Pimpri Chinchwad Education Trust’s

Pimpri Chinchwad College of Engineering

Department of Computer Engineering

Mini Project 1 Report

On

Hospital Report and Record Generator : Testing using Junit

Group Members

BECOB203 : Varun Gadde
BECOB205 : Sanskruti Karpe
BECOB256 : Vaishali Singh Jadon
BECOB270 : Vedant Upganlawar

Guide
Mr. Shrikant Kokate

Index

Sr. No.	Content	Page No.
1.	Introduction	3
2.	Functional Requirements	4
3.	Non - Functional Requirements	5
4.	Design (Block Diagram)	6
5.	Source Code / Functions	7
6.	Output Screenshots	24
7.	Test Plan Details	28
8.	Test Scenarios	33
9.	Test Cases	37
10.	Junit Test Cases	42
11.	Output Screenshots	46

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 Requirements

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.

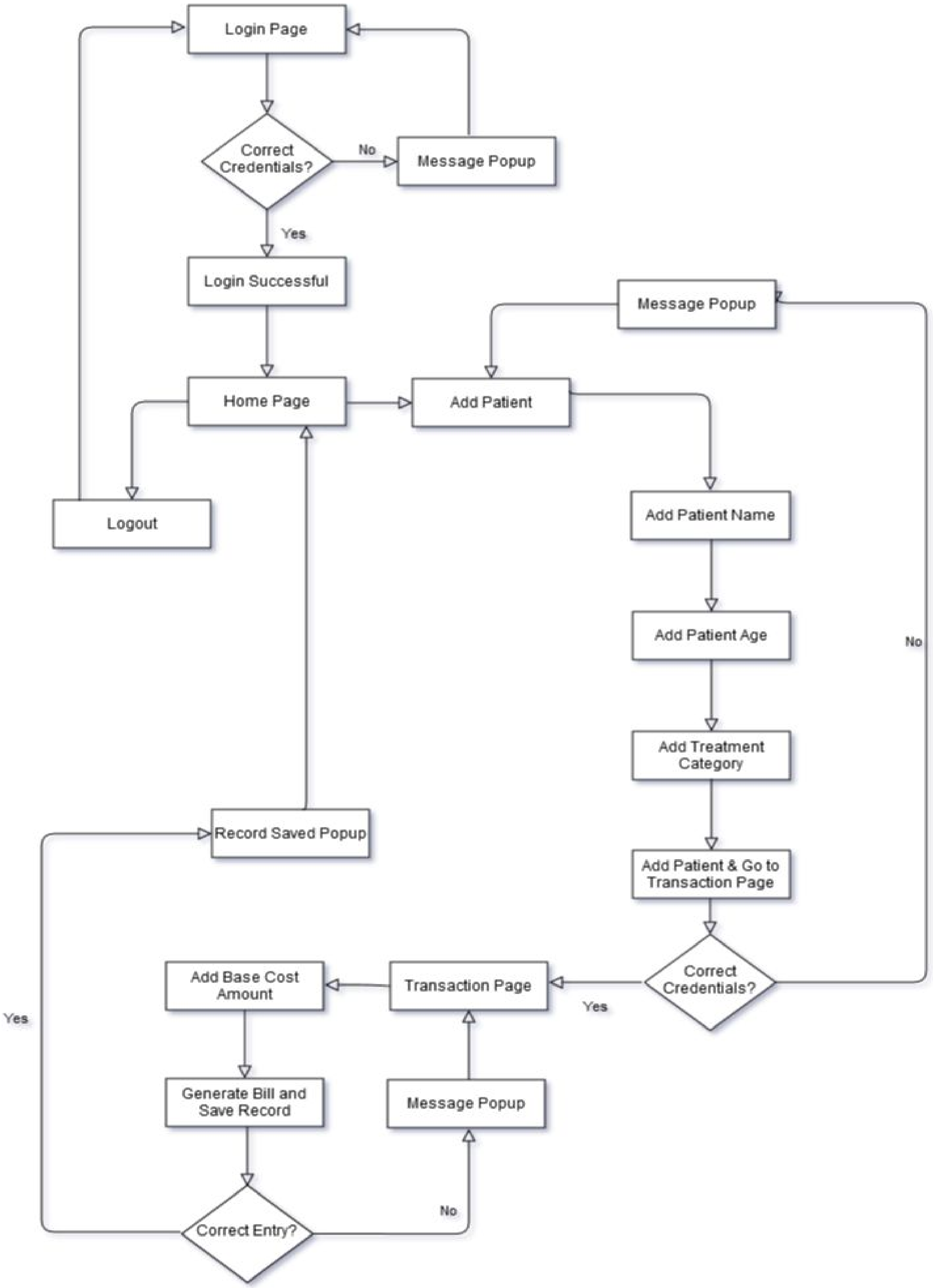
1.1.5 Security

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



Source Code

Login Page:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginTest extends JFrame implements ActionListener {


	private static final long serialVersionUID = 1L;

	Container container = getContentPane();

	private final JLabel welcomeLabel = new JLabel("Hospital Admin
Login");

	JLabel usernameLabel = new JLabel("Enter Username");
	JLabel passwordLabel = new JLabel("Enter Password");

	JTextField usernameTextField = new JTextField();
	JPasswordField passwordField = new JPasswordField();

	JButton loginButton = new JButton("Login");
	JButton clearButton = new JButton("Clear");

	JCheckBox showPassword = new JCheckBox("View Password");

	LoginTest() {
		setLayoutManager();
		setLocationAndSize();
		addComponentsToContainer();
		addActionEvent();

	}
```

```
public void setLayoutManager() {
        container.setLayout(null);
}

public void setLocationAndSize() {

        welcomeLabel.setFont(new Font("Arial", Font.PLAIN, 26));
        welcomeLabel.setBounds(280, 30, 250, 70); //x,y,w,h

        usernameLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        usernameLabel.setBounds(164, 150, 200, 30);

        passwordLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        passwordLabel.setBounds(163, 220, 194, 30);

        usernameTextField.setBounds(433, 150, 150, 30);
        passwordField.setBounds(433, 220, 150, 30);

        showPassword.setBounds(350, 280, 150, 30);

        loginButton.setFont(new Font("Arial", Font.PLAIN, 20));
        loginButton.setBounds(231, 341, 143, 45);
        loginButton.setEnabled(false);

        clearButton.setFont(new Font("Arial", Font.PLAIN, 20));
        clearButton.setBounds(433, 341, 131, 45);

        this.setTitle("STQA Project");
        this.setVisible(true);
        this.setBounds(10, 10, 832, 551);
        this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        this.setResizable(false);
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = this.getSize().width;
        int h = this.getSize().height;
        int x = (dim.width - w) / 2;
        int y = (dim.height - h) / 2;
        this.setLocation(x, y);

    }

    public void addComponentsToContainer() {
        container.add(usernameLabel);
        container.add(passwordLabel);
        container.add(usernameTextField);
        container.add(passwordField);
        container.add(showPassword);
        container.add(loginButton);
        container.add(clearButton);
        getContentPane().add(welcomeLabel);
    }

    public void addActionEvent() {
        loginButton.addActionListener(this);
        clearButton.addActionListener(this);
        showPassword.addActionListener(this);
    }

    @SuppressWarnings("deprecation")
    @Override
    public void actionPerformed(ActionEvent e) {
        String userText;
        String pwdText;
        userText = usernameTextField.getText();
        pwdText = passwordField.getText();

        if (!userText.isEmpty() && pwdText.length() > 0) {
            loginButton.setEnabled(true);
        }

        if (e.getSource() == loginButton) {
            if (userText.equalsIgnoreCase("admin") &&
pwdText.equalsIgnoreCase("admin")) {
```

```
                JOptionPane.showMessageDialog(null, "Login
Successful");
                new HomePage().setVisible(true);
                this.dispose();

            } else {
                JOptionPane.showMessageDialog(null, "Invalid
Username or Password");
            }
        }

        if (e.getSource() == clearButton) {
            usernameTextField.setText("");
            passwordField.setText("");
            loginButton.setEnabled(false);
        }

        if (e.getSource() == showPassword) {
            if (showPassword.isSelected()) {
                passwordField.setEchoChar((char) 0);
            } else {
                passwordField.setEchoChar('*');
            }
        }
    }

    public static void main(String[] args) {
        @SuppressWarnings("unused")
        LoginTest login_frame = new LoginTest();

    }

}
```

Home Page:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class HomePage extends JFrame implements ActionListener {

    private static final long serialVersionUID = 1L;

    Container container = getContentPane();

    private final JLabel menuLabel = new JLabel("Menu");

    JButton addPatientButton = new JButton("Add Patient");
    JButton exitButton = new JButton("Logout");

    HomePage() {
        setLayoutManager();
        setLocationAndSize();
        addComponentsToContainer();
        addActionEvent();

    }

    public void setLayoutManager() {
        container.setLayout(null);
    }

    public void setLocationAndSize() {

        menuLabel.setFont(new Font("Arial", Font.PLAIN, 26));
        menuLabel.setBounds(375, 30, 250, 70);

        addPatientButton.setFont(new Font("Arial", Font.PLAIN, 20));
        addPatientButton.setBounds(280, 100, 250, 30);
        exitButton.setFont(new Font("Arial", Font.PLAIN, 20));
        exitButton.setBounds(280, 200, 250, 30);

        this.setTitle("HomePage");
        this.setVisible(true);
        this.setBounds(10, 10, 832, 551);
        this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        this.setResizable(false);

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = this.getSize().width;
        int h = this.getSize().height;
        int x = (dim.width - w) / 2;
        int y = (dim.height - h) / 2;
        this.setLocation(x, y);

    }

    public void addComponentsToContainer() {
        container.add(addPatientButton);
        container.add(exitButton);
        getContentPane().add(menuLabel);
    }

    public void addActionEvent() {
        addPatientButton.addActionListener(this);
        exitButton.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == addPatientButton) {
            new AddPatient().setVisible(true);
        }
        this.dispose();
    }
```

```
        public static void main(String[] args) {
                @SuppressWarnings("unused")
                HomePage home_frame = new HomePage();
}
```

