



**Pune Institute of Computer Technology Dhankawadi, Pune**

**DEPARTMENT OF COMPUTER ENGINEERING**

**Academic Year 2021-22**

**STQA MINI PROJECT REPORT ON  
HEALTHCARE MANAGEMENT SYSTEM**

**SUBMITTED BY**

Omkar Amilkanthwar	41403
Aniruddha Deshmukh	41405
Atharva Satpute	41409

**Under the guidance of  
Prof. K. C. Waghmare**

<b>Sr. No.</b>	<b>Content</b>	<b>Page No.</b>
1.	Introduction	3
2.	Functional Requirements	4
3.	Non - Functional Requirements	5
4.	Design (Block Diagram)	6
5.	Screenshots	7
6.	Test Plan Details	10
7.	Test Scenarios	12
8.	Test Cases	16
9.	JUnit Test Cases	17
10.	Screenshots	19

## **Introduction**

This project is created with an aim to provide services to the Medical Sector by providing an application to Hospital Administrators so as to keep track of patients along with their medical records in an Hospital and generate invoices of each transaction.

The Medical Record Generation system has basic functionality to add a patient along with diagnosis details, simultaneously providing ease of documentation in terms of business perspective.

Also, development of a system emulator that is capable of functioning more quickly, accurately and updates the records and enables administrators to view medical records instantaneously.

This Medical Report generator can be equipped with a Patient Scheduler and can be deployed in a real world situation.

For example, it could be implemented for the current COVID-19 situation, where the Scheduler prioritizes on the basis of severity and the medical report generator can provide documentation of diagnosis reports with minimum human interaction, thereby adding value to the product.

# Functional Requirement

## 1. Login

1.1. Only validated administrators (admin) are allowed to proceed to the home page.

## 2. Home Page

2.1. Add new patients admitted in the hospital

## 3. Management

3.1. Display the category of treatment availability in the hospital.

3.2. Display projected cost of the meal.

3.3. Display the revenue generated for a specified period of time (days).

3.4. Generate medical diagnosis report of the patient

3.5. Discard entries with invalid input.

3.6 Generate transaction details

## **Nonfunctional Requirements**

Non - Functional requirements define the needs in terms of performance, logical requirements, design constraints, standards compliance, reliability, availability, security, maintainability, and portability.

### **1.1.1 Performance Requirements**

Performance Requirements is the fastness or responsiveness of a system to the application.

- GUI is light weighted and is instantaneous
- The Validation of username and password is a quick process.

### **1.1.2 Design Constraints**

This Medical Report Generator shall be a stand-alone JSwing GUI supported Java based system running in a Windows environment.

### **1.1.3 Standards Compliance**

- Naming conventions in the code base will be user friendly and simulate to real word entities.
- The GUI will have a streamlined navigation with consistent look

### **1.1.4 Reliability**

The system will be consistent in state both functionally and visually making it reliable and available throughout.

