

STUDENT INFORMATION SYSTEM

CS23333 – Object Oriented Programming using Java Project Report

Submitted by

DEVDHARSHAN S G -231001031

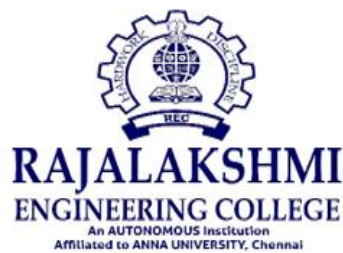
DHIYANESH S -231001036

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER-2024

BONAFIDE CERTIFICATE

Certified that this project titled “STUDENT INFORMATION SYSTEM” is the bona-fide work of “**DEVDHARSHAN S G (231001031, DHIYANESH S (231001036)**” who carried out the project work under my supervision.

SIGNATURE

Dr.P.Valarmathie

HEAD OF THE DEPARTMENT

Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE

Mrs.Usha S

COURSE INCHARGE

Assistant Professor(S.G)

Department of Information Technology
Rajalakshmi Engineering College

This project is submitted for CS23333 – Object Oriented Programming using Java held on_____

INTERNAL EXAMINAR

EXTERNAL EXAMINAR

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	List of Figures	4
	List of Tables	5
1	1.1 Abstract	6
	1.2 Introduction	6
	1.3 Purpose	6
	1.4 Scope of the project	7
	1.5 Software requirement specification	7
2	System flow Diagrams	12
	2.1 Use Case Diagram	12
	2.2 Entity-relationship Diagrams	12
	2.3 Data Flow Diagram	13
3	Module Description	15
4	4.1 Design	16
	4.2 Database Design	16
	4.3 Code	17
	4.4 Outputs	
5	Conclusion	33
6	Reference	33

LIST OF FIGURES

Figure Numbers	Figure Captions	Pg.no
2.1	Use case Diagram	12
2.2	Entity relationship diagram	13
2.3	Data flow diagram	14
4.1.1	User page	16
4.4.1	Adding Student details page	31
4.4.2	Adding student marks page	31
4.4.3	View student page	32
4.4.4	Delete student page	32

List of Tables

Figure number	Figure caption	Pg.no
4.2.1	Student table	17
4.2.2	Student mark table	17

1.1 Abstract:

The primary objective of the JDBC-powered Student Details in Java is to provide a reliable and efficient platform for users to collect the categories, store the data and view every necessary details all while leveraging the power of Java's database connectivity capabilities. This system ensures data integrity and consistency through JDBC transactions, establishing a secure connection with a relational database for real-time updates on report of the student academic status. The user-friendly interface and scalability make it an effective solution for optimizing this process in various educational environments.

1.2 Introduction:

The Student Details is a cutting-edge software solution aimed to manage the entire school data in a secure manner with acting as an integral part of the enterprise solutions that are also responsible for managing human resources and financial services functions in the institute.

This innovative system introduces automation of administrative activities like admission management, identification details, timetable creation, phone book management can take a lot of time and effort of the school administration. By seamlessly integrating technology into academic status, this system offers a more efficient, reducing stress and user-friendly experience, marking a notable departure from traditional methods.

1.3 Purpose:

The purpose of this project is to create an efficient and user-friendly Student Information System that benefits both education supervisors and student users. The system aims to:

- Simplify the process of reserving phone.
 - Storing the identification details of Students
 - Collect the necessary information per student data.
 - Producing an enhanced evaluation reports on phone
 - Store and analyze data for business insights and decision-making.

- Enhance data collection per student counting into the admission amounts for the specific institute.

1.4 Scope of the Project:

The envisioned Student Information System aims to seamlessly interact with administrators and effectively fulfil all proposed functionalities. Built on Java (JDBC) with a MYSQL database, the system ensures efficient management of student details, reducing response time for user queries. This project addresses the complexities associated with manual data processes, storing comprehensive information about phone, identification and basic details.

1.5 Software Requirement Specification:

Introduction:

The Student Information System is designed to manage all aspects of student data, including identification details and phone reports. It serves as an automated alternative to traditional manual evaluation phone report processes, enhancing efficiency in academic management.

Document Purpose:

This SRS document outlines the software requirements for the Student Information System covering design decisions, architectural design, and detailed design necessary for successful implementation. It offers insight into the system's structure and provides information crucial for ongoing software support.

Product Scope:

The Student Information System is developed for widespread use, aiming to replace outdated paper-based evaluation systems. This is significant to store, maintain, process and compile student data, keep track of their regular activities, attendance and performance and offer required guidance.

Definitions, Acronyms, and Abbreviations:

SIS – Student Information System
SRS - Software Requirements Specification.

Overall Description:

The Student Information System provides authorized users with seamless access to student records, streamlining the evaluation process for phone and necessary details in the academic environment. The system simplifies the work of academic supervisors, other educational faculties like teachers, counsellors and student along with parents.

Product Perspective:

Utilizing a client/server architecture, the system is designed to be compatible with the Microsoft Windows Operating System. The front end is developed using Java in Eclipse, while the backend leverages the MySQL server for efficient data management.

Product Functionality:

- a) Insert Student:** Facilitates the addition of new student name details.
- b) View Student:** Allows users to view and update existing student information.
- c) Add mark:** To showcase the evaluation report from the mark status.
- d) Add Id/Roll number:** Ensuring the identification details as per the student enrolment .
- e) Update:** Update the details of Student.
- f) Delete :** Delete the student details
- g) Update mark:** Used to update the marks of the student
- h) Topper :** Used to display the topper of the class.

