR in The Base Table of View.

Syntax:

```
SQL> UPDATE <Table_Name> SET <Specification> WHERE Clause;

SQL> UPDATE EMP SET Comm = NULL WHERE Job 'CLERK';
```

DELETE Statement

- ✓ It is Used To Remove Rows from

Syntax

```
SQL> DELETE [FROM] <Table_Name> [WHERE Condition];

SQL> DELETE FROM EmpWHERE Empno=7864;

SQL> DELETE FROM Emp WHERE Deptno 20;

SQL> DELETE FROM Emp WHERE Deptno (SELECT Deptno FROM Dept WHERE Dname = 'SALES');
```

Transaction Control

- ✓ Oracle Server Ensures Data Consistency Based Upon Transactions.
- ✓ Transactions Consist of DML Statements That Make Up One Consistent Change To The Data. Transaction Start And End Cases

- ✓ A Transaction Begins When The First Executable SQL Statement is Encountered.
- ✓ The Transaction Terminates When The Following Specifications Occur.
- ✓ A COMMIT OR ROLLBACK is Issued.
- ✓ A DDL Statement Issued.
- ✓ COMMIT
- ✓ It Ends The Current Transaction By Making All Pending Data Changes Permanent.

ROLLBACK [To Savepoint Name]

- ✓ It Ends The Current Transaction By Discarding All Pending Data Changes.

Alerting the Table Definition

Syntax for Adding Column

SQL> ALTER TABLE <Table_name> ADD (ColumnName DataType [DEFAULT Exp], ColumnName DataType] ...);

Syntax for Modifying Column

SQL> ALTER TABLE <TableName> MODIFY (ColumnName DataType [DEFAULT Exp], Column DataType] ...);

Adding A Column To A Table

- ✓ The ADD Clause is Used To Add Columns For An Existing Table. SQL> ALTER TABLE Dept30 ADD (Job V ARCHAR2(9));

Guidelines For ADDING Column

- ✓ A Column Can Be ADDED OR MODIFIED But Cannot Be Dropped From A Table.
- ✓ We Cannot Specify The Location Where The Column Can Appear, It By Default Becomes The Last Column
- ✓ If The Table Contains Records, Before The Column is Added, The New Column Contains NULL Values.

Modifying A Column

- ✓ A Column Data Type, Size And Default Value Can Be Changed.
- ✓ A Change To The Default Value Affects Only Subsequent Insertions To The Table.

Guidelines To Modify A Column

- ✓ We Can Increase The Width OR Precision of A Numeric Column.

- ✓ We Can Decrease The Width of A Column If The Column Contains Only NULL Values And If The Table Has No Rows.
- ✓ We Can Change The Data Type If The Column Contains NULL's.
- ✓ We Can Convert A CHAR Column To The VARCHAR2 Data Type OR Convert A V ARCHAR2 Column To The CHAR Data Type If The Column Contains NULL Values OR If The Size is Not Changed.

Dropping A Column

- ✓ A Column Can Be Dropped From A Table By Using The ALTER TABLE Statement.
- ✓ The DROP COLUMN Clause is Used For This Purpose And The Feature is Enabled From Oracle 8i Onwards.

Guidelines To Drop A Column

- ✓ The Column May OR May Not Contain Data.
- ✓ The Table Must Have At Least One Column Remaining in It After It is Altered.
- ✓ Once A Column is Dropped It Cannot Be Recovered.

```
SQL> ALTER TABLE Dept30 DROP Column Job;
```

Dropping A Table

- ✓ It Removes The Definition of The ORACLE TABLE
- ✓ The Command Not Only Drops The TABLE But The ENTIRE DATABASE is Lost Along With. The ASSOCIATED INDEXES.

Syntax

```
SQL> DROP TABLE <tableName> [CASCADE CONSTRAINTS];

SQL> DROP TABLE Dept30 CASCADE CONSTRAINTS;
```

Changing the Name of An Object

- ✓ The RENAME Command Can Be Used To Change The Name of A
- ✓ TABLE
- ✓ VIEW

Syntax

```
SQL> RENAME <OldName> TO <NewName>;
```

Illustration

```
SQL> RENAME Dept TO Department;
```

Truncating a Table

- ✓ It Is Used To Remove All Rows From A TABLE And To Release The STORAGE SPACE Used By The Specific TABLE.
- ✓ The TRUNCATE TABLE Will Not Facilitate For ROLLBACK.

Syntax

```
SQL> TRUNCATE TABLE <TableName>;
```

Illustration

```
SQL> TRUNCATE TABLE Department;
```

Advanced Table Creation Strategies

Creating A Table From An Existing Table ON The FLY Tables

- ✓ Oracle Allows The Creation of A New Table On-The-Fly; Depending on A SELECT Statement on An Already Existing Table.

Syntax

```
SQL> CREATE TABLE <TableName> AS SELECT Columns FROM TableName; [WHERE
Condition];
```

Creating an Exact Copy

```
SQL> CREATE TABLE SampDept AS SELECT * FROM Dept;
```

Creating An Exact Copy With Different Column Names

```
SQL> CREATE TABLE SampDeptl (DeptID, DeptName, Place) AS SELECT * FROM Dept;
```

Creating A Copy With Required Columns

```
SQL> CREATE TABLE SampDept3 AS SELECT DeptNo, Dname FROM Dept;
```

Creating A Copy With Required Columns

```
SQL> CREATE TABLE SampDept3(DeptiD, DeptName ) AS SELECT DeptNo, Dname
FROM Dept;
```

Creating a Copy Without Data

```
SQL> CREATE TABLE Sampdept3 AS SELECT * FROM Dept WHERE ‘apple’=APPLE’.
```

Database Testing

Agenda of the session

1. Explain what is Database Testing?
2. Explain differences between Frontend Testing and DB Testing?
3. Explain Importance of Data Layer Testing in Web Application Architecture?
4. How to understand data before performing database Testing?
5. Explain various DB objects needs to be tested in database?
6. Explain about Tables you have tested in the project?
7. Explain recently tested DB requirements in your project?
8. Explain recently created Test cases in the in the project?
   - 8.1 Null check
   - 8.2 Count check
   - 8.3 Metadata check
   - 8.4 Data Integrity check
   - 8.5 Data Quality
   - 8.6 Duplicate Values check
9. Explain how you executed DB Test cases and report recently identified defects in the project?
10. DB Testing checklist or Guidelines to be followed in the project?
11. DB Testing Interview Questions?

WHAT IS DATABASE TESTING?

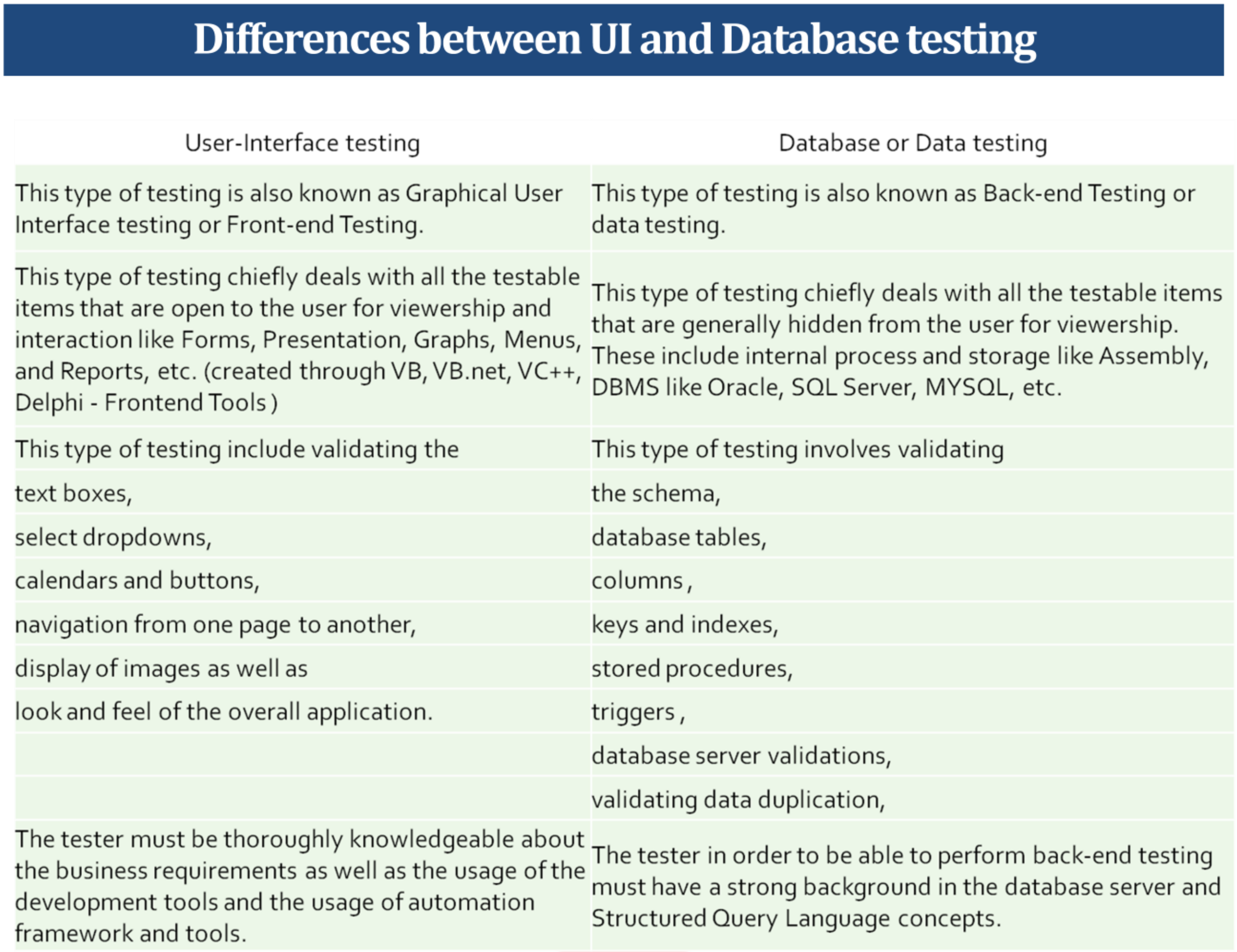
- Computer applications (front ends) are more complex these days. The more complex the front ends, the back ends are even more complicated. So, it is all the more important to learn about DB testing and be able to validate the databases effectively to ensure **‘secure and quality database’**.

- Backend testing mainly includes testing the integration between the application and the database.
- It is more like checking whether the changes made in the database gets reflected in the front end application.

  For example: consider a new column is been added in the table. We can test this by providing values in the front end application and check whether they are stored in the table (backend database).
- It's basically testing data while travelling from **front to back end** or **back end to front end** or **back end to back end** only.

  Type 1 DB Testing: Application to DB
  Type 2 DB Testing: DB to DB
  Type 3 Testing: DB to Application
- It also involves testing the application from the logical storage of the data. Validating the storage data with the UI.
- Front end testing mainly focuses on testing the application from the user perspective, it consists of functionality, usability, and GUI .It is very likely that many tests in a front end only hit a small portion of a backend.
- A backend is the engine of any client/server system. A bug in a backend may raise a serious impact on the entire system. This includes deadlock or data corruption or data loss and bad performance. Too many bugs in a backend will cost tremendous resources to find and fix bugs and delay the system developments.
- Many bugs can be effectively found and fixed in the early development stage

Explain differences between Frontend Testing and DB Testing?



Differences between UI and Database testing

User-Interface testing	Database or Data testing
This type of testing is also known as Graphical User Interface testing or Front-end Testing.	This type of testing is also known as Back-end Testing or data testing.
This type of testing chiefly deals with all the testable items that are open to the user for viewership and interaction like Forms, Presentation, Graphs, Menus, and Reports, etc. (created through VB, VB.net, VC++, Delphi - Frontend Tools)	This type of testing chiefly deals with all the testable items that are generally hidden from the user for viewership. These include internal process and storage like Assembly, DBMS like Oracle, SQL Server, MYSQL, etc.
This type of testing include validating the	This type of testing involves validating
text boxes,	the schema,
select dropdowns,	database tables,
calendars and buttons,	columns ,
navigation from one page to another,	keys and indexes,
display of images as well as	stored procedures,
look and feel of the overall application.	triggers ,
	database server validations,
	validating data duplication,
The tester must be thoroughly knowledgeable about the business requirements as well as the usage of the development tools and the usage of automation framework and tools.	The tester in order to be able to perform back-end testing must have a strong background in the database server and Structured Query Language concepts.

Database schema: A database schema is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects.

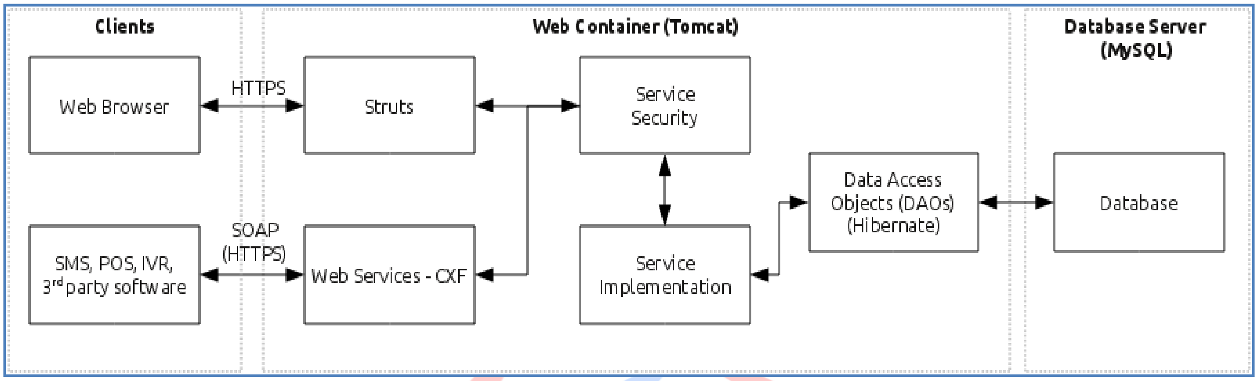
Explain Importance of Data Layer Testing in Web Application Architecture?

Web Application Architecture/3 Tier Architecture:

- ✓ Typically comprise a presentation layer, a business or data access layer, and a data layer. Three layers in the three tier architecture are as follows.
  1. Presentation / Client Layer
  2. Business Logic Layer
  3. Data Layer
- ✓ **UI/Presentation layer** :
  - ➢ Represents UI part of Application.
  - ➢ Where the data is presented to the user or input is taken from user.
  - ➢ Ex: Registration form, labels, Buttons etc.
- ✓ **Business Logic layer:**

  - Validation of Data
  - Logic calculations and implementation
  - Acts as interface b/w Presentation and Data layer

- ✓ **Data layer:**
  - Database connection
  - Store and retrieve data.



How to understand data before performing database Testing?

- Applications are used by end users and they enter a group of raw data
- This data is later collated and used by management to arrive at meaningful information
- Before we first understand the technical aspects of database, we must understand the business data clearly
- Rule 1: In any application, first identify raw data
- Rule 2: Group related data and associate data type and size (summary and detail)
- Rule 3: Create a set of samples for each of these groups for better clarity
- Rule 4: Identify the relationship between the data

Let us take railways reservation as the application

- The raw data could be
- Passenger/customer name
- Age
- Date of journey
- Train name
- From station
- To Station
- PNR number
- Route codes

- Stations covered in the route

The data groups would be

Train related data – train number, train name, route in which it is running

Station related data – station code, station name, station RMS Pincode, station type (junction, station etc)

Ticket related data – PNR Number, passenger name, date of journey, seat or berth, age, from station, to station, train code etc

Unique data means a data that does not repeat itself

- Station code is unique across country
- Train code is unique across country
- PNR number is unique across country
- Coach number is unique within a train, but not unique across system
- Seat number is unique within a coach and train
- Passenger name is not unique
- Ticket price is not unique

Relationship between data:

- One PNR number is associated with one train code
- One PNR number is associated with one or more passenger names
- One train code is associated with one or more stations
- One train code is associated with one route code
- All relationships will fall under one-to-one, one-to-many, many-to-one
- Many-to-many is a combinations of the above
- We need to identify the relationships between the data to understand clearly the dependency between data

Explain various DB objects to be tested in database?

- A physical database installation in a machine has the following logical entities
  - Database (group of tables)
  - Tables (that contain data)
  - Views
  - Index
  - Triggers
  - Stored procedures

Explain Tables you have tested in the project?

Important Tables of Cyclos Application

1. Groups
2. Members
3. Ads
4. Loans

DB Testing checklist or Guidelines to be followed in the project?

	Database Testing Checklist
#	**Check Point**
1)	**Data Integrity**
1	Is the complete data in the database is stored in tables
2	Is the data stored in tables is correct
3	Is there any unnecessary data present
4	Is the data present in the correct table
5	Is the data present in correct field within the table
6	Is the data stored correct with respect to Front End updated data
7	Is LTRIM and RTRIM performed on data before inserting data into database
2)	**Field Validations**
1	Is 'Allow Null' condition removed at database level for mandatory fields on UI
2	Is 'Null' as value not allowed in database
4	Is the Field length specified on UI same as field length specified in table to store same element from UI into database.
5	Is the Data type of each field is as per specifications
3)	**Constraints**
1	Is required Primary key constraints are created on the Tables
2	Is required Foreign key constraints are created on the Tables
3	Are valid references are done for foreign key
4	Is Data type of Primary key and the corresponding foreign key same in two tables
5	Does Primary key's 'Allow Null' condition not allowed
4)	**Stored Procedures/ Functions**
1	Is proper coding conventions followed
2	Is proper handling done for exceptions
3	Are all conditions/loops covered by input data
4	Does TRIM is applied when data is fetched from Database
5	Does executing the Stored Procedure manually gives the correct result
6	Does executing the Stored Procedure manually updates the table fields as expected
7	Does execution of the Stored Procedure fires the required triggers
8	Are all the Stored Procedures/Functions used by the application (i.e. no unused stored procedures present)
5)	**Triggers**
1	Is proper coding conventions followed in Triggers

2	Are the triggers executed for the respective DML transactions
3	Does the trigger updates the data correctly once executed
6)	**Indexes**
1	Are required Clustered indexes created on the tables
2	Are required Non Clustered indexes created on the tables
7)	**Transactions**
1	Are the transactions done correct
2	Is the data committed if the transaction is successfully executed
3	Is the data rollbacked if the transaction is not executed successfully
4	Is the data rollbacked if the transaction is not executed successfully and multiple Databases are invlolved in the transaction
8)	**Security**
1	Is the data secured from unauthorized access
2	Are different user roles created with different permissions
3	Do all the users have access on Database
9)	**Performance**
1	Does Database perform as expected (within expected time) when query is executed for less number of records
2	Does Database perform as expected (within expected time) when query is executed for large number of records
3	Does Database perform as expected (within expected time) when multiple users access same data
4	Is Performance Profiling done
5	Is Performance Benchmarking done
6	Is Query Execution Plan created
7	Is Database testing done when server is behind Load Balancer
8	Is Database normalized

DB Testing Cyclos Project:

Functional Information:

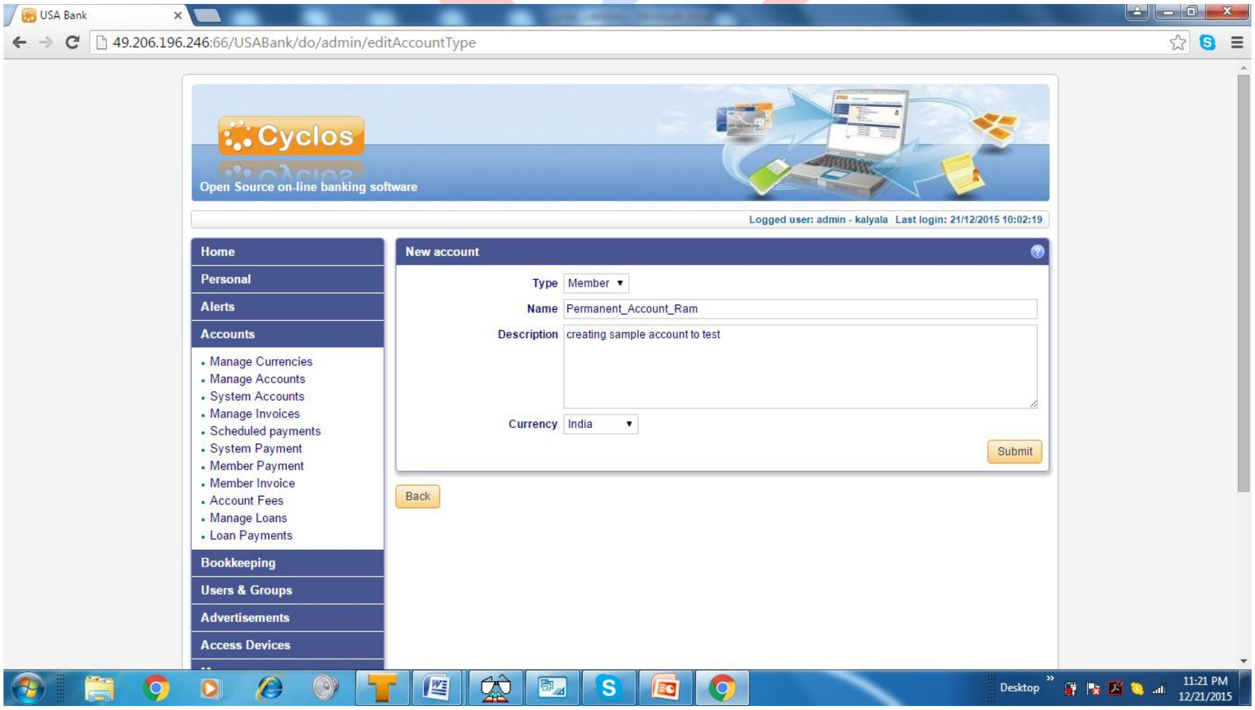
- ✓ Create new group
- ✓ Update Group information
- ✓ Create new member
- ✓ Update member information by admin
- ✓ Update member profile by member
- ✓ Apply for ATM Card
- ✓ Loan Grant
- ✓ Loan Repayment
- ✓ Loan cancelled

- ✓ Money transfer from admin to member
- ✓ Money transfer member to member
- ✓ Creating ad
- ✓ Updating ad information
- ✓ Delete ad by admin
- ✓ Set credit limit by Admin

1. Open an account → new customer
    a. Change account address (front end and back end)
2. Apply for ATM card
3. Loan grant
    a. Loan payment
    b. Check loan payment status and his account balance

CREATING NEW ACCOUNT:

FROM LEFT PANE CLICK ON "MANAGE ACCOUNS" and then enter required details to create a new account, then click on submit.



