PROJECT REPORT
ON
“Student Management System”

Submitted By:
Name: Rahul Kumar
UID: 24MCA20233
Section/Group: 2(B)
Subject Code: 24CAP-652

Under The Guidance of:

Ms. Rohini

April 2025

University Institute of Computing

Chandigarh University,

Mohali, Punjab

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have helped me successfully complete this project titled **“Student Management System.”**
First and foremost, I would like to thank **Chandigarh University** for giving me the opportunity to undertake this project as part of the curriculum for the **Advanced Internet Programming (AIP)** subject.
I extend my heartfelt thanks to my project guide, **Ms. Rohini**, whose guidance, support, and constructive feedback played a vital role throughout the development of this project. Her valuable insights helped me improve the overall design and functionality of the system.
I would also like to thank my friends and classmates for their constant support, motivation, and suggestions, which proved to be helpful during various stages of the project.
Lastly, I am thankful to my family for their continuous encouragement and for providing me with the environment and resources to focus on this project.
This project has not only enhanced my technical skills in Java, HTML, CSS, Bootstrap, and MongoDB but also improved my understanding of developing full-stack web applications with proper integration and data management.

Date: 07.04.2025

Place: Chandigarh University, Mohali, Punjab

Rahul Kumar, UID-24MCA20233

ABSTRACT

The **Student Management System** is a web-based application designed t
efficiently manage student data in an educational institution. The primary goal o
the project is to provide a user-friendly interface for administrators to perform key
operations such as **adding, viewing, and deleting student records** with ease.
This system is developed using **Java Servlets** and **JSP (JavaServer Pages)** on th
backend, following the **Model-View-Controller (MVC)** architecture, whic
enhances the separation of concerns and ensures modularity. The application i
styled using **Bootstrap 5**, offering a responsive and intuitive user experience. I
utilizes **MongoDB**, a NoSQL database, for storing student data in a flexible an
scalable manner.
The application begins with an **admin login system**, ensuring that only authorize
users can access the dashboard. Once logged in, the admin can manage the studen
database through options like "Add Student" and "View All Students." The data i
stored and retrieved using **MongoDB Java Driver**, enabling smooth interactio
with the database.
This project demonstrates core web development concepts, including **sessio
management, form validation, database integration**, and the practical use of Java
EE technologies. It provides a solid foundation for understanding how enterprise-
level applications are structured and developed.

Introduction

In today's digital age, managing student data manually can be a time-consuming and error-prone process. Educational institutions require efficient systems to **store, manage, and retrieve** student information quickly and securely. The **Student Management System** is a web-based application developed to address this need by automating student data management processes.
This project is developed using **Java technologies**, including **Servlets and JSP**, and follows the **Model-View-Controller (MVC)** design pattern to ensure a clear separation between the user interface, data processing, and logic. The system is backed by **MongoDB**, a modern NoSQL database that enables fast and flexible data storage without the need for a predefined schema.
The application provides a secure **admin login** feature to prevent unauthorized access. Once authenticated, the administrator can add new student records, view existing records, or delete them as needed. The user interface is designed using **HTML, CSS, Bootstrap**, and **JavaScript**, making it responsive and user-friendly.
This project serves as a practical example of how **enterprise-level web applications** can be built using Java EE technologies. It also emphasizes the importance of using modern tools like **MongoDB** for scalable and efficient data handling.

OBJECTIVES

The main objective of the Student Management System is to provide an efficient and user-friendly platform for managing student data in educational institutions. The specific objectives of the project are as follows:

1. To develop a secure login system
   To allow only authorized users (admin) to access and manage the system through proper authentication.
2. To enable addition, viewing, and deletion of student records
   Provide CRUD (Create, Read, Delete) operations for managing student information easily.
3. To implement a clean and responsive user interface
   Use technology like HTML, CSS, Bootstrap, and JavaScript to build an intuitive and accessible frontend.
4. To use Java EE technologies for robust backend processing
   Employ Servlets, JSP, and Java Beans to handle business logic and server-side operations efficiently.
5. To connect with a modern NoSQL database (MongoDB)
   Store and retrieve student records using MongoDB for flexible and fast data management.
6. To follow the MVC architecture
   Maintain a clear separation of concerns by using the Model-View-Controller pattern.
7. To generate a scalable and maintainable application
   Build a project that can be extended with new features such as student updates, search, filtering, and more.

