STUDENT RESULT MANAGEMENT SYSTEM

CS23333 –Object Oriented Programming using Java Project Report

Submitted by

DEEPAK K-231001029

AVINASH V-231001023

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER-2024

BONAFIDE CERTIFICATE

Certified that this project titled "Student Result Management System" is the bonafide work of "**DEEPAK K(231001029), AVINASH V(231001023)**" 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 InformationTechnology
Rajalakshmi Engineering College

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

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF Table

1.1 Abstract

The **Student Result Management System** is a comprehensive Java-based application designed to streamline the management of student records. It automates key processes such as student registration, grade entry, and result calculation, significantly improving administrative efficiency. The system facilitates secure user authentication, allowing administrators to manage data, teachers to input grades, and students to access their results seamlessly. With features like automated result computation and report generation, the application minimizes errors and provides timely, accurate feedback to students. Its scalable architecture ensures the system can easily accommodate future enhancements and integration with other educational modules.

1.2 Introduction

The **Student Result Management System** is a Java-based application designed to automate and simplify the management of student data, including registration, grade entry, and result calculation. It serves as a platform for administrators to manage records, teachers to input grades, and students to view their results. The system ensures accurate, timely feedback through automated result computation and report generation. By eliminating manual processes, it enhances efficiency and reduces the potential for errors. The system is also built to be scalable, supporting future improvements and integration with other educational modules.

1.3 Purpose

The purpose of the Student Result Management System is to automate and streamline the management of student academic records, minimizing manual errors and enhancing administrative efficiency. It provides a fast, reliable platform for students, teachers, and administrators to manage and access grades and results. By automating key processes, the system improves the accuracy of record-keeping and ensures timely, secure access to student performance data. This leads to a more efficient workflow and faster decision-making for all users. Ultimately, it aims to provide a seamless experience for managing academic records throughout the institution.

1.4 Scope of the Project:

The Student Result Management System focuses specifically on managing and automating the process of checking and tracking student marks. Its scope includes functionalities like student registration, grade entry by teachers, result calculation, and the generation of individual performance reports. The system allows administrators to manage student records, teachers to input and update grades, and students to access their results in a secure, user-friendly platform.Students are able to access their marks and review their academic history securely. It also facilitates quick and easy result dissemination, improving overall communication between students and faculty.

1.5 Software Requirement Specification

Introduction

Student Result Management System is that it streamlines the entire process of grade entry, result calculation, and report generation, ensuring accuracy, efficiency, and timely access to academic performance data for students, teachers, while eliminating manual errors and paper- based processes.

Product Scope

The product scope of the Student Result Management System is to automate the management of student grades, result calculations, and report generation. It aims to enhance the efficiency and accuracy of academic result handling. The system focuses on streamlining administrative processes and minimizing errors.

References and Acknowledgement

[1] https://www.javatpoint.com/java-swing

Overall Description

The Student Result Management System is a web-based application designed to automate and streamline the process of managing student grades, result calculations, and report generation. It provides an intuitive platform for administrators, teachers, and students to manage and access academic performance data efficiently.

Product Perspective

The system follows a client/server architecture and is compatible with major operating systems such as Windows and Linux. It uses MySQL for backend database management and HTML/CSS for creating a responsive, user-friendly frontend interface. Developed with Java Spring Boot and JDBC, the application ensures secure transactions and robust database connectivity.

Product Functionality

a) Admin Registration: Allows new administrators to register and gain access to manage the system.
b) Admin Login: Provides secure login for authorized administrators to access the system.
c) Add/Update Student Records: Enables administrators and teachers to add or modify student details and results.
d) View Student Results: Allows students, teachers, and administrators to view student grades and academic history.
e) Delete Student Records: Allows administrators to remove outdated or incorrect student data from the system.
f) Generate Reports: Enables the system to generate academic reports for students and faculty based on performance.

User and Characteristics

Qualifications: Administrators and teachers should have basic knowledge of using computer systems and handling academic records, while students need only access credentials to view their results securely.

