**Visva Bharati University**

Department of Computer and System Sciences

B.Sc. Semester VI Examination, 2025

Subject: *Computer Science*

Project Report

On

**Student Result Management System**

**Roll No:** B.Sc (Sem-VI)Comp -

**Registration No:** VB- \_\_\_\_ of 2022-23

**CERTIFICATE**

This is to certify that this report on “**Student Result Management System**” embodies the original work done by \_\_\_\_\_\_\_\_\_\_ .

During this project submission as a partial fulfillment of the requirement for the **PROJECT WORK** of bachelor’s degree in Computer Science under **Visva Bharati University**.

……………………….. ……………………….

In charge of the Project Project Guide

(Prof. Subhasis Banerjee) (Prof. Subhasis Banerjee)

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.NO. | TOPIC | PAGE NO. |
| 1 | Introduction | 3 |
| 2 | Purpose of the Project | 3 |
| 3 | |  | | --- | |  |  |  | | --- | | Problems in the Existing System | | 4 |
| 4 | Problems and Solutions | 4 |
| 5 | Feasibility Report | 5 - 6 |
| 6 | System Requirements | 7 - 8 |
| 7 | |  | | --- | |  |  |  | | --- | | System Design | | 9 - 10 |
| 8 | |  | | --- | |  |  |  | | --- | | Data Flow Diagram (DFD) | | 11 - 13 |
| 9 | Database Design | 14 - 16 |
| 10 | |  | | --- | |  |  |  | | --- | | Screenshots of the Application | | 17 - 21 |
| 11 | |  | | --- | |  |  |  | | --- | | Implementation Details | | 22 - 24 |
| 12 | |  | | --- | |  |  |  | | --- | | Testing and Validation | | 25 - 27 |
| 13 | |  | | --- | |  |  |  | | --- | | Conclusion | | 28 |
| 14 | Bibliography | 29 |

**INTRODUCTION**

The **Student Result Management System** is a web-based application that helps schools and colleges manage student marks and results in an easy and organized way. This system allows students to register and log in to view their results online. It also helps the admin or teacher to enter, update, and manage student marks quickly without using paper. The project uses modern web technologies like Node.js, Express, MongoDB, and EJS, and also supports Python (Flask) for extra features. This system is designed to make result management faster, error-free, and more convenient for everyone.

**Purpose of the Project**

The main purpose of this project is to make the result management process **simple**, **quick**, and **digital**. Instead of writing results by hand or using Excel sheets, the system helps teachers enter marks easily and allows students to check their results from anywhere. It saves time, avoids mistakes, and keeps all the records safe in one place. This system also helps in keeping the result process fair and transparent. Overall, it is designed to improve the way schools and colleges handle student results.

**PROBLEMS AND SOLUTIONS OF THE PROJECT**

**Problems in the Existing System**

1. **Manual Work** – Teachers often have to enter marks by hand or use Excel sheets, which is slow and tiring.
2. **Errors and Mistakes** – Manual result processing can lead to calculation mistakes or typing errors.
3. **No Easy Access** – Students must wait for printed report cards or ask teachers for their results.
4. **Poor Record Keeping** – It’s hard to store, organize, and search old student results.
5. **Lack of Privacy** – Anyone can see paper records or shared files if not handled carefully.
6. **Time Wasted** – Managing, finding, and correcting results takes a lot of time.

**Solutions Provided by the Project**

1. **Digital System** – Teachers can quickly enter and save student marks using a simple web form.
2. **Error Reduction** – Calculations like total marks and percentage are done automatically by the system.
3. **Online Access** – Students can log in anytime to view their results securely from anywhere.
4. **Safe Storage** – All data is saved in a database (MongoDB), which keeps records safe and organized.
5. **Privacy and Security** – Only authorized users (students/admins) can access their information using login credentials.
6. **Time Saving** – The system saves time for both teachers and students by simplifying the entire process.

**FEASIBILITY REPORT**

Preliminary investigation examines project feasibility, the livelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they have unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

**Technical feasibility**

**Operation feasibility**

**Economic feasibility**

**1. Technical Feasibility**

This project is **technically feasible** because it uses easily available technologies like:

* **Node.js**, **Express.js**, and **MongoDB** for backend and database
* **EJS**, **HTML**, **CSS**, and **JavaScript** for frontend
* **Flask + Python** for admin tools or analytics  
  All these tools are open-source, well-documented, and easy to set up on normal computers or servers.

So, the system can be developed and run using the available hardware and software.

**2. Economic Feasibility**

The system is **economically feasible** because:

* It uses free technologies (no expensive licenses)
* Reduces printing and manual work costs
* Saves time for both teachers and students  
  It requires only basic development and hosting costs (if put online), which are minimal compared to the benefits.

The cost of building and maintaining the system is affordable.

**3. Operational Feasibility**

This project is **operationally feasible** because:

* It is user-friendly for both students and admins
* Helps students check results instantly
* Reduces stress on teachers and reduces human errors
* Improves transparency and record management

It will be accepted by users and work smoothly in a real-world school or college environment.

**4. Schedule Feasibility**

This system can be completed within a short time (2–4 weeks), depending on the team and features included.  
The project can be developed module by module — login, registration, result upload, result view — which allows smooth planning.

The system is possible to complete in the available time frame.

**System Requirement Specification (SRS)**

**1. Purpose of the Project**

The main purpose of this project is to develop a web-based system that allows educational institutions to manage student results efficiently and accurately. The system provides features like student registration, login, result entry by admin, and result viewing by students. It helps reduce manual errors, saves time, and provides secure access to academic data.

**2. Features of the Project**

The key features of the Student Result Management System are:

* **Student Registration:** Students can sign up by entering personal and academic details.
* **Login System:** Separate login for students and admin (teacher/staff).
* **Result Entry:** Admin can add, update, and delete student results.
* **View Result:** Students can log in and check their results securely.
* **Dashboard:** Admin can view all student results and analyse performance.
* **Secure Access:** Passwords are stored securely with encryption.
* **Database Integration:** Uses MongoDB for safe and organized data storage.
* **Flask Integration:** Allows admin-side AI/ML features or reports using Python.