SYSTEM Requirements

To successfully develop, run, and maintain the *Student Management System*, the following software and hardware configurations are required:

1. Software Requirements

Software Component	**Version/Description**
Operating System	Windows 10/11 or Linux (Ubuntu preferred)
Java Development Kit (JDK)	JDK 17 or above
IDE (Development Tool)	NetBeans IDE 17 or Eclipse
Web Server	Apache Tomcat 10 or above
Database	MongoDB (NoSQL Database)
Frontend Tools	HTML5, CSS3, Bootstrap 5, JavaScript
Browser	Google Chrome, Mozilla Firefox
Build Tool (optional)	Apache Maven (for managing dependencies)

2. Hardware Requirements

Hardware Component	**Minimum Requirement**
Processor	Intel i3 or equivalent
RAM	Minimum 4 GB (8 GB Recommended)
Hard Disk	Minimum 500 MB free space
Monitor	14" or larger (for UI clarity)
Internet Connection	Required for downloading dependencies & external libraries

3. Additional Tools (Optional but Recommended)

- **Postman** – For testing REST APIs (if extended to web services).
- **Visual Studio Code** – For editing frontend files.
- **MongoDB Compass** – GUI to interact with MongoDB collections.\

Technology Used

The Student Management System was developed using a combination of frontend, backend, database, and web technologies to ensure a smooth, responsive, and user-friendly experience. Below are the major technologies used:

1. Frontend Technologies

Technology	**Description**
HTML5	Used for creating the structure of the web pages.
CSS3	Used to style the web pages with responsive design.
Bootstrap 5	Used to design attractive and responsive user interfaces.
JavaScript	Used for client-side interactivity and validation.

2. Backend Technologies

Technology	Description
Java (Servlets & JSP)	Core backend logic and dynamic content generation.
JSP (JavaServer Pages)	Embeds Java directly into HTML for dynamic page rendering.
Jakarta Servlet API	Manages HTTP requests and responses for web interactions.

3. Database

Technology	Description
MongoDB	NoSQL database used to store and manage student records.
MongoDB Java Driver	Java library to connect and perform database operations.

4. Server & Tools

Tool	Description
Apache Tomcat 10+	Web server used to run Java-based web applications.
NetBeans IDE	Integrated Development Environment used for coding, compiling, and debugging the project.

5. Optional Tools

Tool	Description
MongoDB Compass	Visual tool for managing MongoDB collections.
Git	Version control (if collaboration or backups are needed).

System Design

System Design is the blueprint for developing a software application. It defines the architecture, components, modules, interfaces, and data flow for the complete system. For the *Student Management System*, the design focuses on user interaction, data flow, and system processing.

1. Architecture Diagram

```
+---------------------+
|    Web Browser     |
| (User Interface)  |
+----------+----------+
           |
          v
+----------+----------+
|   JSP Pages (UI)    |
|  login.jsp, add.jsp |
+----------+----------+
          |
          v
+----------+----------+
|   Servlets (Control)|
| LoginServlet,       |
| StudentServlet,     |
| LogoutServlet       |
+----------+----------+
          |
          v
+----------+----------+
|   DAO (Model)       |
| StudentDAO.java     |
+----------+----------+
          |
          v
+---------------------+
|    MongoDB Database |
|   (Student Records) |
+---------------------+
```

2. Modules of the System

1. **Login Module**
   - Authenticates admin using a predefined username and password.
2. **Dashboard Module**
   - Provides navigation to different operations like adding, viewing, and managing student records.
3. **Add Student Module**
   - Accepts student details and stores them in the MongoDB database.
4. **View Student Module**
   - Retrieves and displays all stored student records from the database.
5. **Logout Module**
   - Ends the session and redirects the user to the login page.

3. Data Flow Diagram (DFD - Level 0)

```
+---------------+
|     Admin     |
+-------+-------+
        |
        v
+-------+-------+
|  Web Server   |
| (Servlet/JSP) |
+-------+-------+
        |
        v
+-------+-------+
|   MongoDB     |
|  (Database)   |
+---------------+
```

4. Interaction Flow

1. User opens the system and logs in.
2. Upon successful login, the user is redirected to the dashboard.
3. User can add new students or view existing ones.
4. All data operations are handled by servlets and persisted in MongoDB.
5. Logout ends the session and returns the user to the login page.

Implementation

Implementation is the process of converting the system design into working code and integrating various modules to function together. In this project, we implemented a web-based Student Management System using Java (JSP/Servlets), MongoDB, and Bootstrap for the user interface.

