University Management

## Project submitted to the

SRM University – AP, Andhra Pradesh

## for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

## In

**Computer Science and Engineering School of Engineering and Sciences** Submitted by

Kakumanu Karthik-AP21110010263

Kondeti Lakshya-AP21110010265

Sakkurthi Sashank -AP21110010272

Sikindhar Jaladhi -AP21110010284

P J V Koushik-AP21110010325

## Under the Guidance of

**Srinivasa Rao L**

CONTENTS

Chapter 1) Aim

Chapter 2) Introduction

Chapter 3) Modules

Chapter 4) Code

Chapter 5) Conclusion

Aim:

The aim of this project is to develop an integrated campus management system using Java that incorporates various modules such as login, transport registration, hostel registration, library management, and complaint desk. The system will provide a centralized platform for managing and streamlining administrative tasks within a campus, ensuring efficient and effective operations.

Introduction:

Managing the diverse administrative tasks of

a campus can be a complex and time-consuming process.

The Integrated campus management system aims to simplify these operations by providing a centralized platform where student and administrators can efficiently handle various tasks are login, transport registration,

Hostel registration, Library Management and

Complaint desk related to campus facilities and services.

Modules:

Login:

The login module provides secure access to the system for different users, including students, and administrators. It validates user credentials and grants appropriate access privileges based on user roles. This module ensures that only authorized individuals can access the system and perform designated tasks.

Code:

|  |
| --- |
| import javax.swing.\*; |
|  | import java.awt.\*; |
|  | import java.awt.event.ActionEvent; |
|  | import java.awt.event.ActionListener; |
|  | import java.sql.Connection; |
|  | import java.sql.PreparedStatement; |
|  | import java.sql.ResultSet; |
|  | import java.sql.SQLException; |
|  |  |
|  | public class LoginClientSide extends JFrame { |
|  | private JTextField usernameField; |
|  | private JPasswordField passwordField; |
|  | private JButton loginButton; |
|  |  |
|  | private Connection connection; |
|  | private PreparedStatement preparedStatement; |
|  |  |
|  | public LoginClientSide(){ |
|  | InitializeDatabase initializeDatabase = new InitializeDatabase(); |
|  | initializeDatabase.getDatabase(); |
|  | initializeUI(); |
|  | } |
|  |  |
|  | private void initializeUI() { |
|  | setTitle("Login"); |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setLayout(new FlowLayout()); |
|  |  |
|  | JLabel usernameLabel = new JLabel("Username:"); |
|  | JLabel passwordLabel = new JLabel("Password:"); |
|  | usernameField = new JTextField(20); |
|  | passwordField = new JPasswordField(20); |
|  | loginButton = new JButton("Login"); |
|  |  |
|  | loginButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | login(); |
|  | } |
|  | }); |
|  |  |
|  | add(usernameLabel); |
|  | add(usernameField); |
|  | add(passwordLabel); |
|  | add(passwordField); |
|  | add(loginButton); |
|  |  |
|  | pack(); |
|  | setVisible(true); |
|  | } |
|  |  |
|  | private void login() { |
|  | String username = usernameField.getText(); |
|  | String password = new String(passwordField.getPassword()); |
|  |  |
|  | try { |
|  | String query = "SELECT \* FROM users WHERE username = ? AND password = ?"; |
|  | preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setString(1, username); |
|  | preparedStatement.setString(2, password); |
|  | ResultSet resultSet = preparedStatement.executeQuery(); |
|  |  |
|  | if (resultSet.next()) { |
|  | JOptionPane.showMessageDialog(this, "Login successful!"); |
|  | // Do something after successful login |
|  | } else { |
|  | JOptionPane.showMessageDialog(this, "Invalid username or password."); |
|  | } |
|  |  |
|  | resultSet.close(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } finally { |
|  | try { |
|  | if (preparedStatement != null) { |
|  | preparedStatement.close(); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |
|  | } |

Transport:

The transport registration module allows students to register for campus transportation services. Users can provide their personal details, select preferred routes, and submit their registration requests. This module facilitates efficient management of transportation services by maintaining records of registered users and their route preferences.

