

# **STUDENT MANAGEMENT SYSTEM**

**CS23333-Object Oriented Programming Using JAVA Project Report**

*Submitted by*

**HARIHARARN M -231001052**

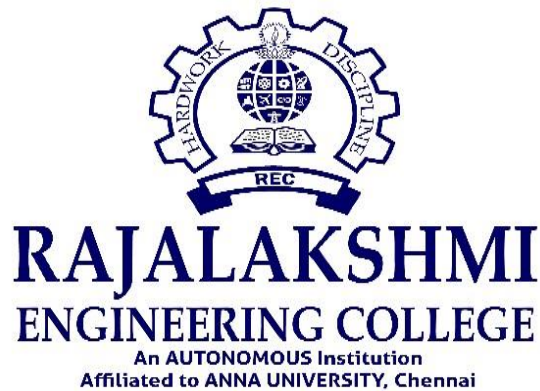
**GOBIKRISHNA J -231001045**

*Of*

**BACHELOR OF TECHNOLOGY**

*In*

**INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**RAJALAKSHMI ENGINEERING COLLEGE**

**NOVEMBER 2024**

## **BONAFIDE CERTIFICATE**

Certified that this project titled “Student Management System is the bonafide work of “**GOBIKRISHNA(231001045),HARI HARAN(231001052)**” who carried out the project work under my supervision.

**SIGNATURE**

**Dr.P.Valarmathie**

**HEAD OF THE DEPARTMENT**

Information Technology  
Rajalakshmi Engineering College  
college

**SIGNATURE**

**Mrs.Usha S**

**COURSE INCHARGE**

**Assistant professor(S.G)**

Information Technology  
Rajalakshmi Engineering

This MiniProject is submitted for CS23333-Object Oriented Programming Using JAVA held on \_\_\_\_\_

**INTERNAL EXAMINAR**

**EXTERNAL EXAMINAR**

## **Table of Contents:**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>1</b>	<b>1.1 ABSTRACT</b>	<b>5</b>
	1.2 INTRODUCTION	5
	1.3 PURPOSE	5
	1.4 SCOPE OF PROJECT	6
	1.5 SOFTWARE REQUIREMENT SPECIFICATION	6
<b>2</b>	<b>SYSTEM FLOW DIAGRAM</b>	<b>12</b>
	2.1 USE CASE DIAGRAM	12
	2.2 ENTITY RELATIONSHIP DIAGRAM	13
	2.3 DATA FLOW DIAGRAM	14

<b>3</b>	<b>MODULE DESCRIPTION</b>	<b>15</b>
<b>4</b>	<b>IMPLEMENTATION</b>	<b>16</b>
	4.1 DESIGN	16
	4.2 DATABASE DESIGN	19
	4.3 CODE	21
<b>5</b>	<b>CONCLUSION</b>	<b>26</b>
<b>6</b>	<b>REFERENCE</b>	<b>26</b>

## **ACKNOWLEDGEMENT**

First, we thank the almighty God for the successful completion of the project. Our sincere thanks to our chairman Mr.S. Meganathan, B.E., F.I.E for his sincere endeavour in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson Dr.Thangam Meganathan, for her enthusiastic motivation which inspired us a lot in completing this project and Vice-Chairman Mr. Abhay Shankar Meganathan B.E., M.S., for providing us with therequisiteinfrastructure.

We also express our sincere gratitude to our college principal, Dr.S.N.Murugesan M.E., PhD., for his kind support and facilities to complete our work on time. We extend heartfelt gratitude to Dr.P.Valarmathie, Professor and Head of the Department of Information Technology for her guidance and encouragement throughout the work. We are very glad to thank our course faculty Mrs. Usha S, Professor of our department for their encouragement and support towards the successful completion of this project. We extend our thanks to our parents, friends, all faculty members, and supporting staff for their direct and indirect involvement in the successful completion of the project for their encouragement and support.

**HARIHARAN M**  
**GOBIKRISHNA J**

# CHAPTER 1

## 1.1 Abstract:

The **JDBC-powered Student Management System in Java** is designed to provide an efficient platform for managing student data such as registration, course enrollment, attendance, and academic records. By leveraging the capabilities of Java's database connectivity, the system ensures seamless interaction with a relational database, ensuring data accuracy and real-time updates. This project emphasizes user-friendliness, scalability, and security, making it a robust solution for modern educational institutions.

