

REPORT
ON
STUDENT RECORD MANAGEMENT SYSTEM (SRMS)

Submitted by:

Bhaumik Hinunia (AP24110010182)

Zeeshan Shaik Suhail (AP24110010183)

Amith Ratna Marisa (AP24110012110)

Manya Srivastava (AP24110010171)

Jayanti Yadav (AP24110010132)

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 SRM UNIVERSITY AP and Computer Science & Engineering, 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

- Introduction
- Objectives
- System Overview
- Technologies Used
- System Architecture
- Modules Description
- Implementation Details
- Test Cases
- Results
- Conclusion
- Future Enhancements
- References

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. Problem Statement

Educational institutions manage large volumes of student-related information such as personal details, attendance records, academic performance, and class schedules. When handled manually or using separate tools, these operations become time-consuming, error-prone, and difficult to maintain. Teachers and administrators often struggle to update data efficiently, while students lack a centralized platform to view their academic information.

Therefore, there is a need for a unified system that can securely store student data, automate attendance and marks entry, provide role-based access for different users, and generate accurate profiles and reports. The problem addressed by this project is the absence of an integrated, user-friendly, and efficient Student Record Management System that simplifies academic data handling for administrators, teachers, and students.

3. 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

4. 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.

5. TECHNOLOGIES USED

5.1 Frontend

- HTML5
- CSS3
- Vanilla JavaScript

5.2 Backend

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

5.3 Storage

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

5.4 Tools

- Visual Studio Code
- Git & GitHub
- Browser DevTools

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 });
});
```

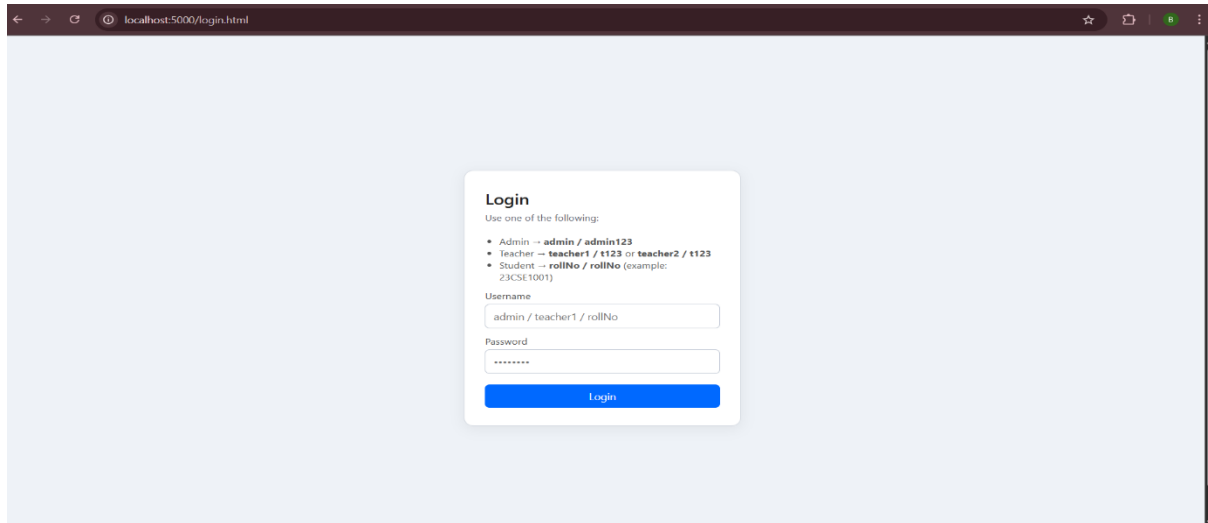
8.5 Timetable Rendering (timetable.js)