Code:

|  |
| --- |
| import java.awt.\*; |
|  | import java.awt.event.\*; |
|  | import java.sql.\*; |
|  | import javax.swing.\*; |
|  | import javax.swing.table.\*; |
|  |  |
|  | class TransportationClientSide extends JFrame { |
|  | private JTextField nameField; |
|  | private JTextField rollNumberField; |
|  | private JTextField routeField; |
|  | private JTextField vehicleNumberField; |
|  | private JButton registerButton; |
|  | private JButton displayButton; |
|  |  |
|  | private Connection connection; |
|  | private PreparedStatement insertStatement; |
|  | private PreparedStatement displayStatement; |
|  |  |
|  | public TransportationClientSide() { |
|  | InitializeDatabase initializeDatabase = new InitializeDatabase(); |
|  | initializeDatabase.getDatabase(); |
|  | initializeUI(); |
|  | } |
|  |  |
|  | private void initializeUI() { |
|  | setTitle("Transport Registration"); |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setLayout(new FlowLayout()); |
|  |  |
|  | JLabel nameLabel = new JLabel("Name:"); |
|  | JLabel rollNumberLabel = new JLabel("Roll Number:"); |
|  | JLabel routeLabel = new JLabel("Route:"); |
|  | JLabel vehicleNumberLabel = new JLabel("Vehicle Number:"); |
|  | nameField = new JTextField(20); |
|  | rollNumberField = new JTextField(20); |
|  | routeField = new JTextField(20); |
|  | vehicleNumberField = new JTextField(20); |
|  | registerButton = new JButton("Register"); |
|  | displayButton = new JButton("Display Transport Details"); |
|  |  |
|  | registerButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | register(); |
|  | } |
|  | }); |
|  |  |
|  | displayButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | displayTransportDetails(); |
|  | } |
|  | }); |
|  |  |
|  | add(nameLabel); |
|  | add(nameField); |
|  | add(rollNumberLabel); |
|  | add(rollNumberField); |
|  | add(routeLabel); |
|  | add(routeField); |
|  | add(vehicleNumberLabel); |
|  | add(vehicleNumberField); |
|  | add(registerButton); |
|  | add(displayButton); |
|  |  |
|  | pack(); |
|  | setVisible(true); |
|  | } |
|  |  |
|  | private void register() { |
|  | String name = nameField.getText(); |
|  | String rollNumber = rollNumberField.getText(); |
|  | String route = routeField.getText(); |
|  | String vehicleNumber = vehicleNumberField.getText(); |
|  |  |
|  | try { |
|  | insertStatement.setString(1, name); |
|  | insertStatement.setString(2, rollNumber); |
|  | insertStatement.setString(3, route); |
|  | insertStatement.setString(4, vehicleNumber); |
|  | int rowsAffected = insertStatement.executeUpdate(); |
|  |  |
|  | if (rowsAffected > 0) { |
|  | JOptionPane.showMessageDialog(this, "Registration successful!"); |
|  | clearFields(); |
|  | } else { |
|  | JOptionPane.showMessageDialog(this, "Registration failed."); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | private void displayTransportDetails() { |
|  | try { |
|  | ResultSet resultSet = displayStatement.executeQuery(); |
|  |  |
|  | StringBuilder details = new StringBuilder(); |
|  | details.append("Transport Details:\n"); |
|  |  |
|  | while (resultSet.next()) { |
|  | String name = resultSet.getString("name"); |
|  | String rollNumber = resultSet.getString("roll\_number"); |
|  | String route = resultSet.getString("route"); |
|  | String vehicleNumber = resultSet.getString("vehicle\_number"); |
|  |  |
|  | details.append("Name: ").append(name).append(", "); |
|  | details.append("Roll Number: ").append(rollNumber).append(", "); |
|  | details.append("Route: ").append(route).append(", "); |
|  | details.append("Vehicle Number: ").append(vehicleNumber).append("\n"); |
|  | } |
|  |  |
|  | JOptionPane.showMessageDialog(this, details.toString()); |
|  | resultSet.close(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | private void clearFields() { |
|  | nameField.setText(""); |
|  | rollNumberField.setText(""); |
|  | routeField.setText(""); |
|  | vehicleNumberField.setText(""); |
|  | } |
|  | } |
|  |  |
|  | class TransportStudentTable extends JFrame { |
|  | private JTable table; |
|  |  |
|  | public TransportStudentTable() { |
|  | setTitle("Transport Student Information"); |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setSize(600, 400); |
|  | setLocationRelativeTo(null); |
|  |  |
|  | // Create table model |
|  | DefaultTableModel model = new DefaultTableModel(); |
|  | model.addColumn("Name"); |
|  | model.addColumn("Roll Number"); |
|  | model.addColumn("Route"); |
|  | model.addColumn("Vehicle Number"); |
|  |  |
|  | // Connect to the database and fetch data |
|  | try { |
|  | Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/transport", "username", "password"); |
|  | Statement statement = connection.createStatement(); |
|  | ResultSet resultSet = statement.executeQuery("SELECT name, rollnumber, route, vehiclenumber FROM students"); |
|  |  |
|  | // Populate the table model with data from the database |
|  | while (resultSet.next()) { |
|  | String name = resultSet.getString("name"); |
|  | int rollNumber = resultSet.getInt("rollnumber"); |
|  | String route = resultSet.getString("route"); |
|  | String vehicleNumber = resultSet.getString("vehiclenumber"); |
|  |  |
|  | model.addRow(new Object[]{name, rollNumber, route, vehicleNumber}); |
|  | } |
|  |  |
|  | resultSet.close(); |
|  | statement.close(); |
|  | connection.close(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | // Create and configure the table |
|  | table = new JTable(model); |
|  | table.setPreferredScrollableViewportSize(new Dimension(500, 300)); |
|  | table.setFillsViewportHeight(true); |
|  |  |
|  | // Add the table to a scroll pane |
|  | JScrollPane scrollPane = new JScrollPane(table); |
|  | add(scrollPane, BorderLayout.CENTER); |
|  | } |
|  | } |