## 1.2 Introduction:

The Student Management System (SMS) is a comprehensive software solution aimed at automating the administrative tasks associated with student information management. The system eliminates manual inefficiencies by providing features for adding, updating, and retrieving student data with minimal effort. Administrators are granted secure access to manage key functionalities such as student registration, attendance tracking, and grade management, ensuring a streamlined academic environment.

## 1.3 Purpose:

The purpose of this project is to create a centralized platform for managing student-related information effectively. The system aims to:

- Simplify the process of student data management.
- Ensure the accuracy and accessibility of academic records.
- Provide educational institutions with tools for attendance and grade tracking.
- Generate detailed reports to support academic decision-making.

## **1.4 Scope of the Project:**

The Student Management System ensures seamless interaction between administrators and student data repositories. Built on Java (JDBC) with a MySQL database, it efficiently manages key information such as student profiles, enrollment details, and attendance records. By incorporating validation, security, and user-friendly interfaces, the system offers a scalable solution that meets the diverse needs of educational institutions.

## **1.5 Software Requirement Specification:**

### **Introduction:**

The Student Management System automates the management of student data, offering functionalities such as course registration, attendance tracking, and performance analysis. It replaces manual processes with a structured, technology-driven approach.

### **Document Purpose:**

This SRS document details the functional and non-functional requirements of the system, providing a framework for development and future maintenance.

### **ProductScope:**

The Student Management System aims to streamline the student data management process, ensuring data consistency, reducing redundancy, and improving administrative efficiency

### **Definitions, Acronyms, and Abbreviations:**

- SMS: Student Management System
- SRS: Software Requirements Specification

**References:**

- [1] <https://www.javatpoint.com/java-awt>
- [2] <https://www.javatpoint.com/java-swing>

**Overall Description:-****Product Perspective:**

The system follows a client-server architecture, with Java AWT and Swing for the front-end and MySQL as the database backend. It runs on Microsoft Windows operating systems and integrates well with existing academic workflows.

**Product Functionality:**

1. Admin Register: Enables the registration of new administrators.
2. Admin Login: Provides secure login access for administrators.
3. Add Student: Allows administrators to add new student details.
4. View Student: Facilitates viewing and updating student information.
5. Delete Student: Permits the removal of student data.
6. Track Attendance: Enables attendance tracking and reporting.
7. Manage Grades: Provides functionality to assign and modify student grades.
8. Remove Admin: Allows the deletion of administrator accounts if required.

**User Characteristics:**

- Qualification: Basic familiarity with computer usage.
- Technical Knowledge: Minimal technical expertise required for interaction with the system.



## **Operating Environment:-**

### **Hardware Requirements:**

- Processor: Intel i3 or above
- OS: Windows 8, 10, or 11
- RAM: 4GB
- Hard Disk: 500GB

### **Software Requirements:**

- Database: MySQL
- Frontend: Java (AWT, Swing)
- Platform: Java Language

### **Constraints:**

- Only administrators have access to sensitive operations.
- Data deletion lacks a multi-level confirmation process, necessitating careful operation.

### **Assumptions:**

- Login credentials will be securely shared with administrators.

## **USER INTERFACE:-**

### **Login Page**

#### **Components:**

1. Heading: "Student Management System Login"
2. Username Label and Textbox
3. Password Label and Password Field
4. Buttons:
  - Login
  - Reset

## **Main Dashboard**

### **Components:**

1. Heading: "Admin Dashboard"
2. Menu Bar:
  - Student Management
    - Add Student
    - View Students
    - Update Student
    - Delete Student
  - Grade Management
    - Add/Update Grades
    - View Grades
  - Admin Options
    - Add Admin
    - Remove Admin
  - Logout

### **Layout:**

- Left Panel: Navigation menu (Student Management, Grade Management, etc.)
- Main Panel: Displays content based on the selected menu option.

