project code

```
import javax.swing.*;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class HomeTuitionManagementSystem extends JFrame {

    private static final String URL = "jdbc:mysql://localhost:3306/TuitionDB"; // Change database
URL as needed
    private static final String USER = "root"; // Change username as needed
    private static final String PASSWORD = "Vigshan@2116"; // Change password as needed

    // Buttons for the main menu
    private JButton addStudentButton, viewStudentsButton, addTeacherButton,
viewTeachersButton, assignTeacherButton;

    public HomeTuitionManagementSystem() {
        // Set up frame
        setTitle("Home Tuition Management System");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Initialize main menu buttons
        addStudentButton = new JButton("Add Student");
        viewStudentsButton = new JButton("View Students");
        addTeacherButton = new JButton("Add Teacher");
        viewTeachersButton = new JButton("View Teachers");
        assignTeacherButton = new JButton("Assign Teacher");

        // Set layout for main panel
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 1, 10, 10));

        panel.add(addStudentButton);
        panel.add(viewStudentsButton);
        panel.add(addTeacherButton);
        panel.add(viewTeachersButton);
        panel.add(assignTeacherButton);

        add(panel);

        // Button event listeners
        addStudentButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                addStudent();
            }
        });
        viewStudentsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewStudentSchedule(); // Show full schedule for students
            }
        });

        addTeacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                addTeacher();
            }
        });

        viewTeachersButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewTeacherSchedule(); // Show full schedule for teachers
            }
        });

        assignTeacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                assignTeacher();
            }
        });
    }

    // Database connection method
    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Add student dialog
    private void addStudent() {
        JTextField nameField = new JTextField(20);
        JTextField ageField = new JTextField(20);
        JTextField contactField = new JTextField(20);
        JTextField feesField = new JTextField(20); // Fees input field

        JPanel panel = new JPanel(new GridLayout(4, 2));
        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Age:"));
        panel.add(ageField);
        panel.add(new JLabel("Contact Number:"));
        panel.add(contactField);
        panel.add(new JLabel("Fees:"));
        panel.add(feesField);

        int option = JOptionPane.showConfirmDialog(this, panel, "Add New Student",
JOptionPane.OK_CANCEL_OPTION);

        if (option == JOptionPane.OK_OPTION) {
            String name = nameField.getText();
            int age = Integer.parseInt(ageField.getText());
            String contact = contactField.getText();
            double fees = Double.parseDouble(feesField.getText()); // Parsing fees

            try (Connection conn = getConnection()) {
                String query = "INSERT INTO students (name, age, contact_number, fees) VALUES (?, ?, ?,
?)";
                try (PreparedStatement ps = conn.prepareStatement(query)) {
                    ps.setString(1, name);
                    ps.setInt(2, age);
                    ps.setString(3, contact);
                    ps.setDouble(4, fees); // Inserting fees
                    ps.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Student added successfully!");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Add teacher dialog
    private void addTeacher() {
        JTextField nameField = new JTextField(20);
        JTextField subjectField = new JTextField(20);
        JTextField contactField = new JTextField(20);

        JPanel panel = new JPanel(new GridLayout(3, 2));
        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Subject:"));
        panel.add(subjectField);
        panel.add(new JLabel("Contact Number:"));
        panel.add(contactField);

        int option = JOptionPane.showConfirmDialog(this, panel, "Add New Teacher",
JOptionPane.OK_CANCEL_OPTION);

        if (option == JOptionPane.OK_OPTION) {
            String name = nameField.getText();
            String subject = subjectField.getText();
            String contact = contactField.getText();

            try (Connection conn = getConnection()) {
                String query = "INSERT INTO teachers (name, subject, contact_number) VALUES (?, ?, ?)";
                try (PreparedStatement ps = conn.prepareStatement(query)) {
                    ps.setString(1, name);
                    ps.setString(2, subject);
                    ps.setString(3, contact);
                    ps.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Teacher added successfully!");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Assign teacher to student
    private void assignTeacher() {
        String[] students = getStudentNames();
        String[] teachers = getTeacherNames();

        if (students != null && teachers != null) {
            String studentName = (String) JOptionPane.showInputDialog(this, "Select Student", "Assign
