STUDENT MANAGEMENT SYSTEM
MINI PROJECT REPORT

Submitted by

Suriya Prakash	**230701352**
Shayaan Shaikh	**230701189**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 25

BONAFIDE CERTIFICATE

Certified that this project report "**STUDENT MANAGEMENT SYSTEM**" is the bonafide work of

"SURIYA PRAKASH(230701352), SHAYAAN SHAIKH (230701189)"

who carried out the project work under my supervision.

Submitted for the Practical Examination held on ________________________

SIGNATURE

Mrs.K.Mahesmeena
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College
Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT:

The **Student Management System** is a comprehensive tool designed to streamline and optimize the management of student records and academic information. This system facilitates efficient tracking of student data, course enrollment, and performance records, providing a centralized solution for administrators and educators to manage and monitor academic activities. By offering a clear overview of student details, attendance records, grades, and other academic information, the system enables informed decision-making for both educators and students.

Through its structured organization of student data and real-time information capabilities, the Student Management System ensures that academic operations run smoothly and efficiently. This fosters better academic planning, aids in progress tracking, and encourages improved student engagement. Users can add, update, and view student details, register courses, and generate performance reports, maintaining transparency and accountability within the academic environment.

The system features an intuitive user interface that includes an admin dashboard, enabling administrators to oversee and manage student accounts, monitor academic progress, and analyze trends in academic performance. With advanced security protocols and efficient database management, the system ensures that all student information is accurate, secure, and up-to-date.

Overall, the Student Management System enhances the efficiency of academic management processes and empowers educators and students to achieve greater success and collaboration in the learning environment.

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

1.1 INTRODUCTION

Effective student management is essential for academic institutions to ensure smooth operations, maintain accurate records, and provide a positive experience for students and staff. The **Student Management System** offers a streamlined, user-friendly platform designed to securely and efficiently manage student information. With a comprehensive suite of tools, administrators and educators can monitor student records, course enrollments, and performance data with ease.

The system provides core functionalities, including student registration, secure login, and academic record management (attendance, grades, and course details). Additionally, it maintains essential information such as personal details, academic performance, and course history, ensuring all student-related data is accessible in one place. These features empower administrators, educators, and students with a holistic view of academic information, fostering better decision-making and collaboration.

Built with Java and MySQL, the **Student Management System** leverages Java for backend logic and an intuitive GUI developed using JFrame. MySQL supports data storage and retrieval, offering a secure and reliable foundation for managing student records. The Java Swing-based interface enhances usability, delivering a modern and interactive experience for users.

This report details the system's development, architecture, and technology integration, demonstrating how Java, MySQL, and NetBeans combine to create a secure and efficient student management solution. The system aims to provide a seamless, reliable user experience to address academic management needs with industry-standard performance.

1.2 OBJECTIVES

- To develop a centralized database for securely managing student profiles, course enrollments, and academic records.
- To enable efficient handling of attendance and grades, providing real-time updates for students and administrators.
- To provide a secure login system for authentication, ensuring the confidentiality of student data.
- To allow users to easily access and view essential student details, including personal information, academic performance, and course registrations, on a streamlined interface.
- To ensure compliance with data security standards for information storage and management.

1.3 MODULES

1. **Student Registration Module**
   The module captures essential student information, such as name, email, password, and course enrollment. Data is securely stored in the database. This module ensures that only authorized users can access student records by implementing credential storage with secure encryption and validation mechanisms.

2. **Profile Management Module**
   This module provides a centralized view for users to access and manage student profiles. Administrators can view critical information such as student IDs, enrolled courses, grades, and

attendance. The system ensures that data is accurate and secure with robust authentication mechanisms.

3. **Attendance Management Module**
   This module facilitates attendance tracking for each student. Educators can update attendance records, and students can view their attendance percentage in real-time.

4. **Performance Tracking Module**
   The module records grades and performance metrics for each student. Educators can update student performance details, while students and parents can view detailed progress reports.

5. **Admin Dashboard Module**
   The dashboard provides administrators with tools to oversee student records, monitor academic progress, and generate performance reports. Role-based access ensures data security and restricts unauthorized access.

6. **Database Management Module**
   This module is responsible for securely storing and retrieving all student, course, and academic data. MySQL serves as the backend database, ensuring efficient data management and high reliability.

Chapter 2: SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The **Student Management System** utilizes a combination of technologies to ensure robust and efficient functionality. The backend is supported by a relational database management system (RDBMS), while the frontend features an interactive and user-friendly interface built using Java Swing. Middleware technologies enable seamless communication between the backend and frontend.

2.1.1 Java

- **Role**: Java serves as the primary programming language for both backend logic and GUI development.
- **Usage**:
  - Backend: Handles operations like student registration, profile management, and academic updates.
  - Frontend: A JFrame-based GUI provides an intuitive and interactive user experience.
  - Middleware: Java Database Connectivity (JDBC) ensures seamless communication with the MySQL database.
- **Advantages**:
  - Platform independence for cross-platform compatibility.
  - Built-in security features to protect sensitive student data.

2.1.2 MySQL

- **Role**: MySQL is used as the relational database for storing and managing all student-related information.
- **Usage**:
  - Stores student profiles, attendance, grades, and course information.
  - Efficient SQL queries enable quick retrieval and management of large datasets.
- **Advantages**:
  - Reliable, open-source database management.
  - Ensures data integrity and supports complex queries for academic reporting.

Chapter 3: REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Functional Requirements

- **User Authentication and Authorization**
  - Enable secure student registration and login.
  - Maintain session details for logged-in users.
- **Student Registration and Profile Management**
  - Allow creation and updating of student profiles with personal and academic information.
  - Provide a home screen displaying student details, such as name, student ID, enrolled courses, and grades.
- **Attendance and Performance Tracking**
  - Enable real-time tracking of attendance and academic performance.
  - Update and display grades immediately after educator input.
- **Database Records Management**
  - Use unique identifiers (e.g., student ID) for managing records.
  - Separate tables for student profiles, course enrollments, attendance, and grades for efficient data handling.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