Add Patient Page:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AddPatient extends JFrame implements ActionListener {

        private static final long serialVersionUID = 1L;

        public String record = "";

        public String getRecord() {
                return this.record;

        }

        Container container = getContentPane();

        private final JLabel patientDetailsLabel = new JLabel("Add Patient
Details");

        JLabel pnameLabel = new JLabel("Patient Name :");
        JTextField pnameTextField = new JTextField();

        JLabel pageLabel = new JLabel("Patient Age :");
        JTextField pageTextField = new JTextField();

        JLabel specialityLabel = new JLabel("Speciality :");
```

```
static ButtonGroup Speciality = new ButtonGroup();
JRadioButton rdbtnOrtho = new JRadioButton("Ortho");
JRadioButton rdbtnDental = new JRadioButton("Dental");

JButton addPatientButton = new JButton("Add Patient");
JButton clearButton = new JButton("Clear");

String pnameText = pnameTextField.getText().trim();
int page = 0;
int flag = 0;

AddPatient() {
        setLayoutManager();
        setLocationAndSize();
        addComponentsToContainer();
        addActionEvent();

}

public void setLayoutManager() {
        container.setLayout(null);
}

public void setLocationAndSize() {

        patientDetailsLabel.setFont(new Font("Arial", Font.PLAIN, 26));
        patientDetailsLabel.setBounds(280, 30, 250, 70);

        pnameLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        pnameLabel.setBounds(164, 150, 200, 30);

        pnameTextField.setFont(new Font("Arial", Font.PLAIN, 22));
        pnameTextField.setBounds(464, 150, 200, 30);
        pnameTextField.setText(null);

        pageLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        pageLabel.setBounds(164, 200, 200, 30);
        pageTextField.setText(null);

        pageTextField.setFont(new Font("Arial", Font.PLAIN, 22));
        pageTextField.setBounds(464, 200, 200, 30);

        specialityLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        specialityLabel.setBounds(164, 260, 200, 30);

        rdbtnOrtho.setFont(new Font("Arial", Font.PLAIN, 22));
        rdbtnOrtho.setBounds(464, 260, 200, 30);

        rdbtnDental.setFont(new Font("Arial", Font.PLAIN, 22));
        rdbtnDental.setBounds(464, 300, 200, 30);

        Speciality.add(rdbtnOrtho);
        Speciality.add(rdbtnDental);

        addPatientButton.setFont(new Font("Arial", Font.PLAIN, 20));
        addPatientButton.setBounds(231, 341, 143, 45);

        clearButton.setFont(new Font("Arial", Font.PLAIN, 20));
        clearButton.setBounds(433, 341, 131, 45);

        this.setTitle("Add Patient Page");
        this.setVisible(true);
        this.setBounds(10, 10, 832, 551);
        this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        this.setResizable(false);
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = this.getSize().width;
        int h = this.getSize().height;
        int x = (dim.width - w) / 2;
        int y = (dim.height - h) / 2;
        this.setLocation(x, y);

    }
    public void addComponentsToContainer() {
        container.add(pnameLabel);
        container.add(pnameTextField);
        container.add(pageLabel);
        container.add(pageTextField);
        container.add(specialityLabel);
        container.add(rdbtnOrtho);
        container.add(rdbtnDental);
        container.add(addPatientButton);
        container.add(clearButton);
        getContentPane().add(patientDetailsLabel);
    }

    public void addActionEvent() {
        addPatientButton.addActionListener(this);
        clearButton.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == addPatientButton) {

            String pname = pnameTextField.getText().trim();
            int page = 0;
            String pspeciality = "";

            try {
                page =
Integer.parseInt(pageTextField.getText().trim());
            }

            catch (NumberFormatException e1) {
                JOptionPane.showMessageDialog(this, "Check
Details! Enter properly.");

            }
            if (rdbtnOrtho.isSelected())
                pspeciality = "Ortho";
            else
                pspeciality = "Dental";

            new Transaction(pname, page, pspeciality);
            dispose();

        }

        if (e.getSource() == clearButton) {
            pnameTextField.setText("");
            pageTextField.setText("");
            Speciality.clearSelection();

        }

    }

    public static void main(String[] args) {
        @SuppressWarnings("unused")
        AddPatient add_patient_frame = new AddPatient();

    }

}
```

Transaction Page:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class Transaction extends JFrame implements ActionListener {

        private static final long serialVersionUID = 1L;
        Container container = getContentPane();
        String pname = "";
        int page = 0;
        String speciality = "";
        float amt = 0;
        int flag = 0;

        String filename = "";
        String record="";

        private final JLabel transactionDetailsLabel = new JLabel("Transaction
Details");

        JLabel baseCostLabel = new JLabel("Base Amount :");
        JTextField baseCostTextField = new JTextField();

        JLabel taxLabel = new JLabel("Service Tax :");
        JLabel taxValueLabel = new JLabel("5%");

        JLabel gstTaxLabel = new JLabel("GST Tax :");
        JLabel gstValueTaxLabel = new JLabel("18%");
```

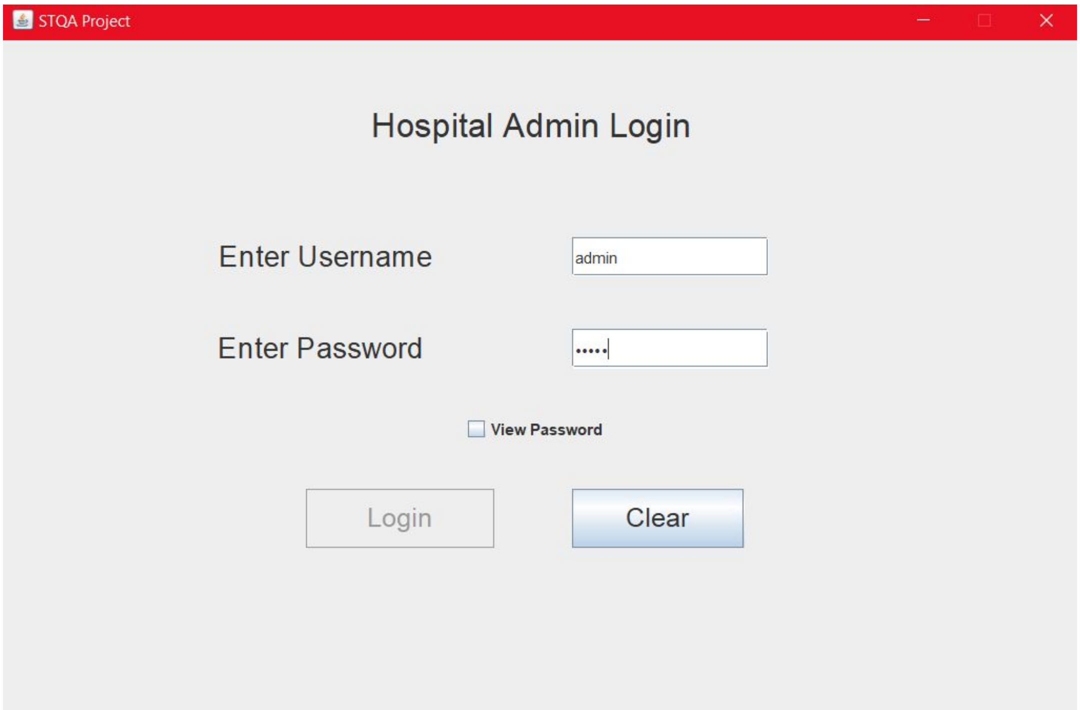
```
    JLabel totalAmt = new JLabel("Gross Payable Amt :");
    JLabel totalAmtLabel = new JLabel("");

    JButton generateBillButton = new JButton("Generate & Save Bill");
    JButton clearButton = new JButton("Clear");

    Transaction(String pname, int page, String pspeciality) {

        this.pname = pname;
        this.page = page;
        this.speciality = pspeciality;


        setLayoutManager();
        setLocationAndSize();
        addComponentsToContainer();
        addActionEvent();

    }

    public void setLayoutManager() {
        container.setLayout(null);
    }

    public void setLocationAndSize() {

        transactionDetailsLabel.setFont(new Font("Arial", Font.PLAIN,
26));
        transactionDetailsLabel.setBounds(280, 30, 250, 70);

        baseCostLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        baseCostLabel.setBounds(164, 150, 200, 30);
        baseCostTextField.setFont(new Font("Arial", Font.PLAIN, 22));
        baseCostTextField.setBounds(464, 150, 200, 30);

        taxLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        taxLabel.setBounds(164, 200, 200, 30);
        taxValueLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        taxValueLabel.setBounds(464, 200, 200, 30);

        gstTaxLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        gstTaxLabel.setBounds(164, 250, 200, 30);
        gstValueTaxLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        gstValueTaxLabel.setBounds(464, 250, 200, 30);

        totalAmt.setFont(new Font("Arial", Font.PLAIN, 22));
        totalAmt.setBounds(164, 300, 200, 30);
        totalAmtLabel.setFont(new Font("Arial", Font.PLAIN, 22));
        totalAmtLabel.setBounds(464, 300, 200, 30);
        totalAmtLabel.setVisible(false);

        generateBillButton.setFont(new Font("Arial", Font.PLAIN, 20));
        generateBillButton.setBounds(175, 350, 275, 30);

        clearButton.setFont(new Font("Arial", Font.PLAIN, 20));
        clearButton.setBounds(500, 350, 131, 30);


        this.setTitle("Transaction Page");
        this.setVisible(true);
        this.setBounds(10, 10, 832, 551);
        this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        this.setResizable(false);
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = this.getSize().width;
        int h = this.getSize().height;
        int x = (dim.width - w) / 2;
        int y = (dim.height - h) / 2;
        this.setLocation(x, y);

    }

    public void addComponentsToContainer() {
        container.add(baseCostLabel);
```

```
		container.add(baseCostTextField);
		container.add(taxLabel);
		container.add(taxValueLabel);
		container.add(gstTaxLabel);
		container.add(gstValueTaxLabel);
		container.add(totalAmt);
		container.add(totalAmtLabel);
		container.add(generateBillButton);
		container.add(clearButton);
		getContentPane().add(transactionDetailsLabel);
	}

	public void addActionEvent() {
		generateBillButton.addActionListener(this);
		clearButton.addActionListener(this);
	}

	public float getAmount() {

		try {
			amt = Integer.parseInt(baseCostTextField.getText().trim());
			if (amt > 0 && pname!="") {
				amt = (float) (amt + amt * 0.05 + amt * 0.18);
				String a = String.valueOf(amt);
				totalAmtLabel.setText(a);
				totalAmtLabel.setVisible(true);
				flag = 1;

				this.record = "Patient Name: " + pname + "\r\n" +
"Patient Age: " + page + "\r\n" + "Speciality Treatment: "
						+ speciality + "\r\n" + "Total Bill Amt: "
+ amt + "\r\n" + "----------------------------";

				this.filename = pname + "_" + page + ".txt";

			} else
```

```
                JOptionPane.showMessageDialog(null, "Amount
Cannot Be 0 or Negative");

        }

        catch (NumberFormatException e1) {
            JOptionPane.showMessageDialog(this, "Check Details!
Enter properly.");

        }
        return amt;
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == generateBillButton) {
            this.getAmount();

            if(pname!="" && amt>0)
            {
                JOptionPane.showMessageDialog(null, "Record
Saved with amt: " + amt);
                new HomePage();
                dispose();
            }

            try {
                FileWriter writer = new FileWriter(filename, true);
                BufferedWriter bufferedWriter = new
BufferedWriter(writer);

         bufferedWriter.write(record);
         bufferedWriter.newLine();
         bufferedWriter.close();

            } catch (IOException e2) {
                // TODO Auto-generated catch block
                e2.printStackTrace();
            }
        }

        if (e.getSource() == clearButton) {
            baseCostTextField.setText("");
            totalAmtLabel.setText("");
        }

    }

    public static void main(String[] args) {
        @SuppressWarnings("unused")
        Transaction add_transaction_frame = new Transaction("A", 58,
"Dental");

    }

}
```

