REPORT

ON

STUDENT RECORD MANAGEMENT SYSTEM (SRMS)

Submitted by:

Bhaumik Hinunia (AP24110010182)

Zeeshan Shaik Suhail (AP24110010183)

Amith Ratna Marisa (AP24110012110)

Prepared in the partial fulfillment of the

Project Based Learning of Course CSE 201 — CODING SKILLS - I

SRM UNIVERSITY AP

Acknowledgements

We would like to express our sincere gratitude to our faculty, **<Faculty Name>**, for providing us with the opportunity to work on this project titled **"Student Record Management System."**
This project has helped us practically understand concepts of full-stack development, API design, and data management.

We also thank our institution, **SRM UNIVERSITY AP**, for providing the necessary resources and an environment to carry out this work successfully.
Finally, we acknowledge the contribution of all team members who collaborated effectively to complete this project within the stipulated time.

ABSTRACT

This project implements a **Student Record Management System (SRMS)** using **HTML, CSS, JavaScript, Node.js, Express.js, and JSON-based storage**.
The system enables educational institutions to efficiently manage student information, attendance, marks, timetable, and student profiles through a unified web interface.

The project supports **role-based access** for Admin, Teacher, and Student, ensuring data security and functional separation. CRUD operations are implemented using REST APIs, and data is persisted using JSON files. The application provides a simple, scalable, and user-friendly solution suitable for college-level academic management.

Table of Contents

1. Introduction

Managing student data manually is time-consuming and inefficient. Educational institutions require systematic digital solutions to maintain student records, attendance, marks, and timetable details.
The **Student Record Management System** is a full-stack web application designed to handle academic data securely and efficiently. The system provides separate roles for Admin, Teacher, and Student, ensuring proper data access control. The frontend is developed using HTML, CSS, and JavaScript, while the backend is built using Node.js and Express. JSON files are used for lightweight data storage, enabling easy deployment and portability.

2. Objectives

The main objectives of this project are:

- To design a simple and interactive web-based system for managing student records.
- To implement role-based access control for Admin, Teacher, and Student.
- To automate processes such as attendance marking and marks entry.
- To create a centralised system for viewing student profiles and report cards.
- To use Node.js and REST APIs for backend communication.
- To store data in JSON format for easy access and modification.

3. SYSTEM OVERVIEW

The system consists of three primary user roles:

3.1 Admin

- Add, edit, delete student information
- Manage timetable
- View attendance and marks

3.2 Teacher

- Mark attendance
- Enter and update marks
- View student profiles
- Access timetable

3.3 Student

- View personal profile
- View attendance percentage
- View marks and grades
- View class timetable

The application runs on a local Node.js server and communicates using REST APIs.

4. TECHNOLOGIES USED

4.1 Frontend

- HTML5
- CSS3
- Vanilla JavaScript

4.2 Backend

- Node.js
- Express.js
- Multer (for photo upload)
- CORS

4.3 Storage

- JSON files
    - students.json
    - attendance.json
    - marks.json
    - timetable.json

4.4 Tools

- Visual Studio Code
- Git & GitHub
- Browser DevTools

5. SYSTEM ARCHITECTURE

Frontend (HTML/CSS/JS)

↓

REST API (Fetch API)

↓

Backend (Node.js + Express)

↓

JSON Storage (Local Database)

- The frontend sends requests using the Fetch API.
- Express.js handles routes and processes data.
- Data is stored in JSON files acting as a lightweight database.

6. MODULES DESCRIPTION

6.1 Student Management Module

- Admin can create, update, and delete students.
- Fields include Roll No, Name, Dept, Semester, CGPA, Phone, Parents, DOB.
- Students' data displayed in a searchable table.

6.2 Attendance Module

- Teachers can mark Present/Absent for each student.
- Attendance records stored date-wise.
- Auto-calculates attendance percentage.

6.3 Marks Module

- Teachers add marks per subject.
- Students can view subject-wise marks and grades.
- Supports CSV export.

