**JAVA SWING BASED – Ayush Hospital Finder- SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

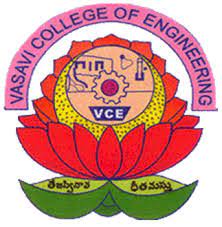
**IN**

**INFORMATION TECHNOLOGY**

**By**

**I. Aniruth Bairi <1602-21-737-007>**

**Under the guidance of Ms B. Leelavathy**



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2022-2023**

**BONAFIDE CERTIFICATE**

This is to certify that this project report titled

‘***Ayush Hospital Finder***’

is a project work of ***I. Aniruth Bairi***

bearing roll no. 1602-21-737-007

who carried out

this project under my supervision

in the IV semester

of the academic year 2022- 2023

Signature Signature

External Examiner Internal Examiner

CONTENTS:

1.Problem Statement

2.Abstract

3.Design Requirements

4.ER Diagram

5.DDL Commands

6.DML Commands

7.Implementation

8.Output

9.Result

10.Discussion and Future Work

11.References

**PROBLEM STATEMENT:**

Develop a application which shows the location and nearby Ayush hospital with opening time and closing timing and by integrating various bio-medical data sources, containing information relevant to the hospital demographics, their inpatient procedure rates, Outpatient department etc.

**ABSTRACT**

The Ayush Hospital Finder application is designed to help users locate and access Ayurveda, Yoga, and Naturopathy hospitals and medical practitioners in their vicinity. The application integrates various biomedical data sources and displays relevant information about the hospitals, including demographics, inpatient procedure rates, outpatient department details, and opening and closing times. Users can search for hospitals based on their location, preferences, and specific criteria, book appointments with doctors, and provide feedback on their experiences. The application provides a user-friendly interface, GPS and map integration, search filters, and secure data management to ensure a comprehensive and seamless user experience. The Ayush Hospital Finder application aims to empower users to make informed healthcare decisions and improve their access to alternative healthcare options.

**Design Requirements:**

List of tables, its attributes and their domains:

1)Table-1: Hospitals Table

* HID- NUMBER (Primary key)
* HName-VARCHAR2
* HBranch-VARCHAR2
* HPhone-NUMBER(Unique)
* Opening\_Time- TIMESTAMP
* Closing\_Time-TIMESTAMP
* Rating-NUMBER (CHECK-between 1 and 10)

2)Table-2: Specializes Table

* SID-NUMBER (Primary Key)
* SName-VARCHAR2
* Cost-NUMBER

3)Table-3: Doctors Table

* DID-NUMBER (Primary key)
* DName-VARCHAR2
* Doctor\_SID-NUMBER (Foreign key to Specializations table)
* Doctor\_HID-NUMBER (Foreign key to Hospitals table)

4)Table-4: Users Table

* UID-NUMBER (Primary Key)
* FName-VARCHAR2
* LName-VARCHAR2
* Blood\_Group-VARCHAR2
* Gender-VARCHAR2
* P­­­hone-NUMBER

5)Table-5: Inpatients Table

* IID-NUMBER (Primary Key)
* IName- VARCHAR2
* Rate-NUMBER
* HID-NUMBER (Foreign Key to Hospitals Table)

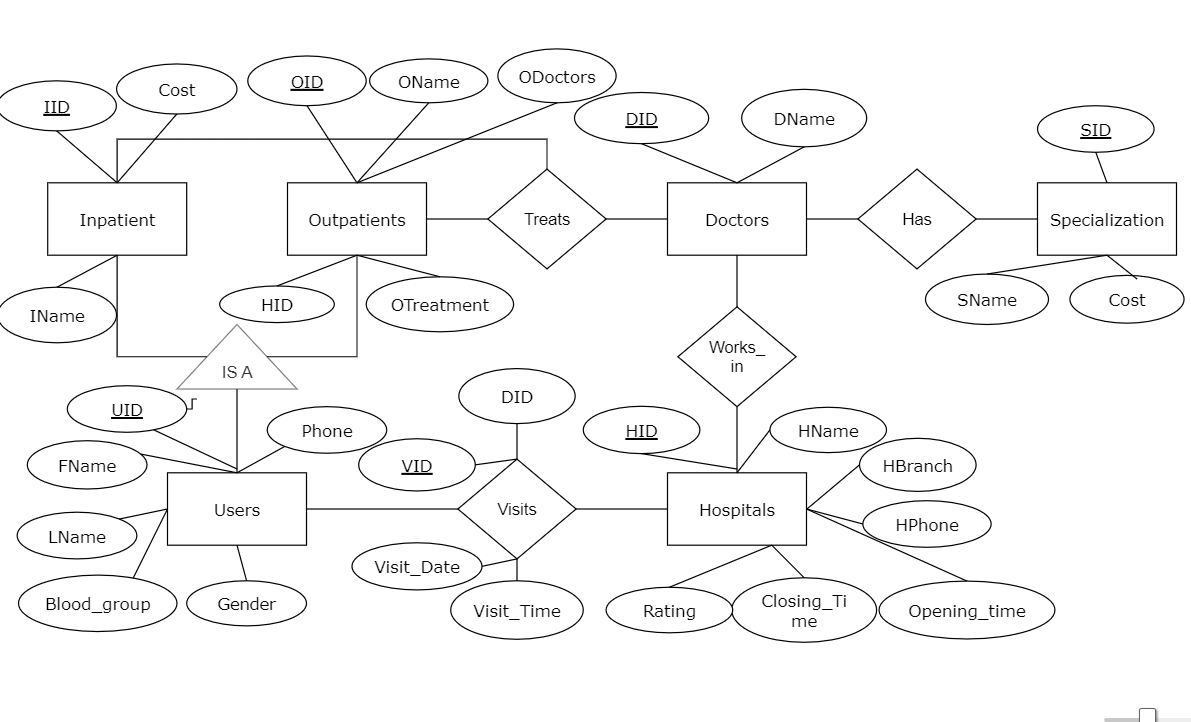
6)Table-6: Outpatients Table

* OID-NUMBER (Primary Key)
* OName-VARCHAR2
* ODoctors-VARCHAR2
* OTreatments-VARCHAR2
* HID-NUMBER (Foreign Key to Hospitals Table)

7)Table-7: Visits

* VID-NUMBER (Primary Key)
* UID-NUMBER (Foreign Key to Users Table)
* HID-NUMBER (Foreign Key to Hospitals Table)
* DID-NUMBER (Foreign Key to Doctors Table)
* Visit\_Date-DATE
* Visit\_Time-TIMESTAMP

**ER DIAGRAM:**

****

**MAPPING CARDINALITIES**

Many Doctors work at one Hospital. Therefore, the mapping cardinality between Hospitals and Doctors is (1-N).

Many patients visit one Hospital. Therefore, the mapping cardinality between Hospital and Patients is (1-N).

One doctor may have more than one Specialization. Therefore, the mapping cardinality between Doctor and Specializations is (1-N).

**DDL COMMANDS:**

1.Creating Table for Hospitals:

**QUERY:**

create table hospitals(

hid number(3) primary key,

hname varchar2(20),

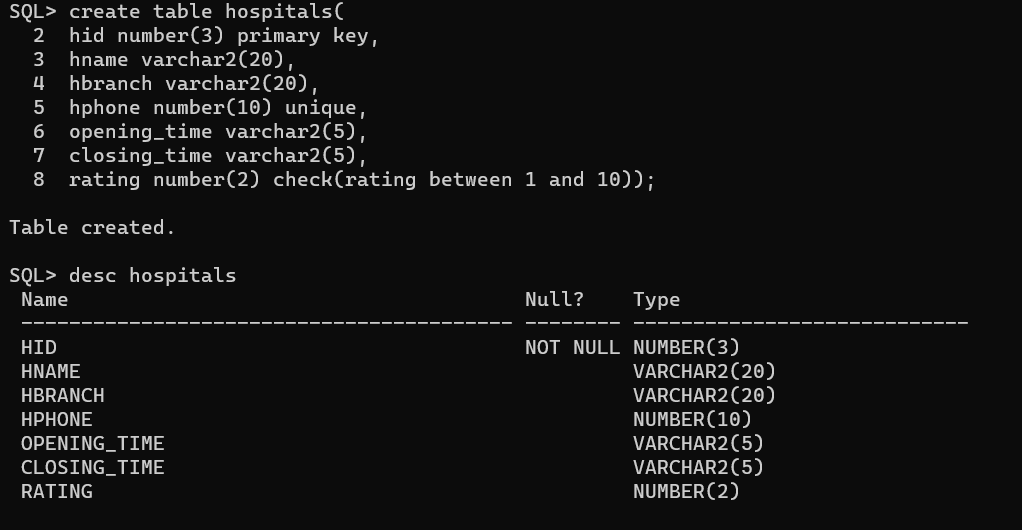
hbranch varchar2(20),

hphone number(10) unique,

opening\_time varchar2(5),

closing\_time varchar2(5),

rating number(2) check(rating between 1 and 10));



2.Creating Table for Specialization:

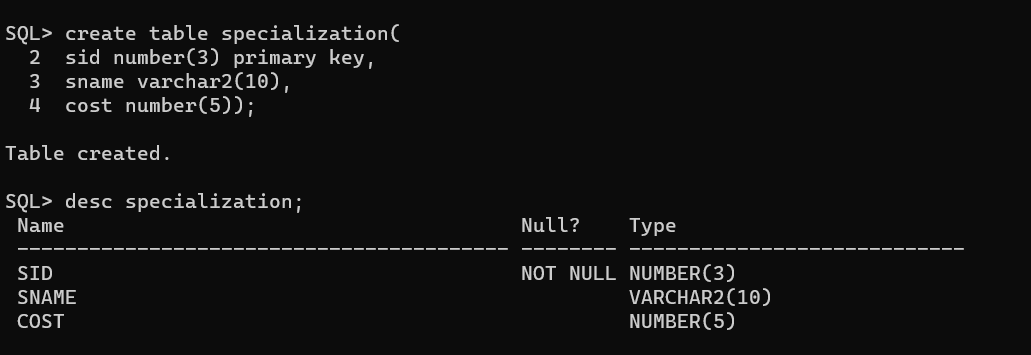
**QUERY:**

create table specialization(

sid number(3) primary key,

sname varchar2(10),

cost number(5));



3.Creating Table for Doctors:

**QUERY:**

create table doctors(

did number(5) primary key,

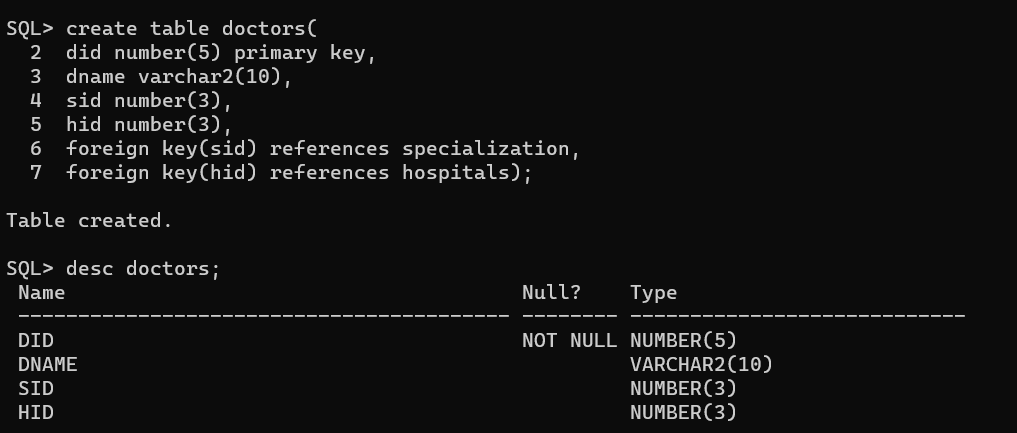
dname varchar2(10),

sid number(3),

hid number(3),

foreign key(sid) references specialization,

foreign key(hid) references hospitals);



4.Creating Table for Users:

**QUERY:**

create table users(

usid number(5) primary key,

fname varchar2(10),

lname varchar2(10),

blood\_group varchar2(5),

gender varchar2(10),

phone number(10) unique);



5.Creating Table for Inpatients:

**QUERY:**

create table inpatients(

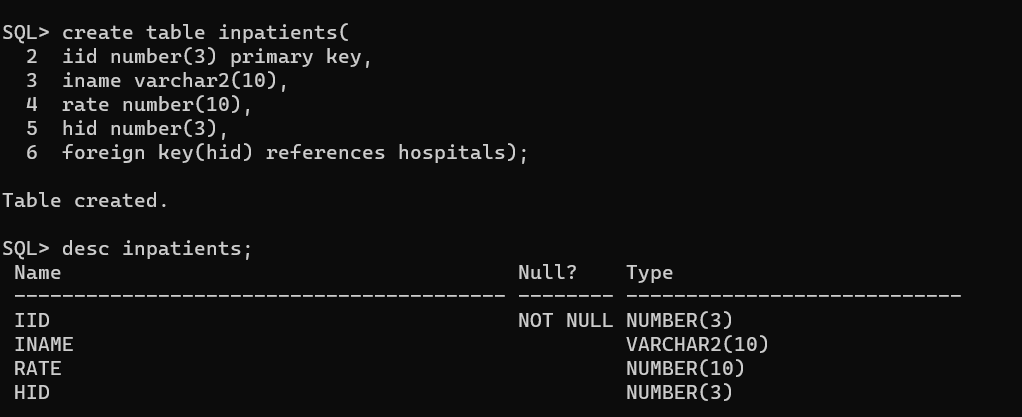
iid number(3) primary key,

iname varchar2(10),

rate number(10),

hid number(3),

foreign key(hid) references hospitals);



6.Creating Table for Outpatients:

**QUERY:**

create table outpatients(

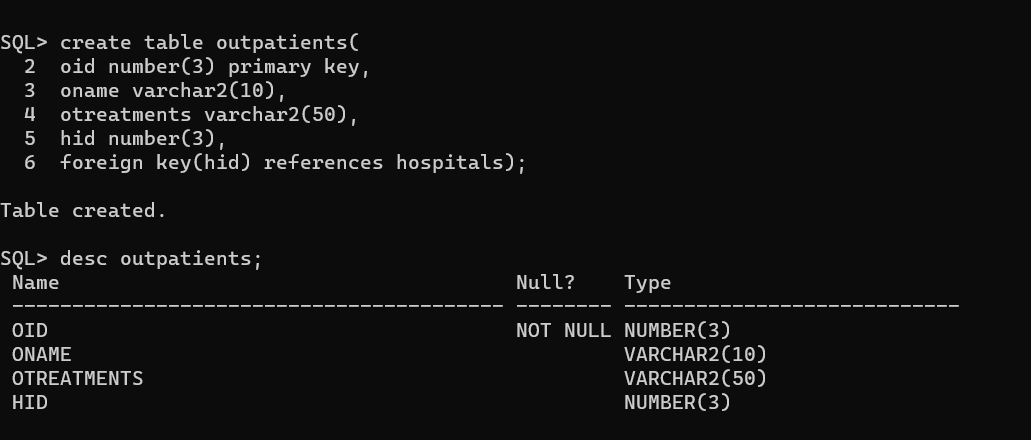
oid number(3) primary key,

oname varchar2(10),

otreatments varchar2(50),

hid number(3),

foreign key(hid) references hospitals);



7.Creating Table for Visits:

**QUERY:**

create table visits(

vid number(3) primary key,

usid number(5),

hid number(3),

did number(5),

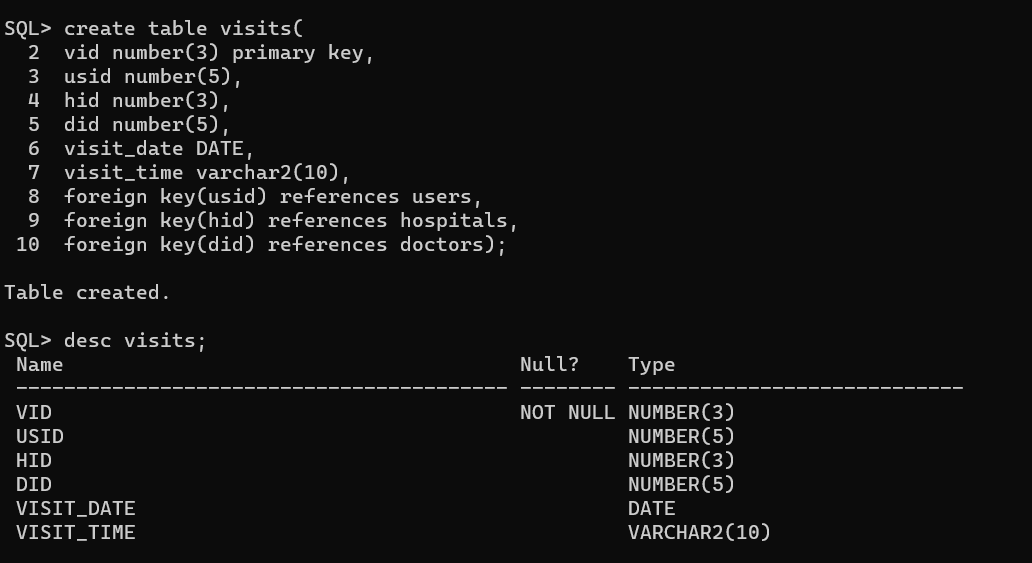
visit\_date DATE,

visit\_time varchar2(10),

foreign key(usid) references users,

foreign key(hid) references hospitals,

foreign key(did) references doctors);