Teacher",
                    JOptionPane.QUESTION_MESSAGE, null, students, students[0]);

            String teacherName = (String) JOptionPane.showInputDialog(this, "Select Teacher", "Assign
Teacher",
                    JOptionPane.QUESTION_MESSAGE, null, teachers, teachers[0]);

            if (studentName != null && teacherName != null) {
                try (Connection conn = getConnection()) {
                    int studentId = getStudentId(studentName, conn);
                    int teacherId = getTeacherId(teacherName, conn);

                    String schedule = JOptionPane.showInputDialog(this, "Enter Schedule");

                    String query = "INSERT INTO assignments (student_id, teacher_id, schedule) VALUES (?,
?, ?)";
                    try (PreparedStatement ps = conn.prepareStatement(query)) {
                        ps.setInt(1, studentId);
                        ps.setInt(2, teacherId);
                        ps.setString(3, schedule);
                        ps.executeUpdate();
                        JOptionPane.showMessageDialog(this, "Teacher assigned to student successfully!");
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    // Get list of student names for assignment
    private String[] getStudentNames() {
        try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement("SELECT name FROM students")) {
            ResultSet rs = ps.executeQuery();
            StringBuilder students = new StringBuilder();
            while (rs.next()) {
                students.append(rs.getString("name")).append(",");
            }
            return students.toString().split(",");
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Get list of teacher names for assignment
    private String[] getTeacherNames() {
        try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement("SELECT name FROM teachers")) {
            ResultSet rs = ps.executeQuery();
            StringBuilder teachers = new StringBuilder();
            while (rs.next()) {
                teachers.append(rs.getString("name")).append(",");
            }
            return teachers.toString().split(",");
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Get student ID by name
    private int getStudentId(String name, Connection conn) throws SQLException {
        String query = "SELECT id FROM students WHERE name = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setString(1, name);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return rs.getInt("id");
            }
        }
        return -1;
    }

    // Get teacher ID by name
    private int getTeacherId(String name, Connection conn) throws SQLException {
        String query = "SELECT id FROM teachers WHERE name = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setString(1, name);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return rs.getInt("id");
            }
        }
        return -1;
    }

    // View full schedule of students
    private void viewStudentSchedule() {
        String query = "SELECT students.name AS student_name, teachers.name AS teacher_name,
assignments.schedule " +
                       "FROM assignments " +
                       "JOIN students ON assignments.student_id = students.id " +
                       "JOIN teachers ON assignments.teacher_id = teachers.id";
        try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement(query)) {
            ResultSet rs = ps.executeQuery();

            StringBuilder scheduleList = new StringBuilder();
            while (rs.next()) {
                scheduleList.append("Student: ").append(rs.getString("student_name"))
                        .append(", Teacher: ").append(rs.getString("teacher_name"))
                        .append(", Schedule: ").append(rs.getString("schedule"))
                        .append("\n");
            }
            if (scheduleList.length() > 0) {
                JOptionPane.showMessageDialog(this, scheduleList.toString(), "Student Schedules",
JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "No schedules found.", "No Data",
JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // View full schedule of teachers
    private void viewTeacherSchedule() {
        String query = "SELECT teachers.name AS teacher_name, students.name AS student_name,
assignments.schedule " +
                       "FROM assignments " +
                       "JOIN teachers ON assignments.teacher_id = teachers.id " +
                       "JOIN students ON assignments.student_id = students.id";
        try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement(query)) {
            ResultSet rs = ps.executeQuery();

            StringBuilder scheduleList = new StringBuilder();
            while (rs.next()) {
                scheduleList.append("Teacher: ").append(rs.getString("teacher_name"))
                        .append(", Student: ").append(rs.getString("student_name"))
                        .append(", Schedule: ").append(rs.getString("schedule"))
                        .append("\n");
            }
            if (scheduleList.length() > 0) {
                JOptionPane.showMessageDialog(this, scheduleList.toString(), "Teacher Schedules",
JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "No schedules found.", "No Data",
JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Main method to launch the application
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new HomeTuitionManagementSystem().setVisible(true);
            }
        });
    }
}import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class HomeTuitionManagementSystem extends JFrame {

    private static final String URL = "jdbc:mysql://localhost:3306/TuitionDB"; // Change database
URL as needed
    private static final String USER = "root"; // Change username as needed
    private static final String PASSWORD = "Vigshan@2116"; // Change password as needed

    // Buttons for the main menu
    private JButton addStudentButton, viewStudentsButton, addTeacherButton,
viewTeachersButton, assignTeacherButton;

    public HomeTuitionManagementSystem() {
        // Set up frame
        setTitle("Home Tuition Management System");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Initialize main menu buttons
        addStudentButton = new JButton("Add Student");
        viewStudentsButton = new JButton("View Students");
        addTeacherButton = new JButton("Add Teacher");
        viewTeachersButton = new JButton("View Teachers");
        assignTeacherButton = new JButton("Assign Teacher");

        // Set layout for main panel
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 1, 10, 10));

        panel.add(addStudentButton);
        panel.add(viewStudentsButton);
        panel.add(addTeacherButton);
        panel.add(viewTeachersButton);
        panel.add(assignTeacherButton);

        add(panel);

        // Button event listeners
        addStudentButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                addStudent();
            }
        });

        viewStudentsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewStudentSchedule(); // Show full schedule for students
            }
        });

        addTeacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                addTeacher();
            }
        });

        viewTeachersButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewTeacherSchedule(); // Show full schedule for teachers
            }
        });

        assignTeacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                assignTeacher();
            }
        });
    }

    // Database connection method
    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Add student dialog
    private void addStudent() {
        JTextField nameField = new JTextField(20);
        JTextField ageField = new JTextField(20);
        JTextField contactField = new JTextField(20);
        JTextField feesField = new JTextField(20); // Fees input field

        JPanel panel = new JPanel(new GridLayout(4, 2));
        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Age:"));
        panel.add(ageField);
        panel.add(new JLabel("Contact Number:"));
        panel.add(contactField);
        panel.add(new JLabel("Fees:"));
        panel.add(feesField);

        int option = JOptionPane.showConfirmDialog(this, panel, "Add New Student",
JOptionPane.OK_CANCEL_OPTION);

        if (option == JOptionPane.OK_OPTION) {
            String name = nameField.getText();
            int age = Integer.parseInt(ageField.getText());
            String contact = contactField.getText();
            double fees = Double.parseDouble(feesField.getText()); // Parsing fees

            try (Connection conn = getConnection()) {
                String query = "INSERT INTO students (name, age, contact_number, fees) VALUES (?, ?, ?,
?)";
                try (PreparedStatement ps = conn.prepareStatement(query)) {
                    ps.setString(1, name);
                    ps.setInt(2, age);
                    ps.setString(3, contact);
                    ps.setDouble(4, fees); // Inserting fees
                    ps.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Student added successfully!");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Add teacher dialog
    private void addTeacher() {
        JTextField nameField = new JTextField(20);
        JTextField subjectField = new JTextField(20);
        JTextField contactField = new JTextField(20);

        JPanel panel = new JPanel(new GridLayout(3, 2));
        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Subject:"));
        panel.add(subjectField);
        panel.add(new JLabel("Contact Number:"));
        panel.add(contactField);

        int option = JOptionPane.showConfirmDialog(this, panel, "Add New Teacher",
JOptionPane.OK_CANCEL_OPTION);

        if (option == JOptionPane.OK_OPTION) {
            String name = nameField.getText();
            String subject = subjectField.getText();
            String contact = contactField.getText();

            try (Connection conn = getConnection()) {
                String query = "INSERT INTO teachers (name, subject, contact_number) VALUES (?, ?, ?)";
                try (PreparedStatement ps = conn.prepareStatement(query)) {
                    ps.setString(1, name);
                    ps.setString(2, subject);
                    ps.setString(3, contact);
                    ps.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Teacher added successfully!");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Assign teacher to student
    private void assignTeacher() {