User and Characteristics:

Qualification: Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English.

Experience: Familiarity with the university mark evaluation process is advantageous.

Technical Experience: Users are expected to have elementary knowledge of computers for optimal system interaction.

Operating Environment:

Hardware Requirements:

- Processor: Any Processor over i3
- Operating System: Windows 8, 10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

Software Requirements:

- Database: MySQL
- Frontend: Java (Eclipse,jdk-22)
- Technology: Java (JDBC)

Constraints:

- System access limited to administrators.
- Delete operation restricted to administrators without additional checks for simplicity.
- Administrators must exercise caution during deletion to maintain data consistency.

Assumptions and Dependencies:

- System administrators create and confidentially communicate phone and student details to users.

Specific Requirements:

Hardware Interface:

- Screen resolution of at least 640 x 480 or above.
- Compatible with any version of Windows 8, 10, 11.

Software Interface:

- a) MS-Windows Operating System
- b) Eclipse using Java for designing the front end

- c) MySQL for the backend
- d) Platform: Java Language(jdk-22)
- e) Integrated Development Environment (IDE): Eclipse

Functional Requirements:

1. Administrator Module (AM)

- The system displays administrative functions.
- Functions include adding and storing student details.
- The "Add" function allows administrators to input new student details which are Name, ID, Phone, Email.
- The "Search" function allows administrators to view the existing stored student details.
- All add and search requests trigger the AM module to communicate with the Server Module (SM) for necessary database changes.

2. Registered Users Module (RUM):

- After successful insertions can navigate through the application.
- Users can view detailed information about Name, Id, Email and Phone evaluation.
- Users can search and maintain student details, including modifying the categories.

3. Server Module (SM):

- SM acts as an intermediary between various modules and the database (DB).
- Receives requests from different modules and formats ps for display.
- Validates and executes requests received from other modules.
- Handles communication with the database, ensuring data consistency and integrity, especially regarding student details and phone evaluation.

Non-functional Requirements:

Performance:

- The system must handle real-time student requests efficiently, ensuring a response time of less than 2 seconds for phone evaluation and confirmation.
- Safety-critical failures, such as duplicating phones, must be addressed instantly to ensure a smooth user experience.

Reliability:

- The system is safety-critical; in case of abnormal operation or downtime, immediate measures should be taken to resolve the issue and restore normal functionality.

Availability:

- Under normal operating conditions, user requests for phone evaluation, including id and Email, should be processed within 2 seconds to maintain a seamless searching experience.
- Immediate refreshed table should appear for the convenience.

Security:

- A robust security mechanism must be in place on the server side to prevent unauthorized access, safeguarding the grading report to prevent duplication, and ensure the integrity of the Information system.
- User privacy, including personal details, must be securely stored and maintained to maintain confidentiality.

Maintainability:

- Design documents outlining software and database maintenance procedures must be available to facilitate regular updates and modifications to the grading system.

2.System Flow Diagram:

2.1. Use Case Diagram

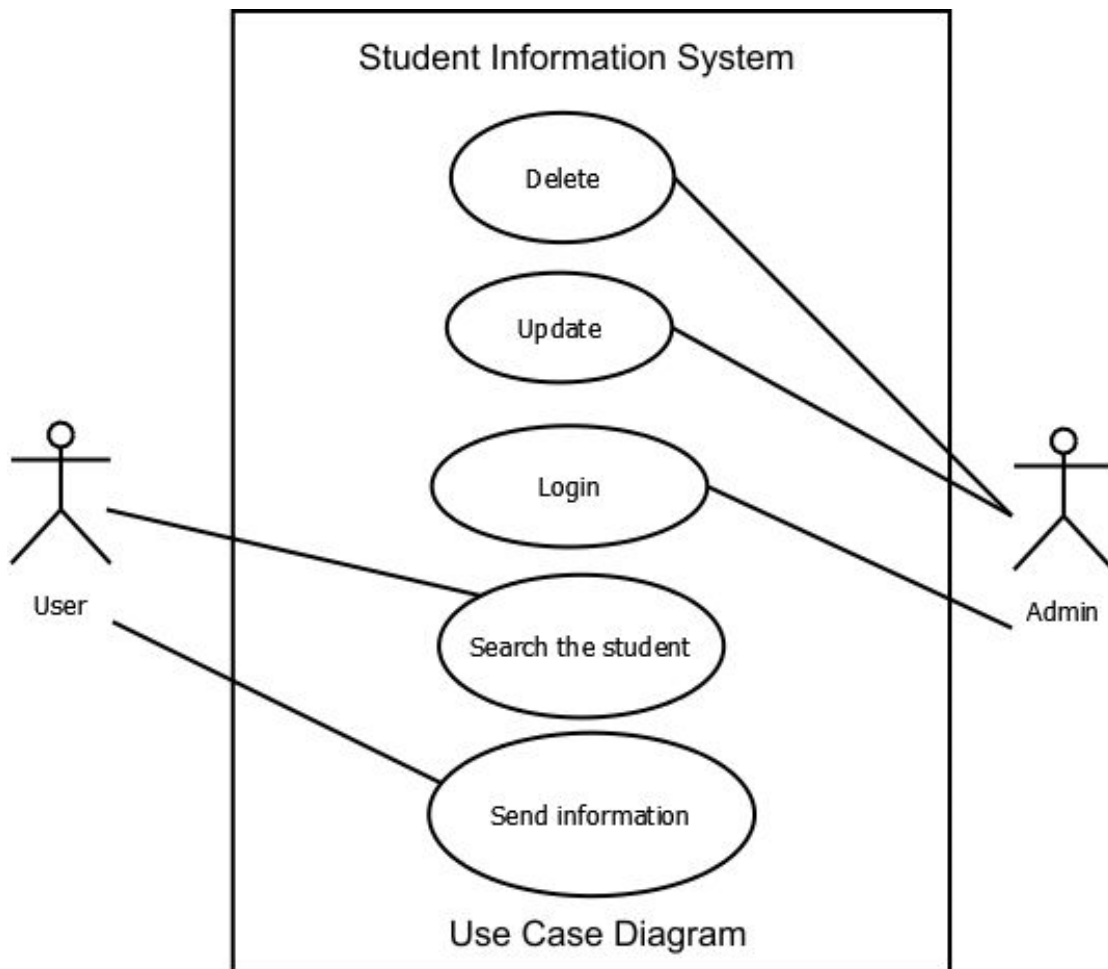


Figure 2.1 Use Case Diagram

2.2. Entity - Relationship Diagram

E-R (Entity Relationship) Diagram is used to represent the relationship between entities in table.