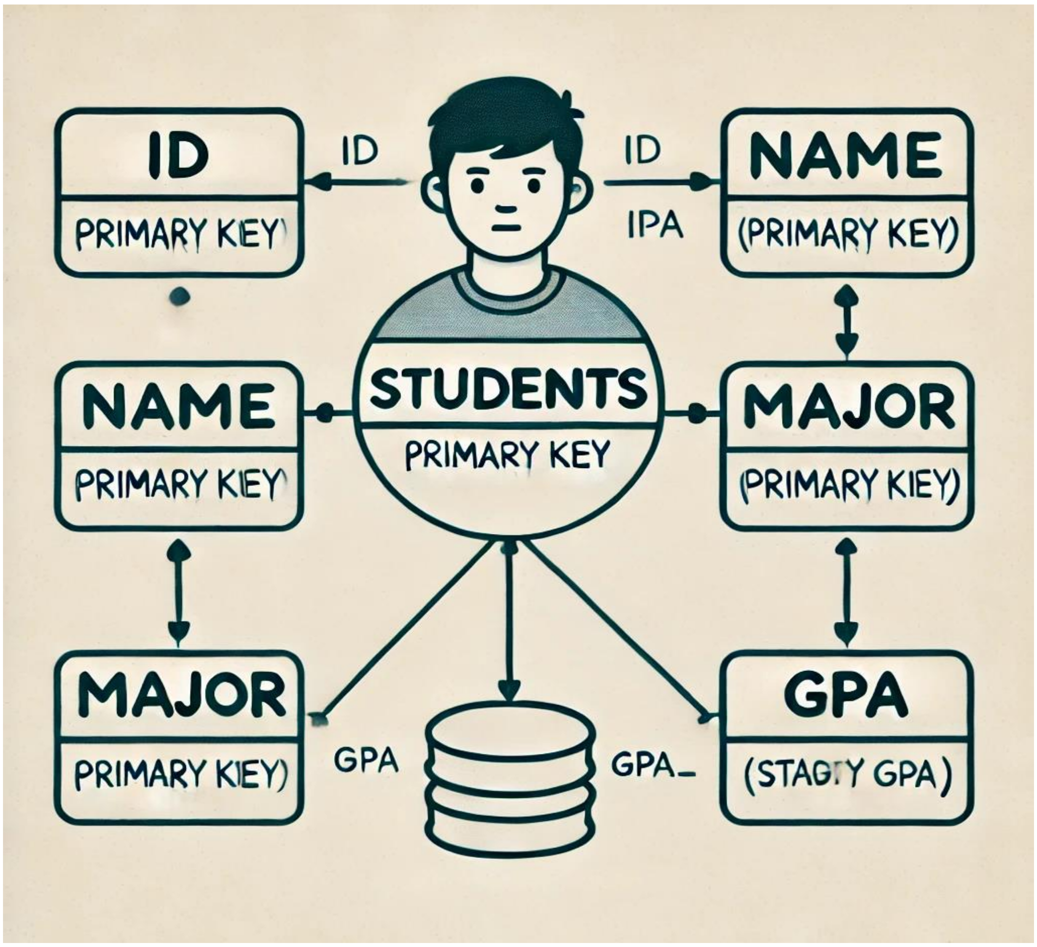
Hardware Requirements

- **Processor**: Intel Core i3 or equivalent for smooth processing.
- **RAM**: 4 GB or higher to handle concurrent database operations.
- **Storage**: At least 500 MB for application files and database storage.
- **Monitor Resolution**: 1024 x 768 or higher.

Software Requirements

- **Operating System**: Windows 10 or higher.
- **Frontend**: Java Swing (JFrame-based interface).
- **Backend**: MySQL for database management.
- **IDE**: NetBeans for development.
- **Version Control**: Git for code versioning and collaboration.

3.3 ER DIAGRAM



Chapter 4 :PROGRAM CODE

1. **Login And Signup Page**

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

public class LoginPage extends JFrame {
   private JTextField usernameField;
   private JPasswordField passwordField;

   public LoginPage() {
      // Set up the frame
      setTitle("Login Page");
      setSize(400, 250);
      setLocationRelativeTo(null);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLayout(new GridLayout(4, 1, 10, 10));

      // Set background color
      getContentPane().setBackground(new Color(102, 51, 153));  // Medium purple

      // Create components
      JLabel usernameLabel = new JLabel("Username:");
      JLabel passwordLabel = new JLabel("Password:");

      usernameField = new JTextField();
      passwordField = new JPasswordField();
```

```
JButton loginButton = new JButton("Login");
JButton signupButton = new JButton("Sign Up");

// Set fonts
Font labelFont = new Font("Verdana", Font.BOLD, 14);  // Bold font for labels
Font buttonFont = new Font("Verdana", Font.BOLD, 14); // Bold font for buttons

// Style components
usernameLabel.setFont(labelFont);
passwordLabel.setFont(labelFont);

// Set text color for username and password labels to white
usernameLabel.setForeground(Color.WHITE); // White text for username label
passwordLabel.setForeground(Color.WHITE); // White text for password label

// Set text color for username and password fields to purple
usernameField.setForeground(new Color(102, 51, 153));  // Purple text for username
passwordField.setForeground(new Color(102, 51, 153));  // Purple text for password
usernameField.setBackground(Color.WHITE);  // White background for username field
passwordField.setBackground(Color.WHITE);  // White background for password field

loginButton.setFont(buttonFont);
signupButton.setFont(buttonFont);

// Set button background color to white
loginButton.setBackground(Color.WHITE);
signupButton.setBackground(Color.WHITE);

// Change text color of buttons to purple
loginButton.setForeground(new Color(102, 51, 153));  // Purple text for Login button
signupButton.setForeground(new Color(102, 51, 153)); // Purple text for Sign Up button
```

```
    // Set border for better visibility of the buttons
    loginButton.setBorder(BorderFactory.createLineBorder(Color.DARK_GRAY, 1));
    signupButton.setBorder(BorderFactory.createLineBorder(Color.DARK_GRAY, 1));

    // Add components to frame
    add(usernameLabel);
    add(usernameField);
    add(passwordLabel);
    add(passwordField);
    add(loginButton);
    add(signupButton);

    // Add button listeners
    loginButton.addActionListener(new LoginAction());
    signupButton.addActionListener(new SignupAction());

    setVisible(true);
  }

  // ActionListener for the login button
  class LoginAction implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
      String username = usernameField.getText();
      String password = new String(passwordField.getPassword());

      // Check the login credentials (username and password)
      if (username.equals("admin") && password.equals("admin")) {
        JOptionPane.showMessageDialog(LoginPage.this, "Login Successful!");
        dispose(); // Close the current login page
        new HomePage(username); // Pass the logged-in username to the HomePage
      } else {
```

```
                JOptionPane.showMessageDialog(LoginPage.this, "Invalid username or password");
            }
        }
    }

    // ActionListener for the signup button
    class SignupAction implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            dispose();
            new SignupPage(); // Open the signup page
        }
    }

    public static void main(String[] args) {
        new LoginPage(); // Show login page
    }
}

import javax.swing.*;
import java.awt.*;
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 SignupPage extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField, confirmPasswordField;
```

```
public SignupPage() {
    // Set up frame
    setTitle("Sign Up Page");
    setSize(400, 300);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout(10, 10)); // BorderLayout with 10px gaps

    // Set the background color of the entire frame to purple
    getContentPane().setBackground(new Color(102, 51, 153));  // Purple background

    // Create a panel for the form components (username, password fields, etc.)
    JPanel formPanel = new JPanel();
    formPanel.setLayout(new GridLayout(4, 2, 10, 10)); // Grid layout for form fields
    formPanel.setBackground(new Color(102, 51, 153));  // Set form panel background to purple

    // Create components
    JLabel usernameLabel = new JLabel("Username:");
    JLabel passwordLabel = new JLabel("Password:");
    JLabel confirmPasswordLabel = new JLabel("Confirm Password:");

    usernameField = new JTextField(20);
    passwordField = new JPasswordField(20);
    confirmPasswordField = new JPasswordField(20);

    // Set text color for labels to white
    usernameLabel.setForeground(Color.WHITE);
    passwordLabel.setForeground(Color.WHITE);
    confirmPasswordLabel.setForeground(Color.WHITE);

    // Set text color inside text fields to purple and background to white
    Color purple = new Color(102, 51, 153);  // Purple text color
```

```
        usernameField.setForeground(purple);  // Purple text for username
        passwordField.setForeground(purple);  // Purple text for password
        confirmPasswordField.setForeground(purple);  // Purple text for confirm password
        usernameField.setBackground(Color.WHITE);  // White background for username field
        passwordField.setBackground(Color.WHITE);  // White background for password field
        confirmPasswordField.setBackground(Color.WHITE);  // White background for confirm password field

        // Add components to form panel
        formPanel.add(usernameLabel);
        formPanel.add(usernameField);
        formPanel.add(passwordLabel);
        formPanel.add(passwordField);
        formPanel.add(confirmPasswordLabel);
        formPanel.add(confirmPasswordField);

        // Add the form panel to the center of the frame
        add(formPanel, BorderLayout.CENTER);

        // Create buttons panel (to be placed at the bottom)
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 10)); // FlowLayout with gap between
buttons

        JButton saveButton = new JButton("Save");
        JButton cancelButton = new JButton("Back");