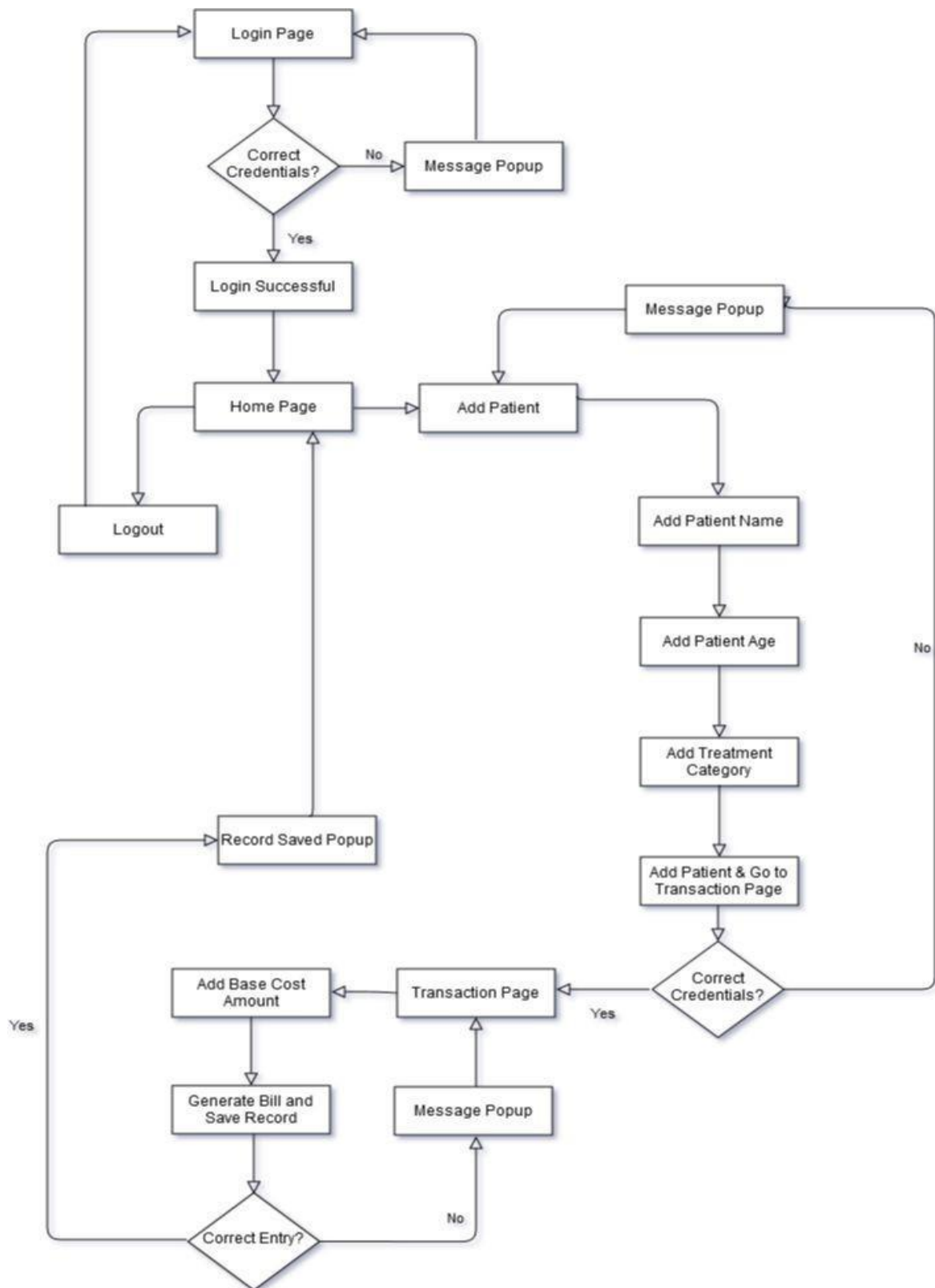
### **1.1.5 Security**

Provision for Login interface so as to ensure that only Hospital Management and concerned authorities/administrators have the access to the system.

### **1.1.6 Maintainability**

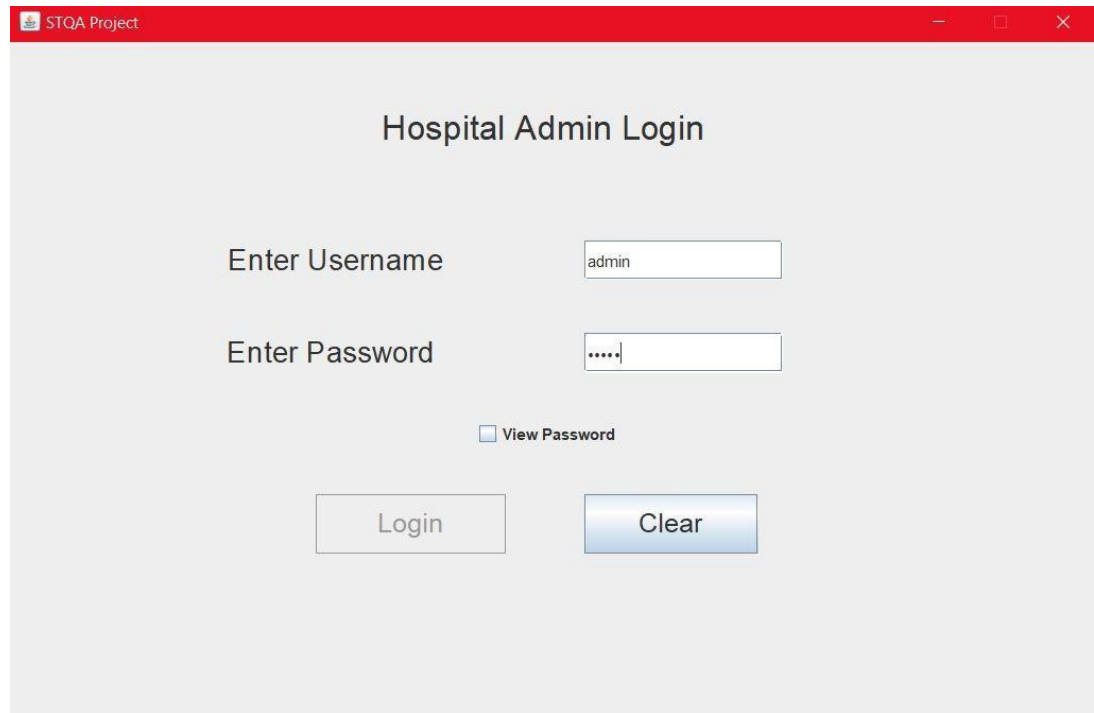
As being a Java based system, it offers promise of long term support and thus ensuring maintainability with regards to reliability, security and robustness.