## **Add Student Page**

### **Components:**

1. Heading: "Add New Student"
2. Input Fields:
  - Student Name
  - Roll Number
  - Age
  - Course/Department
  - Contact Details
3. Buttons:
  - Save
  - Reset
  - Cancel

## **View/Update Students Page**

### **Components:**

1. Heading: "Student Details"
2. Search Box: Search by Roll Number/Name
3. Table Display:
  - Columns: Roll No., Name, Age, Course, Contact
4. Buttons:
  - Edit
  - Delete

## **Grade Management Page**

### **Components:**

1. Heading: "Grade Management"
2. Input Fields:
  - Roll Number
  - Course Name
  - Grade (Drop-down with options: A, B, C, D, F)
3. Buttons:
  - Save
  - Update
  - Clear

## **Hardware Interface for Student Management System (SMS)**

### **1. Server Requirements**

- Processor: Intel Xeon/AMD EPYC (Quad-Core+)
- RAM: 16GB+
- Storage: 500GB SSD+
- Network: 1 Gbps Ethernet
- Backup: UPS for power continuity

### **2. Client Devices**

- Desktops/Laptops: Intel Core i5, 4GB RAM, 256GB SSD
- Mobile Devices: Android/iOS (2GB RAM, 16GB storage)
- Network: Wi-Fi or LAN

### **3. Networking**

- Router: Dual-band (802.11ac+)
- Switch: For wired connections
- Access Points: Campus-wide Wi-Fi

### **4. Peripheral Devices**

- Printers: For reports/IDs
- Biometric Scanners: Attendance tracking
- Barcode Scanners: ID/library management

### **5. Backup Systems**

- Devices: External HDDs or NAS
- Power: UPS or inverter for uninterrupted operation

## **Software Interface for Student Management System**

### **Operating System:**

#### **1. Server:**

- Server: Windows Server 2019/2022 or Linux
- Client: Windows 10/11, macOS, Linux, Android, iOS

#### **2. Front-End Tools:**

- Java (Swing/AWT for desktop GUI)
- HTML/CSS/JavaScript for web-based UI

#### **3. Back-End Tools:**

- Database: MySQL or PostgreSQL
- Connectivity: JDBC

#### **4. Development Tools:**

- IDE: NetBeans/Eclipse
- Version Control: Git

#### **5. Security:**

- Role-based login, encrypted passwords
- SSL/TLS for secure communication

#### **6. Integration:**

- ERP systems, notifications via email/Google Calenda

## **Functional Requirements:**

### **1. Login Module (LM):**

- Ensures secure access to the system through username and password authentication.
- Supports role-based login for different user categories, such as students, teachers, and administrators.
- Includes password masking and recovery features for enhanced security.

### **2. Student Management Module (SMM):**

- Facilitates the registration of new students by collecting details like name, contact information, and enrollment data.
- Allows updating of student records, including personal details, course registrations, and address changes.
- Provides functionality for deleting records of students who have graduated or left the institution.

### **3. Grade Management Module (GMM):**

- Enables the entry and modification of student grades for various subjects and exams.
- Provides tools for analyzing academic performance through graphical summaries and reports.
- Supports the export of grade sheets and transcripts for student reference or administrative use.

## **Non-Functional Requirements:**

### **1. Performance:**

- The system should provide a fast and responsive user experience with response times of under 2 seconds for any action, such as loading student data, updating records, or generating reports.
- It must handle concurrent users efficiently, especially during peak usage times (e.g., during registration periods).
- The system should be optimized for quick data retrieval and updates, ensuring minimal delays in operations.

### **2. Reliability:**

- The system must maintain high availability with minimal downtime, even during system updates or maintenance periods.
- It should include automated backup procedures to prevent data loss and ensure quick recovery in case of system failures.
- Consistency and integrity of student records must be maintained at all times, even during concurrent user operations or transactions.

