STUDENT MARK ANALYSIS SYSTEM

CS23333-OBJECT ORIENTED PROGRAMMING USING JAVA

Submitted by

FAQRUDEEN FAIZAN.Z -231001042

GOKUL KRISHNA.R -231001046

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM
(An Autonomous Institution)

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM ,CHENNAI 600 025

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project titled "Student Mark Analysis System" is the Bonafide work of "**FAQRUDEEN FAIZAN.Z(231001042)"," GOKUL KRISHNA.R(231001046)"** who carried out the project work under my supervision.

SIGNATURE

Dr. P. Valarmathie

HEAD OF THE DEPARTMENT

Information Technology
Rajalakshmi Engineering College,
Rajalakshmi Nagar, Thandalam
Chennai – 602105

SIGNATURE

Mrs. Usha S

COURSE INCHARGE

Information Technology
Rajalakshmi Engineering College
Rajalakshmi Nagar, Thandalam
Chennai – 602105

This project is submitted for IT19341 – Introduction to Oops and Java held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

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 the requisite infrastructure.

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.

PREMKUMAR M
THAMIZHARASU P

1.1 Abstract

The Student Mark Analysis System is a Java-based application that uses JDBC to connect to an SQLite database for storing and analyzing student marks. The system includes features for inserting student marks, calculating statistical metrics such as mean and standard deviation, and identifying top and lowest performers. A secure login mechanism ensures that only authorized users can access the system, providing a robust and efficient solution for educational performance analysis..

1.2 Introduction

The Student Mark Analysis System is an innovative application designed to streamline the management and analysis of student performance data. Utilizing Java and JDBC to interface with an SQLite database, this system enables the efficient storage, retrieval, and statistical evaluation of student marks. Key features include the ability to input student marks, compute vital statistics such as average and standard deviation, and identify top and lowest performers. Enhanced with a secure login mechanism, the system ensures that only authorized users can access and manage sensitive educational data, making it a robust tool for academic performance analysis.

1.3 Purpose

The purpose of this project is to create an efficient and user-friendly student mark analysis system that benefits both students and teachers. The system aims to:

- Provide a comprehensive tool for educators to manage student performance data.
- Facilitate detailed statistical analysis to identify trends and areas for improvement.
- Enable the generation of performance reports and comparative analysis.
- Support data-driven decision-making to enhance the educational experience.

1.4 Scope of the Project

The scope of the Student Mark Analysis System encompasses a range of functionalities designed to enhance the educational process by leveraging data management and analysis tools. The system aims to streamline the process of recording, storing, and analyzing student marks. It provides educators with a robust platform to input student marks, calculate various statistics, and generate comprehensive performance reports.

One of the core components of this project is its integration with a SQLite database using JDBC (Java Database Connectivity). JDBC enables seamless interaction between the application and the SQLite database, allowing for efficient data storage and retrieval. This connectivity ensures that student marks are securely stored and can be accessed for real-time analysis. The system supports multiple features such as the insertion of student marks, calculation of statistical measures like mean and standard deviation, and the identification of top and lowest performers. Additionally, it offers a user-friendly interface for educators to view and analyze student performance data, facilitating informed decision-making and targeted educational interventions.

1.5 Software Requirement Specification

1. Database Integration:
- SQLite Database: The system uses a SQLite database to store student data, including names and marks. This database is connected using JDBC (Java Database Connectivity), ensuring seamless interaction between the application and the database.
- Schema Design: The database schema includes a table named `students` with columns for student names and their respective marks.

2.Backend Development:
-Server Setup: The backend server is implemented using Java with the `HttpServer` class, handling various endpoints for different functionalities.
- Endpoints:
- /api/insert: Handles the insertion of student marks.
-/api/calculateStatistics: Computes and returns statistical measures such as mean, variance, and standard deviation.
- /api/topPerformer: Retrieves the student with the highest marks.
- /api/lowestPerformer: Retrieves the student with the lowest marks.
- /api/login: Manages user authentication for accessing the system.

3. Frontend Development:
- HTML and CSS: The frontend is developed using HTML for structure and CSS for styling, providing a user-friendly interface for inputting student marks, viewing analysis results, and generating performance reports.
- JavaScript: Utilizes JavaScript and Chart.js to dynamically update the frontend, handle form submissions, and render statistical charts.

4. Functionality:
-Data Input: Teachers can input student names and marks through a web form.
-Statistical Analysis: The system calculates key statistical measures and presents them in a comprehensible format.

-Performance Reports: Generates detailed reports highlighting top and lowest performers, along with comparative analysis.
-User Authentication: Secure login mechanism to ensure that only authorized personnel can access the system.

5.Technology Stack:
-Frontend: HTML, CSS, JavaScript, Chart.js
-Backend: Java, SQLite, JDBC
-Tools and Libraries: JSON handling libraries for data exchange, HttpServer for handling HTTP requests and responses.

This comprehensive system aims to streamline the process of student mark analysis, providing educators with valuable insights into student performance through efficient data management and analysis tools.

References and Acknowledgement:

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

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

Overall Description

The Student Mark Analysis System offers a streamlined solution for managing and analyzing student performance data. It provides authorized users, such as teachers and administrators, with tools to input, view, and analyze student marks, thereby simplifying the process of academic performance evaluation.

Product Perspective

The system is built using a client/server architecture, compatible with various operating systems. The frontend is developed with HTML, CSS, and JavaScript, incorporating Chart.js for data visualization. The backend is powered by Java with the HttpServer class for handling HTTP requests, and SQLite for efficient data management using JDBC (Java Database Connectivity).

Product Functionality

a) **Insert Student Marks**: Allows users to input and store student marks in the database.
b) **Calculate Statistics**: Computes statistical measures such as mean, variance, and standard deviation of student marks.
c) **View Top Performer**: Retrieves and displays the student with the highest marks.
d) **View Lowest Performer**: Retrieves and displays the student with the lowest marks. e) **Generate Performance Report**: Provides detailed reports on student performance. f) **User Authentication**: Ensures secure access for authorized users.

User and Characteristics

- **Qualification**: Users should have a basic understanding of educational processes and a basic qualification equivalent to matriculation.
- **Experience**: Familiarity with student performance analysis and basic statistical concepts is beneficial.
- **Technical Skills**: Users are expected to have elementary knowledge of computers and the ability to interact with web-based applications.

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**: SQLite (utilizing JDBC)
- **Frontend**: HTML, CSS, JavaScript (with Chart.js)
- **Backend**: Java

Constraints

- System access is limited to authorized users such as administrators and teachers.
- The delete operation is restricted to administrators without additional checks for simplicity.
- Administrators must exercise caution during deletion to maintain data consistency.

Assumptions and Dependencies

- System administrators are responsible for creating and securely communicating login IDs and passwords to users.
- Users are assumed to have basic knowledge of using web-based applications.

