JAVA MINI PROJECT

University Domitory Management

DESIGN (5)	CODE (5)	SCREENSHOT (5)	TIME MANAGEMENT (5)	TOTAL (20)

Harshini M - 2212030

JAISHREE R– 2212045

SWETHA G - 2212050

Project Description:

The **University Dormitory Management System** is designed to manage and organize student information for university dormitories, focusing on both local and international students. This system provides an efficient way to store, manage, and retrieve student details, ensuring smooth administration of dormitory allocations and student records. The system is built using Java Swing for the front-end user interface, with Java Derby as the database for storing and retrieving student data.

Key Features:

1. Student Information Management

The system enables efficient management of detailed records for both local and international students. Administrators can store essential personal information such as name, gender, phone number, and university details, along with dormitory-specific data like room allocation. This functionality ensures that all necessary student information is organized and accessible within the system.

2. Tabbed Interface for Local and International Students

The user interface incorporates a JTabbedPane, which separates students into two categories: Local Students and International Students. Users can easily switch between these tabs to add, view, or search for student records based on their category. This feature enhances usability by simplifying student management and organizing data based on student types.

3. Search and Data Persistence

The system allows administrators to search for students by their phone number, quickly retrieving all relevant details in a table format. This search feature is essential for fast access to student information. Additionally, the system uses Java Derby as its database, ensuring data persistence. All student records are stored securely and can be retrieved at any time, even after the application is closed, providing reliability and consistency in data management.

System Requirements:

- JDK Version: JDK 17
- Database: Java Derby
- IDE: Netbeans
- User Interface : Java swing

DATABASE DESIGN:

Java Derby:

Java Derby (Apache Derby) is a lightweight, embedded relational database management system written in Java, designed for easy integration with Java applications, providing reliable data storage and access without requiring a separate database server.

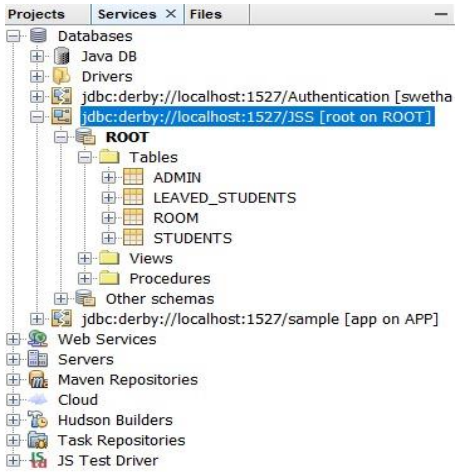


FIG : DataBase Creation

1. **Admin**

   This table stores admin login credentials, where id is the primary key, and username and password are required fields.

   Query :

```
CREATE TABLE admin_login (
    id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

   Insert the values for Admin

   Query:

```
INSERT INTO admin (username, password)
VALUES ('swetha', 's123'), ('jaiz', 'j123');
```

#	ID	USERNAME	PASSWORD
1	1	swetha	s123
2	2	jaiz	j123

FIG : Admin Table

2. **Room**

   This table stores room details, including room number, whether the room is active, and the current status of the room.

   Query:

```
CREATE TABLE Room (
    room_number VARCHAR(10) PRIMARY KEY,
    active VARCHAR(3) CHECK (active IN ('Yes', 'No')),
    room_status VARCHAR(20) CHECK (room_status IN ('Available', 'Booked',
'Under Maintenance'))
);
```

Page Size: 20 Total Rows: 3 Page: 1 of 1 Matching Rows:

#	ROOM_NUMBER	ACTIVE	ROOM_STATUS
1	5	Yes	Booked
2	4	<NULL>	Available
3	7	Yes	Available

FIG : Room Table

3. **Student**

   This table stores detailed student information for both local and international students. It includes personal details, university details, room number, and contact information.

   Query:

```
CREATE TABLE Students (
    name VARCHAR(100) NOT NULL,
    father_name VARCHAR(100),
    gender VARCHAR(10) CHECK (gender IN ('Male', 'Female')) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    university_name VARCHAR(100) NOT NULL,
    degree_program VARCHAR(20) CHECK (degree_program IN ('Bachelor
Program', 'Master Program', 'PhD Program')) NOT NULL,
    room_number VARCHAR(10) NOT NULL,
    dob DATE NOT NULL,
    mother_name VARCHAR(100),
    email VARCHAR(100) NOT NULL,
    university_id VARCHAR(20) NOT NULL,
    address VARCHAR(255)
);
```

#	NAME	FATHER_NAME	GENDER	PHONE_NUMBER	UNIVERSITY_NAME	DEGREE_PROGRAM	ROOM_NUMBER	DATE_OF_BIRTH
1			Female		University A	Bachelor Program	1	2005-05-12
2	karthi	ganesh	Male	965874123	University A	Bachelor Program	2	2004-04-12
3	swe	sads	Female	6565656	University A	Bachelor Program	5	2004-12-05
4	hdh	hdhb	Female	822	University A	Bachelor Program	2	2005-05-05
5	ddsa	ddjbd	Male	555454	University A	Bachelor Program	4	2004-05-05

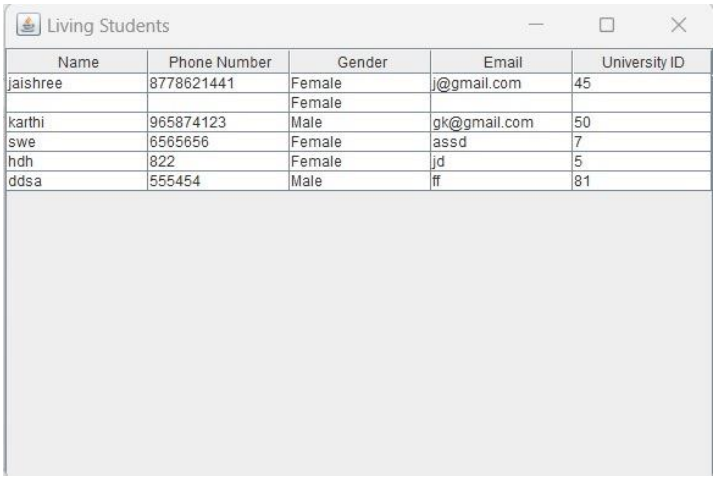
FIG: Student Table

4. **Living students**

   Query:

```
CREATE TABLE Living Students (
    name VARCHAR(100) NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    gender VARCHAR(10) NOT NULL CHECK (gender IN ('Male', 'Female')),
    email VARCHAR(100) NOT NULL,
    university VARCHAR(100) NOT NULL,
```

```
    deletion_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (phone_number)  -- or use a combination of fields that
uniquely identify a record
);
```



Living Students

Name	Phone Number	Gender	Email	University ID
jaishree	8778621441	Female	j@gmail.com	45
		Female		
karthi	965874123	Male	gk@gmail.com	50
swe	6565656	Female	assd	7
hdh	822	Female	jd	5
ddsa	555454	Male	ff	81

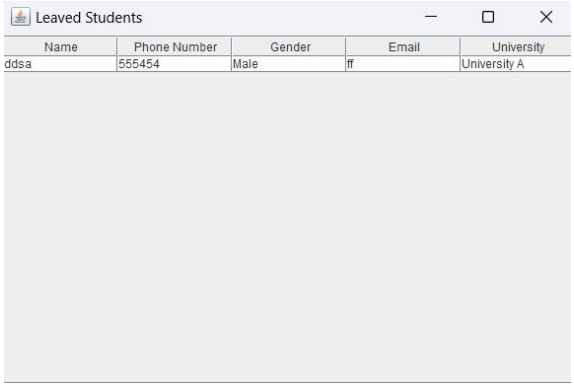
FIG : Living Students

5. **Leaved Students**

   This table stores records of students who have left the dormitory, along with the date of their departure. The phone_number serves as the primary key.

   Query :