49.206.196.246:66/USABank/do/admin/editAccountType

Logged user: admin - kalyala Last login: 21/12/2015 10:02:19

New account

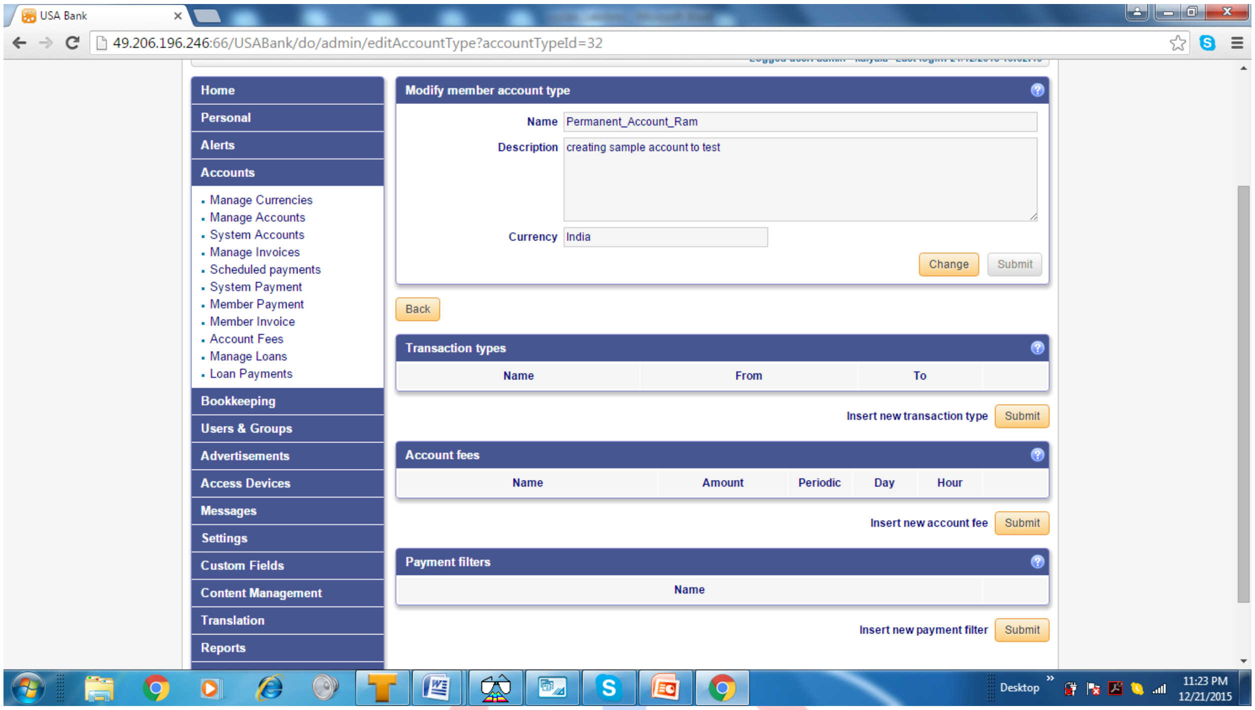
Type: Member

Name: Permanent_Account_Ram

Description: creating sample account to test

Currency: India

You could see below screen once submit the new account.



Modify member account type

Name: Permanent_Account_Ram

Description: creating sample account to test

Currency: India

Transaction types

Name	From	To

Account fees

Name	Amount	Periodic	Day	Hour

Payment filters

Name

Checking new account created in backend or not?

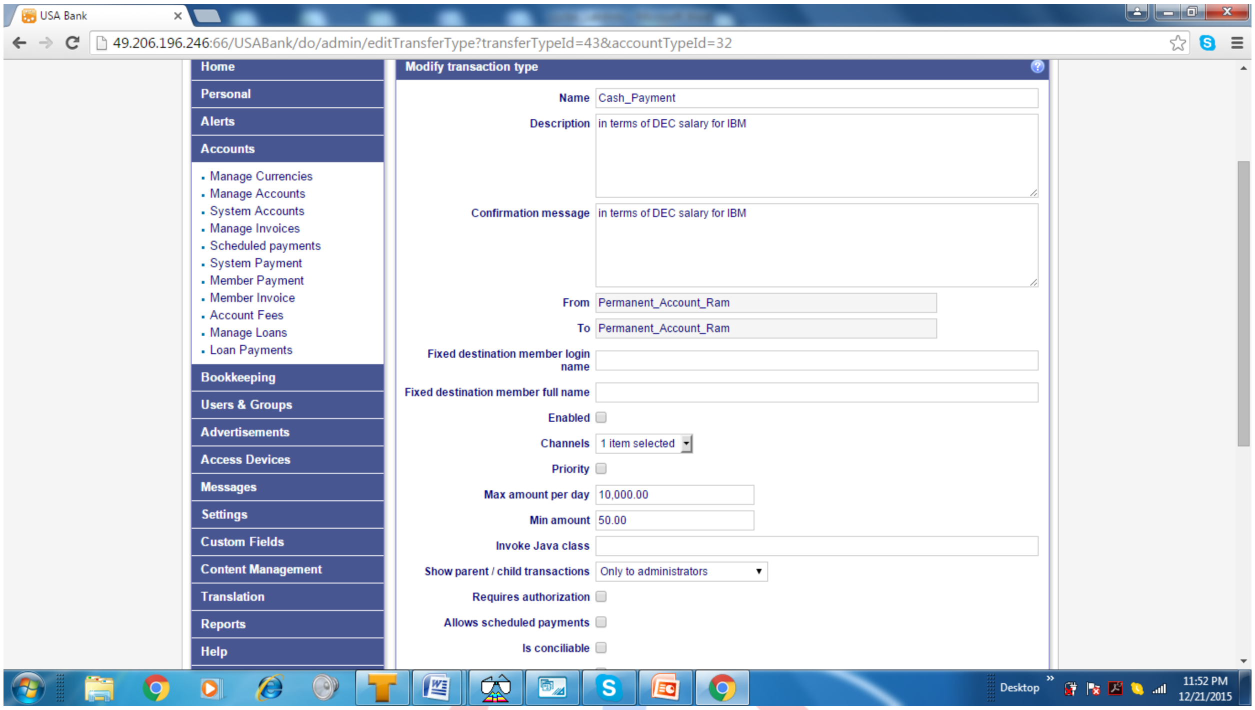
Use below queries:

select * from currencies where name='India' and id=12

select * from account_types where currency_id in (select id from currencies)

 and name='Permanent_Account_Ram'

Insert transaction type.



Modify transaction type

Name: Cash_Payment

Description: in terms of DEC salary for IBM

Confirmation message: in terms of DEC salary for IBM

From: Permanent_Account_Ram

To: Permanent_Account_Ram

Fixed destination member login name:

Fixed destination member full name:

Enabled ☐

Channels: 1 item selected

Priority ☐

Max amount per day: 10,000.00

Min amount: 50.00

Invoke Java class:

Show parent / child transactions: Only to administrators

Requires authorization ☐

Allows scheduled payments ☐

Is conciliable ☐

Check with below query

select * from transfer_types where from_account_type_id=32

DB Testing Requirements:

TID	T. Description	Pre condition	Test Steps	Test data	Expected Result	Actual Result	Testing status	Defect ID
T1	To Test group creation in Groups table	Tester must have access to DB	Step 1: Create group with group name	"2016ICICIcustomers"	"2016ICICIcustomers" must be created			

			as "XXXX'					
			Step 2: Connect to qtpdb01 schema		Connected to qtpdb01 schema			
			Step 3: select * from groups where name='new member group' and id=44		This query must return a row which contanis New group details in Groups table			
T2	To test card info for the group	Card must be assigned to group	Step 1: Silver card assigned to "New Group"	Silver card				
			Step 2: Connect to qtpdb01 schema					
			Step 3: select * from groups where id=44 and card_type_id =3		The value given in front end must match in card_types table and groups table			
T3	To Test meta data data for groups table	Groups table must be created	Step 1: Connect to qtpdb01 schema					
			Step 2: compare table name		show create table groups			
			Step 3: compare column name					
			Step 4: Compare data types					

			Step 5: Compare constraints					
			Step 6: Compare indexes		SHOW INDEX FROM groups			
			Step 7: Compare column Length					
T4	To perform null check on primary key column	Tester must have access to DB	Step 1: Connect to qtpdb01 schema					
			Step 2: select id from groups where id is null		zero records must be populated			
T5	Duplicate check on primary key column	Tester must have access to DB	Step 1: Connect to qtpdb01 schema					
			Step 2: select id,count(*) from groups group by id having count(*)>1		Zero records must be populated			
T6	To Test Valid values check	Tester must have access to DB	Step 1: Connect to qtpdb01 schema					
			Step 2:select distinct transaction_password_length, sms_show_free_threshold from groups		one row must be displayed with(valid values i.e 4,50)			

T7	To update data in groups table '**description**' column	Tester must have write permission to DB	Step 1: Connect to qtpdb01 schema					
			Step 2: update groups set description= 'DB Testing' where id=43;		Updated value must be reflected in front end			
T8	To test the indexes defined on groups table	Tester must have access to DB	Step 1: Connect to qtpdb01 schema		User should be able to connect to DB			
			Step 2: check for the index information, use below query. SHOW INDEX FROM groups		indexes should get display.			
T9	To do orphan check based on primary key and forien key	Tester must have access to DB	Step 1: Connect to qtpdb01 schema		All the values in child table must be in rimaty table card_types-groups			
			Step 2:		index must be created as per requirement			
T10	To compare front end data with groups table	Tester must have access to DB	Step 1: Create group with group name as "New Group"					
			Step 2: Connect to					

			qtpdb01 schema					
			Step 3: select * from groups where name='new member group'					
			Step 4: Compare data in step 1 with data in step 3		Data must match			

DB Testing Check List:

S.No	Type of Testing	Checklist	Explanation

A	Performance Testing Benchmark testing		When a system does not have data problems or user interface bugs, system performance will get much attention. The bad system performance can be found in benchmark testing. Four issues must be included:
1		• System level performance	
2		Major functionality (Pick up most-likely-used functions/features)	
3		· Timing and statistics (Minimal time, maximal time and average time)	
4		· Access volume (A large number of machines and sessions must be involved.)	
B	***Test a back end via a front end***	Sometimes back end bugs can be found by front end testing, especially data problem. The following are minimum test cases:	
5		· Make queries from a front end and issue the searches (It hits SELECT statements or query procedures in a back end)	
6		Pick up an existing record, change values in some fields and save the record. (It involves UPDATE statement or update stored procedures, update triggers.)	
7		Push FILE - NEW menu item or the NEW button in a front end window. Fill in information and save the record. (It involves INSERT statements or insertion stored procedures, deletion triggers.)	
8		Pick up an existing record, click on the DELETE or REMOVE button, and confirm the deletion. (It involves DELETE statement or deletion stored procedures, deletion triggers.)	
9		Repeat the first three test cases with invalid data and see how the back end handles them.	

C	***Login and user security***		
10		Simulate front end login procedure and check if a user with correct login information can login	
11		· Simulate front end login procedure and check if a user with incorrect login information fail to login	
12		· Check concurrent logins (make many users login at the same time.)	
13		· Try to login when a time-consuming query is running to see how long login will take to succeed	
14		· Check for any security-restrict functions and see they are working properly	
15		· See any data view restriction in place, such as, a user can see his data and the data of people who report to him.	
D	***Checking data integrity and consistency***		
16		Data validation before insertion, updating and deletion.	
17		· Check major columns in each table and see if any weird data exist. (Nonprintable characters in name field, negative percentage, and negative number of phone calls per month, empty product and so on)	
18		· Generate inconsistent data and insert them into relevant tables and see if any failure occurs	
19		· Try to insert a child data before inserting its parent's data.	
20		· Try to delete a record that is still referenced by data in other table	
21		· If a data in a table is updated, check whether other relevant data is updated as well.	
E	***Test functions and features***		
22		· For updating functions, make sure data is updated following application rules	
23		· For insertion functions, make sure data is inserted following application rules	

24		· See if error messages are clear and right.	
25		· Find out time-consuming features and provide suggestions to developers	

DB Testing Test cases:

T ID	T Description	Test Steps	Expected Result	Table_Name	Column_Name
T1	To Test Member creation	Step 1: Create member in fron end with Member 'Full Name"	Member created with Full Name in Fron end	members, Accounts, Users	
		Step 2 : select * from members where name=Full Name	Step 2 in Backend must match with Step 1		
T2	T0 update member full name in front end				
T3	To update member email Id in DB	update members set email='liyakhatghori20091@gmail.com' where id=169; update members set email='zzzz_01.gmail.com' where id=170			

DB Testing Interview Questions:

1. Define join and name different type of joins?
2. What is the syntax to add record to a table?
3. How do you add a column to a table?
4. What is the syntax to add record to a table?
5. How do you add a column to a table?
6. Define SQL Delete statement.
7. Define COMMIT?
8. What is a primary key?
9. What are foreign keys?
10. What is CHECK Constraint?
11. Is it possible for a table to have more than one foreign key?

12. What are the possible values for BOOLEAN data field.
13. What is a stored procedure?
14. What is identity in SQL?
15. What is Normalization?
16. What is Trigger?
17. How to select random rows from a table?
18. Write a SQL SELECT query that only returns each name only once from a table?
19. Explain DML and DDL?
20. Can we rename a column in the output of SQL query?
21. Give the order of SQL SELECT?
22. Difference between TRUNCATE, DELETE and DROP commands?
23. What do you mean by ROWID?
24. Define UNION, MINUS, UNION ALL, INTERSECT?
25. What is difference between UNIQUE and PRIMARY KEY constraints?
26. What is a composite primary key?
27. What is an Index?
28. What is the Subquery?
29. What is Referential Integrity?
30. What is Case Function?
31. How we can avoid duplicating records in a query?
32. Explain the difference between Rename and Alias?
33. What is a View?
34. What are the advantages of Views?
35. Do View contain Data?
36. Can a View based on another View?
37. What is difference between Having clause and Where clause?
38. What is Database testing?
39. Why database testing is important?
40. In the Database Testing process, what do we usually check?
41. How to test database procedures and triggers?
42. What is the database trigger, how to verify the trigger is fired or not and can you invoke trigger on demand?
43. After entering the data from the front-end application interface, how do you test whether a database in updated or not?
44. How to test the Stored Procedures?
45. What do you mean by DML?
46. What do you mean by DCL commands and explain the types of commands used by DCL?
47. How to write a query to get the second largest value from a given column of a table?

48. How to write a query to get 10thhighest salary from an employee table?
49. While testing stored procedures what are the steps does a tester takes?
50. How to test database manually?

==

- What is Database testing?
  - Database testing is assessing the quality of the database which is connected to a GUI application. It is also called Back end testing.
- What are the exact roles and responsibilities of a database tester?
  - Role of a tester is to test the features of the database like:
    - Data Retrieval
    - Data Validity
    - Data Security
    - Data Performance
- What is Data redundancy?
  - Data Redundancy is repetition or duplication of data
  - It may occur either if a field is repeated in two or more tables or if the field is repeated within the table.
  - Disadvantages Of Data Redundancy
    - Increases the size of the database unnecessarily.
    - Causes data inconsistency.
    - Decreases efficiency of database.
    - May cause data corruption.
- What is your approach to back end testing?
  - Back end testing is performed by comparing the data at the back end vs front end(i.e the UI of the application)
  - We prepare and execute SQL queries to retrieve the data from the back end in the same format as it appears in the front end to compare between the two.
- What is a primary key and foreign key?
  - A primary key is a column (or columns) in a table that uniquely identifies the rows in that table.
  - A foreign key is a field in a relational table that matches the primary key column of another table
- What is referential integrity?
  - Referential integrity is a database concept that ensures that relationships between tables remain consistent. When one table has a foreign key to another table, the concept of referential integrity states that you may not add a record to the table that contains the foreign key unless there is a corresponding record in the linked table. It also includes the techniques known as cascading

update and cascading delete, which ensure that changes made to the linked table are reflected in the primary table.

- What is a constraint and what are the different types of constraints?
  Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.
  - NOT NULL Constraint: Ensures that a column cannot have NULL value.
  - DEFAULT Constraint: Specifies a default value for a column when none is specified.
  - UNIQUE Constraint: Ensures that all values in a column are different.
  - PRIMARY Key: A combination of NOT NULL and UNIQUE .Uniquely identifies each rows/records in a database table.
  - FOREIGN Key: Uniquely identifies a rows/records in any another database table. Also ensures the referential integrity of the data in one table to match values in another table
  - CHECK Constraint: Ensures that all values in a column satisfy certain conditions.
  - INDEX: Use to create and retrieve data from the database very quickly.
- What is wrong with this SQL query? Correct it so it executes properly.

```
SELECT Id, YEAR(BillingDate) AS BillingYear
FROM Invoices
WHERE BillingYear >= 2010;
SELECT Id, YEAR(BillingDate) AS BillingYear
FROM Invoices
WHERE YEAR(BillingDate)  >= 2010;
```

- Given a table SALARIES, such as the one below, that has m = male and f = femalevalues. Swap all f and m values (i.e., change all f values to m and vice versa) with a single update query and no intermediate temp table.

Id	Name	Sex	Salary
1	A	m	2500
2	B	f	1500
3	C	m	5500
4	D	f	500

```
UPDATE SALARIES SET sex = CASE sex WHEN 'm' THEN 'f' ELSE 'm' END
```

- Write a SQL query to find the 10th highest employee salary from an Employee table. Explain your answer.

```
SELECT TOP (1) Salary FROM
(
SELECT DISTINCT TOP (10) Salary FROM Employee ORDER BY Salary DESC
) AS Emp ORDER BY Salary
```

- Write a SQL query to list out the departments which have a max salary of more than 10 lakhs

  EMP(EMPID, NAME, DEPTID, SAL)
  DEPT (DEPTID,DEPTNAME)

  ```
  Select DEPTNAME, MAX(SAL)
  From EMP E
  JOIN DEPT D
  ON E.DEPTID=D.DEPTID
  Group by DEPTID
  Having max(sal) > 1000000
  ```

- How can you create an empty table from an existing table?

  ```
  Select * into studentcopy from student where 1=2
  ```

- Write a sql query to get the highest mark in each subject

  Students (StudentName, Subject, Marks)

  ```
  Select Subject, max(Marks)
  From Students
  Group by Subject
  ```

- Write a sql query to get the 2nd highest mark in each subject

  Students(StudentName, Subject, Marks)

  ```
  SELECT max(marks)
  FROM Student
  GROUP BY subject
   WHERE marks NOT IN (SELECT max(marks) FROM Student GROUP BY subject);
  ```

- Get all employees names and their corresponding managers names

  Employee (empid, name, mgrid)

  ```
  Select E1.Name,E2.Name
  From Emp E1
  Join Emp E2
  On E1.Mgrid=E2.Empid
  ```

MOBILE APPLICATION TESTING:

Brief history of Mobile:

In our day to day life mobile is an essential one and we are not able to move without mobile. So Mobile is a more important thing in our life ... Is not it? Now a day in addition to telephony, modern mobile phones also support a wide variety of other services such as text messaging, MMS, email, Internet access and short-range wireless communications (infrared, Bluetooth).

Do you know who produce a handheld mobile phone first in the market?

Here is the Answer:

Motorola and Bell Labs raced to be the first to produce a handheld mobile phone. That race ended **on 3 April 1973 when Martin Cooper**, a Motorola researcher and executive, made the first mobile telephone call from handheld subscriber equipment, placing a call to Dr. Joel S. Engel. The prototype handheld phone used by Dr. Martin Cooper weighed 2.5 pounds and measured 9 inches long, 5 inches deep and 1.75 inches wide. The prototype offered a talk time of just 30 minutes and took 10 hours to re-charge.

Here we can see cooper with his first hand held phone.

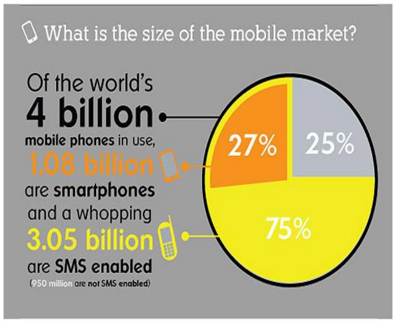
Let's say thanks to Martin cooper for producing the first handheld device.

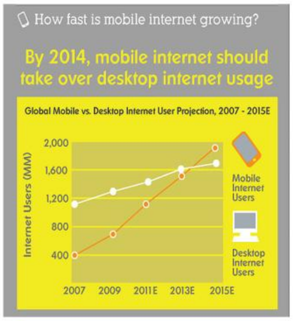
From Copper's first handheld mobile phone to current iPhone5, we have so many models of phones with wide verity of features . In below mentioned diagram we can able to see different models available in Market .

We all know that the growth of the mobile market is huge throughout the globe and constantly reshaping how we interact in our everyday lives.