Specific Requirements

User Interface: The Student Mark Analysis System provides user-friendly, menu-driven interfaces for:

a) **Login**: Secure login for authorized users.
b) **Insert Student Marks**: Input and store new student marks
c) **View Statistics**: Display statistical analysis of student marks.
d) **Top Performer**: Retrieve and display the student with the highest marks.
e) **Lowest Performer**: Retrieve and display the student with the lowest marks.
f) **Generate Report**: Create and view performance reports.
g) **User Authentication**: Secure management of user access.

Hardware Interface:

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

Software Interface for Student Mark Analysis System

- **Operating System**: MS-Windows (Windows 8, 10, 11)
- **Frontend Development**: HTML, CSS, JavaScript (with Chart.js)
- **Backend Development**: Java
- **Database**: SQLite (integrated with JDBC for data management)
- **Integrated Development Environment (IDE)**: NetBeans

Functional Requirements for Student Mark Analysis System

1. **Login Module (LM)**:
    - Users (admins) can access the Login Module via a secure login page.
    - The system supports login using a username and password.
    - Passwords are masked to ensure security.
    - Only authorized admins, whose credentials match those in the database, are granted access to the system.

2. **Registered Users Module (RUM)**:
    - After successful login, users (admins) are granted access to the main features of the application.
    - Users can view detailed information about students and their marks.
    - Admins have the ability to update student marks, insert new data, and perform various administrative tasks related to data management.

3. **Administrator Module (AM)**:
    - After logging in successfully, the system displays the administrative functions.
    - Admins can manage student data: adding new records, updating existing records, and deleting unused data.
    - The "Add" function allows admins to input new student details and marks, while the "Update" function allows modifications to existing student information.
    - All add, update, or delete actions trigger communication with the backend (via the Server Module) to make necessary changes in the database.

4. **Server Module (SM)**:
    - The Server Module acts as an intermediary between the frontend modules and the database.
    - It receives and processes requests from various modules, ensures proper formatting of data, and manages the system's functionality.
    - It handles communication with the database to validate and execute requests, ensuring data consistency and integrity, especially when dealing with student marks and analysis.

Non-functional Requirements

Performance:

- The system must efficiently handle student data analysis requests, ensuring that calculations like mean, standard deviation, and other statistics are completed in under 2 seconds.
- The system should handle a large number of concurrent requests for data input and report generation without significant delays, ensuring high responsiveness for administrators.

Reliability:

- The system must be robust and capable of recovering gracefully from failures. In case of data corruption or abnormal shutdown, the system should provide mechanisms for data recovery and ensure minimal loss of data.
- The system should be thoroughly tested to handle edge cases such as incorrect or missing data entries, ensuring smooth operation under various conditions.

Availability:

- The system should be available 24/7 for administrators to perform tasks like adding, updating, or deleting student data and generating reports.
- It should maintain high availability and perform critical operations, like querying student marks, with minimal downtime or service interruptions.*Security:*

- A robust security mechanism must be in place on the server side to prevent unauthorized access, safeguard user payment information, and ensure the integrity of the reservation system.
- User privacy, including personal details, must be securely stored and managed to maintain confidentiality.

Security:

- The system must implement strong authentication mechanisms to ensure that only authorized administrators can access the backend and modify sensitive student data.
- Passwords and sensitive user data must be encrypted in both transit (using SSL/TLS) and at rest in the database.
- The system should regularly audit user activities to prevent unauthorized access or modifications to student records.

Maintainability:

- The system should be available 24/7 for administrators to perform tasks like adding, updating, or deleting student data and generating reports.
- It should maintain high availability and perform critical operations, like querying student marks, with minimal downtime or service interruptions.

2 System Flow Diagrams

2.1.Use Case Diagrams :

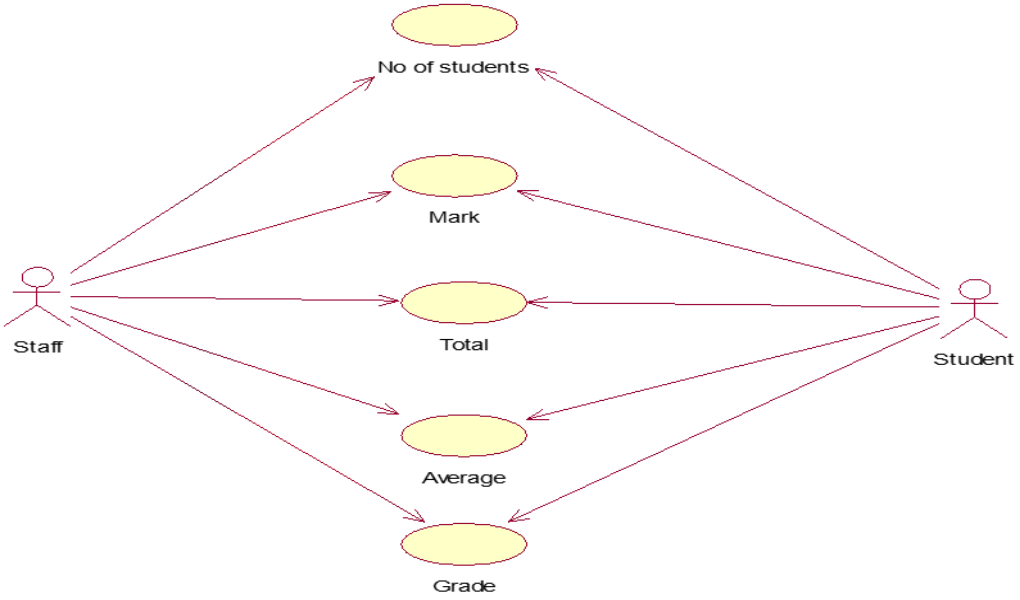


Fig.2.1.Case Diagram

2.2 Entity-relationship diagram

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

Entity -relationship diagram:

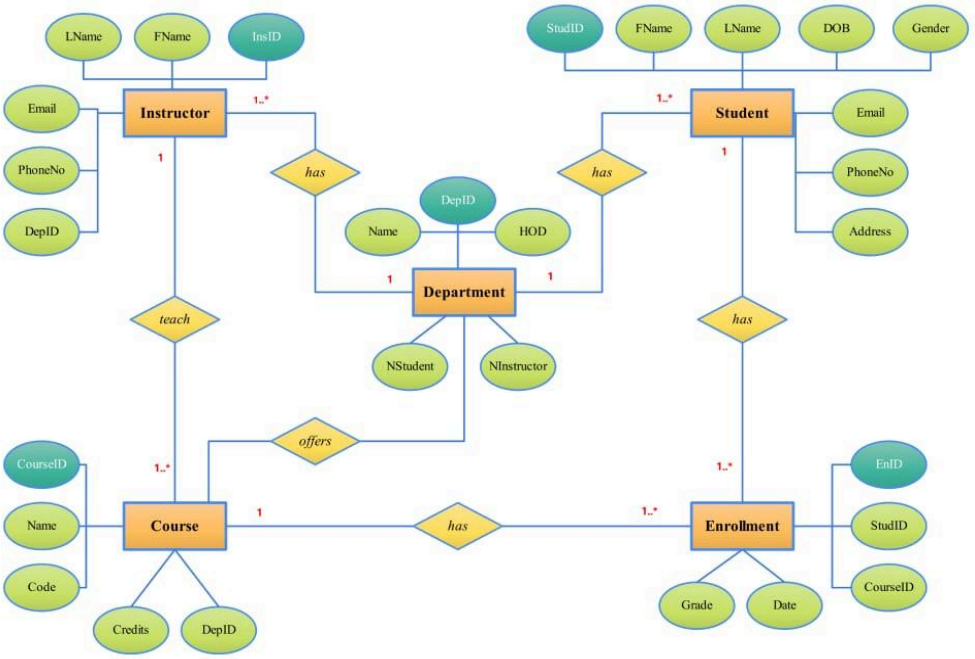


Fig 3.2 ER DIAGRAM OF SMA

2.3 Data-flow diagram:

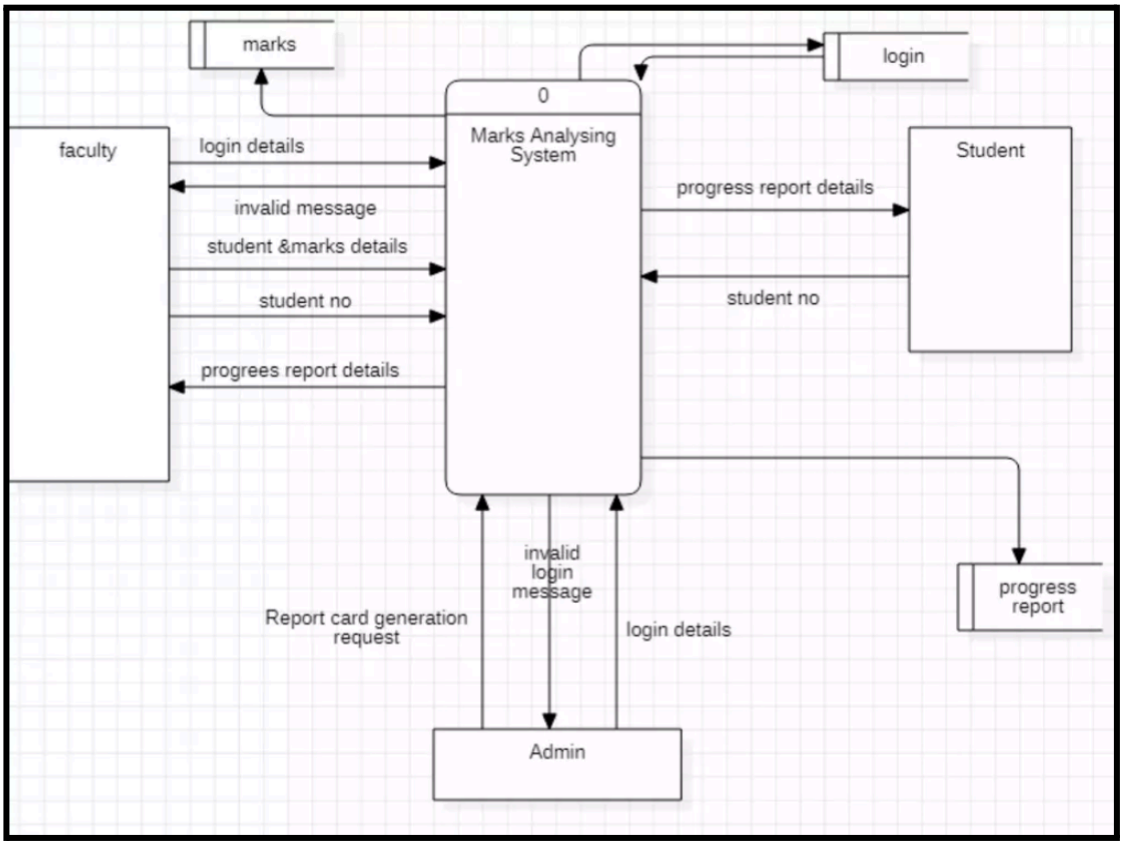


Fig 2.3 Data flow diagram

2. Module description:

1. Register:
- The admin can register an account by providing a unique username and a secure password. The system stores these credentials in a secure manner, ensuring only authorized administrators can access the backend.

2. Login:
- Admins log in using their username and password. The system verifies the credentials against the database, and upon successful login, grants access to the administrative functions.

3.After Login:

- Add Student Marks:
  - The admin can input student details including name and marks. The system saves the data to the database, allowing for future analysis and report generation.

- View Student Marks:
  - Admins can view detailed information about students' marks. The data is displayed in a user-friendly format, and the admin can update or modify the records as needed.

- Update Student Marks:
  - Admins have the ability to modify a student's marks. This allows for corrections or updates to be made to any record if necessary.

- Delete Student Marks:
  - Admins can delete student marks from the system if they are no longer needed. This function is restricted to prevent accidental deletion of important data.

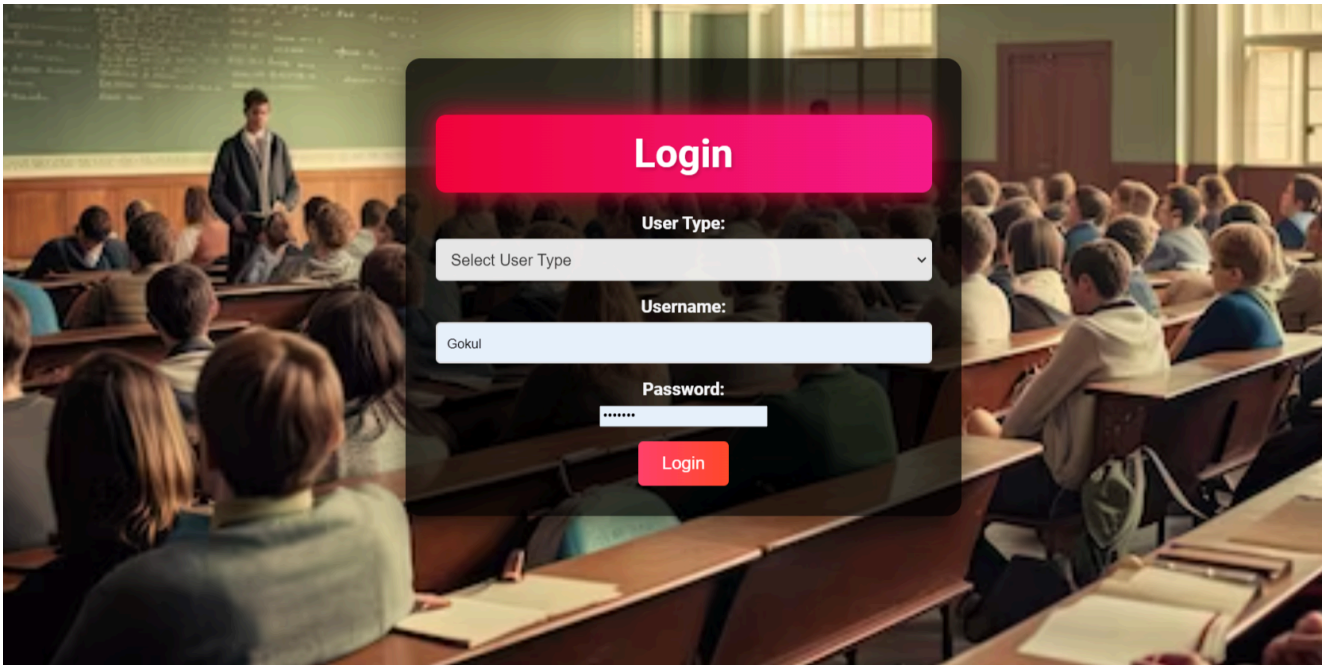
- Generate Statistical Reports:
  - Admins can request detailed statistical analysis of student marks, including mean, variance, and standard deviation. The system calculates and displays the results for easier decision-making.