## Block Diagram

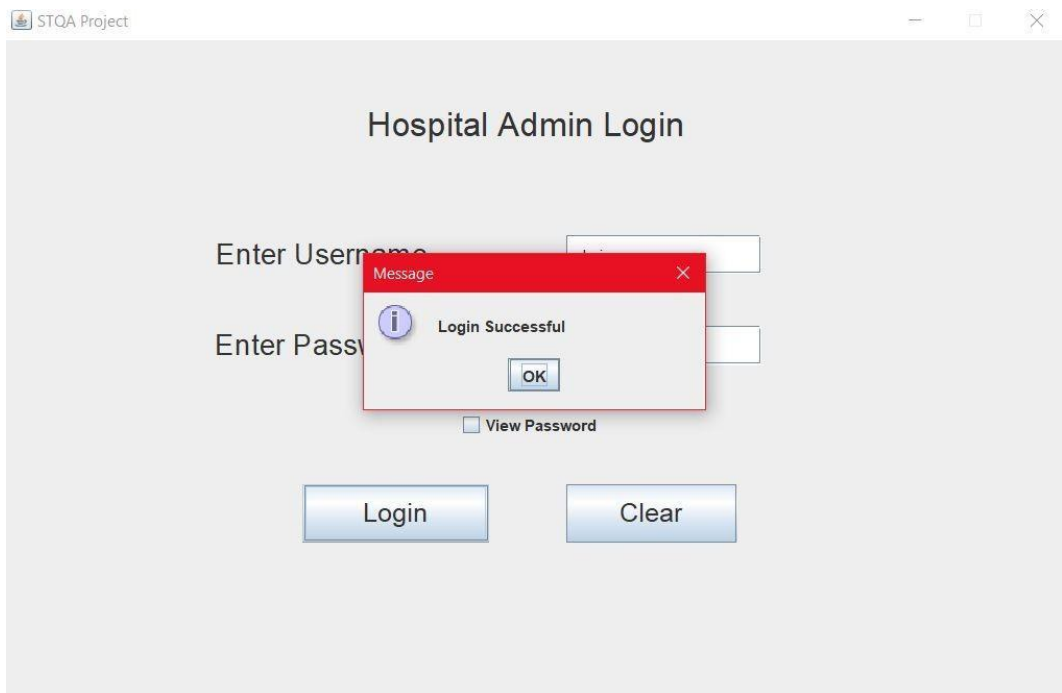


# Screenshots

## Login page initial:

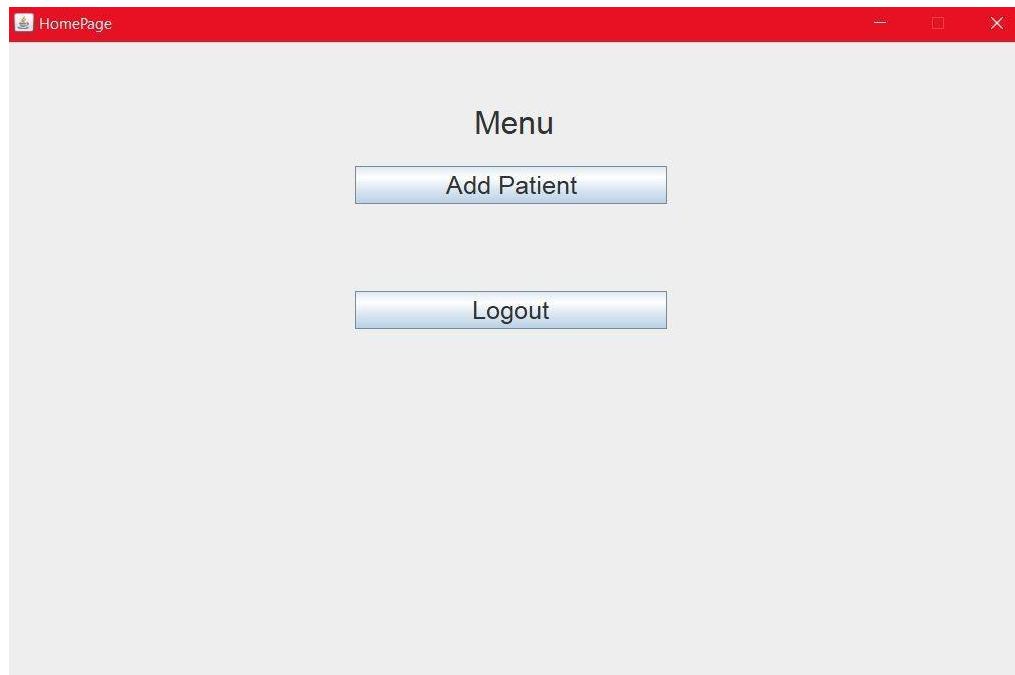


A screenshot of a web application window titled "STQA Project". The main heading is "Hospital Admin Login". Below the heading, there are two input fields: "Enter Username" with the text "admin" and "Enter Password" with masked characters "....". Below the password field is a checkbox labeled "View Password". At the bottom, there are two buttons: "Login" and "Clear".



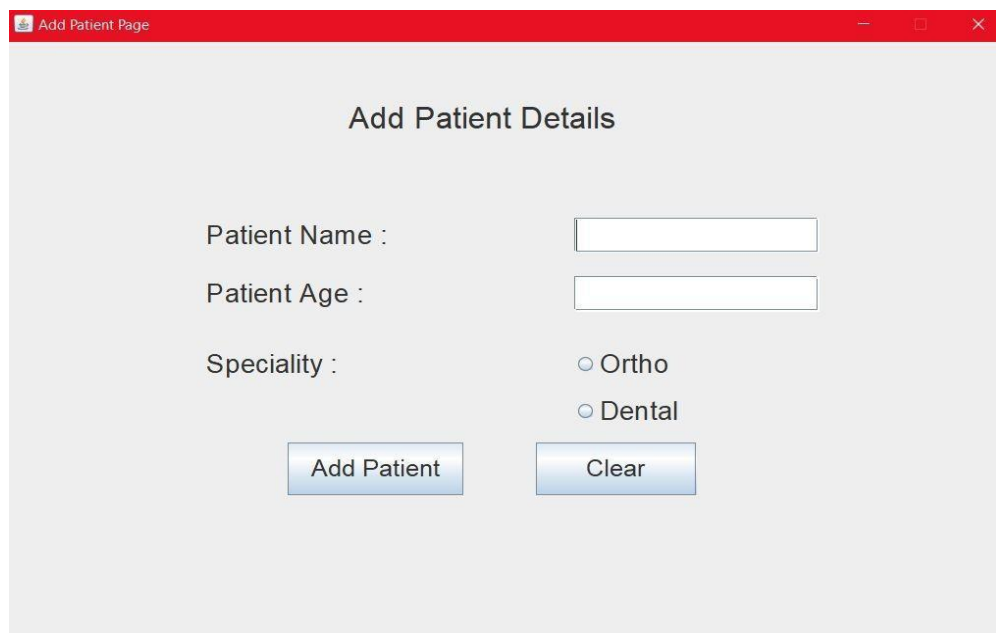
A screenshot of the same "Hospital Admin Login" page, but with a modal message box displayed in the center. The message box is titled "Message" and contains an information icon, the text "Login Successful", and an "OK" button. The background login form is partially obscured by the message box.

## Home Page (Menu):



A screenshot of a web application window titled "HomePage". The window has a red title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains the word "Menu" centered at the top. Below "Menu" are two blue buttons with white text: "Add Patient" and "Logout", stacked vertically.


## Adding Patient:



A screenshot of a web application window titled "Add Patient Page". The window has a red title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains the text "Add Patient Details" centered at the top. Below this text are three labels with corresponding input fields: "Patient Name :" followed by a text box, "Patient Age :" followed by a text box, and "Speciality :" followed by two radio button options, "Ortho" and "Dental". At the bottom of the form are two blue buttons with white text: "Add Patient" and "Clear", positioned side-by-side.



