# **Chapter Four:**

# **Implementation and Testing**

## **4.1 Introduction**

During this phase physical design specification must be turned into working computer code, and provide help for current and future users and take care of the system. And then the code is tested until most of the errors have been detected and corrected. The purpose of this activity is to convert the final physical system specification into working model with reliable software and hardware.

## **4.2 Sample coding**

## **4.3 Testing**

Testing is a process to show the correctness of the program and designed to analyze the logic used in the implementation of the System. In the case of our project we use unit, user acceptance, and integration testing method.

## **4.4 Unite testing**

Each module is tested alone in an attempt to discover any errors in its code. In unit testing, each module (roughly a section of code that performs a single function) is tested alone in an attempt to discover any errors that may exist in the module’s code. We tested the system as shown below in the appendix.

## **4.5 Integration testing**

The process of bringing together for testing purposes all of the modules that a program comprises. Modules are typically integrated in a top-down, incremental fashion. If an error occurs, the process stops, the error is identified and corrected, and the test is redone. The process repeats until the entire program—all modules at all level is successfully integrated and tested with no errors. After the team tested using unit testing the next step is integrating testing. In integrated testing the team tested the system all modules that a program contains.

## **4.6 Acceptance testing**

It is the process whereby actual users test a completed information system, the end result of which is the users’ acceptance of it once they are satisfied with it. Acceptance refers to the fact that users typically sign off on the system and “accept” it once they are satisfied with it. Testing the system in the environment where it will be used. The purpose of acceptance testing is for users to determine whether the system meets their requirements. In acceptance test the team test the system by different user that satisfies the requirements. The user tested by inserting real data.

### **4.7 System testing**

System testing is simply expanded integration testing, where we are testing the interfaces between programs in a system rather than testing the interfaces between modules in a program. System testing is also intended to demonstrate whether a system meets its objectives. It is the final step of testing**.** In this step the team members test the entire system as a whole with all forms, code, modules and tests all the functionalities in the System. This form of testing is popularly known as Black Box testing or System tests. All errors in the forms, functions, modules are tested.

### **4.8 Sample unit test case:**

Table 4.1 Test Case 1*:*

|  |  |  |
| --- | --- | --- |
| **Test Case ID** = Taxi Booking – TestCase01 | | |
| **Unit to Test** = Registration of Drivers | | |
| **Assumptions** = Congratulations! | | |
| **Test Data** = Name (ValidName, empty)  Address(validAddress, empty)  Email(validEmail, empty)  Drivinglicense (valid, empty)  Username (valid Username, empty)  Password (valid password, empty) | | |
| **Steps to be Executed** | **Data** | **Expected Results** |
| Empty Name and all others filled and Click register button | Any valid data for the other fields | “Error: Name can’t be blank” |
| Enter valid Name, empty Username and Click register button | First Name= Abraham  Any valid data for the other fields | “Error: Username can’t be blank” |
| Enter Valid Name and Username, empty password and Click register button | Name = Abraham  Username = Abirsh12 | "Error: Password must contain at least six characters!" |
| Enter Valid Name, Username, and password and Click register button | Name = Abraham  Username = birsh12  Password = ab | "Error: Password must be different from Username!" |
| All fields with valid data and Click register button | All fields with valid data | Display Congratulations! message |

Table 4.2 Test Case 2:

|  |  |  |
| --- | --- | --- |
| **Test Case ID** = Taxi Booking – TestCase02 | | |
| **Unit to Test** = Registration of Taxi | | |
| **Assumptions** = Congratulations! | | |
| **Test Data** = Taxino (Valid Taxino, empty)  Model (valid Model, empty)  Status (valid Status, empty) | | |
| **Steps to be Executed** | **Data** | **Expected Results** |
| Empty Taxino and all others filled and Click register button | Any valid data for the other fields | “Error: Taxino can’t be blank” |
| Enter valid Taxino, empty Model and Click register button | Taxino= 12345  Any valid data for the other fields | “Error: Model can’t be blank” |
| Enter Valid Taxino and Model, empty Status and Click register button | Taxino = 12345  Model =atoz | "Error: Status cannot be empty!" |
| Enter Valid Taxino, Model, Status and Click register button | Taxino = 12345  Model = atoz  Status = 0 | “Congratulations!" |
| All fields with valid data and Click register button | All fields with valid data | Display Congratulations! Message. |