```
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



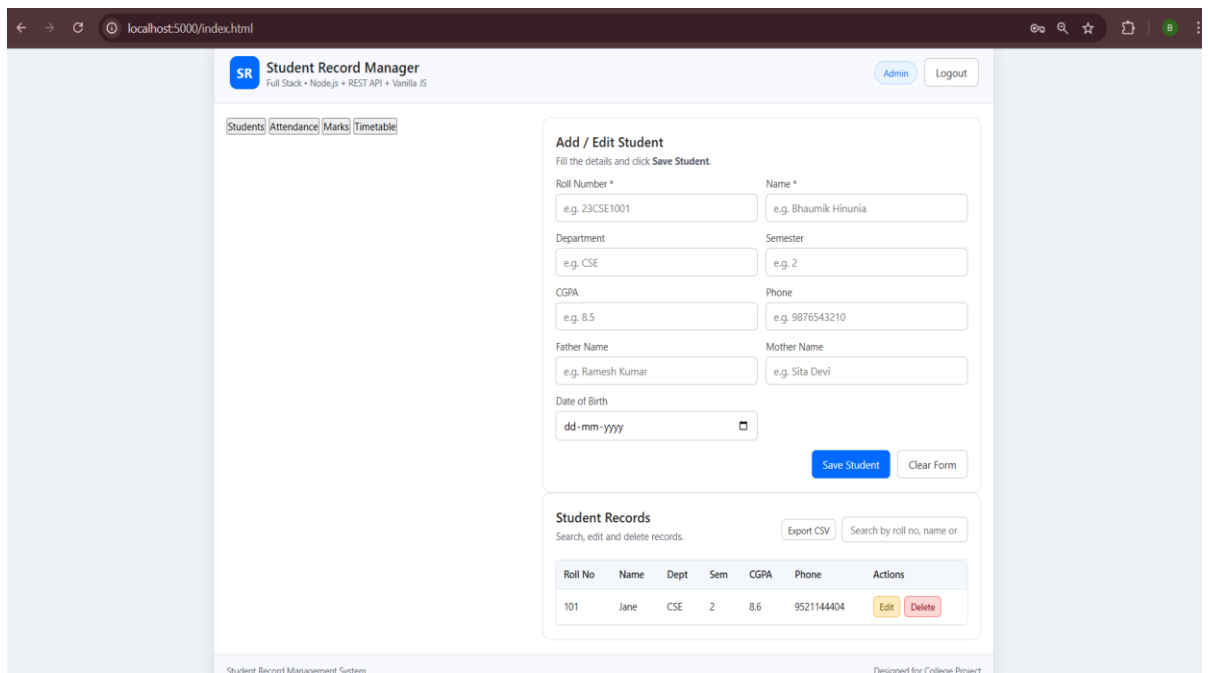
A screenshot of a web browser showing a login page at localhost:5000/login.html. The page has a light blue background. In the center, there is a white login card with a blue border. The card contains the following text and form elements:

- Login**
- Use one of the following:

 - Admin → **admin** / **admin123**
 - Teacher → **teacher1** / **t123** or **teacher2** / **t123**
 - Student → **rollNo** / **rollNo** (example: 23CSE1001)

- Username:
- Password:
- Login** button

9.2 Add Student



A screenshot of a web browser showing the 'Add / Edit Student' page in a 'Student Record Manager' application. The browser address bar shows localhost:5000/index.html. The application header includes a logo 'SR' and the text 'Student Record Manager Full Stack • Nodejs • REST API • Vanilla JS'. There are 'Admin' and 'Logout' buttons in the top right. A navigation bar has tabs for 'Students', 'Attendance', 'Marks', and 'Timetable'. The main content area is titled 'Add / Edit Student' and includes the instruction 'Fill the details and click Save Student.' Below this are several form fields:

- Roll Number * (e.g. 23CSE1001)
- Name * (e.g. Bhaumik Hinunia)
- Department (e.g. CSE)
- Semester (e.g. 2)
- CGPA (e.g. 8.5)
- Phone (e.g. 9876543210)
- Father Name (e.g. Ramesh Kumar)
- Mother Name (e.g. Sita Devi)
- Date of Birth (dd-mm-yyyy)

At the bottom of the form are 'Save Student' and 'Clear Form' buttons. Below the form is a section titled 'Student Records' with the instruction 'Search, edit and delete records.' and an 'Export CSV' button. A search bar says 'Search by roll no, name or'. Below this is a table with student records:

Roll No	Name	Dept	Sem	CGPA	Phone	Actions
101	Jane	CSE	2	8.6	9521144404	Edit Delete

The footer of the application shows 'Student Record Management System' on the left and 'Designed for College Project' on the right.

