Student Management System using HTML, CSS, JavaScript, Node.js, and MongoDB

---

1. INTRODUCTION

In educational institutions, maintaining student records manually is time-consuming and error-prone. With the growth of web technologies, there is a need for a digital system that efficiently manages student information.

The **Student Management System** is a web-based application designed to store, manage, and retrieve student details easily using modern full-stack technologies.

---

2. OBJECTIVE OF THE PROJECT

The main objectives of this project are:

- To develop a web-based system for managing student data
- To reduce manual work and paperwork
- To provide fast and accurate access to student records
- To implement CRUD operations (Create, Read, Update, Delete)
- To understand full-stack development concepts

---

3. SCOPE OF THE PROJECT

The scope of the Student Management System includes:

- Managing student personal and academic details
- Storing data securely in a database
- Providing a simple and user-friendly interface
- Supporting future enhancements such as login, search, and reports

This project is suitable for schools, colleges, and training institutions.

---

4. TECHNOLOGIES USED

Frontend

- HTML – Structure of the web pages
- CSS – Styling and layout
- JavaScript – Client-side logic and API calls

Backend

- Node.js – Server-side runtime environment
- Express.js – Web framework for REST APIs

Database

- MongoDB – NoSQL database for storing student records

Tools

- Visual Studio Code
- Git & GitHub
- Browser (Chrome)

---

5. SYSTEM ARCHITECTURE

The system follows a **three-tier architecture**:

1. **Presentation Layer** – HTML, CSS, JavaScript
2. **Application Layer** – Node.js and Express

3. **Data Layer** – MongoDB database

The frontend communicates with the backend using HTTP requests, and the backend interacts with the database.

---

6. MODULE DESCRIPTION

6.1 Student Module

- Add new student details
- View list of students
- Delete student records
- Store student information in MongoDB

6.2 Backend API Module

- Handles HTTP requests (GET, POST, DELETE)
- Performs database operations
- Sends responses to frontend

---

7. DATABASE DESIGN

Student Collection Structure

```
{
  "name": "Student Name",
  "email": "student@email.com",
  "department": "CSE",
  "year": 2,
  "cgpa": 8.5
}
```

MongoDB stores the data in JSON-like documents, making it flexible and scalable.

---

8. FUNCTIONAL REQUIREMENTS

- User should be able to add student details
- User should be able to view all students
- User should be able to delete a student
- System should store data permanently in database

---

9. NON-FUNCTIONAL REQUIREMENTS

- User-friendly interface
- Fast response time
- Data consistency
- Easy maintenance and scalability

---

10. IMPLEMENTATION DETAILS

- Frontend uses JavaScript Fetch API to communicate with backend
- Backend uses RESTful APIs built with Express.js
- MongoDB is connected using Mongoose
- CORS is enabled to allow frontend-backend communication

---

11. TESTING

Testing Methods Used:

- Manual testing
- API testing using browser
- CRUD operation validation

Test Cases:

- Add student → Successful insertion
- Fetch students → Data displayed correctly
- Delete student → Data removed from database

---

12. OUTPUT RESULT

The final output is a fully functional web application where:

- Users can add student details
- Student records are displayed dynamically
- Data is stored and retrieved from MongoDB
- Records can be deleted successfully

---

13. ADVANTAGES OF THE SYSTEM

- Reduces manual work
- Saves time and effort
- Easy data management
- Scalable and flexible
- Beginner-friendly full-stack project

---

14. LIMITATIONS

- No authentication system
- No update/edit feature (can be added)
- Basic UI design

---

15. FUTURE ENHANCEMENTS

- Login and authentication
- Edit/update student details
- Search and filter options
- Admin dashboard
- Deployment on cloud platform

---

16. CONCLUSION

The Student Management System successfully demonstrates the use of full-stack web technologies to manage student records efficiently. This project helped in understanding frontend-backend integration, database handling, and REST API development. It can be further enhanced to meet real-world requirements.

---

17. REFERENCES

- https://nodejs.org
- https://www.mongodb.com
- https://developer.mozilla.org
- https://expressjs.com