```
CREATE TABLE leaved_students (
    name VARCHAR(100) NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    gender VARCHAR(10) NOT NULL CHECK (gender IN ('Male', 'Female')),
    email VARCHAR(100) NOT NULL,
    university VARCHAR(100) NOT NULL,
    deletion_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (phone_number)  -- or use a combination of fields that
uniquely identify a record
);
```



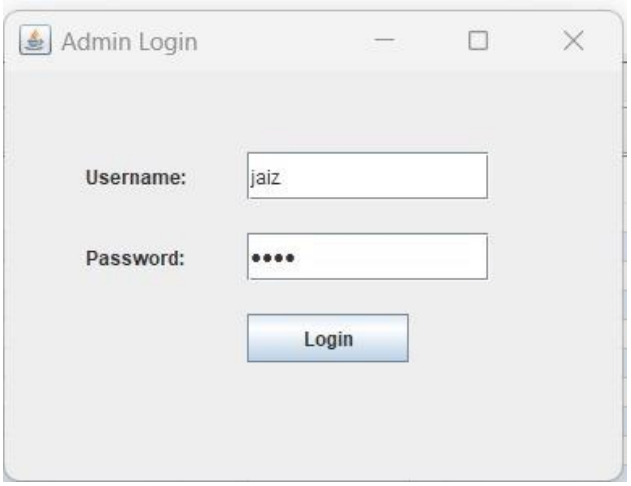
Leaved Students

Name	Phone Number	Gender	Email	University
ddsa	555454	Male	ff	University A

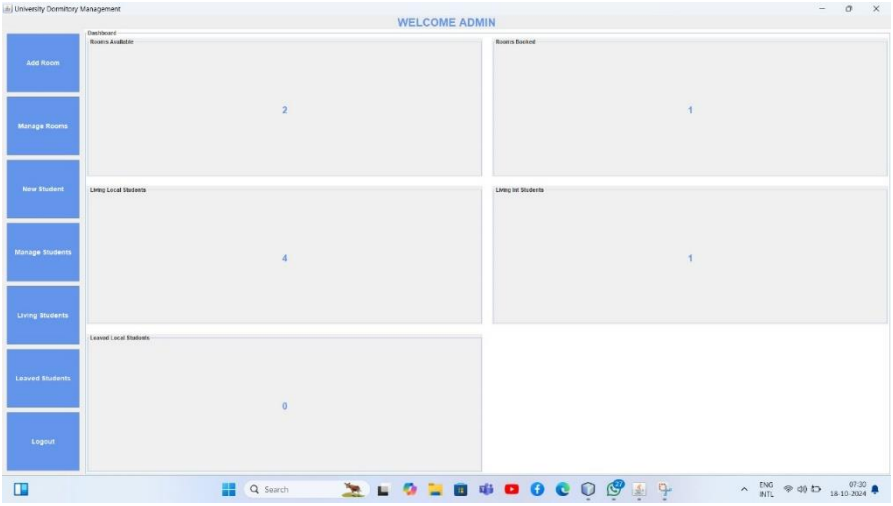
FIG: Leaved Students

GUI Design:

1. Admin Login Page



2. WelcomeAdmin Page



3. Add Room page

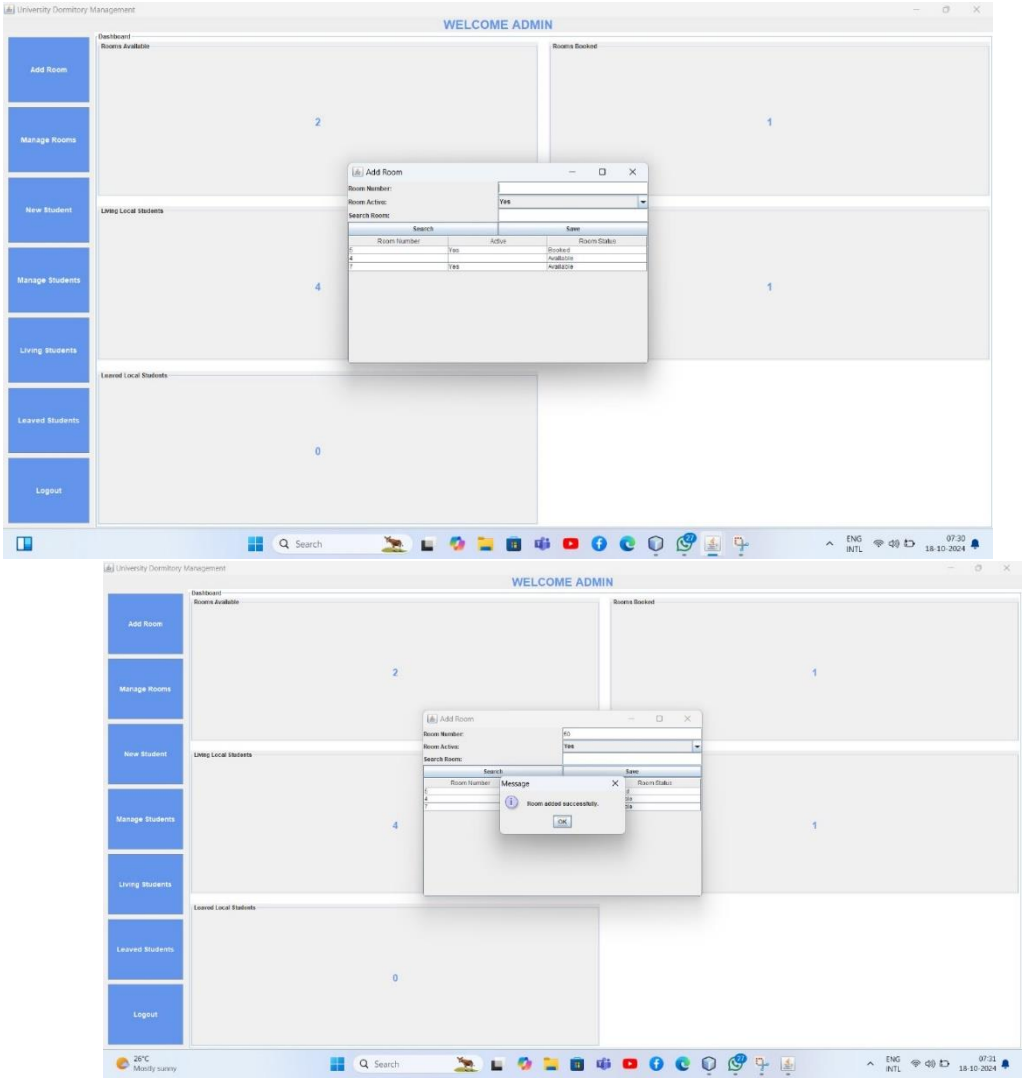
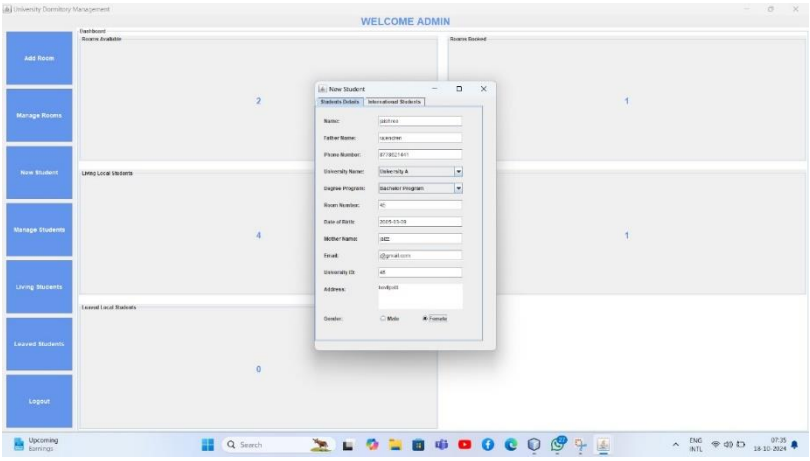
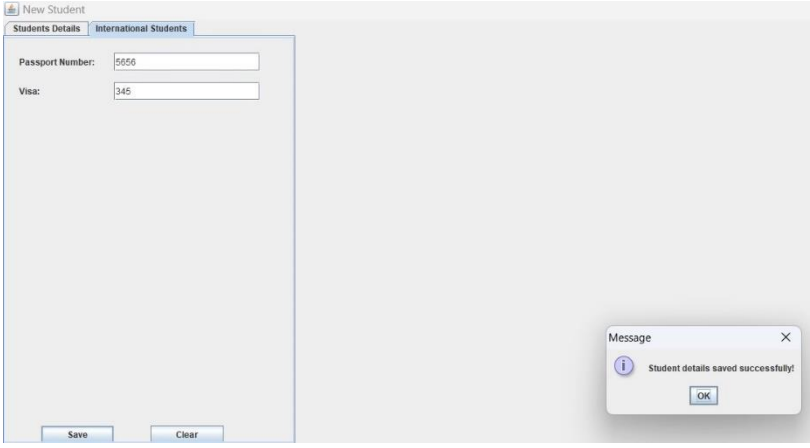


Fig : Room Saved Successfully

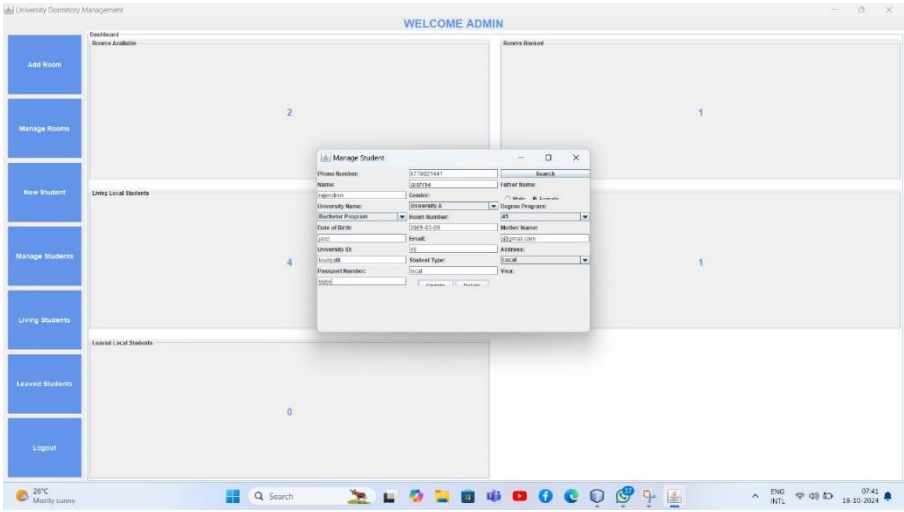
4. New Local Student Entry

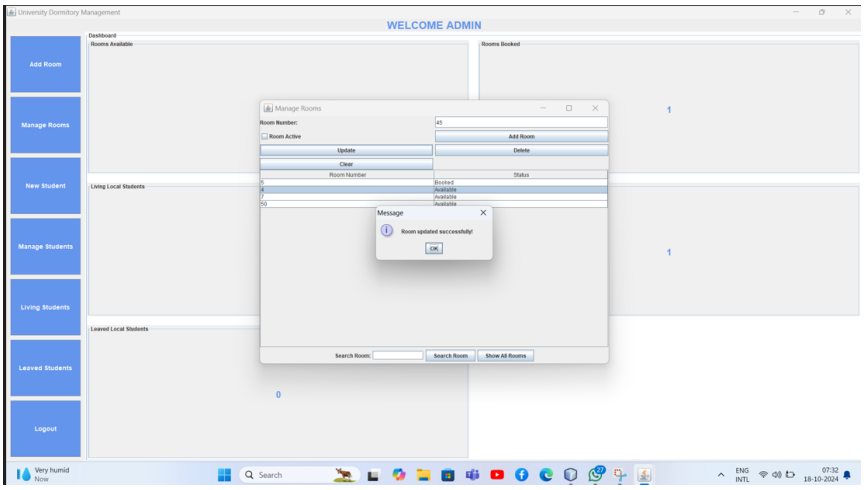


5. International student details

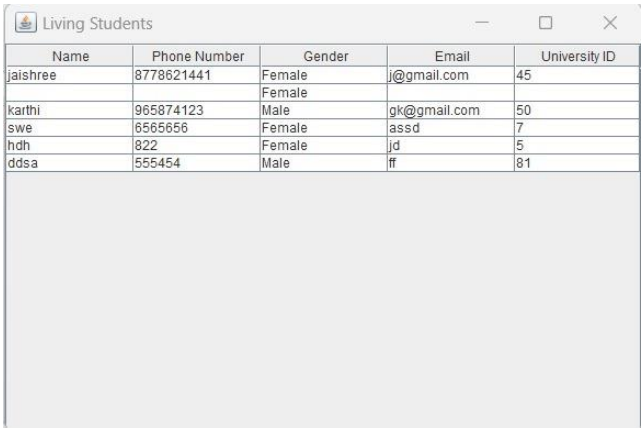


6. ManageStudent page



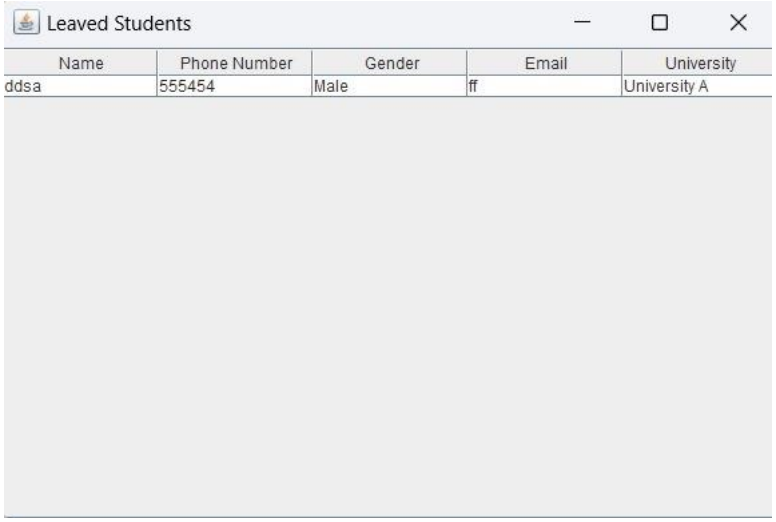


7. **LivingStudents display**



Name	Phone Number	Gender	Email	University ID
jaishree	8778621441	Female	j@gmail.com	45
		Female		
karthi	965874123	Male	gk@gmail.com	50
swe	6565656	Female	assd	7
hdh	822	Female	jd	5
ddsa	555454	Male	ff	81

8. **LeavedStudents Display**



Name	Phone Number	Gender	Email	University
ddsa	555454	Male	ff	University A

CODE:

1. **LoginForm.java**