- Generate Performance Reports:
  - Admins can generate performance reports showing top performers and lowest scorers, helping in the assessment and decision-making process regarding student performance.

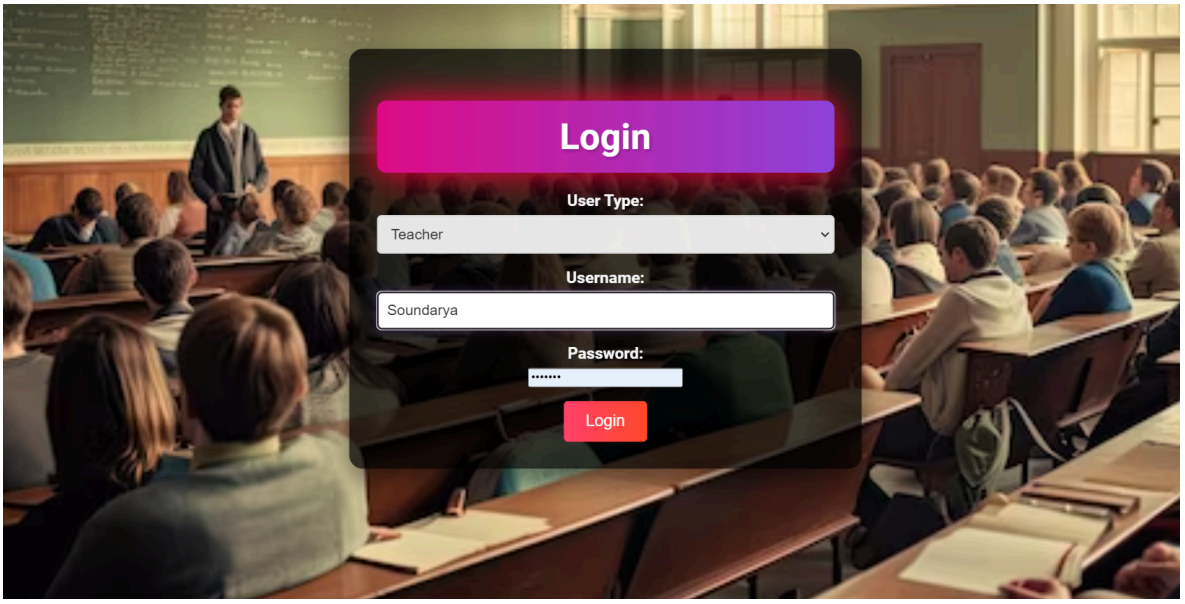
- Remove Admin:
  - If necessary, the admin can remove other administrators from the system, ensuring only authorized personnel have access to sensitive operations.