        // Set fonts for buttons
        Font buttonFont = new Font("Verdana", Font.BOLD, 14);
        saveButton.setFont(buttonFont);
        cancelButton.setFont(buttonFont);

        // Set preferred size of buttons
        saveButton.setPreferredSize(new Dimension(100, 30));  // Reduced button size
```

```
    cancelButton.setPreferredSize(new Dimension(100, 30)); // Reduced button size

    // Set background colors for the buttons
    saveButton.setBackground(Color.WHITE);  // White background for Save button
    cancelButton.setBackground(Color.WHITE);  // White background for Back button

    // Set foreground (text color) of buttons to purple
    saveButton.setForeground(purple);       // Text color for Save button
    cancelButton.setForeground(purple);     // Text color for Back button

    // Add buttons to the button panel
    buttonPanel.add(saveButton);
    buttonPanel.add(cancelButton);

    // Add the button panel to the SOUTH region of the frame
    add(buttonPanel, BorderLayout.SOUTH);

    // Add button listeners
    saveButton.addActionListener(new SaveAction());
    cancelButton.addActionListener(new CancelAction());

    setVisible(true);
}

// ActionListener for the Save button
class SaveAction implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());
        String confirmPassword = new String(confirmPasswordField.getPassword());
        // Validate input
        if (username.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {
            JOptionPane.showMessageDialog(SignupPage.this, "Please fill all fields.");
            return;
        }

        if (!password.equals(confirmPassword)) {
            JOptionPane.showMessageDialog(SignupPage.this, "Passwords do not match.");
            return;
        }

        // Save the user in the database
        saveUserToDatabase(username, password);
    }
}

// ActionListener for the Back to Login button
class CancelAction implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        dispose(); // Close the signup page
        new LoginPage(); // Show the login page
    }
}

// Method to save the user data into the database
private void saveUserToDatabase(String username, String password) {
    String url = "jdbc:mysql://localhost/student_management";
    String dbUsername = "root"; // Your MySQL username
    String dbPassword = "2105"; // Your MySQL password
    String insertQuery = "INSERT INTO users (username, password) VALUES (?, ?)";
```

```
        try (Connection conn = DriverManager.getConnection(url, dbUsername, dbPassword);
             PreparedStatement stmt = conn.prepareStatement(insertQuery)) {

            stmt.setString(1, username);
            stmt.setString(2, password);

            // Execute the insert query
            int rowsAffected = stmt.executeUpdate();

            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(SignupPage.this, "Sign Up Successful!");
                dispose();
                new LoginPage(); // Redirect to login page after successful signup
            } else {
                JOptionPane.showMessageDialog(SignupPage.this, "Error occurred while saving user.");
            }

        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(SignupPage.this, "Database connection error: " +
ex.getMessage());
            ex.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new SignupPage(); // Show signup page
    }
}
```

2. Home Page

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import java.io.File;
import java.io.IOException;

public class HomePage extends JFrame {

    private ImagePanel imagePanel;
    private Timer slideshowTimer;
    private int currentImageIndex = 0;
    private float alpha = 1.0f; // Transparency level (1.0 is fully opaque)
    private final int FADE_SPEED = 50; // Adjust fade speed (increase delay for slower fade)
    private final float FADE_INCREMENT = 0.02f; // Adjust fade increment (smaller values for slower fade)
    private boolean fadingOut = true; // Track whether we're fading out or in

    // List of image file paths for the slideshow
    private String[] imagePaths = {
        "C:\\Users\\Shyaan Shaikh\\OneDrive\\Documents\\Shayaan Shaikh\\Syllabus\\Java\\rec.jpg",
        "C:\\Users\\Shyaan Shaikh\\OneDrive\\Documents\\Shayaan Shaikh\\Syllabus\\Java\\images.jpg",
        "C:\\Users\\Shyaan Shaikh\\OneDrive\\Documents\\Shayaan Shaikh\\Syllabus\\Java\\images2.jpg",
        "C:\\Users\\Shyaan Shaikh\\OneDrive\\Documents\\Shayaan Shaikh\\Syllabus\\Java\\images3.jpg",
        "C:\\Users\\Shyaan Shaikh\\OneDrive\\Documents\\Shayaan Shaikh\\Syllabus\\Java\\image4.jpg"
    };

    public HomePage(String username) {
        // Set up the frame
        setTitle("Home Page");
        setSize(700, 400); // Further increase frame width for a landscape layout
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
setLayout(new BorderLayout());

// Background color set to dull white
Color dullWhite = new Color(245, 245, 245);  // Dull white color

// Button color set to medium purple
Color mediumPurple = new Color(102, 51, 153);  // Medium purple color

// Create buttons panel for Home, Academics, Result, Edit, and List at the top
JPanel buttonPanel = new JPanel(new GridLayout(1, 5, 10, 10));
buttonPanel.setBackground(dullWhite);  // Set background color to dull white

// Create each button with medium purple color
JButton homeButton = createButton("Home", mediumPurple);
JButton academicsButton = createButton("Academics", mediumPurple);
JButton resultButton = createButton("Result", mediumPurple);
JButton editButton = createButton("Edit", mediumPurple);
JButton listButton = createButton("List", mediumPurple);

// Add action listeners to buttons
homeButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      // Reload the page or reset the slideshow
      currentImageIndex = 0; // Reset to the first image
      startSlideshowWithFadeEffect(); // Restart the slideshow
   }
});

academicsButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      // Navigate to the Academics page (replace with actual page)
      new AcademicsPage(); // Assuming you have a separate page for Academics
```

```
        dispose(); // Close current HomePage
    }
});

resultButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Navigate to the Result page (replace with actual page)
        new ResultPage(); // Assuming you have a separate page for Results
        dispose(); // Close current HomePage
    }
});

editButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Open the EditPage for editing
        new EditPage(); // Assuming you have a separate page for editing students
        dispose(); // Close current HomePage
    }
});

listButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Open the ListPage for displaying students
        new StudentListPage(); // Assuming you have a page for student list
        dispose(); // Close current HomePage
    }
});

// Add buttons to the panel
buttonPanel.add(homeButton);
buttonPanel.add(academicsButton);
buttonPanel.add(resultButton);
```

```
    buttonPanel.add(editButton);
    buttonPanel.add(listButton);
    add(buttonPanel, BorderLayout.NORTH);

    // Create the ImagePanel for fading effect in the center
    imagePanel = new ImagePanel();
    add(imagePanel, BorderLayout.CENTER);

    // Create a new panel for the logout button at the bottom right
    JPanel logoutPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
    logoutPanel.setBackground(dullWhite);  // Match background color

    // Change the logout button to red color
    Color redColor = new Color(255, 0, 0);  // Red color
    JButton logoutButton = createButton("Logout", redColor);
    logoutButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        // Perform logout action (close HomePage and open LoginPage)
        new LoginPage(); // Assuming you have a LoginPage to navigate back to
        dispose(); // Close current HomePage
      }
    });

    logoutPanel.add(logoutButton);
    add(logoutPanel, BorderLayout.SOUTH);  // Place the logout panel at the bottom

    // Start the slideshow with fade effect
    startSlideshowWithFadeEffect();

    setVisible(true);
  }
```

```
// Start the slideshow with a fade effect
private void startSlideshowWithFadeEffect() {
    slideshowTimer = new Timer(FADE_SPEED, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (fadingOut) {
                alpha -= FADE_INCREMENT; // Decrease alpha for fade out
                if (alpha <= 0) {
                    fadingOut = false; // Switch to fade in
                    currentImageIndex = (currentImageIndex + 1) % imagePaths.length;
                    updateImage();
                }
            } else {
                alpha += FADE_INCREMENT; // Increase alpha for fade in
                if (alpha >= 1) {
                    fadingOut = true; // Start fading out again
                }
            }
            imagePanel.repaint();
        }
    });
    updateImage();
    slideshowTimer.start();
}

// Helper method to update the image for the next slide
private void updateImage() {
    imagePanel.setImage(loadImage(imagePaths[currentImageIndex]));
    alpha = 0.0f; // Start new image at transparent state
}

// Load image from the file system
```

```
private BufferedImage loadImage(String path) {
    try {
        return ImageIO.read(new File(path));
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}

// Helper method to create styled buttons with specific colors
private JButton createButton(String text, Color color) {
    JButton button = new JButton(text);
    Font boldFont = new Font("Verdana", Font.BOLD, 10);
    button.setFont(boldFont);
    button.setBackground(color);
    button.setForeground(Color.WHITE);
    button.setFocusPainted(false);
    button.setPreferredSize(new Dimension(80, 30));
    return button;
}

// Inner class to handle image drawing with alpha transparency
private class ImagePanel extends JPanel {
    private BufferedImage image;

    public void setImage(BufferedImage image) {
        this.image = image;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (image != null) {
            Graphics2D g2d = (Graphics2D) g;
            g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, alpha));
            // Draw image with further increased width (600 px) and reduced height (200 px) centered
            int x = (getWidth() - 600) / 2;
            int y = (getHeight() - 200) / 2;
            g2d.drawImage(image, x, y, 600, 200, null);
        }
    }
}

    public static void main(String[] args) {
        new HomePage("User");
    }
}
```