```
package university;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.*;
public class LoginForm extends JFrame {
    JLabel userLabel, passLabel;
    JTextField userField;
    JPasswordField passField;
    JButton loginBtn;
    public LoginForm() {
```

```
        setLayout(null);
        setTitle("Admin Login");
        userLabel = new JLabel("Username:");
        passLabel = new JLabel("Password:");
        userField = new JTextField();
        passField = new JPasswordField();
        loginBtn = new JButton("Login");
        userLabel.setBounds(50, 50, 100, 30);
        passLabel.setBounds(50, 100, 100, 30);
        userField.setBounds(150, 50, 150, 30);
        passField.setBounds(150, 100, 150, 30);
        loginBtn.setBounds(150, 150, 100, 30);
        add(userLabel);
        add(passLabel);
        add(userField);
        add(passField);
        add(loginBtn);
        loginBtn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                adminLogin();
            }
        });
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    private void adminLogin() {
        String username = userField.getText();
        String password = new String(passField.getPassword());
        try (Connection con = MyConnection.getConnection()) {
            String query = "SELECT * FROM admin WHERE username = ? AND
password = ?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, username);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                JOptionPane.showMessageDialog(null, "Login successful!");
                new Home().setVisible(true);
                this.dispose();
            } else {
                JOptionPane.showMessageDialog(null, "Invalid credentials!");
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
    public static void main(String[] args) {
        new LoginForm().setVisible(true);
    }
}
```

2. **Home.java**