ENTITY RELATIONSHIP DIAGRAM FOR ADMIN

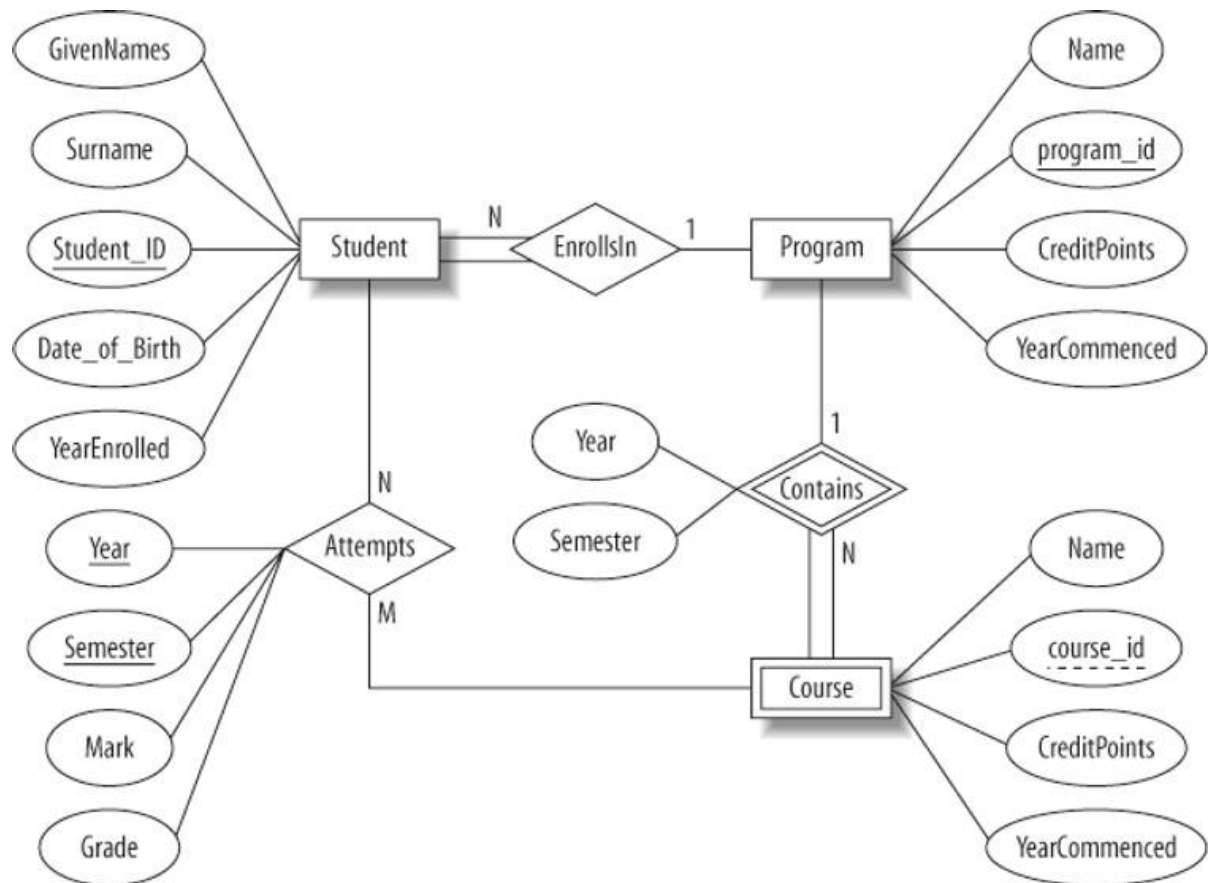


Figure 2.2-Entity Relationship Diagram

2.3 Data Flow Diagram

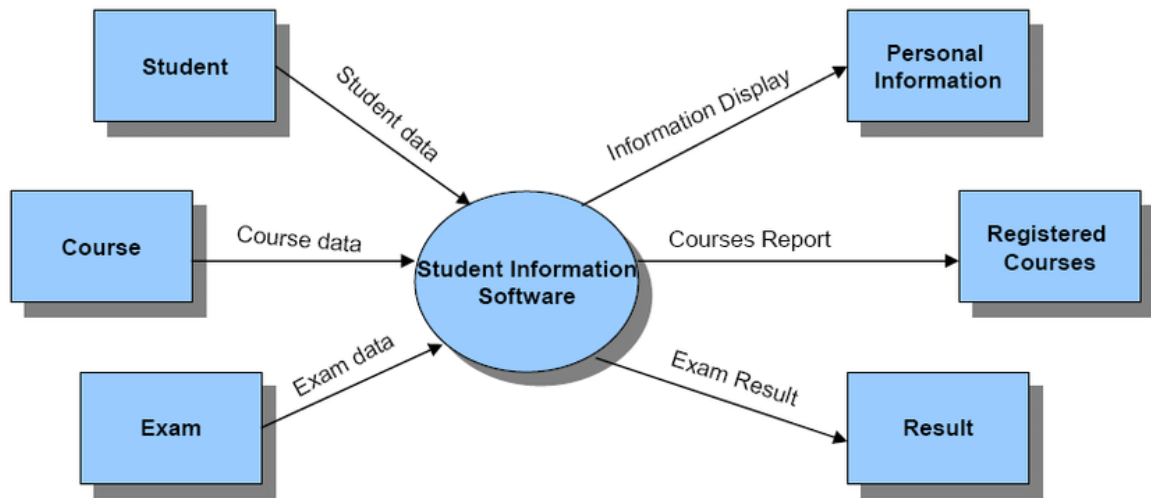


Figure 2.3- Data Flow Diagram

3.Module Description:

Add Student details

In this section, admin can add details about the student including the id, name and other relevant information.

Add Academic details

Insert details of Email and phone as per the given records.

View Student details:

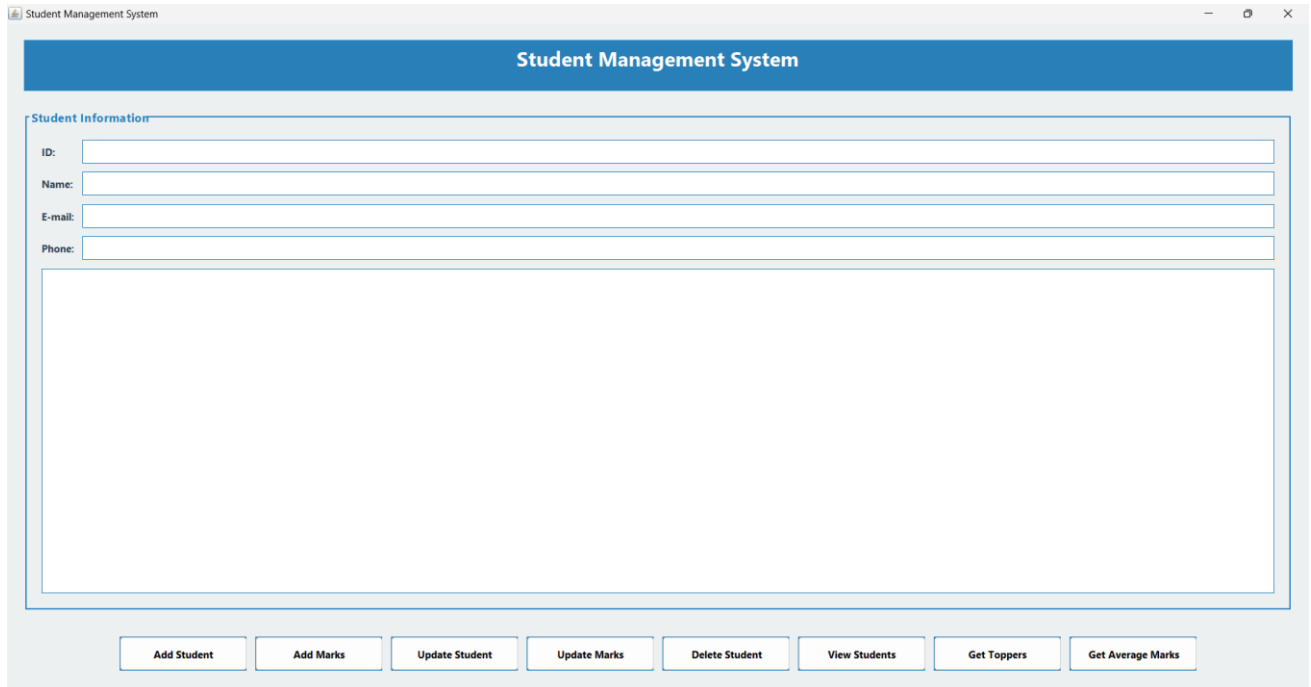
Admin can view and update Student details such as the name, phone, id etc.

Search:

To view the specific student details, the name of the student is entered and the required details are received.

4.1 Implementation of Design:

User page



Student Management System

Student Information

ID:

Name:

E-mail:

Phone:

Figure-4.1.1-User page

4.2 Database Design:

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis st. They are structured and put together to design the data store and retrieval system.

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data stor required, minimizing chances of data inconsistencies and optimizing for updates. The MySQL database has been chosen for developing the relevant databases.