3. Edit Page

```
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 EditPage extends JFrame {
    private JTextField studentIdField, studentNameField, studentMajorField, studentGPAField;

    public EditPage() {
        // Set up the frame
        setTitle("Edit Student Information");
```

```
setSize(500, 400);
setLocationRelativeTo(null);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new BorderLayout());

// Create a panel to hold student input fields
JPanel inputPanel = new JPanel(new GridLayout(5, 2, 10, 10));
inputPanel.setPreferredSize(new Dimension(500, 150));

JLabel studentIdLabel = new JLabel("Student ID:");
JLabel studentNameLabel = new JLabel("Name:");
JLabel studentMajorLabel = new JLabel("Major:");
JLabel studentGPALabel = new JLabel("GPA:");

studentIdField = new JTextField(15);
studentNameField = new JTextField(15);
studentMajorField = new JTextField(15);
studentGPAField = new JTextField(15);

inputPanel.add(studentIdLabel);
inputPanel.add(studentIdField);
inputPanel.add(studentNameLabel);
inputPanel.add(studentNameField);
inputPanel.add(studentMajorLabel);
inputPanel.add(studentMajorField);
inputPanel.add(studentGPALabel);
inputPanel.add(studentGPAField);

add(inputPanel, BorderLayout.NORTH);

// Create buttons panel for Add and Back
JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 20));
```

```
    // Change "Add" button to medium purple color
    Color mediumPurple = new Color(102, 51, 153);  // Medium purple color
    JButton addButton = createButton("Add", mediumPurple);
    buttonPanel.add(addButton);

    add(buttonPanel, BorderLayout.CENTER);

    // Create Back button in the bottom right corner with a red background
    JButton backButton = createButton("Back", Color.RED);  // Set the Back button color to red
    JPanel backButtonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
    backButtonPanel.add(backButton);
    add(backButtonPanel, BorderLayout.SOUTH);

    // Action listeners for buttons
    addButton.addActionListener(new AddButtonListener());
    backButton.addActionListener(new BackButtonListener());

    setVisible(true);
}

private JButton createButton(String text, Color backgroundColor) {
    JButton button = new JButton(text);
    Font boldFont = new Font("Verdana", Font.BOLD, 12);
    button.setFont(boldFont);
    button.setBackground(backgroundColor);
    button.setForeground(Color.WHITE);  // White text color
    button.setFocusPainted(false);
    button.setPreferredSize(new Dimension(100, 30));
    return button;
}
```

```
    // ActionListener for the "Add" button
    class AddButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String studentId = studentIdField.getText();
            String studentName = studentNameField.getText();
            String studentMajor = studentMajorField.getText();
            String studentGPA = studentGPAField.getText();

            if (studentId.isEmpty() || studentName.isEmpty() || studentMajor.isEmpty() ||
studentGPA.isEmpty()) {
                JOptionPane.showMessageDialog(EditPage.this, "Please fill all fields.");
                return;
            }

            // Insert student data into the database
            try (Connection conn = DBConnection.getConnection()) {
                String sql = "INSERT INTO students (student_id, name, major, gpa) VALUES (?, ?, ?, ?)";
                try (PreparedStatement stmt = conn.prepareStatement(sql)) {
                    stmt.setString(1, studentId);
                    stmt.setString(2, studentName);
                    stmt.setString(3, studentMajor);
                    stmt.setString(4, studentGPA);
                    stmt.executeUpdate();
                    JOptionPane.showMessageDialog(EditPage.this, "Student added to database!");
                }
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(EditPage.this, "Error saving student to the database.");
                ex.printStackTrace();
            }

            studentIdField.setText("");
            studentNameField.setText("");
            studentMajorField.setText("");
            studentGPAField.setText("");
        }
    }

    // ActionListener for the "Back" button
    class BackButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            dispose();  // Close the current EditPage
            new HomePage("User");  // Open the home page with a placeholder username
        }
    }

    public static void main(String[] args) {
        new EditPage();  // Launch the EditPage when executed
    }
}
```

4. Academics Page

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

public class AcademicsPage extends JFrame {

    private JTextField searchField;
    private JTextArea resultArea;
```

```
// Database credentials
private static final String DB_URL = "jdbc:mysql://localhost:3306/student_management";
private static final String USER = "root";  // Replace with your DB username
private static final String PASSWORD = "2105";  // Replace with your DB password

public AcademicsPage() {
    setTitle("Academics Page");
    setSize(500, 400);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());  // Use BorderLayout for easy positioning

    // Panel for the search bar
    JPanel searchPanel = new JPanel();
    JLabel label = new JLabel("Search by Student ID or Name:");
    label.setFont(new Font("Verdana", Font.PLAIN, 14));

    searchField = new JTextField(20);
    searchPanel.add(label);
    searchPanel.add(searchField);

    // Text area to display search results
    resultArea = new JTextArea(10, 40);
    resultArea.setEditable(false);
    resultArea.setFont(new Font("Verdana", Font.PLAIN, 12));
    JScrollPane scrollPane = new JScrollPane(resultArea);

    // Add search panel to the top of the frame
    add(searchPanel, BorderLayout.NORTH);

    // Add text area for results in the center
    add(scrollPane, BorderLayout.CENTER);
```

```
// Create search and back buttons (placed at the bottom)
JButton searchButton = new JButton("Search");
JButton backButton = new JButton("Back");

// Set properties for the buttons
searchButton.setPreferredSize(new Dimension(80, 30)); // Set button size
searchButton.setBackground(new Color(102, 51, 153)); // Set background color to purple
searchButton.setForeground(Color.WHITE); // Set text color to white
searchButton.setFont(new Font("Verdana", Font.BOLD, 12)); // Set button font

backButton.setPreferredSize(new Dimension(80, 30)); // Set button size
backButton.setBackground(Color.RED); // Set red background for back button
backButton.setForeground(Color.WHITE); // Set text color to white
backButton.setFont(new Font("Verdana", Font.BOLD, 12)); // Set button font

// Add buttons to the bottom panel
JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));  // Center the buttons
buttonPanel.setBackground(Color.WHITE);  // Set background to white
buttonPanel.add(searchButton);
buttonPanel.add(backButton);  // Add the back button as well
add(buttonPanel, BorderLayout.SOUTH);

// Action listener for the search button
searchButton.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      String query = searchField.getText().trim();
      if (!query.isEmpty()) {
         String result = getStudentInfo(query);
         resultArea.setText(result);  // Display the result in the text area
```