Operating Environment

Hardware Requirements

- Processor: Intel i3 or higher (or equivalent AMD processor)
- Operating System: Windows 8,10, 11
- Processor Speed: 2.0 GHz - RAM: 4GB
- Hard Disk: 500GB

Software Requirements

- Database: MySQL
- Frontend: Java (SPRING BOOT)
- Technology: Java (JDBC)

Constraints

Data Security: The system must ensure secure handling and privacy of sensitive student data.

Internet Dependency: The system requires stable internet connectivity for real-time updates and result access.

User Interface

The Student Result Management System provides user-friendly, menu driven interfaces for

a) Admin Registration: Allows administrators to register and gain access to the system.

b) Admin Login: Provides secure login for authorized administrators to manage student data.

c) Add/Update Student Records: Admins and teachers can add and update student information and grades.

d) View Results: Students, teachers, and admins can view student grades and academic performance.

e) Delete Records: Administrators can delete outdated or incorrect student records.

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) Java AWT and SWING for designing the front end
c) MySQL for the backend
d) Platform: Java Language
e) Integrated Development Environment (IDE): IntelliJ IDEA

Functional Requirements

Student Registration and Login:
The system must allow students to register and securely log in to access their results.

Grade Entry and Calculation:
Teachers should be able to input student grades, and the system must automatically calculate results based on predefined criteria.

Result Viewing:
Students and authorized users (admins, teachers) should be able to view detailed result reports and academic performance.

Report Generation:
The system must generate performance reports and allow downloading of reports for individual students or groups of students.

Non-functional Requirements

1) Performance

The system should load and process results quickly, even with a large number of students and records.

2) Security

User data, including student grades and personal information, must be securely stored and protected against unauthorized access.

3) Availability

The system should be available 24/7 with minimal downtime, ensuring users can access results and manage data at any time.

4) Usability

The interface should be user-friendly, allowing easy navigation administrators, teachers, and students without extensive training.

5) Scalability

The system should be scalable to accommodate growing numbers of students and records as the institution expands.

6) Maintenance:

The system should be easy to maintain, with regular updates and bug fixes. It should allow for quick resolution of issues and enable efficient management of hardware and software components to ensure long-term stability and performance.