6.4 Timetable Module

- Admin/Teacher can manage class schedules.
- Students can view timetable anytime.

6.5 Student Profile & Report Card

- Displays personal details
- Attendance percentage
- Marks summary with grade
- Printable report card

7. IMPLEMENTATION DETAILS

7.1 Frontend

- Built using HTML for structure, CSS for styling, and JavaScript for dynamic behaviour.
- Tab-based navigation for easy access to modules.
- Input validation implemented for all forms.

7.2 Backend

- RESTful API built using Express.js.

- Routes for CRUD operations:
    - /api/students
    - /api/attendance
    - /api/marks
    - /api/timetable
- Multer used for image uploads.

7.3 Data Storage

Each dataset is stored in a separate JSON file:

- Student data → students.json
- Attendance records → attendance.json
- Marks → marks.json
- Timetable → timetable.json

7.4 Security

- Role-based data access (Admin / Teacher / Student).
- Input sanitization.
- Local JSON file access only through backend.

8. Code Snippets

This section provides a few representative code snippets from the project to demonstrate the structure and working of the Student Record Management System. Only the essential parts of the code are presented here. The full code is available in the project repository.

8.1 Backend – Express Server Setup (server.js)

```
const express = require("express");
const fs = require("fs");
const cors = require("cors");
const app = express();

app.use(cors());
app.use(express.json());
app.use(express.static("src"));

// Load students
function loadData(file) {
    return JSON.parse(fs.readFileSync(file, "utf8"));
}

// Save students
function saveData(file, data) {
    fs.writeFileSync(file, JSON.stringify(data, null, 2));
}

app.get("/api/students", (req, res) => {
    res.json(loadData("./backend/students.json"));
});
```

8.2 Frontend – Fetching Students (main.js)

```
async function loadStudents() {
    const res = await fetch("/api/students");
    const students = await res.json();

    const body = document.getElementById("studentsTableBody");
    body.innerHTML = students
        .map(
            (s) => `
            <tr>
                <td>${s.rollNo}</td>
                <td>${s.name}</td>
                <td>${s.department}</td>
                <td>${s.semester}</td>
                <td>${s.cgpa}</td>
                <td>${s.phone}</td>
            </tr>
        `
        )
        .join("");
}
```

8.3 Attendance Saving Logic (attendance.js)

```
document.getElementById("saveAttendanceBtn").addEventListener("click", async ()
    const rows = [...document.querySelectorAll("tr[data-student-id]")];
    const date = document.getElementById("attDate").value;

    const records = rows.map((row) => ({
        studentId: parseInt(row.dataset.studentId),
        status: row.querySelector(".att-status").value,
        date
    }));

    for (const r of records) {
        await fetch("/api/attendance", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify(r)
        });
    }

    alert("Attendance Saved Successfully!");
});
```