        String[] students = getStudentNames();
        String[] teachers = getTeacherNames();

        if (students != null && teachers != null) {
            String studentName = (String) JOptionPane.showInputDialog(this, "Select Student", "Assign
Teacher",
                    JOptionPane.QUESTION_MESSAGE, null, students, students[0]);

            String teacherName = (String) JOptionPane.showInputDialog(this, "Select Teacher", "Assign
Teacher",
                    JOptionPane.QUESTION_MESSAGE, null, teachers, teachers[0]);

            if (studentName != null && teacherName != null) {
                try (Connection conn = getConnection()) {
                    int studentId = getStudentId(studentName, conn);
                    int teacherId = getTeacherId(teacherName, conn);

                    String schedule = JOptionPane.showInputDialog(this, "Enter Schedule");

                    String query = "INSERT INTO assignments (student_id, teacher_id, schedule) VALUES (?,
?, ?)";
                    try (PreparedStatement ps = conn.prepareStatement(query)) {
                        ps.setInt(1, studentId);
                        ps.setInt(2, teacherId);
                        ps.setString(3, schedule);
                        ps.executeUpdate();
                        JOptionPane.showMessageDialog(this, "Teacher assigned to student successfully!");
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    // Get list of student names for assignment
    private String[] getStudentNames() {
        try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement("SELECT name FROM students")) {
            ResultSet rs = ps.executeQuery();
            StringBuilder students = new StringBuilder();
            while (rs.next()) {
                students.append(rs.getString("name")).append(",");
            }
            return students.toString().split(",");
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Get list of teacher names for assignment
    private String[] getTeacherNames() {
        try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement("SELECT name FROM teachers")) {
            ResultSet rs = ps.executeQuery();
            StringBuilder teachers = new StringBuilder();
            while (rs.next()) {
                teachers.append(rs.getString("name")).append(",");
            }
            return teachers.toString().split(",");
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Get student ID by name
    private int getStudentId(String name, Connection conn) throws SQLException {
        String query = "SELECT id FROM students WHERE name = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setString(1, name);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return rs.getInt("id");
            }
        }
        return -1;
    }

    // Get teacher ID by name
    private int getTeacherId(String name, Connection conn) throws SQLException {
        String query = "SELECT id FROM teachers WHERE name = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setString(1, name);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
```

```
            return rs.getInt("id");
        }
    }
    return -1;
}

// View full schedule of students
private void viewStudentSchedule() {
    String query = "SELECT students.name AS student_name, teachers.name AS teacher_name,
assignments.schedule " +
                "FROM assignments " +
                "JOIN students ON assignments.student_id = students.id " +
                "JOIN teachers ON assignments.teacher_id = teachers.id";
    try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement(query)) {
        ResultSet rs = ps.executeQuery();

        StringBuilder scheduleList = new StringBuilder();
        while (rs.next()) {
            scheduleList.append("Student: ").append(rs.getString("student_name"))
                    .append(", Teacher: ").append(rs.getString("teacher_name"))
                    .append(", Schedule: ").append(rs.getString("schedule"))
                    .append("\n");
        }
        if (scheduleList.length() > 0) {
            JOptionPane.showMessageDialog(this, scheduleList.toString(), "Student Schedules",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "No schedules found.", "No Data",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// View full schedule of teachers
private void viewTeacherSchedule() {
    String query = "SELECT teachers.name AS teacher_name, students.name AS student_name,
assignments.schedule " +
                "FROM assignments " +
                "JOIN teachers ON assignments.teacher_id = teachers.id " +
                "JOIN students ON assignments.student_id = students.id";
    try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement(query)) {
        ResultSet rs = ps.executeQuery();

        StringBuilder scheduleList = new StringBuilder();
        while (rs.next()) {
            scheduleList.append("Teacher: ").append(rs.getString("teacher_name"))
                    .append(", Student: ").append(rs.getString("student_name"))
                    .append(", Schedule: ").append(rs.getString("schedule"))
                    .append("\n");
        }
        if (scheduleList.length() > 0) {
            JOptionPane.showMessageDialog(this, scheduleList.toString(), "Teacher Schedules",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "No schedules found.", "No Data",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Main method to launch the application