Result Grid					
	id	student_id	subject1	subject2	subject3
▶	8	1	85	85	60
	9	2	35	21	13
	10	3	82	75	69
	11	4	84	67	74
•	NULL	NULL	NULL	NULL	NULL

Table4.2.1-Student Table

Result Grid				
	id	name	email	phone
▶	1	dev	dev@123	234122
	2	Dhiyanesh	Dhiyanesh@gmail.com	654322677
	3	Aadithya	Aadithya2005@gmail.com	8976885746
	4	Alfred	alfred@gmail.com	236543433
	30	dd	U@rajalakshmi.edu.in	99887655
•	NULL	NULL	NULL	NULL

Table-4.2.2-Student Mark Table

4.3 Code:

//Main class code

```
package stu.man.ger;

import javax.swing.*.*;
import java.awt.*.*;
import javax.swing.border.TitledBorder;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
```

```

import javax.swing.table.TableModel;

import java.sql.*;

public class StudentManagementSystem extends JFrame {
    private static final long serialVersionUID = 1L;
    private static final String url = "jdbc:mysql://localhost:3306/student";
    private static final String user = "root";
    private static final String password = "devdharshan";

    private JTextField idField;
    private JTextField nameField;
    private JTextField emailField;
    private JTextField phoneField;

    private JTextArea resultArea;

    private Color primaryColor = new Color(41, 128, 185); // Blue
    private Color secondaryColor = new Color(44, 62, 80); // Dark Blue
    private Color accentColor = new Color(46, 204, 113); // Green
    private Color backgroundColor = new Color(236, 240, 241); // Light Gray
    private Font labelFont = new Font("Segoe UI", Font.BOLD, 12);
    private Font buttonFont = new Font("Segoe UI", Font.BOLD, 12);
    private Font titleFont = new Font("Segoe UI", Font.BOLD, 24);
    private static final Color PRIMARY_COLOR = new Color(41, 128, 185);
    private static final Color SECONDARY_COLOR = new Color(52, 152, 219);
    private static final Color BACKGROUND_COLOR = new Color(236, 240, 241);
    //private static final Color HEADER_COLOR1= new Color(44, 62, 80);

    private Color HEADER_COLOR = new Color(44, 62, 80);
    private Color TABLE_STRIPE_COLOR = new Color(245, 247, 250);

    private Font HEADER_FONT = new Font("Segoe UI", Font.BOLD, 16);
    private Font TABLE_FONT = new Font("Segoe UI", Font.PLAIN, 14);

```

```

private Font TITLE_FONT = new Font("Segoe UI", Font.BOLD, 20);

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }

    SwingUtilities.invokeLater(() -> {
        StudentManagementSystem window = new StudentManagementSystem();
        window.setVisible(true);
    });
}

public StudentManagementSystem() {
    setTitle("Student Management System");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(800, 600);
    setLocationRelativeTo(null);

    JPanel mainPanel = new JPanel(new BorderLayout(20, 20));
    mainPanel.setBackground(backgroundColor);
    mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

    mainPanel.add(createTitlePanel(), BorderLayout.NORTH);
    mainPanel.add(createFormPanel(), BorderLayout.CENTER);
    mainPanel.add(createButtonPanel(), BorderLayout.SOUTH);

    add(mainPanel);
}

private JPanel createTitlePanel() {
    JPanel titlePanel = new JPanel();

```

```

titlePanel.setBackground(primaryColor);
titlePanel.setPreferredSize(new Dimension(800, 60));
titlePanel.setLayout(new FlowLayout(FlowLayout.CENTER));

JLabel titleLabel = new JLabel("Student Management System");
titleLabel.setFont(titleFont);
titleLabel.setForeground(Color.WHITE);
titlePanel.add(titleLabel);

return titlePanel;
}

GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;

idField = createStyledTextField();
nameField = createStyledTextField();
emailField = createStyledTextField();
phoneField = createStyledTextField();

addFormField(panel, gbc, "ID:", idField, 0);
addFormField(panel, gbc, "Name:", nameField, 1);
addFormField(panel, gbc, "E-mail:", emailField, 2);
addFormField(panel, gbc, "Phone:", phoneField, 3);

resultArea = new JTextArea(10, 10);
resultArea.setFont(new Font("Monospaced", Font.PLAIN, 12));
resultArea.setEditable(false);
resultArea.setMargin(new Insets(5, 5, 5, 5));

JScrollPane scrollPane = new JScrollPane(resultArea);
scrollPane.setBorder(BorderFactory.createLineBorder(primaryColor));

```

```

        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 20;
        gbc.weightx = 1.0;
        gbc.weighty = 1.0;
        gbc.fill = GridBagConstraints.BOTH;
        panel.add(scrollPane, gbc);

    return

    private void addFormField(JPanel panel, GridBagConstraints gbc, String labelText,
JTextField field, int row) {
        gbc.gridx = 0;
        gbc.gridy = row;
        gbc.gridwidth = 1;
        gbc.weightx = 0.0;
        JLabel label = new JLabel(labelText);
        label.setFont(labelFont);
        label.setForeground(secondaryColor);
        panel.add(label, gbc);

        gbc.gridx = 1;
        gbc.weightx = 1.0;
        panel.add(field, gbc);
    }

    private JPanel createButtonPanel() {
        JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
        panel.setBackground(backgroundColor);

        String[] buttonLabels = {"Add Student", "Add Marks", "Update Student", "Update
Marks", "Delete Student", "View Students", "Get Toppers", "Get Average Marks"};

        for (String label : buttonLabels) {

```

```

        JButton button = createStyledButton(label);
        panel.add(button);
    }

    return panel;
}

case "Update Student":
    updateStudent();
    break;
case "Update Marks":
    String studentId1 = idField.getText();
    // Ensure you use the updated studentId here for updating marks
    new UpdateMarks(studentId1).setVisible(true);
    break;
case "Delete Student":
    deleteStudent();
    String studentId11 = idField.getText(); // Get student ID from the input field
    new DeleteMarks(studentId11).setVisible(true);
    break;
case "View Students":
    viewStudents();

    break;
case "Get Toppers":
    getTopStudent();
    break;
case "Get Average Marks":
    getAverageMarks();
    break;
}
}

private void addStudent() {
    try (Connection connection = DriverManager.getConnection(url, user, password)) {

