

STUDENT MARK MANAGEMENT

CS23333 – Object Oriented Programming using Java

Submitted by

AKSHAYAA M -231001010

AKSHAYA M - 231001009

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE
BONAFIDE CERTIFICATE

Certified that this project titled “**STUDENT MARK MANAGEMENT**” is the bonafide work of “**AKSHAYAA M (231001010), AKSHAYA M (231001009)**” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.P.Valarmathie

HEAD OF THE DEPARTMENT

Information Technology

Rajalakshmi Engineering College

SIGNATURE

Mrs.Usha S

COURSE INCHARGE

Information Technology

Rajalakshmi Engineering College

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

INTERNAL EXAMINAR

EXTERNAL EXAMINAR

ABSTRACT

The Student Mark Analysis System is a software application developed to address the inefficiencies of traditional methods of managing academic records in educational institutions. Manual record-keeping and spreadsheet-based systems are often error-prone, time-consuming, and lack the scalability required to handle large datasets. This project offers a digital solution that automates the recording, storage, and analysis of student marks, providing a centralized and efficient platform for academic data management. The system is built using Java for backend logic and MySQL for database operations, ensuring secure, reliable, and scalable handling of academic data. It includes key features such as secure login for authorized access, automated grade computation, and comprehensive performance analysis. The system generates insightful reports, including subject-wise performance, class averages, and rankings, enabling educators to track trends, identify areas for improvement, and make informed decisions. By automating repetitive tasks, the system reduces the chances of human error and significantly saves time for administrators and educators. Its user-friendly interface simplifies data entry and report generation, ensuring accessibility even for users with limited technical expertise. The integration of modern technologies ensures data security and scalability, making it suitable for institutions of various sizes.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	iii
	LIST OF TABLES	iv
1	INTRODUCTION	1
	1.1 PROBLEM STATEMENT	1
	1.2 OBJECTIVE OF THE PROJECT	2
	1.3 ORGANIZATION OF THE REPORT	2
2	SYSTEM DESIGN	3
	2.1 OVERVIEW	3
	2.2 LIST OF MODULES	3
	2.3 UML MODELING	3
	2.3.1 USE CASE DIAGRAM OVERVIEW	3
	2.3.2 ENTITY-RELATIONSHIP DIAGRAM	5
	2.3.3 DATA FLOW DIAGRAM	6
	2.4 SYSTEM SPECIFICATION	7
	2.4.1 FUNCTIONAL REQUIREMENTS	8
3	DESIGN	9
	3.1 DESIGN	9
	3.2 DATABASE DESIGN	11
	3.3 IMPLEMENTATION (CODE)	13
4	CONCLUSION	18
	REFERENCES	19

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
2.1	Use Case Diagram	4
2.2	Entity Relationship Diagram	5
2.3	Data Flow Diagram	6
3.1	Login Page	9
3.2	Main page	9
3.3	Adding Student Marks	10
3.4	View Student Records	10

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
3.1	Student mark Table	11
3.2	User Table	12

CHAPTER 1

INTRODUCTION

1.1.PROBLEM STATEMENT

The *Student Mark Management System* for school students is designed to simplify and automate the process of managing and analyzing student performance data. This web-based system aims to eliminate manual record-keeping and make the management of student marks more efficient, accurate, and transparent. It allows school administrators to easily manage student records, input marks for various subjects, and generate performance reports, while providing students with secure access to their academic results and progress.

The *Admin Module* allows school administrators to add, update, and delete student and subject records. Admins can input student marks for different subjects, calculate total marks, grades, and generate performance reports, including class averages, top performers, pass/fail rates, and other academic insights. The system will also allow the export of these reports in formats such as PDF or Excel for easy distribution and analysis. The *Student Module* provides a secure login for students to view their marks, grades, and overall academic performance. Students can track their progress with visual graphs and performance charts, which show trends in their marks and provide insight into areas of improvement. Additionally, students will receive notifications about important events, such as upcoming exams or grade updates.

