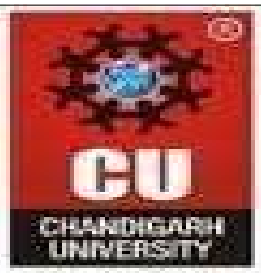


UNIVERSITY INSTITUTE of COMPUTING

Asia's Fastest Growing University

Students Result Management System

https://github.com/hitteshkharyal/student_result_management_system.git **(Mini project)**

Submitted by -

Hitesh Kumar (24MCA20256)

In

Advanced Internet Programming Lab (24CAP-652)

Master Of Computers Application

University Institute Of Computing

Chandigarh University

Submitted to -

Assistant Professor Rohini

Master Of Computers Application

University Institute Of Computing

Chandigarh University

Academic Year: 2024 - 2025

Acknowledgement

I would like to express my heartfelt gratitude to all those who supported and guided me throughout the completion of this project.

First and foremost, I thank the Almighty for giving me the strength, patience, and wisdom to complete this project successfully.

I extend my sincere thanks to **Assistant Professor Rohini**, my project guide, for their valuable guidance, constant encouragement, and constructive feedback throughout the development of the **Student Result Management System**.

I am also grateful to the faculty members, whose insights and support have been instrumental in shaping this project.

I would also like to acknowledge the support of my friends and classmates who provided constant motivation and helped me test and review the application.

Lastly, I thank my family for their unconditional love, motivation, and support throughout the course of my academic journey.

Table of Contents

1. Introduction

The Student Result Management System (SRMS) is a comprehensive web-based application designed to manage and automate the academic result operations of students in an educational institution. In many schools, colleges, and universities, the process of handling student data—such as storing personal information, calculating marks, assigning grades, and publishing results—is still carried out manually or via basic spreadsheets. These conventional methods are often inefficient, time-consuming, error-prone, and challenging to scale or maintain. SRMS aims to overcome these limitations through digitization. This project provides an efficient solution by introducing a centralized, secure, and user-friendly system where all academic results are stored, processed, and retrieved digitally. The system is developed using **Java (JSP and Servlets)** for the backend logic, **JavaBeans** for modular data handling, and **MySQL** as the database for storing student and result information. The front-end is kept clean and responsive using HTML, CSS, and embedded JSP scripts.

The system operates with two types of users: **students** and **administrators**. Students can register themselves (only when registration is enabled by the admin), log in to their dashboard, and view their academic results, including marks in individual subjects, total marks, and grade. Administrators have a more powerful interface where they can add or edit student records, insert subject-wise marks, automatically calculate grades based on total marks, and view all student data. A special feature of the admin dashboard is the **registration toggle button**, which allows the admin to enable or disable student registration at any time, offering complete control over when students can join the system.

The key motivation behind building this system is to **reduce administrative workload**, **minimize errors**, and **provide students with transparent and quick access to their results**. The system also improves data security, as only authorized users (students or admins) can access their respective parts of the system after successful authentication.

Furthermore, the SRMS is designed to be **scalable**, so that additional features such as attendance tracking, assignment uploads, or even performance analytics can be integrated in the future. Its modular design and use of standard web technologies make it an ideal project for educational institutions looking to modernize their result management process.

In summary, the SRMS provides a complete digital solution for managing academic records and results, enhancing the efficiency, reliability, and user experience for both students and administrators.

2. Project Objective

The primary objective of the *Student Result Management System (SRMS)* is to design and implement a secure, efficient, and user-friendly platform that simplifies and automates the academic result management process in educational institutions. Traditional methods of managing student results involve manual entries, paperwork, and a higher risk of human errors. These outdated practices not only consume time and effort but also make it difficult to maintain the integrity and accuracy of data over time. SRMS addresses these challenges by bringing automation, structure, and accessibility to the forefront.

Key Objectives of the Project:

1. **Automation of Result Processing**

One of the main goals is to reduce the manual effort involved in managing results. By automating mark calculation, grade assignment, and data storage, the system saves time and ensures consistent and accurate results.

