DOCTOR APPOINTMENT MANAGEMENT SYSTEM

A MINI-PROJECT BY:

Naveen V - 230701207

S Praveen - 230701245

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"DOCTOR APPOINTMENT MANAGEMENT SYSTEM"** is the bonafide work of **"Naveen V, S Praveen"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Ms. ASWANA LAL

Asst. Professor

Computer Science and Engineering,

Rajalakshmi Engineering College

(Autonomous),

Thandalam,Chennai-602105

SIGNATURE

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The DOCTOR APPOINTMENT MANAGEMENT SYSTEM is a Java Swing-based application integrated with MySQL for managing appointments in a healthcare environment. It allows patients to book appointments, view their schedules, and manage their details, while doctors can view appointments and update their status. The **Admin Dashboard** provides full control over managing doctors, patients, and appointments, with CRUD functionality for each.

The system ensures no scheduling conflicts and offers a user-friendly interface with modern design elements such as background images. It supports seamless transitions between roles, allowing users to efficiently navigate through different views. The application also implements security features, ensuring safe and reliable data management.

This project demonstrates integration between Java Swing and MySQL, providing a scalable and intuitive solution for managing medical appointments while improving user experience and administrative control.

TABLE OF CONTENTS

INTRODUCTION

1.1 INTRODUCTION

The **Doctor-Patient Appointment Scheduling System** is a Java Swing-based application designed for managing doctor-patient appointments. It allows patients to book appointments, view their schedules, and manage personal information. Doctors can manage their appointments and update statuses. The system integrates with MySQL for data management and ensures seamless user interaction.

1.2 IMPLEMENTATION

The system is built using Java Swing for the graphical user interface and MySQL for database management. It features CRUD operations for managing patients, doctors, and appointments. Key functionalities include appointment scheduling, conflict resolution, and a user-friendly interface.

1.3 SCOPE OF THE PROJECT

This system aims to streamline appointment scheduling in medical settings by offering an easy-to-use interface. It covers patient registration, doctor management, and appointment scheduling. Future expansion can include automated notifications and reporting features. The software is scalable and can support growing patient and doctor databases. It reduces administrative workload, enhances scheduling efficiency, and improves the overall healthcare service process.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR : Intel i5

MEMORY SIZE : 4GB(Minimum)