Hostel:

The hostel registration module streamlines the process of allocating and managing hostel accommodations. Students can submit their accommodation requests, specifying their preferences such as room type preferences. The module helps the administration allocate rooms and maintain an up-to-date record of hostel occupancy.

Code:

|  |
| --- |
| import java.awt.\*; |
|  | import java.awt.event.\*; |
|  | import java.sql.\*; |
|  | import javax.swing.\*; |
|  | import javax.swing.table.\*; |
|  |  |
|  |  |
|  | class HostelRegistration extends JFrame { |
|  | private JTextField nameField; |
|  | private JTextField rollNumberField; |
|  | private JTextField towerNumberField; |
|  | private JTextField roomNumberField; |
|  |  |
|  | private Connection connection; |
|  | private PreparedStatement insertStatement; |
|  | private PreparedStatement displayStatement; |
|  |  |
|  | public HostelRegistration() { |
|  | InitializeDatabase initializeDatabase = new InitializeDatabase(); |
|  | initializeDatabase.getDatabase(); |
|  | initializeUI(); |
|  | } |
|  |  |
|  | private void initializeUI() { |
|  | setTitle("Hostel Registration"); |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setLayout(new FlowLayout()); |
|  |  |
|  | JLabel nameLabel = new JLabel("Name:"); |
|  | JLabel rollNumberLabel = new JLabel("Roll Number:"); |
|  | JLabel towerNumberLabel = new JLabel("Tower Number:"); |
|  | JLabel roomNumberLabel = new JLabel("Room Number:"); |
|  | nameField = new JTextField(20); |
|  | rollNumberField = new JTextField(20); |
|  | towerNumberField = new JTextField(20); |
|  | roomNumberField = new JTextField(20); |
|  | JButton registerButton = new JButton("Register"); |
|  | JButton displayButton = new JButton("Display Hostel Details"); |
|  |  |
|  | registerButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | register(); |
|  | } |
|  | }); |
|  |  |
|  | displayButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | displayHostelDetails(); |
|  | } |
|  | }); |
|  |  |
|  | add(nameLabel); |
|  | add(nameField); |
|  | add(rollNumberLabel); |
|  | add(rollNumberField); |
|  | add(towerNumberLabel); |
|  | add(towerNumberField); |
|  | add(roomNumberLabel); |
|  | add(roomNumberField); |
|  | add(registerButton); |
|  | add(displayButton); |
|  |  |
|  | pack(); |
|  | setVisible(true); |
|  | } |
|  |  |
|  | private void register() { |
|  | String name = nameField.getText(); |
|  | String rollNumber = rollNumberField.getText(); |
|  | String towerNumber = towerNumberField.getText(); |
|  | String roomNumber = roomNumberField.getText(); |
|  |  |
|  | try { |
|  | insertStatement.setString(1, name); |
|  | insertStatement.setString(2, rollNumber); |
|  | insertStatement.setString(3, towerNumber); |
|  | insertStatement.setString(4, roomNumber); |
|  | int rowsAffected = insertStatement.executeUpdate(); |
|  |  |
|  | if (rowsAffected > 0) { |
|  | JOptionPane.showMessageDialog(this, "Registration successful!"); |
|  | clearFields(); |
|  | } else { |
|  | JOptionPane.showMessageDialog(this, "Registration failed."); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | private void displayHostelDetails() { |
|  | try { |
|  | ResultSet resultSet = displayStatement.executeQuery(); |
|  |  |
|  | StringBuilder details = new StringBuilder(); |
|  | details.append("Hostel Details:\n"); |
|  |  |
|  | while (resultSet.next()) { |
|  | String name = resultSet.getString("name"); |
|  | String rollNumber = resultSet.getString("roll\_number"); |
|  | String towerNumber = resultSet.getString("tower\_number"); |
|  | String roomNumber = resultSet.getString("room\_number"); |
|  |  |
|  | details.append("Name: ").append(name).append(", "); |
|  | details.append("Roll Number: ").append(rollNumber).append(", "); |
|  | details.append("Tower Number: ").append(towerNumber).append(", "); |
|  | details.append("Room Number: ").append(roomNumber).append("\n"); |
|  | } |
|  |  |
|  | JOptionPane.showMessageDialog(this, details.toString()); |
|  | resultSet.close(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | private void clearFields() { |
|  | nameField.setText(""); |
|  | rollNumberField.setText(""); |
|  | towerNumberField.setText(""); |
|  | roomNumberField.setText(""); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | class HostelStudentTable extends JFrame { |
|  |  |
|  | public HostelStudentTable() { |
|  | setTitle("Hostel Student Information"); |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setSize(600, 400); |
|  | setLocationRelativeTo(null); |
|  |  |
|  | // Create table model |
|  | DefaultTableModel model = new DefaultTableModel(); |
|  | model.addColumn("Name"); |
|  | model.addColumn("Roll Number"); |
|  | model.addColumn("Tower Number"); |
|  | model.addColumn("Room Number"); |
|  |  |
|  | // Connect to the database and fetch data |
|  | try { |
|  | Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/hostel", "username", "password"); |
|  | Statement statement = connection.createStatement(); |
|  | ResultSet resultSet = statement.executeQuery("SELECT name, rollnumber, towernumber, roomnumber FROM students"); |
|  |  |
|  | // Populate the table model with data from the database |
|  | while (resultSet.next()) { |
|  | String name = resultSet.getString("name"); |
|  | int rollNumber = resultSet.getInt("rollnumber"); |
|  | int towerNumber = resultSet.getInt("towernumber"); |
|  | int roomNumber = resultSet.getInt("roomnumber"); |
|  |  |
|  | model.addRow(new Object[]{name, rollNumber, towerNumber, roomNumber}); |
|  | } |
|  |  |
|  | resultSet.close(); |
|  | statement.close(); |
|  | connection.close(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | // Create and configure the table |
|  | JTable table = new JTable(model); |
|  | table.setPreferredScrollableViewportSize(new Dimension(500, 300)); |
|  | table.setFillsViewportHeight(true); |
|  |  |
|  | // Add the table to a scroll pane |
|  | JScrollPane scrollPane = new JScrollPane(table); |
|  | add(scrollPane, BorderLayout.CENTER); |
|  | } |
|  |  |
|  |  |
|  | } |

