Levels of Testing

1) **Unit Testing**
2) **Integration Testing**
3) **System Testing**
4) **User Acceptance Testing**

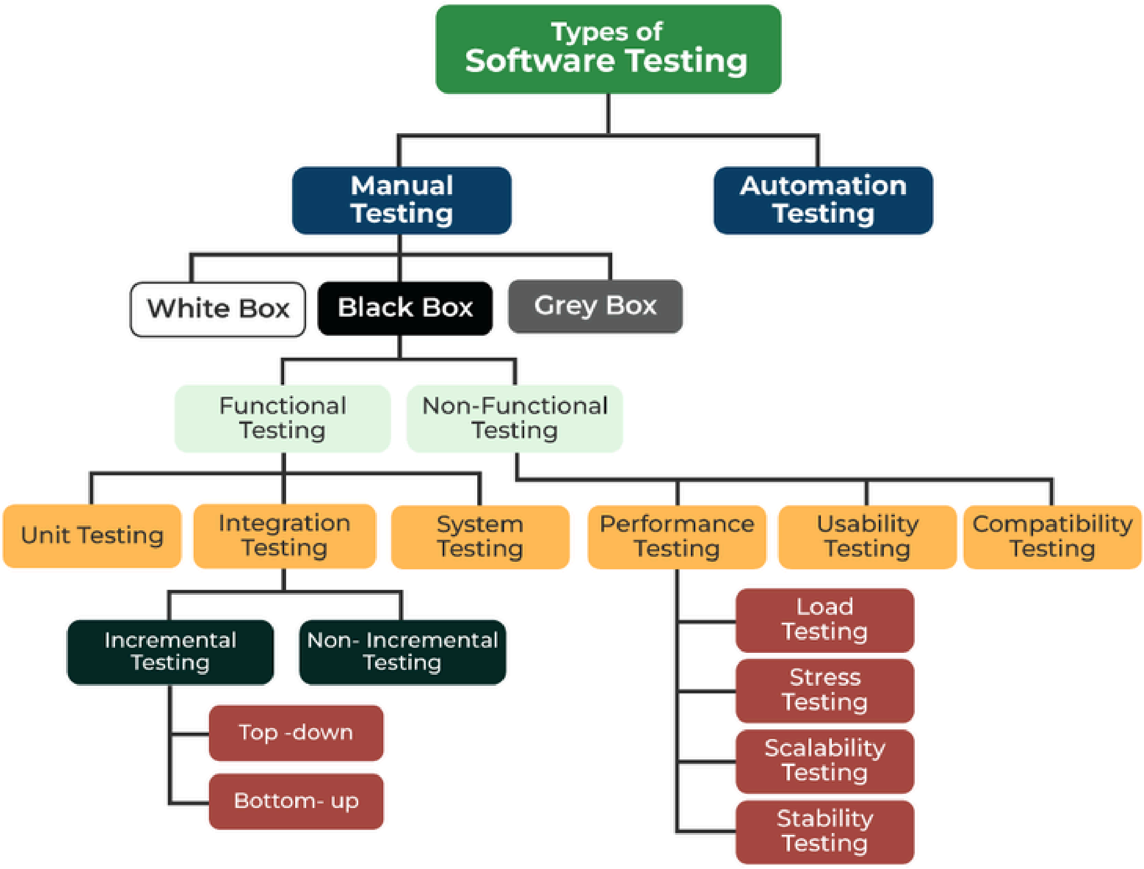
Unit Testing means testing small piece of a code/component

Integration Testing means testing the dataflow between the modules

System testing means testing overall functionality of the application

User Acceptance testing means customer testing to check if the software is ready for production or not

TYPES OF TESTING



Blackbox testing

Testing the application when it is already developed and ready for Testing or when code is not invisible is called as Black Box Testing.

It is performed by Testers.

Types of blackbox testing are:

1) Functional Testing
2) Non Functional Testing

Types of Functional Testing:

1) Component Testing
2) Integration Testing

3) System Testing
4) Smoke Testing
5) Sanity Testing
6) Regression Testing
7) User Acceptance Testing

Component Testing

Testing each and every component of an application rigorously or thoroughly against customer requirement specification is called Component Testing.

It is also called Functional Testing by Testers.

Why Component testing - To check whether each and every component is working as per the customer requirement.

Types of Component Testing

1. Over Testing/Exhaustive Testing - Testing the component by entering more data or information than required is called Over Testing or Exhaustive Testing.

2. Under Testing - Testing the component by entering insufficient set of data is called Over Testing .

3. Optimised Testing - Testing the component by entering data which makes sense is called Optimised Testing.

Types of Optimised Testing

Positive Testing - Testing the application by entering data according to customer requirement is called Positive Testing. Or

Testing the application by entering valid data is called as Positive Testing

Negative Testing - Testing the application by entering data against customer requirements is called Negative Testing. Or

Testing the application by entering invalid data is called Negative Testing.

Scenario: (Component - Password text field)

Requirement:

1) It should have a placeholder
2) It should accept min 10 characters and maximum 20 characters.
3) Space is allowed, numbers and special characters are not allowed.

Positive scenarios:

1) To verify that the password text field has a placeholder.

2) To verify that the password text field accepts 10 characters.

3) To verify that the password text field accepts 11 characters.

4) To verify that the password text field accepts 19 characters.

5) To verify that the password text field accepts 20 characters.

6) To verify that the password text field accepts a combination of 10 characters along with a space.

Negative scenarios:

1) To verify that the password textfield does not accept 9 characters.

2) To verify that the password textfield does not accept 21 characters.

3) To verify that the password textfield does not accept numbers.

4) To verify that the password textfield does not accept special characters.

5) To verify that the password textfield does not accept combinations of alphabets and numbers.

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

7) To verify that the password textfield does not accept combinations of alphabets, numbers and special characters.

8) To verify that the password textfield does not proceed after the blank value.

9) To verify that the password textfield does not accept a combination of alphabets and multiple spaces.

10) To verify that the password textfield does not accept combinations of alphabets with a space, numbers and special characters.

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.

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.

(Here, in the absence of child module, stubs will be used)

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.

(Here, in the absence of parent module, drivers will be used)

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 or when there is issue with time duration.

Positive Integration Testing scenario on Gmail:

- Login as a user, click on compose. Enter some information, now click on cancel. Now click on the 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.

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 fulfil 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 a debit card. Generate one time password. Enter the wrong OTP and check if the payment is done.