By automating the process of mark entry and report generation, the *Student Mark Management System* helps reduce administrative burden and the chances of errors. It provides a more efficient way for administrators to manage school records while giving students immediate access to their academic progress. This transparency allows students to monitor their performance, set academic goals, and identify areas for improvement. The system offers a more effective and reliable solution for managing student marks in schools, ensuring both accuracy and easy access for all users.

1.2. OBJECTIVE OF THIS PROJECT

The primary objective of this project is to develop a *Student Mark Analysis System* that simplifies the process of managing and analyzing student academic records. The system aims to provide a secure and efficient platform for educational institutions to store, process, and retrieve data related to student performance.

This project will enable administrators to manage student details, subjects, and marks while generating detailed performance insights through reports and visual analytics. The system will include user-friendly features for adding, updating, and deleting records, as well as analyzing class performance, identifying top-performing students, and monitoring pass/fail rates.

For students, the system will provide a secure login portal where they can access their individual academic records, including subject-wise marks, grades, and performance graphs. This ensures transparency and facilitates self-assessment.

By automating data management and analysis, the project seeks to reduce manual errors, save time, and enhance decision-making in educational settings. The system will be built using Java for the backend, MySQL for data storage, and a web-based interface for seamless interaction.

In summary, the *Student Mark Analysis System* is designed to enhance the accuracy, efficiency, and accessibility of student performance evaluation, benefiting both administrators and students.

1.3.ORGANIZATION OF THE REPORT

CHAPTER 1 – INTRODUCTION

CHAPTER 2 – SYSTEM DESIGN

CHAPTER 3 – IMPLEMENTATION

CHAPTER 4 – CONCLUSION

CHAPTER 2

SYSTEM DESIGN

2.1. OVERVIEW

The *Student Mark Management System* is a web-based application designed to streamline the process of managing and analyzing student academic performance. It provides a secure platform for storing, processing, and retrieving student data, including subject-wise marks, grades, and performance insights.

The application automates the manual process of mark entry and result analysis, ensuring accuracy, efficiency, and quick report generation. With a MySQL database for data storage, Java for backend logic, and a web interface for user interaction, the system is designed to handle data effectively and provide insightful analytics.

Key features include secure authentication, user-friendly interfaces, and tools for generating performance summaries like pass/fail rates and top scorer reports. This project is ideal for educational institutions looking to digitize their academic record management.

By replacing manual processes with automation, the system reduces errors, saves time, and improves decision-making for both administrators and students.

2.2. LIST OF MODULES

- 1.Authentication Module
- 2.Admin Module
- 3.Student Module
- 4.Mark Entry Module
- 5.Report Generation Module
- 6.Database Management Module
- 7.User Interface Module
- 8.Analytics Module
- 9.Security and Access Control Module