9.3 Student Profile

SR

Student Record Manager

Full Stack • Node.js • REST API • Vanilla JS

Student

Logout

Timetable

Profile

Student Profile & Report Card

View student details, attendance percentage and marks, and print a report card.

B

Bhaumik Hinunia

Roll No: 104

Department: CSE

Semester: 3

CGPA: 9.24

Phone: 9521144404

Father Name: Suresh

Mother Name: Manita

DOB: 2006-04-19

Attendance Summary

Total Classes	Present	Absent	Attendance %
4	4	0	100.0%

Marks Summary

Subject	Exam	Marks
DAA	Final Exam	100

Overall Grade: O

Update Profile Photo:

Choose File

No file chosen

Upload

Print Report Card

Student Record Management System

Designed for College Project

9.4 Admin Add Marks

The screenshot displays a web browser window with the address bar showing 'localhost:5000/index.html'. The application, titled 'Student Record Manager', features a navigation bar with tabs for 'Students', 'Attendance', 'Marks', and 'Timetable'. The 'Marks' tab is currently selected. The main content area is divided into two sections: 'Add Marks' and 'Marks Summary'.

Add Marks Section:

Enter subject, exam type and marks for each student, then click Save.

Roll No	Name	Subject	Exam	Marks	Action
101	Jane	<input type="text" value="e.g. Maths"/>	<input type="text" value="e.g. Mid-1"/>	<input type="text" value="Marks"/>	<input type="button" value="Save"/>

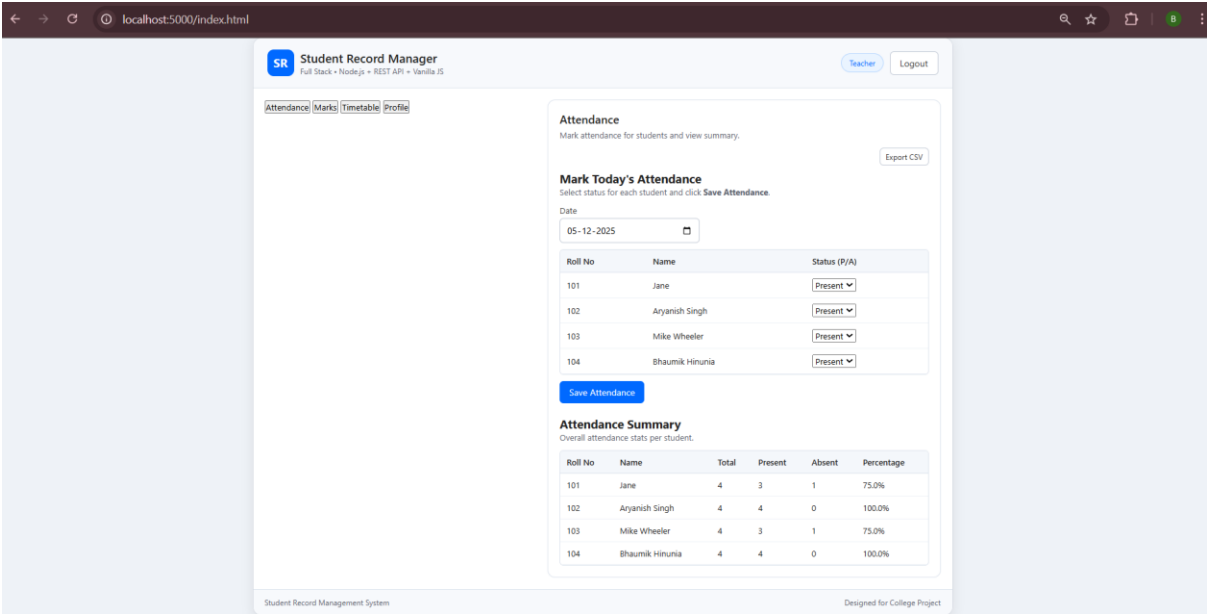
Marks Summary Section:

All marks recorded so far.