Screenshots

Login page initial:



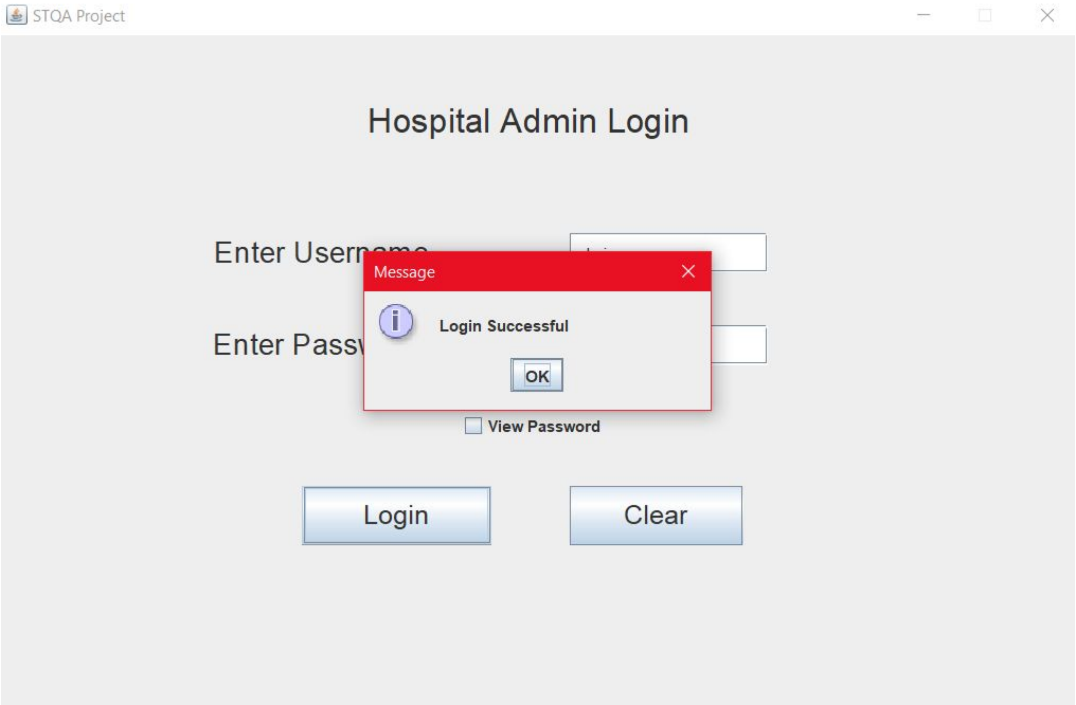
STQA Project

Hospital Admin Login

Enter Username: admin

Enter Password: •••••

Successfully Logged In dialogue box:



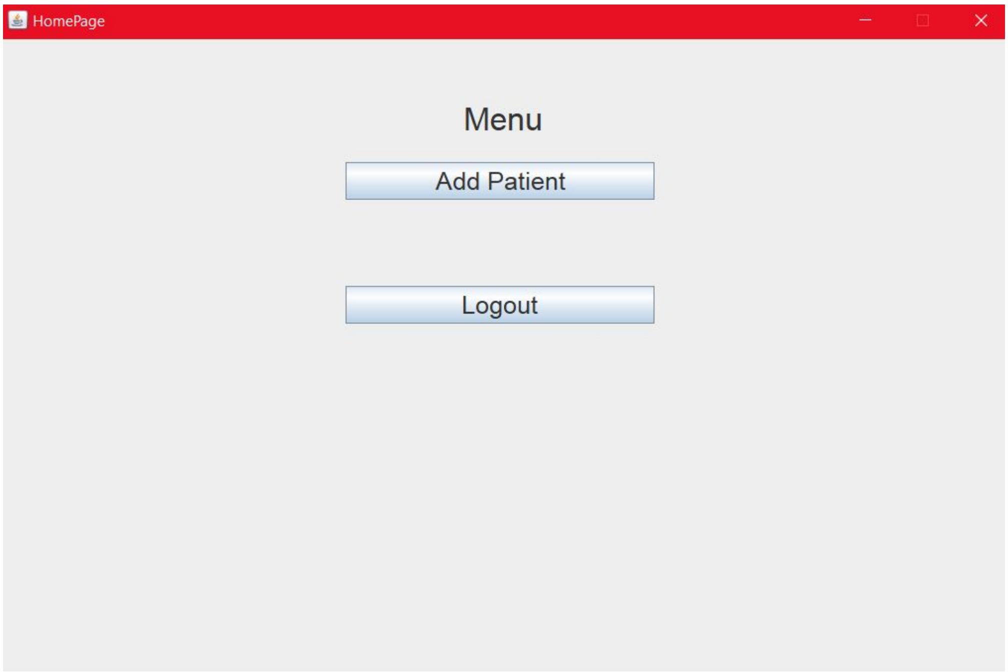
STQA Project

Hospital Admin Login

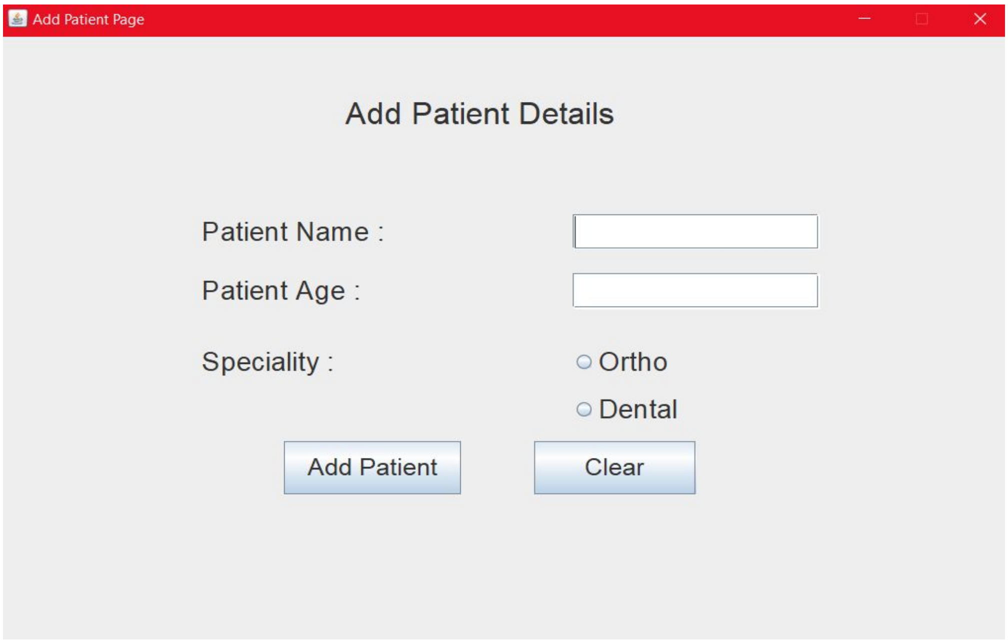
Message

Login Successful

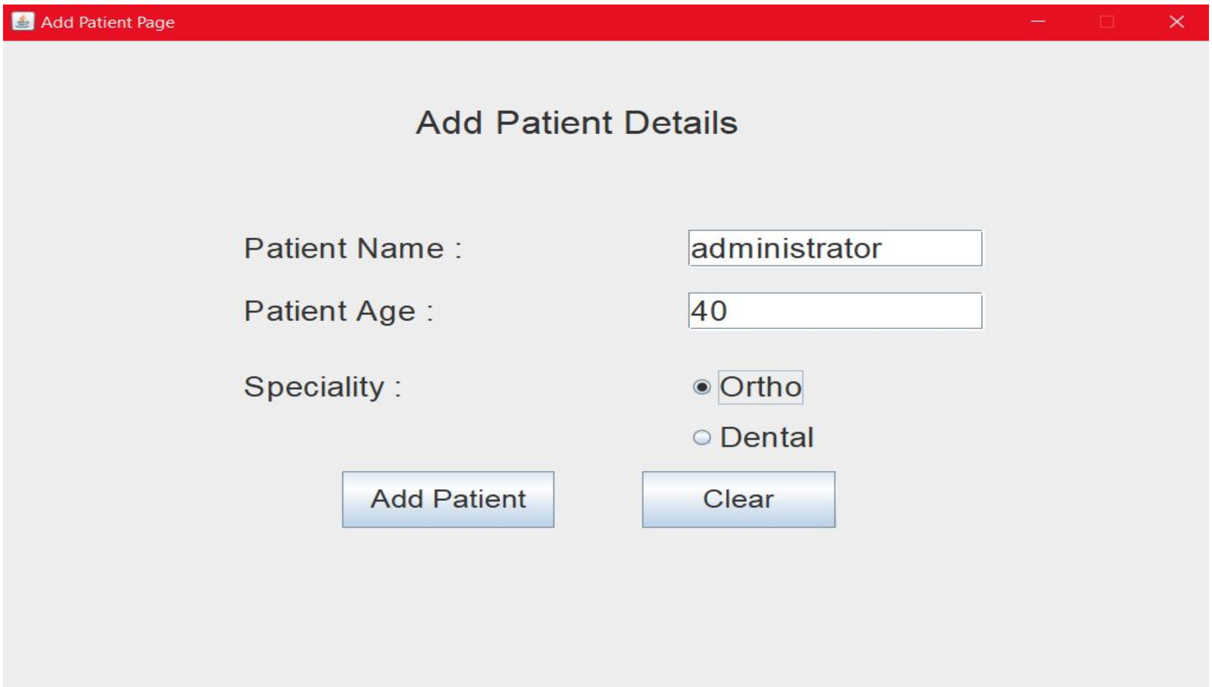
Home Page (Menu):



Adding Patient:



Patient Details:



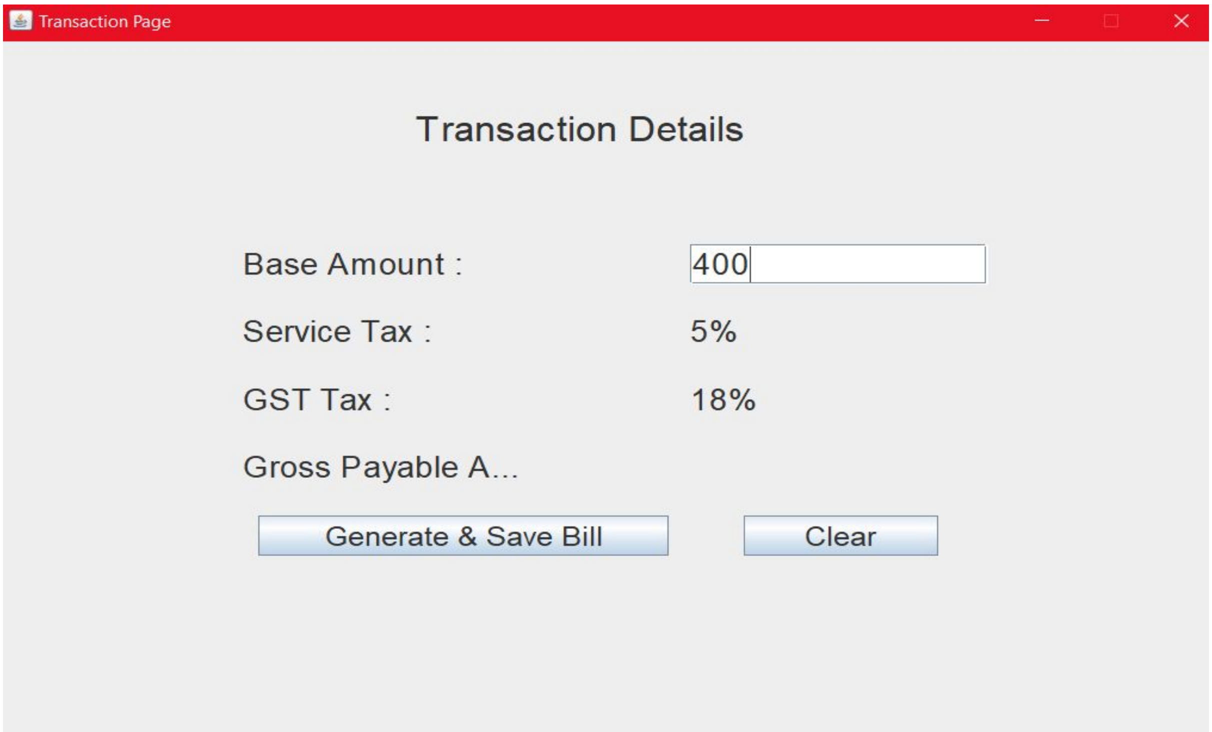
Add Patient Details

Patient Name : administrator

Patient Age : 40

Speciality : ● Ortho ○ Dental

Transaction page:



Transaction Details

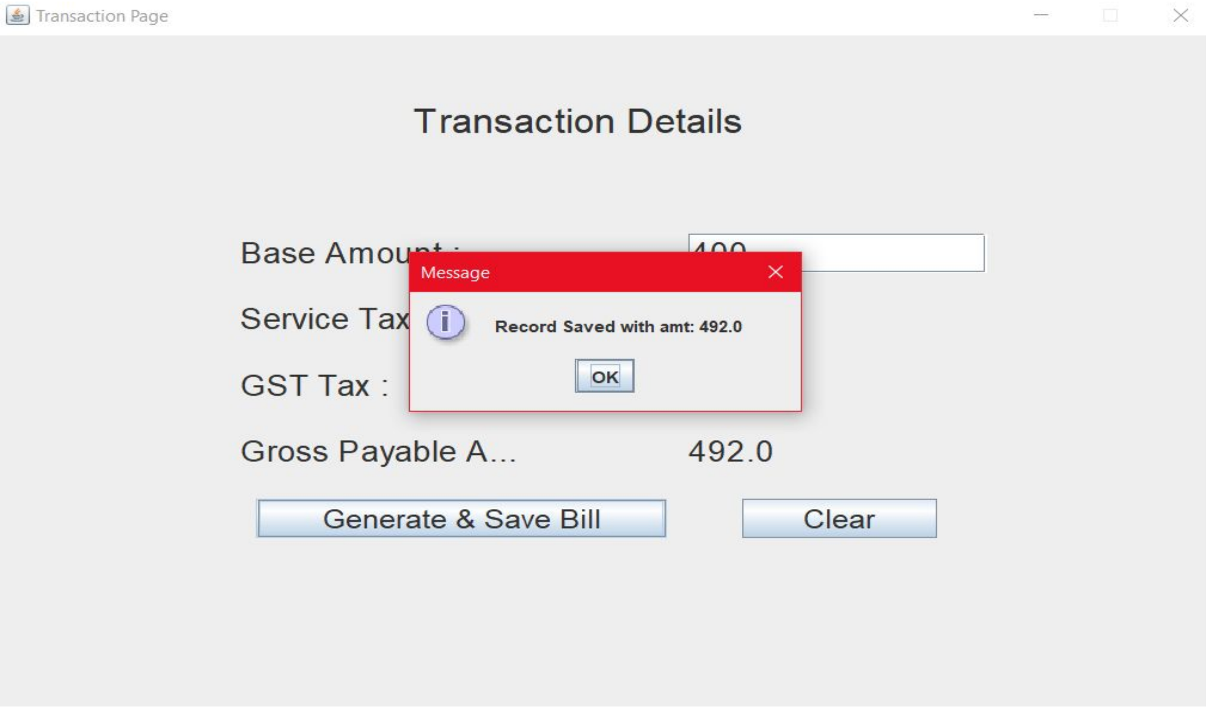
Base Amount : 400

Service Tax : 5%

GST Tax : 18%

Gross Payable A...