2.3 UML MODELING

2.3.1 USE CASE DIAGRAM OVERVIEW:

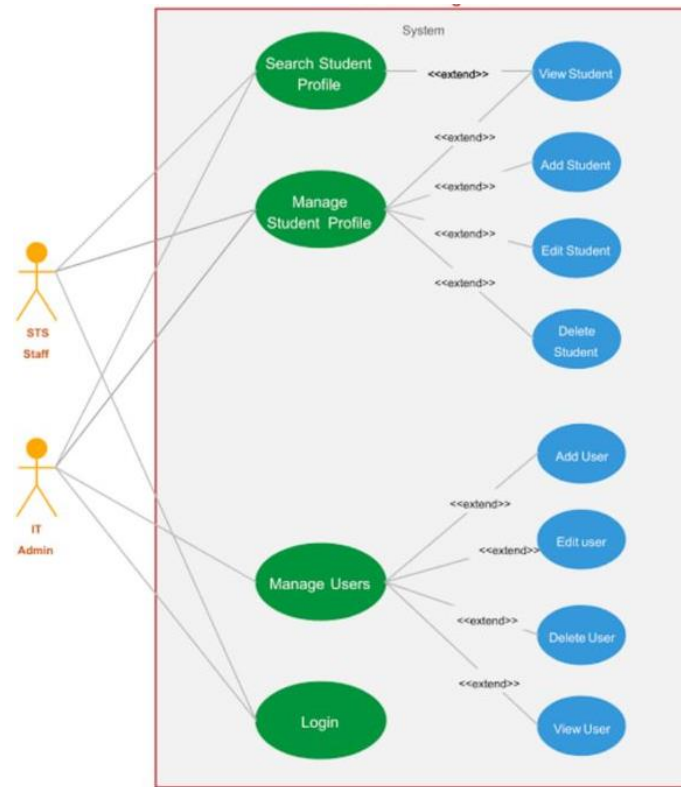


Figure 2.1. Use Case Diagram of student mark management

2.3.2 ENTITY RELATIONSHIP DIAGRAM:

The *Entity-Relationship (ER) Diagram* for the *Student Mark Management System* consists of several key entities and their relationships. The main entities include *Student*, *Subject*, *Marks*, *Admin*, and *Report*. The *Student* entity holds attributes like StudentID, Name, DateOfBirth, Email, etc. The *Subject* entity contains details about each subject, such as SubjectID, SubjectName, and Credits. The *Marks* entity stores the marks obtained by students in different subjects, with attributes like MarkID, StudentID (foreign key), SubjectID (foreign key), MarksObtained, and Grade. The *Admin* entity manages the system and holds attributes like AdminID, Username, and Role. Lastly, the *Report* entity generates performance reports for students, with attributes like ReportID, StudentID (foreign key), ClassAverage, TopScorer, and PassFailRate.

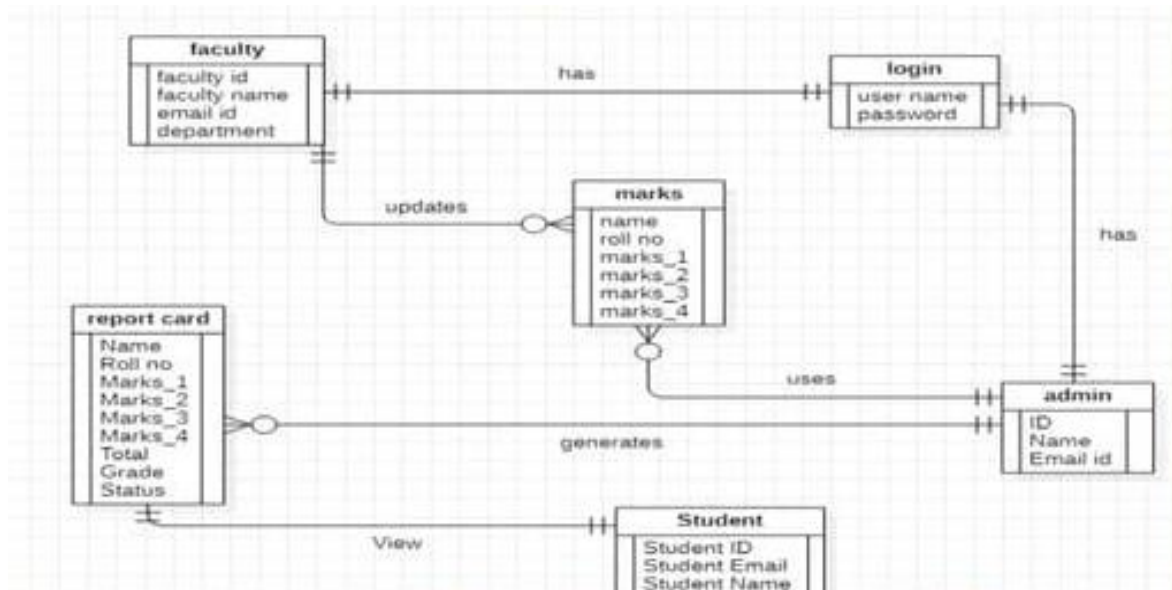


Figure 2.2. Entity Relationship diagram of student mark management

2.3.3. DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) for the Student Mark Management System represents the flow of data within the system, outlining how inputs are processed and how outputs are generated. Below is an overview of a Level 1 DFD for the system, showing the main processes, data stores, and external entities involved.

1. Admin

- Interacts with the system to manage student records, input marks, and generate reports.

Processes:

1. Student Authentication

- Students log in using their credentials. The system verifies credentials and allows access.

2. Mark Entry and Management

- Admins enter and manage student marks for different subjects.