### **3. Security:**

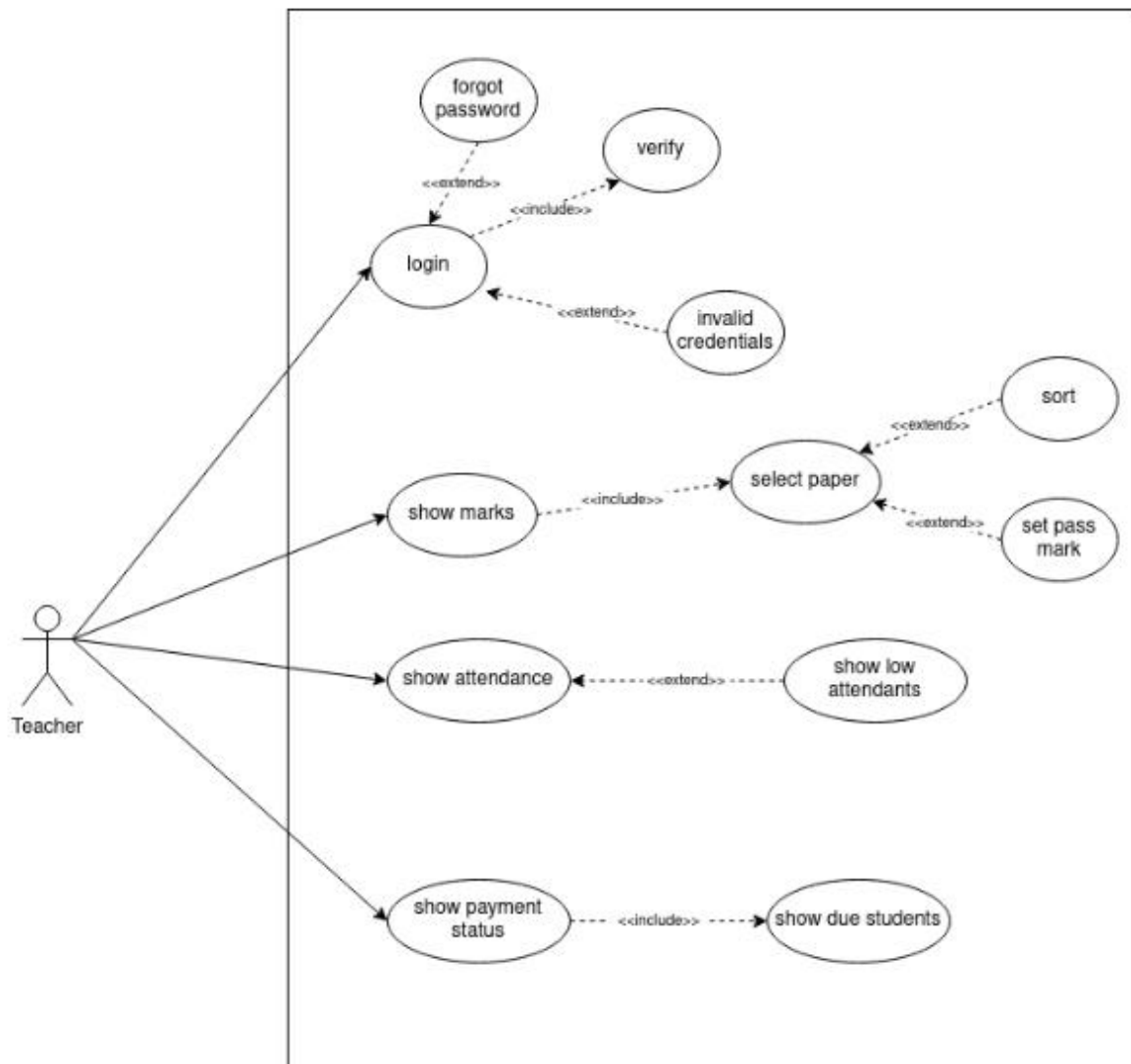
- The system must ensure the secure storage of sensitive student data, including personal information, attendance, and grades.
- Data encryption should be applied for both data at rest and data in transit to prevent unauthorized access or data breaches.
- Role-based access control (RBAC) must be implemented to restrict system access based on user roles (e.g., administrators, teachers, students).
- Regular security audits and compliance with data protection regulations (e.g., GDPR, FERPA) should be conducted