2. SYSTEM FLOW DIAGRAMS

Use Case Diagram Of Student Result Management system

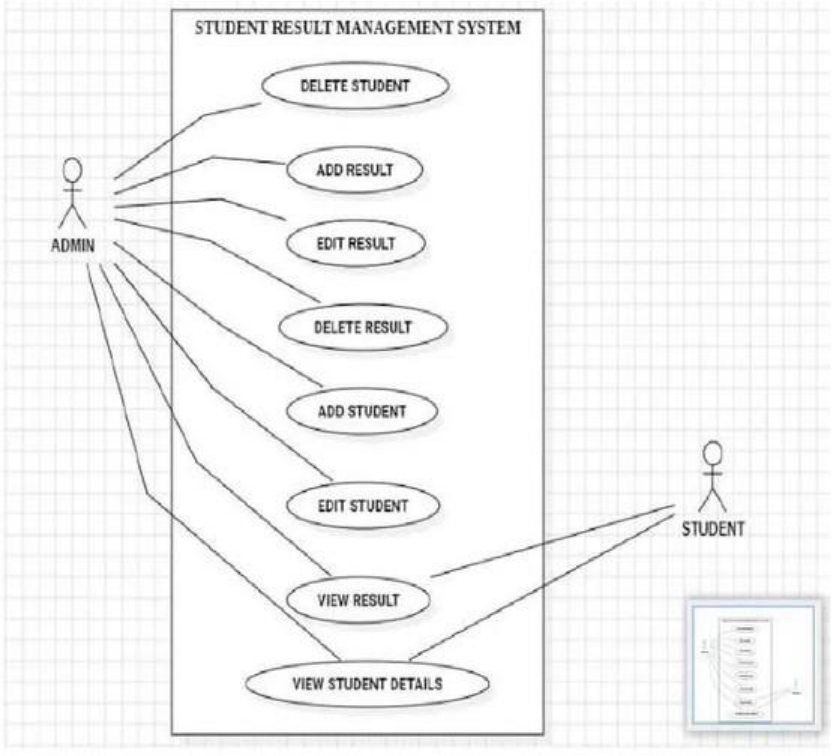


Figure 2.1 Use Case Diagrams

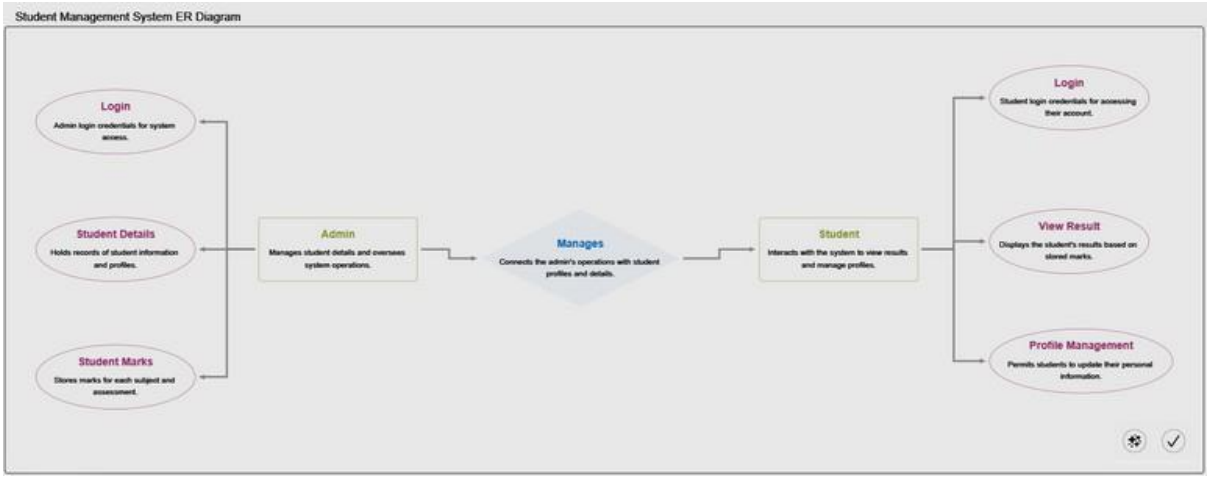


Figure 2.2 Entity Relationship Diagram

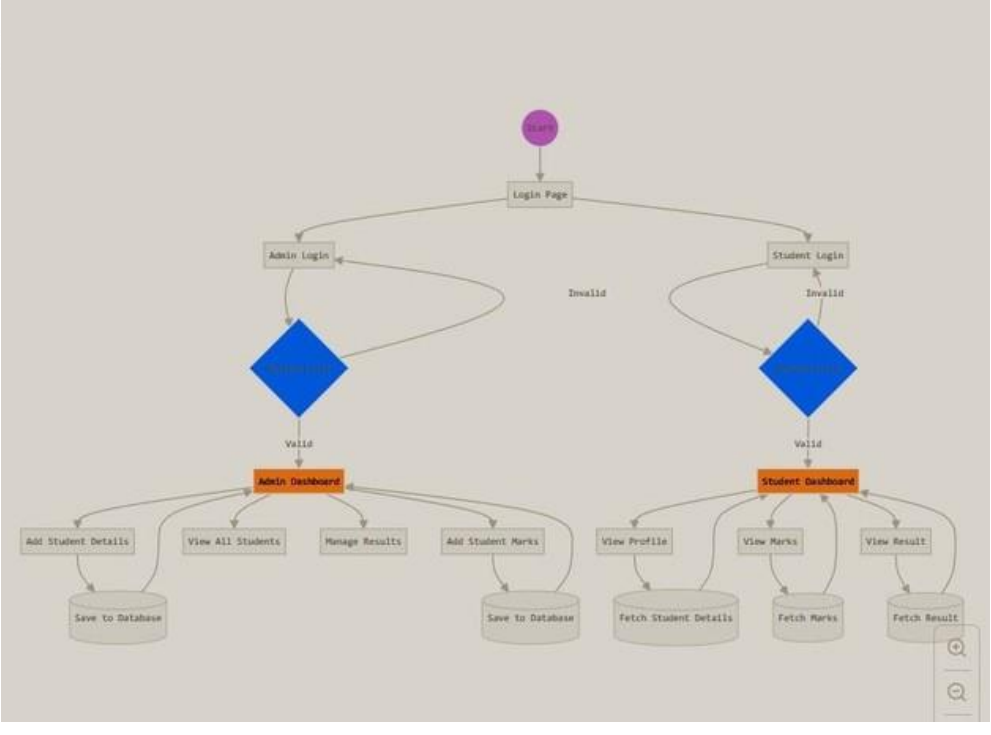


Figure 2.3 Data-flow diagram

3. MODULE DESCRIPTION

Admin Module

Features:

Admin Registration:

Allows new administrators to register and create an account.

Admin Login:

Ensures secure login to prevent unauthorized access.

Manage Student Records:

Administrators can add, update, or delete student records.

Generate Reports:

Admins can generate academic reports based on student results.

Teacher Module

Features:

Grade Entry:

Teachers can input and update student grades.

Result Calculation:

The system automatically calculates final results based on entered grades.

View Student Results:

Teachers can access and review student performance and academic progress.

Student Module

Features:

Login:

Students can securely log in to view their results.

View Results:

Students can view individual grades and overall academic performance.

Access Reports:

Students can access and download their performance reports.

4.1 Design

.