Invoice Generated dialogue box:

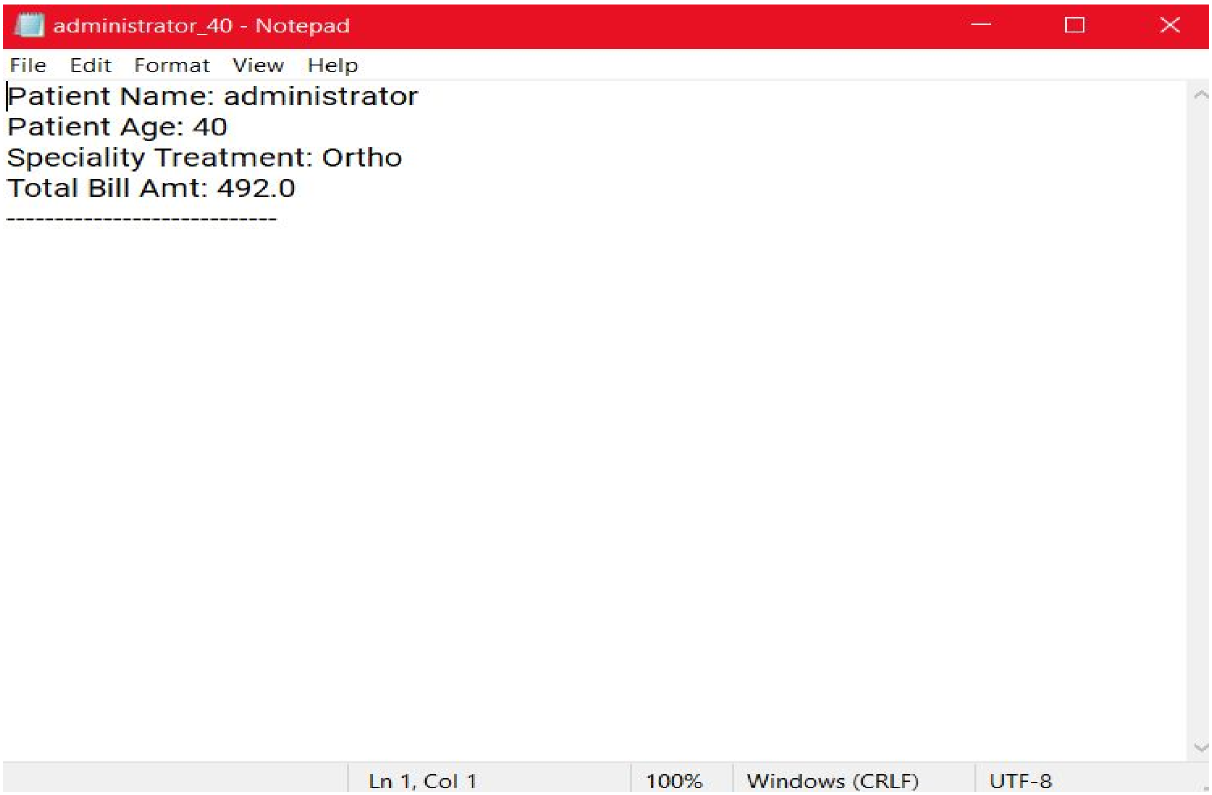


400

492.0

Record Saved with amt: 492.0

Medical Record Files stored in local drive of project:



```
Patient Name: administrator
Patient Age: 40
Speciality Treatment: Ortho
Total Bill Amt: 492.0
----------------------------
```

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

Out of Scope :-

The following types of testing are out of scope for this application :-

▪ Security testing

▪ Performance, volume, and stress testing

Test Objectives :-

Targeted number of defects— 30

▪ Targeted number of test scenario— 50

▪ Targeted number of test cases—20

▪ Test scenario writing per day—10

▪ Test case writing per day—10

▪ Test case execution per day—10

▪ Coverage of requirements P1 = 100% P2 = 50% P3 = 10%

Assumptions of Testing :-

▪ Users understand English and they are capable of using a computer.

▪ Application will be working on Windows 10 professionals. Other operating systems, if any, are compatible.

▪ Application is tested on Core i5, quad-core processor. Other configurations used, if any, are compatible.

▪ Testing considers right-handed users as well as left-handed users.

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.

Workflow :-

1. The project manager communicates the availability of new builds along with the records of reviews and unit testing to the test manager.
2. The application is installed on the machine as defined in prerequisites (Core i5).
3. Test manager checks the entry criteria for testing.
4. If the application passes the entry criteria, it is taken for iteration testing.
5. Test cases are stored in the same library.

Test Design :-

Testing Levels Responsibility

Unit Testing when code is generated - Developers

Integration testing - Developers

System testing - Testers

Test Data Definition :-

Test data is defined using the following techniques:-

▪ Error guessing.

▪ Equivalence Partitioning

▪ Identifying probability of glitches/complex operations.

Schedule and Planned Resources :-

The project started on August 29th 2020

Test planning—August 30th, 2020

Test scenario writing, review and updation— August 30th, 2020

Test case writing, review and updation— September 03rd, 2020

Test case execution 1st iteration— September 04th, 2020

Test case execution 2nd iteration— September 05th, 2020

Test Data Management :-

- Test plan, test scenario, test cases and test reports will be saved on local hard drive.
- Backup will be taken on a cloud platform,remote database associated.

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 Exit Criteria :-

Test cases are completed and all defects found in testing are closed, retested and found to be closed.

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.

Tester's Training :-

The following training is planned for testers:-

- Testers will be trained on the basics of the order taking system in restaurants.
- Tester's training for writing test plans, test scenarios, test cases, execution of test cases and defect logging is already done.

Communication :-

- Test team will be meeting every alternate day in the evening for an hour over an online meeting platform to discuss the progress.
- Test team will be meeting with development team at the end of test iteration, over online platforms.

Tools :-

- Bugs are added in a common repository on the local system.
- Defect communication and build release communication will be through internal communication process

Test Scenario

LOGIN PAGE:

Common for All Test Scenarios (TS00.0—Priority P1) :-

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 (TS00.1—Priority P1) 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 (TS00.2—Priority P1) 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 (TS00.3—Priority P1) 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 (TS00.4—Priority P1) 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 (TS00.5—Priority P1) 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 (TS00—Priority P1) :-

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 (TS01—Priority P1) 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 (TS02—Priority P1) 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 (TS00—Priority P1) :-

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 (TS01—Priority P1) 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 (TS02—Priority P1) 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 (TS03—Priority P2) 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 (TS00—Priority P1) :-

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 (TS01—Priority P1) 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 (TS02—Priority P1) 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 (TS01—Priority P1) 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.
Test Sequence:- NA
Test Scenario Traceability:- TS00.0
Test Case Name and Number:- ST/0001
Type of Testing:- Smoke Testing.
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)
Steps to Reproduce:- Start computer and Click the icon on desktop
Expected Results:- Application is launched (Screen is seen with the login of restaurant).

Test case 2

Test Precondition:- Application is launched
Test Sequence:- ST/0001
Test Scenario Traceability :- TS00.0
Test Case Name and Number :- UI/0001
Type of Testing :- User Interface testing
Objectives :- To check that application opens and occupies the screen
Valid/Invalid Conditions :- Valid condition
Test Data :- Not applicable
Steps to Reproduce:- Start computer Click the icon on desktop
Expected Results :- Complete screen is occupied by the layout mentioned. No scroll bar appears.

Test case 3

Test Precondition:- Application opened.
Test Sequence:- UI/0001
Test Scenario Traceability:- TS00.1
Test Case Name and Number:- UI/0001
Type of Testing:- Functional Testing.
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.
Test Sequence:- UI/0001
Test Scenario Traceability:- TS00.2
Test Case Name and Number:- FUN/0002
Type of Testing:- Functional Testing.
Objectives:- To login in the application.
Valid/invalid Conditions:- InValid
Test Data:- incorrect Username and Password combination.
Expected Results:- Application is not launched.

Test case 5

Test Precondition:- Application is launched and logged in
Test Sequence:- UI/0001
Test Scenario Traceability:- TS00
Test Case Name and Number:- UI/0002
Type of Testing:- User interface testing
Objectives:- To check that menu is shown on opening of the application.
Valid/invalid Conditions:- Valid
Test Data:- NA
Steps to Reproduce:- Launch App
Expected Results:- Home Page launched with no errors.

Test case 6

Test Precondition:- App is launched
Test Sequence:- UI/0002
Test Scenario Traceability:- TS00
Test Case Name and Number:- UI/0003
Type of Testing:-User interface testing
Objectives:- To check that on clicking 'Add Patient' button on the Home Page, user is redirected to the 'Add Patient Details' window.
Valid/invalid Conditions:- Valid
Test Data:- NA
Steps to Reproduce:- Launch App
Expected Results:- 'Add Patient Details' window opened successfully.

