MODULE 2 [MANUAL TESTING]

## Q1 : What is Exploratory Testing ?

ANS :Exploratory Testing is a type of software testing where Test cases are not created in advance but testers check system on the fly.

.They may note down ideas about what to test before test execution.

.The focus of exploratory testing is more on testing as a “thinking” activity.

. This testing is useful when requirement documents are not available or partially available.

# Q2 : What is traceability matrix ?

ANS : There are three types of traceability matrix.

. A requirements traceability matrix (RTM) is a little more complex and involves tracking the business requirements against the functional requirements.

.This requires the mapping of technical needs of a project against the business requirements.

. Forward traceability which is used to check if the project progresses in the right direction for the right product

. It helps uncover if the requirements are applied to the product.

. Second is backward or reverse traceability that helps to assess whether the product is on track.

. This helps to keep the project from suffering from scope drift.

. Finally, bi-directional traceability ensures that all requirements are covered by test cases and then helps you to analyse the impact of changes to requirements .

### Q3 : What is Boundary value testing ?

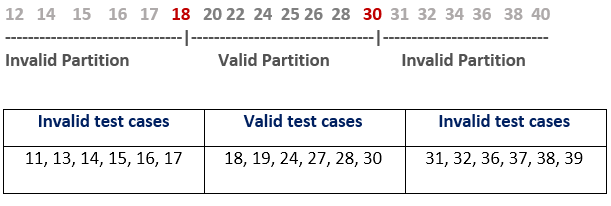
ANS : Boundary value analysis is one of the widely used case design technique for black box testing.

.It is used to test boundary values because the input values near the boundary have higher chances of error.

.Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

.Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.

FOR EXAMPLE : Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.



# Q4 : What is Equivalence partitioning testing ?

ANS : The Equivalence Partitioning Testing technique is one of the popular software testing techniques.

.It is used to reduce the number of test cases by creating boundaries that ensure code paths are not executed for specific input data values.

. [Equivalence Partitioning](https://www.softwaretestingo.com/equivalence-partitioning-testing/), helps you divide input data into several partitions based on the characteristics of the inputs, each having the same set of boundary values for one variable, but all having different sets of boundary values for another variable.

. In other words, each partition is a collection of inputs with the same value(s) for one variable and different values for some other variable.

. Creating partitions by splitting every set of input values into two sub-sets so each partition contains input values with different boundaries for one variable but the same boundary for another variable.

# Q5 : What is Integration testing ?

ANS : Integration testing is known as the second level of the [software testing process](https://www.simplilearn.com/tutorials/devops-tutorial/fundamentals-of-software-testing), following unit testing. Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed.

.As a rule, the usual software project consists of numerous software modules, many of them built by different[programmers.](https://www.simplilearn.com/how-to-become-programmer-article) Integration testing shows the team how well these disparate elements work together. After all, each unit may function perfectly on its own, but the pressing question is, “But can they be brought together and work smoothly.

. So, integration testing is the way we find out if the various parts of a software application can play well with others.

Q6: What is component testing?

**ANS .** Component testing is a type of white box testing where you validate an individual component of the application before testing the entire application.

. Component testing finds bugs and verifies the functionality of software modules/programs which are individually testable.

.Component testing is also known as program or module testing.

.It results in the detection of defects in software modules and checks the functioning of the software.

. Testing each component separately is termed as component testing.

# Q7 : What is functional system testing?

ANS . Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements.

.Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user’s expectations.

.The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

. Some functional testing examples are

. Can users successfully log in to the application once they provide legitimate credentials?

.Does the payment gateway reject the input and display an error message when a user keys in an invalid credit card number?

.Do inputs to the “Add New Record” screen successfully add and save a new record to the database?

**Unit Testing**: This is performed by developers who write scripts that test if individual components/units of an application match the requirements. This usually involves writing tests that call the methods in each unit and validate them when they return values that match the requirements.

**Smoke Testing**: This is done after the release of each build to ensure that software stability is intact and not facing any anomalies.