Library:

The library management module assists in organizing and maintaining the campus library resources. It enables users to search for. Librarians can manage book cataloging. This module ensures smooth library operations and improves accessibility to resources.

Code:

|  |
| --- |
| import java.awt.\*; |
|  | import javax.swing.\*; |
|  | import java.awt.event.\*; |
|  | import java.sql.\*; |
|  | import javax.swing.UIManager; |
|  | import javax.swing.UnsupportedLookAndFeelException; |
|  | import javax.swing.table.DefaultTableModel; |
|  |  |
|  | class LibraryClientSide extends JFrame { |
|  | private final JTextField searchField; |
|  | private final DefaultTableModel tableModel; |
|  |  |
|  | private Statement statement; |
|  |  |
|  | public LibraryClientSide() { |
|  | super("Library Management System"); |
|  | searchField = new JTextField(20); |
|  | JButton searchButton = new JButton("Search"); |
|  | JButton show\_allButton=new JButton("Show all"); |
|  | tableModel = new DefaultTableModel(); |
|  | JTable table = new JTable(tableModel); |
|  |  |
|  | setLayout(new BorderLayout()); |
|  | JPanel inputPanel = new JPanel(new FlowLayout()); |
|  | inputPanel.add(new JLabel("Search by Title:")); |
|  | inputPanel.add(searchField); |
|  | inputPanel.add(searchButton); |
|  | inputPanel.add(show\_allButton); |
|  | inputPanel.add(new JScrollPane(table)); |
|  |  |
|  | add(inputPanel, BorderLayout.NORTH); |
|  | add(new JScrollPane(table), BorderLayout.CENTER); |
|  |  |
|  |  |
|  | searchButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | searchBooks(); |
|  | } |
|  | }); |
|  | show\_allButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | showBooks(); |
|  | } |
|  | }); |
|  |  |
|  |  |
|  |  |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setSize(1900, 800); |
|  | setVisible(true); |
|  |  |
|  |  |
|  | try { |
|  | Class.forName("com.mysql.cj.jdbc.Driver"); |
|  | Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc", "root", "toor"); |
|  | statement = connection.createStatement(); |
|  | } catch (Exception e) { |
|  | e.printStackTrace(); |
|  | } |
|  | tableModel.setColumnIdentifiers(new Object[]{"Book\_Name", "Author"}); |
|  | } |
|  | private void showBooks(){ |
|  | try{ |
|  | String query="SELECT \* from books"; |
|  | ResultSet resultSet=statement.executeQuery(query); |
|  | tableModel.setRowCount(0); |
|  | while (resultSet.next()) { |
|  | String title = resultSet.getString("Book\_Name"); |
|  | String author = resultSet.getString("Author"); |
|  | tableModel.addRow(new Object[]{title, author}); |
|  | } |
|  | }catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error searching books!"); |
|  | } |
|  | } |
|  | private void searchBooks() { |
|  | String keyword = searchField.getText(); |
|  |  |
|  | try { |
|  | String query = "SELECT \* FROM books WHERE Book\_Name LIKE '%" + keyword + "%'"; |
|  | ResultSet resultSet = statement.executeQuery(query); |
|  | tableModel.setRowCount(0); |
|  | while (resultSet.next()) { |
|  | String title = resultSet.getString("Book\_Name"); |
|  | String author = resultSet.getString("Author"); |
|  | tableModel.addRow(new Object[]{title, author}); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error searching books!"); |
|  | } |
|  | } |
|  |  |
|  | public static void main(String[] args) { |
|  | try { |
|  |  |
|  | UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel"); |
|  |  |
|  | SwingUtilities.invokeLater(new Runnable() { |
|  | public void run() { |
|  | new LibraryClientSide(); |
|  | } |
|  | }); |
|  | } catch (ClassNotFoundException | InstantiationException | |
|  | IllegalAccessException | UnsupportedLookAndFeelException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |

Complaint Desk:

The complaint desk module provides a platform for users to submit complaints or report issues related to various campus facilities or services. Users can provide details of the complaint, track its progress, and receive updates on its resolution. Administrators can manage and prioritize complaints, assign them to relevant personnel, and ensure timely resolution. This module aims to enhance user satisfaction and address concerns promptly.

Code:

Client side:

|  |
| --- |
| import java.awt.\*; |
|  | import javax.swing.\*; |
|  | import java.awt.event.\*; |
|  | import java.sql.\*; |
|  | import javax.swing.UIManager; |
|  | import javax.swing.UnsupportedLookAndFeelException; |
|  | import javax.swing.table.DefaultTableModel; |
|  |  |
|  | public class ComplaintsAdminSide extends JFrame { |
|  | private JTextField subjectField, detailsField, locationField; |
|  | private DefaultTableModel tableModel; |
|  | private JTable table; |
|  | private Connection connection; |
|  | private Statement statement; |
|  |  |
|  | public ComplaintsAdminSide() { |
|  | super("Complaint Administration"); |
|  | subjectField = new JTextField(20); |
|  | detailsField = new JTextField(20); |
|  | locationField = new JTextField(20); |
|  |  |
|  | JButton addButton = new JButton("Register Complaint"); |
|  | JButton displayButton = new JButton("Display Complaints"); |
|  | JButton completeButton = new JButton("Set as Completed"); |
|  | JButton editButton = new JButton("Edit Complaint"); |
|  |  |
|  | tableModel = new DefaultTableModel(); |
|  | table = new JTable(tableModel); |
|  | setLayout(new BorderLayout()); |
|  |  |
|  | JPanel inputPanel = new JPanel(new FlowLayout()); |
|  | inputPanel.add(new JLabel("Complaint Subject")); |
|  | inputPanel.add(subjectField); |
|  | inputPanel.add(new JLabel("Complaint Description")); |
|  | inputPanel.add(detailsField); |
|  | inputPanel.add(new JLabel("Location")); |
|  | inputPanel.add(locationField); |
|  | inputPanel.add(addButton); |
|  | inputPanel.add(displayButton); |
|  | inputPanel.add(completeButton); |
|  | inputPanel.add(editButton); |
|  |  |
|  | add(inputPanel, BorderLayout.NORTH); |
|  | add(new JScrollPane(table), BorderLayout.CENTER); |
|  |  |
|  | addButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | addComplaint(); |
|  | } |
|  | }); |
|  |  |
|  | displayButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | displayComplaints(); |
|  | } |
|  | }); |
|  |  |
|  | completeButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | setComplaintCompleted(); |
|  | } |
|  | }); |
|  |  |
|  | editButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | editComplaint(); |
|  | } |
|  | }); |
|  |  |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setSize(800, 600); |
|  | setVisible(true); |
|  |  |
|  | try { |
|  | Class.forName("com.mysql.jdbc.Driver"); |
|  | connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc", "root", "toor"); |
|  | statement = connection.createStatement(); |
|  | } catch (Exception e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | tableModel.setColumnIdentifiers(new Object[] { "Subject", "Description", "Location", "Completed" }); |
|  | } |
|  |  |
|  | private void addComplaint() { |
|  | String subject = subjectField.getText(); |
|  | String description = detailsField.getText(); |
|  | String location = locationField.getText(); |
|  | try { |
|  | String query = "INSERT INTO FAQ (Subject, Details, Location, Completion) VALUES (?, ?, ?, ?)"; |
|  | PreparedStatement preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setString(1, subject); |