```
          } else {
              resultArea.setText("Please enter a Student ID or Name.");
          }
      }
  });

  // Action listener for the back button
  backButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
          dispose();  // Close the current AcademicsPage
          new HomePage("User");  // Open the HomePage with a placeholder username
      }
  });

  setVisible(true);
}

// Method to fetch student info from the database
private String getStudentInfo(String query) {
  String result = "";
  Connection conn = null;
  PreparedStatement stmt = null;
  ResultSet rs = null;

  try {
      // Establish a connection to the database
      conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);

      // Query to search by Student ID or Name
      String sql = "SELECT * FROM students WHERE student_id = ? OR name = ?";
      stmt = conn.prepareStatement(sql);
```

```
        stmt.setString(1, query);
        stmt.setString(2, query);

        // Execute query
        rs = stmt.executeQuery();

        // Process the result
        if (rs.next()) {
          result = "Student ID: " + rs.getString("student_id") + "\n" +
                   "Name: " + rs.getString("name") + "\n" +
                   "Major: " + rs.getString("major") + "\n" +
                   "GPA: " + rs.getString("gpa");
        } else {
          result = "No student found with ID or Name: " + query;
        }
      } catch (SQLException e) {
        e.printStackTrace();
        result = "Database error: " + e.getMessage();
      } finally {
        try {
          if (rs != null) rs.close();
          if (stmt != null) stmt.close();
          if (conn != null) conn.close();
        } catch (SQLException e) {
          e.printStackTrace();
        }
      }

      return result;
    }

    public static void main(String[] args) {
        new AcademicsPage();  // Launch the AcademicsPage when executed
    }
}
```

5. **Result Page**

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

public class ResultPage extends JFrame {

    private JTextField searchField;
    private JTextArea resultArea;

    // Database credentials
    private static final String DB_URL = "jdbc:mysql://localhost:3306/student_management";
    private static final String USER = "root";  // Replace with your DB username
    private static final String PASSWORD = "2105";  // Replace with your DB password

    public ResultPage() {
        setTitle("Result Page");
        setSize(500, 400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());  // Use BorderLayout for easy positioning

        // Panel for the search bar
        JPanel searchPanel = new JPanel();
        JLabel label = new JLabel("Search by Student ID or Name:");
```

```
label.setFont(new Font("Verdana", Font.PLAIN, 14));

searchField = new JTextField(20);
searchPanel.add(label);
searchPanel.add(searchField);

// Text area to display search results
resultArea = new JTextArea(10, 40);
resultArea.setEditable(false);
resultArea.setFont(new Font("Verdana", Font.PLAIN, 12));
JScrollPane scrollPane = new JScrollPane(resultArea);

// Add search panel to the top of the frame
add(searchPanel, BorderLayout.NORTH);

// Add text area for results in the center
add(scrollPane, BorderLayout.CENTER);

// Create search and back buttons (placed at the bottom)
JButton searchButton = new JButton("Search");
JButton backButton = new JButton("Back");

// Set properties for the buttons
searchButton.setPreferredSize(new Dimension(80, 30)); // Set button size
searchButton.setBackground(new Color(102, 51, 153)); // Medium purple color for Search button
searchButton.setForeground(Color.WHITE); // Set text color to white
searchButton.setFont(new Font("Verdana", Font.BOLD, 12)); // Set button font

backButton.setPreferredSize(new Dimension(80, 30)); // Set button size
backButton.setBackground(Color.RED); // Red color for Back button
backButton.setForeground(Color.WHITE); // Set text color to white
backButton.setFont(new Font("Verdana", Font.BOLD, 12)); // Set button font
```

```
// Add buttons to the bottom panel
JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));  // Center the buttons
buttonPanel.setBackground(Color.WHITE);  // Set background to white
buttonPanel.add(searchButton);
buttonPanel.add(backButton);  // Add the back button as well
add(buttonPanel, BorderLayout.SOUTH);

// Action listener for the search button
searchButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String query = searchField.getText().trim();
        if (!query.isEmpty()) {
            String result = getStudentGPA(query);
            resultArea.setText(result);  // Display the result in the text area
        } else {
            resultArea.setText("Please enter a Student ID or Name.");
        }
    }
});

// Action listener for the back button
backButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        dispose();  // Close the current ResultPage
        new HomePage("User");  // Open the HomePage with a placeholder username
    }
});
```

```
        setVisible(true);
    }

    // Method to fetch student's GPA from the database
    private String getStudentGPA(String query) {
        String result = "";
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            // Establish a connection to the database
            conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);

            // Query to search by Student ID or Name
            String sql = "SELECT gpa FROM students WHERE student_id = ? OR name = ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, query);
            stmt.setString(2, query);

            // Execute query
            rs = stmt.executeQuery();

            // Process the result
            if (rs.next()) {
                result = "GPA: " + rs.getString("gpa");
            } else {
                result = "No student found with ID or Name: " + query;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            result = "Database error: " + e.getMessage();
        } finally {
            try {
                if (rs != null) rs.close();
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        return result;
    }

    public static void main(String[] args) {
        new ResultPage();  // Launch the ResultPage when executed
    }
}
```

6. Student List Page

```
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 StudentListPage extends JFrame {
    private DefaultListModel<String> listModel;
    private JList<String> studentList;
```

```java
public StudentListPage() {
    // Set up the frame
    setTitle("Student List");
    setSize(500, 400);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    listModel = new DefaultListModel<>();
    studentList = new JList<>(listModel);

    JScrollPane listScrollPane = new JScrollPane(studentList);
    add(listScrollPane, BorderLayout.CENTER);

    // Fetch students from the database and display them
    loadStudentList();

    // Create a delete button to delete the selected student
    JButton deleteButton = new JButton("Delete");
    deleteButton.setFont(new Font("Verdana", Font.BOLD, 12));
    deleteButton.setBackground(Color.RED);  // Set Delete button color to red
    deleteButton.setForeground(Color.WHITE);

    deleteButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            deleteSelectedStudent();
        }
    });

    // Create a back button to return to the home page with medium purple color
    JButton backButton = new JButton("Back");
```

```
        Color mediumPurple = new Color(102, 51, 153);  // Medium purple color
        backButton.setFont(new Font("Verdana", Font.BOLD, 12));
        backButton.setBackground(mediumPurple);  // Set Back button color to medium purple
        backButton.setForeground(Color.WHITE);

        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Navigate back to the HomePage (assuming you have a HomePage class)
                new HomePage("User");  // Replace "User" with the actual username if needed
                dispose();  // Close the current StudentListPage
            }
        });

        // Create a panel for the buttons (Delete and Back)
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
        buttonPanel.add(deleteButton);
        buttonPanel.add(backButton);
        add(buttonPanel, BorderLayout.SOUTH);

        setVisible(true);
    }

    private void loadStudentList() {
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "SELECT name, student_id FROM students";
            try (PreparedStatement stmt = conn.prepareStatement(sql)) {
                ResultSet rs = stmt.executeQuery();

                while (rs.next()) {
                    String studentName = rs.getString("name");
                    String studentId = rs.getString("student_id");
                    listModel.addElement(studentName + " (" + studentId + ")");
                }
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error fetching students from the database.");
            ex.printStackTrace();
        }
    }

    private void deleteSelectedStudent() {
        // Get the selected student from the list
        String selectedStudent = studentList.getSelectedValue();

        if (selectedStudent == null) {
            JOptionPane.showMessageDialog(this, "Please select a student to delete.");
            return;
        }

        // Extract student ID from the selected item (format: Name (student_id))
        String studentId = selectedStudent.substring(selectedStudent.indexOf("(") + 1,
    selectedStudent.indexOf(")"));

        // Confirm deletion
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete student " +
    selectedStudent + "?",
                "Confirm Deletion", JOptionPane.YES_NO_OPTION);

        if (confirm == JOptionPane.YES_OPTION) {
            // Delete the student from the database
            try (Connection conn = DBConnection.getConnection()) {
                String deleteSql = "DELETE FROM students WHERE student_id = ?";
                try (PreparedStatement stmt = conn.prepareStatement(deleteSql)) {
                    stmt.setString(1, studentId);
                    int rowsAffected = stmt.executeUpdate();

                    if (rowsAffected > 0) {
                        // Remove the student from the list model (UI)
                        listModel.removeElement(selectedStudent);
                        JOptionPane.showMessageDialog(this, "Student deleted successfully.");
                    } else {
                        JOptionPane.showMessageDialog(this, "Student deletion failed.");
                    }
                }
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(this, "Error deleting student from the database.");
                ex.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        new StudentListPage();  // Launch the StudentListPage when executed
    }
    }
```