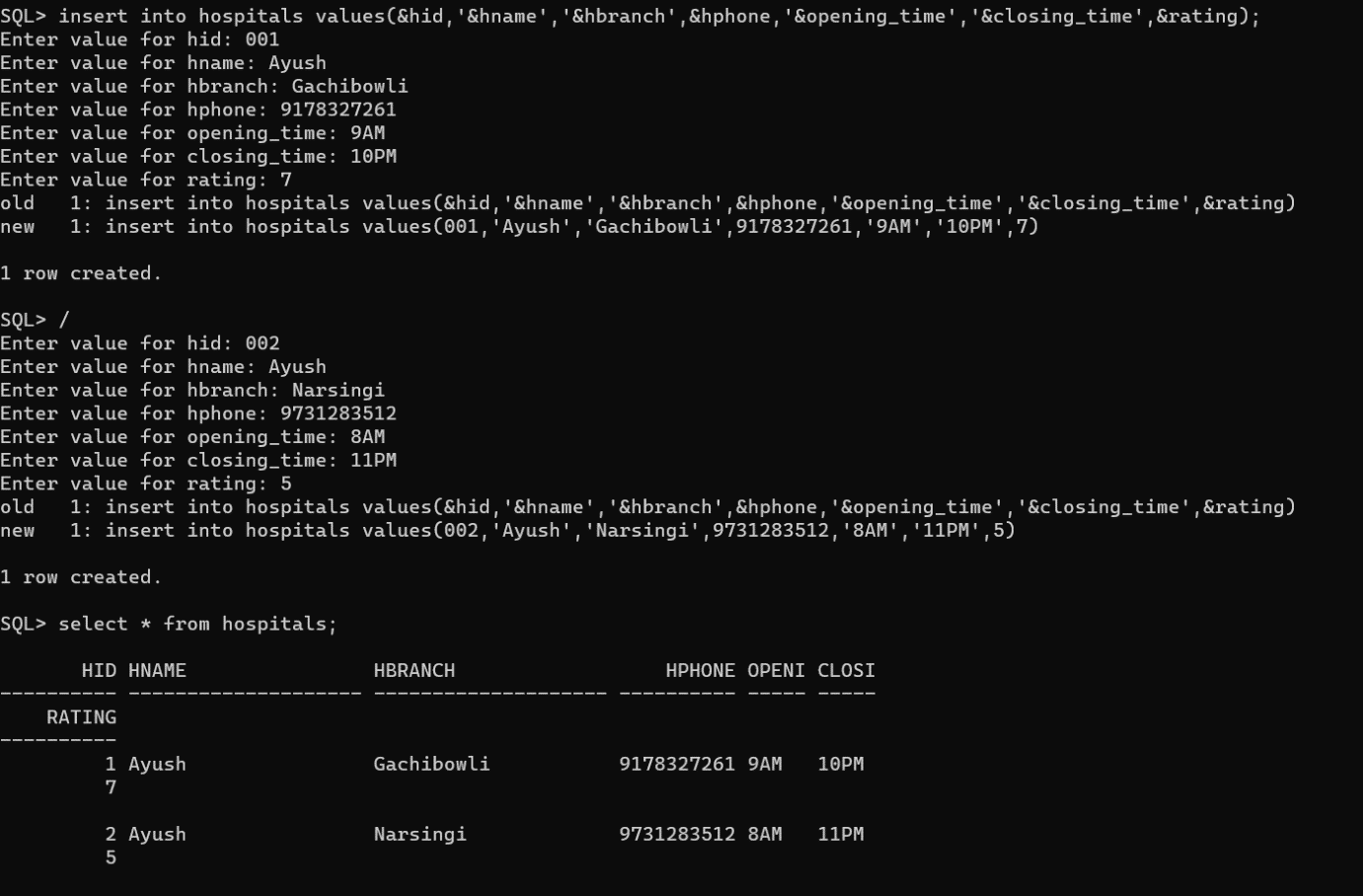


**DML COMMANDS:**

1.Insert values into Hospitals:

**QUERY:**

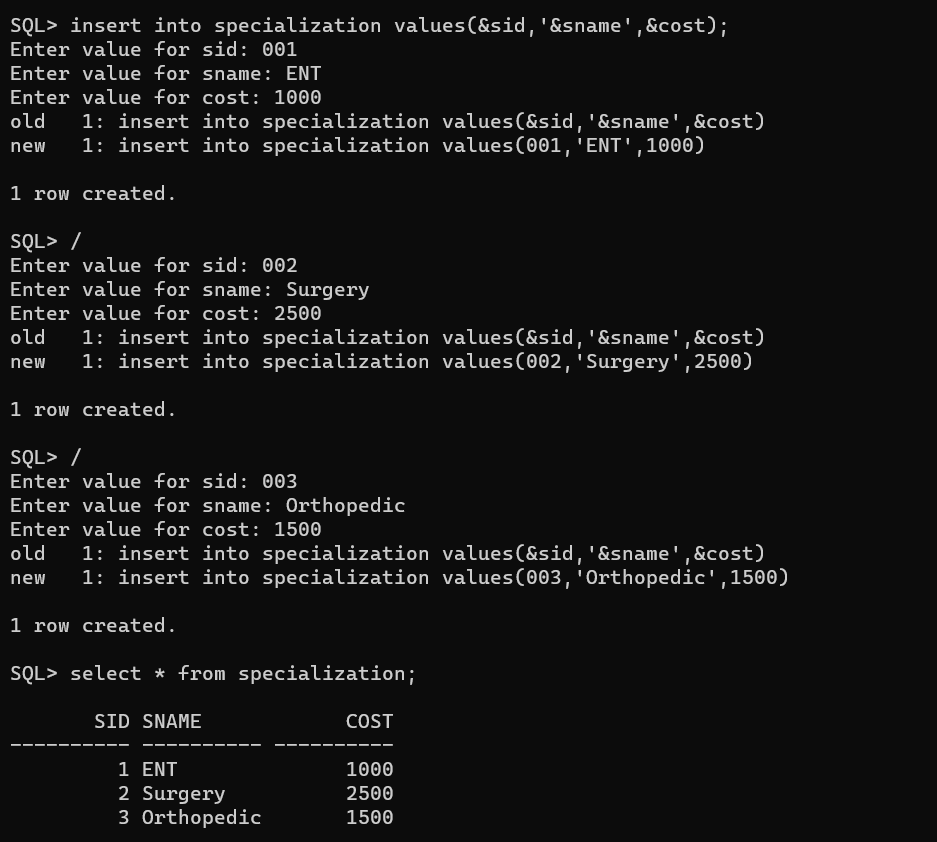
insert into hospitals values(&hid,'&hname','&hbranch',&hphone,'&opening\_time','&closing\_time',&rating);



2.Insert values into specializations:

**QUERY:**

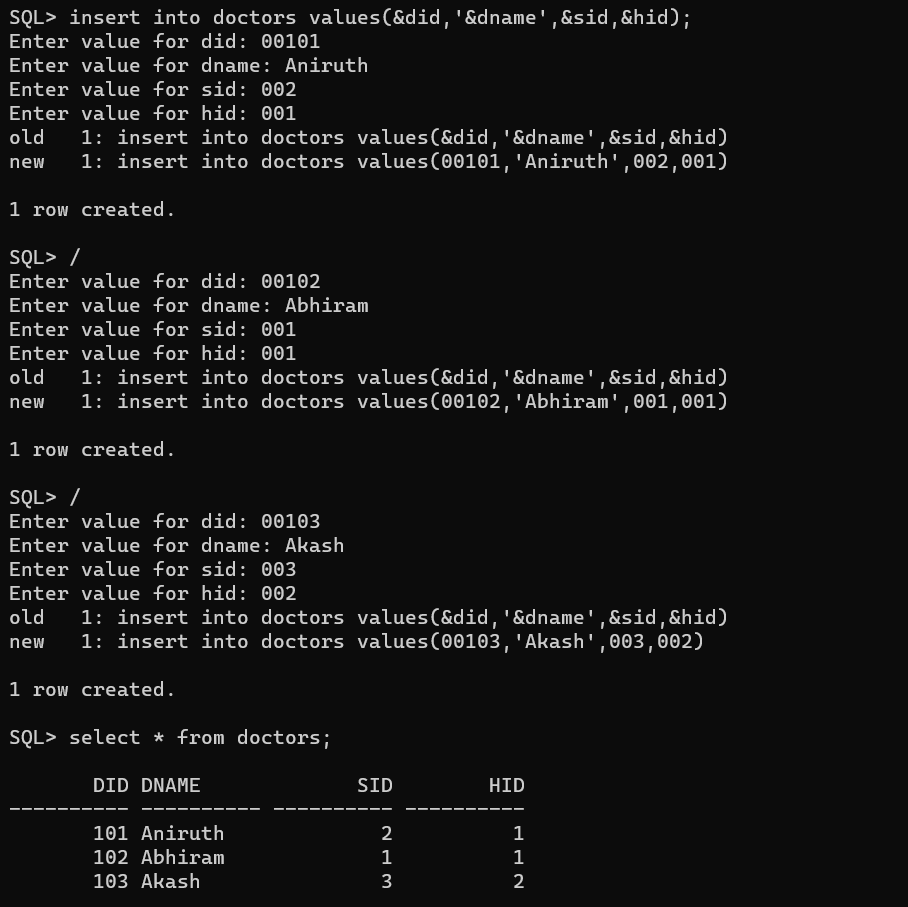
insert into specialization values(&sid,'&sname',&cost);



3.Inserting values into Doctors:

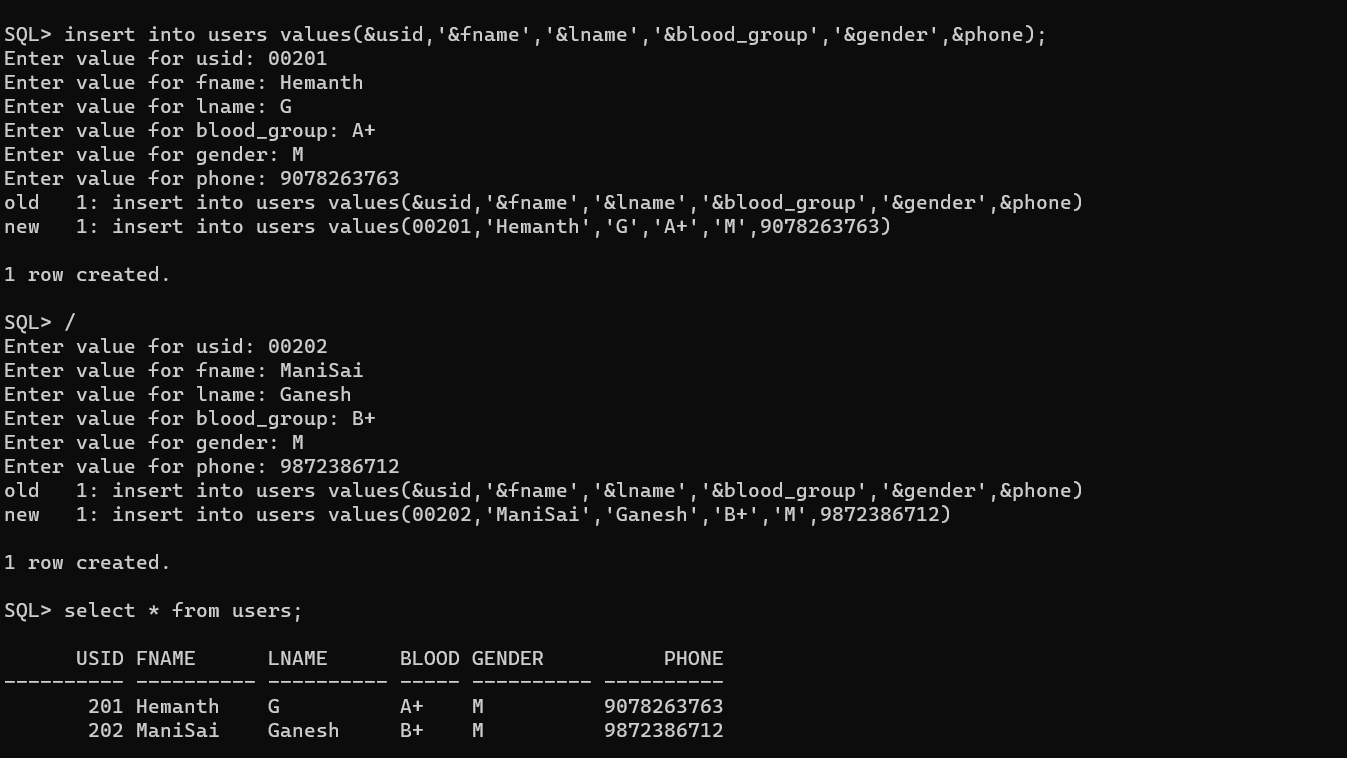
**QUERY:**

insert into doctors values(&did,'&dname',&sid,&hid);

****

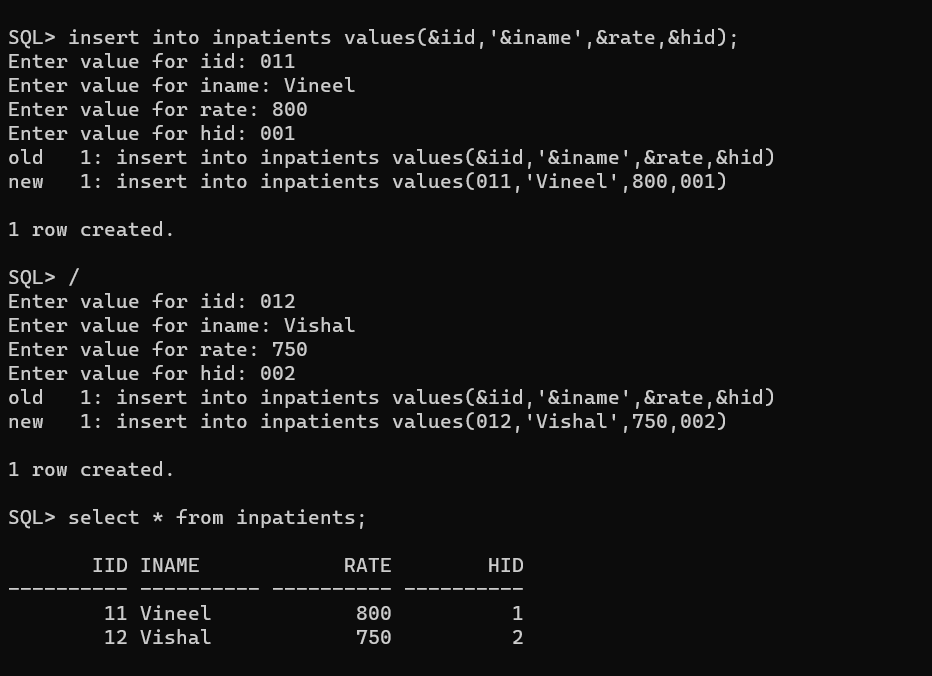
4.Insert values into users:

**QUERY:**insert into users values(&usid,'&fname','&lname','&blood\_group','&gender',&phone);



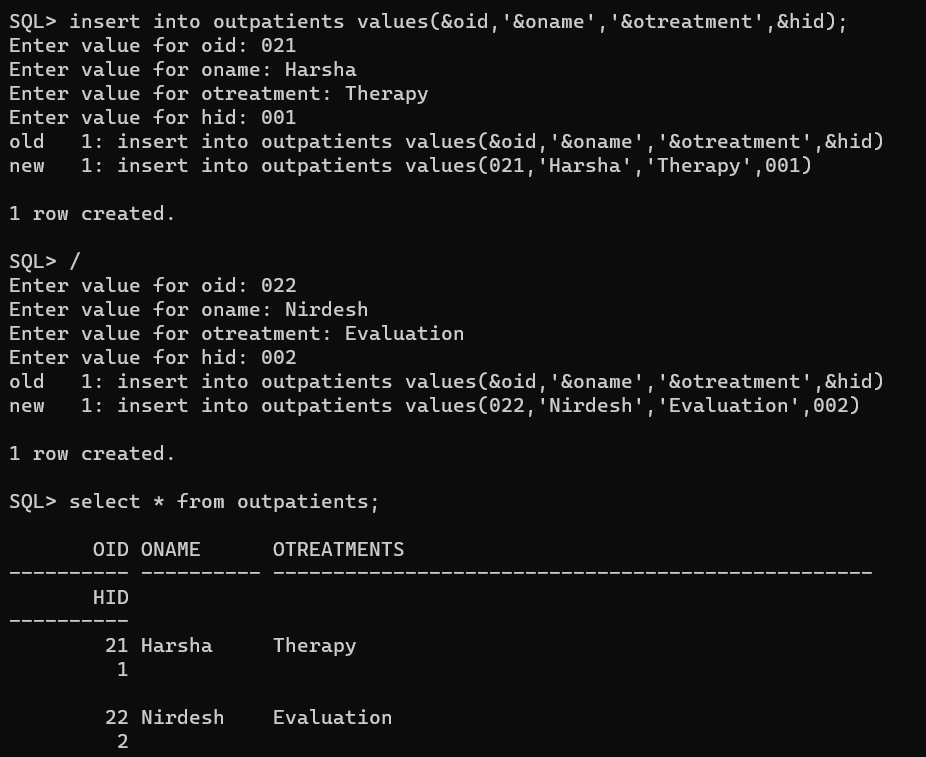
5.Inserting values into inpatients:

**QUERY:** insert into inpatients values(&iid,'&iname',&rate,&hid);

****

6.Inserting values into Outpatients:

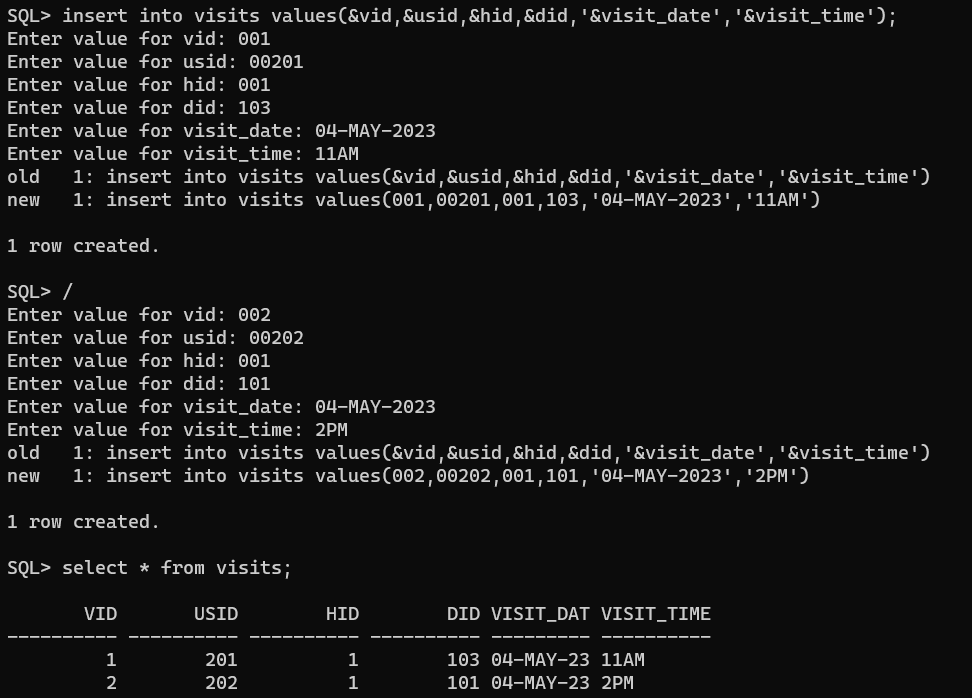
**QUERY:** insert into outpatients values(&oid,'&oname','&otreatment',&hid);



7.Inserting values into Visits Table:

**QUERY:**

insert into visits values(&vid,&usid,&hid,&did,'&visit\_date','&visit\_time');



IMPLEMENTATION

JAVA-SQL CONNECTIVITY USING JDBC:

Java Database Connectivity (JDBC) is an application programming interface

(API) for the programming language Java, which defines how a client may

access a database. It is a Java-based data access technology used for Java

database connectivity. It is part of the Java Standard Edition platform, from

Oracle Corporation. It provides methods to query and update data in a database

and is oriented towards relational databases.

APPLICATION CODE:

import javax.swing.\*;

import java.util.List;

import java.util.concurrent.ExecutionException;

import java.util.ArrayList;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.table.DefaultTableModel;

public class AyushHospitalFinderForm extends JFrame {

private JTextField hospitalIdTextField;

private JTextField hospitalNameTextField;

private JTextField hospitalCityTextField;

private JTextField hospitalContactTextField;

private JTextField hospitalOpenTimeTextField;

private JTextField hospitalCloseTimeTextField;

private JTextField ratingTextField;

private JPanel mainPanel,mainPanel1,mainPanel2;

private JLabel imageLabel;

private JPanel insertPanel,updatePanel, patientPanel, hospitalPanel, deletePanel, appointmentPanel,doctorPanel;

private JTable patientTable;

public AyushHospitalFinderForm() {

setTitle("Ayush Hospital Finder");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(600, 600);

setLocationRelativeTo(null);

imageLabel = new JLabel();

ImageIcon imageIcon = new ImageIcon("D:\\DBMS\\unnamed.png\\");

imageLabel.setIcon(imageIcon);

mainPanel = new JPanel();

setContentPane(mainPanel);

mainPanel.setLayout(new BorderLayout());

JButton adminButton = new JButton("Admin");

adminButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

adminMenu();

}

});

JButton patientButton = new JButton("Patient");

patientButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

patientMenu();

}

});

JPanel buttonPanel = new JPanel();

buttonPanel.add(adminButton);

buttonPanel.add(patientButton);

mainPanel.add(buttonPanel, BorderLayout.PAGE\_START);

mainPanel.add(imageLabel, BorderLayout.CENTER);

pack();

setVisible(true);

}

private void adminMenu() {

JTextField usernameField = new JTextField();

JPasswordField passwordField = new JPasswordField();

Object[] message = {

"Username:", usernameField,

"Password:", passwordField

};

int option = JOptionPane.showConfirmDialog(mainPanel, message, "Admin Login", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

String username = usernameField.getText();

char[] password = passwordField.getPassword();

boolean isAuthenticated = authenticateAdmin(username, password);

if (isAuthenticated) {

setTitle("Ayush Hopsital Finder");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(600, 600);

setLocationRelativeTo(null);

imageLabel = new JLabel();

ImageIcon imageIcon = new ImageIcon("D:\\DBMS\\unnamed.png\\");

imageLabel.setIcon(imageIcon);

mainPanel1 = new JPanel();

setContentPane(mainPanel1);

mainPanel1.setLayout(new BorderLayout());

JMenuBar menuBar1 = new JMenuBar();

setJMenuBar(menuBar1);

JMenu optionsMenu = new JMenu("Admin Options");

menuBar1.add(optionsMenu);

JMenuItem hospitalMenuItem = new JMenuItem("EditHospital");

optionsMenu.add(hospitalMenuItem);

hospitalMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

HospitalForm();

}

});

JMenuItem doctorMenuItem = new JMenuItem("EditDoctor");

optionsMenu.add(doctorMenuItem);

doctorMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

DoctorForm();

}

});

JMenuItem viewDoctor =new JMenuItem("View Doctors");

optionsMenu.add(viewDoctor);

viewDoctor.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

DoctorList();

}

});

JMenuItem viewHospital =new JMenuItem("View Hospitals");

optionsMenu.add(viewHospital);

viewHospital.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

HospitalList();

}

});

JMenuItem backMenuItem = new JMenuItem("Back");

optionsMenu.add(backMenuItem);

backMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new AyushHospitalFinderForm().setVisible(true);

setVisible(false);}

});

mainPanel1.add(imageLabel, BorderLayout.CENTER);

pack();

setVisible(true);

}

} else {

JOptionPane.showMessageDialog(mainPanel, "Invalid username or password. Please try again.", "Authentication Failed", JOptionPane.ERROR\_MESSAGE);

}

}

private boolean authenticateAdmin(String username, char[] password) {

String storedUsername = "admin";

String storedPassword = "admin";

return username.equals(storedUsername) && new String(password).equals(storedPassword);

}

private void patientMenu() {

setTitle("Ayush Hospital Finder");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

setSize(600, 600);

setLocationRelativeTo(null);

imageLabel = new JLabel();

ImageIcon imageIcon = new ImageIcon("D:\\DBMS\\unnamed.png\\");

imageLabel.setIcon(imageIcon);

mainPanel2 = new JPanel();

setContentPane(mainPanel2);

mainPanel2.setLayout(new BorderLayout());

JMenuBar menuBar = new JMenuBar();

setJMenuBar(menuBar);

JMenu optionsMenu = new JMenu("Explore");

menuBar.add(optionsMenu);

JMenuItem patientFMenuItem = new JMenuItem("Patient Form");

optionsMenu.add(patientFMenuItem);

patientFMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

patientForm();

}

});

JMenuItem appointMenuItem = new JMenuItem("Appointment");

optionsMenu.add(appointMenuItem);

appointMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

appointmentForm();

}

});

JMenuItem patientappointmentItem = new JMenuItem("My Appointments");

optionsMenu.add(patientappointmentItem);

patientappointmentItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

viewappointment();

}

});

JMenuItem backMenuItem = new JMenuItem("Back");

optionsMenu.add(backMenuItem);

backMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new AyushHospitalFinderForm().setVisible(true);

setVisible(false);

}

});

mainPanel2.add(imageLabel, BorderLayout.CENTER);

pack();

setVisible(true);

// Add patient-specific actions here

}

private JTextField patientIdTextField;

private JTextField patientFNameTextField;

private JTextField patientLNameTextField;

private JTextField patientBGTextField;

private JTextField patientGenderTextField;

private JTextField patientPhoneTextField;

private void patientForm() {

JLabel patientIdLabel = new JLabel("Patient ID:");

patientIdTextField = new JTextField(10);

JLabel patientFNameLabel = new JLabel("FirstName:");

patientFNameTextField = new JTextField(20);

JLabel patientLNameLabel = new JLabel("LastName:");

patientLNameTextField = new JTextField(20);

JLabel patientBGLabel = new JLabel("BloodGroup:");

patientBGTextField = new JTextField(15);

JLabel patientGenderLabel = new JLabel("Gender:");

patientGenderTextField = new JTextField(15);

JLabel patientPhoneLabel = new JLabel("PhoneNo:");

patientPhoneTextField = new JTextField(15);

// Create buttons

JButton submitButton = new JButton("Submit");

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

insertPatient();

}

});

JButton modifyButton = new JButton("Modify");

modifyButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updatePatient();

}

});

JButton deleteButton = new JButton("Delete");

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deletePatient();

}

});

JButton viewButton = new JButton("View");

viewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

viewPatientDetails();

}

});

// Create panel for form components

patientPanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(6, 6, 6, 6);

patientPanel.add(patientIdLabel, gbc);

gbc.gridx++;

patientPanel.add(patientIdTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

patientPanel.add(patientFNameLabel, gbc);

gbc.gridx++;

patientPanel.add(patientFNameTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 1;

patientPanel.add(patientLNameLabel, gbc);

gbc.gridx++;

patientPanel.add(patientLNameTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

patientPanel.add(patientBGLabel, gbc);

gbc.gridx++;

patientPanel.add(patientBGTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

patientPanel.add(patientGenderLabel, gbc);

gbc.gridx++;

patientPanel.add(patientGenderTextField, gbc);

// ...

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 1;

patientPanel.add(patientPhoneLabel, gbc);

gbc.gridx++;

patientPanel.add(patientPhoneTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 1; // Empty label for spacing

gbc.gridx++;

patientPanel.add(submitButton, gbc);

gbc.gridx++;

gbc.insets = new Insets(0, 10, 0, 5); // Adjusted insets for the Modify button

patientPanel.add(modifyButton, gbc);

gbc.gridx++;

gbc.insets = new Insets(0, 5, 0, 5); // Adjusted insets for the Delete button

patientPanel.add(deleteButton, gbc);

gbc.gridx++;

gbc.insets = new Insets(0, 5, 0, 0); // Adjusted insets for the View button

patientPanel.add(viewButton, gbc);

add(patientPanel, BorderLayout.CENTER);

setSize(600, 600);

setContentPane(patientPanel);

setLocationRelativeTo(null);

}

private DefaultTableModel tableModel;

private void insertPatient()

{

tableModel = new DefaultTableModel();

patientTable = new JTable(tableModel);

int patientId = Integer.parseInt(patientIdTextField.getText());

String patientFName = patientFNameTextField.getText();

String patientLName = patientLNameTextField.getText();

String patientBG = patientBGTextField.getText();

String patientGender = patientGenderTextField.getText();

long patientPhone = Long.parseLong(patientPhoneTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String insertQuery = "INSERT INTO users (USID, FName, LName, Blood\_Group,Gender,Phone) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

preparedStatement.setInt(1, patientId);

preparedStatement.setString(2, patientFName);

preparedStatement.setString(3, patientLName);

preparedStatement.setString(4, patientBG);

preparedStatement.setString(5, patientGender);

preparedStatement.setLong(6, patientPhone);

int rowsInserted = preparedStatement.executeUpdate();

if (rowsInserted > 0) {

JOptionPane.showMessageDialog(this, "Patient inserted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearPFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to insert Patient", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updatePatient() {

int patientId = Integer.parseInt(patientIdTextField.getText());

String patientFName = patientFNameTextField.getText();

String patientLName = patientLNameTextField.getText();

String patientBG = patientBGTextField.getText();

String patientGender = patientGenderTextField.getText();

String phoneText = patientPhoneTextField.getText();

long patientPhone = 0;

if (!phoneText.isEmpty()) {

try {

patientPhone = Long.parseLong(phoneText);

} catch (NumberFormatException ex) {

ex.printStackTrace();

}

}

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

StringBuilder updateQuery = new StringBuilder("UPDATE users SET ");

List<Object> params = new ArrayList<>();

if (!patientFName.isEmpty()) {

updateQuery.append("FName = ?, ");

params.add(patientFName);

}

if (!patientLName.isEmpty()) {

updateQuery.append("LName = ?, ");

params.add(patientLName);

}

if (!patientBG.isEmpty()) {

updateQuery.append("Blood\_Group = ?, ");

params.add(patientBG);

}

if (!patientGender.isEmpty()) {

updateQuery.append("Gender = ?, ");

params.add(patientGender);

}

if (patientPhone != 0) {

updateQuery.append("Phone = ?, ");

params.add(patientPhone);

}

// Remove the trailing comma and space

updateQuery.setLength(updateQuery.length() - 2);

updateQuery.append(" WHERE USID = ?");

params.add(patientId);

PreparedStatement preparedStatement = connection.prepareStatement(updateQuery.toString());

for (int i = 0; i < params.size(); i++) {

preparedStatement.setObject(i + 1, params.get(i));

}

int rowsUpdated = preparedStatement.executeUpdate();

if (rowsUpdated > 0) {

JOptionPane.showMessageDialog(this, "Patient updated successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearPFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to update Patient", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deletePatient()

{

int patientId = Integer.parseInt(patientIdTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String deleteQuery = "DELETE FROM users WHERE USID = ?";

PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);

preparedStatement.setInt(1, patientId);

int rowsDeleted = preparedStatement.executeUpdate();

if (rowsDeleted > 0) {

JOptionPane.showMessageDialog(this, "Patient deleted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearPFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to delete patient", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void viewPatientDetails() {

int patientId = Integer.parseInt(patientIdTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String selectQuery = "SELECT \* FROM users WHERE usid = ?";

PreparedStatement statement = connection.prepareStatement(selectQuery);

statement.setInt(1, patientId);

ResultSet resultSet = statement.executeQuery();

// Create the table model

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Patient ID");

tableModel.addColumn("First Name");

tableModel.addColumn("Last Name");

tableModel.addColumn("Blood Group");

tableModel.addColumn("Gender");

tableModel.addColumn("Phone");

// Populate the table model with data

while (resultSet.next()) {

int id = resultSet.getInt("usid");

String firstName = resultSet.getString("FName");

String lastName = resultSet.getString("LName");

String bloodGroup = resultSet.getString("blood\_Group");

String gender = resultSet.getString("gender");

String phoneNo = resultSet.getString("phone");

Object[] rowData = {id, firstName, lastName, bloodGroup, gender, phoneNo};

tableModel.addRow(rowData);

}

// Create the table with the table model

patientTable = new JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = new JScrollPane(patientTable);

// Create a panel and add the scroll pane to it

JPanel panel = new JPanel();

panel.add(scrollPane);

// Create a frame and add the panel to it

JFrame frame = new JFrame("Patient Details");

frame.getContentPane().add(panel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void clearPFormFields()

{

patientIdTextField.setText("");

patientFNameTextField.setText("");

patientLNameTextField.setText("");

patientBGTextField.setText("");

patientGenderTextField.setText("");

patientPhoneTextField.setText("");