HARD DISK : 500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

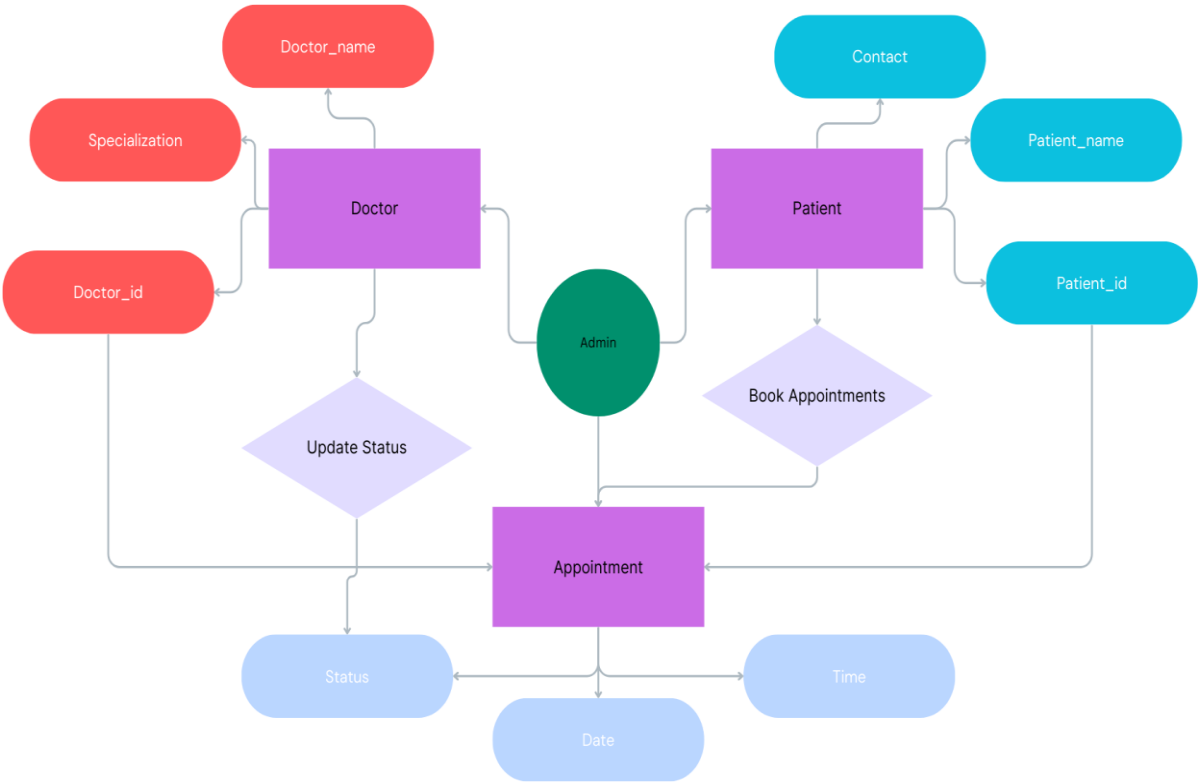
PROGRAMMING LANGUAGE : Java,

MySQL FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 11

ENTITY RELATIONSHIP DIAGRAM



SAMPLE CODE

4.1. LOGIN PAGE

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

public class LoginFrame extends JFrame {
   public LoginFrame() {
      setTitle("Admin Login");
      setLayout(new BorderLayout());
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setExtendedState(JFrame.MAXIMIZED_BOTH);
      JPanel panel = new JPanel();
      panel.setLayout(null);
      panel.setBackground(Color.WHITE);
      JLabel titleLabel = new JLabel("Doctor-Patient Appointment System");
      titleLabel.setFont(new Font("Arial", Font.BOLD, 36));
      titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
      titleLabel.setBounds(400, 50, 600, 50);
      panel.add(titleLabel);
      JLabel usernameLabel = new JLabel("Username:");
      usernameLabel.setBounds(500, 200, 150, 30);
      usernameLabel.setFont(new Font("Arial", Font.PLAIN, 20));
      panel.add(usernameLabel);

      JTextField usernameField = new JTextField();
      usernameField.setBounds(650, 200, 200, 30);
      panel.add(usernameField);

      JLabel passwordLabel = new JLabel("Password:");
      passwordLabel.setBounds(500, 250, 150, 30);
      passwordLabel.setFont(new Font("Arial", Font.PLAIN, 20));
      panel.add(passwordLabel);
      JPasswordField passwordField = new JPasswordField();
      passwordField.setBounds(650, 250, 200, 30);
      panel.add(passwordField);
      JButton loginButton = new JButton("Login");
      loginButton.setBounds(550, 300, 100, 40);
      loginButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            if (username.equals("Shankar") && password.equals("Khushee")) {
```

```
                JOptionPane.showMessageDialog(null, "Login Successful!");
                new AdminDashboardFrame();
                dispose();
            } else {
                JOptionPane.showMessageDialog(null, "Invalid username or password.",
"Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    panel.add(loginButton);
    JButton exitButton = new JButton("Exit");
    exitButton.setBounds(700, 300, 100, 40);
    exitButton.addActionListener(e -> System.exit(0));
    panel.add(exitButton);
    setContentPane(panel);
    setVisible(true);
  }
  public static void main(String[] args) {
    new LoginFrame();
  }
}
```

4.2. ADMIN DASHBOARD

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

public class AdminDashboardFrame extends JFrame {
  public AdminDashboardFrame() {
    setTitle("Admin Dashboard");
    setLayout(new GridLayout(4, 1, 20, 20));
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setExtendedState(JFrame.MAXIMIZED_BOTH);

    JButton manageDoctorsButton = new JButton("Manage Doctors");
    manageDoctorsButton.setFont(new Font("Arial", Font.BOLD, 24));
    manageDoctorsButton.addActionListener(e -> {
      new DoctorManagementFrame();
      dispose();
    });
    JButton managePatientsButton = new JButton("Manage Patients");
    managePatientsButton.setFont(new Font("Arial", Font.BOLD, 24));
    managePatientsButton.addActionListener(e -> {
      new PatientManagementFrame();
      dispose();
    });
```

```
        JButton scheduleAppointmentsButton = new JButton("Schedule Appointments");
        scheduleAppointmentsButton.setFont(new Font("Arial", Font.BOLD, 24));
        scheduleAppointmentsButton.addActionListener(e -> {
            new AppointmentSchedulingFrame();
            dispose();
        });
        JButton viewAppointmentsButton = new JButton("View Appointments");
        viewAppointmentsButton.setFont(new Font("Arial", Font.BOLD, 24));
        viewAppointmentsButton.addActionListener(e -> {
            new ViewAppointmentsFrame();
            dispose();
        });
        add(manageDoctorsButton);
        add(managePatientsButton);
        add(scheduleAppointmentsButton);
        add(viewAppointmentsButton);
        setVisible(true);
    }
}
```