```
package university;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class Home extends JFrame {
    private JLabel roomsAvailableLabel;
    private JLabel roomsBookedLabel;
    private JLabel livingLocalStudentsLabel;
    private JLabel livingIntStudentsLabel;
    private JLabel leavedLocalStudentsLabel;
    public Home() {
        initComponents();
        loadDashboardValues(); // Load values from the database
    }
    // Initialize components and set up the UI
    private void initComponents() {
        // Frame settings
        setTitle("University Dormitory Management");
        setSize(1200, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
        // Create the left panel for navigation
        JPanel leftPanel = new JPanel();
        leftPanel.setLayout(new GridLayout(7, 1, 10, 10)); // Adjusted number of rows
and added gaps
        leftPanel.setBackground(new Color(240, 240, 240)); // Light gray background
        leftPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Add
padding
        // Adding navigation buttons
        String[] buttonLabels = {
            "Add Room", "Manage Rooms", "New Student", "Manage Students",
            "Living Students", "Leaved Students", "Logout"
```

```
        };
        for (String label : buttonLabels) {
            JButton button = new JButton(label);
            button.setFont(new Font("Arial", Font.BOLD, 14));
            button.setBackground(new Color(100, 149, 237)); // Cornflower blue
            button.setForeground(Color.WHITE); // White text
            button.setFocusPainted(false); // Remove focus outline
            button.addActionListener(new ButtonClickListener(label));
            leftPanel.add(button);
        }
        // Create the dashboard panel
        JPanel dashboardPanel = new JPanel();
        dashboardPanel.setLayout(new GridLayout(3, 2, 20, 20)); // Added gaps between
components
        dashboardPanel.setBackground(Color.WHITE);
        dashboardPanel.setBorder(BorderFactory.createTitledBorder("Dashboard"));
        // Dashboard titles and labels
        String[] dashboardTitles = {
            "Rooms Available", "Rooms Booked", "Living Local Students",
            "Living Int Students", "Leaved Local Students"
        };
        // Initialize labels to display dashboard values
        roomsAvailableLabel = createDashboardLabel();
        roomsBookedLabel = createDashboardLabel();
        livingLocalStudentsLabel = createDashboardLabel();
        livingIntStudentsLabel = createDashboardLabel();
        leavedLocalStudentsLabel = createDashboardLabel();
        // Add dashboard components
        dashboardPanel.add(createDashboardPanel(dashboardTitles[0],
roomsAvailableLabel));
        dashboardPanel.add(createDashboardPanel(dashboardTitles[1],
roomsBookedLabel));
        dashboardPanel.add(createDashboardPanel(dashboardTitles[2],
livingLocalStudentsLabel));
        dashboardPanel.add(createDashboardPanel(dashboardTitles[3],
livingIntStudentsLabel));
        dashboardPanel.add(createDashboardPanel(dashboardTitles[4],
leavedLocalStudentsLabel));
        // Add panels to the main frame
        add(leftPanel, BorderLayout.WEST);
        add(dashboardPanel, BorderLayout.CENTER);
        // Set the welcome message at the top
        JLabel welcomeLabel = new JLabel("WELCOME ADMIN", JLabel.CENTER);
        welcomeLabel.setFont(new Font("Arial", Font.BOLD, 24));
        welcomeLabel.setForeground(new Color(100, 149, 237)); // Cornflower blue
        add(welcomeLabel, BorderLayout.NORTH);
```

```
        setVisible(true);
    }
    // Create a dashboard label with styling
    private JLabel createDashboardLabel() {
        JLabel label = new JLabel("0", JLabel.CENTER);
        label.setFont(new Font("Arial", Font.BOLD, 20));
        label.setForeground(new Color(100, 149, 237)); // Cornflower blue
        label.setPreferredSize(new Dimension(150, 60)); // Fixed height for alignment
        return label;
    }
    // Create a panel for each dashboard item
    private JPanel createDashboardPanel(String title, JLabel valueLabel) {
        JPanel panel = new JPanel();
        panel.setBorder(BorderFactory.createTitledBorder(title));
        panel.setBackground(new Color(240, 240, 240)); // Light gray background
        panel.setLayout(new BorderLayout());
        panel.add(valueLabel, BorderLayout.CENTER);
        return panel;
    }
    // Load dynamic dashboard values from the database
    private void loadDashboardValues() {
        try (Connection conn = MyConnection.getConnection(); // Replace with your
database connection method
             Statement stmt = conn.createStatement()) {
            // Example queries to fetch the required data
            String roomsAvailableQuery = "SELECT COUNT(*) FROM Room WHERE
room_status = 'Available'";
            String roomsBookedQuery = "SELECT COUNT(*) FROM Room WHERE
room_status = 'Booked'";
            String livingLocalStudentsQuery = "SELECT COUNT(*) FROM students
WHERE student_type = 'Local'";
            String livingIntStudentsQuery = "SELECT COUNT(*) FROM students
WHERE student_type = 'International'";
            String leavedLocalStudentsQuery = "SELECT COUNT(*) FROM
leaved_students";
            // Execute queries and set the dashboard values
            roomsAvailableLabel.setText(getCount(stmt, roomsAvailableQuery));
            roomsBookedLabel.setText(getCount(stmt, roomsBookedQuery));
            livingLocalStudentsLabel.setText(getCount(stmt, livingLocalStudentsQuery));
            livingIntStudentsLabel.setText(getCount(stmt, livingIntStudentsQuery));
            leavedLocalStudentsLabel.setText(getCount(stmt,
leavedLocalStudentsQuery));
        } catch (SQLException e) {
            e.printStackTrace(); // Handle exception appropriately
        }
    }
```

```
// Helper method to execute a query and get the count
private String getCount(Statement stmt, String query) throws SQLException {
    ResultSet rs = stmt.executeQuery(query);
    if (rs.next()) {
        return rs.getString(1); // Get the count from the result set
    }
    return "0"; // Default value if no result
}
// Button click event handler
private class ButtonClickListener implements ActionListener {
    private String label;

    public ButtonClickListener(String label) {
        this.label = label;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        switch (label) {
            case "Add Room":
                new AddRoom().setVisible(true);
                break;
            case "Manage Rooms":
                new ManageRooms().setVisible(true);
                break;
            case "New Student":
                new NewStudent().setVisible(true);
                break;
            case "Manage Students":
                new ManageStudents().setVisible(true);
                break;
            case "Living Students":
                new LivingStudents().setVisible(true);
                break;
            case "Leaved Students":
                new LeavedStudents().setVisible(true);
                break;
            case "Logout":
                dispose(); // Close the current window
                new LoginForm().setVisible(true); // Redirect to the login form
                break;
            default:
                break;
        }
    }
}
// Main method to run the application
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(Home::new);
    }
}
```

3. **AddRoom.java**

```
package university;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.table.DefaultTableModel;
public class AddRoom extends JFrame {
    private JLabel lblRoomNumber, lblRoomActive, lblRoomStatus, lblSearchRoom;
    private JTextField txtRoomNumber, txtSearchRoom;
    private JComboBox<String> cmbRoomActive;
    private JButton btnSave, btnSearch;
    private JTable tblRoomHistory;
    private DefaultTableModel tableModel;
    // Database connection
    MyConnection myConnection = new MyConnection();
    public AddRoom() {
        initComponents();
        loadRoomHistory(); // Load room history when the form opens
    }
    // Initialize GUI components
    private void initComponents() {
        // Labels
        lblRoomNumber = new JLabel("Room Number:");
        lblRoomActive = new JLabel("Room Active:");
        lblRoomStatus = new JLabel("Room Status:");
        lblSearchRoom = new JLabel("Search Room:");
        // Text fields
        txtRoomNumber = new JTextField(15);
        txtSearchRoom = new JTextField(15);
        // Combo box for Active status (Yes/No)
        cmbRoomActive = new JComboBox<>(new String[]{"Yes", "No"});
        // Buttons
        btnSave = new JButton("Save");
        btnSearch = new JButton("Search");
        // Room history table
        String[] columnNames = {"Room Number", "Active", "Room Status"};
        tableModel = new DefaultTableModel(columnNames, 0);
        tblRoomHistory = new JTable(tableModel);
        // Layout for the form
        setLayout(new BorderLayout());
```