3. **Hardware Requirements**

* A basic computer or laptop (Intel Core i3 or above)
* Minimum 4 GB RAM
* 200 MB of free disk space
* Screen resolution: 1280 x 720 or higher
* Internet connection (for remote MongoDB or Flask integration)

4. **Software Requirements**

* **Operating System**: Windows 10 or later / Linux / macOS
* **Backend Runtime**: Node.js (version 16 or above)
* **Database**: MongoDB (local or MongoDB Atlas cloud)
* **Frontend**: HTML, CSS, JavaScript, EJS (Embedded JavaScript Templates)
* **Python + Flask**: for admin features or ML models
* **Web Browser**: Google Chrome, Firefox, or Edge
* **Development Tools**:
  + Visual Studio Code (or any code editor)
  + npm packages: Express, Mongoose, Bcrypt, dotenv, EJS, nodemon, etc.

**System Design**

System Design is the process of planning how all parts of the software system will work together to meet the goals of the project. It includes the structure of the system, how data will flow between components, and how users will interact with it. The system design of the Student Result Management System is divided into two parts: **High-Level Design (HLD)** and **Low-Level Design (LLD)**.

**1. High-Level Design (HLD)**

This design focuses on the overall structure of the system. It shows how different modules of the system are connected and how they work together.

**Main Modules:**

* **User Module**: Handles student and admin registration, login, and authentication.
* **Result Module**: Allows admins to add, edit, delete, and view student results.
* **Dashboard Module**: Shows summary or individual result information.
* **Database Module**: Stores student details, login credentials, and results.
* **Flask Integration** : Used for admin-only tasks like data analytics or report generation.

**Workflow Overview:**

1. Student/Admin registers → credentials saved in database
2. User logs in using secure password
3. Admin can enter or update student results
4. Students can view their results
5. System calculates total, percentage, and grade automatically
6. All data is stored and retrieved from MongoDB

**2. Low-Level Design (LLD)**

This design focuses on the detailed working of each component, including internal logic, functions, and data handling.

**Backend Routes:**

* POST /register – Save new user data
* POST /login – Verify user and start session
* GET /dashboard – Display result based on logged-in student ID
* POST /add-result – Admin adds result
* GET /logout – Ends session

**Database Collections:**

* **users**  
  Stores student and admin login information  
  Fields: studentId, name, emailId, department, password
* **results**  
  Stores student marks and result details  
  Fields: studentId, subjects, totalMarks, percentage, grade

**Frontend Pages (EJS Views):**

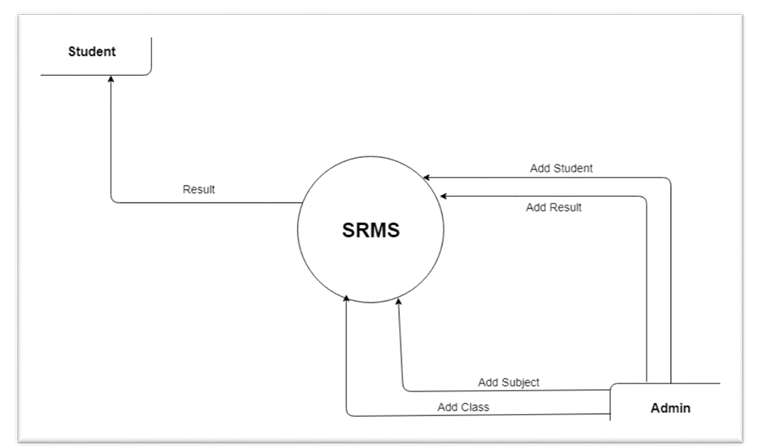
* home.ejs – Home page with admin/student options
* login.ejs – User login form
* register.ejs – Student registration form
* index.ejs – Student dashboard
* myResult.ejs – Result display page for students
* addResult.ejs – Admin page for entering results

**🔐 Technologies Used:**

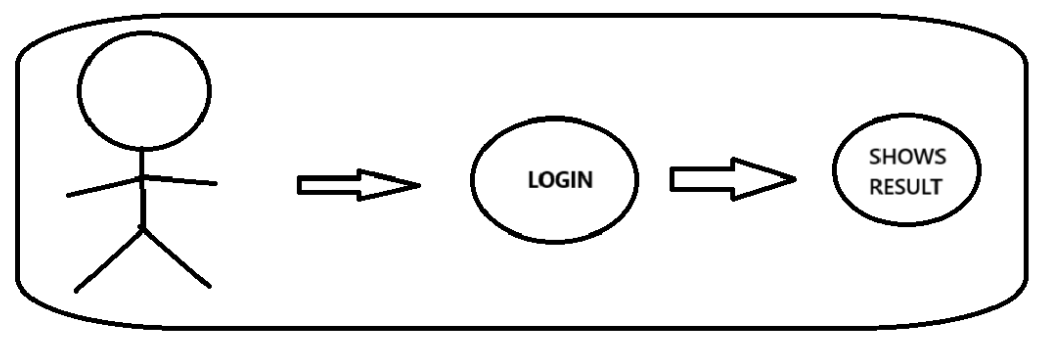
* **Frontend:** HTML, CSS, JavaScript, EJS
* **Backend:** Node.js, Express.js
* **Database:** MongoDB (using Mongoose)
* **Password Encryption:** bcrypt
* **Admin Tools:** Python Flask (for advanced analytics)