3. Report Generation

- Admin generates performance reports (e.g., student results, class averages).

4. View Marks

- Students can view their marks and grades once logged in.

Data Stores:

1. Student Data

- Stores student records (names, IDs, contact details, etc.).

2. Marks Data

- Stores marks entered for each student and subject.

3. Report Data

- Stores generated reports based on student performance and overall class results.

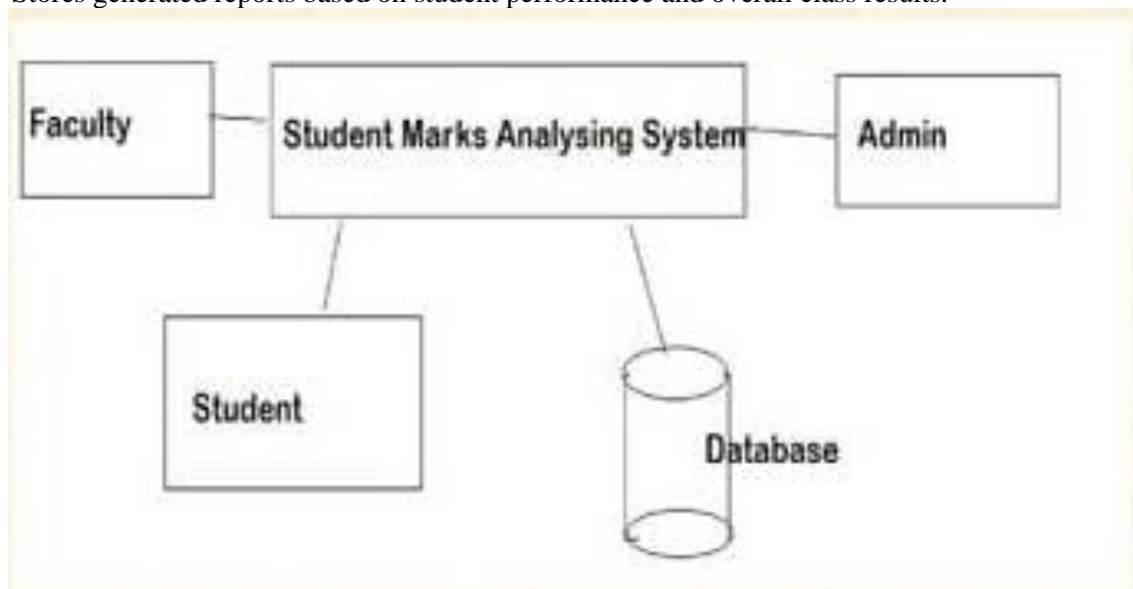


Figure 2.3. Data Flow Diagram of student mark management

2.4. SYSTEM SPECIFICATION

Frontend:

1. HTML: For structuring web pages.
2. CSS: For styling and creating responsive, visually appealing designs.
3. JavaScript: For adding interactivity and handling dynamic user actions.

Backend:

1. Programming Language: Java
2. Database Connectivity: JDBC (Java Database Connectivity)
3. Server: Built-in Java HTTP Server for handling client requests and responses.

Database:

1. Database Management System (DBMS): XAMPP
2. Database Features:
 - Tables for users, products, cart, and orders.
 - Constraints for data integrity (e.g., primary keys, foreign keys, unique constraints).

Development Tools:

1. Integrated Development Environment (IDE): NetBeans
2. Database Tools: Oracle SQL Developer Compatible with Windows
3. Browser Compatibility: Works on modern browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, for frontend interaction.

2.4.1. FUNCTIONAL REQUIREMENTS

1. User Authentication:

- Secure login for admins and students.
- Role-based access (admin: full access).

2. Admin Features:

- Add, update, and delete student records.
- Manage subjects (add/update/delete).
- Enter and update student marks.
- Generate and export academic performance reports (PDF/Excel).

3. Mark Entry & Calculation:

- Automatically calculate final grades.
- Validate entered marks (within valid range).

4. Reports:

- Individual performance reports for students.
- Class performance summary (averages, pass/fail rates).