```

```

String checkQuery = "SELECT COUNT(*) FROM Students WHERE id = ?";
PreparedStatement checkStatement = connection.prepareStatement(checkQuery);
int id = Integer.parseInt(idField.getText());
checkStatement.setInt(1, id);

ResultSet resultSet = checkStatement.executeQuery();
resultSet.next();
int count = resultSet.getInt(1);

if (count > 0) {
    showMessage("A student with this ID already exists. Please use a different ID.",
"Duplicate Entry", JOptionPane.WARNING_MESSAGE);
    return; // Exit if a duplicate is found
}

String insertQuery = "INSERT INTO Students (id, name, email, phone) VALUES (?, ?,
?, ?)";
PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);
preparedStatement.setInt(1, id);
preparedStatement.setString(2, nameField.getText());
preparedStatement.setString(3, emailField.getText());
preparedStatement.setString(4, phoneField.getText());

int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
    showMessage("Student      added      successfully!",      "Success",
JOptionPane.INFORMATION_MESSAGE);
}

clearFields();
} catch (SQLException ex) {
    ex.printStackTrace();
    showMessage("Error      adding      student:      "      +      ex.getMessage(),      "Error",
JOptionPane.ERROR_MESSAGE);

```

```

    } catch (NumberFormatException ex) {
        showMessage("Please enter valid numeric values for ID and phone number!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void clearFields() {
    // TODO Auto-generated method stub

}

private void updateStudent() {
try (Connection connection = DriverManager.getConnection(url, user, password)) {
    String updateQuery = "UPDATE Students SET name = ?, phone = ?, email = ? WHERE
id = ?";
    PreparedStatement preparedStatement = connection.prepareStatement(updateQuery);
    preparedStatement.setString(1, nameField.getText());
    preparedStatement.setString(2, phoneField.getText());
    preparedStatement.setString(3, emailField.getText());
    preparedStatement.setInt(4, Integer.parseInt(idField.getText()));

    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        showMessage("Student      updated      successfully!",      "Success",
JOptionPane.INFORMATION_MESSAGE);
    } else {
        showMessage("No  student  found  with  this  ID.",  "Update  Failed",
JOptionPane.WARNING_MESSAGE);
    }

    clearFields();
} catch (SQLException ex) {
    ex.printStackTrace();
}
}

```



```

        showMessage("Error updating student: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    } catch (NumberFormatException ex) {
        showMessage("Please enter valid numeric values for ID!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void deleteStudent() {
    try (Connection connection = DriverManager.getConnection(url, user, password)) {
        String deleteQuery = "DELETE FROM Students WHERE id = ?";
        PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);
        preparedStatement.setInt(1, Integer.parseInt(idField.getText()));

        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            showMessage("Student deleted successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            showMessage("No student found with this ID.", "Delete Failed",
JOptionPane.WARNING_MESSAGE);
        }

        clearFields();
    } catch (SQLException ex) {
        ex.printStackTrace();
        showMessage("Error deleting student: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    } catch (NumberFormatException ex) {
        showMessage("Please enter a valid numeric value for ID!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

```

```
}
```

```
public void viewStudents() {
```

```
    DefaultTableModel model = new DefaultTableModel() {
```

```
        private static final long serialVersionUID = 1L;
```

```
        @Override
```

```
        public boolean isCellEditable(int row, int column) {
```

```
            return false;
```

```
        }
```

```
    };
```

```
    model.addColumn("ID");
```

```
    model.addColumn("Name");
```

```
    model.addColumn("Email");
```

```
    model.addColumn("Phone");
```

```
    model.addColumn("Subject 1");
```

```
    model.addColumn("Subject 2");
```

```
    model.addColumn("Subject 3");
```

```
    model.addColumn("Average");
```

```
    try (Connection connection = DriverManager.getConnection(url, user, password)) {
```

```
        String query = "SELECT s.*, m.subject1, m.subject2, m.subject3, " +
```

```
            "(m.subject1 + m.subject2 + m.subject3)/3 as average " +
```

```
            "FROM Students s LEFT JOIN Marks m ON s.id = m.student_id";
```

```
    try (Statement statement = connection.createStatement();
```

```
        ResultSet resultSet = statement.executeQuery(query)) {
```

```
        while (resultSet.next()) {
```

```
            Object[] row = new Object[] {
```

```

        resultSet.getInt("id"),
        resultSet.getString("name"),
        resultSet.getString("email"),
        resultSet.getString("phone"),
        resultSet.getObject("subject1"),
        resultSet.getObject("subject2"),
        resultSet.getObject("subject3"),
        resultSet.getObject("average") != null ?
            String.format("%.2f", resultSet.getDouble("average")) : "-"
    };
    model.addRow(row);
}
}
} catch (SQLException ex) {
    ex.printStackTrace();
    showMessage("Error retrieving students: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    return;
}

// Create and style the table
JTable table = createStyledTable(model);

// Create search panel
JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
searchPanel.setBackground(BACKGROUND_COLOR);
JTextField searchField = new JTextField(20);
searchField.setFont(TABLE_FONT);
JButton searchButton = createStyledButton("Search");

searchPanel.add(new JLabel("Search: "));
searchPanel.add(searchField);
searchPanel.add(searchButton);