2. **Centralized Student Information Management**
   The system provides a single platform where all student-related academic data is stored. This includes personal details like roll number, name, and course, as well as academic records such as subject-wise marks, total marks, and grades.

3. **Admin Control & Data Security**
   The application ensures that only authenticated admins can add, edit, or view sensitive academic records. It also provides an admin-only feature to control the availability of the student registration system via an ON/OFF toggle, which enhances operational flexibility.

4. **Student Self-Service Portal**
   The system empowers students by allowing them to register (when enabled), log in securely, and view their results anytime, anywhere. This removes the dependency on administrative staff for result inquiries and enhances transparency.

5. **Scalability & Modularity**
   The project is developed with a modular architecture using JavaBeans, JSP, and Servlets, which makes it easy to maintain and scale. Future enhancements like attendance tracking, notifications, or performance reports can be integrated seamlessly.

6. **Data Accuracy and Validation**
   Through proper validation and use of database constraints, the system ensures that incorrect or duplicate entries are avoided. Marks entered are validated, and grades are calculated dynamically based on business logic to ensure accuracy.

7. **User Experience and Interface Design**
   A clean and responsive UI is provided for both admin and student interfaces, ensuring ease of navigation and usability. The user interface is designed using basic HTML and CSS with simple interactions, making it accessible even to users with limited technical knowledge.

8. **Database Integration and Security**
   The use of MySQL as the backend ensures robust and structured data storage. It also allows the use of primary keys, foreign keys, and SQL queries to maintain data relationships and integrity.

3. Literature Review

The evolution of student result management systems is deeply rooted in the broader development of information technology and its application in educational institutions. This section reviews the existing literature, tools, and technologies that form the foundation of our Student Result Management System (SRMS), highlighting the gaps in traditional systems and how modern technologies bridge them.

Traditional Result Management Approaches

Historically, academic results were managed using paper-based methods. Institutions maintained student records in physical ledgers and report cards, which were prone to damage, loss, or unauthorized access. Updating records manually was time-consuming and often led to errors. Retrieving past data or generating performance reports required intensive labor and effort, especially in institutions with thousands of students.

As institutions started digitizing their processes, early desktop-based systems were introduced. These systems, although helpful, lacked network connectivity, multi-user access, and web integration. They were mostly limited to single-computer usage, which restricted their scalability and accessibility.

Modern Web-Based Systems

With the growth of the internet, institutions began adopting **web-based student information systems** that allowed real-time access to data and facilitated remote interaction between students, faculty, and administrative staff. These systems provided features such as online result publishing, student registration, and administrative dashboards.

Our project draws inspiration from these modern systems but focuses on simplicity, data security, and role-based access. Unlike enterprise-level systems, which are often expensive and complex, this project aims to offer essential functionalities using lightweight tools and open-source technologies.

Relevant Technologies and Frameworks

Several programming languages and frameworks have contributed significantly to the development of web-based student information systems:

- **Java EE (Jakarta EE)**: Java is widely recognized for its portability, robustness, and security features. Using Servlets and JavaBeans in our project ensures a modular design where business logic, data access, and presentation layers are cleanly separated.

- **JSP (JavaServer Pages)**: JSP is used for rendering dynamic content on web pages, allowing seamless interaction with the backend. This enhances the user experience while keeping the structure clear.

- **MySQL Database**: As a reliable and open-source relational database management system, MySQL offers features like ACID compliance, indexing, and referential integrity, making it suitable for storing structured data such as student records and result details.

- **HTML/CSS and JavaScript**: These front-end technologies help in designing interactive and responsive interfaces, enhancing the overall usability of the system.

Security and Role Management in Literature

Literature also emphasizes the importance of **role-based access control** and **data privacy** in academic systems. In our system, this is achieved by:

- Restricting administrative functionalities (like adding marks or toggling registration) to verified admins.
- Providing students with login-based access to view their own results, preventing unauthorized access to other students' data.

Gap Analysis and Justification

While several existing systems offer student result management, many are either too simplistic or overly complex for small-to-medium institutions. Our SRMS fills the gap by providing:

- A balanced solution that is neither too heavy nor too minimal.
- A toggle-based registration system, which is often missing in free solutions.
- A clean admin dashboard integrated with result input and grade calculation logic.