### **4. Maintainability:**

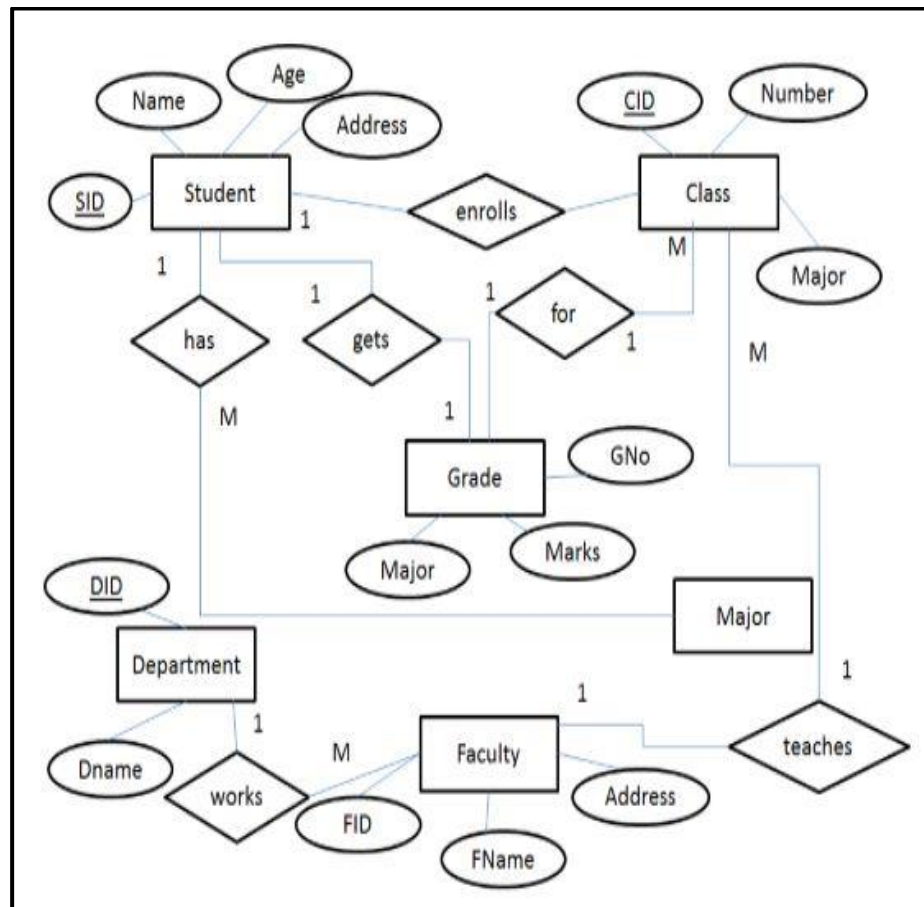
- The system should be modular and well-documented to allow easy updates and future feature additions.
- It must be designed with scalability in mind to accommodate increasing numbers of students, records, and users.
- The architecture should support integration with other systems in the future, such as fee management or communication tools.