```
        JPanel panelForm = new JPanel(new GridLayout(4, 2));
        panelForm.add(lblRoomNumber);
        panelForm.add(txtRoomNumber);
        panelForm.add(lblRoomActive);
        panelForm.add(cmbRoomActive);
        panelForm.add(lblSearchRoom);
        panelForm.add(txtSearchRoom);
        panelForm.add(btnSearch);
        panelForm.add(btnSave);
        // Adding components to the frame
        add(panelForm, BorderLayout.NORTH);
        add(new JScrollPane(tblRoomHistory), BorderLayout.CENTER);
        // Set frame properties
        setTitle("Add Room");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        // Button actions
        btnSave.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveRoom();
            }
        });
        btnSearch.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                searchRoom();
            }
        });
    }
    // Method to save a new room
    private void saveRoom() {
        String roomNumber = txtRoomNumber.getText();
        String roomActive = (String) cmbRoomActive.getSelectedItem();
        if (roomNumber.equals("")) {
            JOptionPane.showMessageDialog(null, "Please enter the room number.");
            return;
        }
        Connection con = myConnection.getConnection();
        String query = "INSERT INTO Room (room_number, active, room_status)
VALUES (?, ?, ?)";
        try {
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, roomNumber);
            ps.setString(2, roomActive);
```

```
            ps.setString(3, "Available"); // Room status default to "Available"
            if (ps.executeUpdate() > 0) {
                JOptionPane.showMessageDialog(null, "Room added successfully.");
                loadRoomHistory(); // Reload room history after adding a new room
                clearFields();
            } else {
                JOptionPane.showMessageDialog(null, "Failed to add room.");
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());
        }
    }
    // Method to search for a room by room number
    private void searchRoom() {
        String searchRoomNumber = txtSearchRoom.getText();
        if (searchRoomNumber.equals("")) {
            JOptionPane.showMessageDialog(null, "Please enter a room number to
search.");
            return;
        }

        Connection con = myConnection.getConnection();
        String query = "SELECT * FROM Room WHERE room_number = ?";
        try {
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, searchRoomNumber);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                txtRoomNumber.setText(rs.getString("room_number"));
                cmbRoomActive.setSelectedItem(rs.getString("active"));
                JOptionPane.showMessageDialog(null, "Room found.");
            } else {
                JOptionPane.showMessageDialog(null, "Room not found.");
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());
        }
    }
    // Method to load room history into the table
    private void loadRoomHistory() {
        tableModel.setRowCount(0); // Clear the table before reloading data
        Connection con = myConnection.getConnection();
        String query = "SELECT * FROM Room";
        try {
            PreparedStatement ps = con.prepareStatement(query);
            ResultSet rs = ps.executeQuery();
```

```
            while (rs.next()) {
                String roomNumber = rs.getString("room_number");
                String active = rs.getString("active");
                String roomStatus = rs.getString("room_status");
                tableModel.addRow(new Object[]{roomNumber, active, roomStatus});
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Error loading room history: " +
ex.getMessage());
        }
    }
    // Method to clear input fields
    private void clearFields() {
        txtRoomNumber.setText("");
        txtSearchRoom.setText("");
        cmbRoomActive.setSelectedIndex(0);
    }
    // Main method to run the AddRoom form
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new AddRoom().setVisible(true);
            }
        });
    }
}
```

4. **ManageRooms.java**

```
package university;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
public class ManageRooms extends JFrame {
    private JTextField txtRoomNumber, txtSearchRoom;
    private JCheckBox checkRoomActive;
    private JButton btnAdd, btnUpdate, btnDelete, btnClear, btnSearch, btnShowAll;
    private JTable tableRooms;
    private DefaultTableModel tableModel;
    public ManageRooms() {
        setTitle("Manage Rooms");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
```

```
        // Create components
        JLabel lblRoomNumber = new JLabel("Room Number:");
        txtRoomNumber = new JTextField(10);
        checkRoomActive = new JCheckBox("Room Active");
        btnAdd = new JButton("Add Room");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnClear = new JButton("Clear");
        btnSearch = new JButton("Search Room");
        btnShowAll = new JButton("Show All Rooms");
        txtSearchRoom = new JTextField(10);
        // Table to display rooms
        tableModel = new DefaultTableModel();
        tableModel.setColumnIdentifiers(new Object[]{"Room Number", "Status"});
        tableRooms = new JTable(tableModel);
        JScrollPane tableScrollPane = new JScrollPane(tableRooms);
        // Layout
        JPanel panel = new JPanel(new GridLayout(4, 2, 5, 5));
        panel.add(lblRoomNumber);
        panel.add(txtRoomNumber);
        panel.add(checkRoomActive);
        panel.add(btnAdd);
        panel.add(btnUpdate);
        panel.add(btnDelete);
        panel.add(btnClear);
        JPanel searchPanel = new JPanel();
        searchPanel.add(new JLabel("Search Room:"));
        searchPanel.add(txtSearchRoom);
        searchPanel.add(btnSearch);
        searchPanel.add(btnShowAll);
        // Add components to frame
        add(panel, BorderLayout.NORTH);
        add(tableScrollPane, BorderLayout.CENTER);
        add(searchPanel, BorderLayout.SOUTH);
        // Event listeners
        btnAdd.addActionListener(e -> addRoom());
        btnUpdate.addActionListener(e -> updateRoom());
        btnDelete.addActionListener(e -> deleteRoom());
        btnClear.addActionListener(e -> clearFields());
        btnSearch.addActionListener(e -> searchRoom());
        btnShowAll.addActionListener(e -> showAllRooms());
        // Load all rooms initially
        showAllRooms();
    }
    private void addRoom() {
        String roomNumber = txtRoomNumber.getText();
```

```
        String status = checkRoomActive.isSelected() ? "Booked" : "Available";
        if (roomNumber.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter room number.");
            return;
        }
        try (Connection con = MyConnection.getConnection()) {
            String query = "INSERT INTO Room (room_number, room_status) VALUES
(?, ?)";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, roomNumber);
            ps.setString(2, status);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Room added successfully!");
            showAllRooms();
            clearFields();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
    private void updateRoom() {
        int selectedRow = tableRooms.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a room to update.");
            return;
        }
        String roomNumber = (String) tableModel.getValueAt(selectedRow, 0);
        String newRoomNumber = txtRoomNumber.getText();
        String status = checkRoomActive.isSelected() ? "Booked" : "Available";
        if (newRoomNumber.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter room number.");
            return;
        }
        try (Connection con = MyConnection.getConnection()) {
            String query = "UPDATE Room SET room_number = ?, room_status = ?
WHERE room_number = ?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, newRoomNumber);
            ps.setString(2, status);
            ps.setString(3, roomNumber);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Room updated successfully!");
            showAllRooms();
            clearFields();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
```

```
    }
    private void deleteRoom() {
        int selectedRow = tableRooms.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a room to delete.");
            return;
        }
        String roomNumber = (String) tableModel.getValueAt(selectedRow, 0);
        String status = (String) tableModel.getValueAt(selectedRow, 1);
        if (status.equals("Booked")) {
            JOptionPane.showMessageDialog(this, "Cannot delete a booked room.");
            return;
        }
        try (Connection con = MyConnection.getConnection()) {
            String query = "DELETE FROM Room WHERE room_number = ?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, roomNumber);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Room deleted successfully!");
            showAllRooms();
            clearFields();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
    private void searchRoom() {
        String roomNumber = txtSearchRoom.getText();
        if (roomNumber.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter a room number to
search.");
            return;
        }
        try (Connection con = MyConnection.getConnection()) {
            String query = "SELECT * FROM Room WHERE room_number = ?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, roomNumber);
            ResultSet rs = ps.executeQuery();
            tableModel.setRowCount(0);
            if (rs.next()) {
                tableModel.addRow(new Object[]{
                        rs.getString("room_number"),
                        rs.getString("room_status")
                });
            } else {
                JOptionPane.showMessageDialog(this, "Room not found.");
            }
```