4.3. APPOINTMENT SCHEDULING FRAME

```
import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class AppointmentSchedulingFrame extends JFrame {

    public AppointmentSchedulingFrame() {
        setTitle("Schedule Appointment");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        JPanel panel = new JPanel(new GridLayout(6, 2, 20, 20));
        panel.setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50));

        JLabel patientLabel = new JLabel("Select Patient:");
        JComboBox<String> patientComboBox = new JComboBox<>();
        JLabel doctorLabel = new JLabel("Select Doctor:");
        JComboBox<String> doctorComboBox = new JComboBox<>();
        JLabel dateLabel = new JLabel("Appointment Date (YYYY-MM-DD):");
        JTextField dateField = new JTextField();
        JLabel timeLabel = new JLabel("Appointment Time (HH:MM):");
        JTextField timeField = new JTextField();
        JButton scheduleButton = new JButton("Schedule Appointment");
        JButton backButton = new JButton("Back");
        loadPatients(patientComboBox);
```

```
        loadDoctors(doctorComboBox);

        scheduleButton.addActionListener(e -> {
            String patient = (String) patientComboBox.getSelectedItem();
            String doctor = (String) doctorComboBox.getSelectedItem();
            String date = dateField.getText();
            String time = timeField.getText();
            if (patient != null && doctor != null && !date.isEmpty() && !time.isEmpty()) {
                scheduleAppointment(patient, doctor, date, time);
            } else {
                JOptionPane.showMessageDialog(this, "All fields are required.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });
        backButton.addActionListener(e -> {
            new AdminDashboardFrame();
            dispose();
        });
        panel.add(patientLabel);
        panel.add(patientComboBox);
        panel.add(doctorLabel);
        panel.add(doctorComboBox);
        panel.add(dateLabel);
        panel.add(dateField);
        panel.add(timeLabel);
        panel.add(timeField);
        panel.add(scheduleButton);
        panel.add(backButton);
        add(panel);
        setVisible(true);
    }

    private void scheduleAppointment(String patient, String doctor, String date, String
time) {
        int patientId = Integer.parseInt(patient.split(" - ")[0]);
        int doctorId = Integer.parseInt(doctor.split(" - ")[0]);

        try (Connection connection = DatabaseConnection.connect()) {
            String checkQuery = "SELECT * FROM appointments WHERE doctor_id = ?
AND date = ? AND time = ?";
            PreparedStatement checkStatement =
connection.prepareStatement(checkQuery);
            checkStatement.setInt(1, doctorId);
            checkStatement.setString(2, date);
            checkStatement.setString(3, time);
            ResultSet resultSet = checkStatement.executeQuery();
```

```
            if (resultSet.next()) {
                JOptionPane.showMessageDialog(this, "The doctor already has an
appointment at this time.", "Error", JOptionPane.ERROR_MESSAGE);
            } else {
                String insertQuery = "INSERT INTO appointments (patient_id, doctor_id,
date, time, status) VALUES (?, ?, ?, ?, 'Scheduled')";
                PreparedStatement insertStatement =
connection.prepareStatement(insertQuery);
                insertStatement.setInt(1, patientId);
                insertStatement.setInt(2, doctorId);
                insertStatement.setString(3, date);
                insertStatement.setString(4, time);
                insertStatement.executeUpdate();

                JOptionPane.showMessageDialog(this, "Appointment scheduled
successfully.");
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error scheduling appointment: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

4.4. VIEW APPOINTMENTS FRAME