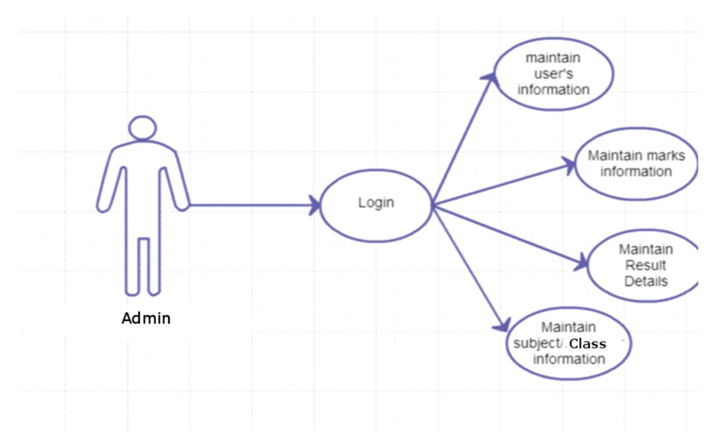
**DATA FLOW DIAGRAMS (DFDs)**

**FIGURE**: *DIAGRAM SHOWING HOW THE ADMIN UPDATES THE DATA AND HOW THE STUDENT GETS THE RESULT.*



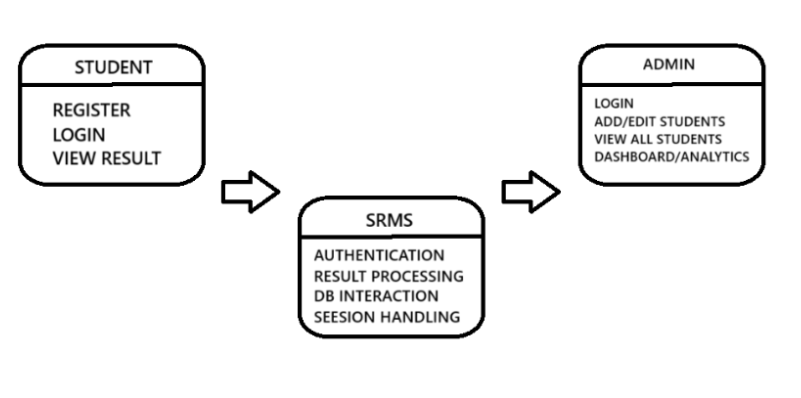
STUDENT USE CASE DIAGRAM:

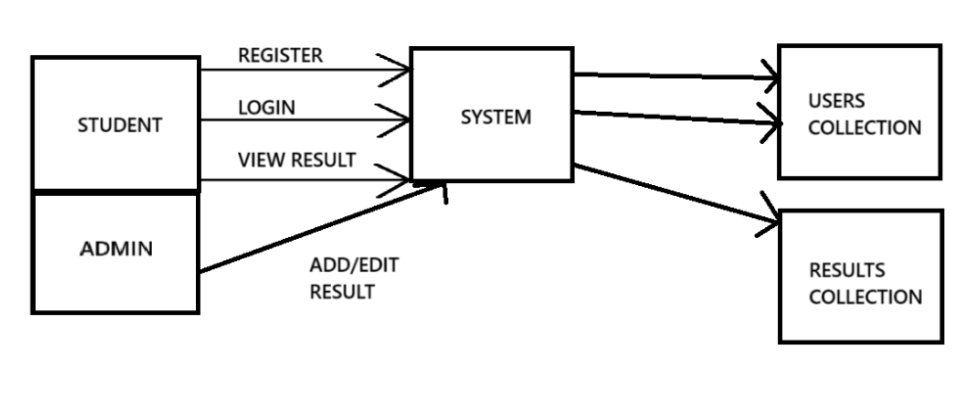


*ADMIN USE CASE DIAGRAM:* 

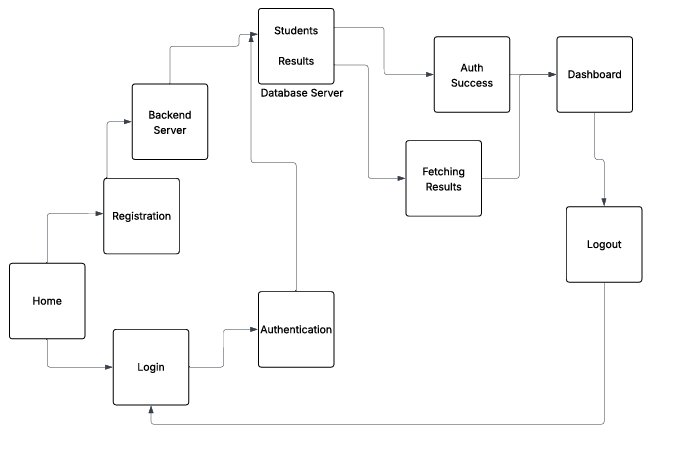
**WHOLE SYSTEM DFD:**

**LEVEL – 1:**

****

****

**LEVEL-2:**

****

**Database Design**

**Database Used :**

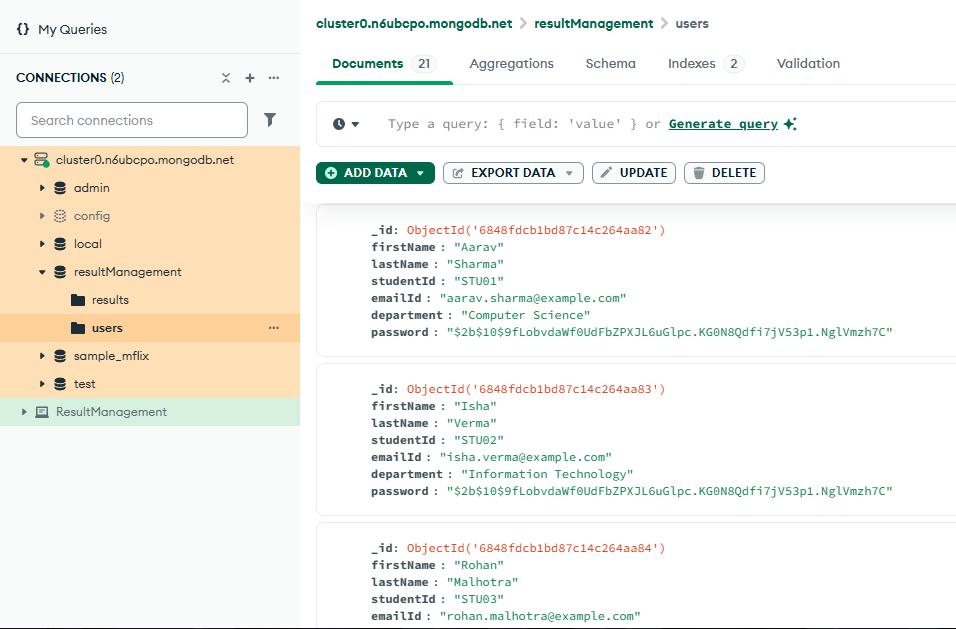
We used MongoDB database for our result management system to store the student’s data and student’s results .