```
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
    private void showAllRooms() {
        try (Connection con = MyConnection.getConnection()) {
            String query = "SELECT * FROM Room";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            tableModel.setRowCount(0);
            while (rs.next()) {
                tableModel.addRow(new Object[]{
                        rs.getString("room_number"),
                        rs.getString("room_status")
                });
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    private void clearFields() {
        txtRoomNumber.setText("");
        checkRoomActive.setSelected(false);
        txtSearchRoom.setText("");
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ManageRooms().setVisible(true));
    }
}
```

5. **NewStudent.java**

```
package university;
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class NewStudent extends JFrame {
    private JTextField nameField;
    private JTextField fatherNameField;
    private JTextField phoneNumberField;
    private JComboBox<String> universityComboBox;
    private JComboBox<String> degreeProgramComboBox;
```

```
    private JTextField roomNumberField;
    private JTextField dateOfBirthField;
    private JTextField motherNameField;
    private JTextField emailField;
    private JTextField universityIDField;
    private JTextArea addressField;
    private JRadioButton maleRadio;
    private JRadioButton femaleRadio;
    private JButton saveButton;
    private JButton clearButton;
    // C;omponents for International Students
    private JTextField passportNumberField; // Add passport field for international
students
    private JTextField visaField; // Add visa field for international students
    public NewStudent() {
        // Frame setup
        setTitle("New Student");
        setSize(450, 650);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(null);

        // Initialize components
        initializeComponents();
        // Set component bounds
        setComponentBounds();
        // Add action listeners
        addActionListeners();
        // Add components to the frame
        addComponentsToFrame();
    }
    private void initializeComponents() {
        // Common fields
        nameField = new JTextField();
        fatherNameField = new JTextField();
        phoneNumberField = new JTextField();
        universityComboBox = new JComboBox<>(new String[]{"University A",
"University B"});
        degreeProgramComboBox = new JComboBox<>(new String[]{"Bachelor
Program", "Master Program"});
        roomNumberField = new JTextField();
        dateOfBirthField = new JTextField();
        motherNameField = new JTextField();
        emailField = new JTextField();
        universityIDField = new JTextField();
        addressField = new JTextArea();
```

```
        maleRadio = new JRadioButton("Male");
        femaleRadio = new JRadioButton("Female");
        saveButton = new JButton("Save");
        clearButton = new JButton("Clear");
        // Add passport and visa fields for international students
        passportNumberField = new JTextField();
        visaField = new JTextField();
        ButtonGroup genderGroup = new ButtonGroup();
        genderGroup.add(maleRadio);
        genderGroup.add(femaleRadio);
    }
    private void setComponentBounds() {
        // Common fields bounds
        nameField.setBounds(150, 20, 200, 25);
        fatherNameField.setBounds(150, 60, 200, 25);
        phoneNumberField.setBounds(150, 100, 200, 25);
        universityComboBox.setBounds(150, 140, 200, 25);
        degreeProgramComboBox.setBounds(150, 180, 200, 25);
        roomNumberField.setBounds(150, 220, 200, 25);
        dateOfBirthField.setBounds(150, 260, 200, 25);
        motherNameField.setBounds(150, 300, 200, 25);
        emailField.setBounds(150, 340, 200, 25);
        universityIDField.setBounds(150, 380, 200, 25);
        addressField.setBounds(150, 420, 200, 60);
        maleRadio.setBounds(150, 490, 100, 25);
        femaleRadio.setBounds(250, 490, 100, 25);
        saveButton.setBounds(50, 530, 100, 25);
        clearButton.setBounds(200, 530, 100, 25);
        // International student fields
        passportNumberField.setBounds(150, 20, 200, 25);
        visaField.setBounds(150, 60, 200, 25);
    }
    private void addActionListeners() {
        saveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveStudent();
            }
        });
        clearButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                clearFields();
            }
        });
    }
```

```
    private void addComponentsToFrame() {
        // Create a JTabbedPane
        JTabbedPane tabbedPane = new JTabbedPane();
        // Panel for Local Students
        JPanel localPanel = new JPanel();
        localPanel.setLayout(null);
        localPanel.add(nameField);
        localPanel.add(fatherNameField);
        localPanel.add(phoneNumberField);
        localPanel.add(universityComboBox);
        localPanel.add(degreeProgramComboBox);
        localPanel.add(roomNumberField);
        localPanel.add(dateOfBirthField);
        localPanel.add(motherNameField);
        localPanel.add(emailField);
        localPanel.add(universityIDField);
        localPanel.add(addressField);
        localPanel.add(maleRadio);
        localPanel.add(femaleRadio);
        localPanel.add(saveButton);
        localPanel.add(clearButton);

        localPanel.add(new JLabel("Name:")).setBounds(20, 20, 120, 25);
        localPanel.add(new JLabel("Father Name:")).setBounds(20, 60, 120, 25);
        localPanel.add(new JLabel("Phone Number:")).setBounds(20, 100, 120, 25);
        localPanel.add(new JLabel("University Name:")).setBounds(20, 140, 120, 25);
        localPanel.add(new JLabel("Degree Program:")).setBounds(20, 180, 120, 25);
        localPanel.add(new JLabel("Room Number:")).setBounds(20, 220, 120, 25);
        localPanel.add(new JLabel("Date of Birth:")).setBounds(20, 260, 120, 25);
        localPanel.add(new JLabel("Mother Name:")).setBounds(20, 300, 120, 25);
        localPanel.add(new JLabel("Email:")).setBounds(20, 340, 120, 25);
        localPanel.add(new JLabel("University ID:")).setBounds(20, 380, 120, 25);
        localPanel.add(new JLabel("Address:")).setBounds(20, 420, 120, 25);
        localPanel.add(new JLabel("Gender:")).setBounds(20, 490, 120, 25);
        // Panel for International Students
        JPanel internationalPanel = new JPanel();
        internationalPanel.setLayout(null);
        internationalPanel.add(passportNumberField);
        internationalPanel.add(visaField);
        internationalPanel.add(saveButton);
        internationalPanel.add(clearButton);
        internationalPanel.add(new JLabel("Passport Number:")).setBounds(20, 20, 120,
25);
        internationalPanel.add(new JLabel("Visa:")).setBounds(20, 60, 120, 25);
        // Add panels to the tabbed pane
        tabbedPane.addTab("Students Details", localPanel);
```

```
        tabbedPane.addTab("International Students", internationalPanel);
        // Add tabbed pane to frame
        tabbedPane.setBounds(0, 0, 400, 580);
        add(tabbedPane);
    }
    private void saveStudent() {
        String name = nameField.getText();
        String fatherName = fatherNameField.getText();
        String phoneNumber = phoneNumberField.getText();
        String universityName = (String) universityComboBox.getSelectedItem();
        String degreeProgram = (String) degreeProgramComboBox.getSelectedItem();
        int roomNumber = Integer.parseInt(roomNumberField.getText());
        String dateOfBirth = dateOfBirthField.getText(); // Ensure format is YYYY-
MM-DD
        String motherName = motherNameField.getText();
        String email = emailField.getText();
        String universityID = universityIDField.getText();
        String address = addressField.getText();
        String gender = maleRadio.isSelected() ? "Male" : "Female";
        String studentType = "Local"; // Default student type

        // Determine if it's an international student
        String passportNumber = passportNumberField.getText();
        String visa = visaField.getText();
        // If it's an international student, adjust student type accordingly
        if (!passportNumber.isEmpty() && !visa.isEmpty()) {
            studentType = "International";
            // Add international student fields to the query
        }
        String query = "INSERT INTO students (name, father_name, gender,
phone_number, university_name, "
                + "degree_program, room_number, date_of_birth, mother_name, email,
"
                + "university_id, address, student_type) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?)";
        try (Connection conn =
DriverManager.getConnection("jdbc:derby://localhost:1527/JSS", "root", "123");
             PreparedStatement pstmt = conn.prepareStatement(query)) {
            pstmt.setString(1, name);
            pstmt.setString(2, fatherName);
            pstmt.setString(3, gender);
            pstmt.setString(4, phoneNumber);
            pstmt.setString(5, universityName);
            pstmt.setString(6, degreeProgram);
            pstmt.setInt(7, roomNumber);
            pstmt.setString(8, dateOfBirth);
```