```
import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class ViewAppointmentsFrame extends JFrame {
    public ViewAppointmentsFrame() {
        setTitle("View Appointments");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout(10, 10));
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
        JLabel titleLabel = new JLabel("Scheduled Appointments",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));
        panel.add(titleLabel, BorderLayout.NORTH);

        JTextArea appointmentsArea = new JTextArea();
```

```
        appointmentsArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(appointmentsArea);
        panel.add(scrollPane, BorderLayout.CENTER);

        JButton backButton = new JButton("Back");
        backButton.setFont(new Font("Arial", Font.BOLD, 18));
        backButton.addActionListener(e -> {
            new AdminDashboardFrame();
            dispose();
        });
        panel.add(backButton, BorderLayout.SOUTH);

        add(panel);

        loadAppointments(appointmentsArea);
        setVisible(true);
    }
    private void loadAppointments(JTextArea appointmentsArea) {
        StringBuilder appointmentList = new StringBuilder();

        try (Connection connection = DatabaseConnection.connect()) {
            String query = "SELECT a.id, p.name AS patient_name, d.name AS
doctor_name, a.date, a.time, a.status " +
                    "FROM appointments a " +
                    "JOIN patients p ON a.patient_id = p.id " +
                    "JOIN doctors d ON a.doctor_id = d.id";
            PreparedStatement statement = connection.prepareStatement(query);
            ResultSet resultSet = statement.executeQuery();
            while (resultSet.next()) {
                appointmentList.append("Appointment ID: ").append(resultSet.getInt("id"))
                        .append("\nPatient: ").append(resultSet.getString("patient_name"))
                        .append("\nDoctor: ").append(resultSet.getString("doctor_name"))
                        .append("\nDate: ").append(resultSet.getString("date"))
                        .append("\nTime: ").append(resultSet.getString("time"))
                        .append("\nStatus: ").append(resultSet.getString("status"))
                        .append("\n_____\n");
            }
        } catch (Exception ex) {
            appointmentList.append("Error loading appointments:
").append(ex.getMessage());
        }
        appointmentsArea.setText(appointmentList.toString());
    }
}
```

4.5. DOCTOR MANAGEMENT FRAME

```
import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class DoctorManagementFrame extends JFrame {
    public DoctorManagementFrame() {
        setTitle("Doctor Management");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        JPanel panel = new JPanel(new GridLayout(4, 1, 20, 20));
        panel.setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50));
        JButton addDoctorButton = new JButton("Add Doctor");
        JButton removeDoctorButton = new JButton("Remove Doctor");
        JButton viewDoctorsButton = new JButton("View All Doctors");
        JButton backButton = new JButton("Back");

        addDoctorButton.setFont(new Font("Arial", Font.BOLD, 20));
        removeDoctorButton.setFont(new Font("Arial", Font.BOLD, 20));
        viewDoctorsButton.setFont(new Font("Arial", Font.BOLD, 20));
        backButton.setFont(new Font("Arial", Font.BOLD, 20));
        panel.add(addDoctorButton);
        panel.add(removeDoctorButton);
        panel.add(viewDoctorsButton);
        panel.add(backButton);
        add(panel);
        addDoctorButton.addActionListener(e -> addDoctor());
        removeDoctorButton.addActionListener(e -> removeDoctor());
        viewDoctorsButton.addActionListener(e -> viewDoctors());
        backButton.addActionListener(e -> {
            new AdminDashboardFrame();
            dispose();
        });
        setVisible(true);
    }
    private void addDoctor() {
        String doctorName = JOptionPane.showInputDialog("Enter Doctor Name:");
        String specialization = JOptionPane.showInputDialog("Enter Specialization:");
        if (doctorName != null && specialization != null) {
            try (Connection connection = DatabaseConnection.connect()) {
                String query = "INSERT INTO doctors (name, specialization) VALUES (?,
?)";
                PreparedStatement statement = connection.prepareStatement(query);
                statement.setString(1, doctorName);
```