Test case 7

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0001
Type of Testing:- Functional Testing
Objectives:- To check that on clicking 'Logout' button on the Home Page, user is redirected to the 'Login' window.
Valid/invalid Conditions:- Valid
Priority:- P1
Steps to Reproduce:- Launch App.
Expected Results:- Logged out successfully and 'Login' window appears.

Test case 8

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0001
Type of Testing:- Functional Testing
Objectives:- To check that on entering null in 'patient name' and 0 in 'patient age', 'Enter Details properly' dialogue box is popped up.
Valid/invalid Conditions:- Invalid

Steps to Reproduce:- Launch App.
Expected Results:- ‘Enter Details properly’ dialogue box is popped up.

Test case 9

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0001
Type of Testing:- Functional Testing
Objectives:- To check that if the user doesn’t opt for any of the speciality(Ortho/Dental), ‘Enter Details properly’ dialogue box is popped up.
Valid/invalid Conditions:- Invalid
Steps to Reproduce:- Launch App.
Expected Results:- ‘Enter Details properly’ dialogue box is popped up.

Test case 10

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0001
Type of Testing:- user interface Testing
Objectives:- Given that all patient details are entered properly, to check if ‘add patient’ button works properly.
Valid/invalid Conditions:- Valid
Steps to Reproduce:- Launch App.
Expected Results:- ‘Transaction’ window appears .

Test case 11

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0002
Type of Testing:- Functional Testing
Objectives:- To check if the base Amount entered is 0 or negative, the application accepts the amount or not.

Valid/invalid Conditions:- Invalid
Steps to Reproduce:- Launch App.
Expected Results:- Dialogue box ‘Base amount cannot be 0 or negative’ is popped up.

Test case 12

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0002
Type of Testing:- Functional Testing
Objectives:- To check whether the ‘generate & save Bill’ button works properly.
Valid/invalid Conditions:- Valid
Steps to Reproduce:- Launch App.
Expected Results:- ‘record saved’ dialogue box is popped out along with value added taxes bill amount.

Test case 13

Test Precondition:- App is launched.
Test Sequence:- UI/0003
Test Scenario Traceability:- TS01
Test Case Name and Number:- FUN/0002
Type of Testing:- Functional Testing
Objectives:- To check on clicking ‘generate & save Bill’ button, medical record is added successfully
Valid/invalid Conditions:- Valid
Steps to Reproduce:- Launch App.
Expected Results:- A new file with Patient’s name is -

- generated if it's the patient’s first appointment.
- updated if the medical record already exists.

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.

A detailed explanation of the bugs and their fixes is also discussed below.

Table :- Tested classes.

CLASS NAME	**TEST FILE**
LoginTest.class	JUnitTesting_LoginPage.class
AddPatient.class	JUnitTesting_AddPatientPage.class
Transaction.class	JUnitTesting_TransactionPage.class

Table :- Tested functions.

TEST CLASS NAMES	**TEST FUNCTION**

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 – Login Button Check
DESCRIPTION :- This JUnit test case was run to ensure that the login button was disable until details were filled properly.
ERRORS FOUND :- **BUG1** was encountered, which was fixed later in 1st iteration.

Test case 3– 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 4 – Null String as Patient name
DESCRIPTION :- This JUnit test case was run to ensure that in the ‘add patient details window’,patient name does not accept null string.
ERRORS FOUND :- **BUG2** was encountered which was fixed in 1st iteration.

Test case 5 – Patient age as a negative integer.
DESCRIPTION :- This JUnit test case was run to ensure that in the ‘add patient details window’,patient age does not accept negative integers.
ERRORS FOUND :- **BUG3** was encountered which was fixed in 1st iteration.

Test case 6 – Transaction Base Cost Amount as a negative value.
DESCRIPTION :- This JUnit test case was run to ensure that in the ‘Transaction Page’, base transaction cost does not accept negative values.
ERRORS FOUND :- **BUG4** was encountered which was fixed in 1st iteration.

Test case 7 – 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.

Test case 8 - Redirection on clicking on Generate Bill and Save Record.
DESCRIPTION :- To check whether, after a new record is saved is generated successfully,the user is redirected to the ‘Home Page’ window.
ERRORS FOUND :- **BUG5** was encountered which was fixed in 2nd iteration.

FOUND AND FIXED BUGS

During testing the following bugs were fixed for the proper functioning of the application.

BUG 1

Description :- Even when the login credentials were not entered initially at the application start, the login button was enabled and working.
Fix :- It was made sure that, when login page credentials are entered then only the login button gets enabled. On proper credentials only the user gets redirected to Homepage

BUG 2

Description :- On the 'Add Patient Details' Page, the Patient name label was accepting the null string.
Fix :- Null String cannot be added as Patient name as it is also the name of the medical record file..

BUG 3

Description :- Quantity text field is empty by default. Hence when an item is selected and if it is added to order without adding the quantity, the app gives garbage results.
Fix :- Added a condition to check if the quantity entered is not null or 0.

BUG 4

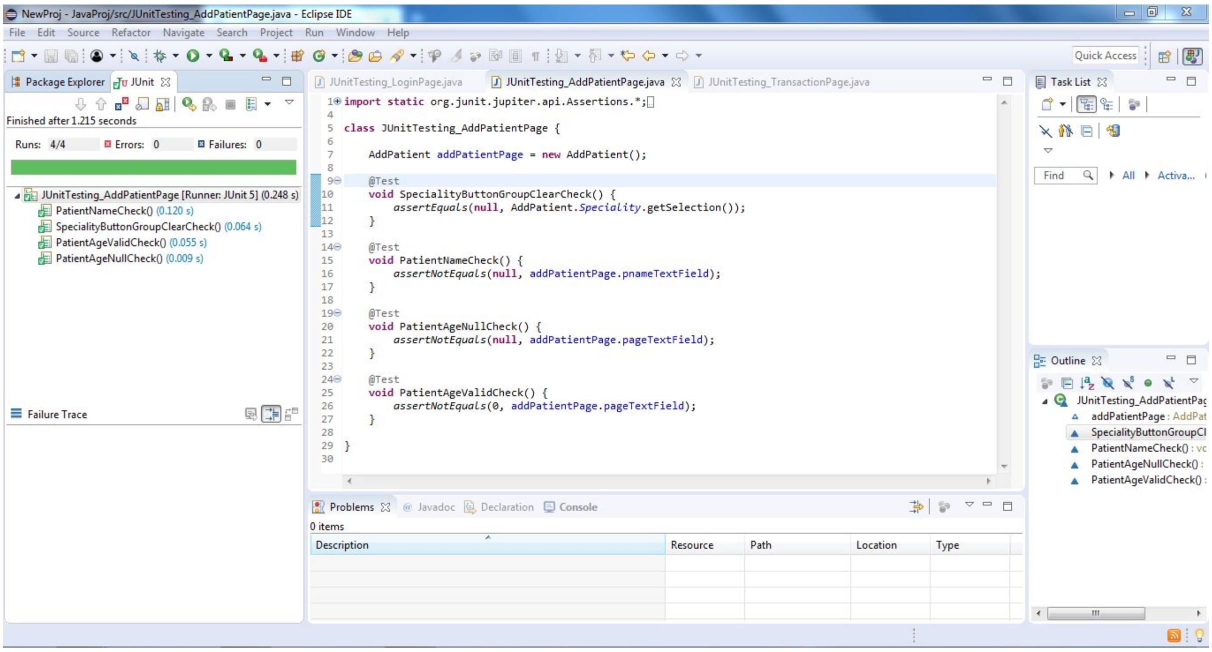
Description :- After successful entry on Add Patient Page, in the 'Transaction Page', the base transaction cost amount was accepting negative values.
Fix :- Applied Java Exception handling on the base transaction cost amount (float) so as to not take negative values.

BUG 5

Description :- On saving the new record and generating the bill,the application stayed on the same window.
Fix :- Once the record is saved and the bill is generated, the user is redirected to the Login Page so that he can further add patients or logout of the application.

JUNIT TESTING

LOGIN PAGE - JUNIT TEST



NewProj - JavaProj/src/JUnitTesting_AddPatientPage.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
JUnit
JUnitTesting_LoginPage.java
JUnitTesting_AddPatientPage.java
JUnitTesting_TransactionPage.java
Task List
Find
All
Activa...
Outline
Problems
Javadoc
Declaration
Console
Failure Trace