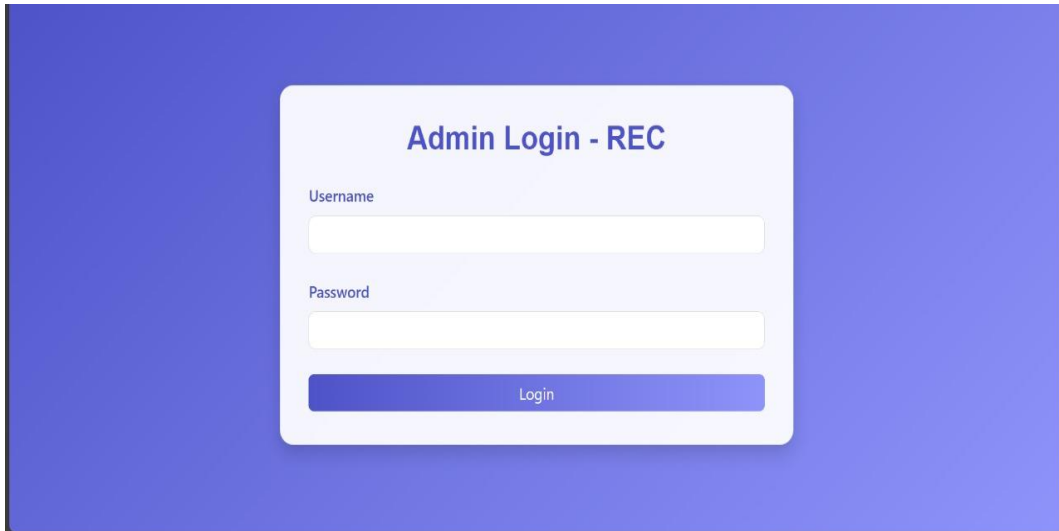
5. Data Security:

- Secure storage of sensitive data (marks, student info).
- Password encryption.