}

private JTextField visitIdTextField;

private JTextField usidTextField;

private JTextField vhidTextField;

private JTextField vdidTextField;

private JTextField visitdateTextField;

private JTextField visittimeTextField;

private void appointmentForm() {

setSize(600, 600);

JLabel visitIdLabel = new JLabel("Visit ID:");

visitIdTextField = new JTextField(10);

JLabel usidLabel = new JLabel("Patient ID:");

usidTextField = new JTextField(20);

JLabel vhidLabel = new JLabel("Hospital ID:");

vhidTextField = new JTextField(20);

JButton locateButton = new JButton("Locate");

locateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

viewHospitals();

}

});

JLabel vdidLabel = new JLabel("Doctor ID:");

vdidTextField = new JTextField(15);

JButton viewButton = new JButton("View");

viewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

viewdoctors();

}

});

JLabel visitdateLabel = new JLabel("Visit Date:");

visitdateTextField = new JTextField(15);

JLabel visittimeLabel = new JLabel("Visit Time:");

visittimeTextField = new JTextField(15);

// Create buttons

JButton submitButton = new JButton("Submit");

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

bookappointment();

}

});

JButton modifyButton = new JButton("Modify");

modifyButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateappointment();

}

});

JButton deleteButton = new JButton("Delete");

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteappointment();

}

});

// Create panel for form components

appointmentPanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(8, 8, 8, 8);

appointmentPanel.add(visitIdLabel, gbc);

gbc.gridx++;

appointmentPanel.add(visitIdTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

appointmentPanel.add(usidLabel, gbc);

gbc.gridx++;

appointmentPanel.add(usidTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 1;

appointmentPanel.add(vhidLabel, gbc);

gbc.gridx++;

appointmentPanel.add(vhidTextField, gbc);

gbc.gridy++;

gbc.gridwidth = 2;

appointmentPanel.add(locateButton, gbc);

gbc.gridx = 0;

gbc.gridy++;

appointmentPanel.add(vdidLabel, gbc);

gbc.gridx++;

appointmentPanel.add(vdidTextField, gbc);

gbc.gridy++;

gbc.gridwidth = 2;

appointmentPanel.add(viewButton, gbc);

gbc.gridx = 0;

gbc.gridy++;

appointmentPanel.add(visitdateLabel, gbc);

gbc.gridx++;

appointmentPanel.add(visitdateTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

appointmentPanel.add(visittimeLabel, gbc);

gbc.gridx++;

appointmentPanel.add(visittimeTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 1;

appointmentPanel.add(submitButton, gbc);

gbc.gridx++;

gbc.insets = new Insets(8, 8, 8, 8); // Adjusted insets for the Modify button

appointmentPanel.add(modifyButton, gbc);

gbc.gridx++;

appointmentPanel.add(deleteButton, gbc);

//add(appointmentPanel, BorderLayout.CENTER);

setContentPane(appointmentPanel);

setLocationRelativeTo(null);

}

private void viewHospitals()

{

JTextField locationField = new JTextField();

Object[] message = {

"", locationField};

int option = JOptionPane.showConfirmDialog(mainPanel, message, "Locator", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION)

{

String location=locationField.getText();

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String selectQuery = "SELECT HID,HBranch,Opening\_Time,Closing\_Time,Rating FROM hospitals where HBranch= ?";

PreparedStatement statement = connection.prepareStatement(selectQuery);

statement.setString(1,location);

ResultSet resultSet = statement.executeQuery();

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Hospital ID");

tableModel.addColumn("Hospital Branch");

tableModel.addColumn("Opening\_Time");

tableModel.addColumn("Closing\_Time");

tableModel.addColumn("Rating");

// Populate the table model with data

while (resultSet.next()) {

int hid = resultSet.getInt("hid");

String hbranch = resultSet.getString("hbranch");

String opentime = resultSet.getString("opening\_time");

String closetime= resultSet.getString("closing\_time");

int rating = resultSet.getInt("rating");

Object[] rowData = {hid,hbranch,opentime,closetime,rating};

tableModel.addRow(rowData);

}

// Create the table with the table model

JTable table = new JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = new JScrollPane(table);

// Create a panel and add the scroll pane to it

JPanel panel = new JPanel();

panel.add(scrollPane);

// Create a frame and add the panel to it

JFrame frame = new JFrame("Nearby Ayush Hospitals");

// frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

frame.getContentPane().add(panel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

catch (SQLException ex) {

ex.printStackTrace();

System.out.println("Failed to connect to the database");

}

}

}

private void viewdoctors()

{

JTextField viewField = new JTextField();

Object[] message = {

"", viewField};

int option = JOptionPane.showConfirmDialog(mainPanel, message, "DocInfo", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION)

{

String view=viewField.getText();

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String selectQuery = "SELECT doctors.did,doctors.dname,doctors.hid,doctors.sid,specialization.sname FROM doctors inner join specialization ON doctors.sid=specialization.sid where specialization.sname= ?" ;

PreparedStatement statement = connection.prepareStatement(selectQuery);

statement.setString(1,view);

ResultSet resultSet = statement.executeQuery();

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Doctor ID");

tableModel.addColumn("Doctor Name");

tableModel.addColumn("Hospital ID");

tableModel.addColumn("Specialization");

// Populate the table model with data

while (resultSet.next()) {

int did = resultSet.getInt("did");

String dname = resultSet.getString("dname");

int hid = resultSet.getInt("hid");

String sname= resultSet.getString("sname");

Object[] rowData = {did,dname,hid,sname};

tableModel.addRow(rowData);

}

// Create the table with the table model

JTable table = new JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = new JScrollPane(table);

// Create a panel and add the scroll pane to it

doctorPanel = new JPanel();

doctorPanel.add(scrollPane);

// Create a frame and add the panel to it

JFrame frame = new JFrame("Doctors Info");

//frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

frame.getContentPane().add(doctorPanel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

catch (SQLException ex) {

ex.printStackTrace();

System.out.println("Failed to connect to the database");

}

}

}

private void bookappointment() {

int visitId = Integer.parseInt(visitIdTextField.getText());

int userId = Integer.parseInt(usidTextField.getText());

int doctorId = Integer.parseInt(vdidTextField.getText());

int hospitalId = Integer.parseInt(vhidTextField.getText());

String visitDate = visitdateTextField.getText();

String visitTime = visittimeTextField.getText();

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String insertQuery = "INSERT INTO visits (vid, usid, did, hid, visit\_date, visit\_time) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

preparedStatement.setInt(1, visitId);

preparedStatement.setInt(2, userId);

preparedStatement.setInt(3, doctorId);

preparedStatement.setInt(4, hospitalId);

preparedStatement.setString(5, visitDate);

preparedStatement.setString(6, visitTime);

int rowsInserted = preparedStatement.executeUpdate();

if (rowsInserted > 0) {

JOptionPane.showMessageDialog(this, "Appointment booked successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearAFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to book appointment", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateappointment() {

int visitid = Integer.parseInt(visitIdTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

StringBuilder updateQuery = new StringBuilder("UPDATE visits SET ");

List<Object> params = new ArrayList<>();

if (usidTextField.getText().trim().length() > 0) {

int usid = Integer.parseInt(usidTextField.getText());

updateQuery.append("USID = ?, ");

params.add(usid);

}

if (vhidTextField.getText().trim().length() > 0) {

int vhid = Integer.parseInt(vhidTextField.getText());

updateQuery.append("HID = ?, ");

params.add(vhid);

}

if (vdidTextField.getText().trim().length() > 0) {

int vdid = Integer.parseInt(vdidTextField.getText());

updateQuery.append("DID = ?, ");

params.add(vdid);

}

if (visitdateTextField.getText().trim().length() > 0) {

String visitdate = visitdateTextField.getText();

updateQuery.append("VISIT\_DATE = ?, ");

params.add(visitdate);

}

if (visittimeTextField.getText().trim().length() > 0) {

String visittime = visittimeTextField.getText();

updateQuery.append("VISIT\_TIME = ?, ");

params.add(visittime);

}

// Remove the trailing comma and space from the query

updateQuery.delete(updateQuery.length() - 2, updateQuery.length());

updateQuery.append(" WHERE VID = ?");

params.add(visitid);

PreparedStatement preparedStatement = connection.prepareStatement(updateQuery.toString());

// Set the parameter values

for (int i = 0; i < params.size(); i++) {

Object paramValue = params.get(i);

preparedStatement.setObject(i + 1, paramValue);

}

int rowsUpdated = preparedStatement.executeUpdate();

if (rowsUpdated > 0) {

JOptionPane.showMessageDialog(this, "Appointment updated successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearAFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to update appointment", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

} catch (NumberFormatException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Invalid input format", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteappointment()

{

int visitId = Integer.parseInt(visitIdTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String deleteQuery = "DELETE FROM visits WHERE VID = ?";

PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);

preparedStatement.setInt(1, visitId);

int rowsDeleted = preparedStatement.executeUpdate();

if (rowsDeleted > 0) {

JOptionPane.showMessageDialog(this, "Patient deleted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearAFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to delete patient", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void clearAFormFields()

{

visitIdTextField.setText("");

usidTextField.setText("");

vhidTextField.setText("");

vdidTextField.setText("");

visitdateTextField.setText("");

visittimeTextField.setText("");

}

private void viewappointment()

{

JTextField viewField = new JTextField();

Object[] message = {

"", viewField};

int option = JOptionPane.showConfirmDialog(mainPanel, message, "AppointmentInfo", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION)

{

int no=Integer.parseInt(viewField.getText()); try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String selectQuery = "SELECT \* FROM visits where usid=?";

PreparedStatement statement = connection.prepareStatement(selectQuery);

statement.setInt(1,no);

ResultSet resultSet = statement.executeQuery();

// Create the table model with column names

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Visit ID");

tableModel.addColumn("Patient ID");

tableModel.addColumn("Hospital ID");

tableModel.addColumn("Doctor ID");

tableModel.addColumn("Visit Date");

tableModel.addColumn("Visit Time");

// Populate the table model with data

while (resultSet.next()) {

int vid = resultSet.getInt("vid");

int usid = resultSet.getInt("usid");

int hid = resultSet.getInt("hid");

int did = resultSet.getInt("did");

String visitdate = resultSet.getString("visit\_date");

String visittime = resultSet.getString("visit\_time");

Object[] rowData = {vid,usid,hid,did,visitdate,visittime};

tableModel.addRow(rowData);

}

// Create the table with the table model

JTable table = new JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = new JScrollPane(table);

// Create a panel and add the scroll pane to it

insertPanel = new JPanel();

insertPanel.add(scrollPane);

// Create a frame and add the panel to it

JFrame frame = new JFrame("Appointment Data");

//frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

frame.getContentPane().add(insertPanel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

System.out.println("Failed to connect to the database");

}

}

else

{

JOptionPane.showMessageDialog(mainPanel, "Enter Patient ID.", "Invalid", JOptionPane.ERROR\_MESSAGE);

}

}

private void HospitalForm() {

// Create form components

JLabel hospitalIdLabel = new JLabel("Hospital ID:");

hospitalIdTextField = new JTextField(10);

JLabel hospitalNameLabel = new JLabel("Hospital Name:");

hospitalNameTextField = new JTextField(20);

JLabel hospitalCityLabel = new JLabel("Hospital City:");

hospitalCityTextField = new JTextField(20);

JLabel hospitalContactLabel = new JLabel("Hospital Contact:");

hospitalContactTextField = new JTextField(15);

JLabel hospitalOpeningTimeLabel = new JLabel("Hospital OpenTime:");

hospitalOpenTimeTextField = new JTextField(15);

JLabel hospitalClosingTimeLabel = new JLabel("Hospital CloseTime:");

hospitalCloseTimeTextField = new JTextField(15);

JLabel hospitalRatingLabel = new JLabel("Rating:");

ratingTextField = new JTextField(10);

// Create buttons

JButton submitButton = new JButton("Submit");

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

insertHospital();

}

});

JButton modifyButton = new JButton("Modify");

modifyButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateHospital();

}

});

JButton deleteButton = new JButton("Delete");

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteHospital();

}

});

/\*JButton viewButton = new JButton("View");

viewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

HospitalList();

}

});\*/

// Create panel for form components

hospitalPanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(7, 7, 7, 7);

hospitalPanel.add(hospitalIdLabel, gbc);

gbc.gridx++;

hospitalPanel.add(hospitalIdTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

hospitalPanel.add(hospitalNameLabel, gbc);

gbc.gridx++;

hospitalPanel.add(hospitalNameTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

hospitalPanel.add(hospitalCityLabel, gbc);

gbc.gridx++;

hospitalPanel.add(hospitalCityTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

hospitalPanel.add(hospitalContactLabel, gbc);

gbc.gridx++;

hospitalPanel.add(hospitalContactTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

hospitalPanel.add(hospitalOpeningTimeLabel, gbc);

gbc.gridx++;

hospitalPanel.add(hospitalOpenTimeTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

hospitalPanel.add(hospitalClosingTimeLabel, gbc);

gbc.gridx++;

hospitalPanel.add(hospitalCloseTimeTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

hospitalPanel.add(hospitalRatingLabel, gbc);

gbc.gridx++;

hospitalPanel.add(ratingTextField, gbc);

// Create panel for buttons

JPanel buttonPanel = new JPanel();

buttonPanel.add(submitButton);

buttonPanel.add(modifyButton);

buttonPanel.add(deleteButton);

/\*buttonPanel.add(viewButton);\*/

gbc.gridy++;

gbc.gridx = 0;

gbc.gridwidth = 2;

hospitalPanel.add(buttonPanel, gbc);

add(hospitalPanel, BorderLayout.CENTER);

setContentPane(hospitalPanel);

setLocationRelativeTo(null);

}

private void insertHospital() {

int hospitalId = Integer.parseInt(hospitalIdTextField.getText());

String hospitalName = hospitalNameTextField.getText();

String hospitalCity = hospitalCityTextField.getText();

long hospitalContact = Long.parseLong(hospitalContactTextField.getText());

String hospitalOpen = hospitalOpenTimeTextField.getText();

String hospitalClose = hospitalCloseTimeTextField.getText();

int rating = Integer.parseInt(ratingTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String insertQuery = "INSERT INTO Hospitals (HID, HName, HBranch, HPhone,Opening\_Time,Closing\_Time,Rating) VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

preparedStatement.setInt(1, hospitalId);

preparedStatement.setString(2, hospitalName);

preparedStatement.setString(3, hospitalCity);

preparedStatement.setLong(4, hospitalContact);

preparedStatement.setString(5, hospitalOpen);

preparedStatement.setString(6, hospitalClose);

preparedStatement.setInt(7, rating);

int rowsInserted = preparedStatement.executeUpdate();

if (rowsInserted > 0) {

JOptionPane.showMessageDialog(this, "Hospital inserted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to insert hospital", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateHospital() {

int hospitalId = Integer.parseInt(hospitalIdTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

StringBuilder updateQuery = new StringBuilder("UPDATE Hospitals SET ");

List<Object> params = new ArrayList<>();

if (hospitalNameTextField.getText().trim().length() > 0) {

String hospitalName = hospitalNameTextField.getText();

updateQuery.append("HName = ?, ");

params.add(hospitalName);

}

if (hospitalCityTextField.getText().trim().length() > 0) {

String hospitalCity = hospitalCityTextField.getText();

updateQuery.append("HBranch = ?, ");

params.add(hospitalCity);

}

if (hospitalContactTextField.getText().trim().length() > 0) {

long hospitalContact = Long.parseLong(hospitalContactTextField.getText());

updateQuery.append("HPhone = ?, ");

params.add(hospitalContact);

}

if (hospitalOpenTimeTextField.getText().trim().length() > 0) {

String hospitalOpen = hospitalOpenTimeTextField.getText();

updateQuery.append("Opening\_Time = ?, ");

params.add(hospitalOpen);

}

if (hospitalCloseTimeTextField.getText().trim().length() > 0) {

String hospitalClose = hospitalCloseTimeTextField.getText();

updateQuery.append("Closing\_Time = ?, ");

params.add(hospitalClose);

}

if (ratingTextField.getText().trim().length() > 0) {

int rating = Integer.parseInt(ratingTextField.getText());

updateQuery.append("Rating = ?, ");

params.add(rating);

}

// Remove the trailing comma and space from the query

updateQuery.delete(updateQuery.length() - 2, updateQuery.length());

updateQuery.append(" WHERE HID = ?");

params.add(hospitalId);

PreparedStatement preparedStatement = connection.prepareStatement(updateQuery.toString());

// Set the parameter values

for (int i = 0; i < params.size(); i++) {

Object paramValue = params.get(i);

preparedStatement.setObject(i + 1, paramValue);

}

int rowsUpdated = preparedStatement.executeUpdate();

if (rowsUpdated > 0) {

JOptionPane.showMessageDialog(this, "Hospital updated successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to update hospital", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