## CHAPTER 2

### System Flow Diagrams

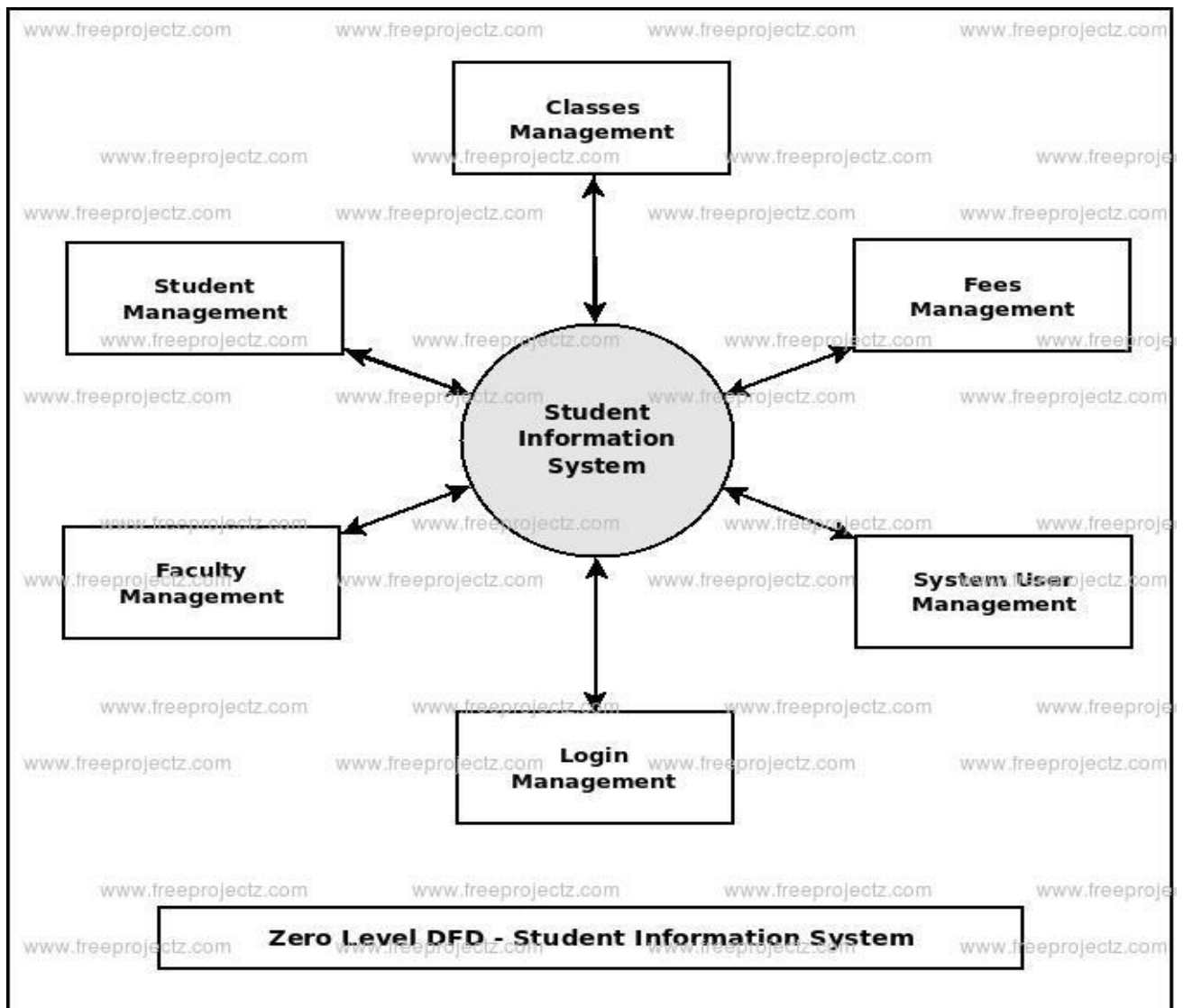


**Figure 2.1 Use Case Diagrams**



**Figure 2.2 Entity Relationship Diagram**





**Figure 2.3 Data Flow Diagram**

## **CHAPTER 3**

### **Module description**

#### **1. Login Module (LM)**

- Provides secure access with username and password.
- Ensures authorized users can access the system.

#### **2. Student Management Module (SMM)**

- Manages student registration, updates, and deletions.
- Allows viewing and modifying student details.

#### **3. Attendance Module (AM)**

- Tracks and marks student attendance for classes.
- Generates attendance reports and summaries.

#### **4. Grade Management Module (GMM)**

- Allows teachers to enter and update student grades.
- Generates grade reports for individual students.

#### **5. Reports Module**

- Generates attendance and grade reports.
- Allows exporting reports in various formats (optional).

#### **6. Admin/Teacher Dashboard**

- Centralized hub for managing the system.
- Provides quick access to all modules.

#### **7. Logout Module**

- Logs out users securely, preventing unauthorized access.

These modules work together to manage and track student information, attendance, and academic performance efficiently