Finished after 1.215 seconds

Runs: 4/4 Errors: 0 Failures: 0

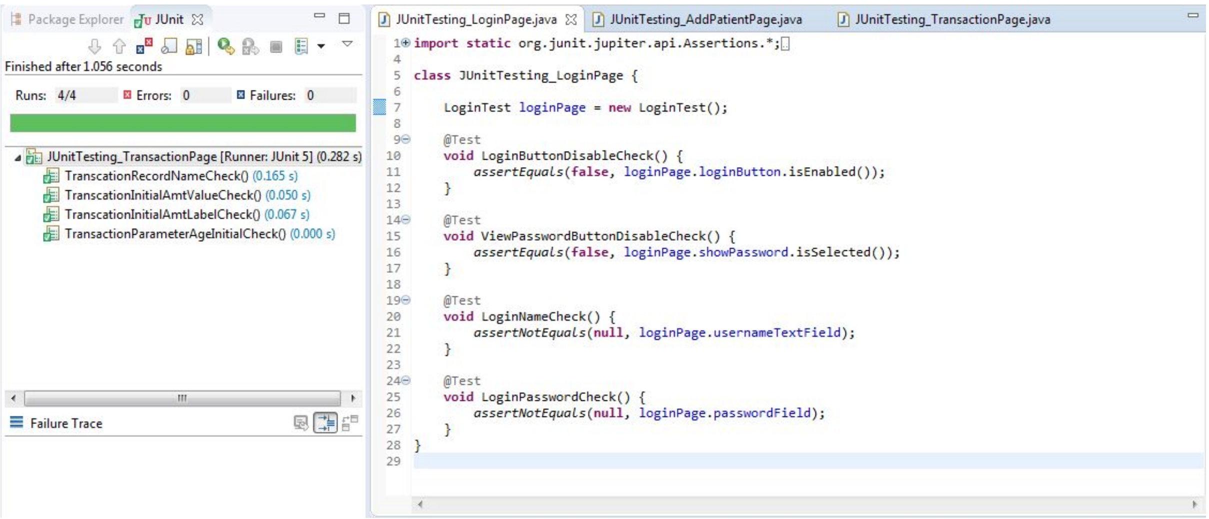
```
JUnitTesting_AddPatientPage [Runner: JUnit 5] (0.248 s)
    PatientNameCheck() (0.120 s)
    SpecialityButtonGroupClearCheck() (0.064 s)
    PatientAgeValidCheck() (0.055 s)
    PatientAgeNullCheck() (0.009 s)
```

```java
import static org.junit.jupiter.api.Assertions.*;

class JUnitTesting_AddPatientPage {

    AddPatient addPatientPage = new AddPatient();

    @Test
    void SpecialityButtonGroupClearCheck() {
        assertEquals(null, AddPatient.Speciality.getSelection());
    }

    @Test
    void PatientNameCheck() {
        assertNotEquals(null, addPatientPage.pnameTextField);
    }

    @Test
    void PatientAgeNullCheck() {
        assertNotEquals(null, addPatientPage.pageTextField);
    }

    @Test
    void PatientAgeValidCheck() {
        assertNotEquals(0, addPatientPage.pageTextField);
    }

}
```

```
JUnitTesting_AddPatientPag
    addPatientPage : AddPat
    SpecialityButtonGroupCl
    PatientNameCheck() : vo
    PatientAgeNullCheck() :
    PatientAgeValidCheck() :
```

0 items

Description	Resource	Path	Location	Type



Package Explorer
JUnit
JUnitTesting_LoginPage.java
JUnitTesting_AddPatientPage.java
JUnitTesting_TransactionPage.java
Failure Trace

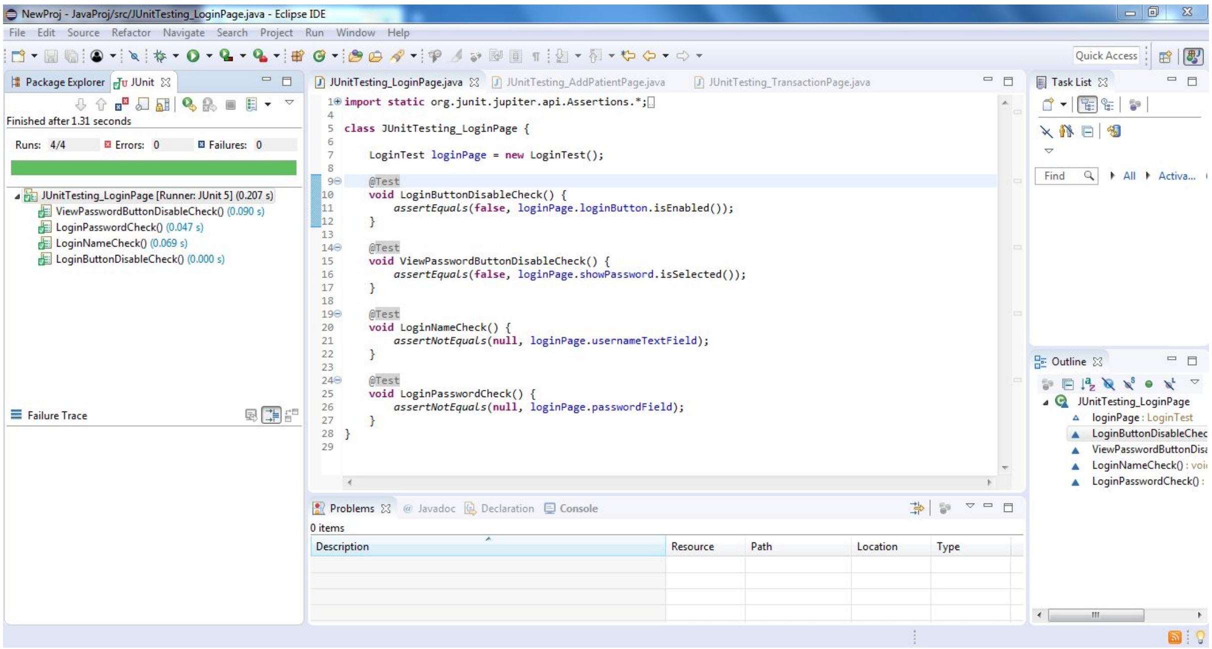
Finished after 1.056 seconds

Runs: 4/4 Errors: 0 Failures: 0

```
JUnitTesting_TransactionPage [Runner: JUnit 5] (0.282 s)
    TranscationRecordNameCheck() (0.165 s)
    TranscationInitialAmtValueCheck() (0.050 s)
    TransactionInitialAmtLabelCheck() (0.067 s)
    TransactionParameterAgeInitialCheck() (0.000 s)
```

```java
import static org.junit.jupiter.api.Assertions.*;

class JUnitTesting_LoginPage {

    LoginTest loginPage = new LoginTest();

    @Test
    void LoginButtonDisableCheck() {
        assertEquals(false, loginPage.loginButton.isEnabled());
    }

    @Test
    void ViewPasswordButtonDisableCheck() {
        assertEquals(false, loginPage.showPassword.isSelected());
    }

    @Test
    void LoginNameCheck() {
        assertNotEquals(null, loginPage.usernameTextField);
    }

    @Test
    void LoginPasswordCheck() {
        assertNotEquals(null, loginPage.passwordField);
    }
}
```

ADD PATIENT PAGE - JUNIT TEST


NewProj - JavaProj/src/JUnitTesting_LoginPage.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
JUnit
JUnitTesting_LoginPage.java
JUnitTesting_AddPatientPage.java
JUnitTesting_TransactionPage.java
Task List
Find
All
Activa...
Outline
Failure Trace
Problems
Javadoc
Declaration
Console

Finished after 1.31 seconds

Runs: 4/4 Errors: 0 Failures: 0

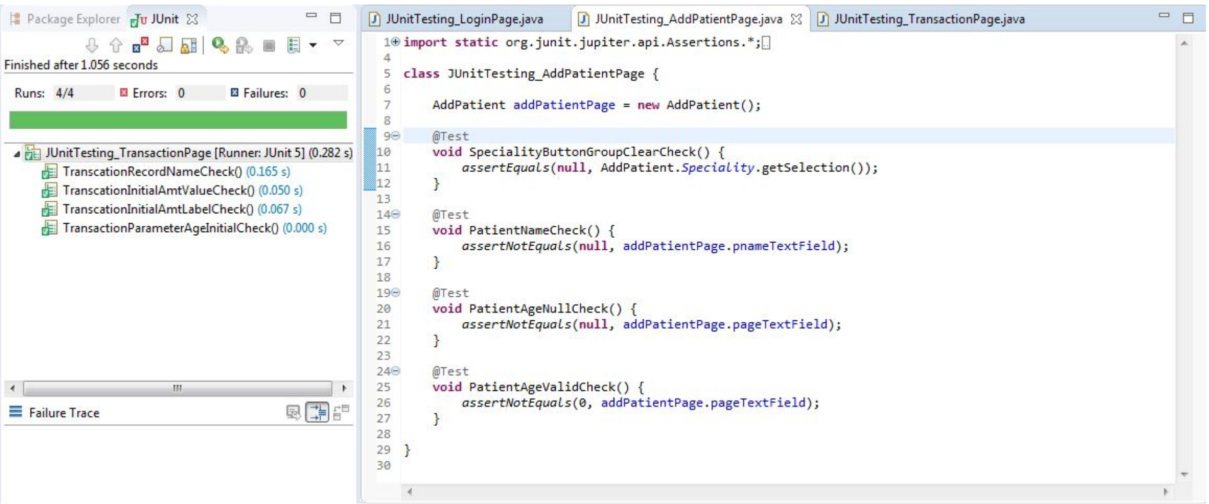
```
JUnitTesting_LoginPage [Runner: JUnit 5] (0.207 s)
    ViewPasswordButtonDisableCheck() (0.090 s)
    LoginPasswordCheck() (0.047 s)
    LoginNameCheck() (0.069 s)
    LoginButtonDisableCheck() (0.000 s)
```

```java
import static org.junit.jupiter.api.Assertions.*;

class JUnitTesting_LoginPage {

    LoginTest loginPage = new LoginTest();

    @Test
    void LoginButtonDisableCheck() {
        assertEquals(false, loginPage.loginButton.isEnabled());
    }

    @Test
    void ViewPasswordButtonDisableCheck() {
        assertEquals(false, loginPage.showPassword.isSelected());
    }

    @Test
    void LoginNameCheck() {
        assertNotEquals(null, loginPage.usernameTextField);
    }

    @Test
    void LoginPasswordCheck() {
        assertNotEquals(null, loginPage.passwordField);
    }
}
```

```
JUnitTesting_LoginPage
    loginPage : LoginTest
    LoginButtonDisableChec
    ViewPasswordButtonDisa
    LoginNameCheck() : voi
    LoginPasswordCheck() :
```