4. System Requirements

To ensure smooth development and deployment of the Student Result Management System, both hardware and software requirements were considered.

Hardware Requirements:

- Processor: Intel Core i3 or above
- RAM: 4 GB minimum (8 GB recommended)
- Hard Disk: 100 GB
- Display: 1366x768 resolution or higher

Software Requirements:

- Operating System: Windows/Linux
- IDE: NetBeans or Eclipse
- Web Server: Apache Tomcat 9 or above
- JDK: Java SE Development Kit 8 or above
- Database: MySQL 5.7 or above
- Browser: Chrome/Firefox (for testing UI)

The selected tools are open-source and widely supported, ensuring accessibility and ease of integration.

5. System Design.

The system design outlines how various components of the Student Result Management System interact with each other. It focuses on the structure, flow of data, and user roles.

Architecture Overview:

The system follows a three-tier architecture:

1. **Presentation Layer** (Frontend): Built using JSP and HTML/CSS for user interaction.
2. **Business Logic Layer** (Servlets + JavaBeans): Handles core logic like result computation, form processing, and session management.
3. **Data Layer** (MySQL): Manages persistent storage for student data, results, and admin access.

Key Design Components:

- **Student Module**:
  - Registration page with on/off control from admin
  - Login to check results
- **Admin Module**:
  - Secure login system
  - Dashboard to manage student data and results
  - Control switch for student registration
- **Database Design**:
  - `students`: Contains roll number, name, course, password
  - `result`: Holds marks, total, and grade
  - `registration_status`: Controls availability of registration

6. Implementation:-

Github Link - https://github.com/hitteshkharyal/student_result_management_system.git

The Student Result Management System was implemented using a **Model-View-Controller (MVC)** architecture. It separates business logic, user interface, and data access for modular development. Below is a breakdown of both the **front-end** and **back-end** implementations.

6.1 Front-End Development (JSP + HTML + CSS)

The front-end was designed using **JSP** pages, styled with **CSS**, and contains embedded Java for dynamic data display.

Student Registration Page (register.jsp):

<form action="../AddStudentServlet" method="post">

```
    Roll No: <input type="text" name="roll"><br>
    Name: <input type="text" name="name"><br>
    Course: <input type="text" name="course"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Register">
</form>
```

CSS Snippet:

```
.form-box {
    background-color: rgba(255, 255, 255, 0.1);
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px #000;
}
```

Registration Toggle Button (in dashboard.jsp):

```
CopyEdit
<form method="post" action="ToggleRegistrationServlet">
    <input type="submit" value="<%= isRegistrationOpen ? "Close Registration" : "Open Registration"
%>">
</form>
```

6.2 Back-End Development (Servlets + JavaBeans + JDBC)

The back-end includes **Servlets** to handle requests, **JavaBeans** for data modeling, and **JDBC** for database interaction.

1. AddStudentServlet.java

```
java
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String roll = request.getParameter("roll");
    String name = request.getParameter("name");
    String course = request.getParameter("course");
    String password = request.getParameter("password");

    Student s = new Student(roll, name, course, password);
    boolean status = StudentDAO.registerStudent(s);

    if (status) {
        response.sendRedirect("added.jsp");
    } else {
        response.sendRedirect("error.jsp");
    }
}
```

2. StudentDAO.java

```
public static boolean registerStudent(Student s) {
   try {
      Connection con = DBUtil.getConnection();
      PreparedStatement ps = con.prepareStatement(
         "INSERT INTO students (roll_no, name, course, password) VALUES (?, ?, ?, ?)");
      ps.setString(1, s.getRollNo());
      ps.setString(2, s.getName());
      ps.setString(3, s.getCourse());
      ps.setString(4, s.getPassword());

      int i = ps.executeUpdate();
      return i > 0;
   } catch (Exception e) {
      e.printStackTrace();
      return false;
   }
}
```

3. ToggleRegistrationServlet.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
   boolean currentStatus = RegistrationDAO.getStatus();
   RegistrationDAO.setStatus(!currentStatus);
   response.sendRedirect("dashboard.jsp");
}
```

4. RegistrationDAO.java