```
            statement.setString(2, specialization);
            statement.executeUpdate();

            JOptionPane.showMessageDialog(this, "Doctor added successfully.");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error adding doctor: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
  }

  private void removeDoctor() {
    String doctorId = JOptionPane.showInputDialog("Enter Doctor ID to Remove:");

    if (doctorId != null) {
      try (Connection connection = DatabaseConnection.connect()) {
        String query = "DELETE FROM doctors WHERE id = ?";
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt(1, Integer.parseInt(doctorId));
        int rowsAffected = statement.executeUpdate();
        if (rowsAffected > 0) {
          JOptionPane.showMessageDialog(this, "Doctor removed successfully.");
        } else {
          JOptionPane.showMessageDialog(this, "Doctor ID not found.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
      } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Error removing doctor: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
      }
    }
  }
  private void viewDoctors() {
    StringBuilder doctorList = new StringBuilder("Doctor List:\n\n");

    try (Connection connection = DatabaseConnection.connect()) {
      String query = "SELECT * FROM doctors";
      PreparedStatement statement = connection.prepareStatement(query);
      ResultSet resultSet = statement.executeQuery();
      while (resultSet.next()) {
        doctorList.append("ID: ").append(resultSet.getInt("id"))
            .append(", Name: ").append(resultSet.getString("name"))
            .append(", Specialization:
").append(resultSet.getString("specialization"))
            .append("\n");
      }
```

```
            JOptionPane.showMessageDialog(this, doctorList.toString(), "Doctors",
JOptionPane.INFORMATION_MESSAGE);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error fetching doctors: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

4.6. PATIENT MANAGEMENT FRAME

```
import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class PatientManagementFrame extends JFrame {
    public PatientManagementFrame() {
        setTitle("Patient Management");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        JPanel panel = new JPanel(new GridLayout(4, 1, 20, 20));
        panel.setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50));
        JButton addPatientButton = new JButton("Add Patient");
        JButton removePatientButton = new JButton("Remove Patient");
        JButton viewPatientsButton = new JButton("View All Patients");
        JButton backButton = new JButton("Back");
        addPatientButton.setFont(new Font("Arial", Font.BOLD, 20));
        removePatientButton.setFont(new Font("Arial", Font.BOLD, 20));
        viewPatientsButton.setFont(new Font("Arial", Font.BOLD, 20));
        backButton.setFont(new Font("Arial", Font.BOLD, 20));

        panel.add(addPatientButton);
        panel.add(removePatientButton);
        panel.add(viewPatientsButton);
        panel.add(backButton);
        add(panel);
        addPatientButton.addActionListener(e -> addPatient());
        removePatientButton.addActionListener(e -> removePatient());
        viewPatientsButton.addActionListener(e -> viewPatients());
        backButton.addActionListener(e -> {
            new AdminDashboardFrame();
            dispose();
        });
        setVisible(true);
    }
```

```
    private void addPatient() {
        String patientName = JOptionPane.showInputDialog("Enter Patient Name:");
        String contact = JOptionPane.showInputDialog("Enter Contact Number:");
        if (patientName != null && contact != null) {
            try (Connection connection = DatabaseConnection.connect()) {
                String query = "INSERT INTO patients (name, contact) VALUES (?, ?)";
                PreparedStatement statement = connection.prepareStatement(query);
                statement.setString(1, patientName);
                statement.setString(2, contact);
                statement.executeUpdate();
                JOptionPane.showMessageDialog(this, "Patient added successfully.");
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this, "Error adding patient: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    private void removePatient() {
        String patientId = JOptionPane.showInputDialog("Enter Patient ID to Remove:");
        if (patientId != null) {
            try (Connection connection = DatabaseConnection.connect()) {
                String query = "DELETE FROM patients WHERE id = ?";
                PreparedStatement statement = connection.prepareStatement(query);
                statement.setInt(1, Integer.parseInt(patientId));
                int rowsAffected = statement.executeUpdate();
                if (rowsAffected > 0) {
                    JOptionPane.showMessageDialog(this, "Patient removed successfully.");
                } else {
                    JOptionPane.showMessageDialog(this, "Patient ID not found.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this, "Error removing patient: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    private void viewPatients() {
        StringBuilder patientList = new StringBuilder("Patient List:\n\n");

        try (Connection connection = DatabaseConnection.connect()) {
            String query = "SELECT * FROM patients";
            PreparedStatement statement = connection.prepareStatement(query);
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()) {
                patientList.append("ID: ").append(resultSet.getInt("id"))
```

```
                .append(", Name: ").append(resultSet.getString("name"))
                .append(", Contact: ").append(resultSet.getString("contact"))
                .append("\n");
        }

        JOptionPane.showMessageDialog(this, patientList.toString(), "Patients",
JOptionPane.INFORMATION_MESSAGE);
      } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Error fetching patients: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
      }
   }
}
```

