**CHAPTER 7**

**SYSTEM TESTING**

**7.1 INTRODUCTION**

Testing accomplishes a variety of things, but most importantly it measures the quality of the software we are developing. This view presupposes there are defects in the software waiting to be discovered and this view is rarely disproved or even disputed.

Several factors contribute to the importance of making testing a high priority of any software development effort. These include:

* Reducing the cost of developing the program.
* Ensuring that the application behaves exactly as we explain to the user for the vast majority of programs, unpredictability is the least desirable consequences of using an application.
* Reducing the total cost of ownership. By providing software that looks and behaves as shown in the documentation, the customers require fewer hours of training and less support from product experts.
  1. **Different types of Testing:**

**7.2.1 Unit Testing:**

Unit testing focuses verification on the smallest unit of software design, the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The unit testing is a white box oriented testing.

First of all the module interface is tested to ensure that the information properly flows into and out of the program until under test. Then the local data structure is tested to ensure the data stored temporarily maintains its integrity during all steps in an execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once. And finally, all errors handling paths are tested. In this project the testing is done according to bottom-up approach. Starting with smallest and lowest level modules and processing one at a time. For each module a driver and corresponding stubs were also written. If any errors found they were corrected immediately and the unit was tested again.

* + 1. **Integration Testing:**

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. The idea is to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up a process are tested together. Any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis.

In this software, the bottom-up integration testing approached has been used, starting with the smallest and lowest level modules and proceeding one at a time. For each module the tests were conducted and the results were noted down.

* + 1. **User Testing:**

User Testing is nothing but the test of the software by the users themselves with live data being fed to the system. This helps in building really robust system. User testing in this system has been done extensively ascertain the results.

**7.3 System Testing**

This tends to affirm the end-to-end quality of the entire system. System test is often based on the functional/requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability, are also checked.