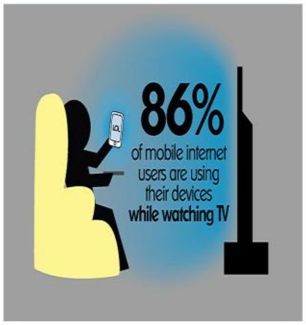
A few stand out stats of Mobile Market:

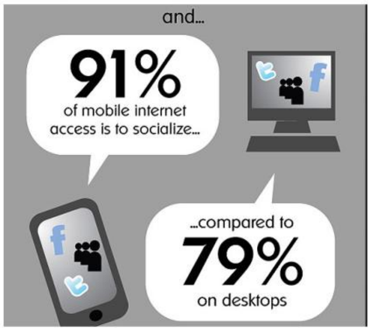
- ✓ In the twenty years from 1990 to 2010, worldwide mobile phone subscriptions grew from 12.4 million to over 4.6 billion.
- ✓ There are currently **6 Billion** mobile subscribers worldwide
- ✓ This equals **87%** of the world's population
- ✓ **China** and **India** account for **30%** of this growth
- ✓ There are over **1.2 Billion** people accessing the web from their mobiles
- ✓ Over **300,000 apps** have been developed in the **past 3 years**
- ✓ Google earns **2.5 Billion** in mobile ad revenue annually
- ✓

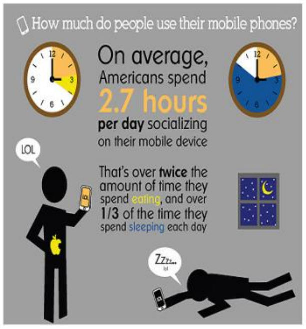


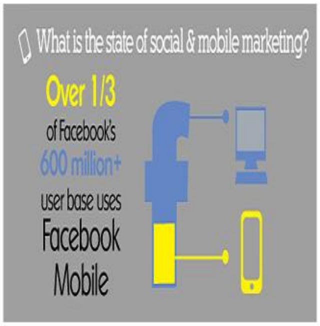


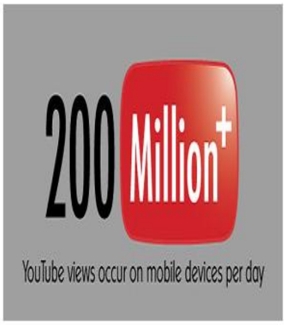
How has mobile usability changed?
Because of the increasing use of mobiles, their versatility is always growing...
One half of all local searches are performed on mobile devices
$$$
Mobile tags provide more product information like comparison shopping offers than traditional barcodes


86%
of mobile internet users are using their devices while watching TV



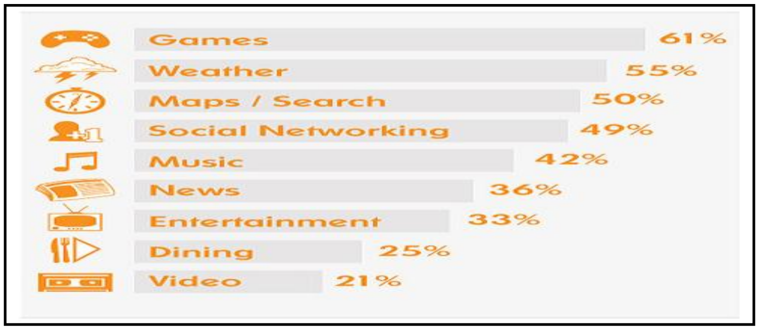

How much do people use their mobile phones?
On average, Americans spend
2.7 hours
per day socializing on their mobile device
That's over twice the amount of time they spend eating, and over 1/3 of the time they spend sleeping each day
LOL
Zzz...





By looking at above stastics we can understand howmuch fastly mobile market growing day by day.

In below mentioned diagram we can able to see different categories of mobile applications and their percentages of market share.



Types of Mobile Devices

1) Touch Mobiles

Types of Touch Mobiles

Single Touch Device

Single action is performed on touching anywhere on the screen.

Next action is not performed until the action is released

Multi Touch Device

Multiple actions are performed on touching on various areas on screen.

Virtual Keypad Device

Though the device is Touch screen, the actions need to be performed using Virtual key pad.

2) Non Touch Mobiles

Types of Non Touch Mobiles

Normal Key Pad

Contains normal key pad with NUM PAD

QWERTY Key Pad

All the Alphabets are displayed in QWERTY Model

AZERTY Key Pad

All the alphabets are displayed with AZERTY Model.

Note: The French version of the standard QWERTY keyboard. AZERTY keyboards differ from the QWERTY keyboard in that the **Q** and **W** keys have been interchanged with the **A** and **Z** keys. Another difference between QWERTY and AZERTY keyboard is that the M key on an AZERTY is to the left of the **L** key.

Mobile Application Testing Basics

What is Mobile application?

A mobile application is a software that runs on a mobile device such as a cell phone or MP3 player that will allow the device to perform specific tasks that are typically restricted to PCs. It also known as downloadable, mobile application are common on most phones, including inexpensive, entry level models.

Types of Mobile Application?

Mobile applications are classified into three types. They are

- **Browser Based application**
- **Pre-installed applications**
- **Installed applications**

Browser Based applications:

These browser based mobile applications are similar to web applications, to work with browser based mobile applications we require a mobile browser, URL of the application and Internet connectivity. Browse the application URL to get the content from server and check the application functionality, look and feel and performance of the application on different types of mobile devices with the help of different types of mobile browser.

Characteristics of the browser based applications:

- Application builds for only mobile devices.

- Can be accessed by entering the specific URL in mobile browser.

- No need to installation and UN-installation.

-No involvement of upgrade.

Critical areas for Browser based application:

-Browser based application always expect the connectivity(Internet connection).

-Speed and coverage are the critical aspect.

-Cache related issues.

Pre Installed Application:

These applications shipped along with the mobile device, no need of installation of these applications. User is able to use these applications but not able to remove or uninstall preinstalled or default application.

Characteristics of the pre installed applications

-Application which are shipped as in built software with device

-No download involved

-No installation and un installation involved

-Automatic upgrade can be done

Critical areas for Pre Installed application

-Prototype testing is very critical for such applications

-Core database affecting directly due to such application crashes

-They cannot be uninstall or deleted ever

-Crashes can cause several damage to ROM

Installable Application:

User can able to download these type of applications from market places or install through cables or OTA services. User have full control over the installable applications. User is able to download, install, Un install or upgrade.

Characteristics of the installable applications are:

-Application executable file can be received by the wireless media and wired media.

- Can be Downloaded from stores/Installed/uninstalled/upgraded

Critical areas for Installable application:

- Installation and UN installation of application in device

- Can be transferred via wireless media like Bluetooth, infra etc

Installable applications further classified into three types. They are

1) Native applications

2) Internet based applications

3) Hybrid applications

Native applications:

These application's do not require any internet connection and so many games and utility applications are comes under this category.

Games, notepad, Spread sheets..etc

Internet based applications:

These application's required internet connectivity continuously, These applications should be useful whenever user having proper internet connectivity. These applications are useless when there is no internet connectivity. All banking and financial applications, online casinos , Instant Messengers, Social networking sites are best example for internet based applications.

EX: Face book, Skype, Gmail..etc

Hybrid applications:

These applications are using internet services and perform some tasks and rest of the tasks will also be done in the absence of internet services .

EX: Ever note, NFS2, Temple run

Based on the installation we can classify above installable applications into two types.

1) Applications only installable on phone memory

We cannot move these type of application into SD card. Based on the phone memory availability only user able to download or install the applications.

2) Applications installable on SD card

These applications can be moved to SD card after installation by using third party tools.

Mobile applications categories:

Based on the mobile application purpose we categorize the mobile application into below mentioned categories.

- ✓ Communications
- ✓ Games
- ✓ Multimedia
- ✓ Productivity
- ✓ Travel
- ✓ Utilities

Communications:

These types of applications establishing a connection between two people.

Examples: Email Clients, IM Clients, Mobile Web and Internet Browsers, News/Information Clients, Social Network Clients.

Games:

- Puzzle/Strategy (e.g., Tetris, Sudoku, Mahjong, Chess, Board Games)
- Cards/Casino (e.g., Solitaire, Blackjack, Roulette, Poker)
- Action/Adventure (e.g., Doom, Pirates of the Caribbean, Role Playing Games)
- Sports (e.g., Football, Soccer, Tennis, Basketball, Racing, Boxing, Skiing)
- Leisure Sports (e.g., Bowling, Pool, Darts, Fishing, Air Hockey)

Multi Media:

- Graphics/Image Viewers
- Presentation Viewers

- Video Players
- Audio Players
- Streaming Players (Audio/Video)

Productivity:

- Calendars
- Calculators
- Diary
- Notepad/Memo/Word Processors
- Spreadsheets
- Directory Services (e.g., yellow pages)
- Banking/Finance

Travel:

- City Guides
- Currency Converters
- Translators
- GPS/Maps
- Itineraries/Schedules
- Weather

Utilities:

- Profile Manager
- Idle Screen/Screen Savers
- Address Book
- Task Manager
- Call Manager

What is mobile application testing?

Mobile application testing is the process of testing the functionality, usability and consistency of an application for hand-held mobile devices, from pre-installed or can-be-installed mobile application software platforms like iTunes, Google Play, etc.

Why Mobile Apps Need Testing?

For any mobile app developer hoping to produce a top quality mobile application, app testing is an essential part of the app development process to identify hidden defects.

Several reasons for getting your application tested by a mobile app testing professional before its consumer release:

1. **To Check the Basic User Experience:**

   After designing and developing a mobile app you will need it to be tested by a group of eager mobile users. This simply requires the application to be test run in its simplest form – fully using the app for its intended purpose. Users at this testing stage should be asked to give feedback on the complete user experience and record any glitches they discover.

2. **To Test Navigation:**

   Whilst basic user testing may bring awareness to navigation problems. This process will check all menu functions are correctly working and that both internal and external links are accurate.

3. **To Test System and Negative Usage:**

   By performing app tests, we can accurately determine how application will function in various conditions. Testing the apps reactions to system changes such as **low memory** or **low battery** as well as putting the application up against negative challenges such as malicious attacks

4. **To Check for Hidden Defects:**

   If all is well with the general user experience of your app, there could still be hidden issues that could cause sporadic performance or later problems. These defects are found through both software and hardware tests and are only completely detectable through professional services.

5. **To Check Connectivity:**

   Monitoring how a mobile app functions in conditions of **low internet connectivity/ low mobile signal** is a very important stage in mobile app testing and will ensure that any problems formed during app development can be corrected before release.

6. **To Test Audio Functionality:**

Another area which needs to be tested is the apps ability to interact with various audio settings on different handsets. App details including audio and vibrate feedback (when a sound or buzz plays on a touch) also need to be thoroughly checked to eliminate any future glitches.

Challenges in Mobile Application Testing:

Mobile app marketing is growing now very fast, but in testing field number of challenges arrives due to variation in handsets, phone carriers, networks supported and application written in different languages. Today, mobile applications deliver complex functionalities on different platforms that have limited resources for computing. Mobile applications can either be standalone applications or web based mobile applications

1. Many types of mobile devices
2. Different types of mobile platforms/operating systems from various companies
3. Different mobile carriers
4. Many types of mobile screens

More and more devices, More versions of operating systems, More screen sizes, GPRS/CDMA/2G/3G/4G/WIFI/Bluetooth …Etc.

Screen Size – Screen size is one of the biggest limitations for mobile devices. The screen size varies from 128 X 128 to 800 X 480. Smaller screens have a **portrait orientation** and larger screens have a **landscape orientation**. Phones that can change their orientation – meaning they are capable of working in both landscape and portrait modes. 240 x 320 is the overall dominant screen size so far. Small screen resolutions tend to makes web pages almost illegible.

Display Resolutions – Different mobiles have different resolutions. A low resolution can degrade the quality of multimedia displayed on the screen of mobile device.

Processing capability - Limitations in processing speed and memory size of mobile devices is a major issue faced during testing. Phones sometimes allow only a single active process in them. Your applications may fail if something like opera-mini is running in the background and you are trying to communicate out.

Diversified platform and devices – An application might work in one phone model and may not in the very next model from the same company. Just because you've tested in one phone model does not mean it will work on other available models. The jar size limits on these phone models can vary. Some popular models don't really allow big downloads. Space is always at a premium so you will have to be at a constant lookout for the download size of your application and flag it when it increases.

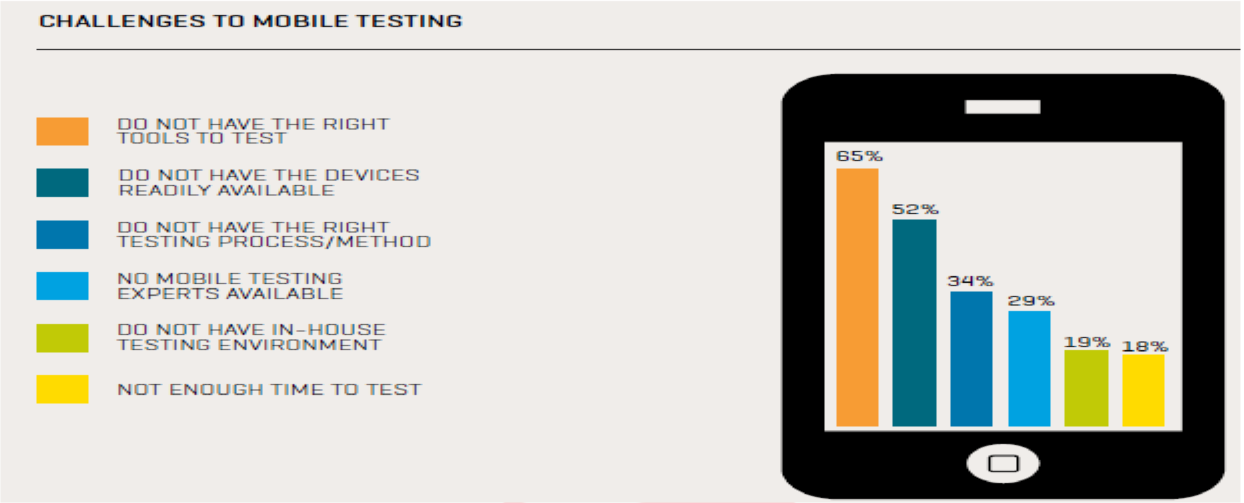
Type of content – Content delivery bothers both mobile user and carrier. For example, when page size is large, the device will not be able to handle it. Page should be cropped or re-sized and delivered without losing relevant information. The page can also be divided into fragments and displayed along multiple pages. Scrolling to read documents or web page will not be user friendly.

Connectivity – Network connectivity largely decides the data download time and also the quality of streaming media. Slow and unreliable wireless network connection with low bandwidth is a common hindrance for mobile applications. Carrier or device should be GPRS, Edge, 3G or 4G compatible.

Data Input Methods – Mobile devices come in two flavors – soft keyboards/touch screens and physical keyboards. Small buttons and labels limit the user's effectiveness and efficiency in entering data. This results in slowing down the input speeds and increasing the chances of error.

Device screen flip capabilities – Page layout or content will change when user access page in different screen modes.

Usability factors - User friendliness, self explanatory features, browser compatibility, help, etc



Few Important Areas to Consider While Testing the Mobile applications

1) Testing in different **Network speeds:**

1. Low 2. Medium 3. High

2) Testing during change of network speed

1. Low to high 2. High to low

3) Testing in different **Network types**

1. 2G (GPRS, CDMA, EDGE) 2. 3G

3. Wi-Fi 4. Different types of plan based on the service provider

4) Testing in different **Battery strategy**

1.Critical 2. Low 3. During charging 4.High

5) Monitoring the **Battery consumption** pattern

1. Observe the battery consumptions where application is running in the background.
2. Observe the battery consumptions where application is running in the foreground.
3. Run the Application for long time.

6) Monitoring on the **Memory consumption** patterns

1. observes the memory use during launching the applications
2. While application running the app in background
3. While application running the app in foreground
4. During exit the applications.
5. Run the application for long time

Types Of Mobile Application Testing on Smart Phones

1. Installation testing
2. Requirement Functionality Testing
3. Widget Testing
4. Phone Interrupt testing
5. Stress testing
6. UI testing
7. Compatibility Test
8. Interoperability testing
9. Recover Testing
10. Orientation testing
11. Certification Compliance Testing
12. **Submission Guidelines Compliance Testing**

1. Installation testing:

Here need focus on the application installation and un installation behavior on the device from market place or other resources like through air or download from website.

2. Requirement Functionality Testing:

Test the application functionality, behavior of the application according to its design test cases, against its requirement.

3. **Widget testing:**

Test the widget by adding widget, using widget as per functionality i.e. music widget should play music.

4. **Phone Interrupt testing:**

Test the application behavior on receiving/dialing the phone call, SMS & other notifications.

5. **Stress testing:**

Test the application's performance to generate the different events simultaneously of different frequencies**.** Behavior of Mobile Application in **Low resources** (Memory/Space), Behavior of mobile website when many mobile users simultaneously access
mobile website

6. **UI testing:**

All mobile platforms have certain submission guidelines to follow before the application can be available.

7. **Compatibility Test:**

Developers do the unit testing on the emulators. Testers need to test on actual environment i.e. devices. There are multiple devices of different resolutions & having various OS. We need to check for compatibility on maximum devices.

8. **Interoperability testing:**

Verifying whether our mobile application is co-existence with other applications in the mobile or not? Our application should not break any other application's functionalities and our application should not be blocked by any existing application. Mainly we should test interoperability testing with respect to Antivirus applications and utilities applications.

9. **Recovery Testing:**

Recovery testing is the activity of testing how well an application is able to recover from **crashes**, hardware failures and other similar problems. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. Recovery testing is basically done in order to check how fast and better the application can recover against any type of crash or hardware failure etc. Type or extent of recovery is specified in the requirement specifications. It is basically testing how well a system recovers from crashes, hardware failures.

Examples of recovery testing:

1. While an application is running, suddenly restart the Mobile device, and afterwards check the validness of the application's data integrity.
2. While an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application's ability to continue receiving data from the point at which the network connection disappeared.
3. Restart the mobile while a browser has a definite number of sessions. Afterwards, check that the browser is able to recover all of them.

10. Orientation testing:

Unlike in a desktop application, the user can rotate the screen through portrait and landscape orientations, both before and while your application is being used. This can be a great thing since your application can take advantage of the different screen dimensions to provide specialized UI but it also means you need to make some decisions about how your app is going to handle these situations. Verifying the application behavior in various orientations and changing the orientation while working with the application and verify the functionality and UI of the application. Orientations change should not cause any damage to the application UI or functionality.

11. **Certification Compliance Testing:**

For downloadable mobile applications, there are various Third party Mobile Quality Certification program for various platforms. True Brew Testing (for BREW Apps), Java Verified program (for J2ME apps), Symbian Signed Test Criteria (for Symbian Apps) are some examples. Apart from regular functional testing, you may need to test your application against the test cases/Testing criteria provided by these certification processes. However, it depends on your client, whether they want to certify their application or not.

12. Submission Guidelines Compliance Testing:

The application needs to adhere to the specified submission guidelines to publish it in any mobile application store. Failure to meet these guidelines may result in rejection of your app on mobile application stores. For example failure to comply with application Submission guidelines for Apple App Store may result in rejection of your app in Apple app store.

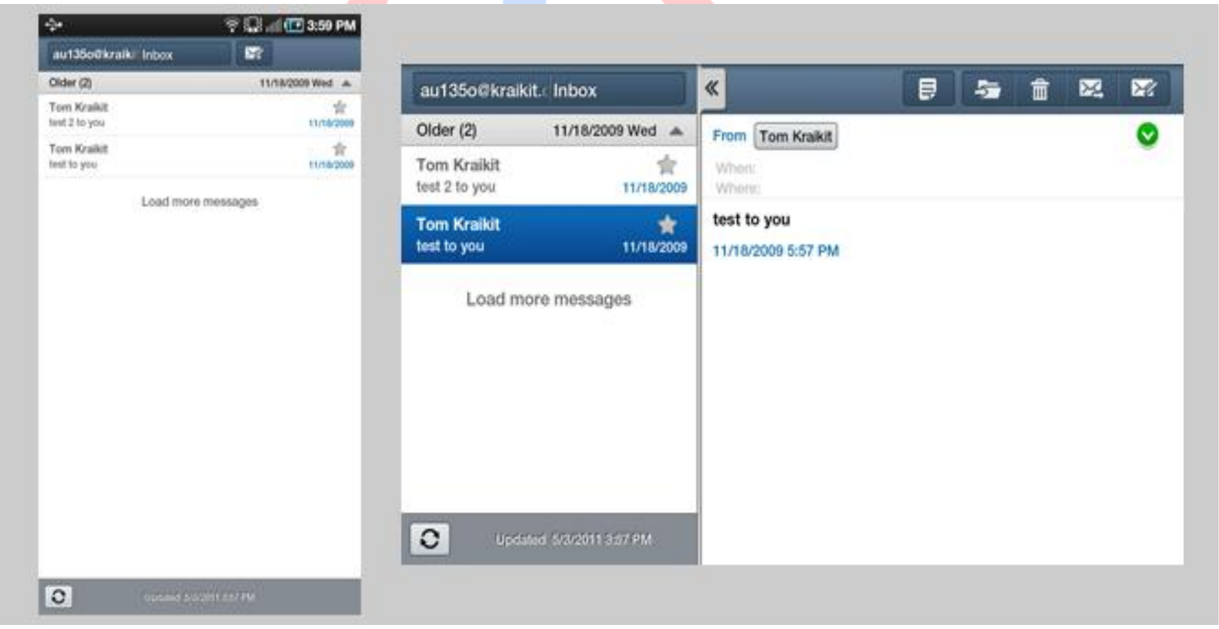
Mobile Orientations Help Guide:

Unlike in a desktop application, the user can rotate the screen through portrait and landscape orientations, both before and while your application is being used. This can be a great thing since your application can take advantage of the different screen dimensions to provide specialized UI but it also means you need to make some decisions about how your app is going to handle these situations.

- **Portrait**: mode where the display is taller than it is wide

- **Landscape**: mode where the display is wider than it is tall

iPad, in its default portrait orientation; Motorola Xoom, by default in landscape orientation



Single-pane layout in portrait orientation; dual-pane layout in landscape orientation

Android

Android is the world's most popular mobile platform. With Android you can use all the Google apps you know and love, plus there are more than 600,000 apps and games available on Google Play to keep you entertained, alongside millions of songs and books, and thousands of movies. Android devices are already smart, and will only get smarter, with new features you won't find on any other platform, letting you focus on what's important and putting you in control of your mobile experience.