**Sanity Testing**: Usually done after smoke testing, this is run to verify that every major functionality of an application is working perfectly, both by itself and in combination with other elements.

**Regressing testing** .This test ensures that changes to the codebase (new code, debugging strategies, etc.) do not disrupt the already existing functions or trigger some instability.

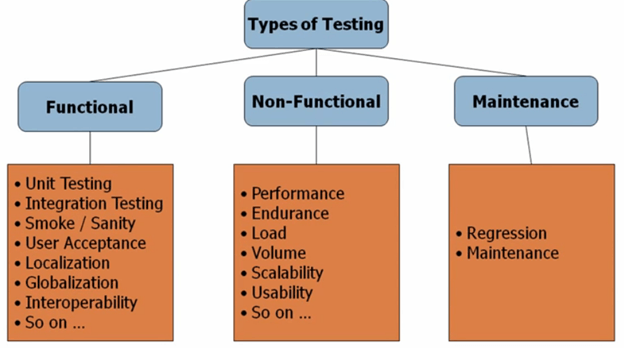
**Integration Testing**: If a system requires multiple functional modules to work effectively, integration testing is done to ensure that individual modules work as expected when operating in combination with each other. It validates that the end-to-end outcome of the system meets these necessary standards.

## Q8 . What is Non-Functional Testing?

ANS: **Non-Functional Testing** is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

.An excellent example of non-functional test would be to check how many people can simultaneously login into a software .

.Non-functional testing is equally important as functional testing and affects client satisfaction.



Q9 .What is GUI Testing?

ANS . GUI testing is described as the testing of the application under the Test system graphical user interface.

.Graphical Testing consists of screen checks using controls such as icons, buttons, and menus with all types of the bar such as dialogues, menu bar, toolbar, Windows, etc. Basically, we can say that what we see is GUI.

.For a computer application, there are two kinds of interfaces.

.The command-line interface is the one that answers to the command text and the computer.

. On the other hand, [GUI](https://www.educba.com/what-is-gui/) is a graphical user interface in which you use pictures rather than text to communicate with the computer.

# Q10 . What is load testing ?

ANS. Load Testing is a type of**performance testing** in which any software, application, or website is subjected to a realistic load.

. Your application could fail badly in real-world scenarios if you don't have it.

.When your software development project is nearing completion, there’s one test that’s essential to understanding its readiness for deployment: load testing.

.  During a load test, testing software will measure the capacity of your web application via transaction response times.

. Load testing helps developers identify issues like system lag, slow page load times, or crashes when different levels of traffic are accessing the application during production rather than post-launch.

# Q11 : What is stress Testing ?

ANS : Stress testing a Non-Functional testing technique that is performed as part of performance testing. During stress testing, the system is monitored after subjecting the system to overload to ensure that the system can sustain the stress.

The recovery of the system from such phase (after stress) is very critical as it is highly likely to happen in production environment.

. It allows the test team to monitor system performance during failures.

.To verify if the system has saved the data before crashing or NOT.

.To verify if the system prints meaning error messages while crashing or did it print some random exceptions.

.To verify if unexpected failures do not cause security issues.

# Q12 : What is white box testing and list the types of white box testing ?

ANS: White Box Testing is a software examining technique that involves testing the product's underlying structure, design, and coding in order to verify input-output flow and improve design, usability, and security

. White box testing is also known as Clear box testing, Open box testing, transparent box testing, Code-based testing, and Glass box testing since the code is visible to the testers.

## . Example of White Box Testing

**Code −**

Printme (int a, int b) {------------ it is a function

int result = a+ b;

If (result > 0)

Print ("Positive", result)

Else

Print ("Negative", result)

} ----------- End of the code

. Types Of White box testing

**1.Unit Testing**:

Unit testing is a fairly straightforward white box test. It tests the structure of a program at the smallest level. The idea is to check whether individual sections of source code are working per design or not. It can also be used to see if specific points in the code are reached or whether any lines are executed during program execution.