## Patient Details:

 Add Patient Page

### Add Patient Details

Patient Name :

Patient Age :

Speciality : ☒ Ortho ☐ Dental

# **Test Plan**

## **Introduction :-**

This application is intended to generate the invoice of a patient's appointment and to automate the medical records system. Operator is required to enter the required Patient details and medical charges amount. The final bill will be generated and given to the user. A copy of the same bill is stored on the admin local system.

## **Intended Audience :-**

This test plan is made for system testing of this application. The test plan will be referred by:-

- Doctors and managing staff
- Hospital receptionists
- Clinic consultants

## **Intended Users :-**

This application will be used by Multispeciality hospitals and clinics.

## **Test Scope :-**

System testing shall cover :-

- User interface testing
- Functionality testing

## **Risk Analysis :-**

- Application will not be accessible by any other credentials than the ones mentioned.
- Application does not avail navigation to the previous page.
- Users may not understand English.

### **Test Design :-**

#### Testing Levels Responsibility

Unit Testing when code is generated - Developers

Integration testing - Developers

System testing - Testers

### **Test Environment :-**

Test environment will be made of individual machines.

- Windows 10 professional
- 8 GB RAM
- Quad-core processor, Core i5.

### **Test Entry Criteria :-**

- All reviews and comments are closed.
- All unit testing defects are closed.
- Application is installed and launched.

### **Test Suspension Criteria :-**

Tests will be suspended in the following circumstances:-

- Test cases written do not find sufficient number of defects.
- Test case writing and test data definition are not completed.
- Application cannot be installed.

# Test Scenario

## **LOGIN PAGE:**

### **Common for All Test Scenarios :-**

The application can be launched by clicking an icon on the desktop. It can be launched from 'Start', 'Programs' and selecting the application from the list. On start a screen with a Title Login Page appears. There are two labels with captions 'Username', 'Password' one below another in the given order. There is also a login button. UI is light weighted and user friendly.

### **Scenario 1 : Positive scenario where login is done successfully :-**

1. On inserting the correct combination of username and password, the application windows minimizes and the Home Page opens.
2. Successfully logged in to the application.

### **Scenario 2 : Negative scenario where username or password is empty :-**

1. Negative scenario as the password and username cannot be validated in order to open the Home Page.

### **Scenario 3 : Negative scenario where username is correct or password is empty or incorrect :-**

1. Negative scenario as the password and username cannot be validated in order to open the Home Page.

### **Scenario 4 : Negative scenario where username is incorrect or empty :-**

1. Negative scenario as the password and username cannot be validated in order to open the Home Page.

**Scenario 5 : Negative scenario where username or password is incorrect:-**

1. Negative scenario as the password and username cannot be validated in order to open the Home Page.

**HOME PAGE:**

**Common for All Test Scenarios :-**

After successful login by the authorized user, on the home page, there are two options available, 'Add Patient' and 'Logout'. On clicking the 'Add Patient' button, the user is redirected to the 'Add Patient Details' window and on 'Logout' to Login Window respectively.

**Scenario 1 : Positive scenario where add patient button is clicked :-**

1. On clicking the 'Add Patient' button, the user is redirected to the 'Add Patient Details' window

**Scenario 2 : Positive scenario where logout button is clicked :-**

1. On clicking the 'Logout' button, the user is redirected to the Login window.

**ADD PATIENT PAGE:**

**Common for All Test Scenarios :-**

On clicking the 'Add Patient' button, the user is redirected to the 'Add Patient Details' window having Form fields to take Patient Name, Age and Treatment Category as input. It has an Add Patient Button along with a Clear button.

**Scenario 1 : Positive scenario where add patient button is clicked :-**

1. On clicking the 'Add Patient' button, if all the entries of Patient name, age and treatment category is valid, it will redirect to

Transaction Window.

**Scenario 2 : Negative scenario where add patient button is clicked :-**

1. On clicking the 'Add Patient' button, if any one of the entries of Patient name, age and treatment category is not valid, it will redirect to Message Popup Window there by a Negative Scenario.
2. There can be eight combinations where equivalence partitioning can be applied.
3. One of the eight combinations, the field(s) which is taken as not valid each time may represent the equivalence class.
4. Negative scenario as patient cannot be added and 'Add Patient Button' doesn't work successfully.

