**CHAPTER**

**SYSTEM TESTING**

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. It provides a way to check the functionality of components, assemblies, sub-assemblies and a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each type of test addresses a specific testing requirement.

**.1 TYPES OF TESTS**

There are many types of tests described in SDLC approach. A few of them has been listed below.

**Unit testing**

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens and fields. Integration tests demonstrate that although the components were individually satisfaction, as shown successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Approach**

i. Top-down approach --- this is used for new systems.

ii. Bottom-up approach --- this is used for existing systems.

**Top-down Approach**

Testing main module without coming sub modules is called top-down approach. We can use temporary programs instead of sub modules is called stub.

**Bottom-up approach:**

Testing sub modules without coming main modules is called bottom-up approach. We can use temporary programs instead of main module is called driver.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the structural and technical requirements.

Functional testing is centered on the following items:

Valid input : identified classes of valid input must be accepted.

Invalid input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System test**

System testing will compare the system specifications against the actual system. The system test design is derived from the system design documents and is used in this phase. Sometimes system testing is automated using testing tools. Once all the modules are integrated several errors may arise. Testing done at this stage is called system testing. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White box testing strategy deals with the internal logic and structure of the code. White box testing is also called as glass, structural, open box or clear box testing. The tests written based on the white box testing strategy incorporate coverage of the code, branches, paths, statements and internal logic of the code etc. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black box testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Acceptance testing**

Acceptance testing is the phase of testing used to determine whether a system satisfies the requirements specified in the requirements analysis phase. The acceptance test design is derived from the requirements document. The acceptance test phase is the phase used by the customer to determine whether to accept the system or not.