**Non Functional Testing**

* **Performance Test Environment ---** Performance Testing – determining how quickly a product handles a variety of events and the runtime performance of the application and its supporting infrastructure under certain conditions. Performance testing is used to measure several system characteristics such as processing speed, response time resource consumption, throughout and efficiency. Performance testing is sometime coupled with stress testing.
* **Load Testing –** testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or falls.
* **Stress Testing –** Term often used interchangeably with ‘load’ and ‘performance’ testing used to measure how well product/ application performs when a load(data volume /user requests) is placed on the system resource that nears and then exceeds capacity. The application is tested against heavy load such as complex numerical values, large number of inputs, large number of queries etc. which checks for the stress load the application can withstand.
* **Volume Testing/ Endurance Testing –** volume testing is done against the efficiency of the application. Huge amount of data is processed through the application (which is being tested) in order to check the extreme limitations of the system.

**Manual Testing**

* **test planning**
* **writing test scenarios**
* **test case design**
* **test data identification**
* **test execution**
* **defect management**
* **status reporting**

**test plan**

**A test plan is a document describing the scope, approach, objectives, resources, and schedule of a software testing effort. It identifies the items to be tested, items not be tested, who will do the testing, the test approach followed, what will be the pass/fail criteria, training needs for team, the testing schedule etc.**

 **1. Test plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Features not to be tested
6. Approach
7. Item pass/fail criteria
8. Test deliverables
9. Testing tasks
10. Environmental needs
11. Responsibilities
12. Staffing and training needs
13. Schedule
14. Risks and contingencies
15. Approvals**

**Test Planning Activities:**

* **To determine the scope and the risks that need to be tested and that are NOT to be tested.**
* **Documenting Test Strategy.**
* **Making sure that the testing activities have been included.**
* **Deciding Entry and Exit criteria.**
* **Evaluating the test estimate.**
* **Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.**
* **The Test artifacts delivered as part of test execution.**
* **Defining the management information, including the metrics required and defect resolution and risk issues.**
* **Ensuring that the test documentation generates repeatable test assets.**

**test case design**

**What is Test case?**

 **A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.**

 **Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.**

**Typical Test Case Parameters:**

* **Test Case ID**
* **Test Scenario**
* **Test Case Description**
* **Test Steps**
* **Prerequisite**
* **Test Data**
* **Expected Result**
* **Test Parameters**
* **Actual Result**
* **Environment Information**
* **Comments**

**Following are the typical design techniques in software engineering:**

1. Deriving test cases directly from a requirement specification or black box test design technique. The Techniques include:

Boundary Value Analysis (BVA)

Equivalence Partitioning (EP)

Decision Table Testing

State Transition Diagrams

Use Case Testing

2. Deriving test cases directly from the structure of a component or system:

Statement Coverage

Branch Coverage

Path Coverage

LCSAJ Testing

3. Deriving test cases based on tester's experience on similar systems or testers intuition:

Error Guessing

Exploratory Testing

**test data identification**

**What is Test Data?**

 **Test Data is data that is used to execute the tests on test ware. Test data needs to be precise and exhaustive to uncover the defects.**

**Test Data Generation Techniques:**

* **Random Test Data Generators**
* **Goal-Oriented Test Data Generators**
* **Pathwise Test Data Generators**
* **Intelligent Test Data Generators**

**test execution**

**What is Test Execution?**

 **Test execution is the process of executing the code and comparing the expected and actual results. Following factors are to be considered for a test execution process:**

* **Based on a risk, select a subset of test suite to be executed for this cycle.**
* **Assign the test cases in each test suite to testers for execution.**
* **Execute tests, report bugs, and capture test status continuously.**
* **Resolve blocking issues as they arise.**
* **Report status, adjust assignments, and reconsider plans and priorities daily.**
* **Report test cycle findings and status.**
* **What is Test Strategy?**

 **Test Strategy is also known as test approach defines how testing would be carried out. Test approach has two techniques:**

 **Proactive - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.**

 **Reactive - An approach in which the testing is not started until after design and coding are completed.**

* **Different Test approaches:**

 **There are many strategies that a project can adopt depending on the context and some of them are:**

* **Dynamic and heuristic approaches**
* **Consultative approaches**
* **Model-based approach that uses statistical information about failure rates.**