1. Project Setup

IDE Used: NetBeans
Server: Apache Tomcat
Backend Language: Java (Servlets & JSP)
Database: MongoDB (NoSQL)
Frontend: HTML, CSS, Bootstrap
Tools: MongoDB Compass for visualizing data

2. Key Files and Their Roles

File / Component	Description
login.jsp	Login form that authenticates users

File / Component	Description
LoginServlet.java	Handles login requests, validates credentials, starts a session
dashboard.jsp	Admin panel for managing student data
addStudent.jsp	Form to collect student details
StudentServlet.java	Controls student-related actions (add, view)
StudentDAO.java	Interacts with MongoDB to add, retrieve, and delete student records
Student.java (Bean)	A JavaBean class for representing student data
LogoutServlet.java	Ends session and redirects to login
StudentAPI.java	Provides RESTful access to student data using JAX-RS

3. Functional Workflow

- **User Login**
  Admin logs in using a valid username and password.
  LoginServlet verifies credentials and starts a session.
- **Dashboard Access**
  On successful login, the user is redirected to dashboard.jsp.
  The dashboard provides buttons for adding or viewing student data.
- **Adding a Student**
  The user fills out the form in addStudent.jsp.
  The form submits to StudentServlet with action=add.
  StudentDAO saves the data to the MongoDB database.
- **Viewing Students**
  The dashboard link sends a request to StudentServlet?action=view.
  StudentDAO fetches all student records.
  Data is displayed in a styled HTML table.
- **Logout**
  Ends the user session and returns to the login page.

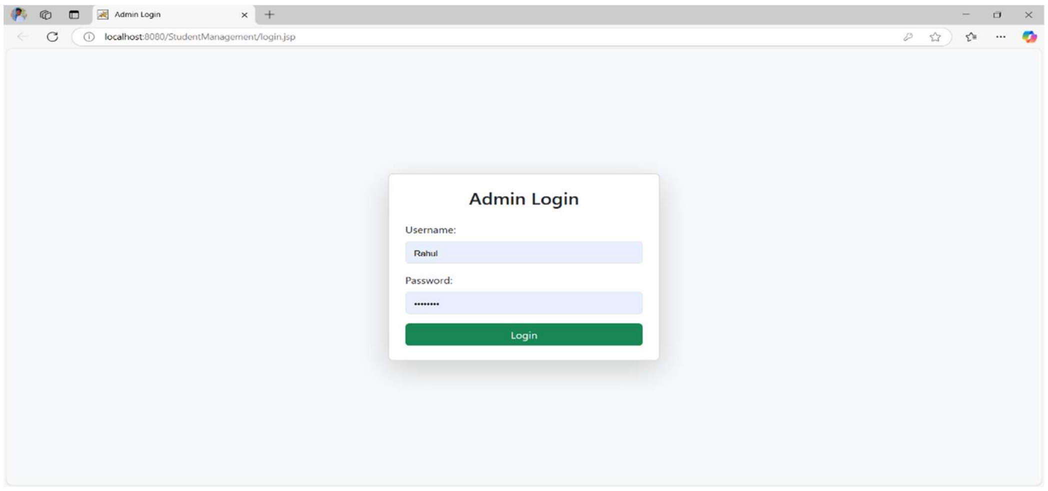
4. MongoDB Integration

- Connected using MongoDB Java Driver
  Operations like insertOne(), find(), and deleteOne() are used for data manipulation

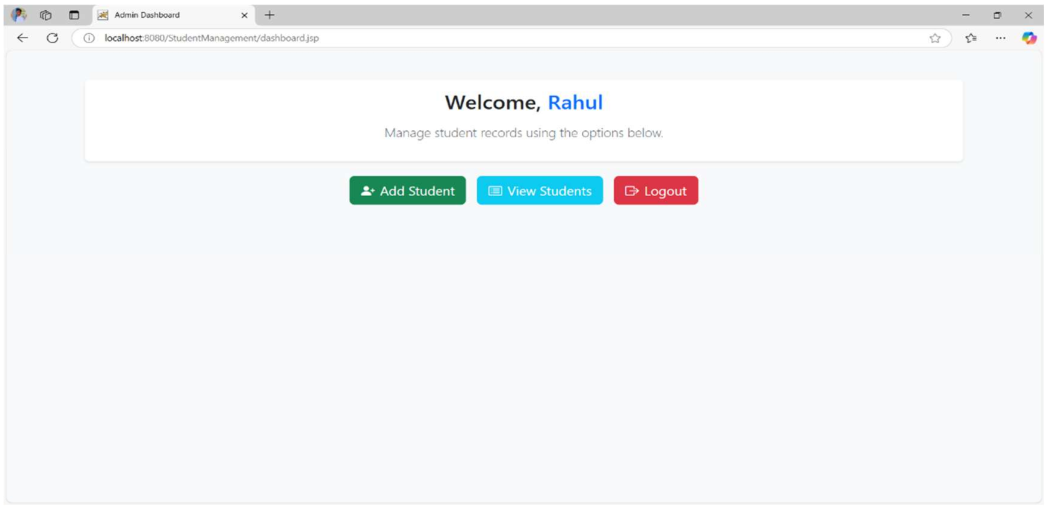
Data stored in JSON-like format in the students collection