Android is a Linux-based operating system[12] designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005,[13] Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.[14] The first Android-powered phone was sold in October 2008

Company / developer	Google Open Handset Alliance Android Open Source Project
Programmed in	C, C++, Java
OS family	Unix-like
Working state	Current
Source model	Open source
Initial release	September 23, 2008
Latest stable release	4.2.2 Jelly Bean / February 11, 2013; 3 months ago
Marketing target	Smartphone Tablet computers
Available language**(s)**	Multi-lingual
Package manager	Google Play, APK
Supported platforms	ARM, MIPS, x86, I.MX
Kernel **type**	Monolithic (modified Linux kernel)
Default user interface	Graphical (Multi-touch)
License	Apache License 2.0 Linux kernel patches under GNU GPL v2
Official website	www.android.com

Version	**Code name**	**Release date**	API **level**	**Distribution (March 4, 2013)**
4.2.x	Jelly Bean	November 13, 2012	17	2.3%
4.1.x	Jelly Bean	July 9, 2012	16	26.1%
4.0.x	Ice Cream	December 16, 2011	15	27.5%

	Sandwich			
3.2	Honeycomb	July 15, 2011	13	0.1%
3.1	Honeycomb	May 10, 2011	12	
2.3.32.3.7	Gingerbread	February 9, 2011	10	38.4%
2.3–2.3.2	Gingerbread	December 6, 2010	9	0.1%
2.2	Froyo	May 20, 2010	8	3.7%
2.0–2.1	Eclair	October 26, 2009	7	1.7%
1.6	Donut	September 15, 2009	4	0.1%
1.5	Cupcake	April 30, 2009	3	

Android 1.0 and Android 1.1

In 23rd of September 2008, the first commercial version of Android 1.0 was released. On 9th February 2009 the Android 1.1 was introduced. The first Android 1.0 device is HTC Dream.

Features of Android 1.0:

- ✓ You can download and update the android market application through the market apps.
- ✓ You can change the camera resolution, quality and white balance etc.
- ✓ Access to web email servers.
- ✓ Gmail, Google Contacts, Google Calendar, Google Maps ,Google search ,Google Talk and Google Sync is also available
- ✓ Instant messaging, text messaging and MMS.
- ✓ Other basic features also available in this version.

Android 1.5 Cupcake:

Cupcake was the first major overhaul of the Android OS. The Android 1.5 SDK was released in April 2009 and brought along plenty of UI changes, the biggest probably being support for widgets and folders on the home screens.

There were plenty of changes behind the scenes, too. Cupcake brought features like improved Bluetooth support, camcorder functions, and new upload services like YouTube and Picasa.

Android 1.5 ushered in the era of the modern Android phone, and the explosion of devices included favorites like the HTC Hero/Eris, the Samsung Moment, and the Motorola Cliq.

Features of cupcake:

- ✓ Animated screen transition
- ✓ Auto- rotation option
- ✓ You can upload videos and photos in you tube and Picasa
- ✓ Copy and paste options in web browsers.

- ✓ Video recording and play back
- ✓ Third party key board and text prediction for support.

Android 1.6 Donut

Donut, released in September 2009, built on the features that came with Android 1.5, and expanded them. While not very rich in the eye-candy department, Android 1.6 made some major improvements behind the scenes, and provided the framework base for the amazing features to come. To the end user, the two biggest changes would have to be the improvements to the Android Market, and universal search.

Behind the screen, Donut brought support for higher resolution touch screens, much improved camera and gallery support, and perhaps most importantly, native support for Sprint and Verizon phones. Without the technology in Android 1.6, there would be no Motorola Droid X or HTC Evo 4G.

Donut features are:

- ✓ Support for WVGA screen resolution
- ✓ Fully integrated camera, Gallery and Cam coder, with faster camera access
- ✓ Voice and text entry search
- ✓ Updated technology support for VPNs, 802.1x.
- ✓ You can delete multiple photos at a time

Android 2.0 and Android 2.1 Eclair

Eclair was a pretty major step up over its predecessors. Introduced in late 2009, Android 2.0 first appeared on the Motorola Droid, bringing improvements in the browser, Google Maps, and a new user interface. Google Maps Navigation also was born in Android 2.0, quickly bringing the platform on par with other stand-along GPS navigation systems. HTC's Desire and Legend phones launched with Android 2.1 later in the year, touting a new and improved Sense user interface.

Features of Eclair

- ✓ Allow to add multiple accounts to a device for synchronization
- ✓ Support Bluetooth 2.1
- ✓ Microsoft exchange email support
- ✓ Camera Support – including flash support, Digital zoom, scene mode, color effect, Macro focus etc.
- ✓ Improved Google maps
- ✓ You can able to search all the saved SMS and MMS messages
- ✓ Ability to tap the contact photos and choose to call and SMS.

Android 2.2 Froyo

Android 2.2 was announced in May 2010 at the Google IO conference in San Francisco. The single largest change was the introduction of the Just-In-Time Compiler -- or JIT -- which significantly speeds up the phone's processing power.

Along with the JIT, Android 2.2 also brings support for Adobe Flash 10.1. That means you can play your favorite Flash-based games in Android's web browser.

Android Froyo features are:

- ✓ Support for Bluetooth enabled car and desk docks, numeric and alphanumeric password, file upload fields in the browser application, Adobe Flash ,High PPI display
- ✓ To optimize speed, memory and performance
- ✓ Support for the Android Cloud to device messaging service.
- ✓ Voice Dialing and contact sharing over Bluetooth
- ✓ Support improved Microsoft exchange, including security policies, auto discovery, and calendar synchronization.

Android 2.3 Gingerbread

Android 2.3 came out of the oven in December 2010, and like Eclair, has a new "Googlephone" to go along with -- the Nexus S. Gingerbread brings a few UI enhancements to Android, things like a more consistent feel across menus and dialogs, and a new black notification bar, but still looks and feels like the Android we're used to, with the addition of a slew of new language support.

Ginger bread Features:

- ✓ User interface element design with simplicity and speed
- ✓ Support for front facing camera for video calling.
- ✓ Support for extra large screen size and resolutions.
- ✓ Support new download manager, with the help of this download manager you can easily download all the files in your device.
- ✓ Support for more sensors
- ✓ Support for Near Field Communication NFC
- ✓ Copy and paste option is allow in this version
- ✓ Audio, Graphical and input enhancement for game developers
- ✓ Garbage collection for increased performance

Android 3.0 and 3.1 Honeycomb

Android 3.0 came out in February 2011 with the Motorola Xoom. It's the first version of Android specifically made for tablets, and brings a lot of new UI elements to the table. Things like a new System bar at the bottom of the screen to replace the Status bar we see on phones and a new recent applications button are a great addition for the screen real estate offered by Android tablets.

Some of the standard Google applications have also been updated for use with Honeycomb, including the Gmail app and the Talk app. Both make great use of fragments, and the Talk app has video chat and calling support built in. Under the hood, 3D rendering and hardware acceleration have been greatly improved.

The Honey comb version is only for tablet and will never come to Smartphone.

Features of Android 3.0 Honeycomb:

- ✓ Simplified copy paste interface
- ✓ Support for multi core processor
- ✓ You can encrypt all users data
- ✓ Camera support – Focus, flash, zoom, time lapse etc.
- ✓ Key apps such as Gmail and you tube
- ✓ Ability to view all your gallery item in full screen mode
- ✓ Added system bar ,featuring quick access to notification, status and soft navigation buttons
- ✓ Added Action bar ,giving access to contextual options, navigation, widgets or other types.
- ✓ Hardware acceleration

Features of Android 3.1 honeycomb:

- ✓ Support for external keyboard, pointing devices, joysticks, gamepads, FLAC audio playback
- ✓ High performance in Wi-Fi Connectivity
- ✓ Connectivity for USB Accessories
- ✓ Resizable home screen widget

Android 4.0 Ice cream sandwich

Android 4.0 Ice cream Sandwich was released in 19 October 2009 based on Linux kernel 3.0.1.This version is more compatible with all the android device. But it was available on 14 November 2011.This version was used in Samsung Galaxy nexus .Ice cream sandwich ICS was designed to merge Ginger bread, android for phones ,together with honey comb.

Ice Cream Sandwich Features:

- ✓ You can able to access apps directly from lock screen
- ✓ Pinch-to-zoom functionality in the calendar
- ✓ Easy to create folder with drag and drop
- ✓ Built in photo editor
- ✓ Hardware acceleration
- ✓ Real time speech to text dictation
- ✓ Integrated screenshot capture
- ✓ Customizable launcher
- ✓ Copy and paste functionality
- ✓ You can able to speed up or slow down voicemail messages.
- ✓ 1080p video recording

- ✓ Wi-fi direct
- ✓ Support for the Web image format
- ✓ Camera support – Zero shutter lag, time lapse setting, panorama mode and able to zoom while recording

Android 4.2 jelly bean

Android 4.2 jelly bean was announced at the Google I/O conference on 27 June 2012.This version is aim to improve the functionality and performance of the user interface. But the event was cancelled. The android 4.2 jelly bean first device were LG nexus 4 and Samsung nexus 10.which were released on 13 November 2012.

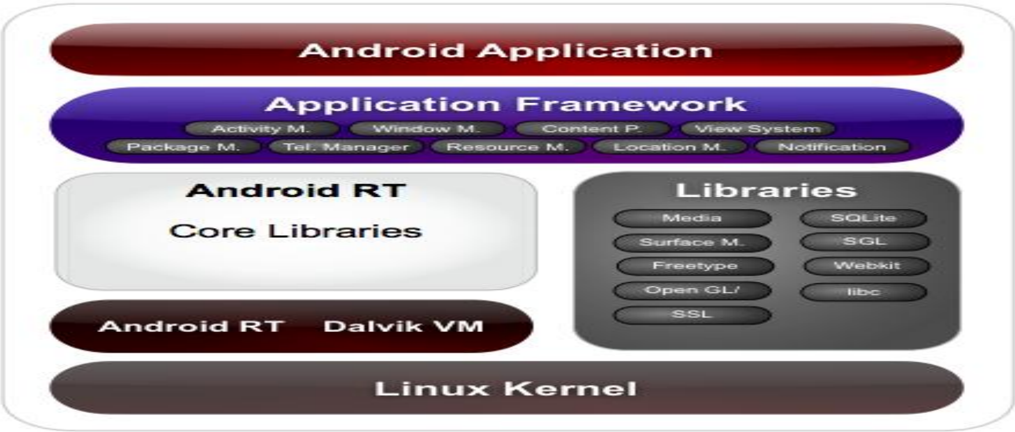
Jelly bean features are:

- ✓ Multiple user accounts
- ✓ Support for wireless display
- ✓ Notification power control
- ✓ Keyboard with gesture typing
- ✓ All device now use the same interface layout
- ✓ Always on VPN
- ✓ Premium SMS confirmation
- ✓ Lock screen improvements

Android Architecture

Being an Android user you may know how the basic functions such as making a call, sending a text message, changing the system settings, install or uninstall apps etc. Well! All Android users know these, but not enough for a developer/Tester. Then what else details are a developer/Tester required to know about Android, Go through the complete document for complete explanation. To be a developer/Tester, you should know all the key concepts of Android. That is, you should know all the nuts and bolts of Android OS.

Android Architecture Diagram:



The above figure shows the diagram of Android Architecture. The Android OS can be referred to as a software stack of different layers, where each layer is a group of several program components. Together it includes operating system, middleware and important applications. Each layer in the architecture provides different services to the layer just above it.

We will examine the features of each layer in detail.

Linux Kernel

The basic layer is the Linux kernel. The whole Android OS is built on top of the Linux 2.6 Kernel with some further architectural changes made by Google. It is this Linux that interacts with the hardware and contains all the essential hardware drivers. Drivers are programs that control and communicate with the hardware.

For example, consider the Bluetooth function. All devices has a Bluetooth hardware in it. Therefore the kernel must include a Bluetooth driver to communicate with the Bluetooth hardware. The Linux kernel also acts as an abstraction layer between the hardware and other software layers. Android uses the Linux for all its core functionality such as Memory management, process management, networking, security settings etc. As the Android is built on a most popular and proven foundation, it made the porting of Android to variety of hardware, a relatively painless task.

Libraries

The next layer is the Android's native libraries. It is this layer that enables the device to handle different types of data. These libraries are written in c or c++ language and are specific for a particular hardware. Some of the important native libraries include the following:

Surface Manager: It is used for compositing window manager with off-screen buffering. Off-screen buffering means you can't directly draw into the screen, but your drawings go to the off-screen buffer. There it is combined with other drawings and form the final screen the user will see. This off screen buffer is the reason behind the transparency of windows.

Media framework: Media framework provides different media codecs allowing the recording and playback of different media formats. (Audio/Video capturing and Audio/video recording)

SQLite: SQLite is the database engine used in android for data storage purposes.

WebKit: It is the browser engine used to display HTML content

OpenGL: Used to render 2D or 3D graphics content to the screen

Android Runtime

Android Runtime consists of Dalvik Virtual machine and Core Java libraries.

Dalvik Virtual Machine: It is a type of JVM used in android devices to run apps and is optimized for low processing power and low memory environments. Unlike the JVM, the Dalvik Virtual Machine doesn't run .class files, instead it runs .dex files. .dex files are built from .class file at the time of compilation and provides higher efficiency in low resource environments.

The Dalvik VM allows multiple instance of Virtual machine to be created simultaneously providing security, isolation, memory management and threading support. It is developed by Dan Bornstein of Google.

Core Java Libraries: These are different from Java SE and Java ME libraries. However these libraries provides most of the functionalities defined in the Java SE libraries.

Application Framework

These are the blocks that our applications directly interacts with. These programs manage the basic functions of phone like resource management, voice call management etc. As a developer/Tester, you just consider these are some basic tools with which we are building our applications.

Important blocks of Application framework are:

Activity Manager: Manages the activity life cycle of applications

Content Providers: Manage the data sharing between applications

Telephony Manager: Manages all voice calls. We use telephony manager if we want to access voice calls in our application.

Location Manager: Location management, using GPS or cell tower

Resource Manager: Manage the various types of resources we use in our Application

Applications

Applications are the top layer in the Android architecture and this is where our applications are fit. Several standard applications comes pre-installed with every device, such as:

- ✓ SMS client app
- ✓ Dialer
- ✓ Web browser
- ✓ Contact manager

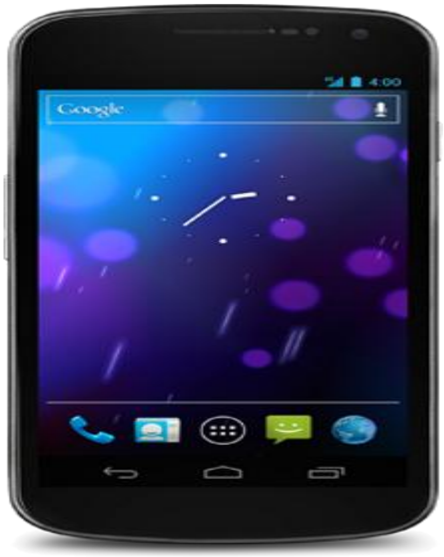
As a developer we are able to write an app which replace any existing system app. That is, you are not limited in accessing any particular feature. You are practically limitless and can whatever you want to do with the android (as long as the users of your app permits it). Thus Android is opening endless opportunities to the developer.

UI Overview of Android

Android's system UI provides the framework on top of which you build your app. Important aspects include the Home screen experience, global device navigation, and notifications.
Your app will play an important part in keeping the overall Android experience consistent and enjoyable to use.

Read on for a quick overview of the most important aspects of the Android user interface.

Home, All Apps, and Recent Apps

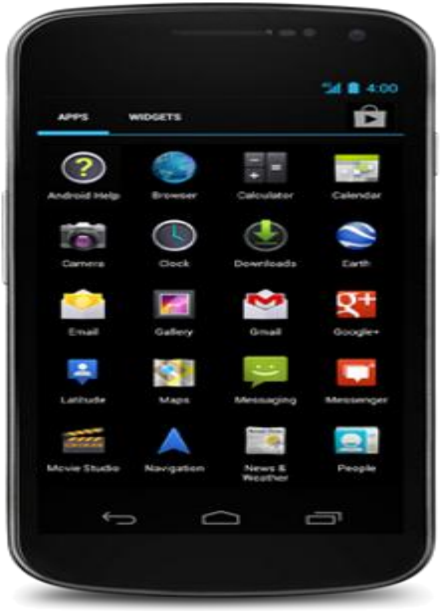


Home screen

Home is a customizable space that houses app shortcuts, folders and widgets. Navigate between different home screen panels by swiping left and right.

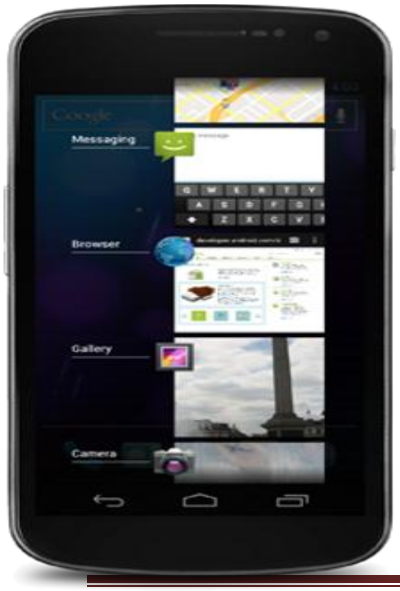
The Favorites Tray at the bottom always keeps your most important shortcuts and folders in view regardless of which panel is currently showing.

Access the entire collection of apps and widgets by touching the All Apps button at the center of the Favorites Tray.



All apps screen

The All Apps screen lets you browse the entire set of apps and widgets that are installed on your device. Users can drag an app or widget icon from the All Apps screen and place it in any empty location on any Home screen.



Recent App screen

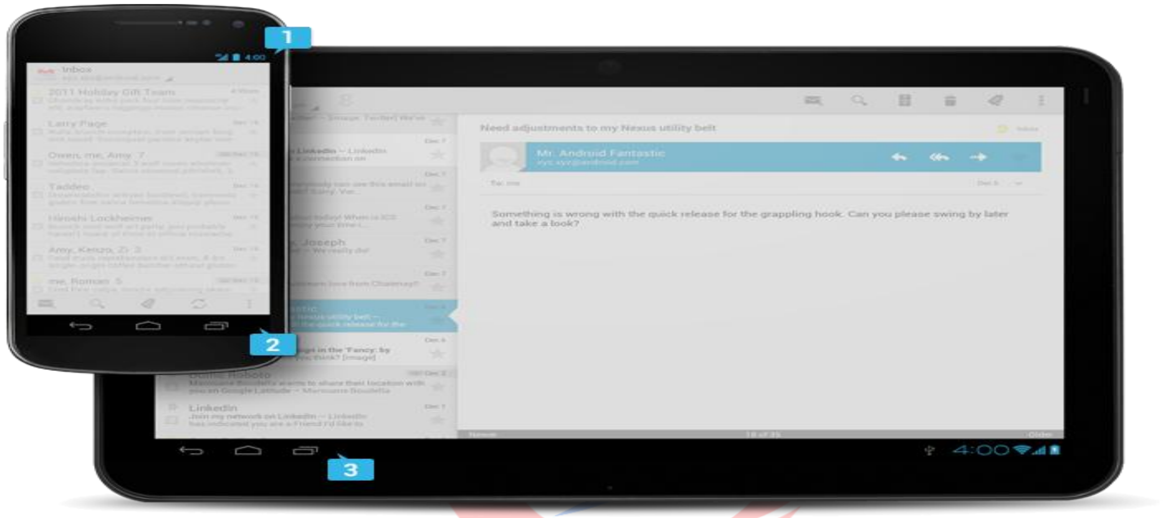
Recents provides an efficient way of switching between recently used applications. It provides a clear navigation path between multiple ongoing tasks. The Recents button at the right side of the navigation bar displays the apps that the user has interacted with most recently. They are organized in reverse

chronological order with the most recently used app at the bottom.

Switch to an app by touching it. Remove an item by swiping left or right.

System Bars

The system bars are screen areas dedicated to the display of notifications, communication of device status, and device navigation. Typically the system bars are displayed concurrently with your app. Apps that display immersive content, such as movies or images, can temporarily hide the system bars to allow the user to enjoy full screen content without distraction.



1. **Status Bar**

   Displays pending notifications on the left and status, such as time, battery level, or signal strength, on the right. Swipe down from the status bar to show notification details.

2. **Navigation Bar**

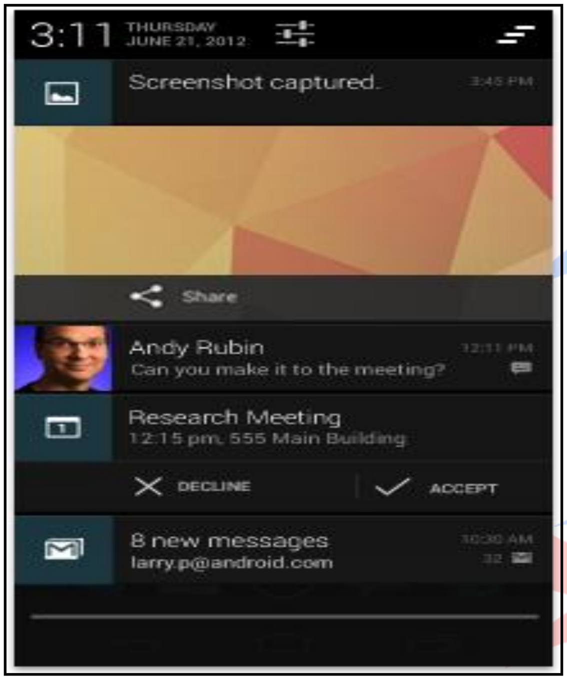
   New for phones in Android 4.0, the navigation bar is present only on devices that don't have the traditional hardware keys. It houses the device navigation controls Back, Home, and Recents, and also displays a menu for apps written for Android 2.3 or earlier.

3. **Combined Bar**

   On tablet form factors the status and navigation bars are combined into a single bar at the bottom of the screen.

Notifications

Notifications are brief messages that users can access at any time from the status bar. They provide updates, reminders, or information that's important, but not critical enough to warrant interrupting the user. Open the notifications drawer by swiping down on the status bar. Touching a notification opens the associated app.