 **Approaches based on risk-based testing where the entire development takes place based on the risk**

 **Methodical approaches which is based on failures.**

 **Standard-compliant approach specified by industry-specific standards.**

* **Factors to be considered:**

 **Risks of product or risk of failure or the environment and the company**

 **Expertise and experience of the people in the proposed tools and techniques.**

 **Regulatory and legal aspects, such as external and internal regulations of the development process**

 **The nature of the product and the domain**

**What is Bug/Defect?**

* **Simple Wikipedia definition of Bug is: “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”**
* **Other definitions can be:
An unwanted and unintended property of a program or piece of hardware, especially one that causes it to malfunction.**
* **or
A fault in a program, which causes the program to perform in an unintended or unanticipated manner.**
* **Lastly the general definition of bug is: “failure to conform to specifications”.**
* **If you want to detect and resolve the defect in early development stage, defect tracking and software development phases should start simultaneously.**

**Bug status description:**

* **These are various stages of bug life cycle. The status caption may vary depending on the bug tracking system you are using.**

**1) New: When QA files new bug.**

**2) Deferred: If the bug is not related to current build or can not be fixed in this release or bug is not important to fix immediately then the project manager can set the bug status as deferred.**

**3) Assigned: ‘Assigned to’ field is set by project lead or manager and assigns bug to developer.**

**4) Resolved/Fixed: When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’ and the bug is passed to testing team.**

**5) Could not reproduce: If developer is not able to reproduce the bug by the steps given in bug report by QA then developer can mark the bug as ‘CNR’. QA needs action to check if bug is reproduced and can assign to developer with detailed reproducing steps.**

**6) Need more information: If developer is not clear about the bug reproduce steps provided by QA to reproduce the bug, then he/she can mark it as “Need more information’. In this case QA needs to add detailed reproducing steps and assign bug back to dev for fix.**

**7) Reopen: If QA is not satisfy with the fix and if bug is still reproducible even after fix then QA can mark it as ‘Reopen’ so that developer can take appropriate action.**

**8 ) Closed: If bug is verified by the QA team and if the fix is ok and problem is solved then QA can mark bug as ‘Closed’.**

**9) Rejected/Invalid: Some times developer or team lead can mark the bug as Rejected or invalid if the system is working according to specifications and bug is just due to some misinterpretation.**

**The two main parameters that form the basis for effective defect tracking and resolution are:**

* **Defect priority**
* **Defect Severity**

 ***These are often a confused concept and are almost used interchangeably amongst not only test teams but also development teams. There’s a fine line between the two and it’s important to understand that there are indeed differences between the two.***

**What is Defect Priority and what is Defect Severity?**

* **Priority by the English definition is used in the comparative of two things or conditions, where one has to be given more importance than the other(s) and has to be tackled with/resolved first before proceeding to the next one(s). Therefore in the context of defects, priority of a defect would indicate the urgency with which it would need to be fixed.**
* **Severity by the English definition is used to describe the gravity of an undesirable occurrence. Hence when it comes to bugs, severity of a bug would indicate the effect it has on the system in terms of its impact.**
* **Priority 1 – Critical (P1): This has to be fixed immediately within 24 hours. This generally occurs in cases when an entire functionality is blocked and no testing can proceed as a result of this. Or in certain other cases if there are significant memory leaks, then generally the defect is classified as a priority -1 meaning the program/ feature is unusable in the current state.**
* **Priority 2 – High (P2): Once the critical defects have been fixed, a defect having this priority is the next candidate which has to be fixed for any test activity to match the “exit” criteria. Normally when a feature is not usable as it’s supposed to be, due to a program defect, or that a new code has to be written or sometimes even because some environmental problem has to be handled through the code, a defect may qualify for a priority 2.**
* **Priority 3 – Medium (P3): A defect with this priority must be in contention to be fixed as it could also deal with functionality issues which is not as per expectation. Sometimes even cosmetic errors such as expecting the right error message during the failure could qualify to be a priority 3 defect.**
* **Priority 4 – Low (P4): A defect with low priority indicates that there is definitely an issue, but it doesn’t have to be fixed to match the “exit” criteria. However this must be fixed before the GA is done. Typically, some typing errors or even cosmetic errors as discussed previously could be categorized in here. Sometimes defects with priority low are also opened to suggest some enhancements in the existing design or a request to implement a small feature to enhance user experience.**
* **Critical / Show Stopper (S1): A defect that completely hampers or blocks testing of the product/ feature is a critical defect. An example would be in case of UI testing where after going through a wizard, the UI just hangs at one pane or doesn’t go further to trigger the function. Or in some other cases, when the feature developed itself is missing from the build.**
* **Major or Severe (S2): A major defect occurs when the functionality is functioning grossly away from the expectations or not doing what it should be doing. An example could be: Say that a VLAN needs to be deployed on the switch and you are using a UI template that triggers this function. When this template to configure VLAN fails on the switch, it gets classified as a severe functionality drawback.**
* **Moderate/ Normal (S3): A moderate defect occurs when the product or application doesn’t meet certain criteria or still exhibits some unnatural behavior, however the functionality as a whole is not impacted. For example in the VLAN template deploy above, a moderate or normal defect would occur when the template is deployed successfully on the switch however there is no indication being sent to the user.**
* **Low or Minor (S4): A minor bug occurs when there is almost no impact to the functionality, but is still a valid defect that should be corrected. Examples of this could include spelling mistakes in error messages printed to user or defects to enhance the look and feel of a feature.**