```

```

// Create the frame
JFrame frame = createStyledFrame("Student Records", 1000, 600);

// Create title panel
JPanel titlePanel = new JPanel();
titlePanel.setBackground(PRIMARY_COLOR);
titlePanel.setPreferredSize(new Dimension(1000, 50));

JLabel titleLabel = new JLabel("Student Records Management");
titleLabel.setFont(TITLE_FONT);
titleLabel.setForeground(Color.WHITE);
titlePanel.add(titleLabel);

// Add components to frame
frame.add(titlePanel, BorderLayout.NORTH);
frame.add(searchPanel, BorderLayout.SOUTH);
frame.add(new JScrollPane(table), BorderLayout.CENTER);

frame.setVisible(true);
}

private JTable createStyledTable(DefaultTableModel model) {
    JTable table = new JTable(model);

    // Add zebra striping
    table.setDefaultRenderer(Object.class, new DefaultTableCellRenderer() {

        private static final long serialVersionUID = 1L;

        public Component getTableCellRendererComponent(JTable table,
Object value,
        boolean isSelected, boolean hasFocus, int row, int column) {
            Component c = super.getTableCellRendererComponent(table, value,
                isSelected, hasFocus, row, column);

```

```

        if (!isSelected) {
            c.setBackground(row % 2 == 0 ? Color.WHITE : TABLE_STRIPE_COLOR);
        }
        ((JLabel) c).setHorizontalAlignment(SwingConstants.CENTER);
        return c;
    }
});

return table;
}

```

```

private JFrame createStyledFrame(String title, int width, int height) {
    JFrame frame = new JFrame(title);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setSize(width, height);
    frame.setLocationRelativeTo(null);
    frame.getContentPane().setBackground(BACKGROUND_COLOR);
    return frame;
}

```

```

@SuppressWarnings("unused")
private JButton createStyledButton1(String text) {
    JButton button = new JButton(text);
    button.setFont(TABLE_FONT);
    button.setForeground(Color.WHITE);
    button.setBackground(PRIMARY_COLOR);
    button.setFocusPainted(false);
    button.setBorder(BorderFactory.createEmptyBorder(5, 15, 5, 15));

    button.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            button.setBackground(PRIMARY_COLOR.brighter());
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {

```

```

        button.setBackground(PRIMARY_COLOR);
    }
});

return button;
}

private void showMessage(String message, String title, int messageType) {
    UIManager.put("OptionPane.messageFont", TABLE_FONT);
    UIManager.put("OptionPane.buttonFont", TABLE_FONT);
    UIManager.put("OptionPane.background", BACKGROUND_COLOR);
    UIManager.put("Panel.background", BACKGROUND_COLOR);

    JOptionPane.showMessageDialog(null, message, title, messageType);
}
}