3:11 THURSDAY JUNE 21, 2012

Screenshot captured. 3:45 PM

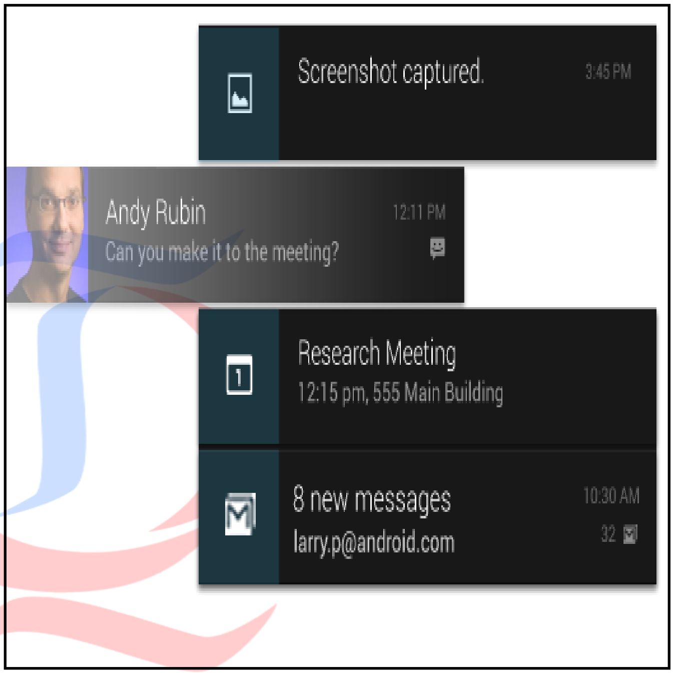
Share

Andy Rubin 12:11 PM
Can you make it to the meeting?

Research Meeting
12:15 pm, 555 Main Building

DECLINE ACCEPT

8 new messages 10:30 AM
larry.p@android.com 32



Screenshot captured. 3:45 PM

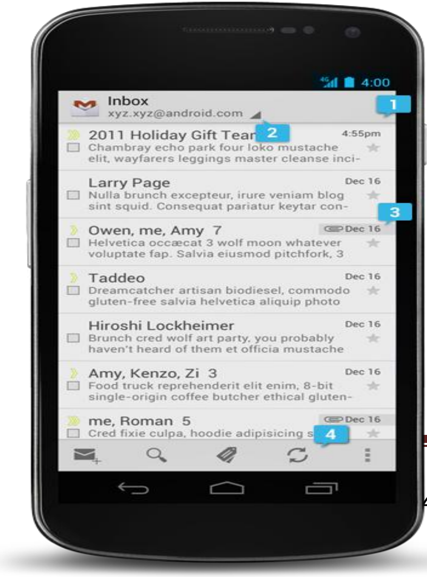
Andy Rubin 12:11 PM
Can you make it to the meeting?

Research Meeting
12:15 pm, 555 Main Building

8 new messages 10:30 AM
larry.p@android.com 32

Notifications can be expanded to uncover more details and relevant actions. When collapsed, notifications have a one-line title and a one-line message. The recommended layout for a notification includes two lines. If necessary, you can add a third line.

Swiping a notification right or left removes it from the notification drawer.



4:00

Inbox
xyz.xyz@android.com

2011 Holiday Gift Tear 4:55pm
Chambray echo park four loko mustache elit, wayfarers leggings master cleanse inci-

Larry Page Dec 16
Nulla brunch excepteur, irure veniam blog sint squid. Consequat pariatur keytar con-

Owen, me, Amy 7 Dec 16
Helvetica occæcat 3 wolf moon whatever voluptate fap. Salvia eiusmod pitchfork, 3

Taddeo Dec 16
Dreamcatcher artisan biodiesel, commodo gluten-free salvia helvetica aliquip photo

Hiroshi Lockheimer Dec 16
Brunch cred wolf art party, you probably haven't heard of them et officia mustache

Amy, Kenzo, Zi 3 Dec 16
Food truck reprehenderit elit enim, 8-bit single-origin coffee butcher ethical gluten-

me, Roman 5 Dec 16
Cred fixie culpa, hoodie adipisicing s

Common App UI

A typical Android app consists of action bars and the app content area.

1. **Main Action Bar**

The command and control center for your app. The main action bar includes elements for navigating your app's hierarchy and views, and also surfaces the most important actions.

2. **View Control**

   Allows users to switch between the different views that your app provides. Views typically consist of different arrangements of your data or different functional aspects of your app.

3. **Content Area**

   The space where the content of your app is displayed.

4. **Split Action Bar**

   Split action bars provide a way to distribute actions across additional bars located below the main action bar or at the bottom of the screen. In this example, a split action bar moves important actions that won't fit in the main bar to the bottom.

How to Install Android SDK

Pre-Installation Check List

Before installing Android SDK, you need to install:

1. Java Development Kit (JDK): Read "How to install JDK".

Now, you are ready to install the Android SDK.

Step 1: Download the Android SDK

Download the Android SDK from http://developer.android.com/sdk/index.html. For novices, choose the installer version by clicking the button "Download the SDK for Windows". For Linux and Mac, select "Other Platforms".

Step 2: Install Android SDK

Unzip the downloaded ZIP file to particular location(D:/Android). Take note of the Unzipped directory.

Step 3: Install Android Platforms and Add-ons via "SDK Manager"

The Android SDK comprises 2 parts: the "tools" and the "Platforms & Add-ons". After running the installer (in the previous step), the basic "tools" are installed, which are executables that support app development. The "Platforms & Add-ons" consist of ALL Android platforms (from Android

1.x to 4.x) and various Google Add-ons (such as Google Map API), which could be selectively installed.

Now, we have to choose our Android "Platforms & Add-ons".

1. Launch Android's "SDK Manager", which is responsible for managing the software components. If you have run the installer, it should have started the SDK Manager after the installation. Otherwise, launch the SDK manager by running (double-clicking) "SDK Manager.exe" under the Android installed directory.
2. In "Add Platforms and Packages", select your target Android platforms and add-ons packages. For novices, select "Android SDK Platform-Tools", and at least one Android platform (e.g., Android 4.1 (API 16)) ⇒ "Install".

Step 4: Create a Android Virtual Device (AVD) (or Emulator) via "AVD Manager"

AVDs are emulators that allow you to test your application without the real devices. You can create AVDs for different android platforms (from Android 1.x to Android 4.x) and configurations (e.g., screen size, orientation, SD card and its capacity).

1. Open Eclipse from unzipped folder>Eclipse
2. Select AVD manager from Eclipse
3. In "Android Virtual Device Manager" dialog ⇒ "New".
4. The "Create New Android Virtual Device (AVD)" dialog appears. In "Name", enter "Android41_Phone". Select the "Target" Android platform, "SD Card Size" (e.g., 10MB, do not set a huge SD Card size, which would take hours to create.) Skin (screen resolution, e.g., WVGA800x480 for smart phone - Wiki "Graphics display resolution" for the various resolution) ⇒ "Create AVD".

You can test your AVD by launching the emulator. Start the AVD manager ⇒ Select a AVD ⇒ Click the "Start" button ⇒ Check "Scale display to real size" to get a smaller screen that could fit in your display ⇒ Launch. Wait patiently! The emulator is very slow and take a few MINUTES to launch. You can change the orientation (between portrait and landscape) of the the emulator via "Ctrl-F11".

We typically create different AVDs to emulate different real devices, e.g., Android41_tablet of resolution (1024x768 XGA).

Step 5: Setup PATH

You can skip this step now if you are not familiar with PATH, but it is needed later.

Include the android's tools directory (unzipped directory\tools) and platform-tools directory (Unzipped directory\platform-tools) to your PATH environment variable.

For Windows: Start "Control Panel" ⇒ "System" ⇒ (Vista/7) "Advanced system settings" ⇒ Switch to "Advanced" tab ⇒ "Environment variables" ⇒ Choose "System Variables" for all users (or "User Variables" for this login user only) ⇒ Select variable "PATH" ⇒ Choose "Edit" for modifying an existing variable ⇒ In variable "Value", APPEND your Unzipped folder\tools directory (e.g., "d:\android\android-sdk\tools"), followed by a semi-colon ';', IN FRONT of all the existing path entries. DO NOT remove any existing entry; otherwise, some programs may not run. Add the platform-tools directory to the PATH too.

Step 6: Write your First Android Program Using Eclipse ADT

Android apps are written in Java, and use XML extensively. I shall assume that you have basic knowledge of Java programming and XML.

Step 7: Create a new Android Project

1. Launch Eclipse.
2. From "File" menu ⇒ New ⇒ Project.. ⇒ Android Application Project ⇒ Next.
3. The "New Android Project" dialog appears:
    1. In "Application Name", enter "Hello Android" (this is the Android appliation name that shows up on the real device).
    2. In "Project Name", enter "HelloAndroid" (this is the Eclipse's project name).
    3. In "Package Name", enter "com.example.helloandroid".
    4. In "Build SDK", select the latest version (e.g., Android 4.1 (API 16)).
    5. In "Minimum Required SDK", select "API 8 Android 2.2 (Froyo)" - almost all of the Android devices meet this minimum requirement ⇒ Next.
4. The "Configure Launcher Icon" dialog appears, which allows you to set the application's icon to be displayed on the devices ⇒ Next.
5. The "Create Activitiy" dialog appears. Check "Create Activity" Box ⇒ Select "BlankActivity" ⇒ Next.
6. The "New Blank Activity" dialog appears.
    1. In "Activity Name", enter "HelloActivity".
    2. In "Layout Name", enter "activity_hello" (default).
    3. In "Title", enter "Hello" (this title will appear as the screen title) ⇒ Finish.

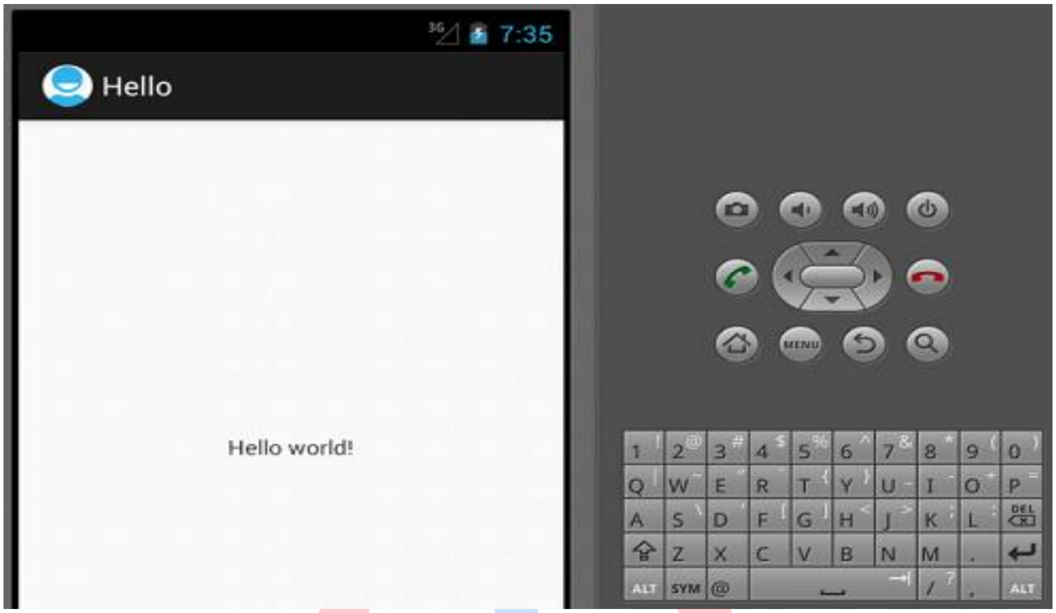
Eclipse ADT creates a default Hello-world Android app.

Step 8: Run the Android App on Emulator

Run the application by right-click on the project node ⇒ "Run As" ⇒ "Android Application".

Be patient! It takes a few MINUTES to fire up the emulator! Watch the Eclipse's status bar for the launching progress; and the console view (or LogCat view) for messages.

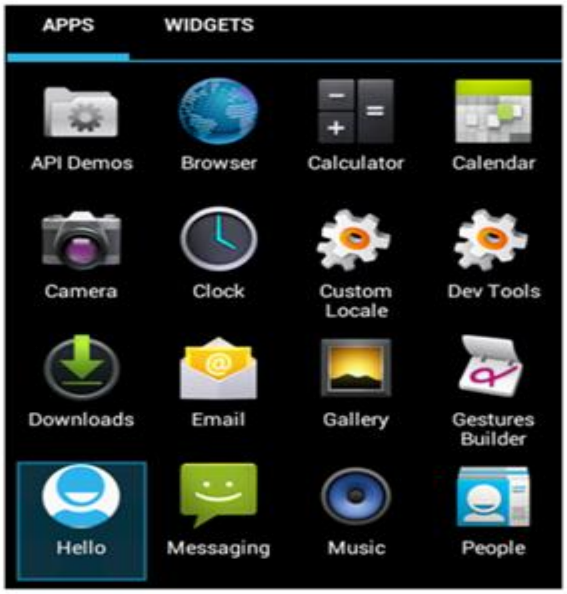
Once the emulator started, unlock the device by holding and sweeping the "lock" to the right (or left). It shall launch your Hello-world app, and displays "Hello, world!" on the screen with a title "Hello".



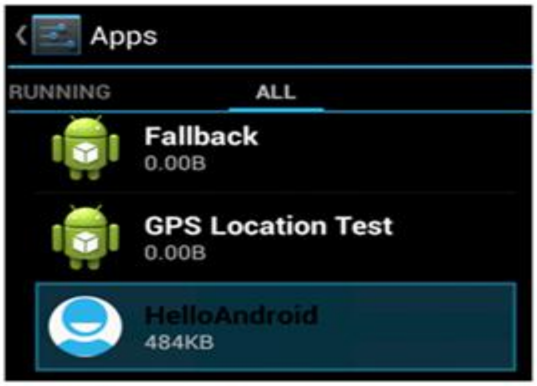
Hello world!

Trying launching the app from "HOME" ⇒ "..." ⇒ Look for the icon "Hello".

Also try "HOME" ⇒ "..." ⇒ "MENU" ⇒ "Manage Apps" ⇒ Select "HelloAndroid" ⇒ Uninstall.



In "Apps", shown as "Hello"
(Title of the Main Activity)



Fallback
0.00B

GPS Location Test
0.00B

HelloAndroid
484KB

In "manage app", shown as
"HelloAndroid" (Application Name)

NOTE: DO NOT CLOSE the emulator, as it really takes a long time to start. You could always re-run or run new applications on the same emulator.

Step 9: Run the Android App on Real Devices

To run the Android app on the real devices:

1. Connect the real device to your computer. Make sure that you have the "USB Driver" for your device installed on your computer. You can find the "Google USB Driver" @ http://developer.android.com/sdk/win-usb.html, and Google's certified "OEM USB Drivers" @ http://developer.android.com/tools/extras/oem-usb.html. If you device is not certified there, good luck! It took me many hours to find a compatible driver for my cheap Pad.
2. Enable "USB Debugging" mode on your real device: from "Settings" ⇒ "Applications" ⇒ "Development" ⇒ Check "USB Debugging". This allows Android SDK to transfer data between your computer and your device.
   Also enable "Unknown source" from "Applications". This allows applications from unknown sources to be installed on the device.
3. You shall see the message "USB Debugging Connected" when you plugs the USB cable into your computer.
4. From Eclipse, right-click on the project node ⇒ Run As ⇒ Android Application.
5. The "Android Device Chooser" dialog appears. Select your real device (instead of the AVD emulator) ⇒ OK.
6. Eclipse ADT installs the app on the connected device and starts it.

Step 10 : Installing the ".apk " file by using Android Debug Bridge

You can also use the "adb" (Android Debug Bridge) tool (Android unzipped location\sdk\platform-tools") to install the ".apk" file ("HelloAndroid.apk") onto the real devices or Emulator:

1. Open the Command prompt (Windows +R and type CMD)
2: Navigate to the Platform tool location (Ex: D:\android\sdk\platform-tools)

```
>adb  install filename.apk        (Command for Emulator)
2402 KB/s (157468 bytes in 0.064s)
        pkg: /data/local/tmp/filename.apk
Success
```

```
> adb -d install filename.apk        (Command for Real Device)
2402 KB/s (157468 bytes in 0.064s)
        pkg: /data/local/tmp/filename.apk
Success
```

> adb --help   >> For more options

List of Mobile Network Operators in India

Rank	Operator's Name	Technology	Subscribers (in millions)	Ownership
1	Airtel	GSM EDGE HSPA TD-LTE	185.92 (September 2012)	Bharti Enterprises (64.76%) SingTel (32%) Vodafone (4.4%)
2	Idea Cellular	GSM EDGE HSPA	115.66 (September 2012)	Aditya Birla (80.9%) Axiata Group Berhad (19.1%)
3	Reliance Communications	CdmaOne EVDO GSM HSPA WiMAX	154.11 (September 2012)	Reliance ADAG (67%) Public (26%)
4	Vodafone	GSM EDGE HSDPA	152.46 (September 2012)	Vodafone India (100%)
5	BSNL	GSM EDGE HSDPA HSPA+ CdmaOne EVDO WiMAX WiFi	96.28 (September 2012)	State-owned
6	Tata DoCoMo (GSM & CDMA) Virgin Mobile (GSM & CDMA) Talk24/T24 (GSM)	CDMA EVDO GSM EDGE HSPA+	90.09 (August 2012)	Tata Teleservices (74%) NTT DoCoMo (26%)
7	Aircel	GSM EDGE HSDPA	66.60 (September 2012)	Maxis Communications (74%) Apollo Hospital (26%)
8	Uninor	GSM EDGE	42.14 (September 2012)	Unitech Wireless Telenor (67.25%) Unitech Group (32.75%)
9	MTS	CDMA	14.01	Sistema (73.71%)

		EVDO	(October 2011)	Shyam Group (23.79%)
10	Videocon	GSM GPRS EDGE	4.45 (September 2012)	Videocon
11	MTNL	GSM HSDPA CDMA	5.10 (September 2012)	State-owned
12	Loop Mobile	GSM EDGE	3.02 (September 2012)	Essar Group (8.0%) Santa Trading Pvt Ltd (85.75%)
13	Ping Mobile	GSM via Videocon CDMA	1.15 (October 2011)	HFCL Infotel Limited

A mobile application tester always come across various careers like verizon, sprint, T mobile, Reliance, TATA and so on while testing a mobile application. Having a wider knowledge will definitely help you grow further in this field.

A focused approach while testing and a wider knowledge via exploration will make a difference. Such kind of Mobile Application Testers brings value not only to application but also to the Business.

Mobile OS and Browsers

OS	Browsers	Features
Android	Android WebKit Opera Mobile Firefox Opera Mini NetFront Life UC	Touch events Meta viewport Media queries Offline storage Documentation
iOS	Safari	Touch events Visual viewport size Meta viewport Media queries Offline storage Documentation
BlackBerry	BB WebKit Bolt	Touch events Visual viewport size Meta viewport Media queries Offline storage Documentation

BlackBerry old	Opera Mini BoltBB old	Visual viewport size Meta viewport Media queries
Brew MP	Opera Mobile Obigo Opera Mini Obigo old	Meta viewport Media queries Documentation
bada	Dolfin Opera Mini UC	Touch events Meta viewport Media queries Offline storage
S40	Nokia WebKit Qt WebKit Opera Mini Ovi	Visual viewport size Media queries Documentation
Symbian	Opera Mobile Nokia WebKit Qt WebKit Opera Mini Bolt UC	Visual viewport size Meta viewport Media queries Documentation
Windows Phone 7	IE7	

Testing Checklist for Mobile Applications

No.	Module	Sub-Module	Test Case Description	Expected Result
1	Installation		Verify that application can be Installed Successfully.	Application should be able to install successfully.
2	Uninstallation		Verify that application can be uninstalled successfully.	User should be able to uninstall the application successfully.
3	Network Test Cases		Verify the behavior of application when there is Network problem and user is performing operations for data call.	User should get proper error message like “Network error. Please try after some time”
4			Verify that user is able to establish data call when Network is back in action.	User should be able to establish data call when Network is back in action.

5	Voice Call Handling	Call Accept	Verify that user can accept Voice call at the time when application is running and can resume back in application from the same point.	User should be able to accept Voice call at the time when application is running and can resume back in application from the same point.
6		Call Rejection	Verify that user can reject the Voice call at the time when application is running and can resume back in application from the same point.	User should be able to reject the Voice call at the time when application is running and can resume back in application from the same point.
7		Call Establish	Verify that user can establish a Voice call in case when application data call is running in background.	User should be able to establish a Voice call in case when application data call is running in background.
8	SMS Handling		Verify that user can get SMS alert when application is running.	User should be able to get SMS alert when application is running.
9			Verify that user can resume back from the same point after reading the SMS.	User should be able to resume back from the same point after reading the SMS.
10	Unmapped keys		Verify that unmapped keys are not working on any screen of application.	Unmapped keys should not work on any screen of application.
11	Application Logo		Verify that application logo with Application Name is present in application manager and user can select it.	Application logo with Application name should be present in application manager and user can select it.
12	Splash		Verify that when user selects application logo in application manager splash is displayed.	When user selects application logo in application manager splash should be displayed.
13			Note that Splash do not remain for more than 3 seconds.	Splash should not remain for more than 3 seconds.
14	Low Memory		Verify that application displays proper error message when device memory is low and exits gracefully from the situation.	Application should display proper error message when device memory is low and exits gracefully from the situation.