CHAPTER 3

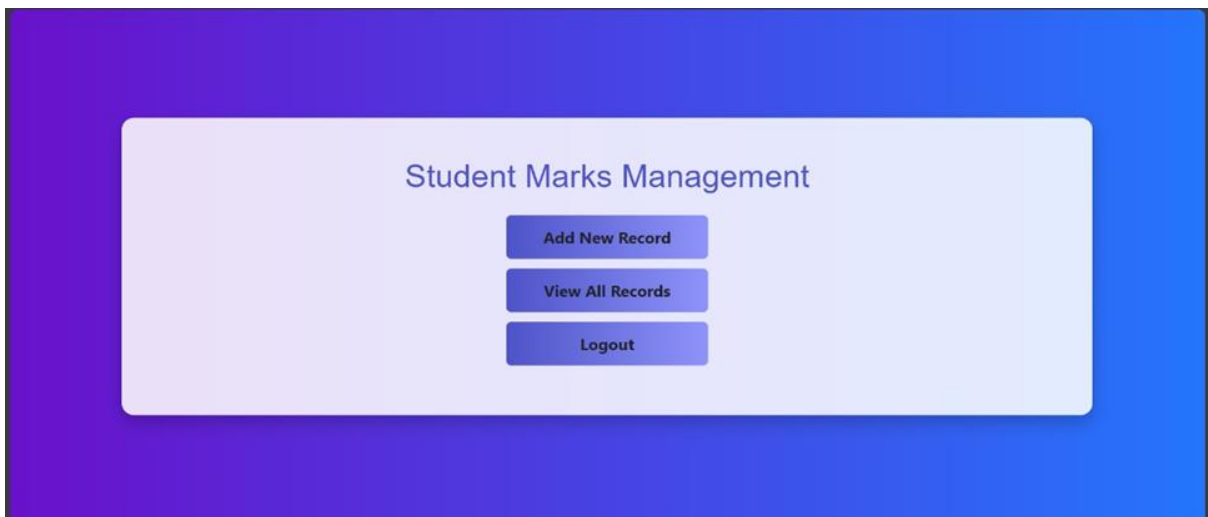
DESIGN

3.1.DESIGN



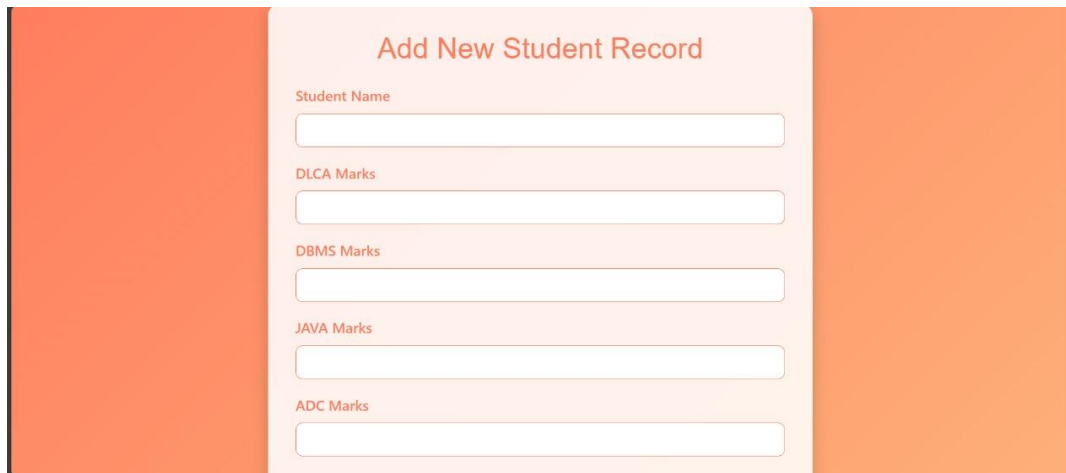
The image shows a login form titled "Admin Login - REC" centered on a blue gradient background. The form is a white rounded rectangle containing two input fields: "Username" and "Password", each with a light blue border. Below these fields is a blue "Login" button with white text.

Figure 3.1. Login Page



The image shows a main page titled "Student Marks Management" centered on a blue gradient background. The page features a white rounded rectangle containing three blue buttons with white text: "Add New Record", "View All Records", and "Logout", stacked vertically.

Figure 3.2.Main page



The image shows a web form titled "Add New Student Record". It is set against a light orange background with darker orange sidebars. The form itself is a white rectangle with a thin orange border. It contains five input fields, each with a label above it: "Student Name", "DLCA Marks", "DBMS Marks", "JAVA Marks", and "ADC Marks". All labels and the form title are in a reddish-orange color.

Figure 3.3.Add New Student Page

Name	DLCA	DBMS	JAVA	ADC	DAA	Total	Average	Result	Actions
akshayaa	90	100	89	95	50	424	84.8	Pass	Delete
bala	90	80	60	86	50	366	73.2	Pass	Delete
uma	60	99	100	78	90	427	85.4	Pass	Delete
anu	86	85	98	86	75	430	86.0	Pass	Delete
priya	5	6	6	8	7	32	6.4	Fail	Delete
sneha	86	87	96	85	78	432	86.4	Pass	Delete
abi	5	5	5	5	5	25	5.0	Fail	Delete

Figure 3.4.view Record Page

3.2 DATABASE DESIGN

1. Students Table

This table holds the core details of students and their marks.

Fields:.

- a. ID: Unique identifier for the student.
- b. STUDENT_NAME: Name of the student.
- c. DLCA: Marks for the subject Digital Logic and Computer Architecture (DLCA).
- d. DBMS: Marks for the subject Database Management System (DBMS).
- e. JAVA: Marks for the subject Java Programming.
- f. ADC: Marks for the subject Analog and Digital Circuits (ADC).
- g. DAA: Marks for the subject Design and Analysis of Algorithms (DAA).
- h. TOTAL: Total marks obtained across all subjects.
- i. AVERAGE: Average marks across all subjects.
- j. RESULT: Result status (PASS or FAIL), determined by minimum marks in each subject.

id	student_name	DLCA	DBMS	JAVA	ADC	DAA	total	average	result
1	akshayaa	90	100	89	95	50	424	84.80	Pass
2	bala	90	80	60	86	50	366	73.20	Pass
3	uma	60	99	100	78	90	427	85.40	Pass
4	anu	86	85	98	86	75	430	86.00	Pass
5	priya	5	6	6	8	7	32	6.40	Fail
6	sneha	86	87	96	85	78	432	86.40	Pass
10	abi	5	5	5	5	5	25	5.00	Fail

Table 3.1. Students Table

2. User Table

This table maintains secure login credentials for students to access their records.