For example, a developer has written a C# method called ComputeTax(). A unit test would call this method with different parameters to see if the result is correct each time.

**2. Smoke testing**

Smoke testing is an initial white box testing that may be performed when delivering a large project or releasing software for internal use. The purpose of smoke testing is to ensure basic functionality. This involves a cursory check to see if the “heart” of a program is working. This usually involves black box testing, but it doesn’t have to be.

For example, you may run a smoke test on a newly developed software program to ensure that the program’s main functions work as expected.

**3. Sanity Testing**

Sanity testing is a more in-depth type of white box testing used to verify a system’s basic functionality after major changes, such as code refactoring or updates to the database. To sanity test a system, you would run all the unit tests and smoke tests that you normally would, but you would also test specific use cases known to break the system.

For example, after making some changes to a web application’s code, you might sanity test the application by loading it in a browser and performing some basic tasks, like entering data into form fields and clicking on buttons.

**4. Code coverage testing**

 Code coverage testing is used to measure the degree to which the source code of a program is tested. This type of white-box testing is usually done by tools that “trace” the program’s execution flow and generate reports on what lines of code were executed.

For example, you can use code coverage tools to determine what percentage of your source code is being executed during the [testing of a program](https://www.hikeqa.com/mobile-app-testing-services/manual-testing-v-s-automated-testing/). This can help spot code sections that are not essential to the program’s success or are difficult to maintain.

**5. Data flow testing**

 Data flow testing is used to determine how data flows through an application by checking individual paths through all components in an application. It is performed by tracing the path of data from input to output.

For example, you can use data flow testing to see if certain inputs result in the expected outputs. This can help you to identify errors in your application.

**6. Path testing**

Path testing is a type of data flow testing that checks the flow through an application along a particular path.  Testing aims to ensure that data travels along expected paths through an application, regardless of its starting point or the inputs used to produce outputs.

For example, one might perform path testing by entering various incorrect inputs into an application and comparing these with the results obtained when using the correct inputs. This would help identify any differences between the expected outputs and those that are reproduced.

**7. Statement testing**

Statement testing is a type of data flow testing that checks whether individual lines of source code perform as expected or not. This type of white box testing verifies the behavior of each line of code in an application.

For example, you can use statement testing to see if a particular line of code is executed when a particular condition is met. This can help point out problems with the code.

## Q13 : What is black box testing? What are the different black box testing techniques?

ANS : Black box testing techniques apply to all levels of testing, as well as functional and non-functional testing types.

.There are four main black box testing techniques: equivalence partitioning, boundary value analysis, decision table testing, and state transition testing.

. The test cases which are designed to test a system plays an important role in testing. The way they are created and the scenarios they cover should be taken into consideration.

Testers can create requirement specification document by using the below techniques:

[Equivalence Partitioning](https://www.educba.com/equivalence-partitioning/)

Boundary Value Analysis

Decision Table Testing

State Transition Testing

Error Guessing

Graph-based Testing Methods

Comparison Testing

Use Case Technique

Following are the Techniques explained below:

1. Equivalence Testing

This technique divides the input values which are provided to the software into different groups or classes. This is done on the basis of the output which will be coming as an outcome. This technique is also known as Equivalence Class Partitioning. By doing this we save the effort of giving different inputs. Instead, we give one value to the group or class to test the outcome for that group or class. This helps in improving the test coverage and in turn reducing the rework. The time is also saved as no separate inputs are to be given. Input for each class is sufficient.

Let us take an example of marks that students score. If a student scores above75% then he/she has secured First class with Distinction. Similarly, if the score is between 60% to 75% then he/she has secured First Class. If the score is between 50% to 60% then Second Class. If the score is between 40% to 50% then Pass class, else fail. Here there will be four classes. These test cases are formed and it is made sure that all possibilities are hence covered. Hence testing with any values in this set is sufficient.

