7. TESTING

7.1 Testing Plan

- **Black-box White-box Testing**

  In black-box testing a software item is viewed as a black box, without knowledge of its internal structure or behavior. Possible input conditions, based on the specifications (and possible sequences of input conditions), are presented as test cases. In white-box testing knowledge of internal structure and logic is exploited. Test cases are presented such that possible paths of control flow through the software item are traced. Hence more defects than black-box testing are likely to be found. The disadvantages are that exhaustive path testing is infeasible and the logic might not conform to specification. Instrumentation techniques can be used to determine the structural system coverage in white box testing. For this purpose tools or compilers that can insert test probes into the programs can be used.

- **Performance Testing**

  Performance testing is design to test the runtime performance of the system within the context of the system. This test is performed at module level as well as at system level. Individual modules developed by Developers are tested for required performance.

- **Code Testing**

  We used Code testing which is a method in which we examined the logic of code written that is to execute each and every instruction of modules. During the testing as well as during testing phase of system, we tried to execute most of the modules. For this we took test cases for individual modules and conducted a fair testing analysis.

- **Statistical Testing**

  Used to test the code's performance and reliability and to check how it works under operational conditions. Tests are designed to reflect the actual user inputs and their frequency. The stages involved in the static analysis for this system are follows.

Control flow analysis

- Unreachable code
- Unconditional branches into loops

Data Use Analysis

- Variable used before initialization
- Variables declared but never used
- Variables assigned twice but never used between assignments
- Possible array bound violations
- Declared variables

Interface Analysis

- Parameter type mismatches
- Parameter number mismatches
- Non-usage of the results of functions
- Uncalled functions and procedures

- **Stress Testing:**

  It executes a system in a manner that demands resources in abnormal quantity, frequency and volume. Applying frequent requests of offers and in large volume tests the project holistically. It takes time to reply but does not fail to give result.

- **Object Testing:**

  Object testing is to test objects as individual components, which are often larger than single functions. Here following activities have taken place.

  - Testing the individual operations associated with objects
  - Testing individual object classes.
  - Testing clusters of objects
  - Testing the object-oriented system.

- **Integration Testing:**

  After completion of testing all the individual modules, we began testing for the entire project. This integration process involves building the system and testing the resultant system for problems that arise from component interactions. We have applied top-down strategy to validate high-level components of a system before design and implementations have been completed. Because, our development process started with high-level components and we worked down the component hierarchy.

7.2 TEST CASES

TEST CASE ID: TC01	**NAME:** Login Successfully
PURPOSE	Login successfully when username and password is correct
INPUT	Enter correct username and password and click on login button
EXPECTED OUTPUT	Successfully login and enter into the application

Table 6.1 Testing: Login Successfully

TEST CASE ID: TC02	**NAME:** Login Failed
PURPOSE	Login failed when username and password is incorrect
INPUT	Enter wrong username and password and click on login button
EXPECTED OUTPUT	Display the toast message

Table 6.2 Login Failed

TEST CASE ID: TC03	**NAME:** On enter intern id.
PURPOSE	Intent to the details grid for particular intern.
INPUT	Enter intern id.
EXPECTED OUTPUT	A grid of details will be displayed through which admin can view details.

Table 6.3 Testing: On enter intern id.

TEST CASE ID: TC04	**NAME:** Add Course
PURPOSE	To add newly introduced course.
INPUT	Enter course name and add.
EXPECTED OUTPUT	Course will be added in the database.

Table 6.4 Add Course

TEST CASE ID: TC05	**NAME:** Add Faculty
PURPOSE	To add new faculty.
INPUT	Fill faculty registration form.
EXPECTED OUTPUT	Faculty will be registered.

Table 6.5 Testing: Add Faculty

TEST CASE ID: TC06	**NAME:** Student Request
PURPOSE	To know about newly registered student and assign faculty.
INPUT	Assign faculty to student by adding.
EXPECTED OUTPUT	Faculty is assigned to student.

Table 6.6 Testing: Student Request

TEST CASE ID: TC07	**NAME:** View Students Profile
PURPOSE	View records of student.
INPUT	Enter student id.
EXPECTED OUTPUT	Records of particular student will be displayed.

Table 6.7 Testing: View Student Profile