**Functional Testing**

Testing the functionality of an application according to customer requirements. Here, we check if all modules/functionalities are working fine or not.

**Importance of Functional Testing:** Functional testing focuses on verifying that a software application's features and functionalities work as intended and meet the specified requirements. It is an integral part of the software development life cycle, contributing to the overall success of the software product.

**Reasons why Functional Testing is so important.**

● Functional Testing verifies the functionality of core features.

● Functional Testing ensures that the software is developed according to customer requirements.

● Functional Testing helps to identify defects related to software functionality in the early stage itself.

● By conducting this Functional Testing we can ensure that software is user friendly and contributes to the overall user experience.

● This includes user interaction scenarios, by conducting Functional Testing, we can ensure that the users can interact with the software as intended.

Functional Testing are of 3 types:

1) Exhaustive Testing - Nothing testing the functionality of the software by entering more number of inputs. This is not preferred as it consumes a lot of time.

2) Under Testing - Testing the functionality of the software by entering an insufficient set of data. This testing is also not preferred because if we enter insufficient data, we will be missing potential bugs.

3) **Optimized Testing** - Testing the functionality of the software by entering the data which makes sense and which is in accordance with customer requirements is called Optimized Testing. This is preferred testing.

**Optimized Testing is of 2 Types: Positive Functional Testing and negative Functional Testing.**

● **Positive Functional Testing** - Testing the functionality of the application by entering only valid data or data which is in accordance with customer requirements is called Positive Functional Testing.

● **Negative Functional Testing** - Testing the functionality of the software by entering only invalid data or data which is not in accordance with the customer requirements is called

Negative Functional Testing.

Let's write some scenarios:

Assume that we need to test the password text field. We have a requirement that it should accept only alphabets and numbers, special characters are not allowed. Write Functional scenarios.

**Positive Functional scenarios:**

1) To verify that the password text field accepts a combination of alphabets and numbers. 2) To verify that the password text field accepts only alphabets.

3) To verify that password text field accepts only numbers

**Negative Functional scenarios:**

1) To verify that the password textfield does not accept a combination of alphabets and special characters.

2) To verify that the password textfield does not accept a combination of numbers and special characters.

3) To verify that when the password text field is blank, check whether the error message is displayed or not.

(Above example needs to be written with specific alphabets and numbers as test data in the test case template and needs to be tested accordingly.)

**INTEGRATION TESTING**

Testing the data flow between two or more modules is called Integration Testing.

**Importance of Integration Testing**: Integration testing is a critical phase in the software testing process that focuses on verifying the interactions and interfaces between different components or systems within a software application.

**Integration testing is important because of following reasons:**

● It ensures proper data flow - Integration testing validates the proper flow of data between different modules and ensures that data is transferred accurately. Data consistency and accuracy are crucial for the correct functioning of the entire system. Integration testing helps uncover data-related issues that may arise when different components exchange information.

● Validating business workflows - Users interact with software applications through specific workflows. Integration testing ensures that these workflows, which often involve multiple

components, are seamless and error-free.

● It helps to detect Interface issues - Integration testing helps identify issues related to the interaction between different modules, components, or services.

● It ensures System Stability - By verifying the integration of components, integration testing helps ensure that the system operates cohesively and reliably, meeting performance and stability requirements.

**There are 2 types of Integration Testing:**

a) Incremental Integration Testing

b) Non-incremental Integration Testing

a) **Incremental Integration Testing –** Incrementally adding the modules and testing the data flow between the modules is called Incremental Integration Testing.

It is of 2 types: **Top Down approach and Bottom-up approach**

1) **Top Down approach** – Incrementally adding the modules and testing the data flow between the modules and ensuring that the modules we are adding are the child of the previous module is called the Top Down approach. Here data will flow from top to bottom.

2) **Bottom-up approach** – Incrementally adding the modules and testing the data flow between the modules and ensuring that the modules we are adding are the parent of the previous module is called the Bottom-up approach. Here data will flow from bottom to top.

b) **Non-Incremental Integration Testing** – Here we randomly test the data flow between all other modules. We go for this testing, when we don’t know which parent and child module or when requirements are complex to understand, we can do Non-Incremental Integration Testing.

**Positive Integration Testing scenario on Gmail:**

● Login as a user, click on compose. Enter some information, now click on

cancel. Now click on draft option, check that the information written so far

should be displayed.