Figure 4.1.1 Home Page

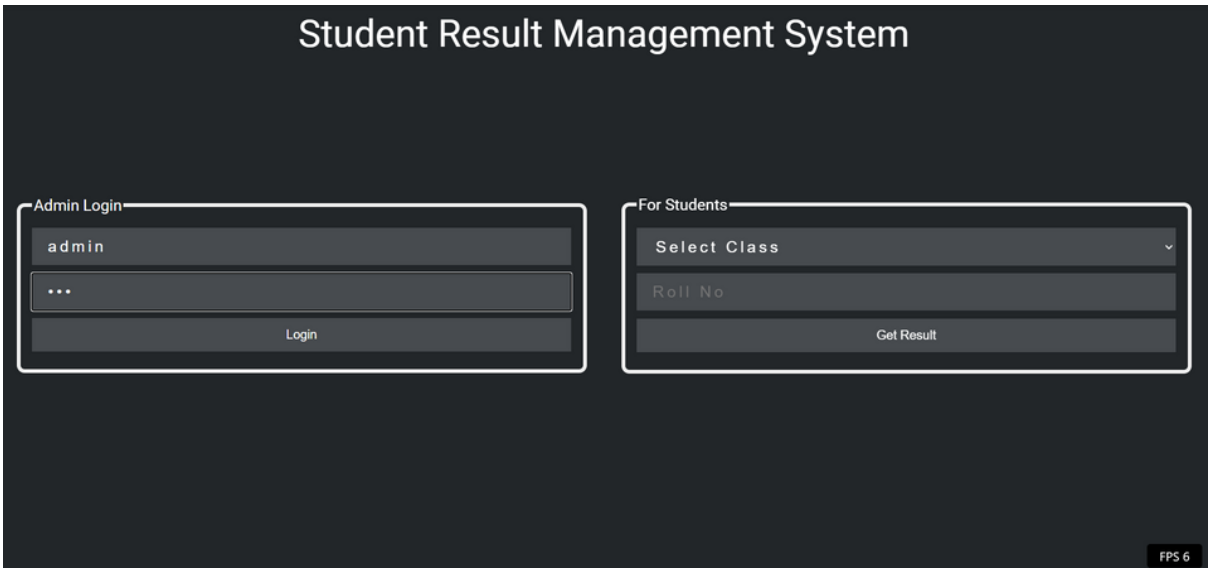


Figure 4.1.2 Login Page

Figure 4.1.3 Dashboard Page

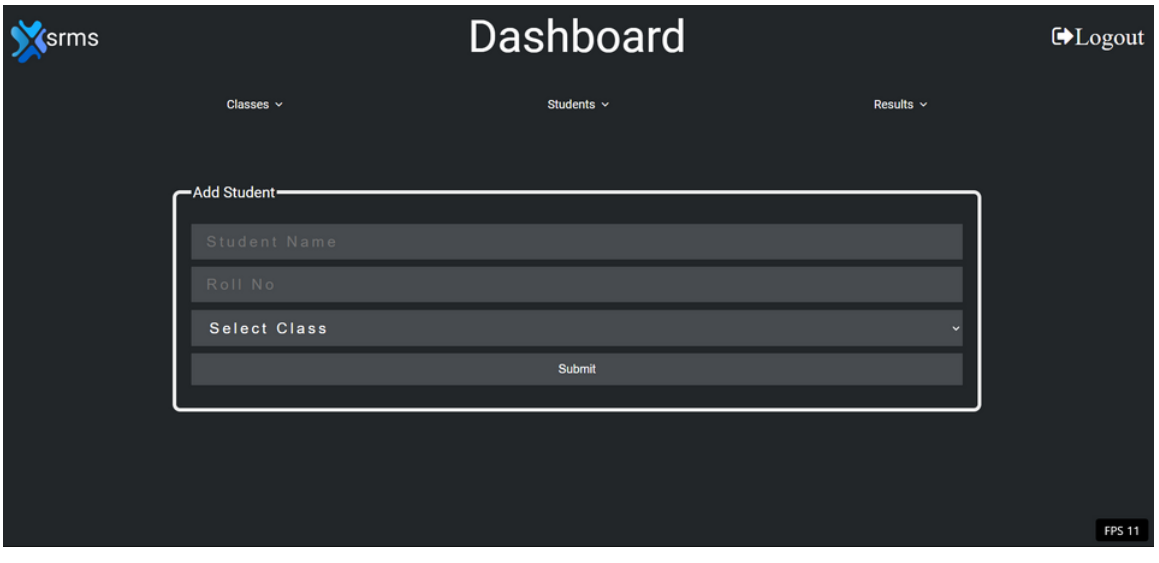


Figure 4.1.4 Student Page

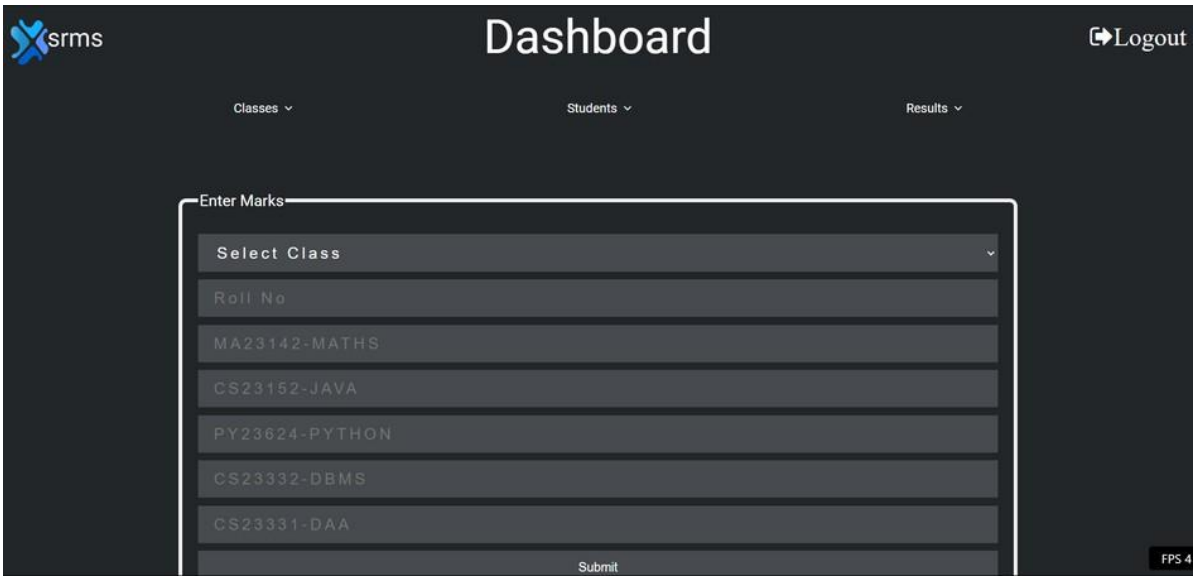
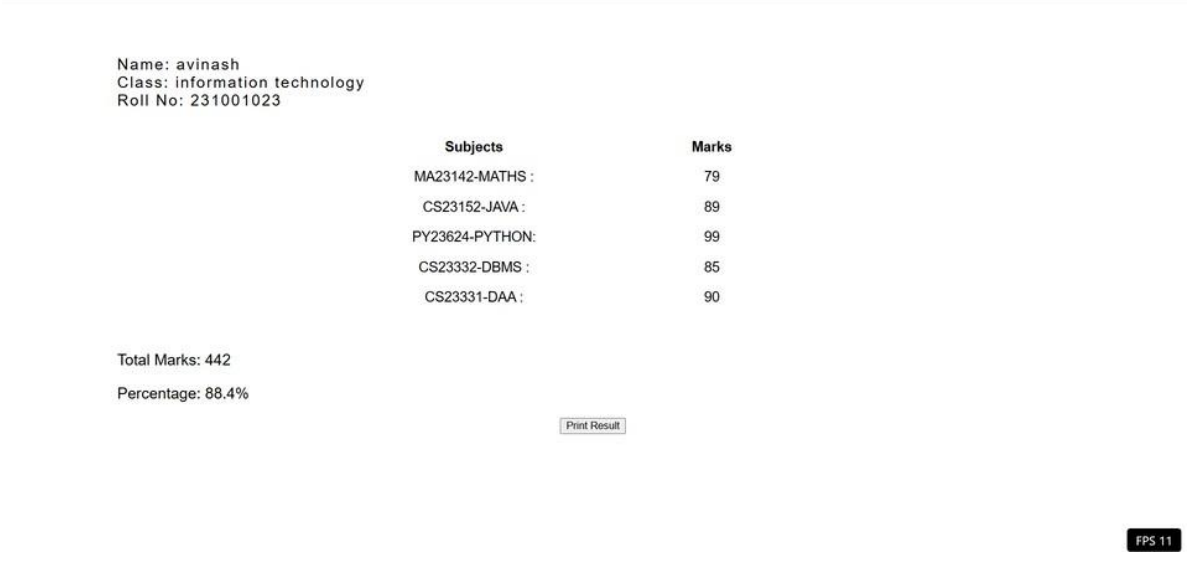


Figure 4.1.5 Add Result Page



Name: avinash
Class: information technology
Roll No: 231001023

Subjects	Marks
MA23142-MATHS :	79
CS23152-JAVA :	89
PY23624-PYTHON:	99
CS23332-DBMS :	85
CS23331-DAA :	90

Total Marks: 442

Percentage: 88.4%

Print Result

FPS 11

Figure 4.1.6 Result 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 stage. They are structured and put together to design the data storage 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.