4.7. JDBC CONNECTION

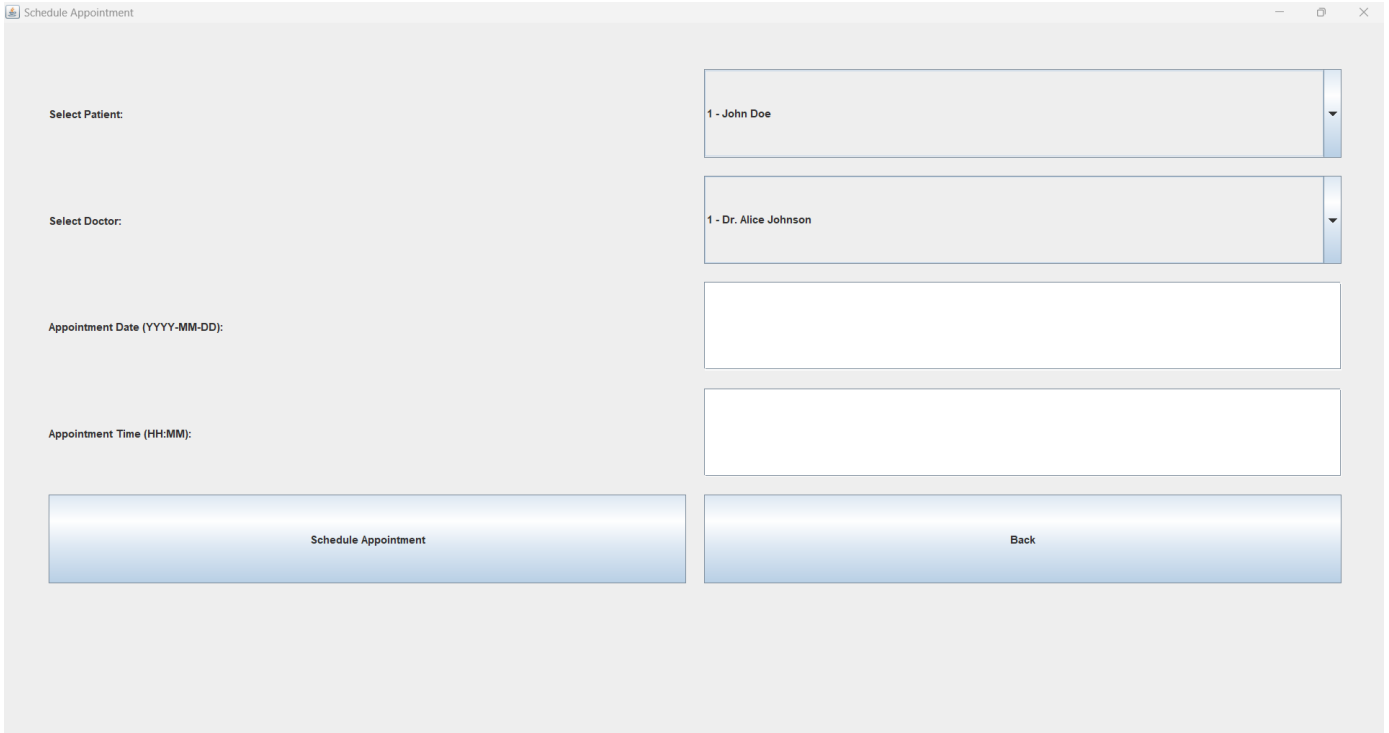
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
   private static final String URL =
"jdbc:mysql://localhost:3306/doctor_patient_system";
   private static final String USERNAME = "root";
   private static final String PASSWORD = "Shankysathu2!";
   public static Connection connect() {
      try {
         return DriverManager.getConnection(URL, USERNAME, PASSWORD);
      } catch (SQLException e) {
         System.err.println("Database connection failed!");
         e.printStackTrace();