Inside our **resultManagement** database we have two collections users and results.

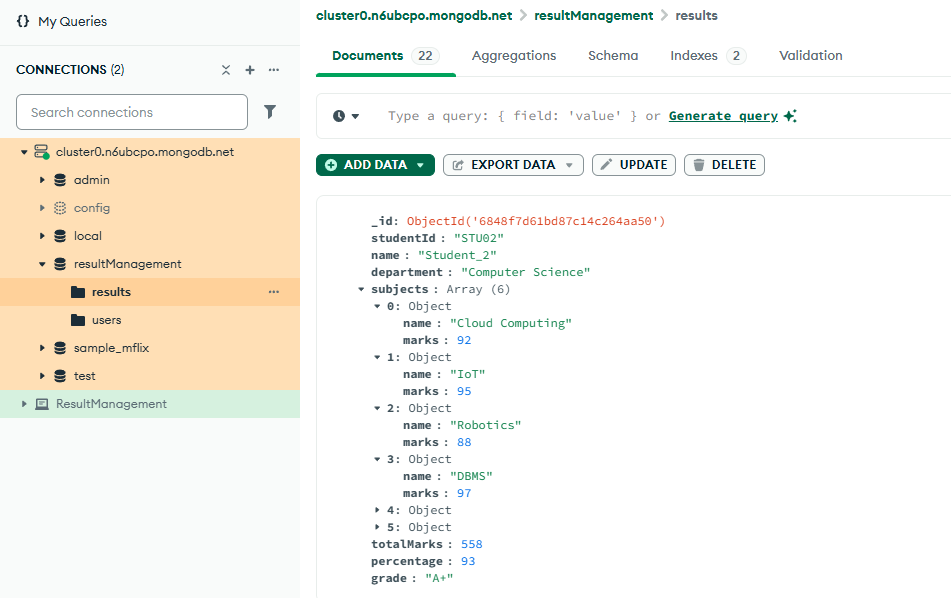
**Users Collection:**

Inside Users Collection we have

* **Firstname** – to store the first name of the student
* **Lastname** - to store the last name of the student
* **studentID** – to store the unique ID of the student and it is the our primary attribute to check for authentication and it links to results collection .
* **emailID** – to store the email id of the student
* **department** – to store the department name of the student.
* **Password** – to store the student’s unique password and used for log into website.

