4.TESTING

Testing is one of the most important phases in the software development activity. In software development life cycle (SDLC), the main aim of testing process is the quality; the developed software is tested againstattaining the required functionality and performance. During the testing process the software is worked with some particular test cases and the output of the test cases are analyzed whether the software is working according to the expectations or not.

Levels of Testing

Since the errors in the software can be injured at any stage. So, we have to carry out the testing process at different levels during the development. The basic levels of testing are Unit, Integration, System and Acceptance Testing.

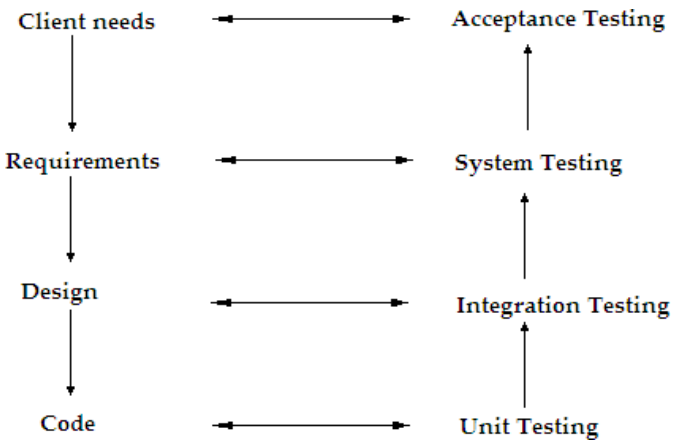

Fig 4.1: Levels of Testing

The Unit Testing is carried out on coding. Here different modules are tested against the specifications produced during design for the modules.In case of integration testing different tested modules are combined into subsystems and tested in case of the system testing the full software is tested and in the next level of testing the system is tested with user requirement.