## CHAPTER 4

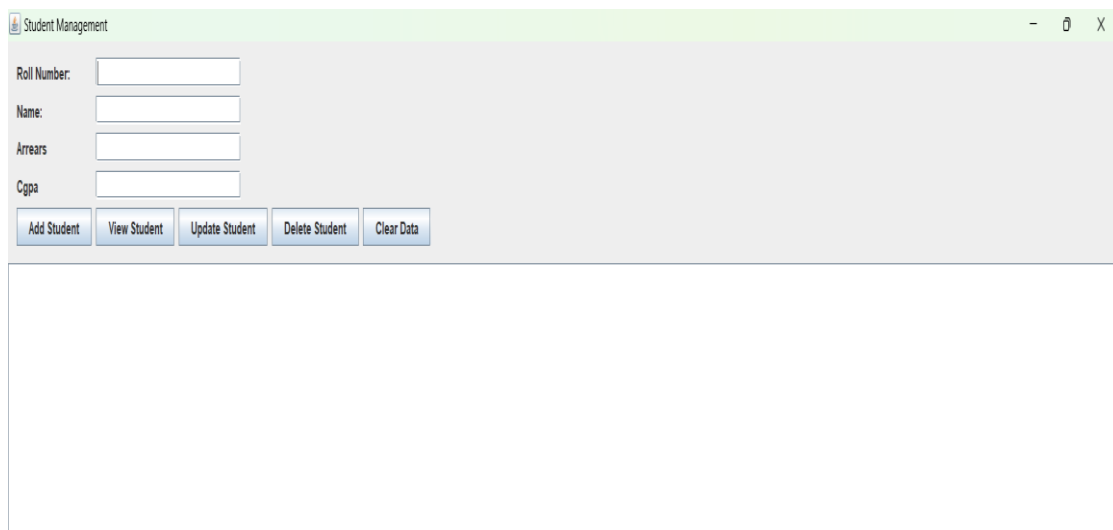
### Implementation

#### 4.1 Design:



A screenshot of a Java Swing window titled "Login". The window has a light green title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background. It contains two labels, "Username:" and "Password:", positioned to the left of two white text input fields. Below the input fields are two blue buttons with white text: "Login" on the left and "Sign Up" on the right.

**Figure.4.1.1 Login Page**



A screenshot of a Java Swing window titled "Student Management". The window has a light green title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background. On the left side, there are four labels: "Roll Number:", "Name:", "Arrears", and "Cgpa", each followed by a white text input field. Below these input fields is a row of five blue buttons with white text: "Add Student", "View Student", "Update Student", "Delete Student", and "Clear Data". The bottom half of the window is a large, empty white rectangular area.

**Figure 4.1.2 Main Page**

## 4.2 Database Design:

### Database Design for Student Management System (SMS)

The database for the Student Management System (SMS) is designed to store all relevant information, including student records, attendance, grades, and user login details. Below is a simple database schema that supports the system's functionality:

Stores information about the students.

Column Name	Data Type	Description
student_id	INT (PK)	Unique identifier for each student
first_name	VARCHAR(100)	Student's first name
last_name	VARCHAR(100)	Student's last name
date_of_birth	DATE	Student's date of birth
address	VARCHAR(255)	Student's address
phone_number	VARCHAR(15)	Student's phone number
email	VARCHAR(100)	Student's email address
enrollment_date	DATE	Date the student enrolled

**Table 4.2.1 Student Table**

Column Name	Data Type	Description
user_id	INT (PK)	Unique identifier for each user
username	VARCHAR(50)	User's login username
password	VARCHAR(255)	User's password (hashed for security)
role	ENUM('Admin', 'Teacher')	Role of the user

**Table 4.2.2 User Table**

### 4.3 Implementation(CODE):

```

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

public class LoginFrame extends JFrame {
    private String enteredUsername="";
    private finalProject mainApp;

    private JTextField usernameField;
    private JPasswordField passwordField;

```

```

public LoginFrame(finalProject mainApp) {
    this.mainApp = mainApp;

    setTitle("Login");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 150);
    setLocationRelativeTo(null);

    JPanel panel = new JPanel(new GridLayout(3, 2, 5, 5));

    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");

    loginButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            enteredUsername = usernameField.getText();
            char[] enteredPasswordChars = passwordField.getPassword();
            String enteredPassword = new String(enteredPasswordChars);

            // Check the entered credentials against the database
            if (authenticate(enteredUsername, enteredPassword)) {
                setVisible(false);

                // Pass the entered username to finalProject
                mainApp.setEnteredUsername(enteredUsername);
            }
        }
    });
}

```

```

        mainApp.setExtendedState(JFrame.MAXIMIZED_BOTH);
        mainApp.setVisible(true);
    } else {
        JOptionPane.showMessageDialog(LoginFrame.this, "Invalid username or password");
    }

    // Clear the password field for security reasons
    passwordField.setText("");
}
});

```

```

signUpButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SignUpFrame signUpFrame = new SignUpFrame();
        signUpFrame.setVisible(true);
    }
});

```

```

panel.add(usernameLabel);
panel.add(usernameField);
panel.add(passwordLabel);
panel.add(passwordField);
panel.add(loginButton);
panel.add(signUpButton);

add(panel);
}

```

```

private boolean authenticate(String enteredUsername, String enteredPassword) {
    // Add your authentication logic here

    // For simplicity, let's assume you have a "users" table in your database with columns
    "username" and "password"
}