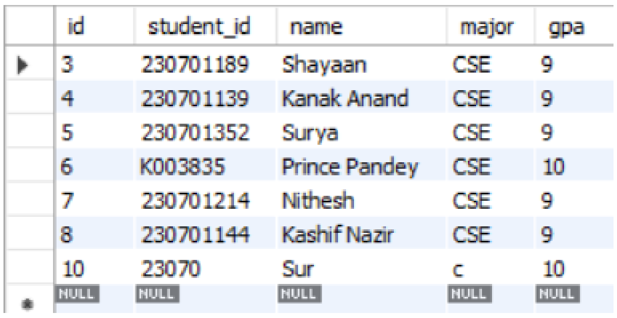
7. Java/MySQL Connectivity

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

public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:/student_management";
    private static final String USER = "root"; // Your MySQL username
private static final String PASSWORD = "2105"; // Your MySQL password
private static Connection connection;

public static Connection getConnection() {
  try {
    // Ensure MySQL driver is loaded
    Class.forName("com.mysql.cj.jdbc.Driver");

    // Create a new connection if none exists or if the connection is closed
    if (connection == null || connection.isClosed()) {
      connection = DriverManager.getConnection(URL, USER, PASSWORD);
      System.out.println("Database connection established successfully.");
    }
  } catch (ClassNotFoundException e) {
    System.out.println("MySQL JDBC Driver not found.");
    e.printStackTrace();
  } catch (SQLException e) {
    System.out.println("SQL Exception: " + e.getMessage());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("SQL State: " + e.getSQLState());
    e.printStackTrace();
  }
  return connection;
}

public static void closeConnection() {
  try {
    if (connection != null && !connection.isClosed()) {
      connection.close();
      System.out.println("Database connection closed.");
    }
  } catch (SQLException e) {
```

```
            System.out.println("Error closing database connection.");
            e.printStackTrace();
        }
    }
}

import java.sql.Connection;  // Import the Connection class from java.sql

public class App {
    public static void main(String[] args) {
        // Example usage of the DBConnection class
        Connection conn = DBConnection.getConnection(); // Use DBConnection, not dbconnection

        if (conn != null) {
            System.out.println("Database connection established successfully.");
        } else {
            System.out.println("Failed to establish database connection.");
        }

        // Closing the connection when done
        DBConnection.closeConnection(); // Use DBConnection, not dbconnection
    }
}
```

MySQL CODE

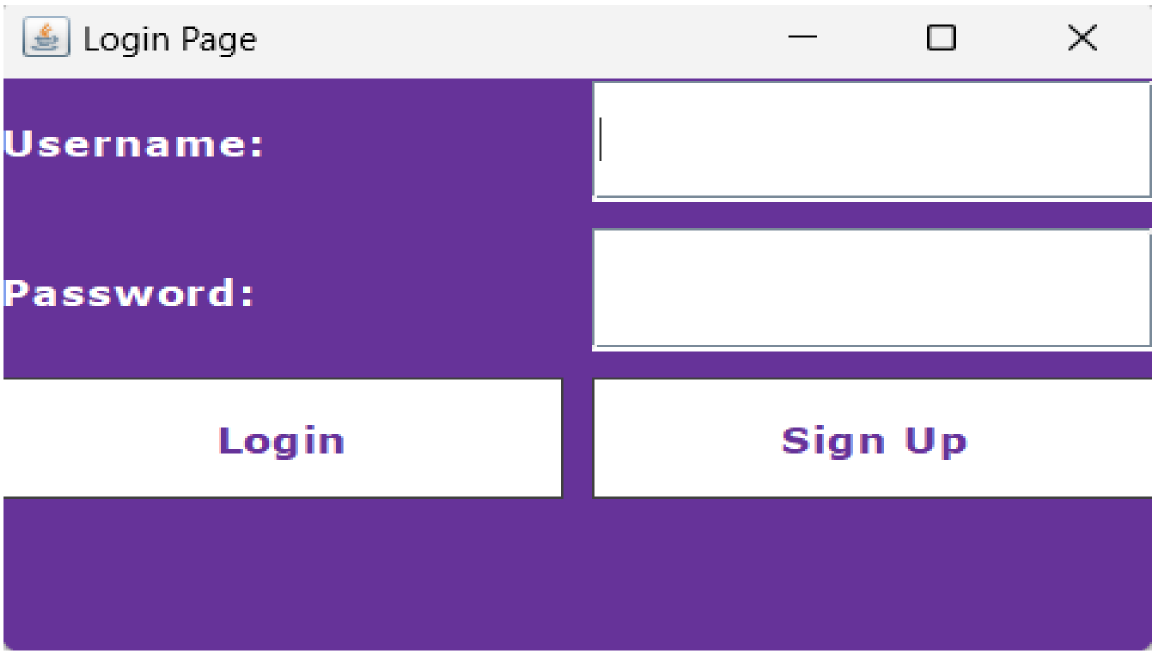
```
CREATE DATABASE student_management;

USE student_management;


CREATE TABLE users (

   username VARCHAR(50) PRIMARY KEY,

   password VARCHAR(255) NOT NULL

);


CREATE TABLE students (

   id VARCHAR(10) PRIMARY KEY,

   name VARCHAR(100),

   major VARCHAR(100),

   gpa VARCHAR(10)

