Chapter 8
Testing

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

8.1 Black Box Testing

It is testing without knowledge of the internal workings of the item being tested. For example, when black box testing is applied to software engineering, the tester would only know the "legal" inputs and what the expected outputs should be, but not how the program actually arrives at those outputs. It is because of this that black box testing can be considered testing with respect to the specifications, no other knowledge of the program is necessary. For this reason, the tester and the programmer can be independent of one another, avoiding programmer bias toward his own work. For this testing, test groups are often used, "Test groups are sometimes called professional idiots...people who are good at designing incorrect data." Also, do to the nature of black box testing, the test planning can begin as soon as the specifications are written. The opposite of this would be glass box testing, where test data are derived from direct examination of the code to be tested. For glass box testing, the test cases cannot be determined until the code has actually been written. Both of these testing techniques have advantages and disadvantages, but when combined, they help to ensure thorough testing of the product.

8.1.1 Advantages of Black Box Testing

- More effective on larger units of code than glass box testing

- Tester needs no knowledge of implementation, including specific programming languages
- Tester and programmer are independent of each other
- Tests are done from a user's point of view
- It will help to expose any ambiguities or inconsistencies in the specifications
- Test cases can be designed as soon as the specifications are complete.

8.1.2 Disadvantages of Black Box Testing

- Only a small number of possible inputs can actually be tested, to test every possible input stream would take nearly forever
- Without clear and concise specifications, test cases are hard to design
- There may be unnecessary repetition of test inputs if the tester is not informed of test cases the programmer has already tried
- May leave many program paths untested

8.2 White Box Testing

White box testing is performed based on the knowledge of how the system is implemented. White box testing includes analyzing data flow, control flow, information flow, coding practices, and exception and error handling within the system, to test the intended and unintended software behavior. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities.

White box testing requires access to the source code. Though white box testing can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing during the unit testing phase.

The general outline of the white box testing process is as follows:

- Perform risk analysis to guide the whole testing process.

- Develop a test strategy that defines what testing activities are needed to accomplish testing goals.
- Develop a detailed test plan that organizes the subsequent testing process.
- Prepare the test environment for test execution.
- Execute test cases and communicate results.
- Prepare a report.

For a complete software examination, both white box and black box tests are required.

Testing is applied to find that how much our program is efficient. Testing is critical phase of software quality assurance. It indicates the review of specification, design & code generation. After completing source code software must be tested to find some uncover error, before delivered it.

Thus, a series of test cases has to be designed that have a high likelihood of finding an error. To accomplish this task, software testing methods are used.

These methods are:

- We exercise the internal logic of the software component.
- We exercise program's input & output domains, thus uncovering error in program function, behaviour, & performance.

8.3 Test Cases and Results

- **Unit Testing**

  In unit testing we test each individual frame of the software. In this testing we insert wrong inputs on place of amount and then performer operation on it. We test each frame by running it. The design of the frame must be convenient for the user. Unit testing is essentially for verification of the code produce during the coding phase. This testing is typically performed by the programmer. After the completion of unit testing we delivered it for integrated testing.

- **Integration Testing**

  The individual program components, once been tested, must then be integrated to develop a complete system. In this testing first integrate a minimal system configuration & test it. Then add components to this & test it after each added increment.

  By applied these testing we find out the efficiency, portability, quality & also durability. After then we delivered that software to the customer.

8.3.1 Advantages

- No special configuration or changes are need on users PCs.
- Lower costs.
- Centralized data is secure and easy to backup
- Updates can be made quickly and easily.
- Access to other technologies.
- Reduced network traffic.
- Single Sign-On.
- Always up-to-date.
- Everybody has a browser. Familiar interface encourages use.
- Information is accessible to a wide audience anywhere in the world.