0 items

Description	Resource	Path	Location	Type


Package Explorer
JUnit
Failure Trace
JUnitTesting_LoginPage.java
JUnitTesting_AddPatientPage.java
JUnitTesting_TransactionPage.java

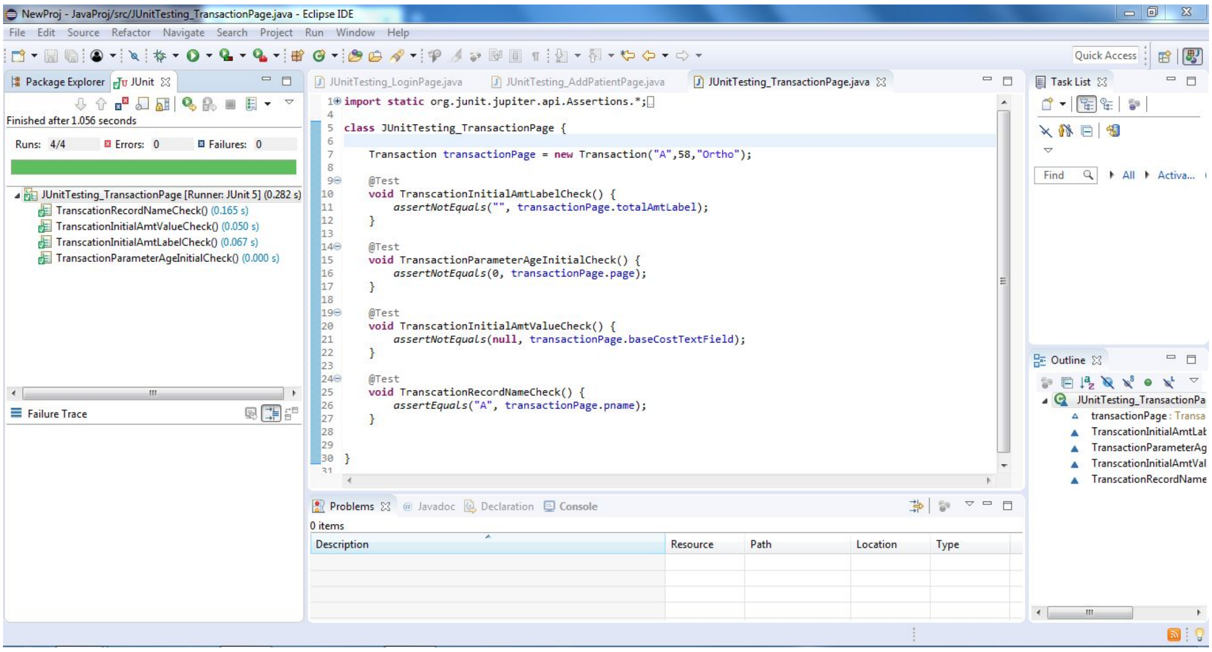
Finished after 1.056 seconds

Runs: 4/4 Errors: 0 Failures: 0

```
JUnitTesting_TransactionPage [Runner: JUnit 5] (0.282 s)
    TranscationRecordNameCheck() (0.165 s)
    TranscationInitialAmtValueCheck() (0.050 s)
    TranscationInitialAmtLabelCheck() (0.067 s)
    TransactionParameterAgeInitialCheck() (0.000 s)
```

```java
import static org.junit.jupiter.api.Assertions.*;

class JUnitTesting_AddPatientPage {

    AddPatient addPatientPage = new AddPatient();

    @Test
    void SpecialityButtonGroupClearCheck() {
        assertEquals(null, AddPatient.Speciality.getSelection());
    }

    @Test
    void PatientNameCheck() {
        assertNotEquals(null, addPatientPage.pnameTextField);
    }

    @Test
    void PatientAgeNullCheck() {
        assertNotEquals(null, addPatientPage.pageTextField);
    }

    @Test
    void PatientAgeValidCheck() {
        assertNotEquals(0, addPatientPage.pageTextField);
    }
}
```

TRANSACTION PAGE - JUNIT TESTING



NewProj - JavaProj/src/JUnitTesting_TransactionPage.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
JUnit
JUnitTesting_LoginPage.java
JUnitTesting_AddPatientPage.java
JUnitTesting_TransactionPage.java
Task List
Find
All
Activa...
Outline
Failure Trace
Problems
Javadoc
Declaration
Console
0 items
Description
Resource
Path
Location
Type

Finished after 1.056 seconds

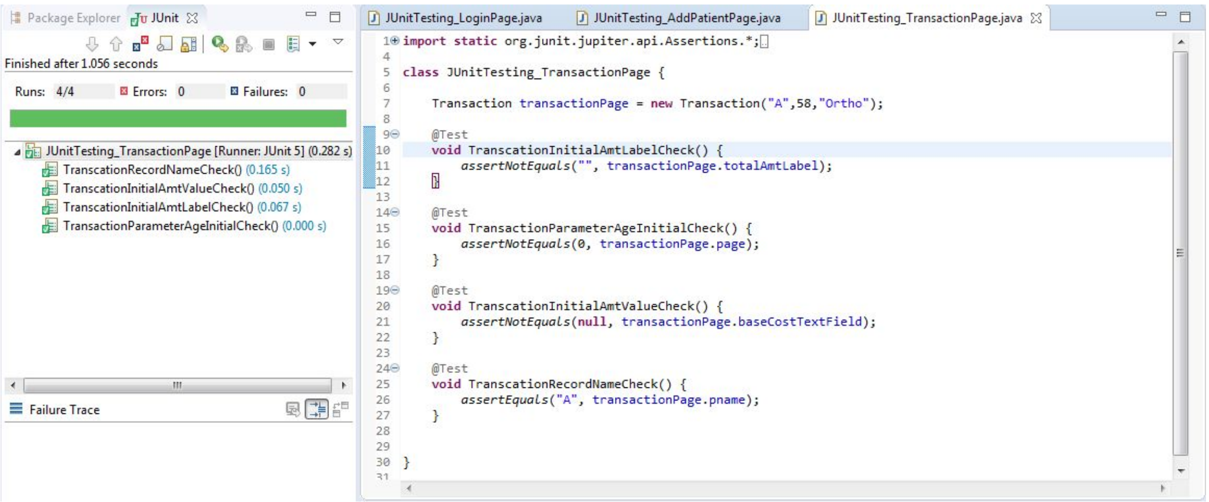
Runs: 4/4 Errors: 0 Failures: 0

- JUnitTesting_TransactionPage [Runner: JUnit 5] (0.282 s)
  - TranscationRecordNameCheck() (0.165 s)
  - TranscationInitialAmtValueCheck() (0.050 s)
  - TranscationInitialAmtLabelCheck() (0.067 s)
  - TransactionParameterAgeInitialCheck() (0.000 s)

```java
import static org.junit.jupiter.api.Assertions.*;

class JUnitTesting_TransactionPage {

	Transaction transactionPage = new Transaction("A",58,"Ortho");

	@Test
	void TranscationInitialAmtLabelCheck() {
		assertNotEquals("", transactionPage.totalAmtLabel);
	}

	@Test
	void TransactionParameterAgeInitialCheck() {
		assertNotEquals(0, transactionPage.page);
	}

	@Test
	void TranscationInitialAmtValueCheck() {
		assertNotEquals(null, transactionPage.baseCostTextField);
	}

	@Test
	void TranscationRecordNameCheck() {
		assertEquals("A", transactionPage.pname);
	}


}
```

Outline:

- JUnitTesting_TransactionPa
  - transactionPage : Transa
  - TranscationInitialAmtLak
  - TransactionParameterAg
  - TranscationInitialAmtVal
  - TranscationRecordName



Package Explorer
JUnit
JUnitTesting_LoginPage.java
JUnitTesting_AddPatientPage.java
JUnitTesting_TransactionPage.java
Failure Trace

Finished after 1.056 seconds

Runs: 4/4 Errors: 0 Failures: 0

- JUnitTesting_TransactionPage [Runner: JUnit 5] (0.282 s)
  - TranscationRecordNameCheck() (0.165 s)
  - TranscationInitialAmtValueCheck() (0.050 s)
  - TranscationInitialAmtLabelCheck() (0.067 s)
  - TransactionParameterAgeInitialCheck() (0.000 s)

```java
import static org.junit.jupiter.api.Assertions.*;

class JUnitTesting_TransactionPage {

	Transaction transactionPage = new Transaction("A",58,"Ortho");

	@Test
	void TranscationInitialAmtLabelCheck() {
		assertNotEquals("", transactionPage.totalAmtLabel);
	}

	@Test
	void TransactionParameterAgeInitialCheck() {
		assertNotEquals(0, transactionPage.page);
	}

	@Test
	void TranscationInitialAmtValueCheck() {
		assertNotEquals(null, transactionPage.baseCostTextField);
	}

	@Test
	void TranscationRecordNameCheck() {
		assertEquals("A", transactionPage.pname);
	}


}
```