```

```

        try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3000/db",
"admin", "admin123")) {
            String selectQuery = "SELECT * FROM users WHERE username = ? AND password = ?";
            try (PreparedStatement statement = connection.prepareStatement(selectQuery)) {
                statement.setString(1, enteredUsername);
                statement.setString(2, enteredPassword);
                ResultSet resultSet = statement.executeQuery();
                return resultSet.next(); // If there is a matching user, authentication is successful
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            return false;
        }
    }
}

```

```

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        finalProject mainApp = new finalProject(); // Create an instance of finalProject
        LoginFrame loginFrame = new LoginFrame(mainApp);
        loginFrame.setVisible(true);
    });
}
}

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

```

```

public class SignUpFrame extends JFrame {

    private JTextField newUsernameField;
    private JPasswordField newPasswordField;

```



```

public SignUpFrame() {
    setTitle("Sign Up");
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setSize(300, 150);
    setLocationRelativeTo(null);

    JPanel panel = new JPanel(new GridLayout(3, 2, 5, 5));

    JLabel newUsernameLabel = new JLabel("New Username:");
    JLabel newPasswordLabel = new JLabel("New Password:");

    newUsernameField = new JTextField();
    newPasswordField = new JPasswordField();

    JButton signUpButton = new JButton("Sign Up");

    signUpButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String newUsername = newUsernameField.getText();
            char[] newPasswordChars = newPasswordField.getPassword();
            String newPassword = new String(newPasswordChars);

            // Add your registration logic here
            if (registerUser(newUsername, newPassword)&&
createDatabase(newUsername,newPassword)) {
                JOptionPane.showMessageDialog(SignUpFrame.this, "User registered successfully");
                dispose(); // Close the SignUpFrame after successful registration
            } else {
                JOptionPane.showMessageDialog(SignUpFrame.this, "Error registering user");
            }
            // Clear the password field for security reasons
            newPasswordField.setText("");
        }
    });
}

```

```

    }
});

panel.add(newUsernameLabel);
panel.add(newUsernameField);
panel.add(newPasswordLabel);
panel.add(newPasswordField);
panel.add(new JLabel()); // Empty label for spacing
panel.add(signUpButton);

add(panel);
}

private boolean registerUser(String newUsername, String newPassword) {
    try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/db",
"root", "root")) {
        // Check if the username already exists
        String checkUsernameQuery = "SELECT * FROM users WHERE username = ?";
        try (PreparedStatement checkUsernameStatement =
connection.prepareStatement(checkUsernameQuery)) {
            checkUsernameStatement.setString(1, newUsername);
            ResultSet resultSet = checkUsernameStatement.executeQuery();

            if (resultSet.next()) {
                // Username already exists, show a prompt and return false
                JOptionPane.showMessageDialog(this, "Username already exists. Please choose a
different username.");
                return false;
            }
        }

        // If the username doesn't exist, proceed with user registration
        String insertQuery = "INSERT INTO users (username, password) VALUES (?, ?)";
        try (PreparedStatement statement = connection.prepareStatement(insertQuery)) {
            statement.setString(1, newUsername);

```

```

        statement.setString(2, newPassword);
        statement.executeUpdate();
        return true;
    }
} catch (SQLException ex) {
    ex.printStackTrace();
    return false;
}
}

private boolean createDatabase(String newUsername, String newPassword)
{
    try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/db",
"root", "root"))
    {
        String createQuery = "Create table "+newUsername+"(sno int,name varchar(100),qty
int,price float)";
        try(PreparedStatement statement = connection.prepareStatement(createQuery))
        {
            statement.executeUpdate();
            return true;
        }
    } catch (SQLException e)
    {
        e.printStackTrace();
        return false;
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        SignUpFrame signUpFrame = new SignUpFrame();
        signUpFrame.setVisible(true);
    });
}

```

}

## **5.CONCLUSION:**

The Student Management System streamlines the management of student information, automating tasks like registration, attendance, grades, and reports. It reduces administrative effort, minimizes errors, and enhances data accessibility while ensuring security. The project demonstrates scalability and provides a strong foundation for further improvements, such as adding real-time analytics, mobile compatibility, and advanced features to enhance usability and functionality.

## **6.REFERENCE:**

- [1] <https://www.javatpoint.com/java-awt>
- [2] <https://www.javatpoint.com/java>