15	Clear Key		Verify that clear key should navigate the user to previous screen.	Clear key should navigate the user to previous screen.
16	End Key		Verify that End Key should navigate the user to native OEM screen.	End Key should navigate the user to native OEM screen.(Original Equipment manufacturer)
17	Visual Feedback		Verify that there is visual feedback when response to any action takes more than 3 seconds.	There should be visual feedback given when response time for any action is more than 3 second. EX: Progress bar
18	Continual Keypad Entry		Verify that continual key pad entry do not cause any problem.	Continual key pad entry should not cause any problem in application.
19	Exit Application		Verify that user is able to exit from application with every form of exit modes like Slider, End Key or Exit option in application and from any point.	User should be able to exit with every form of exit modes like Slider, End Key or Exit option in application and from any point.
20	Charger Effect		Verify that when application is running then inserting and removing charger do not cause any problem and proper message is displayed when charger is inserted in device.	When application is running then inserting and removing charger should not cause any problem and proper message should be displayed when charger is inserted in device.
21	Low Battery		Verify that when application is running and battery is low then proper message is displayed to the user.	When application is running and battery is low then proper message is displayed to the user telling user that battery is low.
22	Removal of Battery		Verify that removal of battery at the time of application data call is going on do not cause interruption and data call is completed after battery is inserted back in the device.	Removal of battery at the time of application data call is going on should not cause interruption and data call should be completed after battery is inserted back in the device.
23	Battery Consumption		Verify that application does not consume battery	The application should not consume battery excessively.

			excessively.	
24	Application Start/ Restart		1. Find the application icon and select it 2. “Press a button” on the device to launch the app. 3.Observe the application launch In the timeline defined	Application must not take more than 25s to start.
25	Application Side Effects		Make sure that your application is not causing other applications of device to hamper.	Installed application should not cause other applications of device to hamper. **(Interoperability Testing)**
26	External incoming communication – infrared	If device is having IR capability we needs to check this scenario	Application should gracefully handle the condition when incoming communication is made via Infra Red [Send a file using Infrared (if applicable) to the device application presents the user]	When the incoming communication enters the device the application must at least respect one of the following: a) Go into pause state, after the user exits the communication, the application presents the user with a continue option or is continued automatically from the point it was suspended at b) Give a visual or audible notification **The application must not crash or hung.**

UNIX:

Importance of Unix for Test Engineers:

- ✓ Most of the Projects Build is deployed in Unix Servers. So it is Tester responsibility to understand Deployment instructions in UNIX.
- ✓ Understanding UNIX will give added advantage to understand the Functionality of more security applications like Banking and Insurance
- ✓ All Product based companies recruiting Test Engineers based on UNIX Knowledge.

Contents

Rename a file

mv

Listing files and directories

ls

Changing file permissions

chmod

Removing a file

rm

Directory related commands

mkdir

rmdir

cd

3 Essential Unix Commands

passwd

File related commands

wc

sort

cut

grep

fgrep

Viewing files

head

tail

4 Process in Unix

What is running right now?

Background processes

Killing a process

History of UNIX:

UNIX is a CUI (Command Unser Interface) operating system which was first developed in the 1960s. **Operating System:** An operating system can be defined as the software that controls the H/W resources of the computer and provides an environment under which programs can run.

UNIX is almost 45 year old OS. Before development of UNIX OS at AT & T Bell labs, s/w team lead by Ken Thomson, Dennis Ritchie and Rudd Canday worked on MULTICS project (Multi Information Computing System) .Initially, MULTICS was developed for only two users. Based on the same concept in 1969, UNICS (Uniplexed Information Computing System) OS was developed for 100's of users. In 1973 named as UNIX. It is open source OS.

Linux almost had same Unix Like feature for e.g.

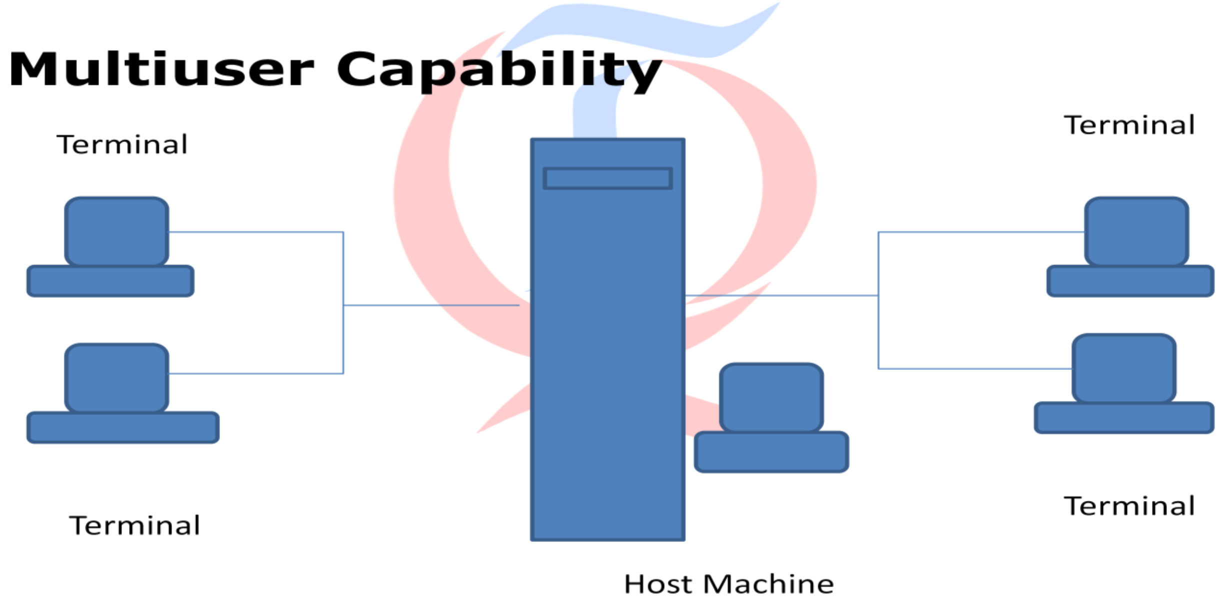
- Like UNIX, Linux is also written is C.
- Like Unix, Linux is also the Multi-user/Multitasking OS
- Like Unix, Linux runs on different hardware platform (Portable)

Flavours of UNIX:

- Aix by IBM
- Macos by apple
- Red hat linus by red hat s/w
- Solaries by sun solaries

Features of UNIX:

The Unix OS offers several features, the important of which are discussed below.



Multitasking Capability

Performing tasks simultaneously rather than sequentially. A multi tasking operating system allows more than one program to be running at a time

Communication

Communication between different terminals

Security

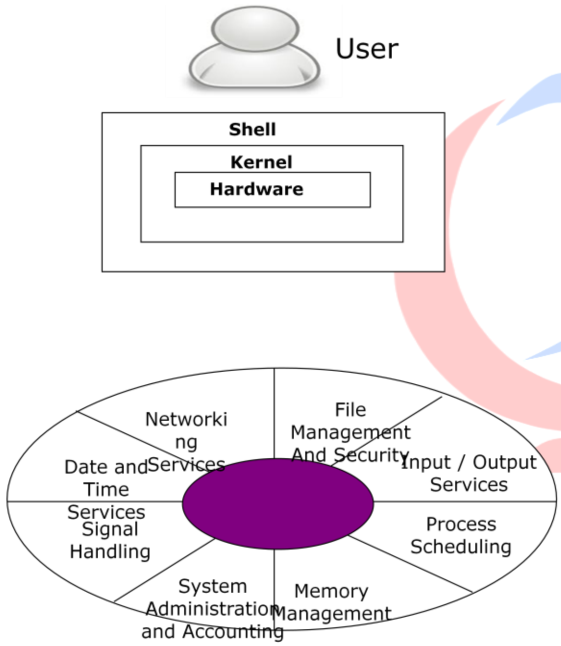
UNIX provides 3 levels of security to protect data.

- ✓ Assigning passwords and login names to individual users.
- ✓ At file level
- ✓ File encryption utility.

Portability:

It can be ported (Transfer from one system to another) to almost any computer system.

Architecture of the UNIX operating system



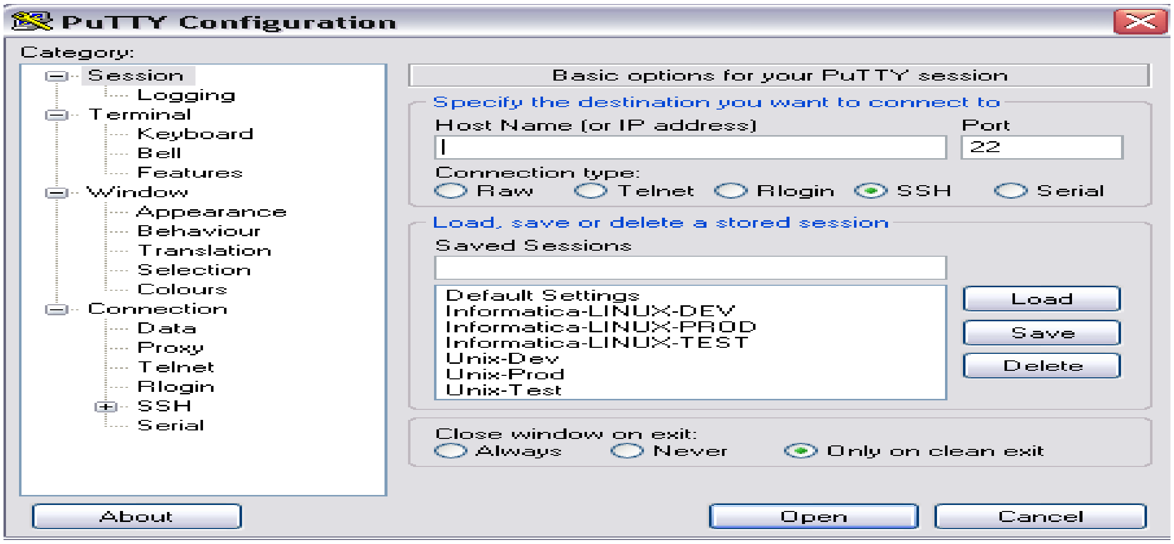
Shell:
The shell reads your commands
And interprets them as a requests
And then conveys them to the
kernel which ultimately execute them

Kernel: (Heart of nix)
Which interacts with the actual
H/W In machine language.

Functions of Kernel:
It manages files.
Manages Memory.
Scheduling of various programs.

The First Faltering Steps:

When you try to access your system, UNIX will display a prompt that looks something like this:



Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

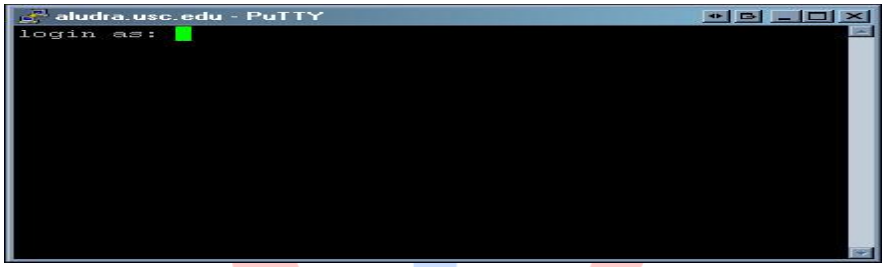
Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions

```
Default Settings
Informatica-LINUX-DEV
Informatica-LINUX-PROD
Informatica-LINUX-TEST
Unix-Dev
Unix-Prod
Unix-Test
```

Close window on exit: ○ Always ○ Never ● Only on clean exit



```
login as:
```

Login:

Password:

$who am i

It displays current user name, terminal number, date and time at which you logged in.

$who

```
Aa1     tty3a   Jan 16 01:25
Ravi    tty6c   May 22 15:10
Ramana tty3bJune 18 10:19
```

It displays login name, terminal number/serial port, date & time when logged in. note that this shown only for users who are currently logged in.

$pwd

It displays the present working directory.

$logname: It prints user's login name

$date: it displays system date and time (current date and time)

$cal 9 2003

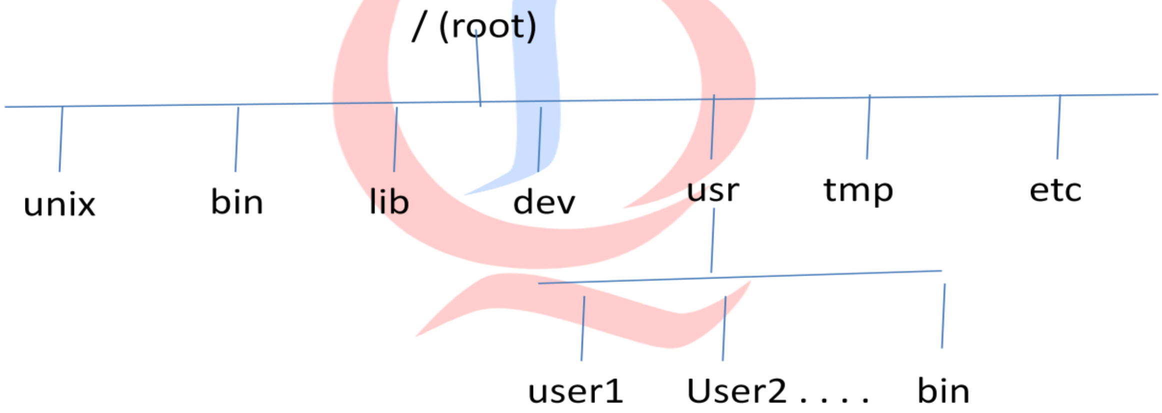
It will display calendar of September 2003.

$cal 2010

It will display calendar of entire year 2010.

UNIX File System:

A file is the basic structure used to store information on the UNIX system. All utilities, applications, data in UNIX is stored as files. Even a directory is treated as a file which contains several other files. An UNIX file system resembles an upside down tree. File system begins with a directory called **root.** The root directory is denoted as slash (/).



unix: Unix kernel itself
Bin: Directory contains executable files
Lib: Directory all the libery functions provided by Unix.
Dev: Directory contains files that controls various I/P, O/P devices
Bin: Which contains additional Unix commands.
Etc: binary executable files.

Creating files:

$touch sample

This creates a file called sample. **The size of the file would be zero bytes** since **touch does not allow you to store anything in a file.**

Then does touch serve any purpose? Yes, to create several empty files quickly.

```
$touch sample1 sample2 sample3 sample4
```

But what if you want to store a few lines in a file. Just type the command

```
$cat > sample1

******
******
******
Ctrl + d
```

To append data to the existing file.

```
$cat >> sample1
------
------
Ctrl+d
```

To view the contents of an existing file.

```
$cat filename
```

Copy a file:

Syntax: cp source file target file

```
$cp sample1 sample2
```

This will copy contents of sample1 into a sample2. If sample2 already existed it overwrites.

$cp –i sample1 sample2 → if sample2 already existed then it asks the confirmation.

Rename a file:

If you want to rename the file test to sample we would say:

```
$mv test sample
```

mv command also has the power to rename directories.

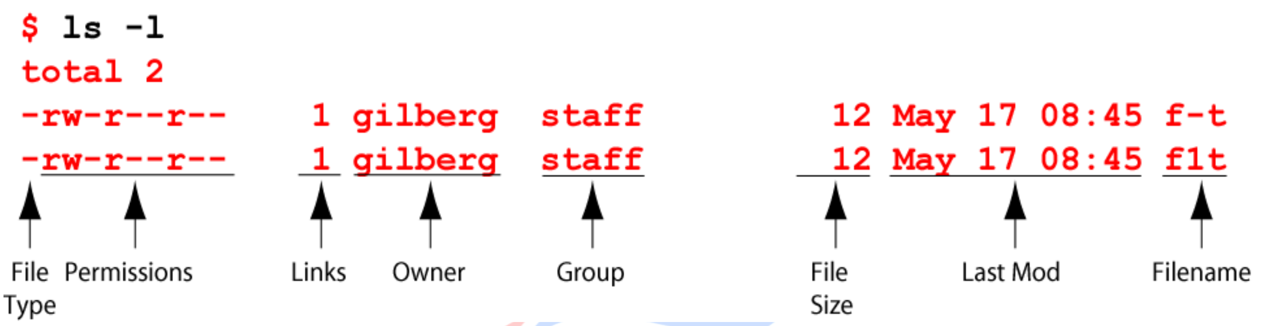
```
$mv olddir newdir
```

Note: Moving a file implies removing it from its current location and copying it at a new location.

mv file1 file2 newdir

Listing files and directories

$ls –l



```
$ ls -l
total 2
-rw-r--r--    1 gilberg  staff      12 May 17 08:45 f-t
-rw-r--r--    1 gilberg  staff      12 May 17 08:45 f1t
```

ls -> Show contents of working directory

ls file1 -> list file1, if it exists in working directory

ls dir1 -> show contents of the directory dir1

ls -a -> shows all your files, including hidden ones

ls -al -> give detailed listing of contents

ls *.doc - show all files with suffix ".doc"

ls -lt -> Time of last modification will come first (last modified/created files display first on the screen)

ls –ltr -> Time of last modification will come last.

Changing file permissions:
chmod: chmod is the command to change file permissions or directory permissions.

Permissions	**weight**
r- read	4
w- write	2
x- execute	1

Ex: $chmod 700 filename
u for user or owner

g for group
o for others

Removing a file:
$rm file1

It removes file1, if file permissions permit.

Remove multiple files:

$rm file1 file2 file3

$rm –i filename → i- interactively

Directory related commands:
$mkdir dir1

Make/create directory called dir1 in your working directory.

$mkdir dir1 dir2 dir3 dir4

To create multiple directories.

$mkdir –p dir1/dir2/dir3/dir4

Creates all the parent directories specified in the given path.

$rmdir dir1

It removes directory dir1.

Note: Directories must be empty before you remove them.

To recursively remove nested directories, use the rm command with the **-r option**:

$rm -r directory name

Changing directory:

$cd dirname

$cd .. → To change into parent directory

Essential Unix Commands:

$ passwd

Updates a user authentication.

File related commands:

$wc filename

This command is used to count the number of lines, words & characters from a file.

Options

-l	→ Lines
-w	→ words
-c	→ characters

```
$wc –l filename

$wc –w filename

$wc –lw filename

$wc –c filename

$wc –l file1 file2 file3
```

sort command:

1. Sort command can be used for sorting the contents of a file.

2. It can merge multiple sorted files and store the result in the specified output file.

3. Sort can display unique lines.

```
$sort myfile1

$sort file1 file2 file3
```

$sort –o myresuly file1 file2 file3 → here, with –o option write result to myresult instead of standard output

$sort –u –o result file1 file2 file3 → **-**u option is to display **unique** lines

$sort –m file1 file2 -m → Merge file1 content with file2.

cut Command:

Like sort, cut is also a filter. It cuts or picks up a given number of characters or fields from the specified file. (Here, cut command assumes that fields are separated by tab character).

```
$cut –f 2 file1
```

It displays second filed in file1.

```
$cut –f 2,4 file1
```

It displays 2,4th fields in file1.

```
$cut –f 1-5 file1
```

It displays 1 to 5th fields in file1.

Let us say, each piece of information is separated by a “,” then command would be

```
$cut –f 1-5 –d”,” file2
```

It displays 1 to 5th fields in file12.

```
$cut –c 1-3,5-8 abc
```

c: character by character.

It displays 1-3 characters and 5-8 characters from file **abc.**

grep command:

Globally search a regular expression.

Syntax: grep "word-to-find" {file-name}.

```
$grep hyderabad sample1
```

grep will locate all lines for the " hyderabad " pattern and print all (matched) such line(s) on-screen.

Options

-c → it returns only number of matches.

-i → ignores case while searching.

-v → returns lines that do not match the test.

fgrep Command:

It is almost similar to grep, but by using fgrep you can search for multiple patterns. But it doesn't allow you to use regular expressions.

```
$fgrep "string1

> string 2

> string 3" filename
```

Viewing files:

So far we have used the cat command to view the contents of a file. However, if the file is large in size then the matter would naturally scroll off the screen. To overcome scroll off the screen **head** and **tail** commands help in viewing lines at the beginning or end of the file.

head: Head prints the first N number of data lines of the given input. By default, it prints first 10 lines of each given file.

Syntax: head –n filename

$head -20 file1 → it displays first 20 lines from file1

tail: Tail prints the last N number of lines from given input. By default, it prints last 10 lines of each given file.

Syntax: tail -5 filename

Command: tail -5 file1 → it displays last 5 lines from file1.

Process in UNIX:

Process is kind of program or task carried out by your PC.

"An instance of running command is called **process** and the number printed by shell is called **process-id (PID)**, this PID can be used to refer specific running process."

What is running right now:

$ps

To see currently running process at your terminal.

$ps –a → processes of all the users.

Background processes:

To run command in background, you end it with an &.

Command: cp file1 file2 &

Killing a process:

Kill command is used to terminate the process or kill the process.

Syntax: kill pid

SESSION - 08

Interview Questions and Answers:

1. Tell me about yourself?

- Working as Software Test Engineer from last 3 years
- Current company and designation
- Previous company and Role
- Education details
- Native Place and Family details
- Achievements
- Future aspirations

Real Time Scenario:

a. S/w Experience:

This is XXXX working as Software Test Engineer from last 3 years. Currently I am working for XXX Company as QA.I have been started my carrier as QA in XXX Company.

Till now I have involved in all The Testing activities, Good experience in Insurance domain and Agile Methodology with Rational Tools or Mercury Tools.

b. Educational Background:

I have completed my XXX in XXXX year from XXX University with XX Percentage. My specialization is XXX

C. Family and Native place:

My Father is XXXX (Profession) and we settled in XXX place

D. Achievements and Hobbies (Optional):

Paper Presentations, Awards, Significant roles played in education or Company

E. Future aspirations (Optional):

Looking for challenging roles, Areas of interest

2. Briefly Explain about your Professional Summary?

- How you selected to the organization
- About your Training in first organization
- Your first project, client and domain
- Roles and Responsibilities
- About other companies and projects
- Focus on current project

- Summarize the discussion
  - Projects you have done till now
  - Roles you have played till now
  - Technologies you have worked on
  - Methodologies you have followed till now
  - Testing Tolls used till now
  - Domains you worked till now

Real Time scenario:

After completion of my education I got campus placement in XXXX organization.

I have joined as a trainee over there Trained on Manual, Automation Testing.

After completion of training placed in INSURANCE project as a team member in Testing.

INSURANCE is XX (Country name) based insurance company offering services in Property and Casualty, Auto, BIA insurance. My module is BIA Insurance. In this module I have involved in Integration, System Test Case design and Execution. In my second project I got chance to handle the team. **As a POC in BIA Insurance Roll Out** States I have involved in all the Testing artifacts like Test Plan, TICM, and Test Case design in various levels, Review process, Defect Tracking, Estimation, and Risk Analysis.

Coming to tools I worked on **Rational Tools like Rational Requisite Pro, Rational Clear Quest or Quality centre**

Coming to Automation experience I am having one year experience in **QTP or RFT**.

Coming to methodologies I worked in **Waterfall/V/Iteration Model (Agile Methodology)**

Coming to domain I worked in various types of **Insurance projects** and having exposure

In banking sector. Very good in offshore onsite communication.