Fields:

- a. ID: Unique identifier for the user.
- b. USERNAME: Username for login authentication.
- c. PASSWORD: Password for login authentication.
- d. FULL_NAME: Full name of the user.
- e. EMAIL: Email address of the user.
- f. ROLE: Role of the user (Admin, Student, etc.).

id	username	password	full_name	email	role
1	akshayaa	akshayaa123	akshayaamurugan	akshayaa@gmail.com	admin
2	akshaya	akshaya123	akshayamohan	akshaya@gmail.com	admin

Table 3.2. User Table

3.3. IMPLEMENTATION (CODE)

1. Server Initialization

```
package com.studentmarks.servlet;
import com.studentmarks.dao.UserDAO;
import com.studentmarks.model.User;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
public class LoginServlet extends HttpServlet {
    private UserDAO userDAO;

    public void init() {
        userDAO = new UserDAO();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // Handle logout
        String logout = request.getParameter("logout");
        if (logout != null) {
            HttpSession session = request.getSession(false);
            if (session != null) {
                session.invalidate();
            }
            response.sendRedirect(request.getContextPath() + "/login.jsp");
            return;
        }

        // If user is already logged in, redirect to index
        HttpSession session = request.getSession(false);
        if (session != null && session.getAttribute("user") != null) {
            response.sendRedirect(request.getContextPath() + "/student");
            return;
        }
    }
}
```

2. User Login

```
request.getRequestDispatcher("/login.jsp").forward(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

```

String username = request.getParameter("username");
String password = request.getParameter("password");

try {
    User user = userDAO.authenticateUser(username, password);

    if (user != null) {
        // Create session and add user
        HttpSession session = request.getSession();
        session.setAttribute("user", user);
        session.setAttribute("username", user.getUsername());

        // Redirect to main page
        response.sendRedirect(request.getContextPath() + "/student");
    } else {
        // Authentication failed
        request.setAttribute("errorMessage", "Invalid username or password");
        request.getRequestDispatcher("/login.jsp").forward(request, response);
    }
} catch (Exception e) {
    // Handle database errors
    request.setAttribute("errorMessage", "Database error: " + e.getMessage());
    request.getRequestDispatcher("/login.jsp").forward(request, response);
}
}
}

```

3.Database connection

```

import com.studentmarks.dao.StudentDAO;
import com.studentmarks.model.Student;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;

@WebServlet("/student/*")
public class StudentServlet extends HttpServlet {
    private StudentDAO studentDAO;

    @Override
    public void init() {
        studentDAO = new StudentDAO();
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String action = request.getPathInfo();
    }
}

```

```

    try {
        switch (action) {
            case "/add":
                addStudent(request, response);
                break;
            case "/update":
                updateStudent(request, response);
                break;
            case "/delete":
                deleteStudent(request, response);
                break;
            default:
                response.sendRedirect(request.getContextPath() + "/index.jsp");
                break;
        }
    } catch (IOException | SQLException ex) {
        throw new ServletException(ex);
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String action = request.getPathInfo();
    try {
        if (action == null) {
            listStudents(request, response);
        } else {
            switch (action) {
                case "/edit":
                    showEditForm(request, response);
                    break;
                default:
                    listStudents(request, response);
                    break;
            }
        }
    } catch (IOException | SQLException | ServletException ex) {
        throw new ServletException(ex);
    }
}

private void listStudents(HttpServletRequest request, HttpServletResponse response)
    throws SQLException, ServletException, IOException {
    request.setAttribute("students", studentDAO.getAllStudents());
    request.getRequestDispatcher("/index.jsp").forward(request, response);
}

private void showEditForm(HttpServletRequest request, HttpServletResponse response)
    throws SQLException, ServletException, IOException {

```