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new HomeTuitionManagementSystem().setVisible(true);
        }
    });
}
}import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class HomeTuitionManagementSystem extends JFrame {

private static final String URL = "jdbc:mysql://localhost:3306/TuitionDB"; // Change database
URL as needed
private static final String USER = "root"; // Change username as needed
private static final String PASSWORD = "Vigshan@2116"; // Change password as needed

// Buttons for the main menu
private JButton addStudentButton, viewStudentsButton, addTeacherButton,
viewTeachersButton, assignTeacherButton;

public HomeTuitionManagementSystem() {
    // Set up frame
    setTitle("Home Tuition Management System");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);

    // Initialize main menu buttons
    addStudentButton = new JButton("Add Student");
    viewStudentsButton = new JButton("View Students");
    addTeacherButton = new JButton("Add Teacher");
    viewTeachersButton = new JButton("View Teachers");
    assignTeacherButton = new JButton("Assign Teacher");

    // Set layout for main panel
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(6, 1, 10, 10));

    panel.add(addStudentButton);
    panel.add(viewStudentsButton);
    panel.add(addTeacherButton);
    panel.add(viewTeachersButton);
    panel.add(assignTeacherButton);

    add(panel);

    // Button event listeners
    addStudentButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            addStudent();
        }
    });

    viewStudentsButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            viewStudentSchedule(); // Show full schedule for students
        }
    });

    addTeacherButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            addTeacher();
        }
    });

    viewTeachersButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            viewTeacherSchedule(); // Show full schedule for teachers
        }
    });

    assignTeacherButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            assignTeacher();
        }
    });
}

// Database connection method
public static Connection getConnection() {
    try {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

// Add student dialog
private void addStudent() {
    JTextField nameField = new JTextField(20);
    JTextField ageField = new JTextField(20);
    JTextField contactField = new JTextField(20);
    JTextField feesField = new JTextField(20); // Fees input field

    JPanel panel = new JPanel(new GridLayout(4, 2));
    panel.add(new JLabel("Name:"));
    panel.add(nameField);
    panel.add(new JLabel("Age:"));
    panel.add(ageField);
    panel.add(new JLabel("Contact Number:"));
    panel.add(contactField);
    panel.add(new JLabel("Fees:"));
    panel.add(feesField);
    int option = JOptionPane.showConfirmDialog(this, panel, "Add New Student",
JOptionPane.OK_CANCEL_OPTION);

    if (option == JOptionPane.OK_OPTION) {
        String name = nameField.getText();
        int age = Integer.parseInt(ageField.getText());
        String contact = contactField.getText();
        double fees = Double.parseDouble(feesField.getText()); // Parsing fees

        try (Connection conn = getConnection()) {
            String query = "INSERT INTO students (name, age, contact_number, fees) VALUES (?, ?, ?,
?)";
            try (PreparedStatement ps = conn.prepareStatement(query)) {
                ps.setString(1, name);
                ps.setInt(2, age);
                ps.setString(3, contact);
                ps.setDouble(4, fees); // Inserting fees
                ps.executeUpdate();
                JOptionPane.showMessageDialog(this, "Student added successfully!");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

// Add teacher dialog
private void addTeacher() {
    JTextField nameField = new JTextField(20);
    JTextField subjectField = new JTextField(20);
    JTextField contactField = new JTextField(20);

    JPanel panel = new JPanel(new GridLayout(3, 2));
    panel.add(new JLabel("Name:"));
    panel.add(nameField);
    panel.add(new JLabel("Subject:"));
    panel.add(subjectField);
    panel.add(new JLabel("Contact Number:"));
    panel.add(contactField);

    int option = JOptionPane.showConfirmDialog(this, panel, "Add New Teacher",
JOptionPane.OK_CANCEL_OPTION);

    if (option == JOptionPane.OK_OPTION) {
        String name = nameField.getText();
        String subject = subjectField.getText();
        String contact = contactField.getText();

        try (Connection conn = getConnection()) {
            String query = "INSERT INTO teachers (name, subject, contact_number) VALUES (?, ?, ?)";
            try (PreparedStatement ps = conn.prepareStatement(query)) {
                ps.setString(1, name);
                ps.setString(2, subject);
                ps.setString(3, contact);
                ps.executeUpdate();
                JOptionPane.showMessageDialog(this, "Teacher added successfully!");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

// Assign teacher to student
private void assignTeacher() {
    String[] students = getStudentNames();
    String[] teachers = getTeacherNames();

    if (students != null && teachers != null) {