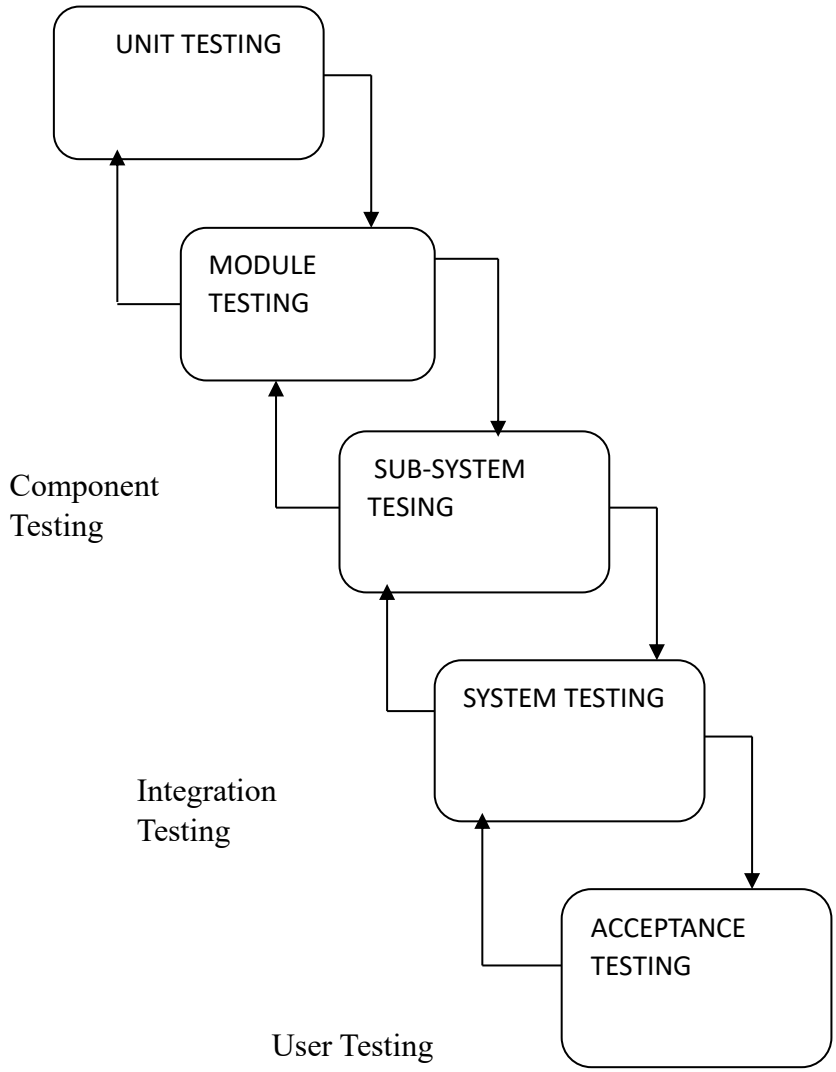

Fig 4.2: Testing Methodologies

Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design, the module. The unit testing we have is white box-oriented, and in some modules, the steps are conducted in parallel.

Test Plan

A test plan is a general document for the entire project that defines the scope, approach to be taken, and personnel responsible for different activities of testing. The inputs for forming the test plane are

- Project plan
- Requirements document
- System design

Test Case Specification

Although there is one test plan for the entire project, test cases have to be specified separately for each test case. A test case specification is given for each item to be tested. All test cases and outputs expected for those test cases.

Test Case Execution and Analysis

The steps to be performed for executing the test cases are specified in a separate document called the test procedure specification. This document specifies any specific requirements that exist for setting the test environment and describes the methods and formats for reporting the results of testing.

Test Approach

Testing can be done in two ways.

- bottom-up Approach
- top-down Approach

Bottom-up Approach

Testing can be performed starting with the smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing, a short programme executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom-level modules are tested, attention turns to those on the next level that use the lower-level ones; they are tested individually and then linked with the previously examined lower-level modules.

Top-down Approach

This type of testing starts with upper-level modules. Since the detailed activities usually performed in the lower-level routines are not provided, stubs are written. A stub is a module shell called by an upper-level module that, when reached properly, will return a message to the calling module

indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower-level module.

4.1 Testing Methodologies

Black Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black box testing. The tester is unaware of the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are processed. An approach to testing where the programme is considered a ‘black box’ The programme test cases are based on the system specification, which is the testing process in which testers can perform testing on an application without having any internal structural knowledge of the application.

Advantages	**Disadvantages**
Well suited and efficient for large code segments.	Limited Coverage since only a selected number of test scenarios areactually performed.
Code Access not required.	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.

Table 4.1.1: Black Box Testing

White Box Testing

Is the testing process in which testers can perform testing on an application with internal structural knowledge. All independent paths have been exercised at least once. All logical decisions have been exercised on their true and false sides. All loops are executed at their boundaries and within their operational bounds.All internal data structures have been exercised to assure their validity. To follow the concept of white box testing, each form

has been tested independently to verify that the data flow is correct, all conditions are checked for validity, and all loops are executed within their boundaries.White box testing is a detailed investigation of the internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal workings of the code.

Advantages	**Disadvantages**
As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.	Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.
It helps in optimizing the code.	Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths will tested.
Extra lines of code can be removedwhich can bring in hidden defects.	It is difficult to maintain white box testing as the use of specialized tools like code analyzers, Debugging tools are required.

Table 4.1.2: White Box Testing

Grey Box Testing

Grey Box testing is a technique to test an application with limited knowledge of its internal workings. In software testing, the term the more you know, the better carries a lot of weight when testing an application. Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. With this knowledge, the tester is able to better prepare test data and test scenarios.

Functional Testing

Functional tests provide systematic demonstrations that the functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g., components in a software system or one-step-up software applications at the company level, interact without error.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated, and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects, conditions, andThus, system testing is a confirmation that all is correct and an opportunity to show the user that the system works. The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black box testing, with success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested, and individual subsystems are exercised through their input interface.

Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations,

and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base, which is then used to support the integration testing of further assemblages.

Top-down Integration

Top-down integrations are an incremental approach to the construction of programme structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control programme. Modules subordinate to the main programme are incorporated into the structure either breath-first or depth-first.