```
public static boolean getStatus() {
   try {
      Connection con = DBUtil.getConnection();
      ResultSet rs = con.createStatement().executeQuery("SELECT status FROM registration_status
LIMIT 1");
      return rs.next() && rs.getBoolean("status");
   } catch (Exception e) {
      return false;
   }
}

public static void setStatus(boolean status) {
   try {
      Connection con = DBUtil.getConnection();
      PreparedStatement ps = con.prepareStatement("UPDATE registration_status SET status=?");
      ps.setBoolean(1, status);
      ps.executeUpdate();
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

This modular code design ensures the system is easy to maintain, secure, and extendable. It also enforces a clear separation between UI, business logic, and database operations.

7. Core Features

The Student Result Management System was designed with core features that ensure effective interaction between students and administrators, making result handling seamless and efficient. Below are the major features categorized by user roles:

7.1 For Students:

- **Student Registration**
  Students can register themselves by entering roll number, name, course, and password. This registration is controlled by the admin (can be turned on/off).

- **Secure Login**
  Registered students can securely log in using their roll number and password to view their results.

- **Result View**
  Once logged in, students can view their subject-wise marks, total, and grade, if available.

7.2 For Admins:

- **Admin Authentication**
  Only registered admins can access the admin dashboard, ensuring system control remains with authorized personnel.

- **Dashboard Access**
  Admins have a user-friendly dashboard displaying a list of all registered students, with editable fields for marks entry.

- **Add/Edit Marks**
  Admins can input or update marks for three subjects for each student. The system automatically calculates total and grade.

- **Result Management**
  The system checks if a student's result exists:

  - If yes, it updates the result.
  - If no, it inserts a new record.

- **Registration Toggle**
  Admins can enable or disable student registration through a toggle button in the dashboard. This

adds an extra layer of control.

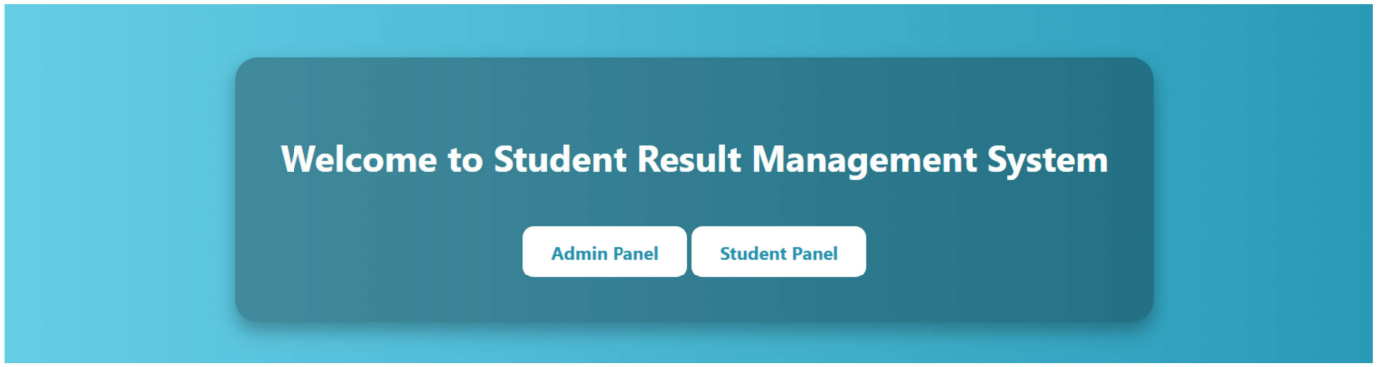
7.3 Data Validation & Feedback:

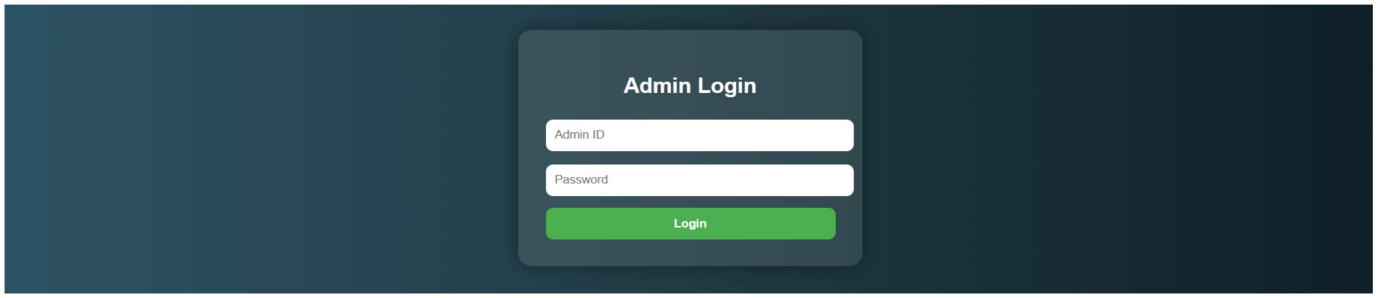
- **Validation**
  The system validates all numeric inputs for marks and ensures the required fields are filled.

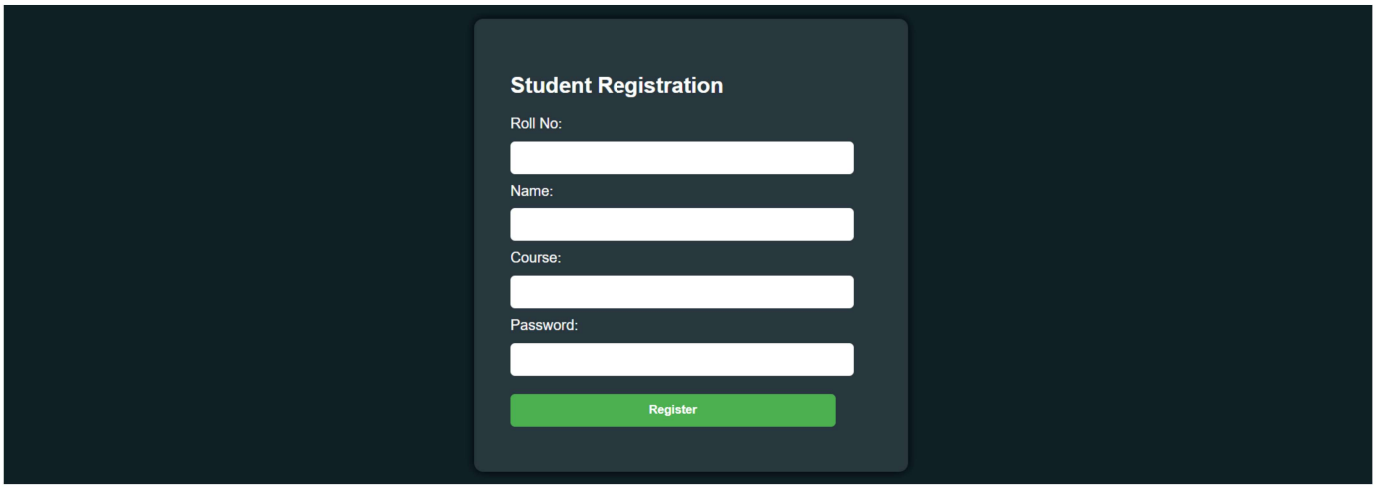
- **User Feedback**
  Success or error messages are displayed to users after each action (e.g., “Registered Successfully”, “Error: Duplicate Entry”).