4. DESIGN

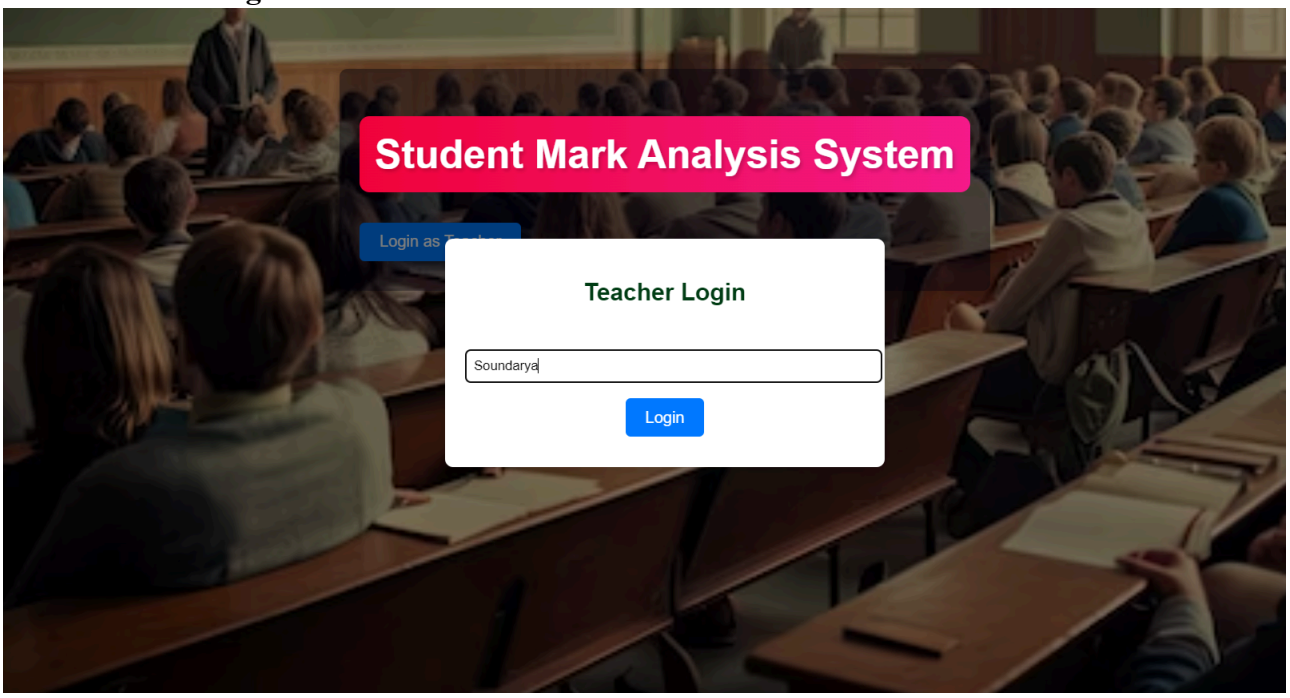
Home Page: LOGIN



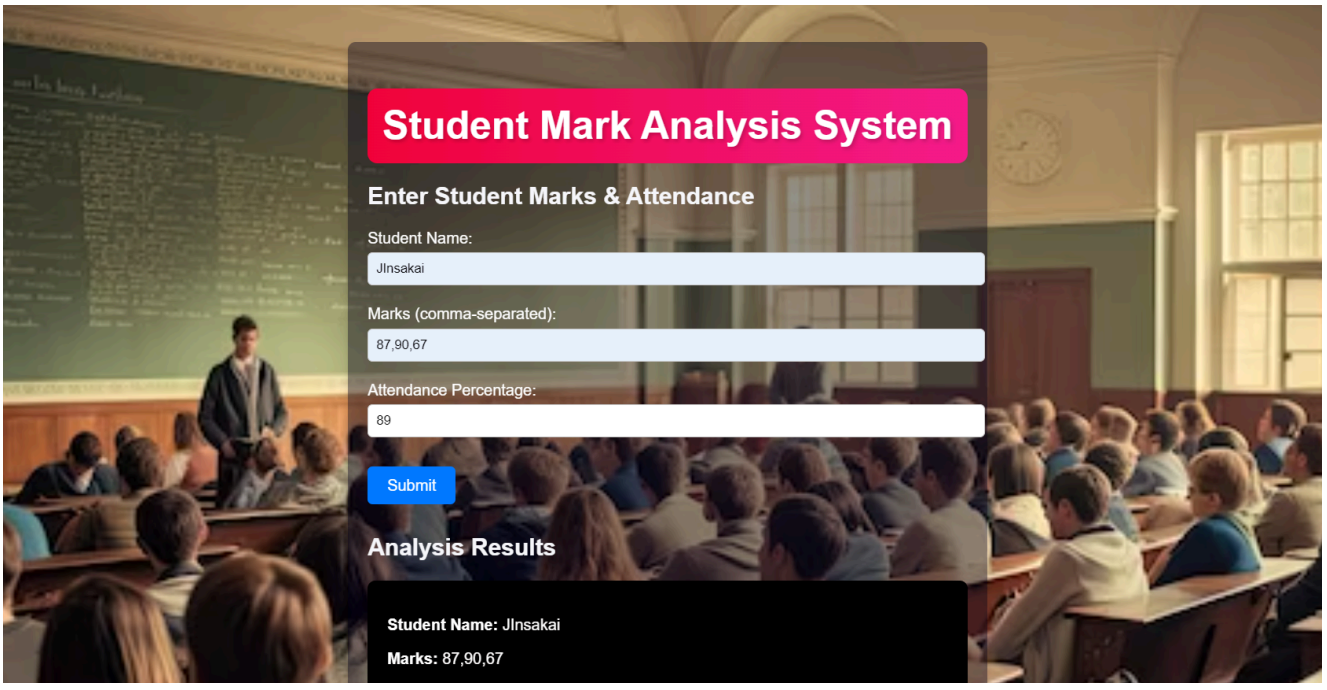
Teacher Login



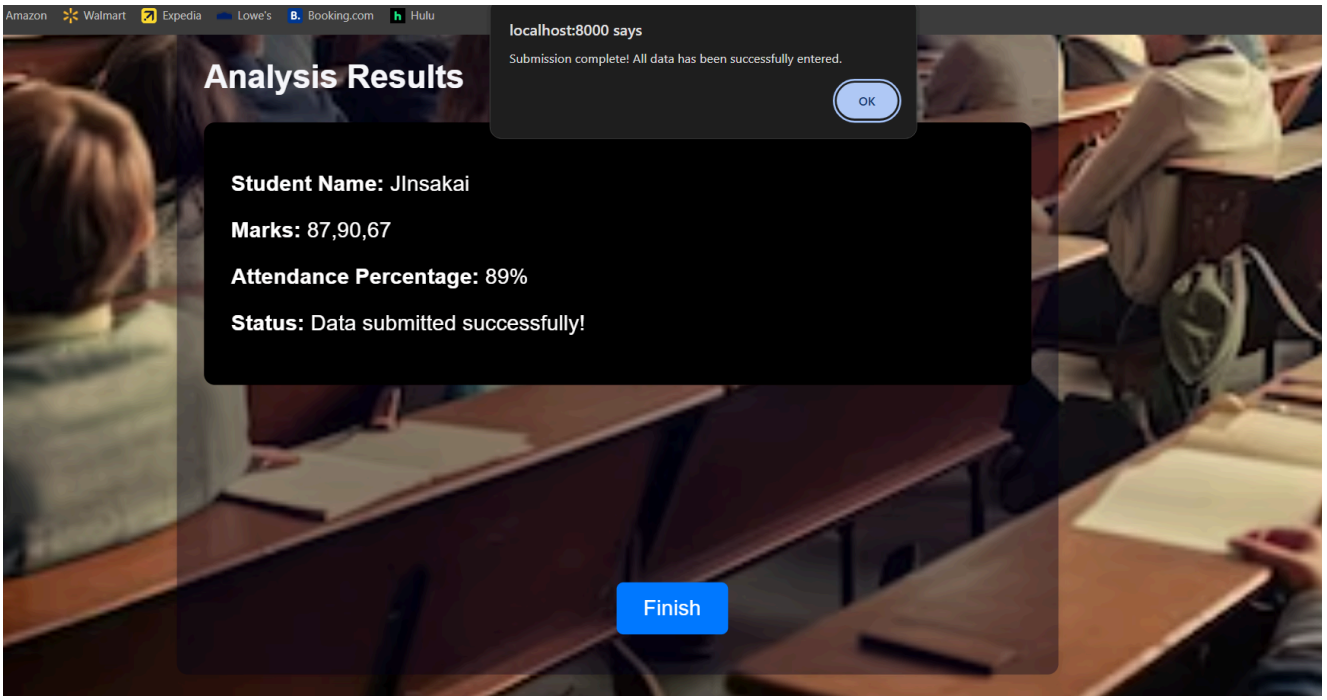
Inside Teacher Login



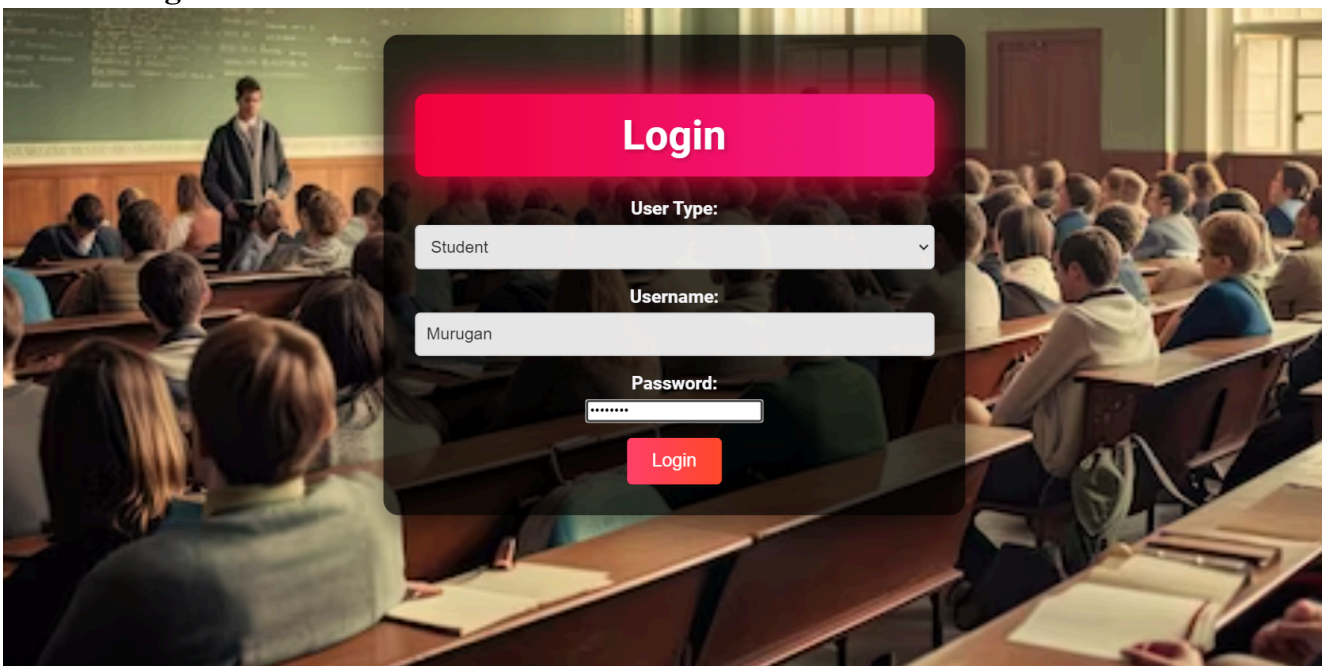
Teacher Entering the data

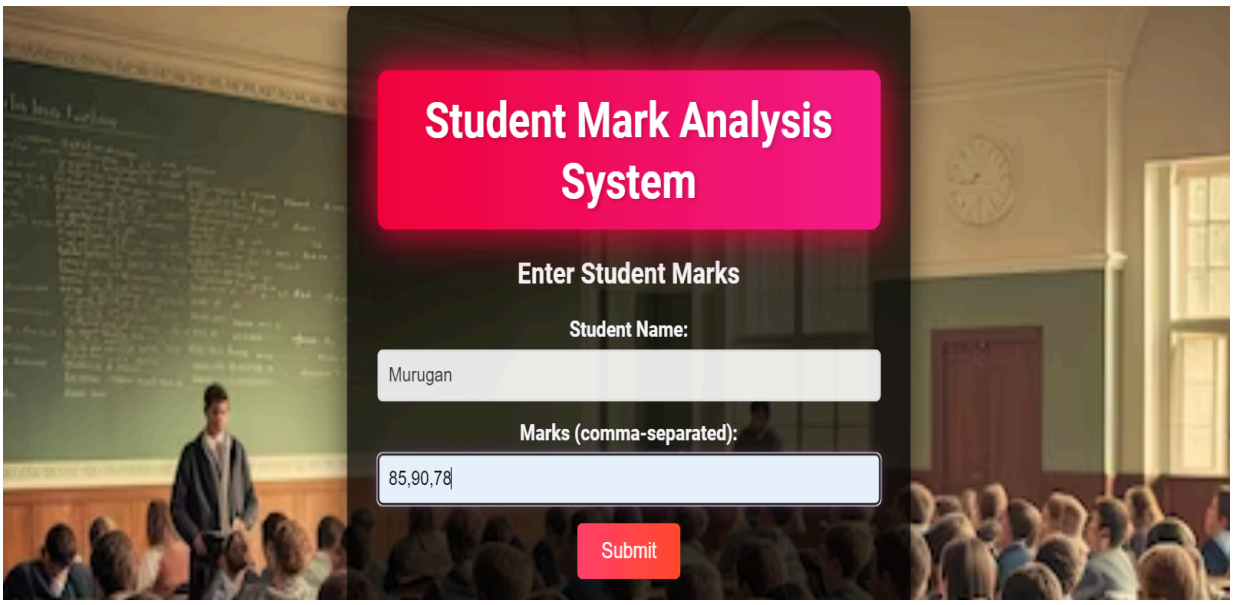


Successful data submission

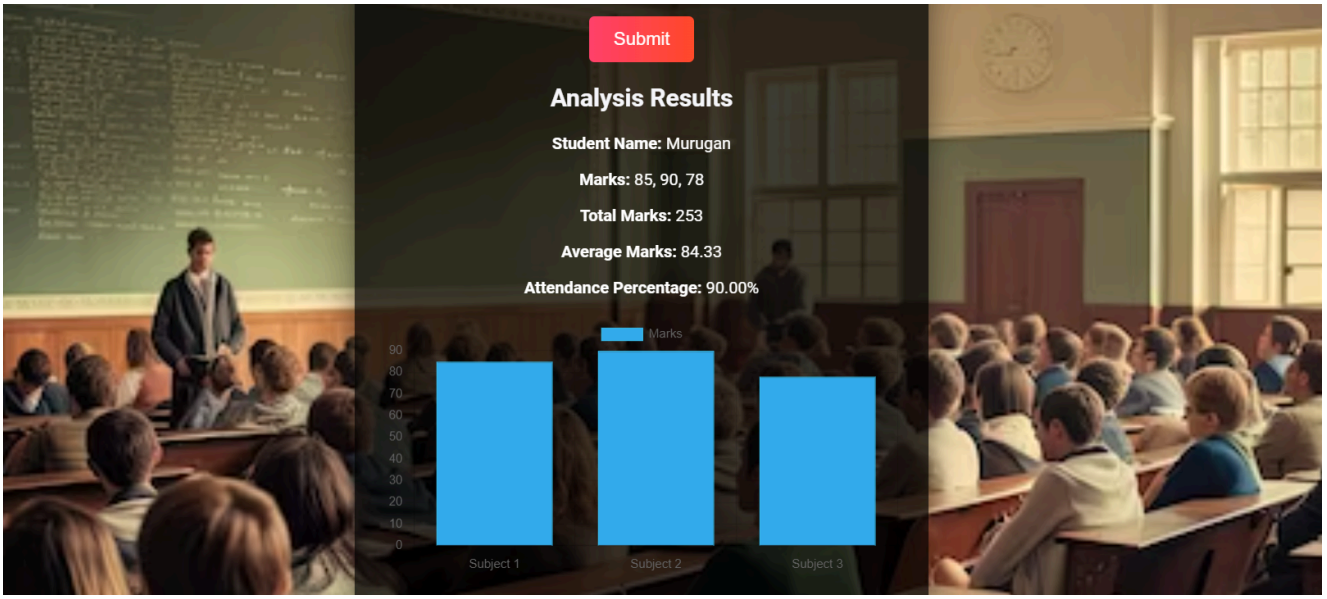


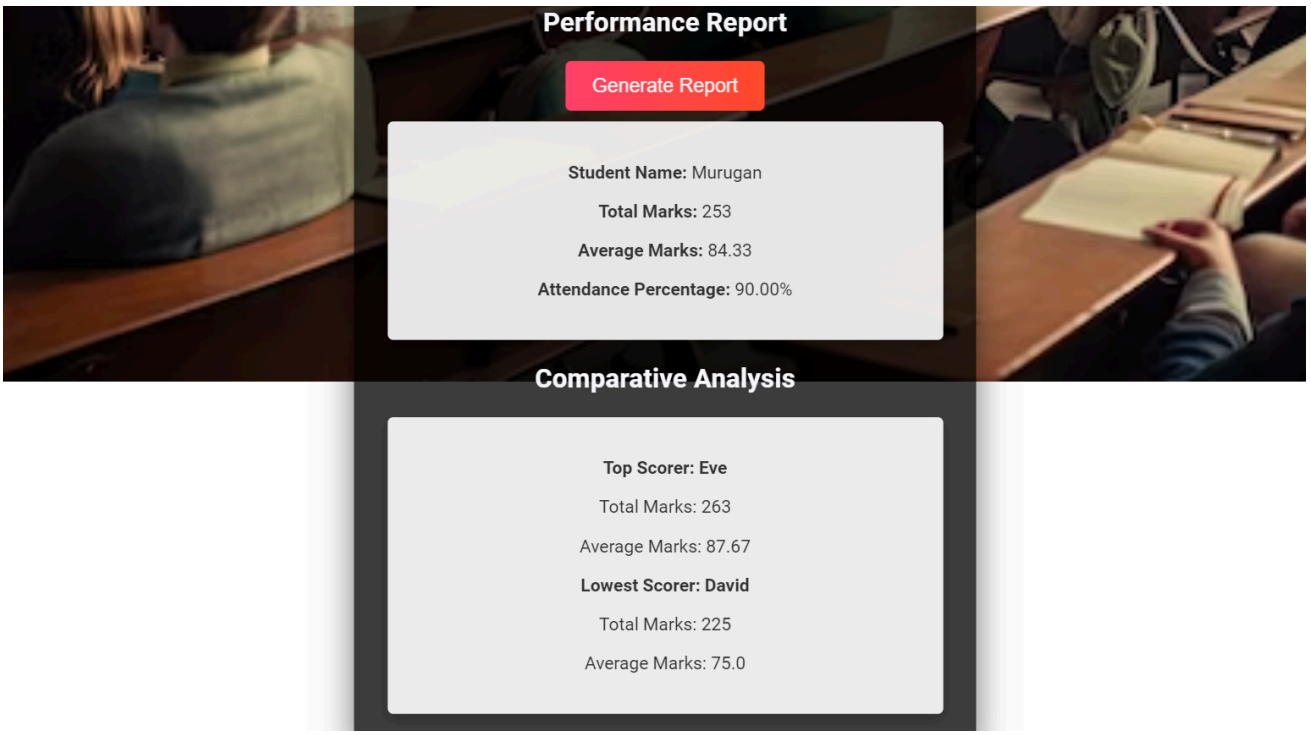
Student Login





Analysis of Student marks





4.2 Database Design

For the Student Mark Analysis System, the data is stored and retrieved from a MySQL database, which is chosen for its ability to handle structured data efficiently. The system uses a database to store student marks, perform statistical analyses, and manage administrator activities.

Database Design

Data Elements and Structures: At the analysis stage, the required data elements are identified, such as student names, marks, and statistical analysis results. These elements are structured and organized to facilitate storage and retrieval.

Normalization: The database schema undergoes normalization to ensure internal consistency, minimize redundancy, and optimize data storage. This process helps avoid unnecessary duplication of data and ensures that the database is scalable and efficient.

Data Integrity: Relationships between various data items are established, ensuring that data integrity is maintained. The normalization process helps in minimizing the chances of data inconsistencies and allows for easier updates.

MySQL Database:_ MySQL is selected due to its widespread use, efficiency, and flexibility in handling relational data. It supports quick, reliable, and flexible access to the data for various users (administrators) while minimizing data inconsistencies.

The database is designed to be efficient, minimizing storage requirements and ensuring high stability.

Student Mark Analysis System SQL Tables

	id	name	marks	total_marks	average
	Fil...	Filter	Filter	Filter	Filter
1	1	Alice	85, 90, 78	253	84.33
2	2	Bob	88, 76, 92	256	85.33
3	3	Charlie	95, 80, 85	260	86.67
4	4	David	70, 75, 80	225	75.0
5	5	Eve	90, 85, 88	263	87.67