        String studentName = (String) JOptionPane.showInputDialog(this, "Select Student", "Assign
Teacher",
                JOptionPane.QUESTION_MESSAGE, null, students, students[0]);

        String teacherName = (String) JOptionPane.showInputDialog(this, "Select Teacher", "Assign
Teacher",
                JOptionPane.QUESTION_MESSAGE, null, teachers, teachers[0]);

        if (studentName != null && teacherName != null) {
            try (Connection conn = getConnection()) {
                int studentId = getStudentId(studentName, conn);
                int teacherId = getTeacherId(teacherName, conn);

                String schedule = JOptionPane.showInputDialog(this, "Enter Schedule");

                String query = "INSERT INTO assignments (student_id, teacher_id, schedule) VALUES (?,
?, ?)";
                try (PreparedStatement ps = conn.prepareStatement(query)) {
                    ps.setInt(1, studentId);
                    ps.setInt(2, teacherId);
                    ps.setString(3, schedule);
                    ps.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Teacher assigned to student successfully!");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

// Get list of student names for assignment
private String[] getStudentNames() {
    try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement("SELECT name FROM students")) {
        ResultSet rs = ps.executeQuery();
        StringBuilder students = new StringBuilder();
        while (rs.next()) {
            students.append(rs.getString("name")).append(",");
        }
        return students.toString().split(",");
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

// Get list of teacher names for assignment
private String[] getTeacherNames() {
    try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement("SELECT name FROM teachers")) {
        ResultSet rs = ps.executeQuery();
        StringBuilder teachers = new StringBuilder();
        while (rs.next()) {
            teachers.append(rs.getString("name")).append(",");
        }
        return teachers.toString().split(",");
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

// Get student ID by name
private int getStudentId(String name, Connection conn) throws SQLException {
    String query = "SELECT id FROM students WHERE name = ?";
    try (PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setString(1, name);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return rs.getInt("id");
        }
    }
    return -1;
}

// Get teacher ID by name
private int getTeacherId(String name, Connection conn) throws SQLException {
    String query = "SELECT id FROM teachers WHERE name = ?";
    try (PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setString(1, name);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return rs.getInt("id");
        }
    }
    return -1;
}

// View full schedule of students
private void viewStudentSchedule() {
    String query = "SELECT students.name AS student_name, teachers.name AS teacher_name,
assignments.schedule " +
                       "FROM assignments " +
                       "JOIN students ON assignments.student_id = students.id " +
                       "JOIN teachers ON assignments.teacher_id = teachers.id";
    try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement(query)) {
        ResultSet rs = ps.executeQuery();

        StringBuilder scheduleList = new StringBuilder();
        while (rs.next()) {
            scheduleList.append("Student: ").append(rs.getString("student_name"))
                    .append(", Teacher: ").append(rs.getString("teacher_name"))
                    .append(", Schedule: ").append(rs.getString("schedule"))
                    .append("\n");
        }
        if (scheduleList.length() > 0) {
```

```
            JOptionPane.showMessageDialog(this, scheduleList.toString(), "Student Schedules",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "No schedules found.", "No Data",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }

  // View full schedule of teachers
  private void viewTeacherSchedule() {
    String query = "SELECT teachers.name AS teacher_name, students.name AS student_name,
assignments.schedule " +
                "FROM assignments " +
                "JOIN teachers ON assignments.teacher_id = teachers.id " +
                "JOIN students ON assignments.student_id = students.id";
    try (Connection conn = getConnection(); PreparedStatement ps =
conn.prepareStatement(query)) {
        ResultSet rs = ps.executeQuery();

        StringBuilder scheduleList = new StringBuilder();
        while (rs.next()) {
            scheduleList.append("Teacher: ").append(rs.getString("teacher_name"))
                    .append(", Student: ").append(rs.getString("student_name"))
                    .append(", Schedule: ").append(rs.getString("schedule"))
                    .append("\n");
        }
        if (scheduleList.length() > 0) {
            JOptionPane.showMessageDialog(this, scheduleList.toString(), "Teacher Schedules",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "No schedules found.", "No Data",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }

  // Main method to launch the application
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new HomeTuitionManagementSystem().setVisible(true);
        }
    });
  }
}
```