} catch (NumberFormatException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Invalid input format", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteHospital() {

int hospitalId = Integer.parseInt(hospitalIdTextField.getText());

SwingWorker<Integer, Void> worker = new SwingWorker<Integer, Void>() {

@Override

protected Integer doInBackground() throws Exception {

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

JOptionPane.showMessageDialog(AyushHospitalFinderForm.this, "Hospital deleted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

String deleteQuery = "DELETE FROM Hospitals WHERE HID = ?";

PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);

preparedStatement.setInt(1, hospitalId);

return preparedStatement.executeUpdate();

} catch (SQLException ex) {

ex.printStackTrace();

return -1;

}

}

@Override

protected void done() {

try {

int rowsDeleted = get();

if (rowsDeleted > 0) {

JOptionPane.showMessageDialog(AyushHospitalFinderForm.this, "Hospital deleted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearFormFields();

} else {

JOptionPane.showMessageDialog(AyushHospitalFinderForm.this, "Failed to delete hospital", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (InterruptedException | ExecutionException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(AyushHospitalFinderForm.this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

};

worker.execute();

}

private void clearFormFields() {

hospitalIdTextField.setText("");

hospitalNameTextField.setText("");

hospitalCityTextField.setText("");

hospitalContactTextField.setText("");

hospitalOpenTimeTextField.setText("");

hospitalCloseTimeTextField.setText("");

ratingTextField.setText("");

}

private JTextField didTextField;

private JTextField dnameTextField;

private JTextField sidTextField;

private JTextField hidTextField;

private void DoctorForm() {

setTitle("Doctor Form");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(600, 600);

setLocationRelativeTo(null);

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(10, 10, 10, 10);

JLabel didLabel = new JLabel("Doctor ID:");

didTextField = new JTextField(10);

JLabel dnameLabel = new JLabel("Doctor Name:");

dnameTextField = new JTextField(10);

JLabel sidLabel = new JLabel("Specialization ID:");

sidTextField = new JTextField(10);

JLabel hidLabel = new JLabel("Hospital ID:");

hidTextField = new JTextField(10);

mainPanel.add(didLabel, gbc);

gbc.gridx++;

mainPanel.add(didTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

mainPanel.add(dnameLabel, gbc);

gbc.gridx++;

mainPanel.add(dnameTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

mainPanel.add(sidLabel, gbc);

gbc.gridx++;

mainPanel.add(sidTextField, gbc);

gbc.gridx = 0;

gbc.gridy++;

mainPanel.add(hidLabel, gbc);

gbc.gridx++;

mainPanel.add(hidTextField, gbc);

gbc.gridy++;

gbc.gridx = 0;

gbc.gridwidth = 2;

mainPanel.add(new JLabel(), gbc); // Empty label for spacing

JButton submitButton = new JButton("Submit");

gbc.gridy++;

gbc.gridwidth = 1;

mainPanel.add(submitButton, gbc);

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

insertDoctor();

}

});

JButton modifyButton = new JButton("Modify");

gbc.gridx++;

mainPanel.add(modifyButton, gbc);

modifyButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateDoctor();

}

});

JButton deleteButton = new JButton("Delete");

gbc.gridx++;

mainPanel.add(deleteButton, gbc);

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteDoctor();

}

});

/\*JButton viewButton = new JButton("View");

gbc.gridx++;

mainPanel.add(viewButton, gbc);

viewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

DoctorList();

}

});\*/

setContentPane(mainPanel);

setVisible(true);

}

private void insertDoctor()

{

int doctorId = Integer.parseInt(didTextField.getText());

String doctorName = dnameTextField.getText();

int specializationId = Integer.parseInt(sidTextField.getText());

int hospId = Integer.parseInt(hidTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String insertQuery = "INSERT INTO doctors (did, dname, sid, hid) VALUES (?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

preparedStatement.setInt(1, doctorId);

preparedStatement.setString(2, doctorName);

preparedStatement.setInt(3, specializationId);

preparedStatement.setInt(4, hospId);

int rowsAffected = preparedStatement.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Doctor data inserted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearDFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to insert doctor data", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateDoctor() {

int doctorId = Integer.parseInt(didTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

StringBuilder updateQuery = new StringBuilder("UPDATE doctors SET ");

List<Object> params = new ArrayList<>();

if (dnameTextField.getText().trim().length() > 0) {

String doctorName = dnameTextField.getText();

updateQuery.append("dname = ?, ");

params.add(doctorName);

}

if (sidTextField.getText().trim().length() > 0) {

int specializationId = Integer.parseInt(sidTextField.getText());

updateQuery.append("sid = ?, ");

params.add(specializationId);

}

if (hidTextField.getText().trim().length() > 0) {

int hospId = Integer.parseInt(hidTextField.getText());

updateQuery.append("hid = ?, ");

params.add(hospId);

}

// Remove the trailing comma and space from the query

updateQuery.delete(updateQuery.length() - 2, updateQuery.length());

updateQuery.append(" WHERE did = ?");

params.add(doctorId);

PreparedStatement preparedStatement = connection.prepareStatement(updateQuery.toString());

// Set the parameter values

for (int i = 0; i < params.size(); i++) {

Object paramValue = params.get(i);

preparedStatement.setObject(i + 1, paramValue);

}

int rowsUpdated = preparedStatement.executeUpdate();

if (rowsUpdated > 0) {

JOptionPane.showMessageDialog(this, "Doctor data updated successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearDFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to update doctor data", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

} catch (NumberFormatException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Invalid input format", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteDoctor()

{

int doctorId = Integer.parseInt(didTextField.getText());

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String deleteQuery = "DELETE FROM doctors WHERE did = ?";

PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);

preparedStatement.setInt(1, doctorId);

int rowsDeleted = preparedStatement.executeUpdate();

if (rowsDeleted > 0) {

JOptionPane.showMessageDialog(this, "Doctor deleted successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

clearDFormFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to delete Doctor", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void clearDFormFields()

{

didTextField.setText("");;

dnameTextField.setText("");

sidTextField.setText("");

hidTextField.setText("");

}

private void HospitalList()

{

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String selectQuery = "SELECT \* FROM hospitals";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(selectQuery);

// Create the table model with column names

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Hospital ID");

tableModel.addColumn("Hospital Name");

tableModel.addColumn("Location");

tableModel.addColumn("Contact");

tableModel.addColumn("OpeningTime");

tableModel.addColumn("ClosingTime");

tableModel.addColumn("Rating");

// Populate the table model with data

while (resultSet.next()) {

int hid = resultSet.getInt("hid");

String hname = resultSet.getString("hname");

String location = resultSet.getString("hbranch");

long contact=resultSet.getLong("hphone");

String OpenTime = resultSet.getString("opening\_time");

String CloseTime = resultSet.getString("closing\_time");

int rating=resultSet.getInt("rating");

Object[] rowData = {hid, hname, location,contact,OpenTime,CloseTime,rating};

tableModel.addRow(rowData);

}

// Create the table with the table model

JTable table = new JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = new JScrollPane(table);

// Create a panel and add the scroll pane to it

JPanel panel = new JPanel();

panel.add(scrollPane);

// Create a frame and add the panel to it

JFrame frame = new JFrame("Hospital Data");

//frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

frame.getContentPane().add(panel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

System.out.println("Failed to connect to the database");

}

}

private void DoctorList()

{

try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "aniruth", "vasavi")) {

String selectQuery = "SELECT \* FROM doctors";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(selectQuery);

// Create the table model with column names

DefaultTableModel tableModel = new DefaultTableModel();

tableModel.addColumn("Doctor ID");

tableModel.addColumn("Doctor Name");

tableModel.addColumn("Specialization ID");

tableModel.addColumn("Hospital ID");

// Populate the table model with data

while (resultSet.next()) {

int did = resultSet.getInt("did");

String dname = resultSet.getString("dname");

int specialization = resultSet.getInt("sid");

int hid = resultSet.getInt("hid");

Object[] rowData = {did, dname, specialization, hid};

tableModel.addRow(rowData);

}

// Create the table with the table model

JTable table = new JTable(tableModel);

// Create a scroll pane and add the table to it

JScrollPane scrollPane = new JScrollPane(table);

// Create a panel and add the scroll pane to it

JPanel panel = new JPanel();

panel.add(scrollPane);

// Create a frame and add the panel to it

JFrame frame = new JFrame("Doctor Data");

//frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

frame.getContentPane().add(panel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

System.out.println("Failed to connect to the database");

}

}

private void removePreviousPanel()

{

if(insertPanel==null)

{

getContentPane().remove(insertPanel);

}

if(updatePanel==null)

{

getContentPane().remove(updatePanel);

}

if(patientPanel==null)

{

getContentPane().remove(patientPanel);

}

if(hospitalPanel==null)

{

getContentPane().remove(hospitalPanel);

}

if(deletePanel==null)

{

getContentPane().remove(deletePanel);

}

if(appointmentPanel==null)

{

getContentPane().remove(appointmentPanel);

}

if(mainPanel==null)

{

getContentPane().remove(mainPanel);

}

if(mainPanel1==null)

{

getContentPane().remove(mainPanel1);

}

if(mainPanel2==null)

{

getContentPane().remove(mainPanel2);

}

if(doctorPanel==null)

{

getContentPane().remove(doctorPanel);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new AyushHospitalFinderForm().setVisible(true);

}

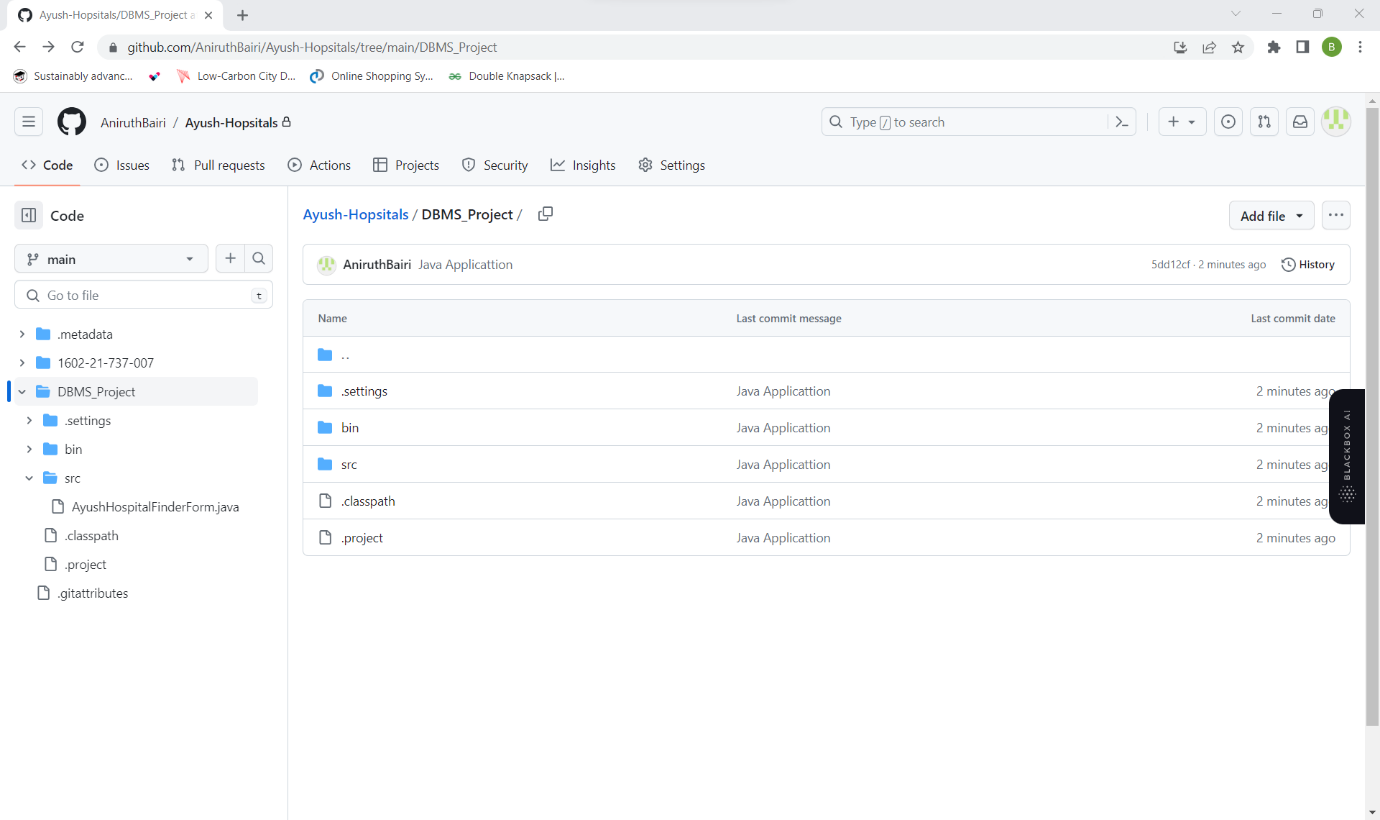
});

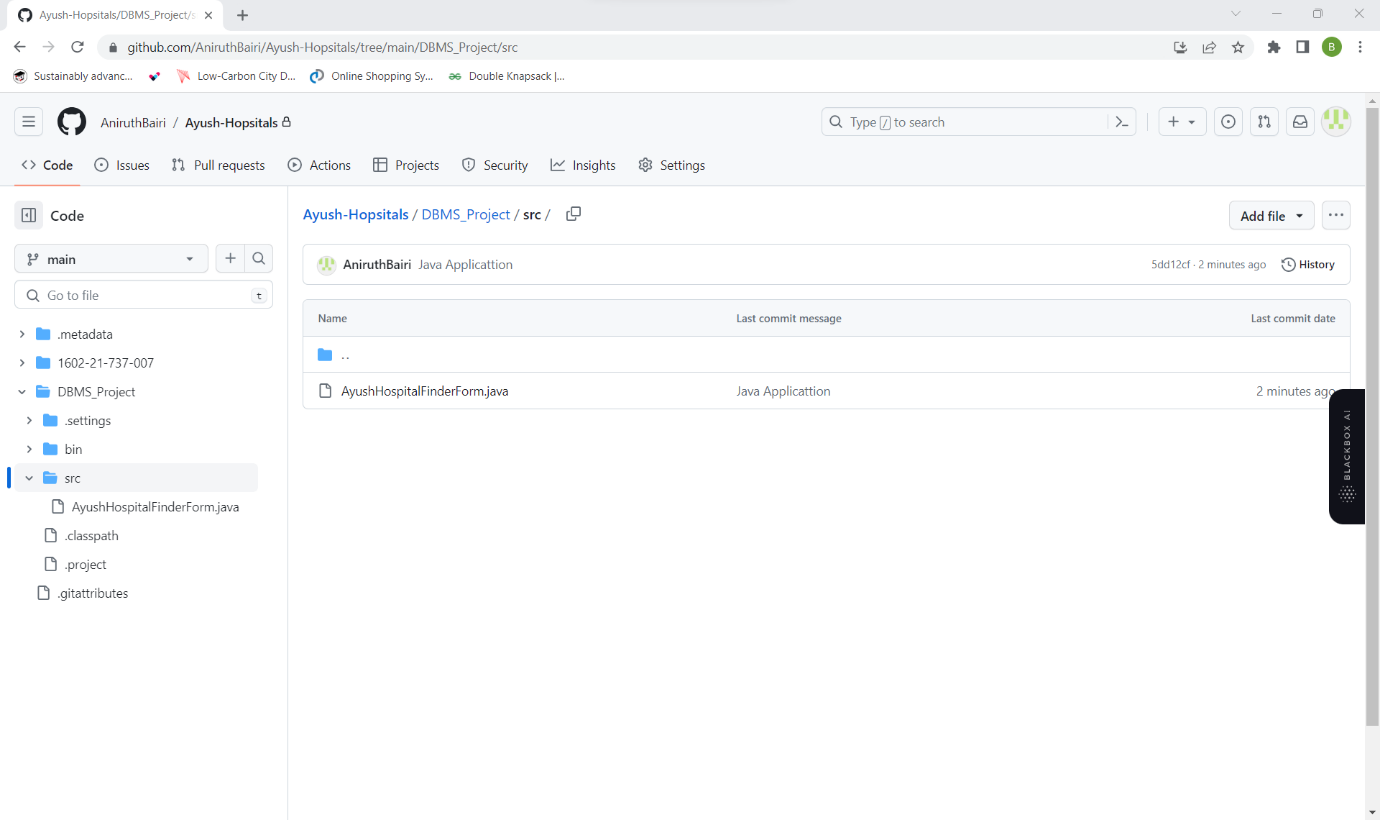
}

}

GitHub Links and Folder Structure:

GitHub Link: <https://github.com/AniruthBairi/Ayush-Hopsitals>

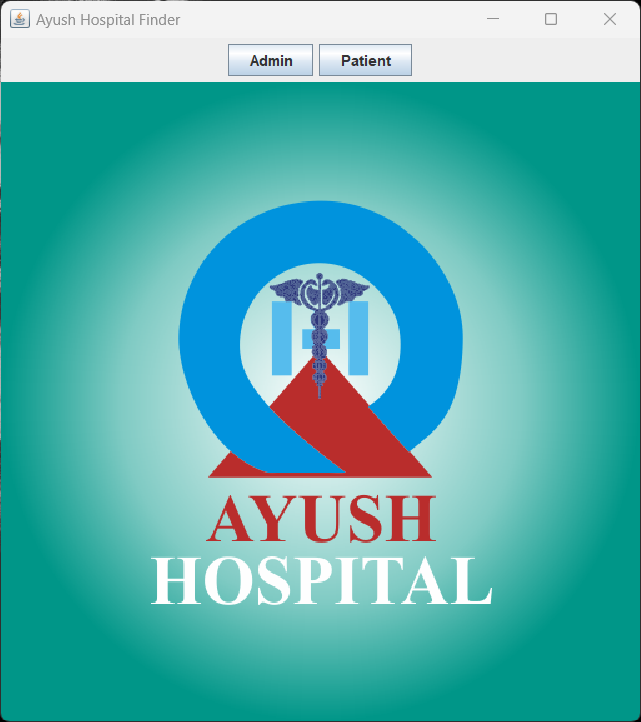




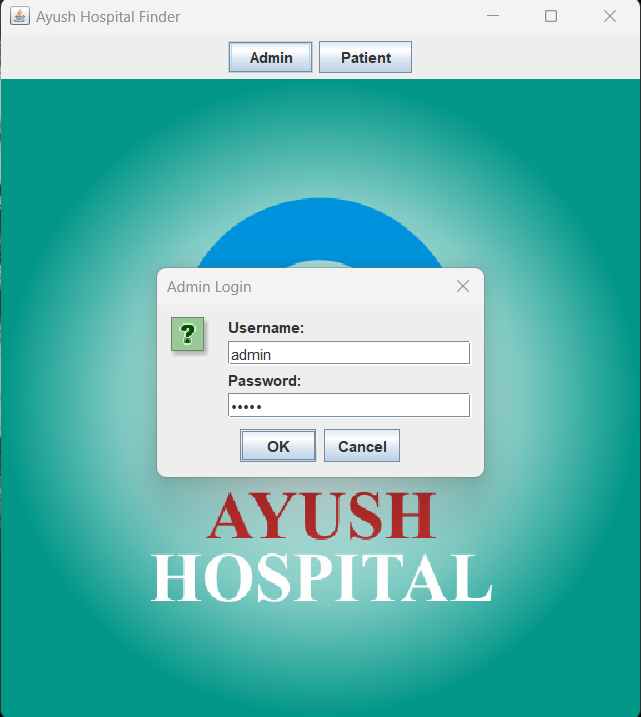
OUTPUT

SCREENSHOTS:

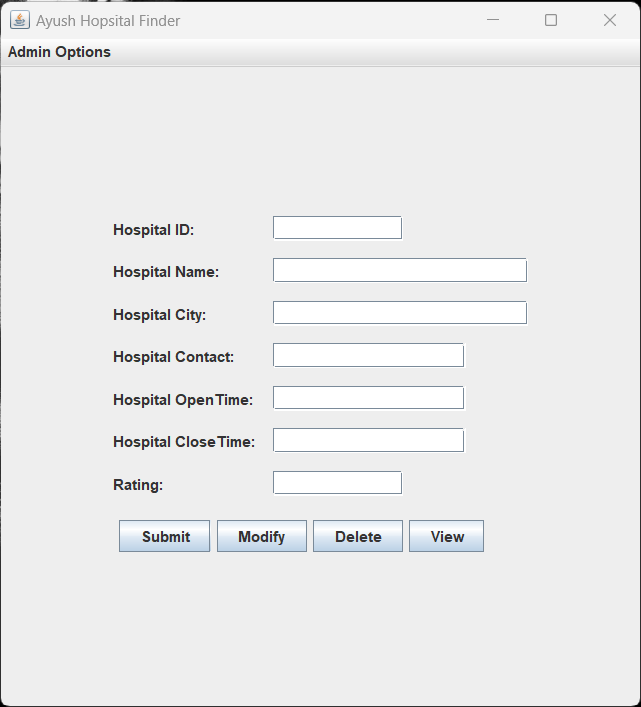
HOME

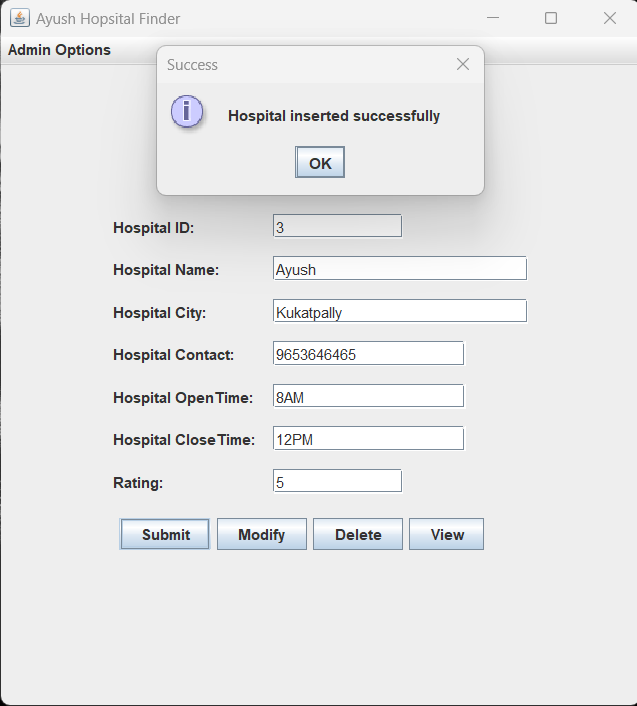


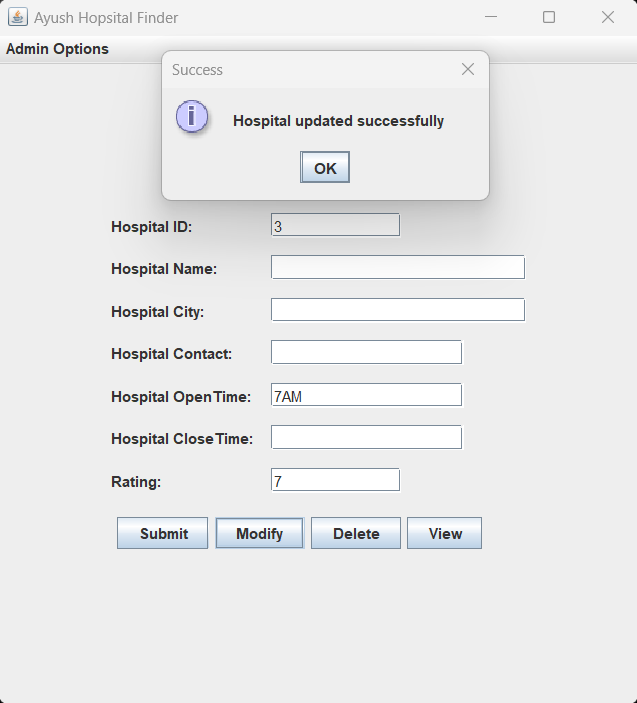
IF USER IS ADMIN:

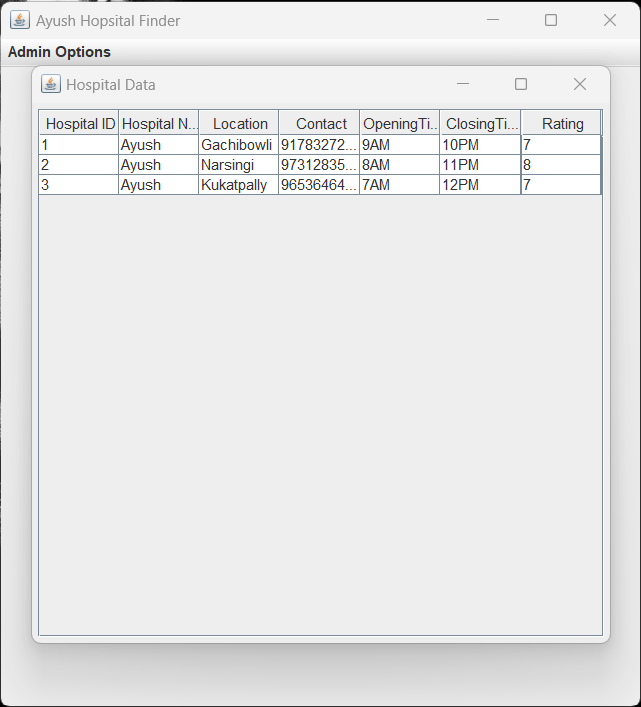


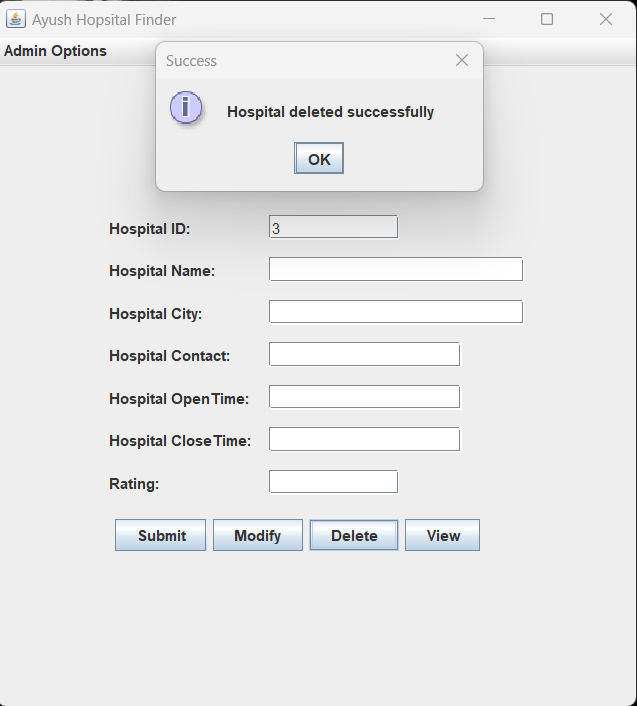


BASED ON OPTIONS SELECTED BY ADMIN: (Hospital Form)

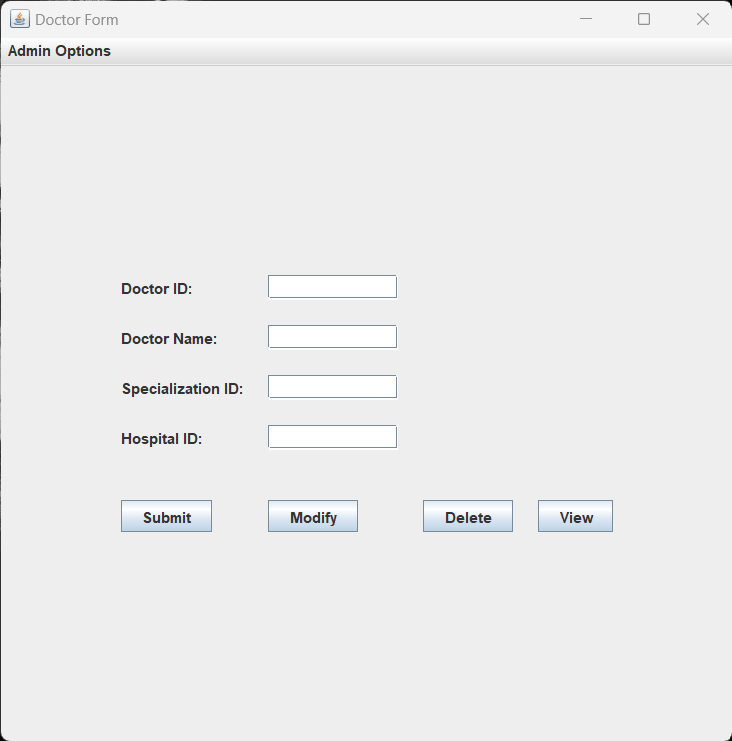


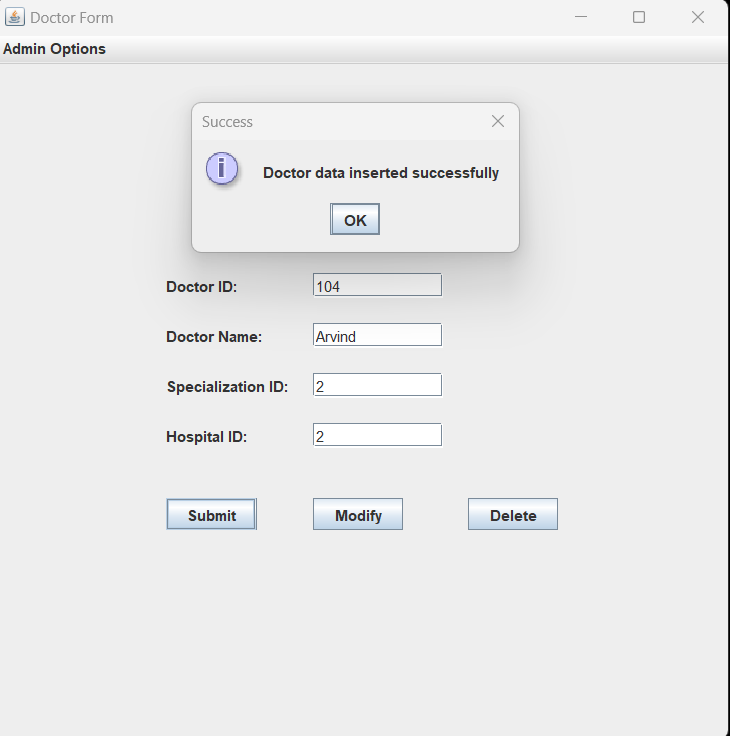


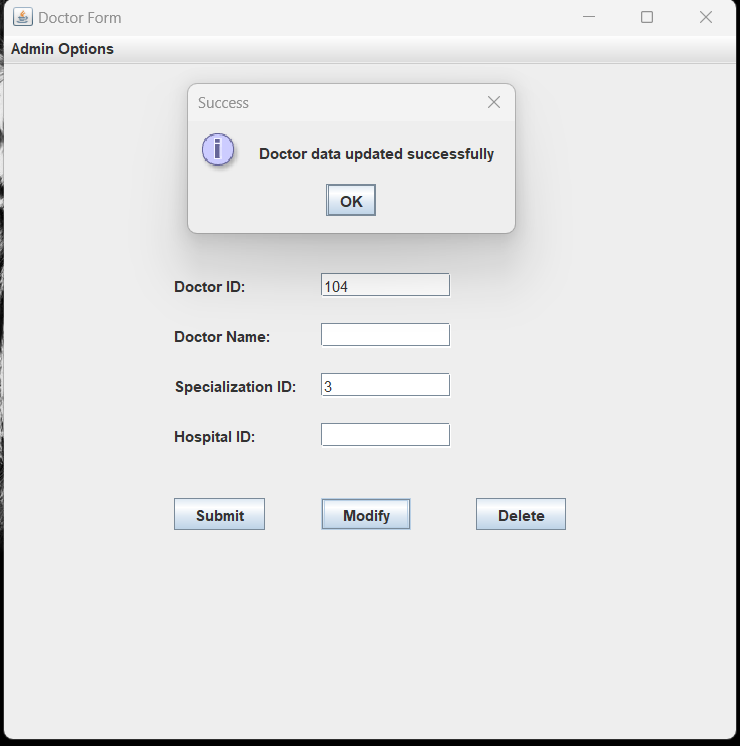


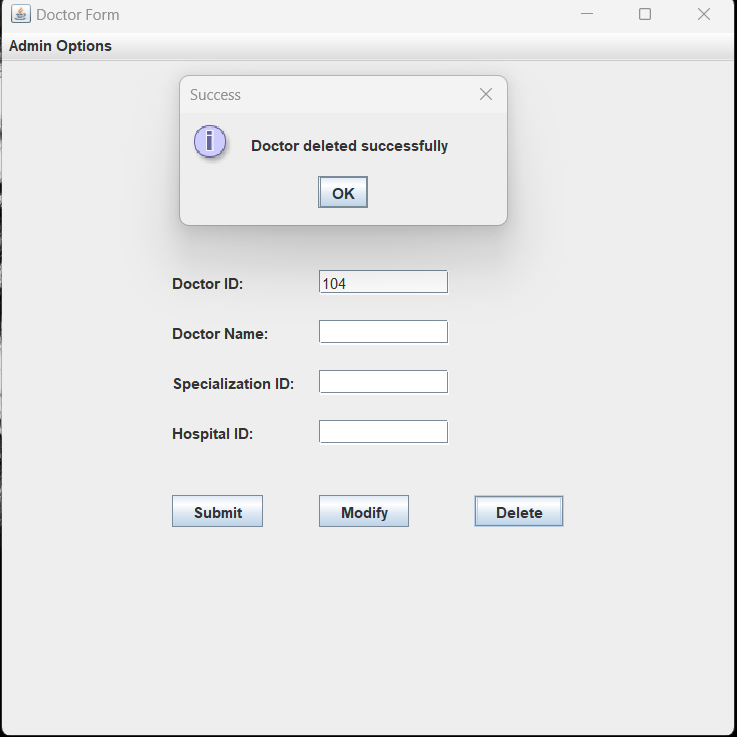


(Doctor Form)