);
```



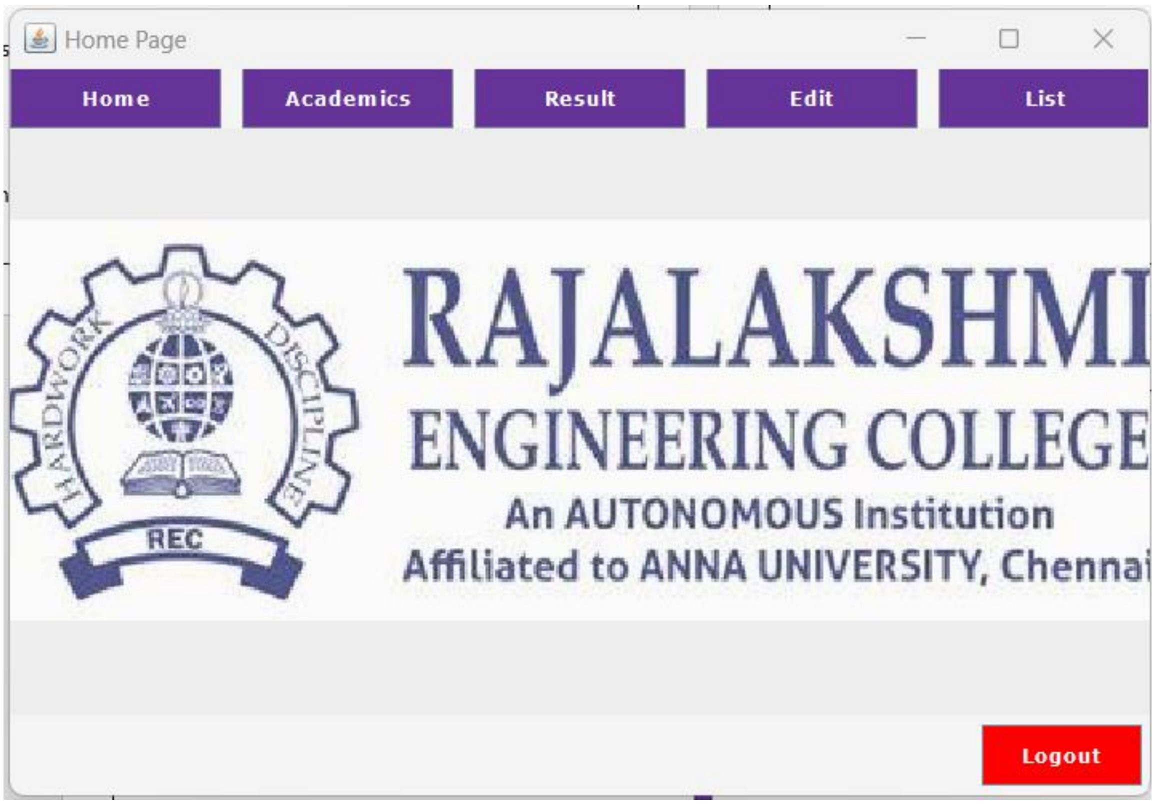
id	student_id	name	major	gpa
3	230701189	Shayaan	CSE	9
4	230701139	Kanak Anand	CSE	9
5	230701352	Surya	CSE	9
6	K003835	Prince Pandey	CSE	10
7	230701214	Nithesh	CSE	9
8	230701144	Kashif Nazir	CSE	9
10	23070	Sur	c	10
NULL	NULL	NULL	NULL	NULL

Chapter 5: RESULT AND DISCUSSION

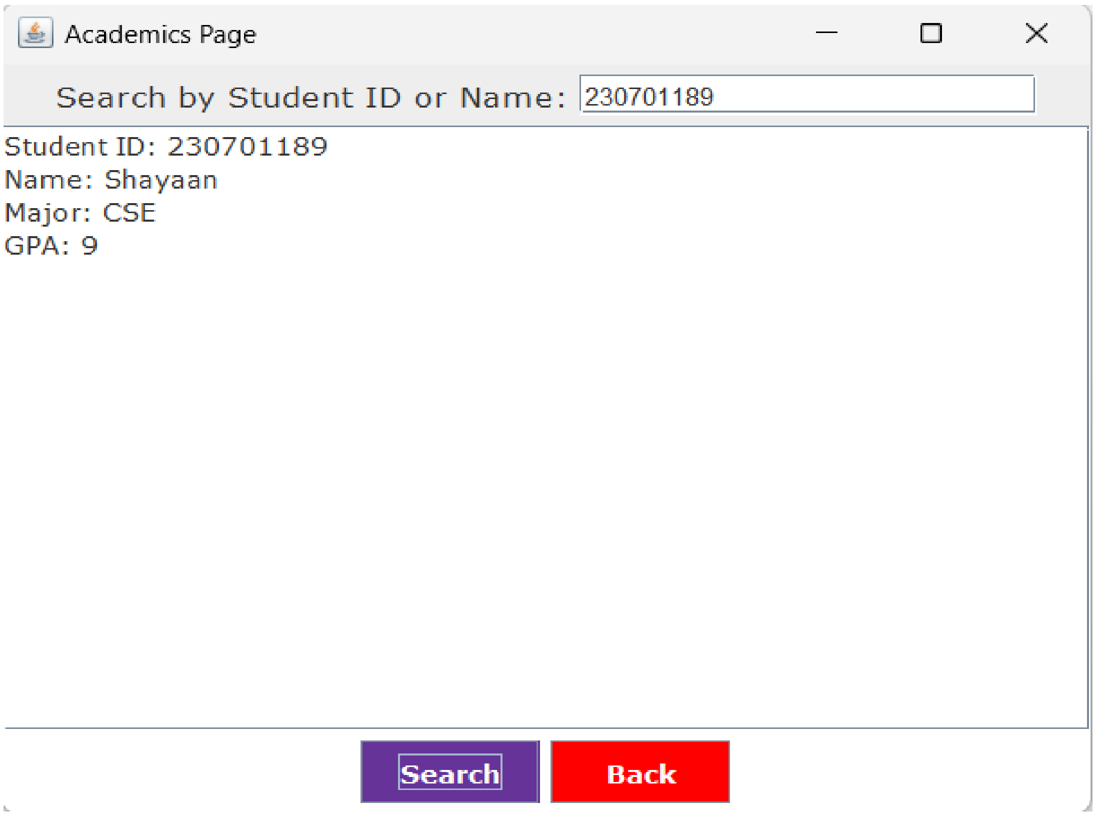
1. **Login Page:**



2. **Home Page**

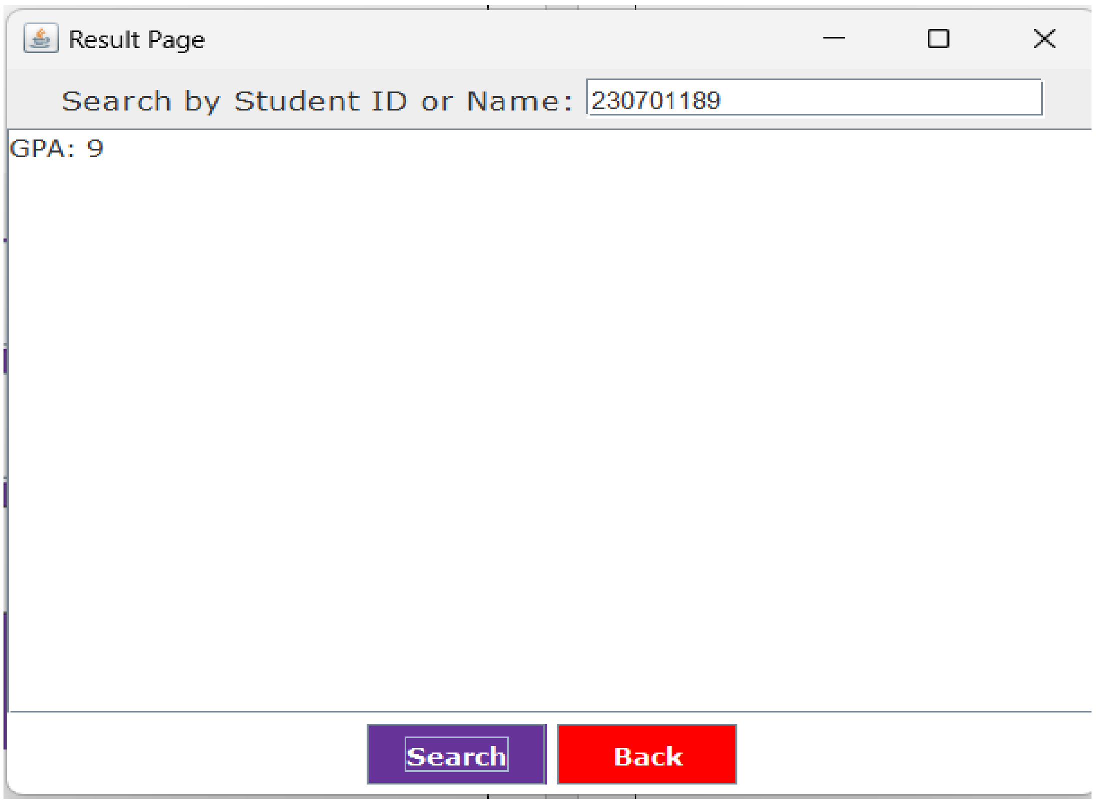


3. **Edit Page**



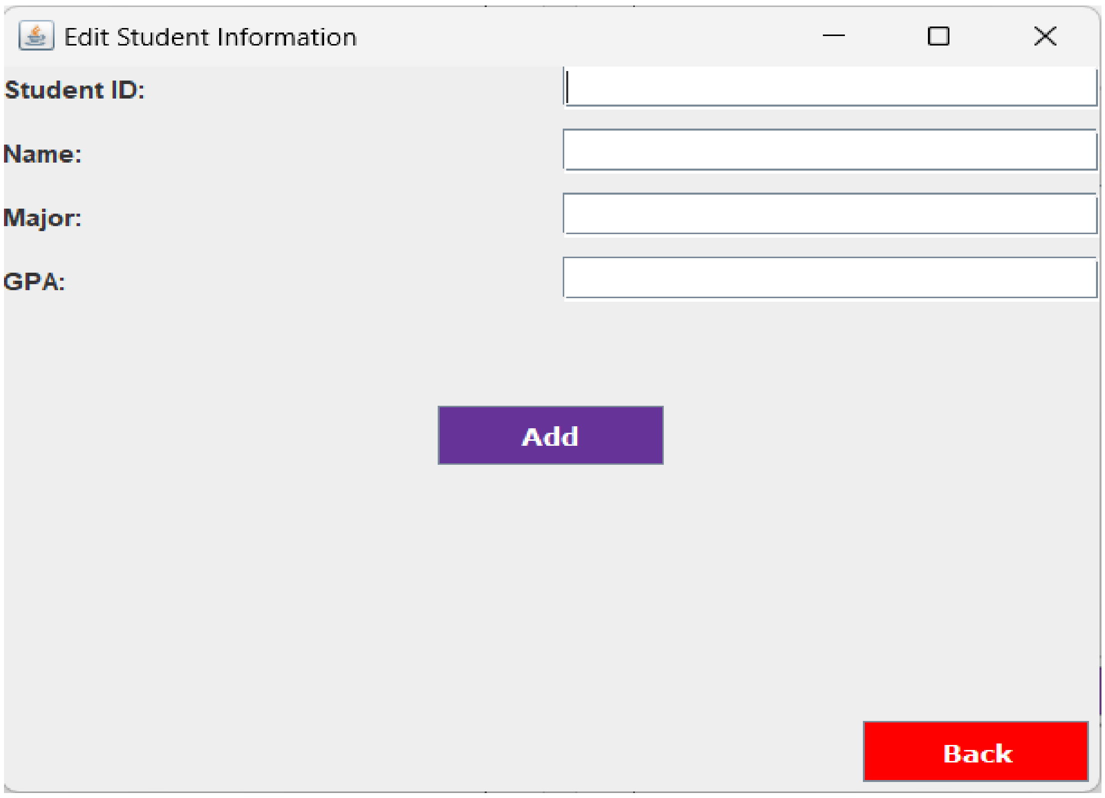
Student ID: 230701189
Name: Shayaan
Major: CSE
GPA: 9

4. **Result Page**

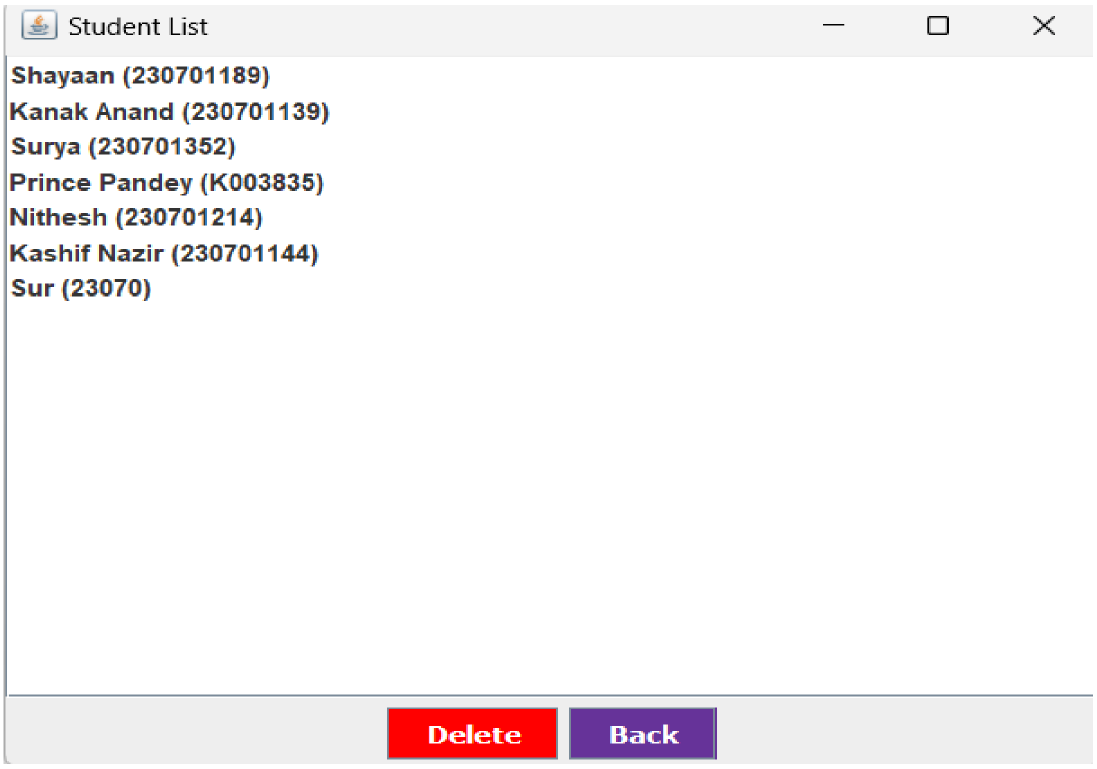


GPA: 9

5. **Adding Student Information**



6. **Student List Page**



Shayaan (230701189)
Kanak Anand (230701139)
Surya (230701352)
Prince Pandey (K003835)
Nithesh (230701214)
Kashif Nazir (230701144)
Sur (23070)

Chapter 6: Conclusion

6.1 Conclusion

The development of the **Student Management System** represents a significant advancement in academic record management, enhancing accessibility to essential student and course information. This project successfully achieves its core objectives of simplifying the management of student records, tracking academic performance, and providing a comprehensive view of student profiles, ultimately improving administrative efficiency and decision-making processes.

Built using Java with a JDBC connection to a MySQL database, the system is secure, robust, and efficient in handling real-time academic data. The use of Java JFrame ensures a user-friendly, sleek interface, making interactions such as student registration, attendance tracking, and grade updates intuitive and streamlined. MySQL contributes to efficient database management, supporting high volumes of student data with optimal performance and reliability.

With comprehensive functional capabilities, including user authentication, profile management, attendance tracking, and academic reporting, the system meets critical requirements for student management. Non-functional aspects such as security, performance, scalability, and maintainability further reinforce the system's operational stability, ensuring administrators, educators, and students experience uninterrupted, secure, and responsive service.

In conclusion, the **Student Management System** effectively addresses key challenges in academic management through an integrated approach, employing robust backend technologies and a user-focused design. It sets a standard for efficiency and security in student record management, providing a scalable solution that can adapt to future enhancements while consistently delivering an exceptional user experience and reliable academic management tools.

Chapter 7: REFERENCE

7.1 REFERENCES

[1] https://stackoverflow.com

[2] https://www.youtube.com/watch?v=OGP2R29vzAw

[3] https://www.youtube.com/watch?v=jHSBrX8lLWk