|  | preparedStatement.setString(2, description); |
|  | preparedStatement.setString(3, location); |
|  | preparedStatement.setBoolean(4, false); |
|  | preparedStatement.executeUpdate(); |
|  | JOptionPane.showMessageDialog(this, "Complaint registered successfully"); |
|  | subjectField.setText(""); |
|  | detailsField.setText(""); |
|  | locationField.setText(""); |
|  | displayComplaints(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error registering complaint"); |
|  | } |
|  | } |
|  |  |
|  | private void displayComplaints() { |
|  | try { |
|  | String query = "SELECT \* FROM FAQ"; |
|  | ResultSet resultSet = statement.executeQuery(query); |
|  | tableModel.setRowCount(0); |
|  | while (resultSet.next()) { |
|  | String subject = resultSet.getString("Subject"); |
|  | String description = resultSet.getString("Details"); |
|  | String location = resultSet.getString("Location"); |
|  | boolean completed =resultSet.getBoolean("Completion"); |
|  | tableModel.addRow(new Object[]{subject, description, location, completed}); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error displaying complaints"); |
|  | } |
|  | } |
|  |  |
|  | private void setComplaintCompleted() { |
|  | int selectedRow = table.getSelectedRow(); |
|  | if (selectedRow == -1) { |
|  | JOptionPane.showMessageDialog(this, "No complaint selected"); |
|  | return; |
|  | } |
|  |  |
|  | String subject = (String) table.getValueAt(selectedRow, 0); |
|  | try { |
|  | String query = "UPDATE FAQ SET Completion = ? WHERE Subject = ?"; |
|  | PreparedStatement preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setBoolean(1, true); |
|  | preparedStatement.setString(2, subject); |
|  | preparedStatement.executeUpdate(); |
|  | JOptionPane.showMessageDialog(this, "Complaint marked as completed"); |
|  | displayComplaints(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error marking complaint as completed"); |
|  | } |
|  | } |
|  |  |
|  | private void editComplaint() { |
|  | int selectedRow = table.getSelectedRow(); |
|  | if (selectedRow == -1) { |
|  | JOptionPane.showMessageDialog(this, "No complaint selected"); |
|  | return; |
|  | } |
|  |  |
|  | String subject = (String) table.getValueAt(selectedRow, 0); |
|  | String description = (String) table.getValueAt(selectedRow, 1); |
|  | String location = (String) table.getValueAt(selectedRow, 2); |
|  |  |
|  | String newSubject = JOptionPane.showInputDialog(this, "Enter new subject:", subject); |
|  | String newDescription = JOptionPane.showInputDialog(this, "Enter new description:", description); |
|  | String newLocation = JOptionPane.showInputDialog(this, "Enter new location:", location); |
|  |  |
|  | try { |
|  | String query = "UPDATE FAQ SET Subject = ?, Details = ?, Location = ? WHERE Subject = ?"; |
|  | PreparedStatement preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setString(1, newSubject); |
|  | preparedStatement.setString(2, newDescription); |
|  | preparedStatement.setString(3, newLocation); |
|  | preparedStatement.setString(4, subject); |
|  | preparedStatement.executeUpdate(); |
|  | JOptionPane.showMessageDialog(this, "Complaint updated successfully"); |
|  | displayComplaints(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error updating complaint"); |
|  | } |
|  | } |
|  |  |
|  | public static void main(String[] args) { |
|  | SwingUtilities.invokeLater(() -> { |
|  | HostelRegistration hostelRegistration = new HostelRegistration(); |
|  | hostelRegistration.setVisible(true); |
|  | HostelStudentTable table = new HostelStudentTable(); |
|  | table.setVisible(true); |
|  | }); |
|  | } |
|  | } |

Admin side:

|  |
| --- |
| import java.awt.\*; |
|  | import javax.swing.\*; |
|  | import java.awt.event.\*; |
|  | import java.sql.\*; |
|  | import javax.swing.UIManager; |
|  | import javax.swing.UnsupportedLookAndFeelException; |
|  | import javax.swing.table.DefaultTableModel; |
|  |  |
|  | public class ComplaintsAdminSide extends JFrame { |
|  | private JTextField subjectField, detailsField, locationField; |
|  | private DefaultTableModel tableModel; |
|  | private JTable table; |
|  | private Connection connection; |
|  | private Statement statement; |
|  |  |
|  | public ComplaintsAdminSide() { |
|  | super("Complaint Administration"); |
|  | subjectField = new JTextField(20); |
|  | detailsField = new JTextField(20); |
|  | locationField = new JTextField(20); |
|  |  |
|  | JButton addButton = new JButton("Register Complaint"); |
|  | JButton displayButton = new JButton("Display Complaints"); |
|  | JButton completeButton = new JButton("Set as Completed"); |
|  | JButton editButton = new JButton("Edit Complaint"); |
|  |  |
|  | tableModel = new DefaultTableModel(); |
|  | table = new JTable(tableModel); |
|  | setLayout(new BorderLayout()); |
|  |  |
|  | JPanel inputPanel = new JPanel(new FlowLayout()); |
|  | inputPanel.add(new JLabel("Complaint Subject")); |
|  | inputPanel.add(subjectField); |
|  | inputPanel.add(new JLabel("Complaint Description")); |
|  | inputPanel.add(detailsField); |
|  | inputPanel.add(new JLabel("Location")); |
|  | inputPanel.add(locationField); |
|  | inputPanel.add(addButton); |
|  | inputPanel.add(displayButton); |
|  | inputPanel.add(completeButton); |
|  | inputPanel.add(editButton); |
|  |  |
|  | add(inputPanel, BorderLayout.NORTH); |
|  | add(new JScrollPane(table), BorderLayout.CENTER); |
|  |  |
|  | addButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | addComplaint(); |
|  | } |
|  | }); |
|  |  |
|  | displayButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | displayComplaints(); |
|  | } |
|  | }); |
|  |  |
|  | completeButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | setComplaintCompleted(); |
|  | } |
|  | }); |
|  |  |
|  | editButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | editComplaint(); |
|  | } |
|  | }); |
|  |  |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setSize(800, 600); |
|  | setVisible(true); |
|  |  |
|  | try { |
|  | Class.forName("com.mysql.jdbc.Driver"); |
|  | connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc", "root", "toor"); |
|  | statement = connection.createStatement(); |
|  | } catch (Exception e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | tableModel.setColumnIdentifiers(new Object[] { "Subject", "Description", "Location", "Completed" }); |
|  | } |
|  |  |
|  | private void addComplaint() { |
|  | String subject = subjectField.getText(); |
|  | String description = detailsField.getText(); |
|  | String location = locationField.getText(); |
|  | try { |
|  | String query = "INSERT INTO FAQ (Subject, Details, Location, Completion) VALUES (?, ?, ?, ?)"; |
|  | PreparedStatement preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setString(1, subject); |
|  | preparedStatement.setString(2, description); |
|  | preparedStatement.setString(3, location); |
|  | preparedStatement.setBoolean(4, false); |
|  | preparedStatement.executeUpdate(); |
|  | JOptionPane.showMessageDialog(this, "Complaint registered successfully"); |
|  | subjectField.setText(""); |
|  | detailsField.setText(""); |
|  | locationField.setText(""); |
|  | displayComplaints(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error registering complaint"); |
|  | } |
|  | } |
|  |  |
|  | private void displayComplaints() { |
|  | try { |
|  | String query = "SELECT \* FROM FAQ"; |
|  | ResultSet resultSet = statement.executeQuery(query); |
|  | tableModel.setRowCount(0); |
|  | while (resultSet.next()) { |
|  | String subject = resultSet.getString("Subject"); |
|  | String description = resultSet.getString("Details"); |
|  | String location = resultSet.getString("Location"); |
|  | boolean completed =resultSet.getBoolean("Completion"); |
|  | tableModel.addRow(new Object[]{subject, description, location, completed}); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error displaying complaints"); |
|  | } |
|  | } |
|  |  |
|  | private void setComplaintCompleted() { |
|  | int selectedRow = table.getSelectedRow(); |
|  | if (selectedRow == -1) { |
|  | JOptionPane.showMessageDialog(this, "No complaint selected"); |
|  | return; |
|  | } |
|  |  |
|  | String subject = (String) table.getValueAt(selectedRow, 0); |
|  | try { |
|  | String query = "UPDATE FAQ SET Completion = ? WHERE Subject = ?"; |
|  | PreparedStatement preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setBoolean(1, true); |
|  | preparedStatement.setString(2, subject); |
|  | preparedStatement.executeUpdate(); |
|  | JOptionPane.showMessageDialog(this, "Complaint marked as completed"); |
|  | displayComplaints(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error marking complaint as completed"); |
|  | } |
|  | } |
|  |  |
|  | private void editComplaint() { |
|  | int selectedRow = table.getSelectedRow(); |
|  | if (selectedRow == -1) { |
|  | JOptionPane.showMessageDialog(this, "No complaint selected"); |
|  | return; |
|  | } |
|  |  |
|  | String subject = (String) table.getValueAt(selectedRow, 0); |
|  | String description = (String) table.getValueAt(selectedRow, 1); |
|  | String location = (String) table.getValueAt(selectedRow, 2); |
|  |  |
|  | String newSubject = JOptionPane.showInputDialog(this, "Enter new subject:", subject); |
|  | String newDescription = JOptionPane.showInputDialog(this, "Enter new description:", description); |
|  | String newLocation = JOptionPane.showInputDialog(this, "Enter new location:", location); |
|  |  |
|  | try { |
|  | String query = "UPDATE FAQ SET Subject = ?, Details = ?, Location = ? WHERE Subject = ?"; |
|  | PreparedStatement preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setString(1, newSubject); |
|  | preparedStatement.setString(2, newDescription); |
|  | preparedStatement.setString(3, newLocation); |
|  | preparedStatement.setString(4, subject); |
|  | preparedStatement.executeUpdate(); |
|  | JOptionPane.showMessageDialog(this, "Complaint updated successfully"); |
|  | displayComplaints(); |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | JOptionPane.showMessageDialog(this, "Error updating complaint"); |
|  | } |
|  | } |
|  |  |
|  | public static void main(String[] args) { |
|  | SwingUtilities.invokeLater(() -> { |
|  | HostelRegistration hostelRegistration = new HostelRegistration(); |
|  | hostelRegistration.setVisible(true); |
|  | HostelStudentTable table = new HostelStudentTable(); |
|  | table.setVisible(true); |
|  | }); |
|  | } |
|  | } |