**Negative Integration Testing scenario on Gmail:**

Login as user, click on compose. Enter some information and click on

cancel. Click on Trash and check if the entered details are displayed

**SYSTEM TESTING**

**It** is an end to end testing conducted by the Test engineers in a testing environment which is similar to a production environment.

**What is End to end testing?**

Navigating through all the features and checking whether the end feature is working as expected or not is called end to end testing.

End-to-end testing is a comprehensive software testing approach that evaluates the entire software application from start to finish, including all integrated components and external dependencies. The primary goal of end-to-end testing is to ensure that the entire system functions as expected and meets the specified business requirements. This type of testing simulates real-world scenarios to validate the software's behavior in a production-like environment.

In this testing, we test all the end-to-end features and check whether it is according to the customer requirements. It is very essential to do end to end testing because customers in real time do verify end to end features of an application and if it is working fine or not.

**System testing is a critical phase in the software testing process that evaluates the entire software system as a whole. It is conducted after integration testing and before acceptance testing, focusing on verifying that the integrated system functions according to the specified requirements.**

**Importance of System testing:**

● System testing verifies the entire system including all integrated components and functions as intended.

● System testing identifies all System level defects, that may surface only when complete system functionalities are assembled.

● It verifies all individual components when integrated collectively fulfill the intended business and functional requirements.

● System testing ensures end to end business scenarios are met for better user experience. **Positive System testing scenario on Flipkart:**

● Log in to flipkart, click on mobiles feature, click on samsung, select one of the samsung mobile. Now click on buy now, select the payment option as phonepe, complete the payment. Click on the orders and check the product details are displayed.

**Negative System testing scenario on Flipkart:**

● Login to flipkart, search for fastrack watches, select one of the watches. Click on buy now, click on continue, select the payment option as debit card. Generate one time password. Enter the wrong OTP and check if the payment is done.

**Smoke & Sanity Testing**

**Smoke Testing** also called as Positive Testing, Build verification Testing, Sanity Testing

Now, let’s assume, we have in 1 Sprint which has duration of 10 days and Release duration is 30 days and we need to test 3 modules A, B and C. Now, Testing team has planned in such a way that they have divided 3 modules into duration of 4,3, 3 days to complete the task. Now, let's say we completed testing for module A, now on 5th day we just started testing for module B, now there is a blocker defect and because of this defect we are unable to proceed further i,e,. user is blocked. Now, TE will inform the developer to fix it ASAP else the release will get postponed. Developer will say he needs at least 1 day to fix this issue. Once developers fix this issue, we need to start testing again from functional testing. If we go with this flow, it will take 35 days to release the build to the customer instead of 30 days. Customer might not accept this issue as it will affect his business. Hence to avoid these kinds of issues we should do Smoke testing in the beginning stage itself.

**So, remember Smoke Testing is the first testing done by Test Engineers. Followed by Functional Testing, Integration Testing, System Testing, Regression Testing etc.**

**What is Smoke Testing?**

It is High level testing conducted before doing rigorous testing or other types of testing to ensure that there is no blocker or critical defects that affects the customer business workflow.

**Q) Why Smoke Testing is necessary?**

● Because it helps in early detection of Blocker and critical bugs

● It verifies that the latest build or release is stable enough for further testing. If the build passes the smoke test, it indicates that the basic components are intact and functioning as expected.

● It ensures basic functionalities of the software is working fine

● It saves time and resources as we detect the blocker and critical bugs in the early stage itself.

**Q) When to do Smoke Testing?**

● Whenever new build is released

● Before conducting rigorous testing i,e,. Functional, Integration, Regression, End to end testing etc.

**What is Sanity Testing?**

It is also a kind of Smoke testing but it is conducted usually when we find a bug, then once retesting is done, we do sanity testing, meaning we check the application in depth and verify all dataflows if it is working correctly or not. This is called Sanity Testing.

It is basically a focused and narrow verification process performed on a specific functionality or a small set of functionalities within a software application. The primary purpose of sanity testing is to quickly check whether a particular aspect of the software is working correctly after changes, updates, or bug fixes.

**Q) Why is Sanity Testing necessary?**

● It helps to quickly validate specific/key functionalities of a software after some changes are done

● It helps to save time and resource as we focus on critical functionalities ● It acts like a build verification where it ensures if the current build is stable enough for further testing or not.