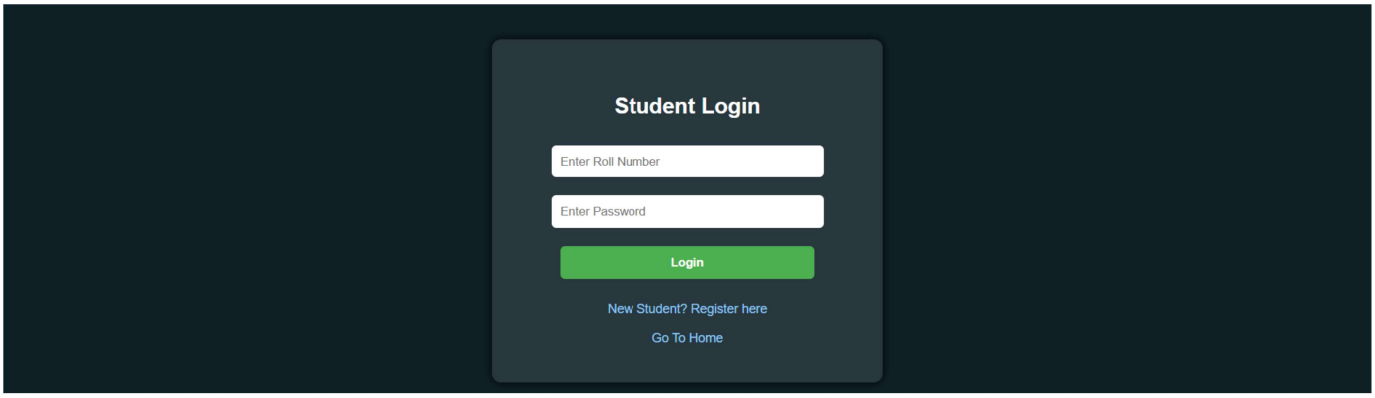
These core features not only improve the functionality of the system but also enhance the user experience and administrative efficiency.