3. Explain about the project?

a. Explain about client
b. Explain about project description
c. Explain about Business background of this project
d. Explain about Project architecture
e. Explain about user Interface flow
f. Explain about Process we are following in the project
g. Explain about tools we have used in the project
h. Explain about levels of testing in this project
i. Explain about Testing artifacts used in this project
j. Explain about offshore onsite communication in this project
k. Explain about Automation testing used in your project

4. about Client:

a. My client in name is INSURANCE COMPANY.
b. It is one of the big insurance companies in world.
c. My client is mainly focused in financial sector. I.e. Insurance, Banking, Credit Card, Mutual Funds
d. My client is focused in different lines of Insurance like Personal Insurance, BIA

5. about project description:

a. Currently I am working on BIA Insurance module

b. BIA Insurance is fall under Personal Lines of insurance

c. This is agent based application.

d. The main objective of this project is to validate the "BIA INSURANCE" application in different phases of testing.

e. BIA Insurance application is used by the insurance agents to gather information of the customers and provide them with a complete view of the Product, Coverage Limits, Premium amounts to be paid etc...

f. The Insurance Co. offers protection for the BIA of the customers and their dependents.

g. In this project we designed the Test Inventory Coverage Matrix, Design Based test cases, System test cases and Regression test cases. During Execution, we performed Integration, System, and Acceptance testing of the BIA (Insurance) application.

h. The main challenge was to exhaustively and coherently test the entire functionality of the application, while carrying forward knowledge of known problems and associated fixes to further levels of testing, thereby coordinating with other dependant Projects.

6. Business background of this project:

1. **Existing process in the organization is**
   a. Agents are manually collecting the required data from end user
   b. Agents are sending these documents to Underwrite
   c. After underwriter validation, Company is issuing the policy to the end user
   d. End users are paying the premiums manually
   e. Underwriter processing the supporting documents manually.

So, to avoid the above requirements client is looking for the new process with below requirements

- ✓ Web based application.
- ✓ Agent will collect the details from the end user and he should be able to calculate the quote
- ✓ Agent will collect the required documents through online process
- ✓ Agent can do the payment process by credit card or debit card
- ✓ Underwriter will validate the required data automatically
- ✓ Agent will submit the policy to the underwriter.

7. Project architecture:

Business Architecture:

This project is having 3 modules.

1. Agent User Interface section(AUI)
2. Underwriter section
3. Print Receipt

Step 1: Agent will collect the details from the end user

Step 2: Agent will calculate the quote

Step 3: Agent will submit the application to the underwriter

Step 4: Underwrite will validate the application as per the company policies

Step 5: After underwriter approval Print copy of the policy will be issued

8. What is the difference between AUI, Underwriter, and Print section?

Ans: Basically agent deals with front end part of the application; Agent has the choice to select any data in the application.

Only selected values are reflected to the Underwriter.

Print copy is the read only format for End-user reference

9. **Technical Architecture:**

1. It is 3-tier architecture
2. Front end is developed in J2EE
3. Backend is DB2
4. Model is developed in Mainframes

10. User Interface flow or Application Flow:

1. Agent Login
2. Product Selection
3. Policy Selection
4. Coverages
5. About You
6. Payroll
7. Quotation
8. Bill Payments
9. Attachments
10. Submit
11. Underwriter Login

12. Underwriter Approval
13. Underwriter Rejection
14. Print copies for office

11. Explain about functionality about all the screens

Ans: Refer Notes

12. Explain about Business Requirements?

Ans: Refer Notes

13. **Process we are following in the project:**

- Iteration Model
- Agile Methodology
- Scrum Framework

1. We are implementing this project in Iteration model
2. In Iteration model we are following Agile Methodology with SCRUM Framework
3. Stpe1: In Agile methodology all requirements are divided into stories.
4. Step 2: Agile methodology Works in Iterations. Iteration means time line fixed to deliver the Stories without defects. In general Iteration durations are 1 week, 2 weeks or 1month
5. Step 3: Business Analysts will send stories to the team in the starting day of iteration.
6. Step 4: When Iteration starts Developers start working on coding at the same time Testers work on designing of test cases. When the Stories are ready for testing Testers will execute all the test cases.
7. Step 5: Scrum Master Coordinates the Agile project and in Scrum meetings with all teams (BA's, Modelers, Developers, Testing team) discuss about the status on the story
8. Step 5: By the end of iteration all the stories testing should be completed without any defects
9. Step 6: Product Owners i.e. Customers will test the product

15. **About tools we have used in the project:**

JIRA for Agile Management
JIRA for Test Management
JIRA for defect Management

14. **Explain about levels of testing in this project:**

- System Testing
- System Integration Testing

I have involved in System, and System Integration testing.

UI testing focus should be on

- Navigations between different screens
- Allowed values(If applicable)
- Tabbing Sequence
- Caption display
- Accepting the value based on functionality for that particular field
-
- Basically this is Page level Testing, Interconnectivity between different screens and in some cases it focus on end to end scenario's also

System testing focus should be on

- The purpose of the system test is to ensure that the customer's documented requirements are met
- How the system is behaving as a whole when we are validating the entire system with different set of input Test data.
- Test cases and scenarios are designed to accomplish this purpose.
- Happy path scenario-→Shortest path
- Non Happy path scenario→Longest path
- Billing scenario→Main focus on mutually exclusive scenario's
- Mutually exclusive scenario's-→If we are unable to test in same scenario. we have to create new scenario to test mutually exclusive functionalities

15. **Explain about Testing artifacts used in this project:**

1. *Test Plan---*
2. *Test Case Inventory Matrix(TICM)*
3. *Query Tracker*
4. *Test Case Design check list*
5. *UI Test Cases*
6. *System Test Cases*
7. *Test Case Review check list*
8. *Test Case review Tracker*
9. *Defect Reports*
10. *Estimations, Risk analysis, Retrospective document*

I have involved in the all the testing artifacts from Test plan to retrospective document.

1. We used to get Test plan from onsite, we will look into that Test plan and we update onsite with our comments.
2. After completion of the test plan we used to prepare Test case Inventory matrix. TICM is the traceability between the requirement Id and Test case Id.
3. We have to prepare Query tracker after analyzing the requirements
4. After understanding the requirements we have to prepare test cases related to that level like Integration Test cases, System Test Cases with following the Test Case design checklist
5. After Test Case design phase Peer review, Formal review will be conducted. We will document all our observations in Review tracker
6. While doing Test Case execution we will raise defects in Rational Clear Quest
7. After execution we will prepare defect reports like Defect Trend report, defect Type, Defect by Iteration wise, Defect resolution time reports
8. We will follow weekly status reports like how many test cases designed, reviewed, executed per person and Black jack techniques for estimates
9. I have participated in maintain Risk log, Issue log, Dependency log
10. While completion of iteration we will prepare Retrospective document to know about what went well, what went wrong ,Challenges in that particular Iteration.

16. **Explain about offshore onsite communication in this project:**

We are working in Onsite Offshore model. We are getting the inputs like Test plan, requirements, Estimations, Deadlines from the client through our onsite coordinator. In offshore we have the team size as 8,Out of it 6 manual testers and 2 automation testers. we used to have client calls on weekly basis and daily basis call with Onsite coordinator.

17. Explain about Automation testing used in your project

We are using QTP to automate Functional Testing.

We are using QTP to

1. Regression Testing
2. Smoke testing for Rollouts

Explain scrum process in the project?

Stpe1: In Agile methodology all requirements are divided into stories.

- Step 2: Agile methodology Works in Iterations. Iteration means time line fixed to deliver the Stories without defects. In general Iteration durations are 1 week, 2 weeks or 1month
- Step 3: Business Analysts will send stories to the team in the starting day of iteration.
- Step 4: When Iteration starts Developers start working on coding at the same time Testers work on designing of test cases. When the Stories are ready for testing Testers will execute all the test cases.
- Step 5: Scrum Master Coordinates the Agile project and in Scrum meetings with all teams (BA's, Modelers, Developers, Testing team) discuss about the status on the story
- Step 5: By the end of iteration all the stories testing should be completed without any defects
- Step 6: Product Owners i.e. Customers will test the product.
- Success criteria for Iteration: 1.No critical defects by end of the Iteration, No high defects by end of the iteration

What testers can expect

- New features are added to the software in each of the Iteration.
- Test Teams should expect to run one or more complete test cycle in each iteration of the project, including regression testing previously delivered features as well as the current feature.
- Testers, Business Analysts, and Developers will work together as an integrated team to define, build, and test software. Testing spans the iteration (time) and spans across the team (roles).
- Requirements and code continue to change during test cycles.
- Environments and test data must be ready earlier and more closely managed.

How testers will benefit from an agile approach

- More features are fully completed earlier in the project, removing the need for a long, high-pressure testing cycle right before the product moves to production.
- Business Analysts and Testers have more opportunity to provide and receive feedback on the software earlier in the project, leading to fewer major changes late in the effort.
- More Developer testing reduces the number of "common" defects that the testers need to worry about and reduces the amount of time spent managing large amounts of defects.
- The defect report/fix cycle is dramatically shortened.
- An Agile approach can more easily accept changes in customer requests.

Challenges for testers

- Tests must be able to run quickly and repeatedly. Iterations last 2-6 weeks. Each iteration should include one to many full testing cycles. This is a huge challenge for efforts that rely on mostly manual testing.
- Test case creation should begin concurrently with the elaboration of the feature.
- Test execution must start as soon as features are ready, requiring environments and data to be set up during the first iteration.
- Performance Testing may need to occur earlier than traditional waterfall efforts.
- Requirements and code are changed during test cycles, requiring more flexible test schedules, and more regression testing.
- Integrating with non-agile projects, can present a timing challenge, as other projects might not have their side of integration points built early enough to do extensive testing.
- Managing test data and environments on shorter time frames.

Story Analysis

In the Agile Methodology, all the requirements are stored in the form of Stories. Testers should go through the requirements in detail without missing any points given by the client before writing test cases. Understanding Scope/purpose of the story will help to judge the degree of testing required. These stories are created in the form of Work Items in Rational Clear Quest for each of the Iteration. In order to access these Stories from Clear Quest, we need to import the query.

Iteration Tracker contains all the applicable stories for a particular Iteration with high level inputs on each story of the Iteration.

- Go through all the applicable stories to get high level idea like…
  Which are stories we need to design?
  Which are the stories we need to execute?

Based on Onsite updates, Offshore prepares the **Work allocation tracker**, which specifies who need to work on which story.

- Before starting to work on story follow below steps.
  Identify all your doubts
  Discuss your doubts within the team
  Consolidate all the doubts and send to onsite through POC ASAP
  Keep on reminding until you get clarification

Best Practices Followed:

- We identify all the doubts during this Story Analysis stage itself, so that we get the response early which saves our time.
- All the team members are having exposure to all stories even though they are not working on that particular story.

Test case Design

For designing test cases we follow the below steps:

- Story contains information in the form of Description or attachments, Go through Details, Success Criteria, Test Cases section.
  - Refer Requirements, UI Spec, and Take daily back-up for your scripts.
  - Follow Test Case Design Checklist

Test Case Template

- **Test Case ID:** This contains a unique Test Case identification number. This identifier often uses a prefix to denote the level of test, such as UTC for Unit Test Case.
- **Req. ID:** Enter the ID(s) of the requirement(s) the test case is validating.
- **Test Case Title:** A optional title to identify the test case. Often the Test Condition is sufficient to uniquely identify the test case.
- **Test Case Purpose:** This describes what is to be tested; the condition or transaction the test case is meant to validate.
- **Priority:** The execution priority of the test case. This priority often, but not always, relates to the priority of the requirement that is being validated. This is a REQUIRED FIELD for the row to be included in the Test Log metrics and Graphs.
- **Regression:** This flag designates if the test case tests new functionality - in which case the value should be set to "No"- or if it tests pre-existing functionality, in which case it should be set to "Yes". This is a REQUIRED FIELD for the row to be included in the Test Log metrics and Graphs.
- **Author:** This is a State Farm User ID of the person who created the test case.
- **Review Status:** This is provided by the reviewer to indicate the level of test case development/maturity. Its values are: -
  - Draft = test case is under development
  - Review Ready = test case is ready to review
  - Approved = test case reviewed and approved

  - Deferred = test case will not be used at this time

- **Pre-Conditions:** Any information, parameters, or other conditions that must exist to run this test.
- Test Steps: Detailed description about each and every action
- **Input Test Data:** The data used to execute the test case is recorded here. This would be a label of the data element(s), followed by the value(s) to be used.
- **Reset Actions:** Actions to return the application under test (AUT) and/or the test environment to its original, pre-test execution state.
- **Expected Results:** Pass criteria for the test step/case. What action/feedback the tester should expect.
- **Actual Results:** The actual behavior of the application being testes is recorded here. Record any anomalous behavior (i.e. screen turned green) here.
- **Execution Status:** This notes the status of the test case execution. If blank, the assumption is that the test case is ready, but has not yet been executed (assuming that the Regression and Priority fields have been completed - otherwise the row is skipped entirely by the Test Log worksheet.

  - Passed - Actual results match Expected Results
  - Failed - Actual results DO NOT match Expected Results
  - In Progress - Test case execution was started, but is not completed
  - Blocked - Test case cannot yet be executed
  - Deferred - Test case will not be executed in this test cycle

- **Defect ID (if applicable):** from the Defect Management System
- **Executed By:** This is a name of the person, who executed the test case. Entered by the Test Executioner.
- **Executed On:** This is the date the test case was run.
- **Last Modified**: This is the date on which the test case last edited.
- **Remarks:** This field is for entry of the changes made to a particular test case. This field is related to Last Modified field.

Test case Execution

Test execution should be done carefully based on the test cases. It is very important to use appropriate test data. It is better to create different set of test data during test case creation itself. The test data should cover valid format, invalid format and boundary values. Test result (pass/fail) should be clearly updated for each test case. It is good practice to mention

Actual behavior if the test case fails. The test results should be communicated to the other parties (developers, business/client) daily even if all the test cases are not executed. In this case, we should add a note to indicate that the test execution is still in progress. The test execution summary document/mail should clearly mention date of execution, environment, test name and test result.

Test case execution activities consists of Story testing, Smoke testing, Regression testing and Adhoc requests from Onsite.

- Smoke testing is done in morning and evening on daily basis as per guidelines.
- Look into the Iteration tracker to know the status of the story; if it is ready for testing start working on it.
- If the environment is down log a defect and update the down time sheet immediately.
- Check the environment details correctly while executing any story.
    - Ex: Release 9(R15411), Release 8, Release 7, DEV or SYS.
- Send the execution status to POC with number of scenarios executed, Number of rules executed and their status, POC will update the counts in Iteration Tracker.
- In general, stories will contain following functionalities
    - Page Level Testing
    - Rule Based test cases
    - Happy path
    - Single Location Happy path
    - Multi Location Happy path
    - Non Happy path
    - Single Location Non Happy path
    - Multi Location Non Happy path
    - Premium related fields verification
    - Quote related, Validate related error messages
    - State exceptions
    - Print functionalities
        - Agent copy
        - Customer copy
        - Quote

Regression Testing
What is Regression Testing?

Regression Testing is the selective retesting of a software system that has been modified to ensure other previously-working functions have not failed as a result of the change.

Regression test packages are made of subsets of the various types of test and takes place multiple times over the BIA of a system that represents a product. It occurs at all levels of test during the development BIA-cycle. Production systems may be involved in regression testing when there is a significant chance that the system might be affected by changes elsewhere, such as to adjacent applications or infrastructure. Regression Testing can be conducted for several distinct reasons, which provide benefits to the successful testing of the work effort:

- To verify that newly added features have not created problems with a previous version of the software

- To validate that changes made to a product and/or system have not caused any unintended affects for the system. This is often run on a final version of a code build before going into production

- Re-execution of test cases to validate that new code builds have not caused any unintended affects for the system

- To validate a particular component change or defect fix has not broken other product or system functionality

Regression Testing in BIA

Below are the items that will be present in Regression Testing Tracker

- Critical Rules sheet: we updated all the screens important rules in Critical rules
- Scenario's: We identified scenario's with covering critical Functionalities
- Functionalities: Uncovered functionalities like error handling, Validating app with high premiums
- Defects: We again tested all high withdrawn defects to make sure all functionalities are working fine.
- New Rules: We identified new rules like 1200 rules and retested most of them

Daily activities in BIA

- **Update all the trackers on daily basis**
- **Send status mail to POC on daily basis with all the details like**
  - Design count
  - Execution count
  - Doubts
  - Progress of the story
- **Smoke Testing**

- Evening Smoke Testing follow standard approach POC will send to onsite
- Morning we are doing on need Basis, If story come story comes testing then we are testing

Project

- Project description,business background of peoject,project architecture
- Client description,sdlc process followed in project
- Business requirements
- Functional requirements
- User interface
- Testplan
- Rtm
- Story analysis,query tracker
- Testcase design,test case design checklist
- Testcase review,testcase review tracker
- Testcase execution
- Defect reports
- Challenges faced in project
- Risk analysis in project,issues in peoject
- Retrospective document,iteration planing meeting,iteration review meeting
- Estimation

===

Recently asked interview Questions:

1. Give examples for positive Testing and Negative Testing?
2. Give example for EQP from your project?
3. Give example for Critical severity defect you have identified

In the project?

4. Give one example for one recently retested defect?
5. Give one example which is recently reopened defect?
6. Give one example for Low severity and Low priority defect from Your project?
7. Give examples for BVA technique for your project?
8. Explain why we need Regression Testing with example?
9. Explain one DB issues identified in the project?
10. Explain End to End flow testing for Insurance Project?
11. Explain UI Testing?
12. Explain Field Level Validations?
13. Explain Business Rules Testing?
14. Explain Compatibility Testing?
15. Explain Happy Path Testing?
16. Explain Non Happy Path Testing?
17. Explain DB Testing?

===

1. Explain Test approach in your project?

Ans: 1. UI Testing
2. Field Level ValidationS
3. Business Rules Testing
4. End to End flow Testing
5. DB Validations
6. Server Logs Validations
7. Compatibility Testing
8. Performance Testing

2. Explain recently created test cases in the project?

Ans: I have created test cases for Business Rules associated with coverage's.

I have created test cases for the field called Coverage Amount.

I have created Test cases for Story 32 which is associated with changes in premium

1. Explain critical test case you have designed in the project?

Recently I have created non happy test cases based on nearly 30 business rues understanding, it too nearly 2 days to complete test case design

4. Explain challenges you have faced while creating the test cases in the project?

4.1 Frequent changes in requirements

4.2 More non testable requirements

Ex: Missing info

4.3 Lack of KT sessions from Business team to understand requirements

Ex: Coverages

Explain how much time you need create test cases? tell me how many test cases you can design per day?

Ans:

1. Complexity of the requirement

2. Quality of the information provided in the requirement

3. KT documents available to understand the requirement

UI→30 TC per day

Field Level-→20 TC per day

Business Rules→10 day's

End to End flow→1 or 2

6. Explain Test coverage? How you ensure test coverage?

Ans: RTM document will be helpful to know all the requirements are covered in Test cases

7. What are the inputs required to create the test cases?

7.1 DFS and Test Plan

Agile: Stort cards+Test Approach

8. Expain below test case design techniques with examples from

your project?

8.1 Equivaance partitoning

8.2 Boundary value anaysis

8.3 Error Guessing

8.4 Decision Table

9. Explain differences between SYS test cases and SIT Test cases?

Ans: In my project we have 3 modules(A,Un,PS) , To test these 3 modules we have different testing teams but after completion of 3 modules SYS testing we need to check major connectivity between these 3 modules in SIT server

EX: SIT testing main focus is:

1. Submit application to und
2. Und Accept
3. Send it to policy Servicing
4. Policy Servicing will generate print copies

10. Why we need End to end flow test cases?

11. Give examples to positive and Negative test cases from your project?

12. How you decide "How many number of test cases are enough" to Test compete project?

Ans: Traceability or RTM or TICM

13. Explain smoke test cases in the project?

Ans: Happy path we have created for Retail policy

14. How you identify Suitable Test cases for Regression?

Ans: Regression optimization

15. Explain the Test case review process in the project?

16. Expain Test case Signoff process in the project?

17. Expain below test case review techniques in the project?

17.1 Peer review

17.2 Formal review

17.3 Walkthrough

18. Explain Test case design process in QC/ALM?

19. Explain Test case design process in JIRA?

20. Explain test case design guidelines followed in the project?

21. Explain your project test case template?

INTERVIEW QUESTIONS:
Test environment Prep, Test execution, Defect Tracking, Defect reporting

1. What is Test execution?
2. How to execute Test cases?
3. What is the outcome of Test execution?
4. How to execute Test cases by using QC?
5. How to log the defect?
6. Explain defect life cycle?
7. Explain defect reports?
8. Explain Metrics?
9. Explain differences between Priority vs Severity?
10. What are the reasons for raising Invalid defects?
11. What do you mean by Root cause Analysis?
12. What are the fields required to close the defect?
13. Explain production defects you have identified in the project?
14. Explain deferred defects with examples?
15. Who will prepare sign off/Release Note/Test Summary report in your project?
16. Tell me one of the critical defects you have identified in your project?
17. Tell me last 3 defects you have identified in the project?
18. How many defects you can identify per day?
19. What are the differences between mistake, error, bug, defect and failure?
20. Explain environment related defects in your project?
21. What are entry criteria to start Test execution?
22. What are exit criteria to stop test execution?
23. Tell me differences between Manual execution and Automation execution?
24. Your dev team is not able to fix defect in current release? Then how you're testing handles this scenario?
25. Your BA/Dev asked to increase Test coverage in last minute then how your team face this scenario?
26. Suddenly one your team member went on long sick leave, but client asked your team to complete the testing as per Test schedule.. Then how to overcome this scenario?
27. Client is expecting more regression coverage then how to increase the test execution coverage?
28. Dev team failed to give new build according to the Test schedule? Then what will you do?