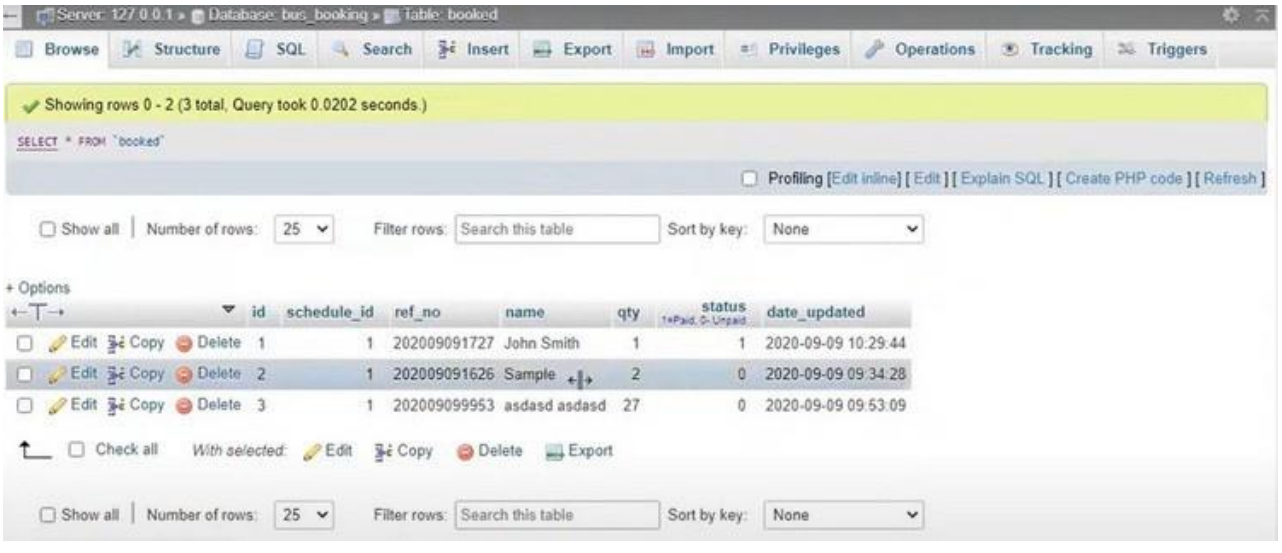


Figure 4.2.1 MySql Table

SELECT * FROM `students`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	name	rno	class_name
Edit Copy Delete	ram	231001038	computer science
Edit Copy Delete	hariprasnth	231001055	csbs
Edit Copy Delete	john	231001040	ece
Edit Copy Delete	ameen	231001012	information technology
Edit Copy Delete	avinash	231001023	information technology
Edit Copy Delete	bharath kumar	231001028	information technology
Edit Copy Delete	deepak	231001029	information technology
Edit Copy Delete	gowtham	231001050	information technology
Edit Copy Delete	hari vignesh	231001051	information technology

Check all *With selected:* Edit Copy Delete Export

Figure 4.2.2 Student Table

Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

SELECT * FROM `admin_login`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	userid	password
Edit Copy Delete	admin	123

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Figure 4.2.3 User Table

4.3 IMPLEMENTATIONS (CODE)

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

public class StudentResultManagementGUI {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/student_db";
    private static final String USER = "root";
    private static final String PASSWORD = "admin";

    private JFrame frame;
    private JTextField idField, nameField, marksField;
    private JTextArea displayArea;

    public StudentResultManagementGUI() {
        frame = new JFrame("Student Result Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        frame.setLayout(new BorderLayout());

        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(4, 2));
        inputPanel.add(new JLabel("ID:"));
        idField = new JTextField();
        inputPanel.add(idField);
        inputPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        inputPanel.add(nameField);
        inputPanel.add(new JLabel("Marks:"));
        marksField = new JTextField();
        inputPanel.add(marksField);

        JButton addButton = new JButton("Add Student");
        JButton displayButton = new JButton("Display All");
        JButton topScorerButton = new JButton("Top Scorer");
        JButton lowScorerButton = new JButton("Lowest Scorer");
        JButton averageButton = new JButton("Average Marks");

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(1, 5));
        buttonPanel.add(addButton);
        buttonPanel.add(displayButton);
        buttonPanel.add(topScorerButton);
        buttonPanel.add(lowScorerButton);
```