**Examples of sorting defects with priority and severity**

* **High priority, high severity: Any defects due to which the testing cannot continue at any cost or causes a severe system failure fall under this category. Say for example clicking on a particular button doesn’t load the feature itself. Or performing a particular function brings down the server consistently and causes data loss. The red lines in the above figure indicate these kind of defects.**
* **High priority, low severity: Defects which have to be fixed but do not affect the application come under this category. Say for example the feature is expected to display a particular error to the user with respect to its return code. In this case, functionally the code will throw an error, but the message will need to be more relevant to the return code generated. The blue lines in the figure indicate these kinds of defects.**
* **High Severity, low priority: Defects which have to be fixed but not immediately. This can specifically occur during ad-hoc testing. It means that the functionality is affected to a large extent, but is observed only when certain uncommon input parameters are used. Say for example a particular functionality can be used only on a later version of the firmware, so in order to verify this – the tester actually downgrades his system and performs the test and observes a serious functionality issue which is valid. In such a case the defects will be classified in this category denoted by pink lines, as normally end users will be expected to have a higher version of firmware.**
* **Low severity, low priority: These defects are classified in the green lines as shown in the figure and occur when there is no functionality impact, but still not meeting the standards to a small degree. Generally cosmetic errors or say dimensions of a cell in a table on UI are classified in here.**

**What is a Web Application ?**

* **A simple definition of a web application is a software program which the user accesses over a network with a web browser.**
* **Most people are familiar with programs that are installed directly on their computers such as Microsoft Office Word.  Program files are stored directly on the computer’s hard drive.  For the program to properly function, a computer needs to meet certain requirements such as operating system compatibility, RAM and hard drive, memory, or processing speed.**
* **Web applications require only a web browser such as Internet Explorer, enabling the users to access the application from almost any computer.  Some web applications require the use of a specific web browser and most require browser add-ins and browser languages such as JAVA.  Data can be shared with everyone in an organization regardless of location.  It is important to keep in mind, response time of the application is dependent on the connection speed (For Internet: dial-up, DSL, Fiber Optics, etc).**

**The Web server**

* **A Web server handles the HTTP protocol. When the Web server receives an HTTP request, it responds with an HTTP response, such as sending back an HTML page. To process a request, a Web server may respond with a static HTML page or image, send a redirect, or delegate the dynamic response generation to some other program such as CGI scripts, JSPs (JavaServer Pages), servlets, ASPs (Active Server Pages), server-side JavaScript, or some other server-side technology. Whatever their purpose, such server-side programs generate a response, most often in HTML, for viewing in a Web browser.**
* **Understand that a Web server's delegation model is fairly simple. When a request comes into the Web server, the Web server simply passes the request to the program best able to handle it. The Web server doesn't provide any functionality beyond simply providing an environment in which the server-side program can execute and pass back the generated responses. The server-side program usually provides for itself such functions as transaction processing, database connectivity, and messaging.**
* **While a Web server may not itself support transactions or database connection pooling, it may employ various strategies for fault tolerance and scalability such as load balancing, caching, and clustering—features oftentimes erroneously assigned as features reserved only for application servers.**

**Apache HTTP Server**

 **Apache HTTP Server (also referred to as simply "Apache") has, at the time of writing, been the most popular web server on the web since 1996. Apache is developed and maintained by the Apache Software Foundation, which consists of a decentralized team of developers. The software is produced under the Apache license, which makes it free and open source. Apache is available for a range of operating systems, including Unix, Linux, Novell Netware, Windows, Mac OS X, Solaris, and FreeBSD.**