```
            pstmt.setString(9, motherName);
            pstmt.setString(10, email);
            pstmt.setString(11, universityID);
            pstmt.setString(12, address);
            pstmt.setString(13, studentType); // Add student type to database
            pstmt.executeUpdate();
            JOptionPane.showMessageDialog(this, "Student details saved successfully!");
            clearFields(); // Clear fields after saving
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error saving student details: " +
e.getMessage());
        }
    }

    private void clearFields() {
        nameField.setText("");
        fatherNameField.setText("");
        phoneNumberField.setText("");
        universityComboBox.setSelectedIndex(0);
        degreeProgramComboBox.setSelectedIndex(0);
        roomNumberField.setText("");
        dateOfBirthField.setText("");
        motherNameField.setText("");
        emailField.setText("");
        universityIDField.setText("");
        addressField.setText("");
        maleRadio.setSelected(true);
        passportNumberField.setText("");
        visaField.setText("");
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new NewStudent().setVisible(true));
    }
}
```

6. **ManageStudents.java**

```
package university;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class ManageStudents extends JFrame {
    // Components for student details
    private JTextField txtName, txtFatherName, txtPhoneNumber, txtDOB,
txtMotherName, txtEmail, txtUniversityID, txtAddress;
```

```
    private JComboBox<String> cmbDegreeProgram, cmbRoomNumber,
cmbUniversityName, cmbStudentType;
    private JTextField txtPassportNumber, txtVisa; // Fields for international students
    private JRadioButton rbtnMale, rbtnFemale;
    private JButton btnSearch, btnUpdate, btnDelete, btnClear;
    private ButtonGroup genderGroup;
    // Database connection
    MyConnection myConnection = new MyConnection();
    public ManageStudents() {
        initComponents();
    }
    // Initialize components for the form
    private void initComponents() {
        setLayout(new GridLayout(15, 2, 5, 5)); // Adjusted for student type and
international fields
        // Labels and Input fields
        JLabel lblName = new JLabel("Name:");
        JLabel lblFatherName = new JLabel("Father Name:");
        JLabel lblGender = new JLabel("Gender:");
        JLabel lblPhoneNumber = new JLabel("Phone Number:");
        JLabel lblUniversityName = new JLabel("University Name:");
        JLabel lblDegreeProgram = new JLabel("Degree Program:");
        JLabel lblRoomNumber = new JLabel("Room Number:");
        JLabel lblDOB = new JLabel("Date of Birth:");
        JLabel lblMotherName = new JLabel("Mother Name:");
        JLabel lblEmail = new JLabel("Email:");
        JLabel lblUniversityID = new JLabel("University ID:");
        JLabel lblAddress = new JLabel("Address:");
        JLabel lblStudentType = new JLabel("Student Type:"); // Added label for student
type
        JLabel lblPassportNumber = new JLabel("Passport Number:"); // International
student
        JLabel lblVisa = new JLabel("Visa:"); // International student
        // Text fields
        txtName = new JTextField(15);
        txtFatherName = new JTextField(15);
        txtPhoneNumber = new JTextField(15);
        txtDOB = new JTextField(15);
        txtMotherName = new JTextField(15);
        txtEmail = new JTextField(15);
        txtUniversityID = new JTextField(15);
        txtAddress = new JTextField(15);
        txtPassportNumber = new JTextField(15); // Added for international students
        txtVisa = new JTextField(15); // Added for international students
        // Combo box for degree programs
```

```
        cmbDegreeProgram = new JComboBox<>(new String[]{"Bachelor Program",
"Master Program", "PhD Program"});
        // Combo box for room numbers (example: 1-100)
        cmbRoomNumber = new JComboBox<>();
        for (int i = 1; i <= 100; i++) {
            cmbRoomNumber.addItem(String.valueOf(i));
        }
        // Combo box for universities
        cmbUniversityName = new JComboBox<>(new String[]{"University A",
"University B", "University C"});
        // Combo box for student types
        cmbStudentType = new JComboBox<>(new String[]{"Local", "International"});
// Added student type options
        // Gender radio buttons
        rbtnMale = new JRadioButton("Male");
        rbtnFemale = new JRadioButton("Female");
        genderGroup = new ButtonGroup();
        genderGroup.add(rbtnMale);
        genderGroup.add(rbtnFemale);
        // Buttons
        btnSearch = new JButton("Search");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnClear = new JButton("Clear");
        // Add components to the frame
        add(lblPhoneNumber);
        add(txtPhoneNumber);
        add(btnSearch);
        add(lblName);
        add(txtName);
        add(lblFatherName);
        add(txtFatherName);
        add(lblGender);
        JPanel genderPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        genderPanel.add(rbtnMale);
        genderPanel.add(rbtnFemale);
        add(genderPanel);
        add(lblUniversityName);
        add(cmbUniversityName);
        add(lblDegreeProgram);
        add(cmbDegreeProgram);
        add(lblRoomNumber);
        add(cmbRoomNumber);
        add(lblDOB);
        add(txtDOB);
        add(lblMotherName);
```

```
add(txtMotherName);
add(lblEmail);
add(txtEmail);
add(lblUniversityID);
add(txtUniversityID);
add(lblAddress);
add(txtAddress);
add(lblStudentType); // Added student type label
add(cmbStudentType); // Added student type combo box
// For international students
add(lblPassportNumber); // Passport label
add(txtPassportNumber); // Passport field
add(lblVisa); // Visa label
add(txtVisa); // Visa field
// Add buttons to the bottom
JPanel buttonPanel = new JPanel();
buttonPanel.add(btnUpdate);
buttonPanel.add(btnDelete);
buttonPanel.add(btnClear);
add(buttonPanel);
// Set frame properties
setTitle("Manage Student");
setSize(600, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLocationRelativeTo(null);
// Button actions
btnSearch.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        searchStudent();
    }
});
btnUpdate.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateStudent();
    }
});
btnDelete.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        deleteStudent();
    }
});
btnClear.addActionListener(new ActionListener() {
    @Override
```

```
            public void actionPerformed(ActionEvent e) {
                clearFields();
            }
        });
    }
    // Method to search for a student by phone number
    private void searchStudent() {
        String phoneNumber = txtPhoneNumber.getText();
        if (phoneNumber.equals("")) {
            JOptionPane.showMessageDialog(null, "Please enter a phone number.");
            return;
        }
        Connection con = myConnection.getConnection();
        String query = "SELECT * FROM students WHERE phone_number = ?";
        try {
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, phoneNumber);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                // Populate fields with student data
                txtName.setText(rs.getString("name"));
                txtFatherName.setText(rs.getString("father_name"));
                String gender = rs.getString("gender");
                if (gender.equals("Male")) {
                    rbtnMale.setSelected(true);
                } else {
                    rbtnFemale.setSelected(true);
                }
                cmbUniversityName.setSelectedItem(rs.getString("university_name"));
                cmbDegreeProgram.setSelectedItem(rs.getString("degree_program"));
                cmbRoomNumber.setSelectedItem(rs.getString("room_number"));
                txtDOB.setText(rs.getString("date_of_birth"));
                txtMotherName.setText(rs.getString("mother_name"));
                txtEmail.setText(rs.getString("email"));
                txtUniversityID.setText(rs.getString("university_id"));
                txtAddress.setText(rs.getString("address"));
                cmbStudentType.setSelectedItem(rs.getString("student_type"));
                // For international students
                if (rs.getString("student_type").equals("International")) {
                    txtPassportNumber.setText(rs.getString("passport_number"));
                    txtVisa.setText(rs.getString("visa"));
                }
            } else {
                JOptionPane.showMessageDialog(null, "Student not found.");
            }
        } catch (SQLException ex) {
```