8.4 Marks Entry API Route (server.js)

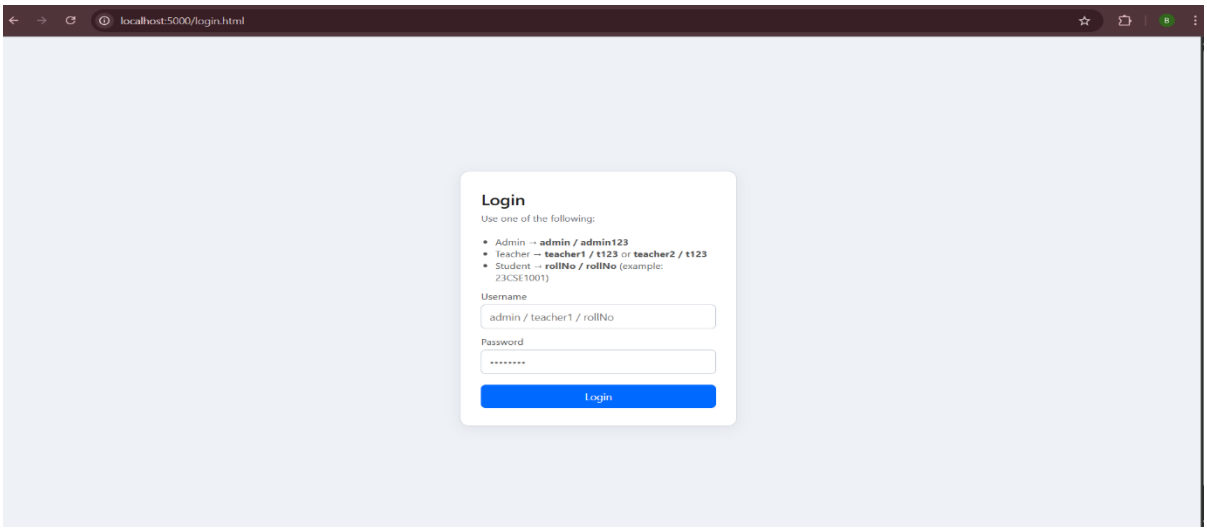
```
app.post("/api/marks", (req, res) => {
    const marks = loadData("./backend/marks.json");
    const newRecord = { id: Date.now(), ...req.body };
    marks.push(newRecord);
    saveData("./backend/marks.json", marks);
    res.json({ success: true });
});
```

8.5 Timetable Rendering (timetable.js)

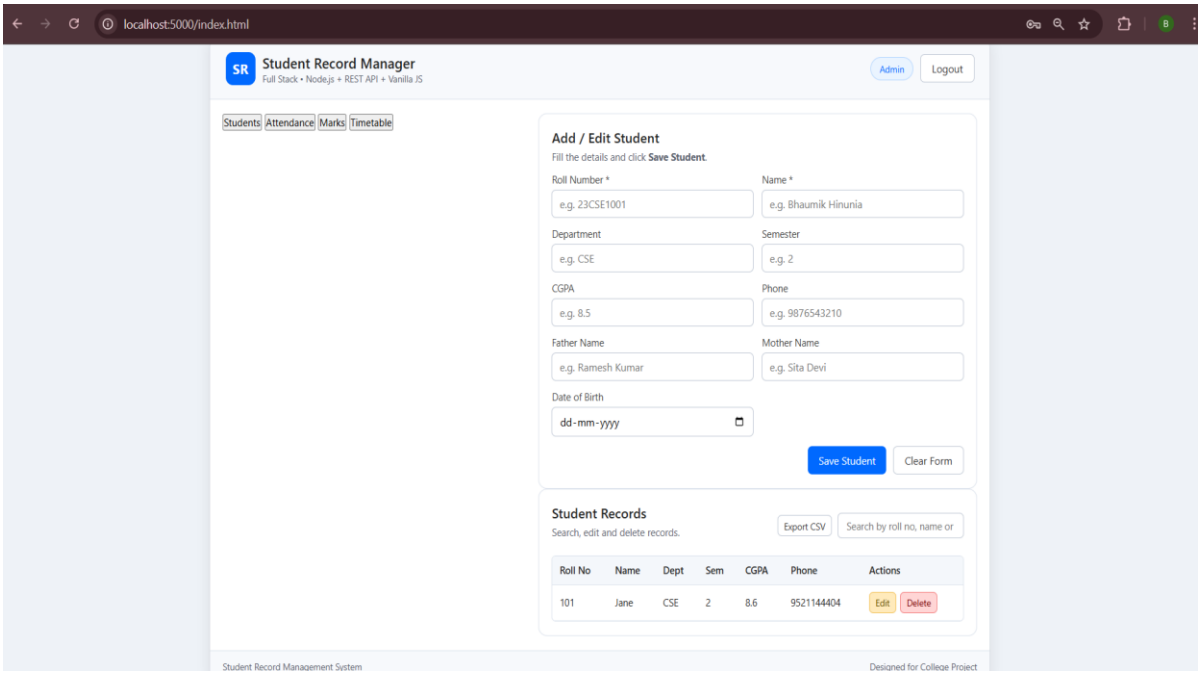
```
async function loadTimetable() {
    const res = await fetch("/api/timetable");
    const table = await res.json();

    document.getElementById("timetableContainer").innerHTML = table
        .map(
            (t) => `
            <div class="timetable-row">
                <b>${t.day}</b> - ${t.subject} (${t.time}) in Room ${t.room}
            </div>
        `
        )
        .join("");
}
```