**Scenario 3: Positive scenario where clear button is clicked :-**

1. On clicking the 'Clear' button, all the entries of Patient name, age and treatment category are resorted to blank.

**TRANSACTION PAGE:**

**Common for All Test Scenarios :-**

On clicking the 'Add Patient' button, the user is redirected to the 'Transaction' window having Form fields to take Base Cost Amount as input. It has a Generate Bill and Save Record Button along with a Clear button.

**Scenario 1 : Positive scenario where generate bill button is clicked :-**

1. On clicking the 'Generate Bill' button, if the entry of Base Cost Amount is valid, it will redirect to a message popup with confirmation message.
2. When the Transaction is executed successfully, the medical record file is saved to local repository -
  - Created if it was the patient's first appointment.
  - Updated if the patient already has an existing medical history.

3. Users will be redirected to the Home Page Window.

**Scenario 2 : Negative scenario where add patient button is clicked :-**

1. On clicking the 'Generate Bill' button, if the entry of Base Cost Amount, if is not valid, it will redirect to Message Popup Window there by a Negative Scenario.
2. Medical Report will not be generated and saved to the local project repository (default).
3. Negative scenario as patient cannot be added and 'Generate Bill' button doesn't work successfully.

**Scenario 3 : Positive scenario where clear button is clicked :-**

1. On clicking the 'Clear' button, the Base Cost Amount field is restored.

# Test Cases

## **Test case 1**

Test Precondition:- Application must be installed and Computer must be working.

Objectives:- To check whether an app can be launched or not.

Valid/invalid Conditions:- Valid

Test Data:- Launching an application by clicking an icon on the desktop. (This is considered as equivalent to launching application from 'Start', 'Programs' and selecting the application from the list)

Expected Results:- Application is launched (Screen is seen with the login of restaurant).

---

## **Test case 2**

Test Precondition:- Application is launched

Objectives :- To check that application opens and occupies the screen Valid/Invalid

Conditions :- Valid condition

Test Data :- Not applicable

Expected Results :- Complete screen is occupied by the layout mentioned. No scroll bar appears.

---

## **Test case 3**

Test Precondition:- Application opened.

Objectives:- To login in the application. Valid/invalid Conditions:- Valid

Test Data:- correct Username and Password

Expected Results:- Application is launched (Screen is seen with the Home Page).

---

## **Test case 4**

Test Precondition:- Application opened.

Objectives:- To login in the application.

Valid/invalid Conditions:- InValid

Test Data:- incorrect Username and Password combination.

Expected Results:- Application is not launched.



## **JUnit Test Cases**

Unit tests were created as JUnit tests. Majority of test cases were considered. The tested classes, the functions of classes and JUnit test are tabularized.

JUnit tests were coded and executed using Eclipse and JUnit version 5.

By following Spiral methodology, The JUnit tests were run once that led to identifying bugs which were detected, solved and closed in the next run.

After certain spirals or iterations, no problems were found, after all bugs were fixed.

Table :- Tested classes.

<b>CLASS NAME</b>	<b>TEST FILE</b>
LoginTest.class	JUnitTesting_LoginPage.class AddPatient.class JUnitTesting_AddPatientPage.class Transaction.class JUnitTesting_TransactionPage.class

Table :- Tested functions.

<b>TEST CLASS NAMES</b>	<b>TEST FUNCTION</b>
JUnitTesting_LoginPage.class	LoginButtonDisableCheck() ViewPasswordButtonDisableCheck() LoginNameCheck() LoginPasswordCheck()
JUnitTesting_AddPatientPage.class	SpecialityButtonGroupClearCheck() PatientNameCheck() PatientAgeNullCheck() PatientAgeValidCheck()
JUnitTesting_TransactionPage.class	TranscationInitialAmtLabelCheck() TransactionParameterAgeInitialCheck() TranscationInitialAmtValueCheck() TranscationRecordNameCheck()

## JUNIT Test cases:-

### Test case 1 – UI Checking

DESCRIPTION :- This JUnit test case was run to ensure that the UI is loaded correctly on successful login.

ERRORS FOUND :-No errors or failures were collected.

### Test case 2 – View Password Button Check

DESCRIPTION :- This JUnit test case was run to ensure that the on enabling this if the password was visible to user or not.

ERRORS FOUND :- No errors or failures were collected.

### Test case 3 – Speciality Button Group Check

DESCRIPTION :- This JUnit test case was run to ensure that one of the treatment speciality group options is selected.

ERRORS FOUND :- No errors or failures were collected.

# JUNIT TESTING

