ASSIGNMENT – 2

FUNCTIONAL TESTING

DEFINITION :

Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications.

Functional testing is basically defined as a type of testing that verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned with the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output, and comparing the actual output with the expected output. This testing focuses on checking the user interface, APIs, database, security, client or server application, and functionality of the Application Under Test. Functional testing can be manual or automated.

Purpose of Functional Testing

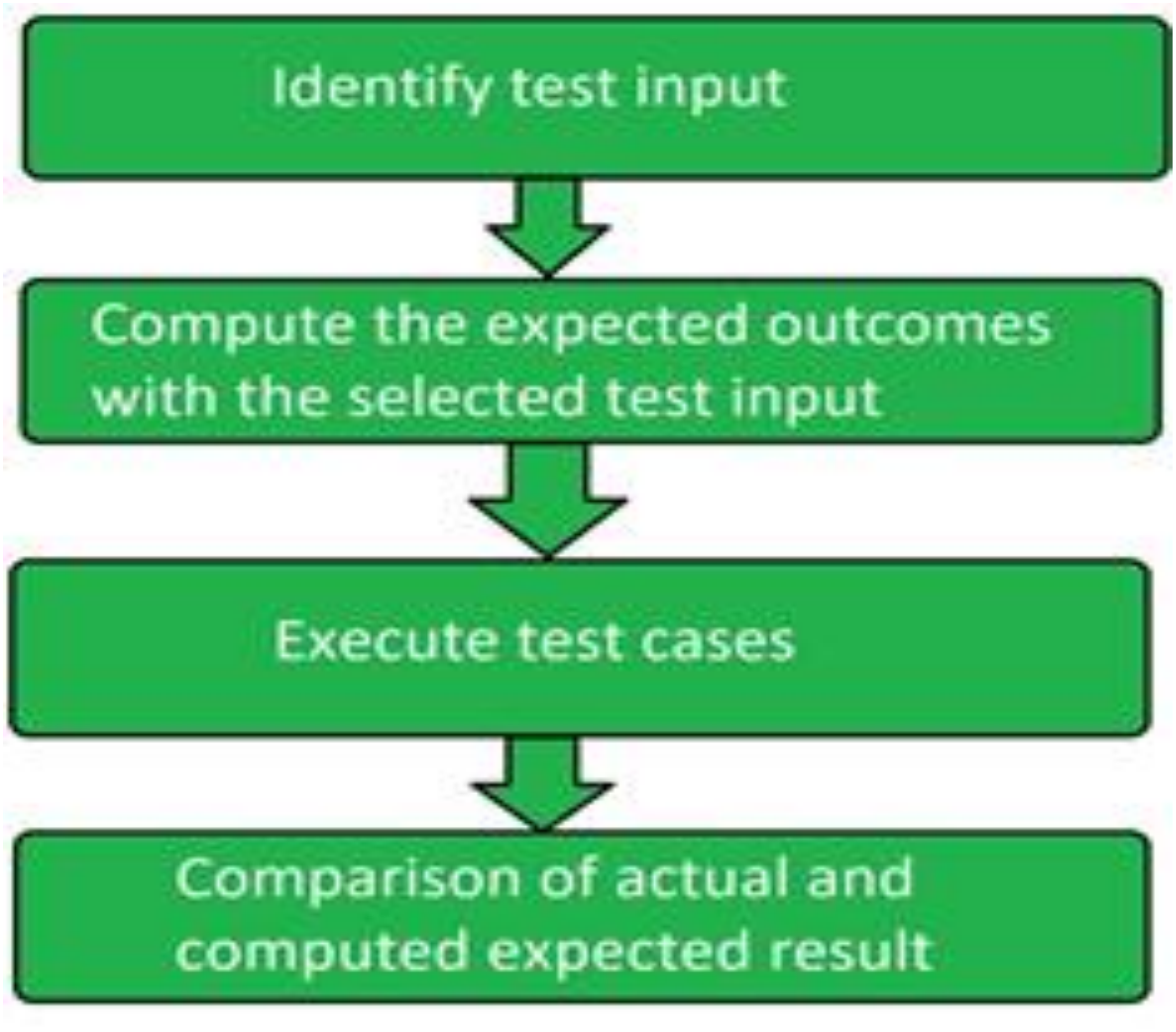
Functional testing mainly involves black box testing and can be done manually or using automation. The purpose of functional testing is to:

- **Test each function of the application:** Functional testing tests each function of the application by providing the appropriate input and verifying the output against the functional requirements of the application.
- **Test primary entry function:** In functional testing, the tester tests each entry function of the application to check all the entry and exit points.
- **Test flow of the GUI screen:** In functional testing, the flow of the GUI screen is checked so that the user can navigate throughout the application.