```

4.4 Outputs

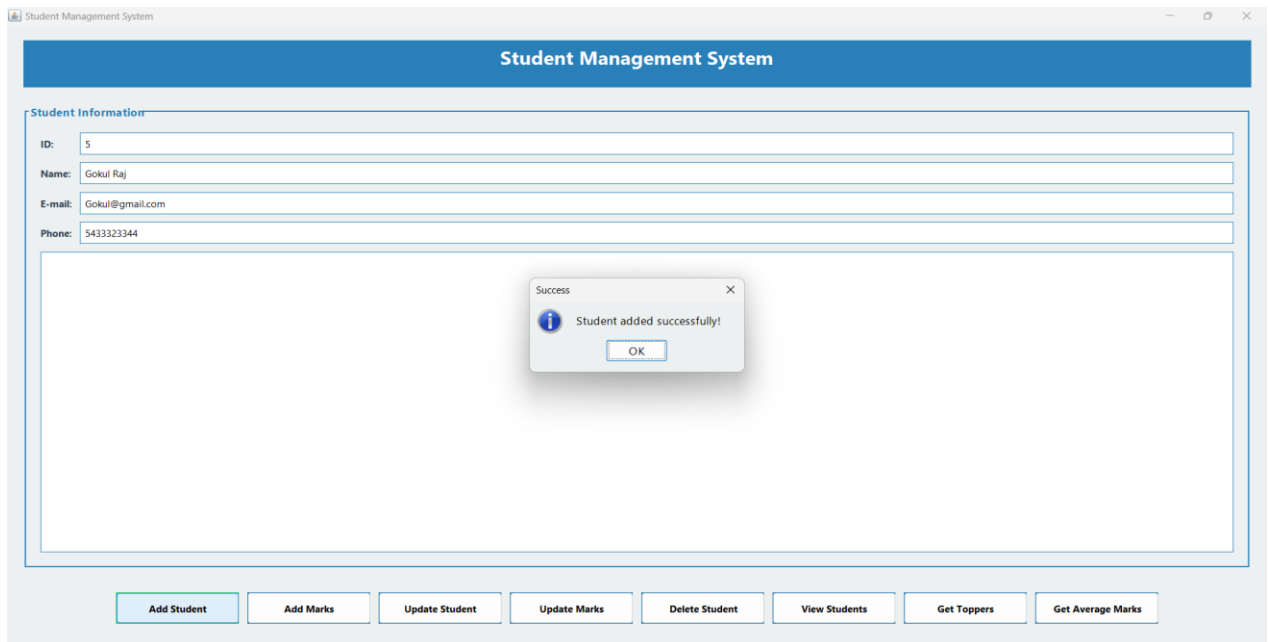


Figure-4.4.1-Adding Student page

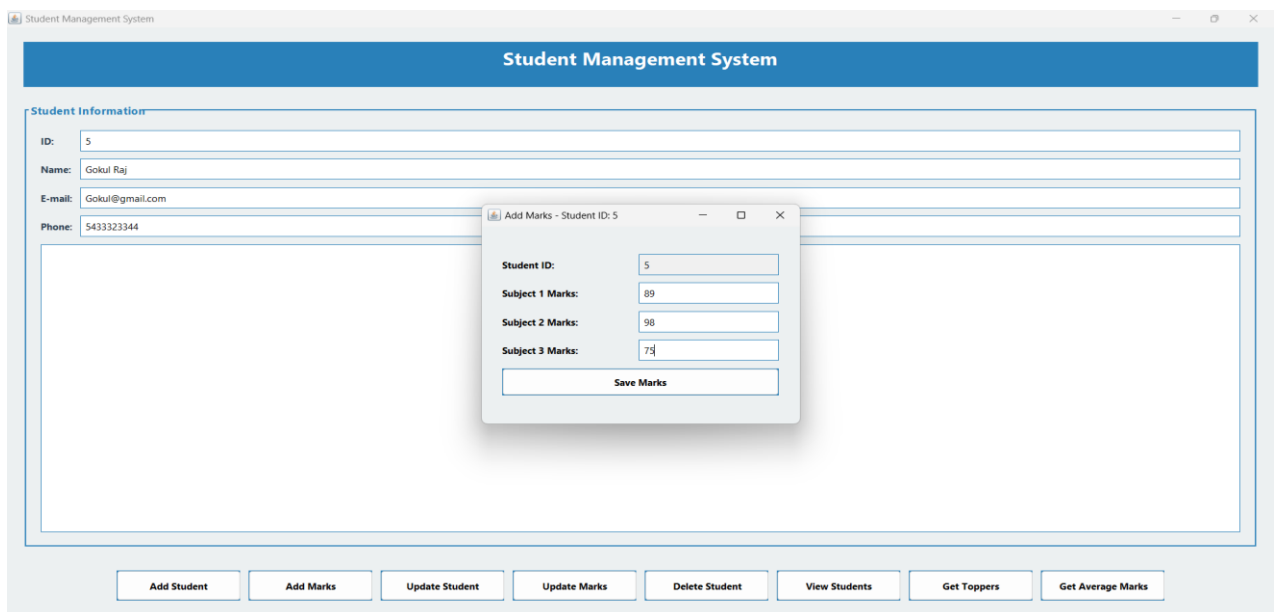


Figure-4.4.2-Adding Student Mark page

Student Records Management							
ID	Name	Email	Phone	Subject 1	Subject 2	Subject 3	Average
1	dev	dev@123	234122	85.0	85.0	60.0	76.67
2	Dhiyanesh	Dhiyanesh@gmai...	654322677	35.0	21.0	13.0	23.00
3	Aadithya	Aadithya2005@...	8976885746	82.0	75.0	69.0	75.33
4	Alfred	alfred@gmail.com	236543433	84.0	67.0	74.0	75.00
5	Gokul Raj	Gokul@gmail.com	5433323344	89.0	98.0	75.0	87.33
30	dd	U@rajalakshmi.e...	99887655				-

Search:

Figure-4.4.3-View Student page

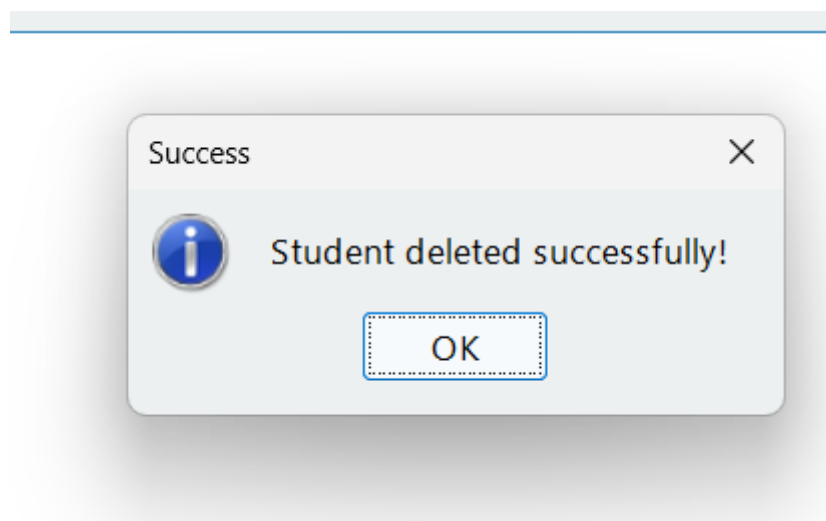


Figure-4.4.4-Delete Student page

5. Conclusion:

The Student detail project, executed under experienced guidance, embodies a meticulous approach to design and implementation. With a focus on user-friendly functionalities like adding/viewing movies and reservation management, the system ensures a seamless movie-going experience. Robust security measures, particularly in the "Remove Admin" module, highlight a commitment to data integrity and system protection. The project stands as a well-rounded solution, meeting current requirements while allowing for future adaptability and enhancements.

Reference links:

- **Java Database Connectivity with MySQL - javatpoint**
<https://www.javatpoint.com/example-to-connect-to-the-mysql-database>
- <https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/>