Fig 4.2. SQL Database

4.3 IMPLEMENTATION(CODE)

//SERVER.JAVA

```
5   import com.sun.net.httpserver.HttpServer;
6   import com.sun.net.httpserver.HttpHandler;
7   import com.sun.net.httpserver.HttpExchange;
8   import java.io.IOException;
9   import java.io.OutputStream;
10  import java.net.InetSocketAddress;
11  import org.json.JSONObject;
12  import java.util.List;
13  import java.util.ArrayList;
14  import java.io.File;
15  import java.io.FileInputStream;
16
17  public class StartServer {
18
19      public static void main(String[] args) throws IOException {
20          HttpServer server = HttpServer.create(new InetSocketAddress(8000), 0);
21          StudentMarkAnalysis analysis = new StudentMarkAnalysis();
22
23          // Endpoint for inserting student marks
24          server.createContext("/api/insert", new HttpHandler() {
25              @Override
26              public void handle(HttpExchange exchange) throws IOException {
27                  if (exchange.getRequestMethod().equalsIgnoreCase("POST")) {
28                      JSONObject requestBody = new JSONObject(new
    String(exchange.getRequestBody().readAllBytes()));
29                      String name = requestBody.getString("studentName");
30
31                      // Convert JSONArray to List<Integer>
32                      List<Integer> marks = new ArrayList<>();
33                      requestBody.getJSONArray("marks").forEach(item -> {
34                          marks.add(((Number) item).intValue());  // Convert Object to
    Integer
35                      });
36
37                      analysis.insertStudentMarks(name, marks);
38
39                      String response = "{\"status\":\"success\"}";
40                      exchange.sendResponseHeaders(200, response.length());
41                      try (OutputStream os = exchange.getResponseBody()) {
42                          os.write(response.getBytes());
43                      }
44                  }
45              }
46          });
47
48          // Endpoint for calculating statistics
49          server.createContext("/api/calculateStatistics", new HttpHandler() {
50              @Override
51              public void handle(HttpExchange exchange) throws IOException {
52                  JSONObject responseJson = analysis.calculateStatistics();
53                  String response = responseJson.toString();
54                  exchange.sendResponseHeaders(200, response.length());
55                  try (OutputStream os = exchange.getResponseBody()) {
56                      os.write(response.getBytes());
57                  }
58              }
59          });
60
61          // Endpoint for getting top performer
62          server.createContext("/api/topPerformer", new HttpHandler() {
63              @Override
```

```
64                public void handle(HttpExchange exchange) throws IOException {
65                    JSONObject topPerformer = analysis.getTopPerformer();
66                    String response = topPerformer.toString();
67                    exchange.sendResponseHeaders(200, response.length());
68                    try (OutputStream os = exchange.getResponseBody()) {
69                        os.write(response.getBytes());
70                    }
71                }
72            });
73
74            // Endpoint for getting lowest performer
75            server.createContext("/api/lowestPerformer", new HttpHandler() {
76                @Override
77                public void handle(HttpExchange exchange) throws IOException {
78                    JSONObject lowestPerformer = analysis.getLowestPerformer();
79                    String response = lowestPerformer.toString();
80                    exchange.sendResponseHeaders(200, response.length());
81                    try (OutputStream os = exchange.getResponseBody()) {
82                        os.write(response.getBytes());
83                    }
84                }
85            });
86
87            // Endpoint for login
88            server.createContext("/api/login", new HttpHandler() {
89                @Override
90                public void handle(HttpExchange exchange) throws IOException {
91                    if (exchange.getRequestMethod().equalsIgnoreCase("POST")) {
92                        JSONObject requestBody = new JSONObject(new
    String(exchange.getRequestBody().readAllBytes()));
93                        String username = requestBody.getString("username");
94                        String password = requestBody.getString("password");
95                        String role = requestBody.getString("role");
96
97                        // Here, implement your login logic. For simplicity, assuming login
    is always successful
98                        JSONObject response = new JSONObject();
99                        response.put("status", "success");
100
101                       exchange.sendResponseHeaders(200, response.toString().length());
102                       try (OutputStream os = exchange.getResponseBody()) {
103                           os.write(response.toString().getBytes());
104                       }
105                   }
106               }
107           });
108
109           // Serve static files from the frontend directory
110           serveStaticFiles(server);
111
112           server.setExecutor(null);
113           server.start();
114           System.out.println("Server started on port 8000");
115       }
116
117       private static void serveStaticFiles(HttpServer server) {
118           server.createContext("/", exchange -> {
119               String requestedFile = exchange.getRequestURI().getPath().substring(1);
120               if (requestedFile.isEmpty()) {
121                   requestedFile = "login.html"; // Default file to login.html
122               }
123
124               File file = new File("frontend/" + requestedFile);
125               if (file.exists()) {
```

```
                    exchange.sendResponseHeaders(200, file.length());
                    try (FileInputStream fis = new FileInputStream(file);
                         OutputStream os = exchange.getResponseBody()) {
                        byte[] buffer = new byte[1024];
                        int bytesRead;
                        while ((bytesRead = fis.read(buffer)) != -1) {
                            os.write(buffer, 0, bytesRead);
                        }
                    }
                } else {
                    exchange.sendResponseHeaders(404, -1); // Not Found
                }
            });
        }
    }

```

//STUDENTMARKANALYSIS.JAVA

```
import java.sql.*;
import java.util.*;
import org.json.JSONObject;

public class StudentMarkAnalysis {

    // Method to insert student marks and return success message as JSONObject
    public JSONObject insertStudentMarks(String name, List<Integer> marks) {
        String url = "jdbc:sqlite:D:\\Java_Project\\backend\\Student.db";
        try (Connection conn = DriverManager.getConnection(url)) {
            String insertSQL = "INSERT INTO students (name, marks) VALUES (?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {
                for (int mark : marks) {
                    pstmt.setString(1, name);
                    pstmt.setInt(2, mark);
                    pstmt.executeUpdate();
                }
            }
            JSONObject response = new JSONObject();
            response.put("status", "success");
            return response;
        } catch (SQLException e) {
            e.printStackTrace();
            JSONObject response = new JSONObject();
            response.put("status", "error");
            response.put("message", e.getMessage());
            return response;
        }
    }

    // Method to calculate statistics and return as JSONObject
    public JSONObject calculateStatistics() {
        String url = "jdbc:sqlite:D:\\Java_Project\\backend\\Student.db";
        JSONObject result = new JSONObject();
        try (Connection conn = DriverManager.getConnection(url)) {
            String selectSQL = "SELECT marks FROM students";
            try (Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(selectSQL)) {

                List<Integer> marks = new ArrayList<>();
                int total = 0;
                while (rs.next()) {
```

```
                int mark = rs.getInt("marks");
                marks.add(mark);
                total += mark;
            }

            double mean = total / (double) marks.size();
            double variance = 0;
            for (int mark : marks) {
                variance += Math.pow(mark - mean, 2);
            }
            variance /= marks.size();
            double stdDev = Math.sqrt(variance);

            result.put("mean", mean);
            result.put("variance", variance);
            result.put("standardDeviation", stdDev);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        result.put("error", e.getMessage());
    }
    return result;
}

// Method to get top performer and return as JSONObject
public JSONObject getTopPerformer() {
    String url = "jdbc:sqlite:D:\\Java_Project\\backend\\Student.db";
    JSONObject result = new JSONObject();
    try (Connection conn = DriverManager.getConnection(url)) {
        String bestSQL = "SELECT name, marks FROM students ORDER BY marks DESC LIMIT 1";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(bestSQL)) {

            if (rs.next()) {
                result.put("name", rs.getString("name"));
                result.put("marks", rs.getInt("marks"));
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        result.put("error", e.getMessage());
    }
    return result;
}

// Method to get lowest performer and return as JSONObject
public JSONObject getLowestPerformer() {
    String url = "jdbc:sqlite:D:\\Java_Project\\backend\\Student.db";
    JSONObject result = new JSONObject();
    try (Connection conn = DriverManager.getConnection(url)) {
        String worstSQL = "SELECT name, marks FROM students ORDER BY marks ASC LIMIT 1";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(worstSQL)) {

            if (rs.next()) {
                result.put("name", rs.getString("name"));
                result.put("marks", rs.getInt("marks"));
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        result.put("error", e.getMessage());
```

```
        }
        return result;
    }
}
```

Conclusion:

The “Student Mark Analysis System” project, developed with attention to detail, the design emphasizes a seamless user experience while ensuring efficient functionality for administrators. Key features include adding, viewing, and managing student marks, with the ability to perform statistical analysis and generate reports. The system maintains robust security, especially in sensitive operations like modifying and removing administrator accounts, to ensure data integrity and safeguard against unauthorized access. This system is designed to meet the current needs of educational institutions while being adaptable for future improvements, ensuring long-term effectiveness and flexibility in handling student performance data.

Reference links:

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