****

Screenshot of the users collection

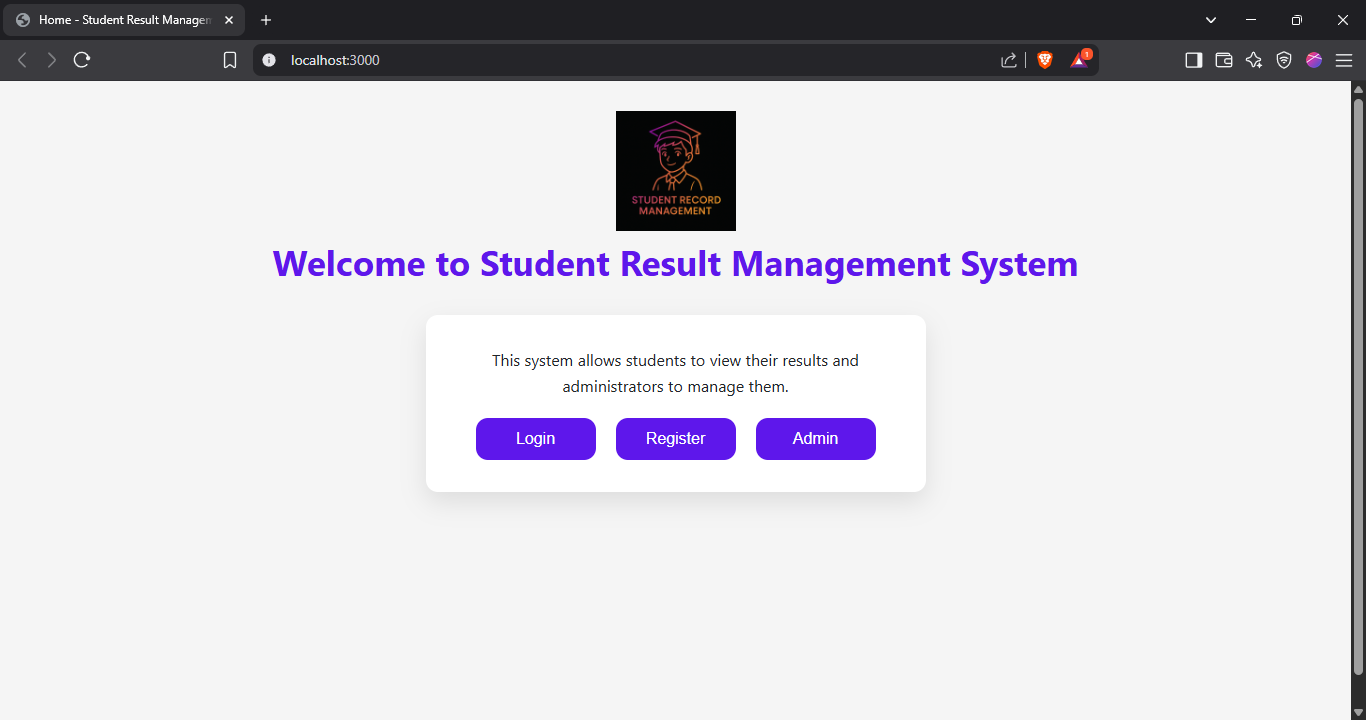
****

Screenshot of the results collection

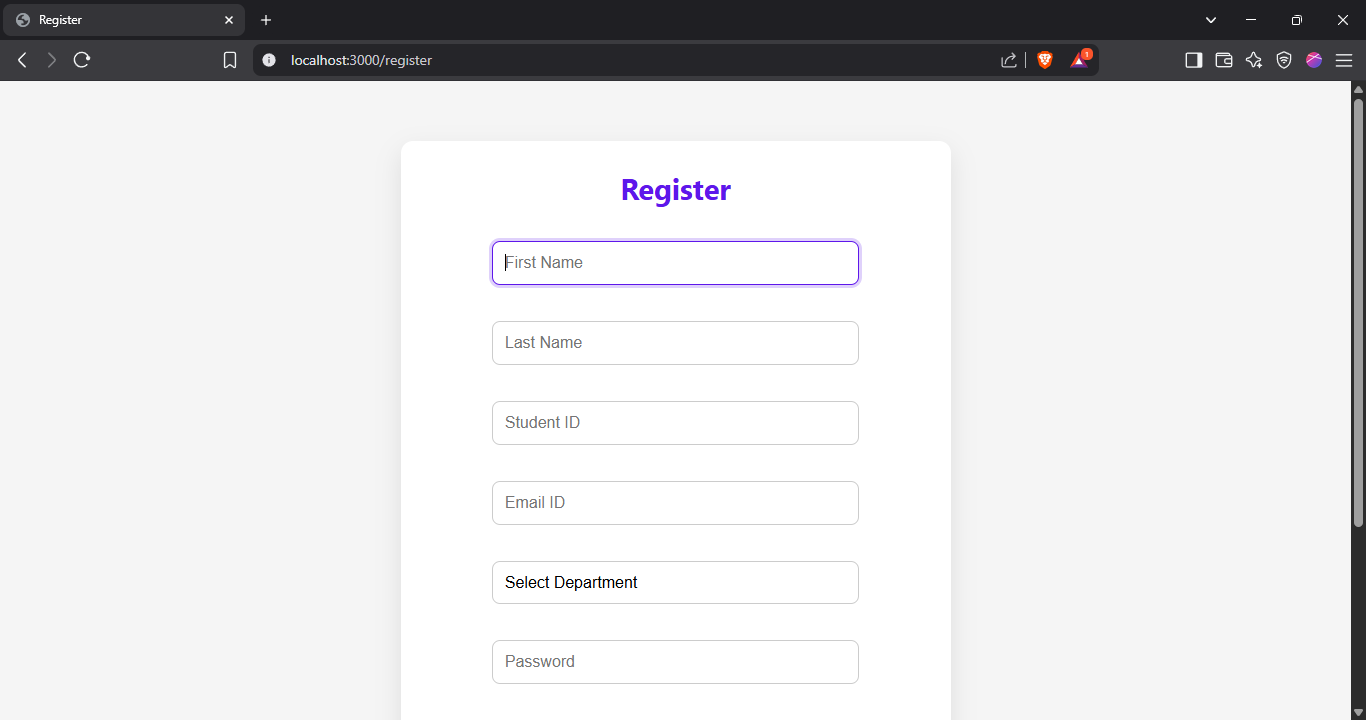
**Results Collection:** Inside results collection we have

* **studentID**: this links to the users collection
* **name**: to store the students name
* **department**: to store the department name
* **subjects**: It is an array of objects which stores the name of the subject and marks.
* **Totalmarks**: this calculates and stores the total marks of the student.
* **Percentage**: this calculates and stores the total percentage of the student.
* **Grade**: this stores the final grade of the student