9. Output Images

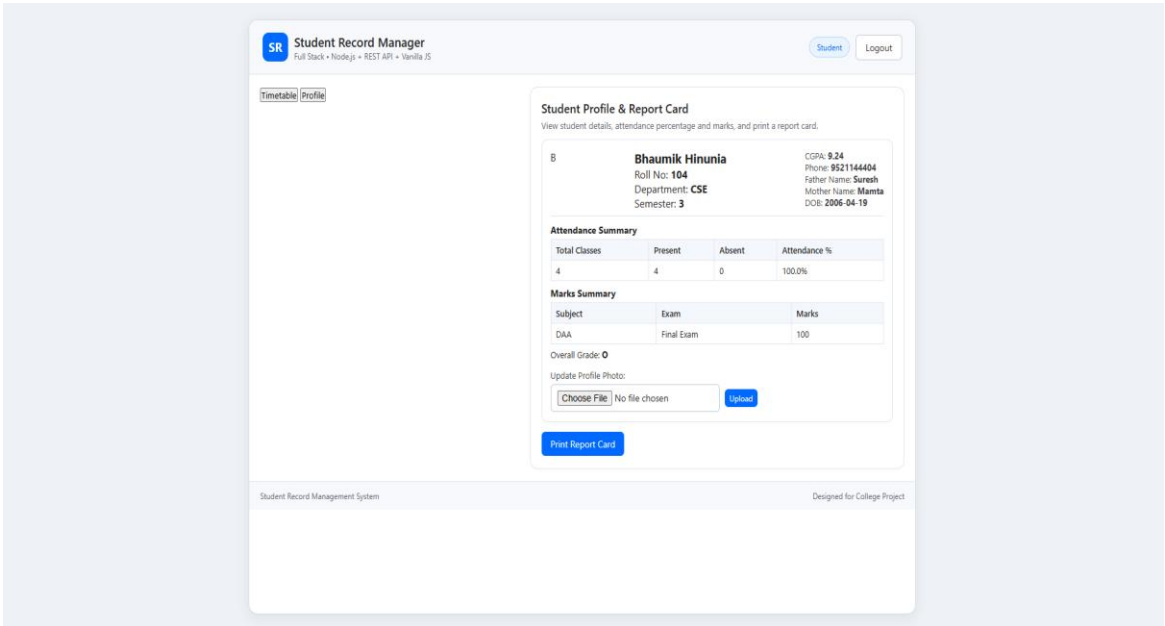
9.1 Login Page



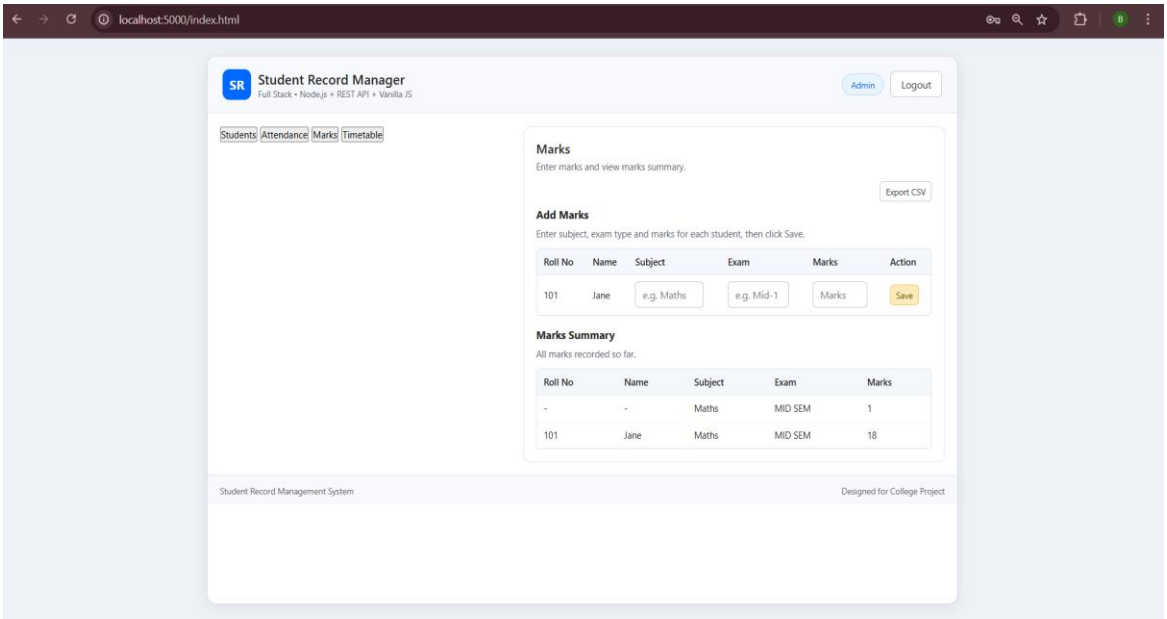
9.2 Add Student



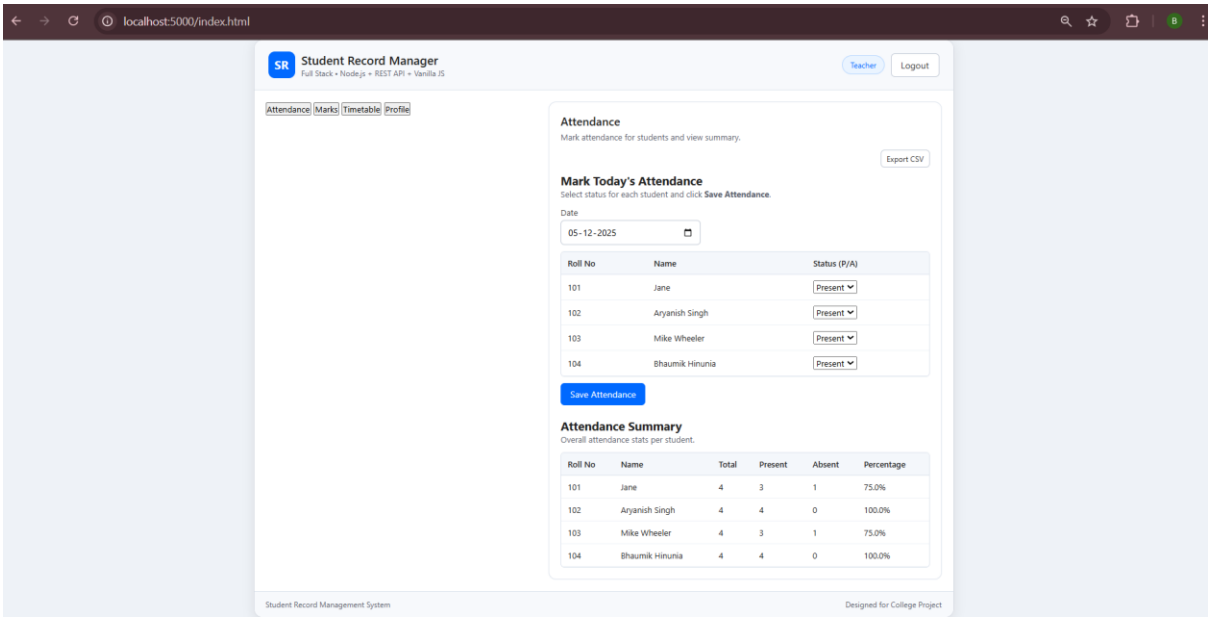
9.3 Student Profile



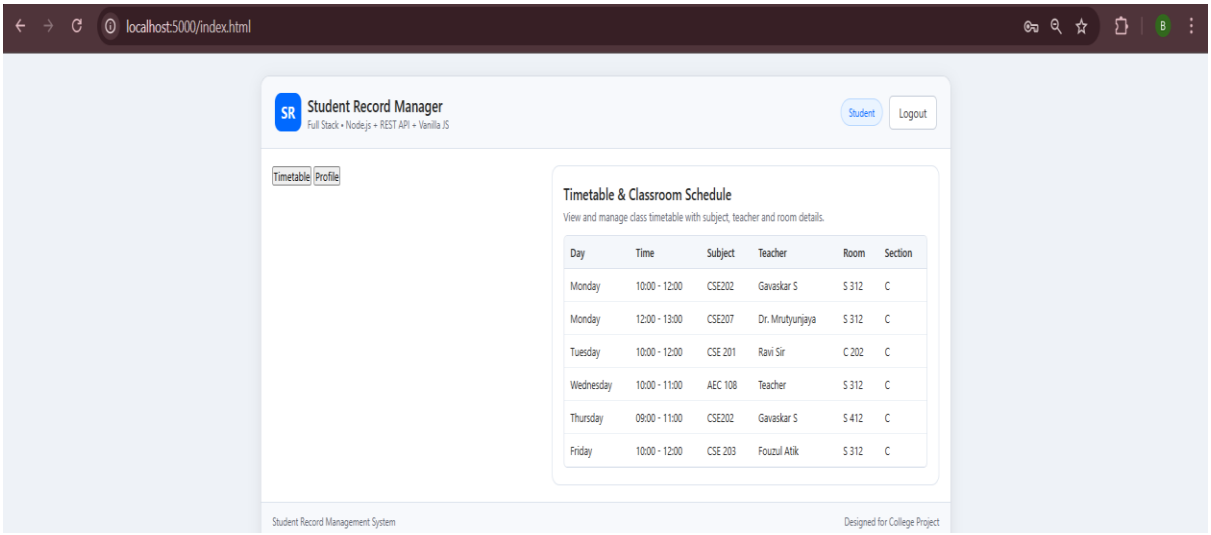
9.4 Admin Add Marks



9.5 Teacher Attendance Marking



9.6 Student Timetable



10. TEST CASES

Test Case	Input	Expected Output	Result
Add Student	Valid details	Student added	Pass
Add Student	Missing name	Error message	Pass
Mark Attendance	Select P/A	Saved successfully	Pass
Add Marks	Valid marks	Marks saved	Pass
Student Login	Valid roll no	Profile loads	Pass

---

11. RESULTS

The developed system successfully provides:

- Accurate student data management
- Automated attendance and marks processing
- User-friendly dashboard for each role
- Working backend with complete CRUD operations
- Clean, responsive interface

All modules were tested and function as expected.

12. CONCLUSION

The Student Record Management System meets all project objectives and provides an efficient solution for managing academic records.

Using Node.js and JSON files made the system simple, portable, and suitable for college-level implementation.

The project enhanced our understanding of full-stack development, REST APIs, and modular design principles.

---

13. FUTURE ENHANCEMENTS

- Replace JSON storage with MySQL or MongoDB
- Add authentication using JWT
- Add analytics dashboards with charts
- Introduce SMS/Email notifications
- Cloud deployment using Render/Vercel

---

14. REFERENCES

- Node.js Official Documentation
- Express.js Guide
- MDN Web Docs (HTML, CSS, JS)
- Stack Overflow
- W3Schools Tutorial