<u>RESULTS & SCREENSHOTS</u>

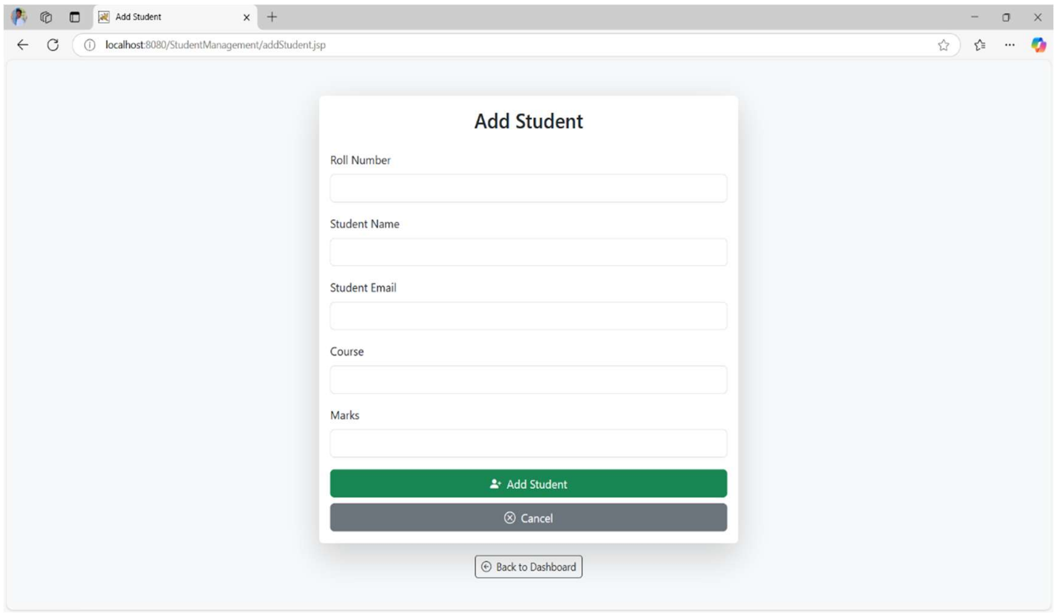
Login page:



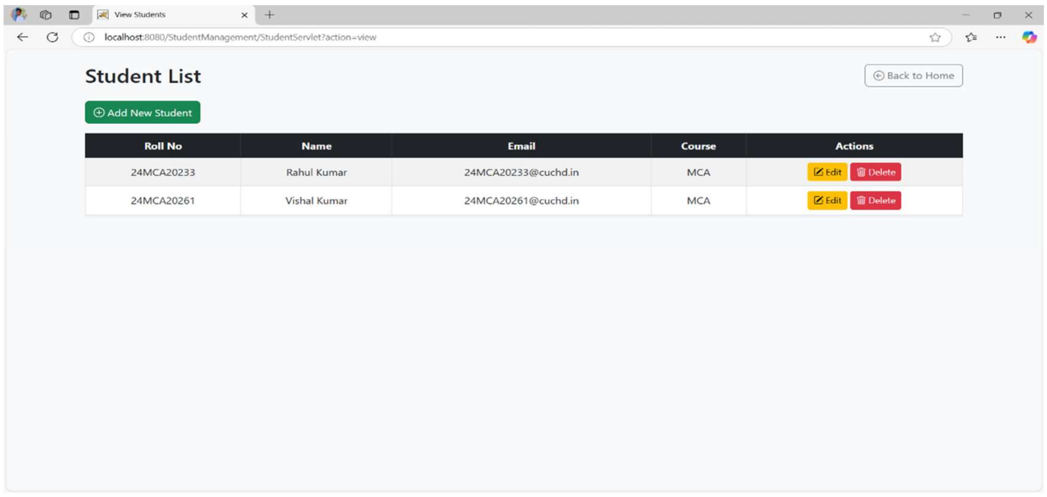
Dashboard:


Admin Dashboard
localhost:8080/StudentManagement/dashboard.jsp
Welcome, Rahul
Manage student records using the options below.
Add Student
View Students
Logout

Add Student Form



View Students:



Student List

Roll No	Name	Email	Course	Actions
24MCA20233	Rahul Kumar	24MCA20233@cuchd.in	MCA	Edit Delete
24MCA20261	Vishal Kumar	24MCA20261@cuchd.in	MCA	Edit Delete

Testing

Testing is a crucial phase in the software development life cycle to ensure that the application is working correctly, securely, and meets the requirements. The **Student Management System** was tested using **manual testing** techniques across different modules and functionalities.

1. Types of Testing Performed

Testing Type	**Description**
Unit Testing	Each module (like login, add student, view student) was tested individually.
Integration Testing	Verified interactions between servlets, JSP pages, and MongoDB.
Functional Testing	Checked if the application behaves as expected with valid/invalid input.
System Testing	End-to-end testing of the complete flow from login to logout.
Usability Testing	Tested for user-friendly UI and smooth navigation.

2. Test Cases and Results

Test Case	**Input**	**Expected Output**	**Status**
Login with valid credentials	Username: Rahul, Password: Rahul123	Redirect to dashboard.jsp	Passed
Login with invalid credentials	Wrong username or password	Show "Invalid Credentials" error message	Passed
Add Student	Valid name, email, course	Student added successfully	Passed
View Students	Click "View All Students"	Display list of students in table format	Passed
Logout	Click logout button	Redirect to login page	Passed
Direct URL access to dashboard	Open dashboard without login	Redirect to login.jsp	Passed

3. Bug Handling

- *Issue:* Login did not redirect properly when credentials were wrong
  Fix: Forwarded back to login page with error message.
- *Issue:* Student form accepted empty fields
  Fix: Added HTML required fields and backend validation.

CONCLUSION

The **Student Management System** project was developed with the objective of simplifyin and streamlining the process of managing student records in an educational institution. Th system provides functionalities such as user login, adding new students, viewing all students and secure logout—all through a user-friendly web interface.

During the development process, we applied various web technologies such as **HTML, CSS Bootstrap, JavaScript, Java Servlets, JSP, and MongoDB** to ensure a dynamic an responsive user experience. We also focused on implementing secure session management an validating user input to enhance security and performance.

The project has successfully met its objectives:

- It is easy to use and manage.
- Reduces paperwork and manual errors.
- Allows real-time access to student data.
- Provides a reliable and scalable platform for student record management.

Through this project, we gained valuable experience in **web development using Java E technologies**, working with **MongoDB**, handling **server-side logic with servlets**, an improving **problem-solving and project management skills**.

In conclusion, this system can serve as a foundational application that can be further extende with features like student search, update, delete, and advanced analytics.

FUTURE SCOPE

The **Student Management System** developed as part of this project serves as a foundationa tool for managing basic student information. However, there are several areas where this syste can be improved and expanded in the future:

1. **Student Profile Enhancements**
   - Addition of more detailed student information like attendance records, grades guardian details, and fee structure.
   - Uploading student documents and profile pictures.
2. **Role-Based Access**
   - Integration of different user roles like Admin, Faculty, and Student with specifi access permissions and dashboards.
3. **Search and Filter Functionality**
   - Advanced search, sort, and filter options to find student data quickly.
4. **Integration with SMS/Email APIs**
   - Sending automated notifications regarding results, attendance, and announcements t students and parents.
5. **Mobile App Development**

    - Building a mobile-friendly version or Android/iOS app for accessing the system on the go.

6. **Analytics and Reports**
    - Generating detailed reports and graphical analysis for student performance and demographics.
7. **Cloud Deployment**
    - Hosting the system on cloud platforms (like AWS, GCP, or Azure) for better scalability and global access.
8. **Security Improvements**
    - Implementing encryption, two-factor authentication, and better session management for enhanced security.

GitHub Link https://github.com/rahulkumarpass/Student_Management

Reference

- **Websites: w3schools.com, javatpoint.com, MongoDB Docs, Bootstrap Docs, StackOverflow**