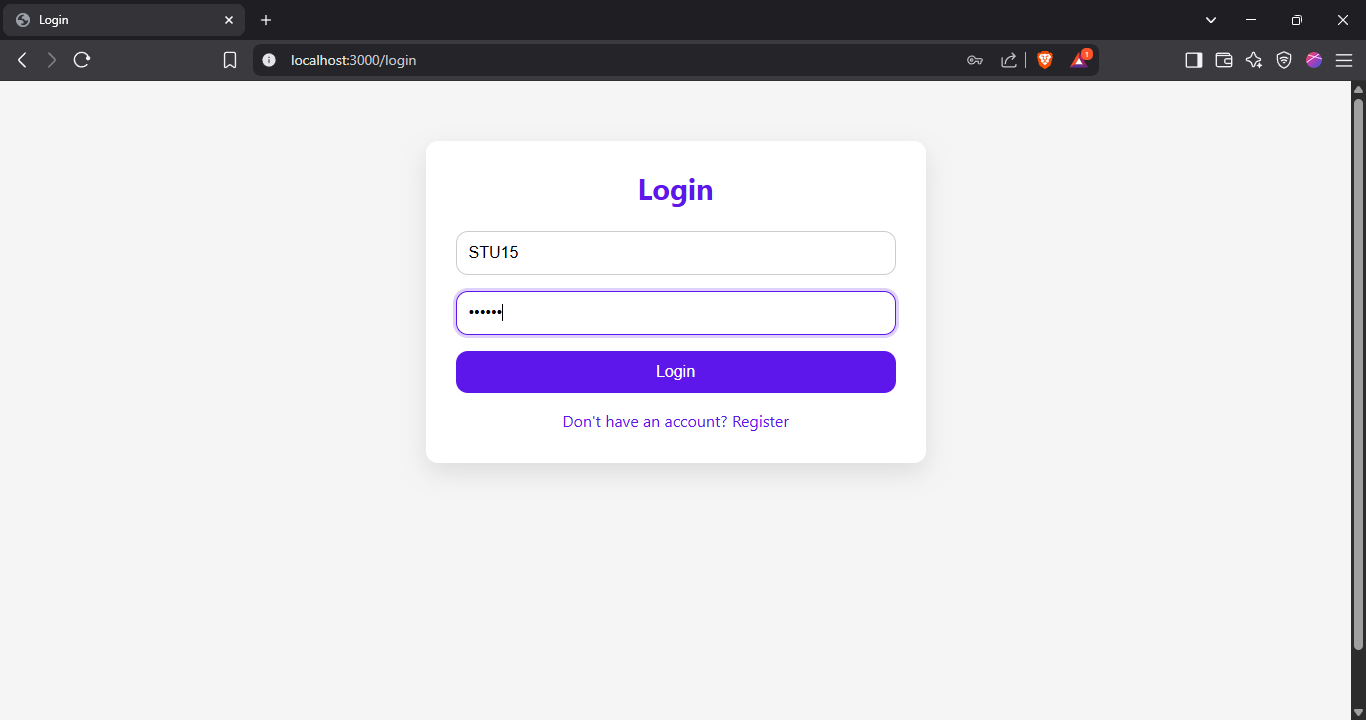
**INTERFACE OF THE WEBSITE**



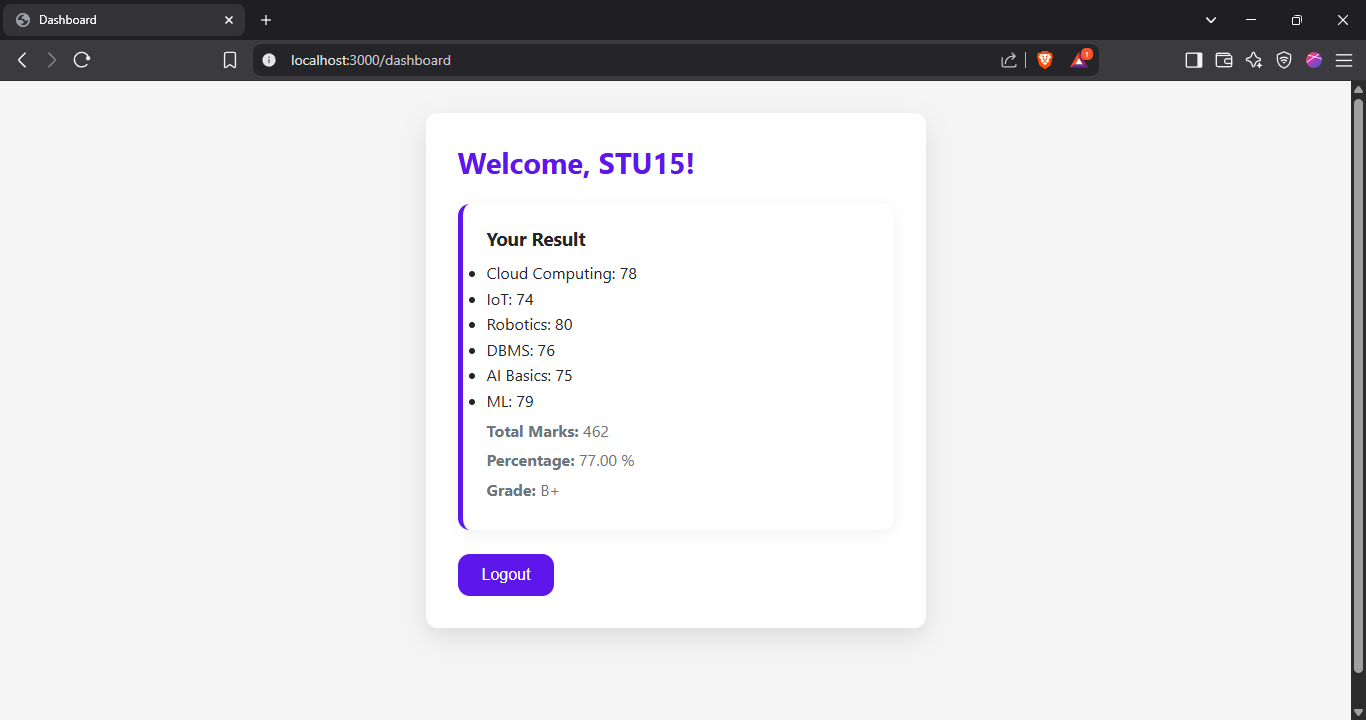
Screenshot of Home page



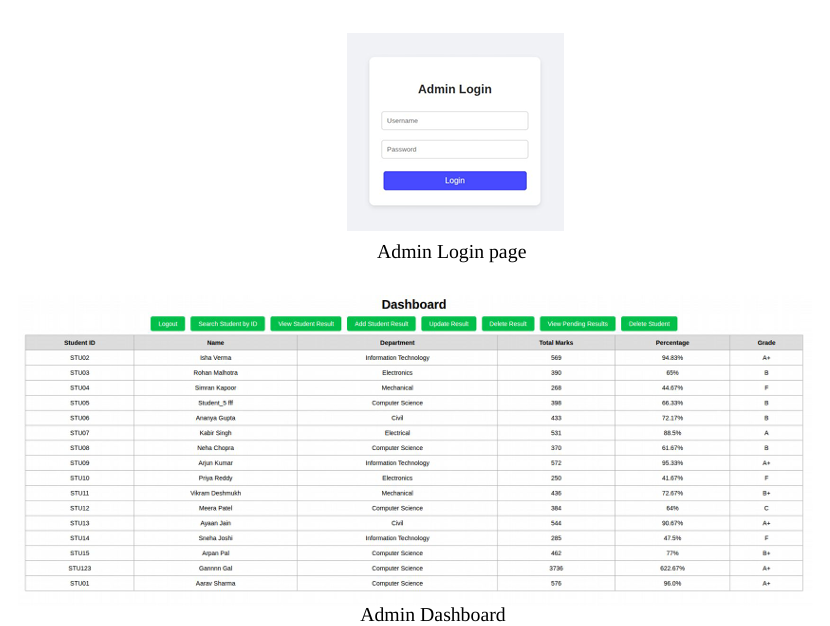
Screenshot of Register Page

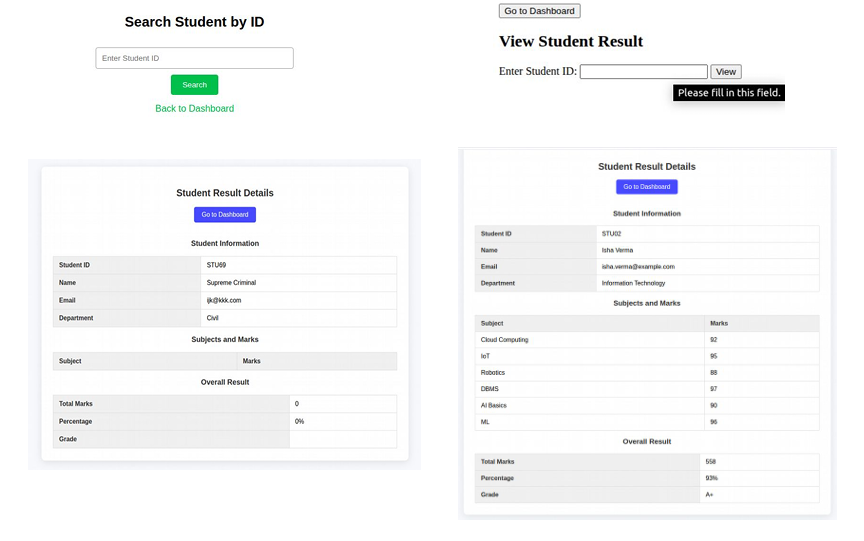


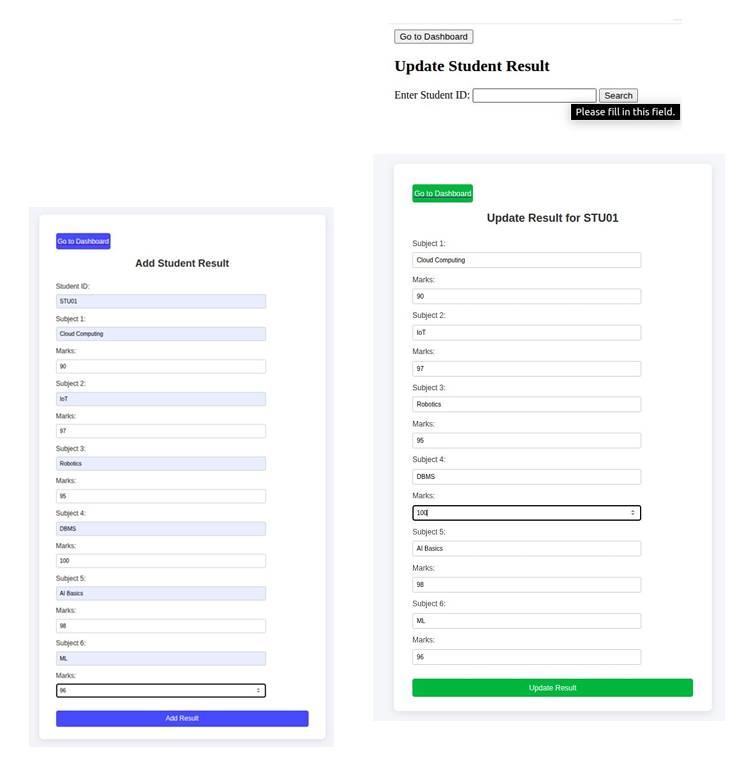
Screenshot of Login Page

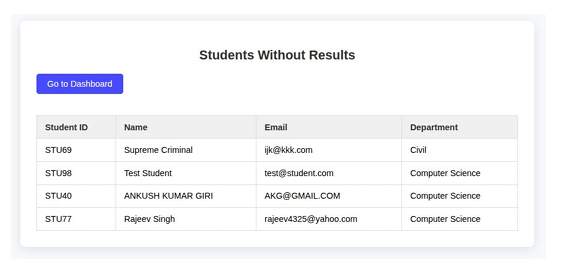


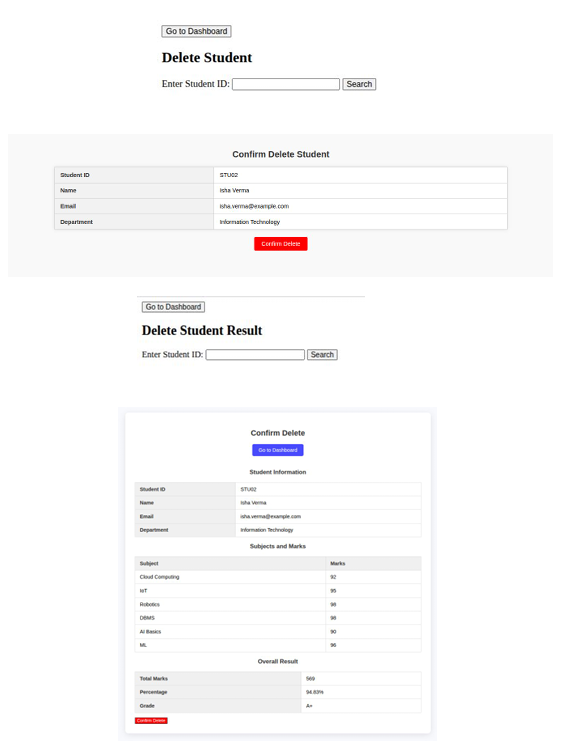
Screenshot of Student Result Page











**Implementation Details**

**Technologies Used**

**1. Frontend:**

* HTML5 – For building the structure of web pages.
* CSS3 & Bootstrap – For designing responsive and attractive layouts.
* JavaScript – For client-side form validation and user interaction.
* EJS (Embedded JavaScript) – Templating engine used to dynamically generate HTML content from the server.