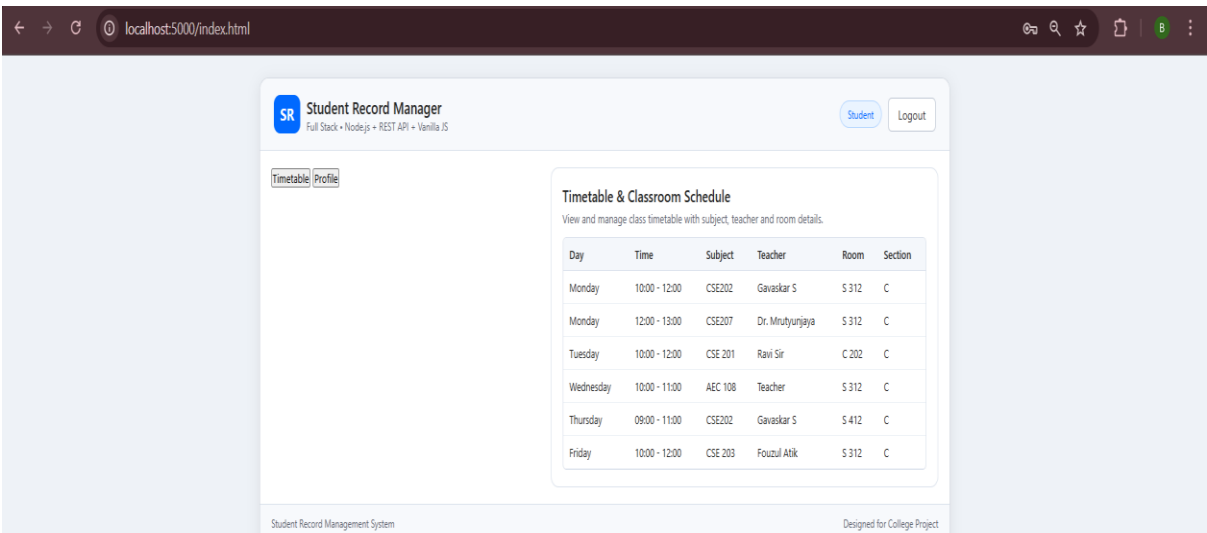
Roll No	Name	Subject	Exam	Marks
-	-	Maths	MID SEM	1
101	Jane	Maths	MID SEM	18

The footer of the application shows 'Student Record Management System' on the left and 'Designed for College Project' on the right.