29. Tell me some of the show stopper defects from your project?
30. You have identified a defect but developer is not agreeing with this defect? then what will you do?
31. Dev is your close friend, you have identified a critical defect, if you log this defect..That may impact your fried.. so he requested you, not to log the defect? What will you do?
32. What do you mean by Duplicated defect?
33. When we will reopen the defects?
34. Do you have any knowledge any defect tracking Tool other than QC?
35. Explain TRIAGE team?
36. One defect is in in progress state from long time? Then what will you do?
37. How to link the defects to Test cases and requirement using QC?
38. Give me examples for High severity and Low priority defects?
39. Give me examples for Low severity and High priority defects?
40. How many defects you have identified till date?
41. Explain Test case execution procedure in your project?
42. If environment is down then what will you do?
43. Who will involve in deploying the build in your current project?
44. Tell me how you deploy the build in QA server?
45. Do you have control on UAT and Production environments? if not why?
46. What do you mean by configuration management?
47. Do you have any knowledge on Version controlling Tools like SVN, clear case
48. Client has not given you the complete requirements but asking you to execute the Test cases? Then what will you do?
49. What is Risk Based execution/Risk Based Testing?
50. When you accept the build from dev team to start execution?
51. Tell me the differences between Regression testing and retesting?
52. Tell me the differences between adhoc testing and exploratory testing?
53. Tell me the differences between smoke testing and Sanity Testing
54. Tell me the differences between scripted testing and exploratory testing?
55. Tell me the differences between structured testing and non structured testing?
56. What is the objective of Regression testing?
57. What is the objective of smoke testing?
58. What is the objective of Exploratory Testing?
59. Explain about sanity Regression testing?
60. Explain about bug exploratory testing?
61. Explain about feature sanity testing?
62. Have you involved in Regression testing? If yes explain how you selected Regression suite for each release? And tell me last 3 critical defects you have identified while doing Regression testing?
63. Have you involved in exploratory testing? If yes explain defects which you have identified while doing exploratory testing? And tell me your exploratory approach in current project?
64. Have you involved smoke testing? If yes, explain your smoke testing criteria in current project?

65. Give me one practical difference between your smoke test approach and sanity test approach?
66. How much % of test cases in your project are using for Regression?
67. Is Full regression possible? If not tell me how identify regression suite in your project?
68. Automation testing is strong candidate for Regression testing? Explain the above statement?
69. When we can start Regression testing?
70. When we can start exploratory testing?
71. When we can start Smoke testing?
72. When we can start sanity testing?
73. Is retesting useful to check the stability of Application?
74. How much % of Regression is not automated? Why?
75. Role-Smoke, Regression, Exploratory?
76. Interview Questions on Test case Design?
77. What are the differences between Test Scenario and Test case?
78. Explain differences between Test case and Test script?
79. Explain about your Test case template?
80. Explain how you design test cases in QC?
81. Explain Test case design procedure in your project?
82. Explain your Project Test Approach?
83. Explain what are the documents required to design the test cases?
84. How you decide on "How many test cases are enough"?
85. What is the benefit of maintaining requirement Traceability?
86. How test plan is helpful to you for writing test cases?
87. How many test cases you can design per day?
88. If you identify any gap in requirement then what will you do?
89. What is the purpose of reviewing the test cases?
90. Explain Test case review procedure in the project?
91. Explain differences between Peer Review and Formal Review?
92. Do you follow any guidelines while creating test cases?
93. Have you involved in Test case review? if yes tell me one of review comment you have identified till now?
94. How you write test cases for Pen, Note pad, Mobile, Pen drive, Elevator, Triangle, DOB, Login page, Sign off, Change PWD, Cricket s/w....
95. Your client is asking you to write test cases without giving proper requirements? or with incomplete requirements? Then how you design test cases?
96. What is the benefit of maintaining Test case design techniques?
97. Explain below test case design techniques
    - Equivalence partitioning
    - Boundary Value Analysis
    - Error guessing
    - Decision table
98. What is the importance of Test data while creating test cases?
99. Explain your project smoke test cases?
100. Explain how you select regression suite for your project?

101.Explain Test coverage?
102.Currently you are working on web Application testing?
103.Where you have 3 layers presentation layer, Application and Db....which component is most important?
104.Explain different type of test cases in your project?
105.How many test cases you have designed till now?
106.Tell me diff between EQP and BVA?
107.Tell me critical test cases for Banking website, Face book, IRCTC, Shopping portal?
108.What is Negative testing? Explain Negative test cases?
109.What is a difference between Test cases created for system testing and UAT testing?
110.What are the various test case design tools?
111.How to upload a test case from excl to QC?
112.How to upload a test case to JIRA?
113.Do you have experience in creating Back end test cases? if yes explain few test cases?
114.Do you have experience in creating test cases for Mobile Apps? If yes explain few test cases?
115.Explain one critical test case you have created till now?
116.What will you do if test data is not available?
117.What are the characteristics of good test cases?
118.Explain Reusable test cases?
119.What is Test suite?
120.What are the benefits of maintating good test cases in project?
121.How to improve the quality of a test case?
122.What is difference between Manual Test case and Automation script?
123.What is difference between Functional test case and Perf test script?
124.Why we need test cases?
125.Why we need test case review?
126.Why we need Requirement Analysis?
127.How many test cases we will get to test below requirement?
128.Field 1 ,Field 2, Field 3 are mandatory to login
129.Why traceability is important in the project?

SESSION - 09

SAMPLE RESUMES:

MANUAL TESTING FRESHER RESUME:

Quality Thought	Phone: 9963486280
	E-mail: qthought99@gmail.com

CAREER OBJECTIVE:

Associate with a progressive organization that gives me scope to apply my knowledge, experience and skills in the area of design, development and testing of software applications which effectively contribute for company's growth.

PROFESSIONAL COMPETENCY:

- ✓ Working with **XXXXX** as a Software Engineer Trainee.
- ✓ Hands on experience in Manual Testing at all levels of STLC (Software Testing Life Cycle).
- ✓ Very good understanding on **Agile Scrum** methodology.
- ✓ Proficient in Functional testing, GUI testing, Regression testing, Automation testing, UAT testing on client/server as well as Web-based applications.
- ✓ Experience in design and execution of Test Cases.
- ✓ Hands on experience in Test and Bug Management tools like QC and Jira.
- ✓ Quick learner with the ability to grasp new technologies.
- ✓ Possess good communication skills, self-motivated, pro-active, task oriented, good team player, and quick learn at new technologies and systems.
- ✓ Excellent knowledge and working experience with test case and test script creation, test execution and test results analysis.
- ✓ Self-starter with strong communication and presentation skills.
- ✓ Having good Knowledge in Automation Testing

SKILLS SUMMARY:

Programming Languages	**:** C, C++, CORE JAVA
Data base	**:** SQL SERVER
Operating Systems	**:** WINDOWS, UNIX
Testing Tools	**:** QC, SELENIUM
Scripting Languages	**:** HTML, JAVA SCRIPT

EDUCATION DETAILS :

- **, in 2015 with 0%.
- **, in 2011 with 0%.
- **, in 2009 with 0%.

- **, in 2008 with %.

PROJECT :

Project Title : *****************

Duration : 3 months

Database : MS SQL Server 2005

Web Technologies : HTML, CSS, JavaScript, and ASP.NET with VC#.NET

IDE & Tools : Microsoft Visual Studio .Net-2008, Web Services

Description:

*** ** *** ***

PERSONAL PROFILE:

Name : ************

Date of Birth : ************

Gender : *****************

Languages Known : Telugu, English and Hindi

Nationality : Indian

Address : *****************

DECLARATION:

I hereby declare that the information furnished above is true and correct to the best of my knowledge and belief.

Place:

Date:

(@@@@@@@@@@)

MANAUL TESTING EXPERIENCE RESUME:

Venkata Ramana

E-Mail ID: qtramana@gmail.com
Mobile: 9963486280

CAREER OBJECTIVES

Seeking a challenging career in your esteemed organization, to put in my hard work and with innovative ideas, to enhance the services provided by the organization. Capable of working with end users as well as experts.

EXPERIENCE SUMMARY

- ❖ **3 years** of professional Manual testing expertise and having a good knowledge in Functional, Product testing **– FLEXCUBE, FINACLE & Web application testing**.
- ❖ Practical exposure on complete Test cycle includes **Functional, Regression, System Testing and System integration testing**
- ❖ Preparation of **Test scenarios, Test cases, Test Results, Defect Report and Requirement Traceability Matrix Document**.
- ❖ Good functional knowledge in Banking & Financial and Insurance domain.
- ❖ Have good experience in test management tools like **HP QC and JIRA.**
- ❖ Highly motivated and excellent team player with strong communication and inter-personal skills.
- ❖ Have achieved Client recognition for major and critical releases.
- ❖ Worked on **Agile Methodology with Scrum Framework**
- ❖ Positive approach, Problem solving and have the ability to work independently.
- ❖ Always adhered to the process work flow and Coordinate the team for streamlining the process.

PROFESSIONAL EXPERIENCE

- ➢ Working as Test Engineer in XXXXXX reputed to the Client XXXXXX from Aug 2014 to till date.
- ➢ Worked as a Test Engineer in Oracle Financial Services From (January 2012 – November 2012)

EDUCATIONAL QUALIFICATION

- ***, in 2015 with 0%.
- **, in 2011 with 0%.

TECHNICAL SKILL SET

Testing Tools	: HP Quality Center (QC), JIRA,
Database	: SQL
Functional Expertise	: Banking and Insurance
Product	: Flexcube, Finacle, Cyclos Web Application

CERTIFICATIONS

- Banking 100 and Banking 200
- Cognizant certified Professional in CSQA
- Cognizant Certified Professional in Software Testing
- Cognizant Certified Professional in General Insurance

TRAININGS ACHIEVED

- Flexcube UBS V 12.0 from OFSS and certified.
- Software Testing from EDISTA Certification which conducted in Microsoft Corporation
- Soft skills : Customer Centricity, Assertive Communication

PROJECT DETAILS

Project 1

Project Name:	: ***************************************
Work Experience	: Aug 2014 – till date
Client	: UNICREDIT BANK, Russia (UCBR)
Location	: IBM India Pvt ltd, Bangalore
Team Size	: 15
Role	: Test Engineer

Brief Project Profile:

UniCredit Group is one of the largest accounts of IBM worldwide. UCBR is using 2 Core Banking platforms now – MIDAS which supports Corporate and SME business and FLEXCUBE 6.3 (from Oracle) which supports the Retail Business. The strategic goal of the project, is to simplify the overall IT landscape and to replace MIDAS with FLEXCUBEV12 from OFSS

Responsibilities:

- ✓ Understanding the project requirements from Business Specification document
- ✓ **Involved in discussions with functional analyst and User to get details about the scope Of requirements.**
- ✓ Raising Queries/Ambiguities for the gaps in requirement document
- ✓ Converting Business Specification documents to test cases.
- ✓ Test Cases Review and **update the test cases with Client Requirements/Comments**
- ✓ Have interacted with the **project development team /User of Application to clarify all functional and technical issues related to the module under testing**.
- ✓ Worked on **System Test Execution.** Test Case execution for different kinds of functionalities.
- ✓ Worked on **GUI, Interface, Functional and End to End Test Execution and Defect Logging with Reporting.**
- ✓ Tracked all the defects in Quality Centre Tool, and ensured all the Defects are tracked to closure.
- ✓ **Created Trace Matrix and Test Logs, Reporting the Defects. Executed Functional Testing, System Testing and Adhoc testing.**

Project 2:

Project Name:	: ***************************************
Work Experience	: Jan 2012 – November 2012
Client	: National Australian Bank
Location	: Oracle Financial Services, Bangalore
Team Size	: 20
Role	: Individual Contributor

Brief Project Profile:

FLEXCUBE – Oracle's Core Banking application which focus on testing core banking modules like Loans, Payments , CASA etc for NAB Customers and also provide details on the other products that are being integrated and interfaced with in the end to end solution.

Responsibilities:

- Played a role as Individual Contributor
- Having Knowledge and worked in modules **Payments, and Loans of Core Banking**
- Test Execution of **Integrated Unit Testing, System Testing and Regression Testing**
- Analyzing **Customer Requirements and update Test Cases.**
- **Involved in preparing test scenarios, test cases and execution of test cases**
- Reviewing the **test cases for the team members**
- **Defect logging and tracking the status of defects**
- Providing KT to the new team members

Project 3:

Project Name:	: **************************************
Work Experience	: Feb 2011 – Dec 2011
Client	: Standard Bank Africa
Location	: Wipro Technologies, Bangalore
Team Size	: 5
Role	: Module Lead- Term Deposits

Brief Project Profile:

Finacle UBS - The Universal banking solution from Infosys, today is the core banking solution of choice for banks across the world. Finacle Core Banking is a completely web enabled, centralized, multi-currency, multi-lingual, fully integrated, customer centric solution that addresses retail, corporate and trade finance requirements of banks.

Responsibilities:

- Played a role of Test Engineer for Term Deposits of Core Banking
- Good Experience in Core Banking solutions like Deposits, General Ledger, and Operations etc.
- Functional Test Case Preparation.
- End-to-end testing of allocated modules

- Participate in status review meeting with Standard bank Africa team and vendors team
- Handle project escalations and work out resolutions
- Ensure the high quality and timely delivery of deliverables

DECLARATION:

I hereby declare that the information furnished above is true and correct to the best of my knowledge and belief.

Place:

Date:

(@@@@@@@@@@)

DB Testing Resume:

Quality Thought
Mobile: +91- 9963486280
E-mail: qthought99@gmail.com

Professional Summary:

- **6 years in 9 months** of experience in IT and presently working as a **Senior Quality Assurance Engineer**. The following main categories of **Software Testing.**
  - Black Box Testing
  - Regression Testing
  - Smoke & Sanity Testing
  - Database Testing
  - EDI Testing
- Experience in Descriptive Programming (DP) and Framework Designing.
- Expertise in preparing VB Script Functions and DP in QTP to design and manage the Test Scripts as per the designed Test Plan
- Proficient in writing and executing Test scripts, VB scripts and SQL Queries
- Proficient in preparation of Test Scenarios and Test Cases
- Have extensive experience in Smoke Testing and GUI Testing
- Good experience in Bug Reporting by using WPBN
- Proficient in preparing the following Testing Reports
  - Work Package Document
  - Product Functional Document
  - Preparing Test Case Review Reports and Defect Reports
- Good knowledge in Scrum, SDLC, STLC and Data Flow Diagrams
- Good exposure in Automation Testing

Summary of Experience:

- ❖ Presently working as a Senior Quality Assurance Engineer in ***********************., Somajiguda and Hyderabad from April 2014 to till date.
- ❖ Worked as a Senior Quality Assurance Engineer in **************, Mindspace IT Park, Maximus 2A, Madhapur and Hyderabad from November 2010 to April 2014.
- ❖ Worked as a Quality Assurance Engineer in ******************* Madhapur, near Image Hospital, Hyderabad from April 2008 to October 2010.

Educational Qualification:

- ➢ **M.C.A. Computers** from ********************************2008 with **71.8%.**

Technical Skills:

Quality Assurance	**:**	Manual Testing and Automation Testing
Bug Reporting Tools	**:**	Jira, WPBN (Whizible Project by Net) and QC
Database	**:**	MySQL and Oracle
Automation Tools	**:**	Knowledge in QTP
Deployment Tools	**:**	Bamboo, Bit Bucket
Operating Systems	**:**	Win XP/2007/VISTA/2003 Server

Project #4

Project Name	**:**	***
Client	**:**	***
Team size	**:**	4
Duration	**:**	Nov-2010 to till date
Technologies	**:**	Java, J2EE, JSP, Servlets, Hibernate, JavaScript, DWR, My SQL and Oracle
Environment	**:**	JBoss, OC4J, Eclipse and PL/SQL
Testing Approach	**:**	Manual and Automation Testing
Role	**:**	Sr. QA Engineer

Product Description:

This is a web-based product for Custom house agents to send advanced information about the goods coming into the country before the arrival of goods into the respective country.
Incoming Messages from Customs department:
Response messages which will come from the customs department as EDIFACT file. We process these messages and showed as the transaction status in the application, handling the CBSA response messages, EDIFACT incoming message processing module has developed by the CBSA guidelines.

Responsibilities:

- Understanding the client requirements and CR's.
- Involved in team meetings and KT sessions.
- Involved in writing of Test Cases and Scenarios.
- Responsible for performing Smoke, Sanity Testing, Black box Testing, Integration Testing, System Testing & Regression Testing**.**
- Involved in Preparation of Weekly Reports and Monthly Reports.
- Defect Analyzing, Defect Verification and Defect Reporting.

Project #3

Project Name	**:**	***
Clients	**:**	***
Team size	**:**	2
Duration	**:**	Nov-2010 to till date
Technologies	**:**	Java, J2EE, JSP, Servlets, Hibernate, JavaScript, DWR, MySQL and Oracle
Environment	**:**	JBoss, OC4J, Eclipse and PL/SQL
Testing Approach	**:**	Manual Testing
Role	**:**	Sr. QA Engineer

Product Description:

e-Customs is a Product that helps the Brokers / filers to file the Cargo Imported / Exported to from India from various countries to the customs and border protection department of India and thus getting Release of goods. This product have developed mainly handling freight forwarding operations done as a part of Importing and Exporting goods to / from India via Air / Ocean also helps in filing data for Indian customs clearance. The operations and transmission instructions have developed based on the instructions from Indian Customs Department. This product mainly categorized into two parts: Bill of Lading (IMPORT) and Shipping (EXPORT), Transactions and EDI modules.

Responsibilities:

- Participated on the Test Case Reviews, SRS Reviews with the Client.

- Preparing the Release Notes for every new builds.
- Guiding and supporting the Team in learning the Product, Bug logging, Testing processes etc.
- Preparing of User Manuals for Help and Support.
- Review the feedback and work closely with development team to deliver Quality.

Project#2

Project Name	**:**	**
Client	**:**	**
Environment	**:**	ASP.Net 2008, Mysql5, WINDOWS 2003-Advanced Server
Team Size	**:**	5
Testing Approach	**:**	Manual Testing
Role	**:**	Test Engineer

Description:

Academy involved in upgrading, Integration of User Application of Finance and Employee Leave Application, Advances for purchase of vehicle, Scheduling of Training programs for IPS officers Indoor & Outdoor, Along with Admin Details. Where Administrator enters all the details of employees and Leave Application maintains Leave balances of individual employee, Finance includes User Applications like TADA, LTC, CGHS and GPF will be raised by user which includes hierarchical approval till directors is also done through software and proceeds to respective section for sanction of applications. Where all the application is finally accepted, respective users of that heads do Integration from User individual application to Admin.

Responsibilities:

- Understanding the Client Requirements and Project Functionalities.
- Involved in GUI, Functional, Regression and Sanity Testing.
- Review of Functional and System Test cases.
- Involved in team meeting and KT sessions.
- Involved in Preparation of Test Status Reports.
- Involved in Preparation of Weekly Reports and Monthly Reports.

Project #1

Project Name	**:**	**
Client	**:**	**
Environment	**:**	ASP.Net, MySql5, WIN 2003-Advanced Server
Testing Approach	**:**	Manual Testing
Team Size	**:**	4
Role	**:**	Test Engineer

Description:

IFMS is involved in upgrading the existing system and Employee Pay & Accounts section, which have based on Offline Data Collection and Single Point Information Routing. The project is designed with the aim of information sharing and providing realistic online data to the user through the NT networking capabilities of the operating system and .NET, Mysql5.0 database also an Electronic movement of information is present which reduces the cycle time for action at every stage as compared to the physical movement of document increasing the cycle time for every action.

Achievements:

- I got **Star** award for exceeding expectations in creativity and innovation.

Place: Hyderabad

Date:

(*************************)

Mobile Application Testing Resume:

NAME Email: **************@gmail.com

Mobile number: ************

Summary Of Qualifications

- Good Knowledge in **Mobile Application Testing on Android and iOS.**
- Good Knowledge in all the stages of SDLC and STLC.
- Good Knowledge with all levels of testing which includes Functional Testing, Integration Testing, System Testing, Retesting, Regression Testing, End to End testing, Compatibility Testing, Recovery Testing, Installation Testing, Interoperability Testing, Localization Testing, Location based Testing, Interruption Testing and User Acceptance Testing.
- Good Knowledge in Multi browser Testing and Cross browser Testing.
- Good Knowledge in **Agile-SCRUM** process.
- Good Knowledge in process managementtool **JIRA, Agile dash boards and charts.**
- Good Knowledge in QC, Bugzilla and Testopia.
- Good Knowledge in creating and executing test cases in multiple environments like Windows, Linux and Mac.
- Good Knowledge in SQL statements creation and execution on **Postgres** database using **Pgadmin**.

Technical Expertise

Operating systems	Windows, Mac & Linux (Ubuntu, Cent OS)
Languages	C, Java
Management Tool	JIRA, Quality Center, Testopia
Database	Postgres, SQLite
Mobile Operating Systems	BB10,Android and iOS
Automation Tools	Monkey talk, Jmeter

Bug Tracking Tool	Bugzilla
Mobile Apps Testing tools	BDT, Toolbert, BBLink, iTunes, AVD manager, ADB, Logcat and DDMS
Sniffer toool	Httpfox, IE inspector &Wireshark

EDUCATION AND CERTIFICATIONS

- B.Tech (EIE) from JNTU Hyderabad.

PROJECT DETAILS

I.	Project	:	****************	Aug 2013-Till date
	Client	:	****************	
	Role	:	Module Lead	

Description:

****
****
****
****
****

Responsible Modules:

- ✓ OOBE
- ✓ Weather
- ✓ SIM Security
- ✓ File Manager
- ✓ Adobe Reader
- ✓ OTA

DECLARATION:

I hereby declare that the information furnished above is true and correct to the best of my knowledge and belief.

Place:

Date:

(@@@@@@@@@@)

OFFERED COURSES @ QUALITY THOUGHT

1. MANUAL TESTING
2. SELENIUM with CORE JAVA
3. ETL TESTING (TERADATA, INFORMATICA & COGNOS)
4. MOBILE AUTOMATION TESTING (APPIUM)
5. LOADRUNEER
6. MOBILE APPLICATION TESTING
7. JMETER / BLAZEMETER /CLOUD PERFORMANCE TESTING
8. WEB SERVICES TESTING (SOAPUI)
9. HADOOP DEVELOPMENT
10. HADOOP DB
11. DB TESTING
12. MANUAL LIVE PROJECT
13. SELENIUM LIVE PROJECT

With

Live Projects