Functional Testing Process

Functional testing involves the following steps:

1. **Identify test input:** This step involves identifying the functionality that needs to be tested. This can vary from testing the usability functions, and main functions to error conditions.
2. **Compute expected outcomes:** Create input data based on the specifications of the function and determine the output based on these specifications.
3. **Execute test cases:** This step involves executing the designed test cases and recording the output.
4. **Compare the actual and expected output:** In this step, the actual output obtained after executing the test cases is compared with the expected output to determine the amount of deviation in the results. This step reveals if the system is working as expected or not.


Identify test input
Compute the expected outcomes with the selected test input
Execute test cases
Comparison of actual and computed expected result

UNIT TESTING :

DEFINITION :

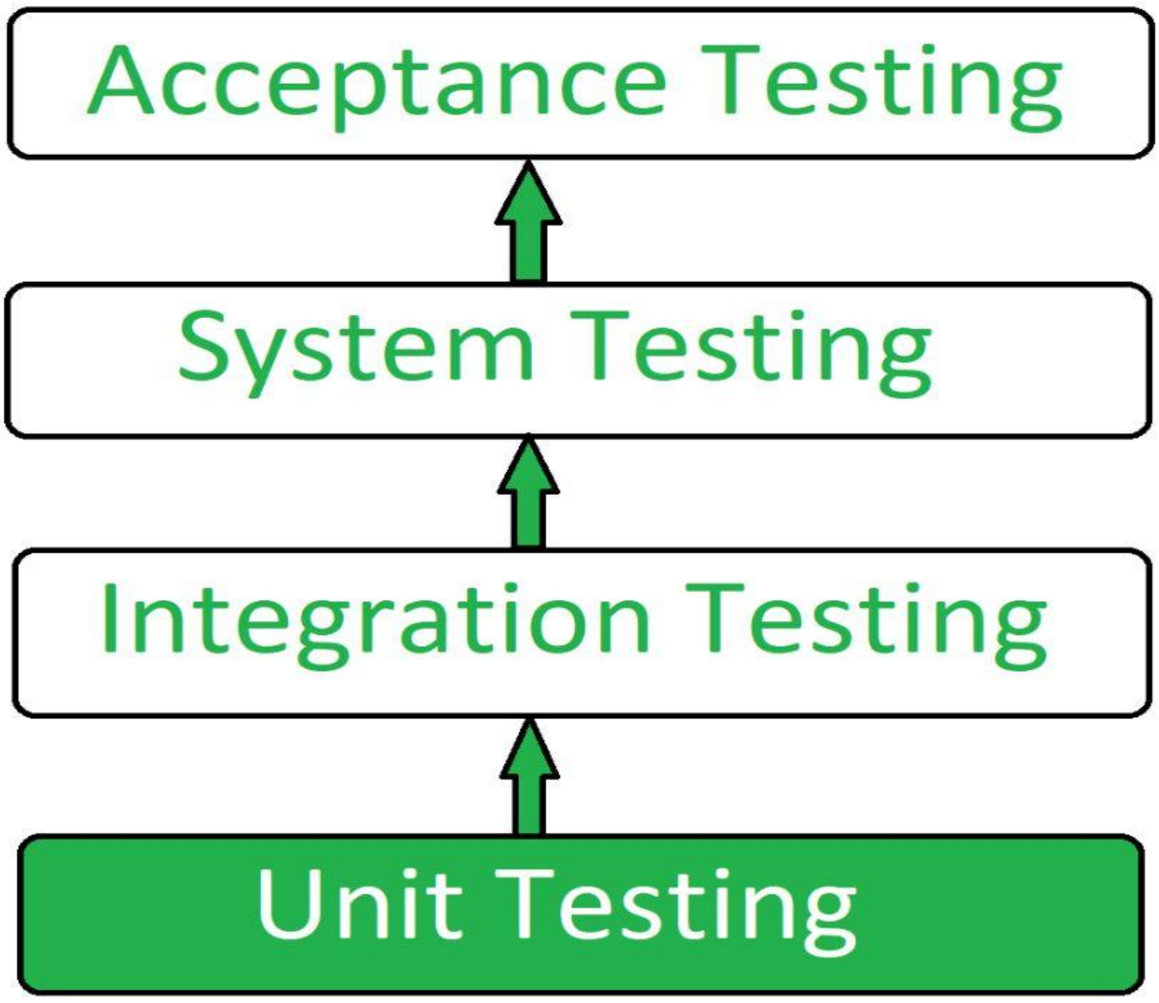
Unit testing is a type of software testing that focuses on individual units or components of a software system. The purpose of unit testing is to validate that each unit of the software works as intended and meets the requirements. Unit testing is typically performed by developers, and it is performed early in the development process before the code is integrated and tested as a whole system.

Unit Testing is a software testing technique using which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application.