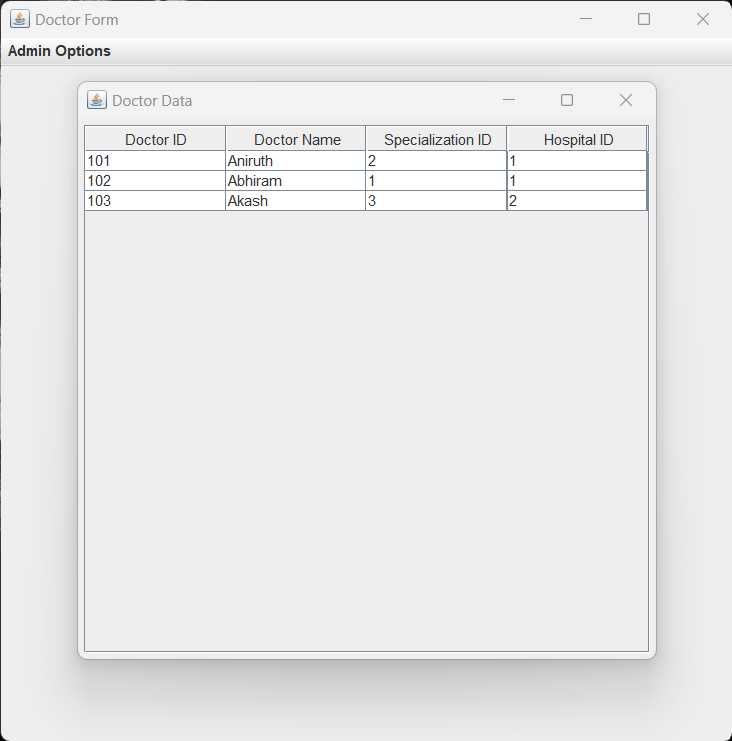




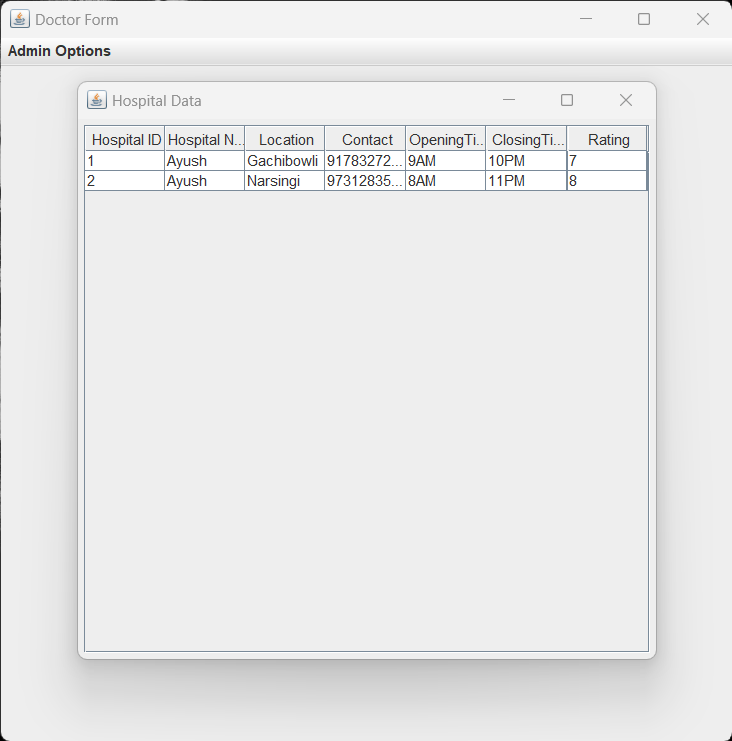




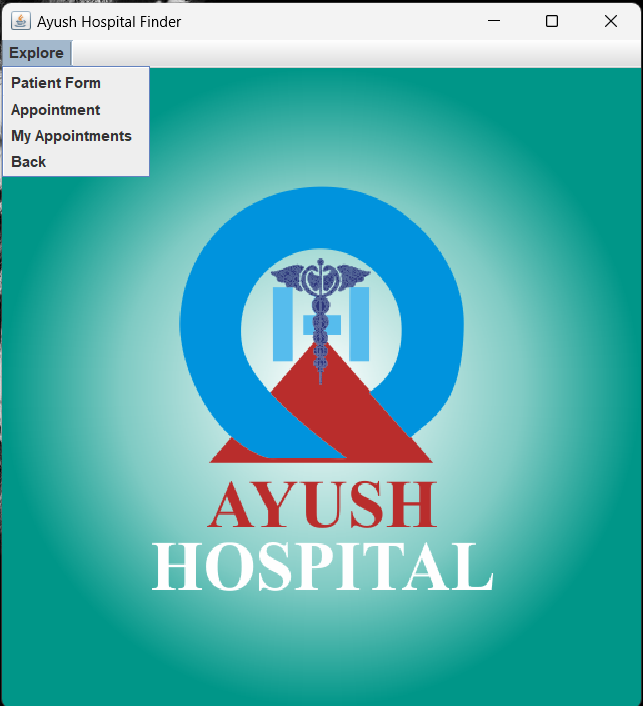
(View Doctors Option)



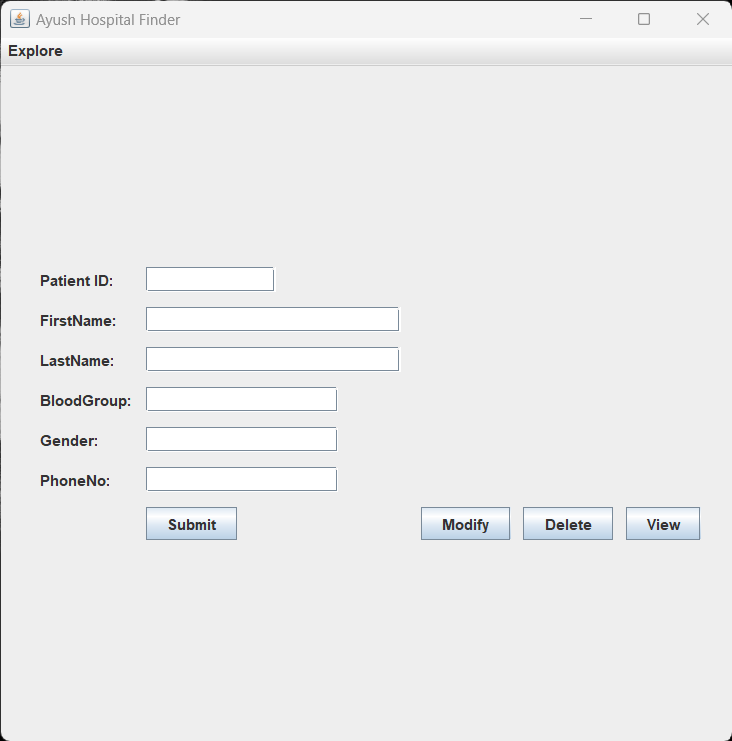
(View Hospitals Option)

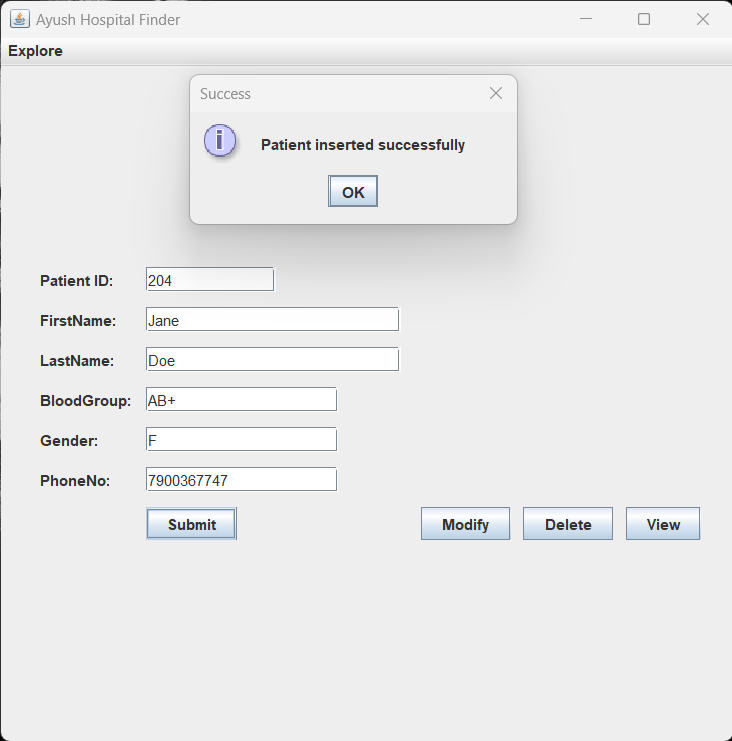


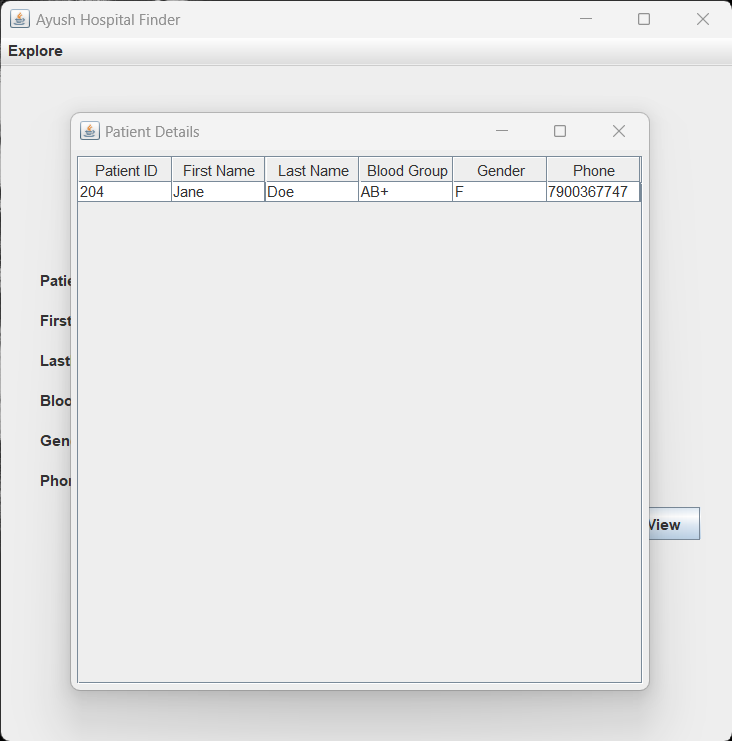
(IF USER CHOOSES PATIENT OPTION)

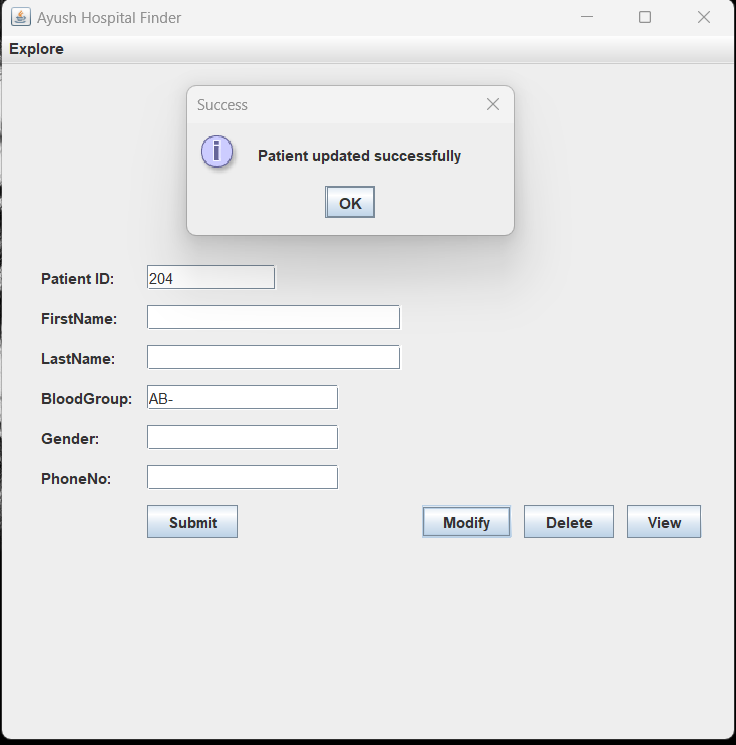


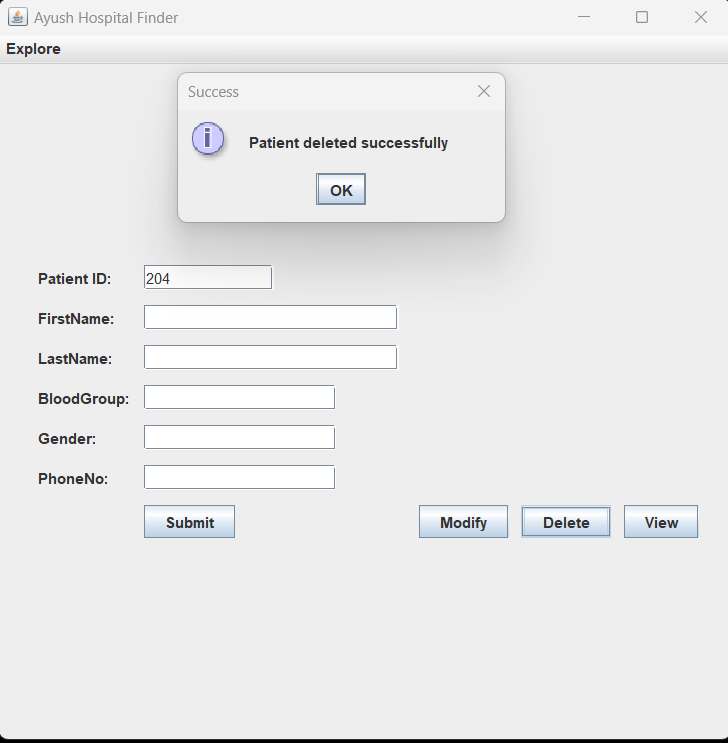
(PATIENT FORM)