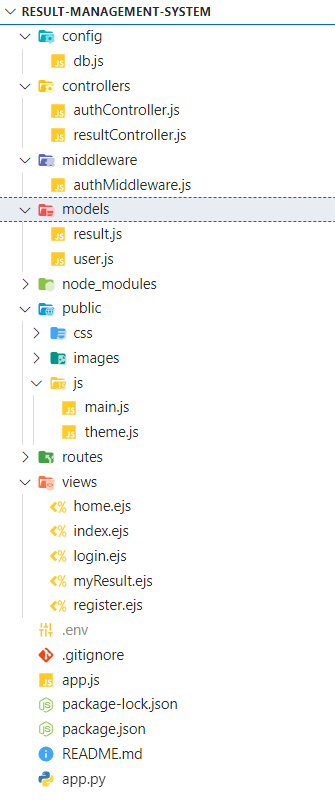
**2. Backend:**

* Node.js – JavaScript runtime environment to build the server.
* Express.js – Web application framework for routing and middleware handling.
* MongoDB – NoSQL database used to store user and result data.
* Mongoose – ODM (Object Data Modeling) library to interact with MongoDB using models and schemas.
* bcrypt – For securely hashing and verifying passwords.
* dotenv – To load environment variables from a .env file (e.g., DB credentials).
* express-session – To manage user login sessions.

**3. Integration:**

* Flask (Python) – Used for advanced admin-side analytics, reporting, or ML integration.

**FOLDER STRUCTURE:**



**How the System Works (Workflow):**

1. **User Registration/Login:**
   * Students register using their name, ID, department, email, and password.
   * Passwords are encrypted before storing in the database.
   * On login, sessions are created and managed using express-session.
2. **Admin Functionalities:**
   * Admin logs in and can add, edit, or delete student results.
   * Each result includes subject-wise marks, total, percentage, and grade.
3. **Student Functionalities:**
   * After login, students can view their result dashboard.
   * Results are fetched using the student’s studentId.
4. **Database Operations:**
   * All user and result data is stored and retrieved from MongoDB using Mongoose models.
   * The system is designed to be scalable and easy to maintain.

**Testing and Validation**

Testing is a crucial part of software development to ensure that the application works correctly, is free of bugs, and meets user expectations. In the *Student Result Management System*, testing was done at different stages to verify that all functionalities work as intended and data is handled securely.

**Types of Testing Performed**

**1. Unit Testing**

* Each part of the system (like user registration, login) was tested individually.
* Example: Testing if percentage and grade are calculated correctly based on entered marks.

**2. Form Validation Testing**

* Input fields were tested to ensure that empty, invalid, or incorrect values are not accepted.
* Example:
  + Registration fails if any required field is empty.
  + Password must be at least 8 characters.
  + Email format must be correct.
  + Password and Confirm Password must match.

**3. Login Validation Testing**

* Checked login with correct and incorrect credentials.
* Confirmed:
  + User is redirected to dashboard only if credentials are valid.
  + Incorrect login shows proper error message.
  + Passwords are securely encrypted and verified using bcrypt.

**4. Session Testing**

* Tested user sessions using express-session:
  + Users can only view their dashboard when logged in.
  + Logging out destroys the session.
  + Unauthorized access to /dashboard without login is blocked.

**5. Database Testing**

* MongoDB operations like insertion, update, and retrieval were tested using Compass and Mongoose.
* Tested that:
  + User details and results are saved correctly.
  + Duplicate Student ID is not allowed.
  + Results display based on student ID.

**6. Integration Testing**

* Verified complete flow from registration → login → result view.
* Admin actions like result entry and editing were tested for consistency and correctness.

**7. Browser Compatibility**

* Tested the system on multiple browsers (Chrome, Firefox, Edge) to ensure layout and functionality work the same.

**8. User Feedback Testing**

* Conducted informal testing by giving the system to a few users (students and faculty) to check for usability and clarity.
* Feedback was used to improve layout, form messages, and button placements.

**Conclusion:**

All major functionalities were successfully tested and validated. The system is reliable, user-friendly, and secure for managing and viewing student results. Minor improvements were made based on feedback to enhance user experience.

**Conclusion**

The *Student Result Management System* successfully achieves its goal of providing a digital platform for managing student academic results in an efficient, secure, and user-friendly way. The system reduces the workload of administrators by automating tasks like result entry, calculation of total marks, percentages, and grades. It also allows students to access their results easily through a secure login system.

By using modern technologies such as Node.js, Express, MongoDB, and EJS, the project ensures smooth performance and scalability. Additionally, the use of encrypted passwords and session management improves data privacy and security. Another integration with Python Flask allows future expansion into advanced features like data analytics or machine learning.

Overall, the system offers a simple, practical, and paperless approach to result management that can be adopted by educational institutions of any size.

**Bibliography**

Below are the tools, resources, and websites referred to during the development of this project:

1. **Node.js Official Documentation**  
   https://nodejs.org/en/docs/
2. **Express.js Documentation**  
   <https://expressjs.com/>
3. **MongoDB Documentation**  
   <https://www.mongodb.com/docs/>
4. **Mongoose (MongoDB ODM for Node.js)**  
   <https://mongoosejs.com/>
5. **Bootstrap (for frontend design)**  
   <https://getbootstrap.com/>
6. **EJS Templating Documentation**  
   <https://ejs.co/>
7. **bcrypt for Node.js Password Hashing**  
   <https://www.npmjs.com/package/bcrypt>
8. **Python Flask (admin tool)**  
   https://flask.palletsprojects.com/
9. **W3Schools (HTML, CSS, JavaScript references)**  
   <https://www.w3schools.com/>
10. **GitHub & Stack Overflow**  
    Used for community solutions, troubleshooting, and implementation tips