Objective of Unit Testing:

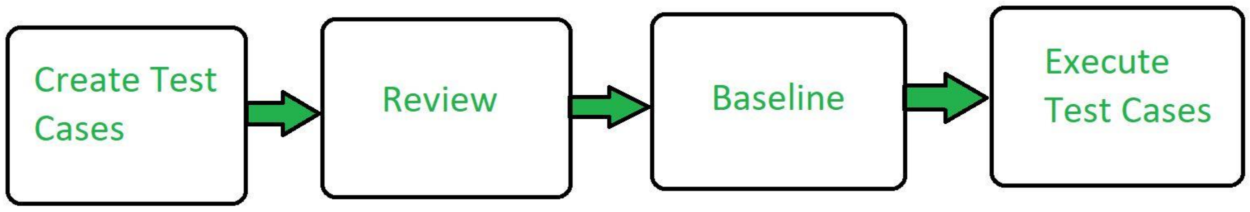
The objective of Unit Testing is:

1. To isolate a section of code.
2. To verify the correctness of the code.
3. To test every function and procedure.
4. To fix bugs early in the development cycle and to save costs.
5. To help the developers understand the code base and enable them to make changes quickly.
6. To help with code reuse.



Types of Unit Testing:

There are 2 types of Unit Testing: **Manual**, and **Automated**.
Workflow of Unit Testing:



Unit Testing Techniques:

There are 3 types of Unit Testing Techniques. They are

1. **Black Box Testing:** This testing technique is used in covering the unit tests for input, user interface, and output parts.
2. **White Box Testing:** This technique is used in testing the functional behavior of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.
3. **Gray Box Testing:** This technique is used in executing the relevant test cases, test methods, and test functions, and analyzing the code performance for the modules.

Unit Testing Tools:

Here are some commonly used Unit Testing tools:
1. Jtest
2. Junit
3. NUnit
4. EMMA
5. PHPUnit

Advantages of Unit Testing:

1. Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
2. Unit testing allows the programmer to refine code and make sure the module works properly.
3. Unit testing enables testing parts of the project without waiting for others to be completed.
4. Early Detection of Issues: Unit testing allows developers to detect and fix issues early in the development process before they become larger and more difficult to fix.

Disadvantages of Unit Testing:

1. The process is time-consuming for writing the unit test cases.
2. Unit Testing will not cover all the errors in the module because there is a chance of having errors in the modules while doing integration testing.
3. Unit Testing is not efficient for checking the errors in the UI(User Interface) part of the module.

4. It requires more time for maintenance when the source code is changed frequently.
5. It cannot cover the non-functional testing parameters such as scalability, the performance of the system, etc.

Integration testing

DEFINITION :

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface.

Integration testing can be done by picking module by module. This can be done so that there should be a proper sequence to be followed. And also if you don't want to miss out on any integration scenarios then you have to follow the proper sequence. Exposing the defects is the major focus of the integration testing and the time of interaction between the integrated units.

Integration test approaches – There are four types of integration testing approaches. Those approaches are the following:

1. **Big-Bang Integration Testing –** It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module testing.

   Big-bang integration testing is a software testing approach in which all components or modules of a software application are combined and tested at once.

Advantages:

1. It is convenient for small systems.
2. Simple and straightforward approach.
3. Can be completed quickly.
4. Does not require a lot of planning or coordination.
5. May be suitable for small systems or projects with a low degree of interdependence between components.

Disadvantages:

1. There will be quite a lot of delay because you would have to wait for all the modules to be integrated.

2. High-risk critical modules are not isolated and tested on priority since all modules are tested at once.
3. Not Good for long projects.

2. Bottom-Up Integration Testing – In bottom-up testing, each module at lower levels are tested with higher modules until all modules are tested. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower-level modules.

Advantages:

- In bottom-up testing, no stubs are required.
- A principal advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.
- It is easy to create the test conditions.
- Best for applications that uses bottom up design approach.
- It is Easy to observe the test results.

Disadvantages:

- Driver modules must be produced.
- In this testing, the complexity that occurs when the system is made up of a large number of small subsystems.
- As Far modules have been created, there is no working model can be represented.

3. Top-Down Integration Testing – Top-down integration testing technique is used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First, high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

Advantages:

- Separately debugged module.
- Few or no drivers needed.
- It is more stable and accurate at the aggregate level.
- Easier isolation of interface errors.
- In this, design defects can be found in the early stages.
-

Disadvantages:

- Needs many Stubs.
- Modules at lower level are tested inadequately.
- It is difficult to observe the test output.
- It is difficult to stub design.

4. **Mixed Integration Testing –** A mixed integration testing is also called sandwiched integration testing. A mixed integration testing follows a combination of top down and bottom-up testing approaches. In top-down approach, testing can start only after the top-level module have been coded and unit tested. In bottom-up approach, testing can start only after the bottom level modules are ready. This sandwich or mixed approach overcomes this shortcoming of the top-down and bottom-up approaches. It is also called the hybrid integration testing. also, stubs and drivers are used in mixed integration testing.

Advantages:

- Mixed approach is useful for very large projects having several sub projects.
- This Sandwich approach overcomes this shortcoming of the top-down and bottom-up approaches.
- Parallel test can be performed in top and bottom layer tests.

Disadvantages:

- For mixed integration testing, it requires very high cost because one part has a Top-down approach while another part has a bottom-up approach.
- This integration testing cannot be used for smaller systems with huge interdependence between different modules.