Bottom-up Integration

This method, as the name suggests, begins construction and testing with atomic modules, i.e., modules at the lowest level. Because the modules are integrated in a bottom-up manner, the processing required for the modules subordinate to a given level is always available, eliminating the need for stubs.

Validation Testing

At the end of integration testing, software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in a manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. The information contained in those sections forms the basis for a validation testing approach.

System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform the allocated functions.

Security Testing

Attempts to verify the protection mechanisms built into the system.

Performance Testing

This method is designed to test the runtime performance of software within the context of an integrated system.

Basis Path Testing

The established technique of a flow graph with Cyclamate complexity was used to derive test cases for all the functions. The main steps in deriving test cases were Use the design of the code and draw a correspondent flow graph as follows: Determine the cyclic complexity of the resultant flow graph using a formula.

V(G)=E-N+2 or V(G)=P+1 or V(G)=Number Of Regions

Where V(G) is Cyclometer complexity

- E is the number of edges
- N is the number of flow graph nodes
- P is the number of predicate nodes

This type of testing ensures that

- ❖ All independent paths have been exercised at least once.
- ❖ All logical decisions have been exercised on their true and false sides.
- ❖ All loops are executed at their boundaries and within their operational bounds.
- ❖ All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing, we have tested each form. We have created independently to verify that the data flow is correct, All conditions are checked for validity, and all loops are executed on their boundaries.

Conditional Testing

In this part of the testing, each of the conditions was tested for both true and false aspects. And all the resulting paths were tested. So that each path that may be generated under a particular condition is traced to uncover any possible errors.

Data Flow Testing

This type of testing selects the path of the programme according to the location of the definition and use of variables. This kind of testing was used only when some local variables were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

Loop Testing

In this type of testing, all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

- All the loops were tested at their limits, just above and just below.
- All the loops were skipped at least once.
- For nested loops, test the innermost loop first and then work.
- For concatenated loops, the values of dependent loops were set with the help of connected loops.
- Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

Alpha Testing

For this project, alpha testing is carried out by the customer within the organisation along with the developer. The Alpha tests are conducted in a controlled manner.

Beta Testing

Beta testing has been performed by selecting groups of customers. The developer is not present at the site, and the user will inform the developer of any problems that are encountered. When future problems are reported, they are rectified by the software developer.

Functional Testing

In Functional testing, test cases are decided solely on the basis of the requirements of the programme or module, and the internals of the programme or module are not considered for the selection of test cases. This is also called black Box Testing.

Structural Testing

In Structural testing, test cases are generated based on the actual code of the programme or module to be tested. This is called white Box Testing.

Testing Process

A number of activities must be performed for testing software. Testing starts with a test plan. A test plan identifies all testing-related activities that need to be performed, along with the schedule and guidelines for testing. The plan also specifies the levels of testing that need to be done by identifying the different testing units. For each unit specified in the plan, the test cases and reports are first produced. These reports are analysed.

4.2 Test Cases

The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving, and quitting operations are tested in terms of GUI testing, Positive testing, and negative testing using the corresponding GUI test cases, respectively.

Test Case No	**Test Case Description**	**Expected Output**	**Observed Output**	**Result**
1	Faculty login with correct user name and password	Faculty homepage will be displayed	Faculty home page successfully displayed	Pass

2	Faculty try to add new item details	New item will be successfully created	New item is sucessfully created	Pass
3	Faculty generates the QR Code	QR Code will be successfully generated	QR Code is successfully generated	Pass
4	Faculty observe the student data	Student data will be successfully displayed	Student data is successfully displayed	Pass
5	Student login with the correct student id	Student home page will be displayed	Student home page successfully displayed	Pass
6	Student check the attendance	Number of days attendance will be displayed	Number of days attendance is displayed	Pass
7	Faculty try to login with wrong username and password	Invalid username or password message will be displayed	Invalid message user name and password is displayed	Pass
8	Student try to login with wrong student id	Invalid studentid message will be displayed	Invalid studentid message is displayed	Pass

Table 4.2.1: Test Cases for Overall System