```
    buttonPanel.add(averageButton);

    displayArea = new JTextArea();
    displayArea.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(displayArea);

    frame.add(inputPanel, BorderLayout.NORTH);
    frame.add(buttonPanel, BorderLayout.CENTER);
    frame.add(scrollPane, BorderLayout.SOUTH);

    addButton.addActionListener(e -> addStudent());
    displayButton.addActionListener(e -> displayAllStudents());
    topScorerButton.addActionListener(e -> displayTopScorer());
    lowScorerButton.addActionListener(e -> displayLowestScorer());
    averageButton.addActionListener(e -> calculateAverageMarks());

    frame.setVisible(true);

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD)) {
        createTable(conn);
    } catch (SQLException ex) {
        showError(ex.getMessage());
    }
}

private void addStudent() {
    int id = Integer.parseInt(idField.getText());
    String name = nameField.getText();
    int marks = Integer.parseInt(marksField.getText());

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD)) {
        String insertSQL = "INSERT INTO students (id, name, marks) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {
            pstmt.setInt(1, id);
            pstmt.setString(2, name);
            pstmt.setInt(3, marks);
            pstmt.executeUpdate();
            displayArea.append("Student added: " + name + "\n");
        }
    } catch (SQLException ex) {
        showError(ex.getMessage());
    }
}

private void displayAllStudents() {
    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
         Statement stmt = conn.createStatement();
```

```
            ResultSet rs = stmt.executeQuery("SELECT * FROM students")) {

            displayArea.setText("ID\tName\tMarks\n");
            while (rs.next()) {
                displayArea.append(rs.getInt("id") + "\t" + rs.getString("name") + "\t" + rs.getInt("marks") +
"\n");
            }
        } catch (SQLException ex) {
            showError(ex.getMessage());
        }
    }

    private void displayTopScorer() {
        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT name, marks FROM students ORDER BY marks
DESC LIMIT 1")) {

            if (rs.next()) {
                displayArea.append("Top Scorer: " + rs.getString("name") + " - " + rs.getInt("marks") + "
marks\n");
            } else {
                displayArea.append("No students found.\n");
            }
        } catch (SQLException ex) {
            showError(ex.getMessage());
        }
    }

    private void displayLowestScorer() {
        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT name, marks FROM students ORDER BY marks ASC
LIMIT 1")) {

            if (rs.next()) {
                displayArea.append("Lowest Scorer: " + rs.getString("name") + " - " + rs.getInt("marks") + "
marks\n");
            } else {
                displayArea.append("No students found.\n");
            }
        } catch (SQLException ex) {
            showError(ex.getMessage());
        }
    }

    private void calculateAverageMarks() {
```

```
    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT AVG(marks) AS average FROM students")) {

        if (rs.next()) {
            displayArea.append("Average Marks: " + rs.getDouble("average") + "\n");
        } else {
            displayArea.append("No students found.\n");
        }
    } catch (SQLException ex) {
        showError(ex.getMessage());
    }
}

private void createTable(Connection conn) throws SQLException {
    String createTableSQL = "CREATE TABLE IF NOT EXISTS students (" +
            "id INT PRIMARY KEY, " +
            "name VARCHAR(50), " +
            "marks INT)";
    try (Statement stmt = conn.createStatement()) {
        stmt.execute(createTableSQL);
    }
}

private void showError(String message) {
    JOptionPane.showMessageDialog(frame, message, "Error", JOptionPane.ERROR_MESSAGE);
}

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

5. CONCLUSION

The Student Result Management System streamlines the process of managing student academic records, offering an intuitive interface built with technologies like Java, Spring Boot, MySQL, and JDBC. It allows administrators to efficiently handle student data, course enrollments, exam results, and grade management while ensuring data integrity and reducing redundancy through a well-structured database design. This system minimizes manual errors, improves accuracy, and saves time for both students and faculty. As a scalable solution, it can be further enhanced with features such as automated report generation, advanced analytics, and integration with other educational tools, making it an invaluable asset for educational institutions.

6. REFERENCES

[1] Reference link for java-awt at javatpoint : https://www.javatpoint.com/java-awt

[2] Reference link for java-swing at javatpoint : https://www.javatpoint.com/java-swing