         return null;
      }
   }
}
```

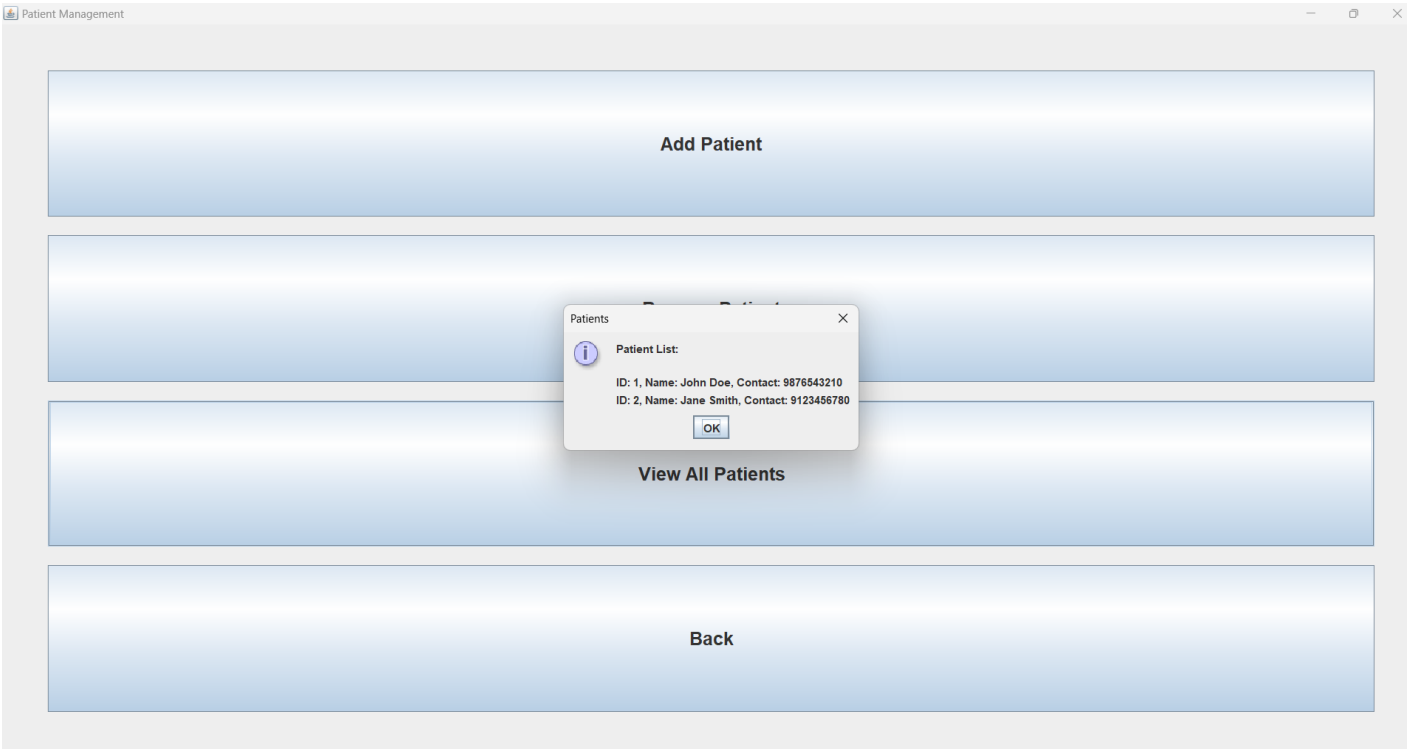
SNAPSHOTS

5.1 ADMIN DASHBOARD

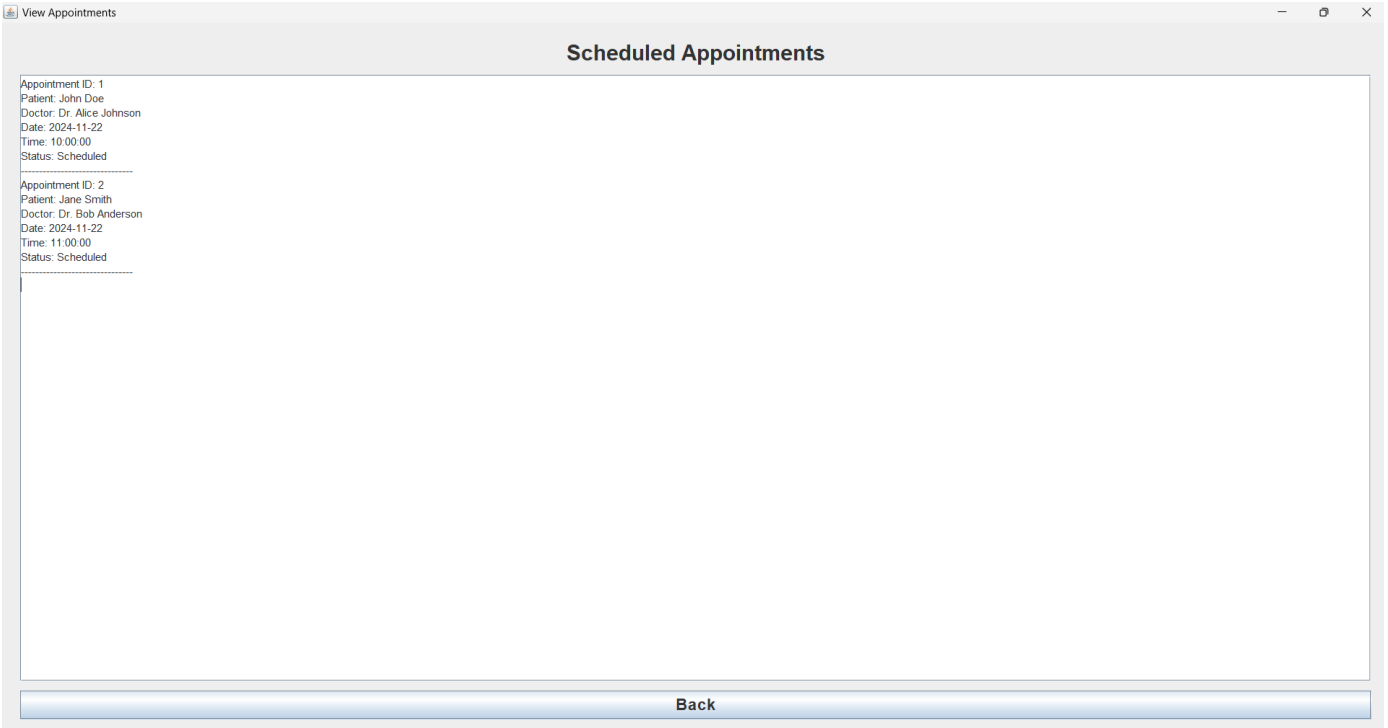
5.2 APPOINTMENT SCHEDULING



5.3. PATIENT MANAGEMENT



5.4 VIEWING SCHEDULED APPOINTMENTS



CONCLUSION

The Doctor-Patient Appointment Scheduling System is a Java-based application with MySQL backend, designed to streamline the management of doctor-patient appointments. It offers a user-friendly interface with features for managing doctors, patients, and appointments. Admins can add, remove, and view doctors and patients, while doctors can manage their schedules and update appointment statuses. Patients can schedule appointments and view available doctors.

The system's intuitive design, full-screen mode, and seamless transitions between interfaces enhance the user experience. With secure database integration, the system offers a scalable solution for managing medical appointments and can be adapted for real-world healthcare environments.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL | Codecademy
5. https://chatgpt.com/