```

        int id = Integer.parseInt(request.getParameter("id"));
        Student student = studentDAO.getStudentById(id);
        request.setAttribute("student", student);
        request.getRequestDispatcher("/form.jsp").forward(request, response);
    }

    private void addStudent(HttpServletRequest request, HttpServletResponse response)
        throws SQLException, IOException {
        String name = request.getParameter("studentName");

        // Get marks for the five subjects
        int DLCA = Integer.parseInt(request.getParameter("DLCA"));
        int DBMS = Integer.parseInt(request.getParameter("DBMS"));
        int JAVA = Integer.parseInt(request.getParameter("JAVA"));
        int ADC = Integer.parseInt(request.getParameter("ADC"));
        int DAA = Integer.parseInt(request.getParameter("DAA"));

        // Create a new Student object with the data
        Student student = new Student(name, DLCA, DBMS, JAVA, ADC, DAA);
        studentDAO.addStudent(student);

        // Redirect to the student list
        response.sendRedirect(request.getContextPath() + "/student");
    }

    private void updateStudent(HttpServletRequest request, HttpServletResponse response)
        throws SQLException, IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        String name = request.getParameter("studentName");

        // Get marks for the five subjects
        int DLCA = Integer.parseInt(request.getParameter("DLCA"));
        int DBMS = Integer.parseInt(request.getParameter("DBMS"));
        int JAVA = Integer.parseInt(request.getParameter("JAVA"));
        int ADC = Integer.parseInt(request.getParameter("ADC"));
        int DAA = Integer.parseInt(request.getParameter("DAA"));

        // Create a new Student object with the data
        Student student = new Student(name, DLCA, DBMS, JAVA, ADC, DAA);
        student.setId(id);
        studentDAO.updateStudent(student);

        // Redirect to the student list
        response.sendRedirect(request.getContextPath() + "/student");
    }

    private void deleteStudent(HttpServletRequest request, HttpServletResponse response)
        throws SQLException, IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        studentDAO.deleteStudent(id);
    }

```

```
// Redirect to the student list
response.sendRedirect(request.getContextPath() + "/student");
}}
```


CHAPTER 4

CONCLUSION

In conclusion, the *Student Mark Analysis System* is a comprehensive and efficient solution for managing and analyzing student academic performance. By automating the process of recording marks, calculating grades, and generating performance reports, the system reduces manual effort, minimizes errors, and enhances the accuracy of academic data.

The system's use of a *Java* backend and *MySQL* database ensures robust data storage, security, and scalability, while the frontend, built with *HTML*, **CSS, and **JavaScript*, offers a user-friendly and intuitive interface. The inclusion of role-based access control further strengthens the security, ensuring that sensitive student data is only accessible by authorized users.

This project not only streamlines the administrative tasks of educational institutions but also provides teachers and administrators with valuable insights into student performance. The ability to generate detailed reports helps identify areas of improvement for students, contributing to better learning outcomes. Furthermore, the system's scalability makes it suitable for educational institutions of all sizes, from small schools to large universities.

Overall, the *Student Mark Analysis System* serves as a powerful tool for educational institutions, improving efficiency, security, and decision-making. It facilitates better academic performance tracking, allowing educators to focus more on student development and less on administrative work. The system's implementation is a significant step toward modernizing the management of academic data and enhancing the overall educational experience.

REFERENCES

- 1.<https://github.com/codegenius2/Student-Mark-Management>
- 2.<https://www.codewithfaraz.com/article/131/explore-50-java-projects-with-source-code-for-all-skill-levels>
- 3.<https://beginnersbook.com/2017/09/java-program-to-calculate-and-display-student-grades/>
- 4.<https://www.studentprojects.live/>
- 5.<https://www.javaguides.net//2020/02/student-management-system-project-in-java-swing.html>
- 6.<https://codeziips.com/java/java-projects-with-source-code-free-download-2022/>
- 7.https://github.com/kaligu/Students_marks_management_system
- 8.https://github.com/AmbarZaidi/Student_Result_Processing_System
- 9.<https://github.com/Ayan7439/Student-Grade-Management-System-using-JAVA>
- 10.<https://github.com/uts58/Student-Grade-Management-System>