Test Types

A test type is focused on a particular test objective, which could be the testing of the function to be performed by the component or system; a non-functional quality characteristics, such as reliability or usability; the structure or architecture of the component or system; or related to changes, i.e confirming that defects have been fixed (confirmation testing or retesting) and looking for unintended changes (regression testing).

Four software test types

1. Functional testing
2. Non-functional testing
3. Structural testing
4. Change related testing

Functional testing (Testing of functions)

- Basically the testing of the functions of component or system is done. - activities that verify a specific action or function of the code.
- Functional test tends to answer the questions like "can the user do this" or "does this particular feature work".
- The techniques used for functional testing are often specification-based.
- Testing functionality can be done from two perspective:
    - **Requirement-based testing:** requirements are prioritized depending on the risk criteria and accordingly the tests are prioritized. This will ensure that the most important and most critical tests are included in the testing effort.
    - **Business-process-based testing:** the scenarios involved in the day-to-day business use of the system are described. It uses the knowledge of the business processes. For example, a personal and payroll system may have the business process along the lines of: someone joins the company, employee is paid on the regular basis and employee finally leaves the company.

Non-functional testing (Testing of software product characteristics)

The quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action

Non-functional testing includes:

- **Functionality testing**: to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.
- **Reliability testing:** exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer's reliability requirements.
- **Usability testing:** Testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not. Is the system able to have **Learnability, Efficiency, Errors, and Satisfaction** features**?**
- **Efficiency testing:** test the amount of code and testing resources required by a program to perform a particular function.
- **Maintainability testing:** how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.
- **Portability testing:** process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. Results are measured in terms of the time required to move the software and complete the documentation updates.
- **Baseline testing:** validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
- **Compliance testing –legality/government:** related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
- **Documentation testing:** As per the IEEE Documentation describing plans for the testing of a system or component. Types include test case specification, test incident report, test log, test plan, test procedure, test report.
- **Endurance testing:** testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. Ex: a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
- **Load testing:** to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system's behavior under both normal and at peak conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are in creased then how much CPU, memory will be consumed, what is the network and bandwidth response time.

- **Performance testing:** to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes like it can demonstrate that the system meets performance criteria. It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
- **Compatibility testing:** testing of the application or the product built with the computing environment. It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.
- **Security testing:** to check that whether the application or the product is secured or not. It is a process to determine that an information system protects data and maintains functionality as intended. Authorization/hacking
- **Scalability testing:** for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.
- **Volume testing:** testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application's performance on it.
- **Stress testing:** testing beyond normal operational capacity, often to a breaking point, in order to observe the results. It is a form of testing that is used to determine the stability of a given system. Greater emphasis on robustness, availability, and error handling under a heavy load.
- **Recovery testing:** to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when 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 got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.
- **Internationalization testing and Localization testing:** process of designing a software application so that it can be adapted to various languages and regions without any changes. Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

Structural testing (Testing of software structure/architecture)

- Testing of the structure of the system or component.
- a.k.a 'white box' or 'glass box' or 'clear-box testing' because we are interested in what is happening 'inside the system/application'.

- Testers are required to have the knowledge of the internal implementations of the code. How it works? How the s/w does it? Like how loops in the software are working? Different test cases may be derived to exercise the loop once, twice, and many times.
- Developers use structural testing in component testing and component integration testing, especially where there is good tool support for code coverage.

Confirmation testing ~ re-testing

- When a test fails because of the defect then that defect is reported and a new version of the software is expected that has had the defect fixed. In this case we need to execute the test again to confirm that whether the defect got actually fixed or not. This is known as confirmation testing and also known as re-testing.
- Important to ensure that the test is executed in exactly the same way it was the first time using the same inputs, data and environments.

Regression testing

- To verify that modifications in the software or the environment have not caused any unintended adverse side effects and that the system still meets its requirements.
- Mostly automated because in order to fix the defect the same test is carried out again and again and it will be very tedious to do it manually.
- Executed whenever the software changes, either as a result of fixes or new or changed functionality.
- During confirmation testing the defect got fixed and that part of the application started working as intended.
- But there might be a possibility that the fix may have introduced or uncovered a different defect elsewhere in the software. The way to detect these '**unexpected side-effects**' of fixes is to do regression testing.