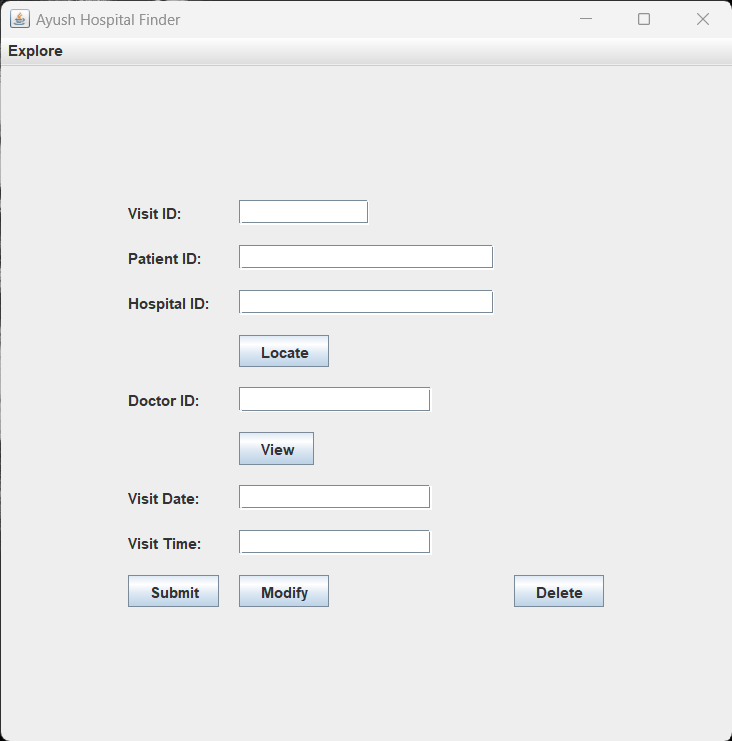




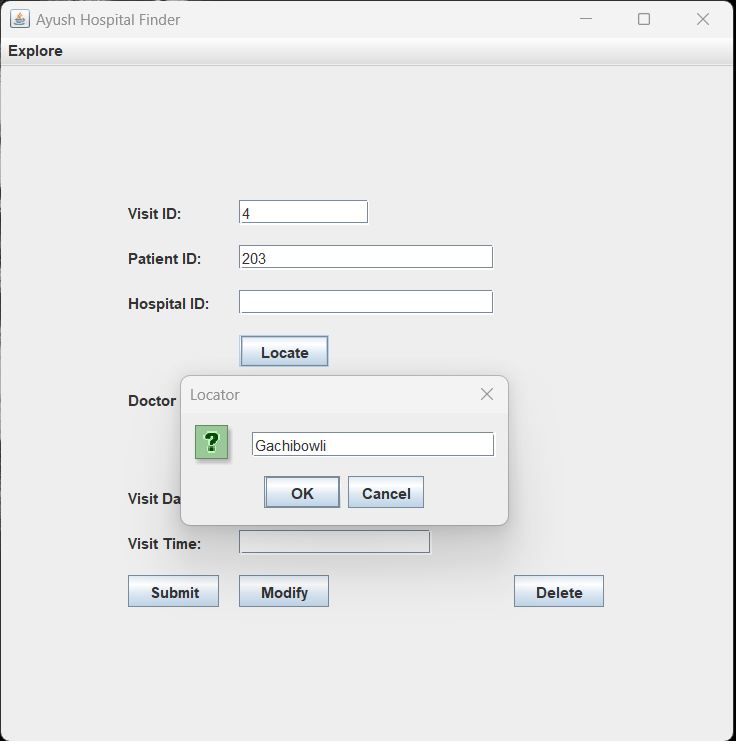


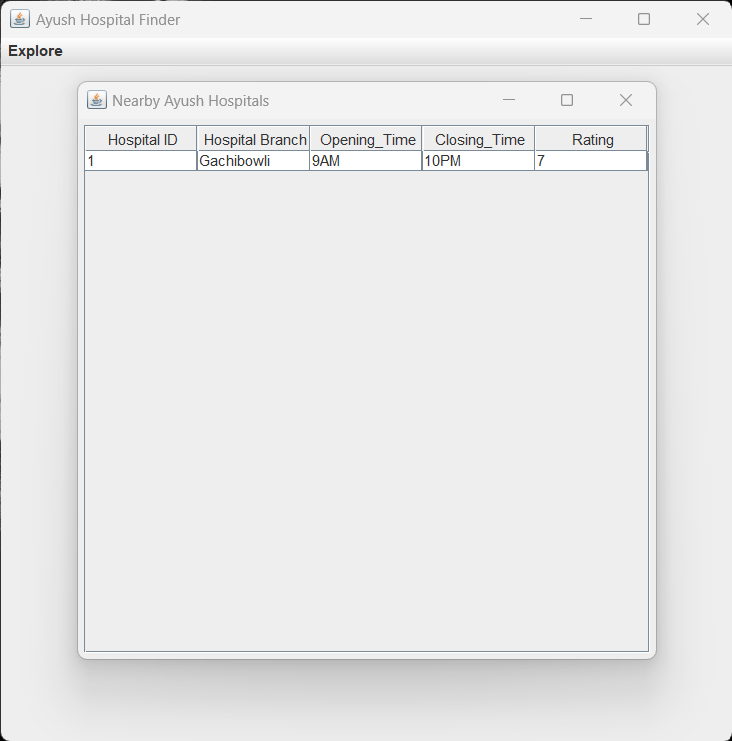


(APPOINTMENT FORM)

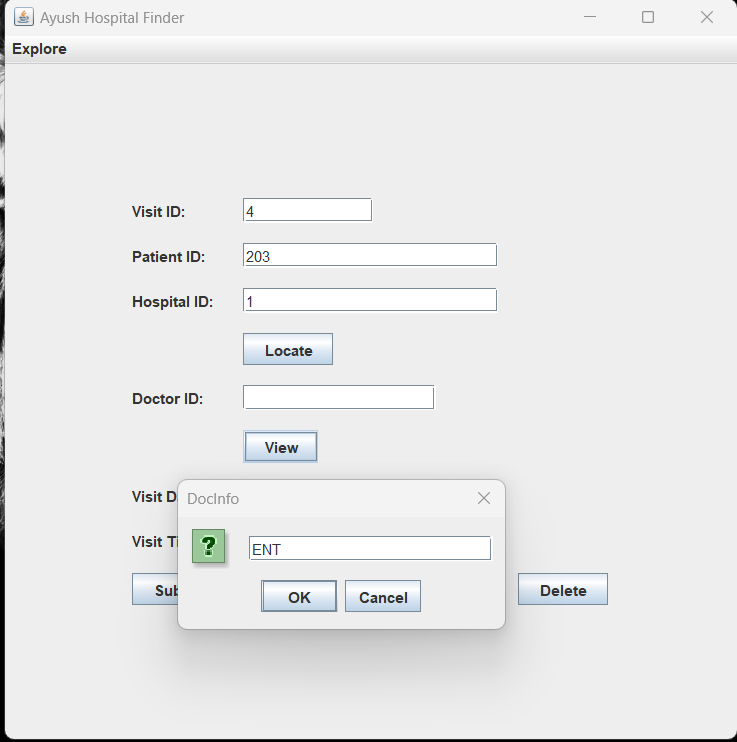


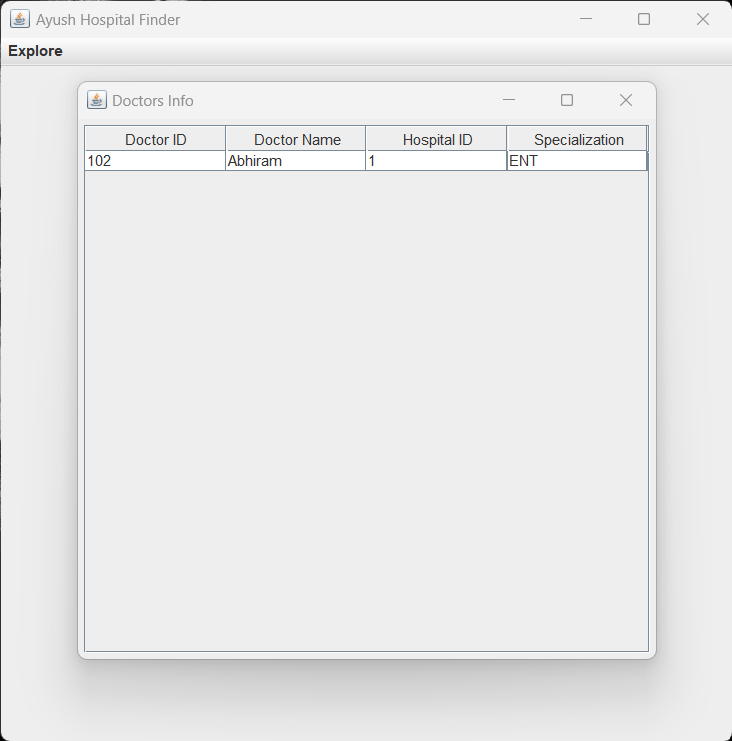
(IF PATIENT DOESN’T KNOW DESIRED HOSPITAL ID)

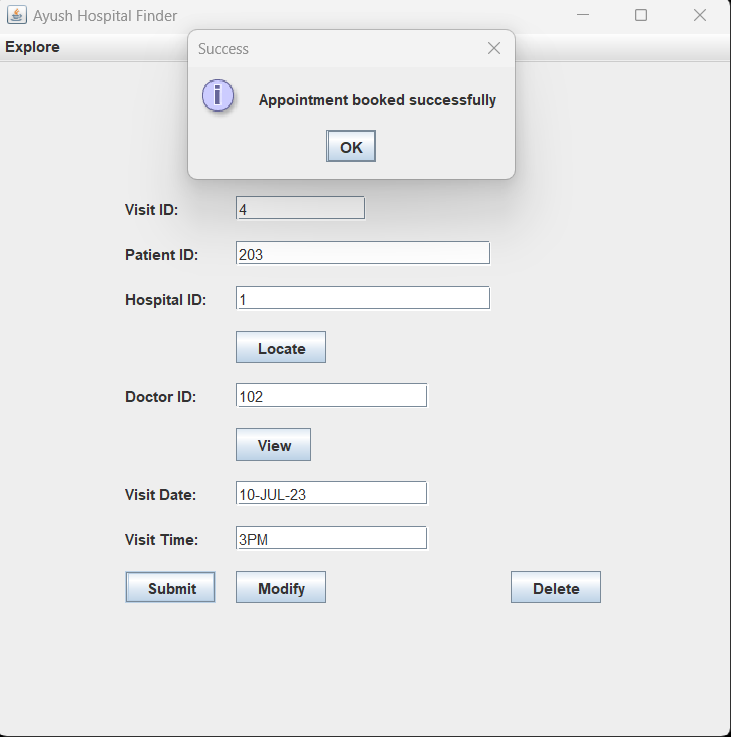


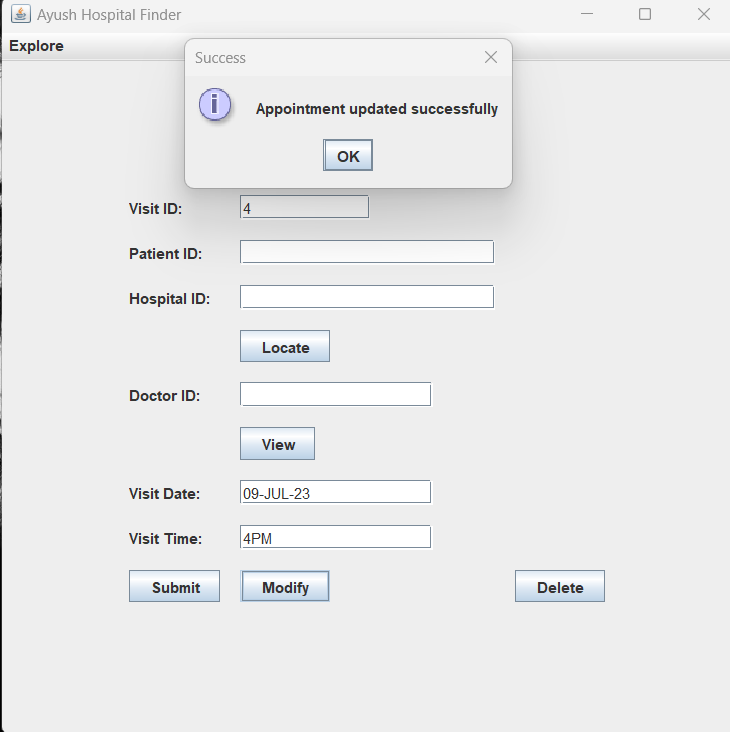


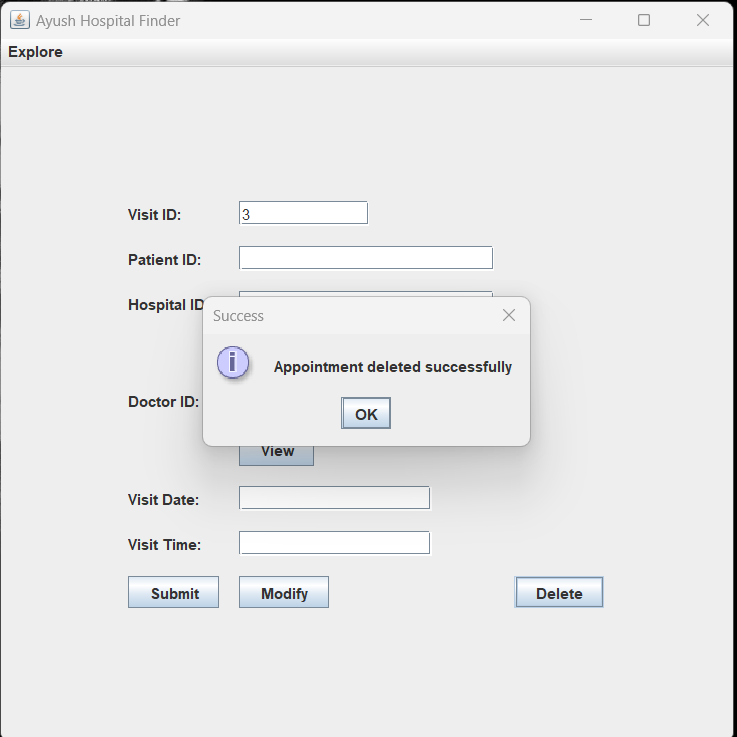
(IF PATIENT DOESN’T KNOW REQUIRED DOCTOR ID)



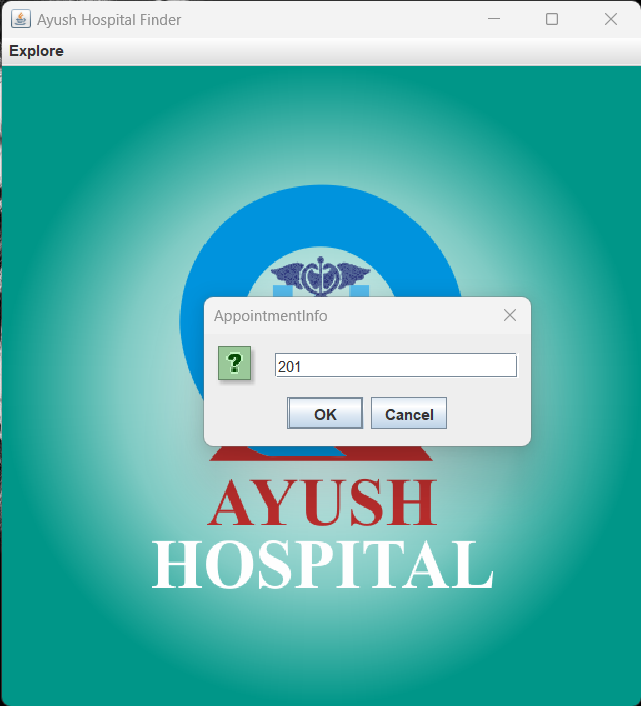


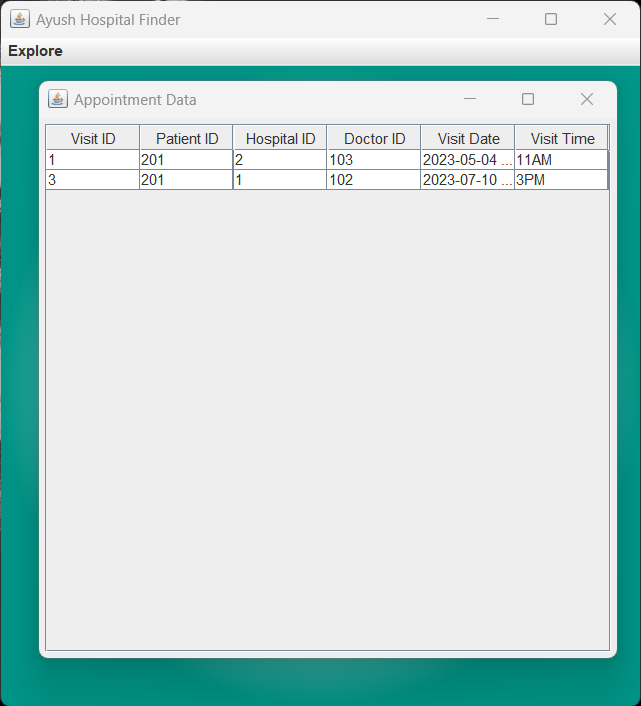






(MY APPOINTMENTS)





RESULTS

I have successfully completed the mini-project “**AYUSH HOSPITAL FINDER**”.

DISCUSSION AND FUTURE WORK

The Ayush Hospital Finder is an important tool that aims to provide users with information about Ayurveda, Yoga, Naturopathy, Unani, Siddha, and Homeopathy (AYUSH) hospitals in their area. It can help individuals locate and access alternative healthcare options and promote the use of traditional and holistic healing practices.

User experience should be a priority in the development of the Ayush Hospital Finder. The interface should be intuitive, user-friendly, and accessible on various devices, including smartphones, as mobile usage continues to rise. Incorporating features such as search filters (e.g., by location, treatment type, hospital rating) and map integration can enhance the usability of the tool.

Expansion of Database: The Ayush Hospital Finder should aim to continually expand its database to cover a wider geographic area. Collaborations with AYUSH regulatory bodies and partnerships with local healthcare institutions can facilitate the acquisition of comprehensive data on AYUSH hospitals nationwide or even globally.

Integration of Telemedicine: With the rise of telemedicine and remote consultations, integrating telemedicine options into the Ayush Hospital Finder can provide users with the convenience of accessing AYUSH treatments and consultations remotely. This can broaden the reach of the tool and cater to individuals who may not have local AYUSH hospitals in their vicinity.

Treatment Information: Enhancing the tool by including detailed information about the various treatments offered by AYUSH hospitals can empower users to make informed decisions. Descriptions of treatments, their benefits, potential risks, and success stories can help users understand and choose the most suitable treatment options.

REFERENCES

https://docs.oracle.com/javase/7/docs/api/

https://www.javatpoint.com/java-swing

https://stackoverflow.com/