 **Microsoft Internet Information Services (IIS)**

 **IIS is, at the time of writing, the second most popular web server on the web. It is however, gaining market share, and if the current trend continues, it won't be long before it overtakes Apache. IIS comes as an optional component of most Windows operating systems. You can install IIS by using*Add/Remove Windows Components* from *Add or Remove Programs* in the Control Panel.**

 **Sun Java System Web Server**

 **Based on the Sun One Web Server, the Sun Java System Web Server is designed for medium to large business applications. Sun Java System Web Server is available for most operating systems.**

* **Apache HTTP Server website:**[**http://httpd.apache.org**](http://httpd.apache.org/)
* **Microsoft IIS website:**[**http://www.microsoft.com/iis**](http://www.microsoft.com/iis/)

 **Sun Java System Web Server website:**[**http://www.sun.com/software/products/web\_srvr/home\_web\_srvr.xml**](http://www.sun.com/software/products/web_srvr/home_web_srvr.xml)

**The application server**

**As for the application server, according to our definition, an application server exposes business logic to client applications through various protocols, possibly including HTTP. While a Web server mainly deals with sending HTML for display in a Web browser, an application server provides access to business logic for use by client application programs. The application program can use this logic just as it would call a method on an object (or a function in the procedural world).**

**Such application server clients can include GUIs (graphical user interface) running on a PC, a Web server, or even other application servers. The information traveling back and forth between an application server and its client is not restricted to simple display markup. Instead, the information is program logic. Since the logic takes the form of data and method calls and not static HTML, the client can employ the exposed business logic however it wants.**

**In most cases, the server exposes this business logic through a component API, such as the EJB (Enterprise JavaBean) component model found on J2EE (Java 2 Platform, Enterprise Edition) application servers. Moreover, the application server manages its own resources. Such gate-keeping duties include security, transaction processing, resource pooling, and messaging. Like a Web server, an application server may also employ various scalability and fault-tolerance techniques.**

**An Application Server is a complete server which provides an environment for running the business components (EJBs, ADF BCs, etc.) in addition to providing the capabilities of a Web Container as well as of a Web Server. Example: Bea WebLogic, IBM WebSphere, Oracle Application Server, etc.**

**Client/server is a program relationship in which one program (the client) requests a service or resource from another program (the server).  A client/server application is a piece of software that runs on a client computer and makes requests to a remote server.**

 **Client Server - Application is one where you have a designated server and its client - basically like a network setup in college, company - called an intranet. Or the Unix and Linux server and clients accessing the server setup for real time access. This is a simplest way to say about the client server, of course the Client Server can be connected to internet and so many complications.

Web Application - Where you access the web server using the browsers, which can go thru your intranet.**

* **1. Client server application using two tier architecture.
1.1 . using client and server only.
1.2 changes can't reflect in this. used to install again and test whether application works or not.
1.3 Single user system.
1.4 difficult to test scripting problem in this.
1.5 Using functionality, Regression, load testing.**

 **2. Web application using multi tier architecture
2.1 many users can use at the same time.
2.2 three tier architecture which are user client, middled tier which is for Database and application server for this.
2.3 changes regarding scripting reflecting at the moment only.
2.4 Easy for testing.
2.5 using functionlity, regression, load, stress, performance,security, etc.**

**Testing Web application**

**Web application is a bit different and complex to test as tester don’t have that much control over the application. Application is loaded on the server whose location may or may not be known and no exe is installed on the client machine, you have to test it on different web browsers. Web applications are supposed to be tested on different browsers and OS platforms so broadly Web application is tested mainly for browser compatibility and operating system compatibility, error handling, static pages, backend testing and load testing.**

**Testing Desktop application**

* **Desktop application runs on personal computers and work stations, so when you test the desktop application you are focusing on a specific environment. You will test complete application broadly in categories like GUI, functionality, Load, and backend i.e DB.**

**Testing client server**

 **In client server application you have two different components to test. Application is loaded on server machine while the application exe on every client machine. You will test broadly in categories like, GUI on both sides, functionality, Load, client-server interaction, backend. This environment is mostly used in Intranet networks. You are aware of number of clients and servers and their locations in the test scenario.**