Registration:

Code:

|  |
| --- |
| import javax.swing.\*; |
|  | import java.awt.\*; |
|  | import java.awt.event.ActionEvent; |
|  | import java.awt.event.ActionListener; |
|  | import java.sql.Connection; |
|  | import java.sql.PreparedStatement; |
|  | import java.sql.SQLException; |
|  |  |
|  | public class RegistrationsClientSide extends JFrame { |
|  | private JTextField usernameField; |
|  | private JPasswordField passwordField; |
|  | private JButton registerButton; |
|  |  |
|  | private Connection connection; |
|  | private PreparedStatement preparedStatement; |
|  |  |
|  | public RegistrationsClientSide() { |
|  | InitializeDatabase initializeDatabase = new InitializeDatabase(); |
|  | initializeDatabase.getDatabase(); |
|  | initializeUI(); |
|  | } |
|  |  |
|  | private void initializeUI() { |
|  | setTitle("Registration"); |
|  | setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |
|  | setLayout(new FlowLayout()); |
|  |  |
|  | JLabel usernameLabel = new JLabel("Username:"); |
|  | JLabel passwordLabel = new JLabel("Password:"); |
|  | usernameField = new JTextField(20); |
|  | passwordField = new JPasswordField(20); |
|  | registerButton = new JButton("Register"); |
|  |  |
|  | registerButton.addActionListener(new ActionListener() { |
|  | public void actionPerformed(ActionEvent e) { |
|  | register(); |
|  | } |
|  | }); |
|  |  |
|  | add(usernameLabel); |
|  | add(usernameField); |
|  | add(passwordLabel); |
|  | add(passwordField); |
|  | add(registerButton); |
|  |  |
|  | pack(); |
|  | setVisible(true); |
|  | } |
|  |  |
|  | private void register() { |
|  | String username = usernameField.getText(); |
|  | String password = new String(passwordField.getPassword()); |
|  |  |
|  | try { |
|  | String query = "INSERT INTO users (username, password) VALUES (?, ?)"; |
|  | preparedStatement = connection.prepareStatement(query); |
|  | preparedStatement.setString(1, username); |
|  | preparedStatement.setString(2, password); |
|  | int rowsAffected = preparedStatement.executeUpdate(); |
|  |  |
|  | if (rowsAffected > 0) { |
|  | JOptionPane.showMessageDialog(this, "Registration successful!"); |
|  | // Do something after successful registration |
|  | } else { |
|  | JOptionPane.showMessageDialog(this, "Registration failed."); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } finally { |
|  | try { |
|  | if (preparedStatement != null) { |
|  | preparedStatement.close(); |
|  | } |
|  | } catch (SQLException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |
|  | } |

Conclusion:

The development of the Integrated Campus Management System has been successfully achieved, providing a comprehensive solution for streamlining and automating administrative tasks within the educational institution. The project has encompassed modules for login authentication, transport registration, hostel registration, library management, and a complaint desk, all implemented using Java programming language.

In conclusion, the Integrated Campus Management System has achieved its goal of simplifying and automating administrative processes within the educational institution. The project demonstrates the power of Java programming language in creating a robust and user-friendly system, ultimately contributing to a more efficient and well-organized campus environment.

CODE LINK: https://github.com/sakkurthi-sashank/java-project/tree/main/Java-Project/src