9.5 Teacher Attendance Marking

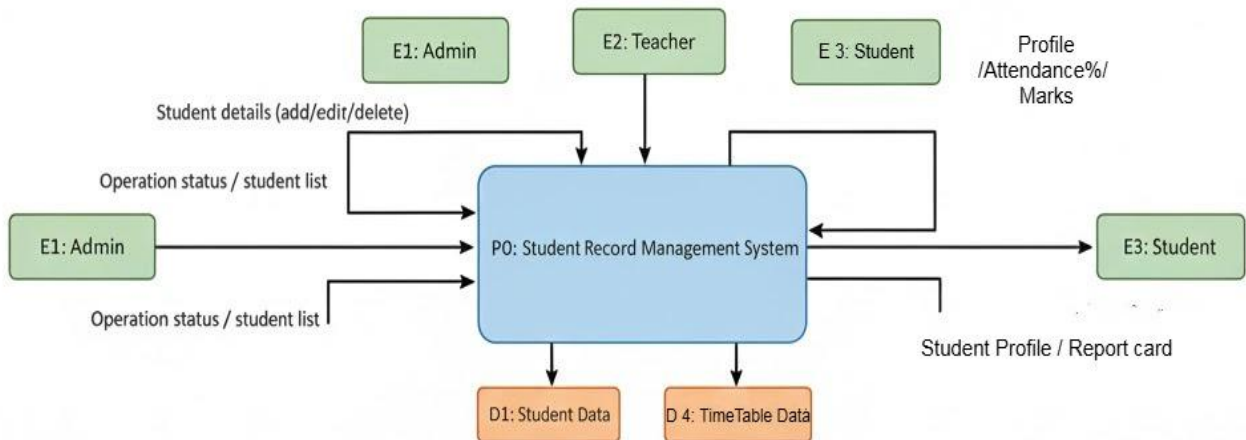


9.6 Student Timetable

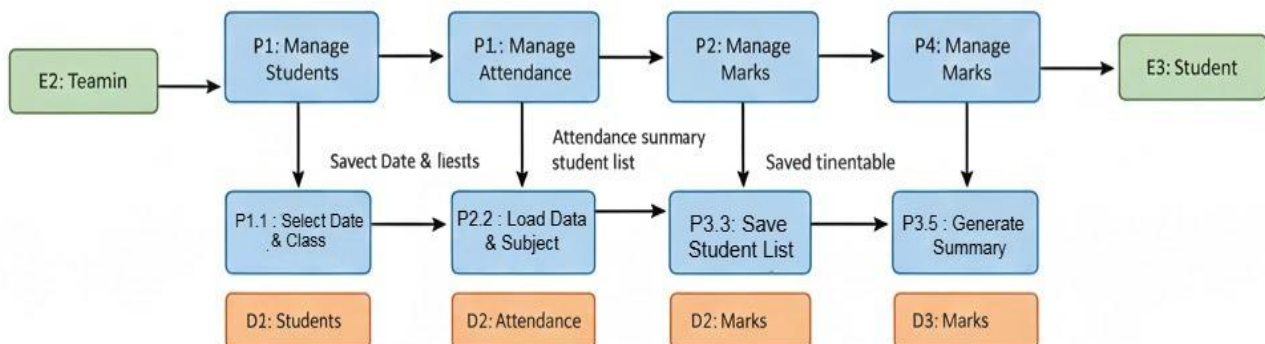


10. Data Flow Diagram

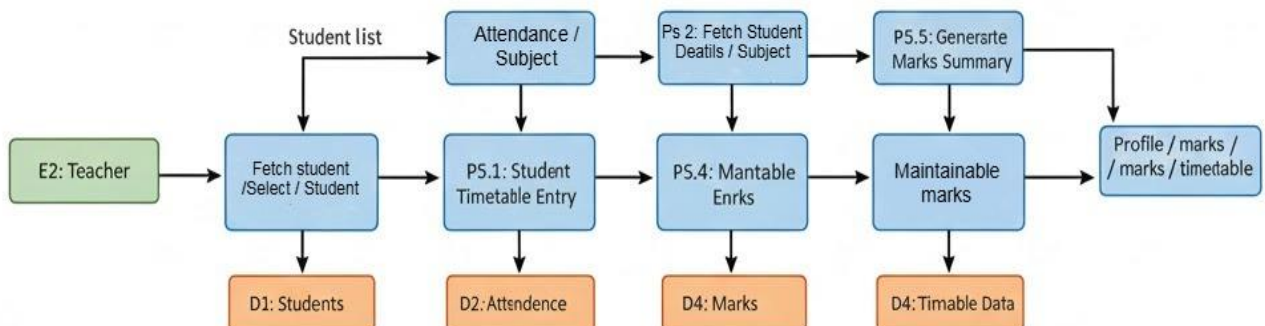
DFD – Level 0 for P1): Manage Diagram)



DFD – Level 2 (for P2): Manage Timetble



DFD – Level 2 for P4): Profile & Report



11. 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

12. 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.

13. 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](#)