```
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error while searching: " +
ex.getMessage());
        }
    }
    // Method to update a student's details in the database
    private void updateStudent() {
        String phoneNumber = txtPhoneNumber.getText();
        if (phoneNumber.equals("")) {
            JOptionPane.showMessageDialog(null, "Please enter a phone number to
update.");
            return;
        }
        String name = txtName.getText();
        String fatherName = txtFatherName.getText();
        String gender = rbtnMale.isSelected() ? "Male" : "Female";
        String universityName = (String) cmbUniversityName.getSelectedItem();
        String degreeProgram = (String) cmbDegreeProgram.getSelectedItem();
        String roomNumber = (String) cmbRoomNumber.getSelectedItem();
        String dob = txtDOB.getText();
        String motherName = txtMotherName.getText();
        String email = txtEmail.getText();
        String universityID = txtUniversityID.getText();
        String address = txtAddress.getText();
        String studentType = (String) cmbStudentType.getSelectedItem();
        // For international students
        String passportNumber = studentType.equals("International") ?
txtPassportNumber.getText() : "";
        String visa = studentType.equals("International") ? txtVisa.getText() : "";
        // SQL query to update student details
        String query = "UPDATE students SET name = ?, father_name = ?, gender = ?,
university_name = ?, " +
                "degree_program = ?, room_number = ?, date_of_birth = ?, mother_name =
?, email = ?, " +
                "university_id = ?, address = ?, student_type = ?, passport_number = ?, visa
= ? WHERE phone_number = ?";
        try (Connection con = myConnection.getConnection(); PreparedStatement ps =
con.prepareStatement(query)) {
            ps.setString(1, name);
            ps.setString(2, fatherName);
            ps.setString(3, gender);
            ps.setString(4, universityName);
            ps.setString(5, degreeProgram);
            ps.setString(6, roomNumber);
            ps.setString(7, dob);
            ps.setString(8, motherName);
```

```
            ps.setString(9, email);
            ps.setString(10, universityID);
            ps.setString(11, address);
            ps.setString(12, studentType);
            ps.setString(13, passportNumber);
            ps.setString(14, visa);
            ps.setString(15, phoneNumber);
            int rowsUpdated = ps.executeUpdate();
            if (rowsUpdated > 0) {
                JOptionPane.showMessageDialog(null, "Student details updated
successfully.");
                clearFields();
            } else {
                JOptionPane.showMessageDialog(null, "No student found with that phone
number.");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error while updating student: " +
ex.getMessage());
        }
    }

    // Method to delete a student from the database
  private void deleteStudent() {
    String phoneNumber = txtPhoneNumber.getText();
    if (phoneNumber.equals("")) {
        JOptionPane.showMessageDialog(null, "Please enter a phone number to
delete.");
        return;
    }
    Connection con = myConnection.getConnection();
    try {
        // Fetch the student details before deletion
        String fetchQuery = "SELECT * FROM students WHERE phone_number = ?";
        PreparedStatement fetchPs = con.prepareStatement(fetchQuery);
        fetchPs.setString(1, phoneNumber);
        ResultSet rs = fetchPs.executeQuery();
        if (rs.next()) {
            // Store the student details in variables
            String name = rs.getString("name");
            String gender = rs.getString("gender");
            String email = rs.getString("email");
            String university = rs.getString("university_name");
            // Insert the student details into the leaved_students table
```

```
            String insertLeavedStudent = "INSERT INTO leaved_students (name,
phone_number, gender, email, university) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement insertPs = con.prepareStatement(insertLeavedStudent);
            insertPs.setString(1, name);
            insertPs.setString(2, phoneNumber);
            insertPs.setString(3, gender);
            insertPs.setString(4, email);
            insertPs.setString(5, university);
            insertPs.executeUpdate();
            // Now delete the student from the students table
            String deleteQuery = "DELETE FROM students WHERE phone_number =
?";
            PreparedStatement deletePs = con.prepareStatement(deleteQuery);
            deletePs.setString(1, phoneNumber);
            deletePs.executeUpdate();
            JOptionPane.showMessageDialog(null, "Student deleted successfully and
moved to leaved students.");
            clearFields();
        } else {
            JOptionPane.showMessageDialog(null, "Student not found.");
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error while deleting student: " +
ex.getMessage());
    }
}
    // Method to clear all fields
    private void clearFields() {
        txtName.setText("");
        txtFatherName.setText("");
        txtPhoneNumber.setText("");
        txtDOB.setText("");
        txtMotherName.setText("");
        txtEmail.setText("");
        txtUniversityID.setText("");
        txtAddress.setText("");
        genderGroup.clearSelection();
        cmbDegreeProgram.setSelectedIndex(0);
        cmbRoomNumber.setSelectedIndex(0);
        cmbUniversityName.setSelectedIndex(0);
        cmbStudentType.setSelectedIndex(0);
        txtPassportNumber.setText("");
        txtVisa.setText("");
    }
    public static void main(String[] args) {
```

```
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new ManageStudents().setVisible(true);
            }
        });
    }
}
```

7. **LivingStudents.java**

```
package university;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;
public class LivingStudents extends JFrame {
    private JTable table;
    private DefaultTableModel tableModel;
    // Database connection
    MyConnection myConnection = new MyConnection();
    public LivingStudents() {
        setTitle("Living Students");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        // Create table model and JTable
        String[] columnNames = {"Name", "Phone Number", "Gender", "Email",
"University ID"};
        tableModel = new DefaultTableModel(columnNames, 0);
        table = new JTable(tableModel);
      // Load data from database
        loadLivingStudents();
        // Set up the layout
        setLayout(new BorderLayout());
        add(new JScrollPane(table), BorderLayout.CENTER);
    }
    private void loadLivingStudents() {
        String query = "SELECT name, phone_number, gender, email, university_id
FROM students";
        try (Connection con = myConnection.getConnection();
             Statement stmt = con.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            // Clear previous data
            tableModel.setRowCount(0);
            // Populate table model with data
            while (rs.next()) {
```

```
                String name = rs.getString("name");
                String phoneNumber = rs.getString("phone_number");
                String gender = rs.getString("gender");
                String email = rs.getString("email");
                String universityId = rs.getString("university_id");
                // Add a row to the table model
                tableModel.addRow(new Object[]{name, phoneNumber, gender, email,
universityId});
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error loading data: " +
ex.getMessage());
        }
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new LivingStudents().setVisible(true));
    }
}
```

8. **LeavedStudents.java**

```
package university;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;
public class LeavedStudents extends JFrame {
    private JTable table;
    private DefaultTableModel tableModel;
    // Database connection
    MyConnection myConnection = new MyConnection();
    public LeavedStudents() {
        setTitle("Leaved Students");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        // Create table model and JTable
        String[] columnNames = {"Name", "Phone Number", "Gender", "Email",
"University"};
        tableModel = new DefaultTableModel(columnNames, 0);
        table = new JTable(tableModel);
        // Load data from database
        loadLeavedStudents();
        // Set up the layout
        setLayout(new BorderLayout());
```

```
        add(new JScrollPane(table), BorderLayout.CENTER);
    }
    private void loadLeavedStudents() {
        String query = "SELECT name, phone_number, gender, email, university FROM
leaved_students";
        try (Connection con = myConnection.getConnection();
             Statement stmt = con.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            // Clear previous data
            tableModel.setRowCount(0);
            // Populate table model with data
            while (rs.next()) {
                String name = rs.getString("name");
                String phoneNumber = rs.getString("phone_number");
                String gender = rs.getString("gender");
                String email = rs.getString("email");
                String university = rs.getString("university");
                // Add a row to the table model
                tableModel.addRow(new Object[]{name, phoneNumber, gender, email,
university});
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error loading data: " +
ex.getMessage());
        }
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new LeavedStudents().setVisible(true));
    }
}
```

9. **MyConnection.java**

```
package university;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class MyConnection {
    public static Connection getConnection() {
        Connection connection = null;
        try {
            // Load Derby JDBC driver
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            // Connect to Derby database
            connection =
DriverManager.getConnection("jdbc:derby://localhost:1527/JSS", "root", "123");
```

```
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

Conclusion:

The University Dormitory Management System provides a streamlined solution for managing student information in university dormitories. By utilizing Java Derby for data persistence and Java Swing for the user interface, this system offers an easy-to-use platform for dormitory administrators to manage and access student details efficiently, whether for local or international students.