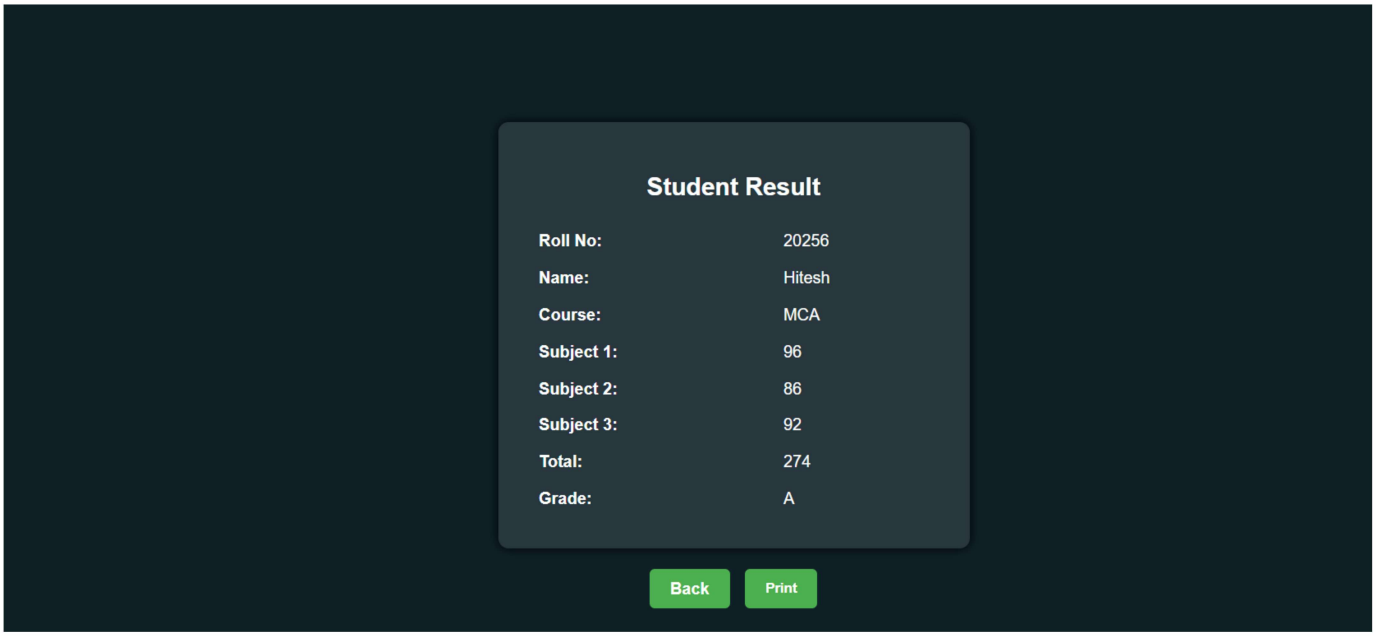
8. Outputs:-

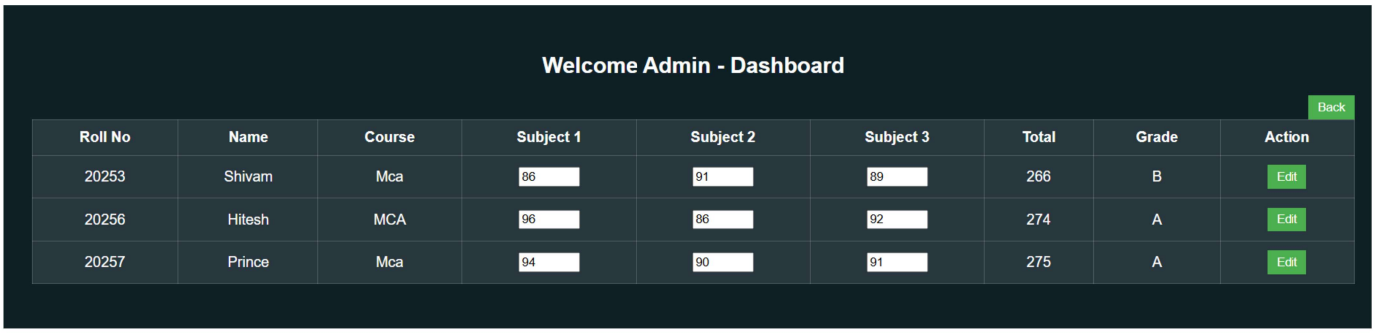












9. Conclusion

The **Student Result Management System** effectively addresses the challenges faced by educational institutions in managing and maintaining student records, marks, and result generation. Through this project, we demonstrated the power of **Java EE technologies (Servlets, JSP, JavaBeans)** combined with **MySQL** to create a dynamic and secure web-based application that streamlines academic data management.

This system allows:

- **Students** to register (when allowed), log in securely, and view their results in a user-friendly format.
- **Administrators** to efficiently manage student data, input subject-wise marks, and control the registration window with ease.

The modular structure of the application—with separate DAO, Model, and Servlet layers—makes the system **scalable**, **maintainable**, and **easy to extend**. The front-end, built using HTML, CSS, and embedded JSP, ensures a clean and intuitive user experience.

By implementing this project, we've achieved our primary objectives of **automating result management**, **reducing manual errors**, and **providing real-time access** to academic records.

This system not only benefits current academic workflows but also lays a solid foundation for future enhancements such as:

- Email notifications of results
- Exporting results as PDF
- Admin role management and analytics dashboards

In summary, the Student Result Management System is a practical, impactful, and scalable solution for managing academic results in modern educational institutions.

10. References:-

- **Head First Servlets and JSP** by Bryan Basham, Kathy Sierra & Bert Bates
  – Helped in designing servlet-based architecture and understanding MVC.
- **Java™: The Complete Reference** by Herbert Schildt
  – Used to understand Java Servlets, JSP, and core Java concepts.
- **MySQL Documentation** (https://dev.mysql.com/doc/)
  – Referenced for setting up database schema and executing SQL queries.
- **Jakarta EE (Jakarta Servlet & JSP Specifications)**
  – Used to structure web components according to standard practices.
- **W3Schools** (https://www.w3schools.com)
  – For learning and referencing HTML, CSS, and form handling.

- **Stack Overflow** (https://stackoverflow.com)
  – Community support for resolving technical errors and implementation issues.

- **GeeksforGeeks** (https://www.geeksforgeeks.org)
  – Referred for JavaBeans usage, Servlet implementation, and example projects.

- **Official Java Tutorials** (https://docs.oracle.com/javase/tutorial/)
  – Used to reinforce Java programming logic and structure.

- **Bootstrap Documentation** (https://getbootstrap.com)
  – Used for styling and making the UI responsive