2. Boundary Value Analysis

Here the focus is on the values which are present at the boundaries. This is because usually there are many issues found when it comes to testing with values that focus on boundaries. Boundary focuses on values near the limit where the behaviour of the system changes. In boundary value analysis both inputs, which are valid and invalid are to be tested.

For example, if we want to test values that range from 1 to 100 then we should check if how the program works for values like 1-1, 1+1, 1, 100-1, 100+1, etc. This helps in saving time again as we can only check for values like 0, 1, 2, 99, 100 and 101.

3. Decision Table Testing

Whenever there are logical conditions or decision-making steps then this technique is to be used. These can be like if a particular condition is not satisfied then Action A should be performed, else Action B is to be performed. The tester needs to identify the input and actions which are to be performed based on the conditions. A decision table is created based on these. Consider an example where an odd number of vehicles are allowed only on Monday, Wednesday, Friday and Sunday, while even a number of vehicles are allowed on Tuesday, Thursday and Saturday. In this case, there are two conditions and two actions. Condition 1 being odd vehicles and Condition 2 being even vehicles. The actions being the days when these vehicles can be on the roads. The total number of test cases, in this case, can be four and hence the decision table can be derived accordingly.

4. State Transition Testing

In this technique, the test case tries to test the system under different states. This state can change depending upon different conditions or events. When a particular event occurs then these scenarios can be tested.

5. Error Guessing

This technique is mainly based on experience. Once a tester has experience working on any application its behavior and functionalities are known to him/her. This is a way through which many issues can be found out. By using this experience, it is easy for the testers to guess where most developers are prone to make mistakes. These can be handling null values, accepting the submit button without any value, uploading a file without any attachment, uploading a file with less than or more than the limit size specified, etc.

6. Graph-based Testing

Each application is built by using some objects. All the objects which are used are noted and a graph is prepared. From this graph, the relationship of every object is identified, and test cases are written accordingly.

7. Comparison Testing

In this technique, [different versions](https://www.educba.com/tableau-versions/) of the same software are used and then compared in order to test the entire system. The behaviour is noted and compared for all versions and any deviations are noted.

8. Use Case Technique

This technique is used to identify all the [test cases in use](https://www.educba.com/test-cases-vs-test-scenario/) as per the system. All scenarios are noted which help in understanding the complete functionality of each function in an end to end way. The test cases should have cases that cover all scenarios from beginning to end as per the system usage.

# Q14 : Mention what are the categories of defects?

ANS : 1) Wrong: If the requirements are implemented incorrectly, then they are stated as Wrong defects.

2) Missing: If the requirement is not done which is given by the customer. ...

3) Extra: If a requirement is not given by the end user and if it is done,then it is called as an extra defect.

# Q15 : Mention what bigbang testing is?

ANS: Big Bang Integration Testing is an integration testing strategy, wherein all units are linked at once, which results in a complete and efficient system.

. In this type of integration testing all the components as well as the modules of the software are integrated simultaneously, after which everything is tested as a whole.

.During the process of big bang integration testing, most of the developed modules are coupled together to form a complete software system or a major part of the system, which is then used for integration testing.

.This approach of software testing is very effective as it enables software testers to save time as well as their efforts during the integration testing process.

# Q16 : What is the purpose of exit criteria?

ANS : Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

Verify if All tests planned have been run.

Verify if the level of requirement coverage has been met.

Verify if there are NO Critical or high severity defects that are left outstanding.

Verify if all high risk areas are completely tested.

Verify if software development activities are completed within the projected cost.

Big Bang Integration Testing is an integration testing strategy, wherein all units are linked at once, which results in a complete and efficient system. In this type of integration testing all the components as well as the modules of the software are integrated simultaneously, after which everything is tested as a whole. During the process of big bang integration testing, most of the developed modules are coupled together to form a complete software system or a major part of the system, which is then used for integration testing. This approach of software testing is very effective as it enables software testers to save time as well as their efforts during the integration testing process.