

College Name :**Thangavelu Engineering College**

College Code : **3126**

Student Name : **PAVITHRA M**

Department: **computer science and engineering**

Naan Mudhalvan (NM) ID:

CE605E921B07F46B1B14E0322728D87C

Phone Number : **6374008135**

Email ID :**pavithramoorthi63@gmail.com**

GitHub Repository L

STUDENT MANAGEMENT APP

1. INTRODUCTION

The **Student Management App** is a full-stack web application developed using **JavaScript** for both frontend and backend.

This application helps to manage student records such as name, roll number, department, email, and phone number using CRUD operations.

2. PROJECT OBJECTIVE

- To understand full-stack development using JavaScript
 - To implement CRUD operations
 - To connect frontend and backend
 - To store data using MongoDB
 - To gain hands-on experience in REST API development
-

3. TECHNOLOGIES USED

- **Frontend:**
 - HTML
 - CSS
 - JavaScript
- **Backend:**
 - Node.js
 - Express.js
- **Database:**
 - MongoDB
- **Tools:**

- VS Code
- GitHub
- Postman

4. SYSTEM REQUIREMENTS

- Windows 10 or above
- Node.js installed
- MongoDB installed or MongoDB Atlas
- Web Browser (Chrome)

5. PROJECT FOLDER STRUCTURE

student-management-app

```
|
|
|└─ backend
|  |
|  |└─ models
|  |  |└─ Student.js
|  |└─ routes
|  |  |└─ studentRoutes.js
|  |└─ server.js
|  └─ package.json
|
|
|└─ frontend
|  |└─ index.html
|  |└─ style.css
|  └─ script.js
|
└─ README.md
```

6. BACKEND IMPLEMENTATION

Step 1: Initialize Backend

- Create backend folder
- Run npm init
- Install required packages

Step 2: MongoDB Connection

- Connect MongoDB using mongoose
- Verify database connection

Step 3: Create Student Schema

Fields Used:

- Name
- Roll Number
- Department
- Email
- Phone Number

Step 4: Create REST API Routes

- POST – Add Student
- GET – View Students
- PUT – Update Student
- DELETE – Delete Student

Step 5: Error Handling

- Handle server and database errors

7. FRONTEND IMPLEMENTATION

- Create HTML form for student details
- Design UI using CSS
- Handle user actions using JavaScript

- Send requests to backend using fetch API
-

8. CRUD OPERATIONS EXPLANATION

- **Create:** Add new student details
 - **Read:** Display all students
 - **Update:** Modify existing student details
 - **Delete:** Remove student record
-

9. CONNECTING FRONTEND AND BACKEND

- Frontend sends HTTP requests
 - Backend processes requests
 - MongoDB stores data
 - Response sent back to frontend
-

10. TESTING & OUTPUT

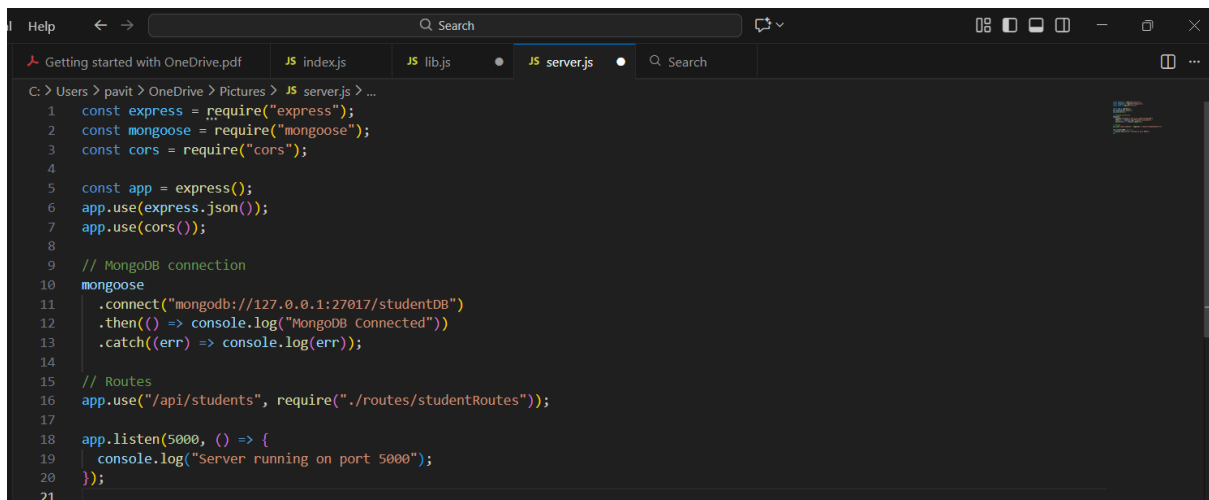
- Tested APIs using Postman
 - Verified CRUD operations
 - Checked frontend form submission
 - Data successfully stored in MongoDB
-

11. CONCLUSION

The **Student Management App** successfully demonstrates a full-stack application using JavaScript.

This project helped in understanding backend development, frontend interaction, and database integration.

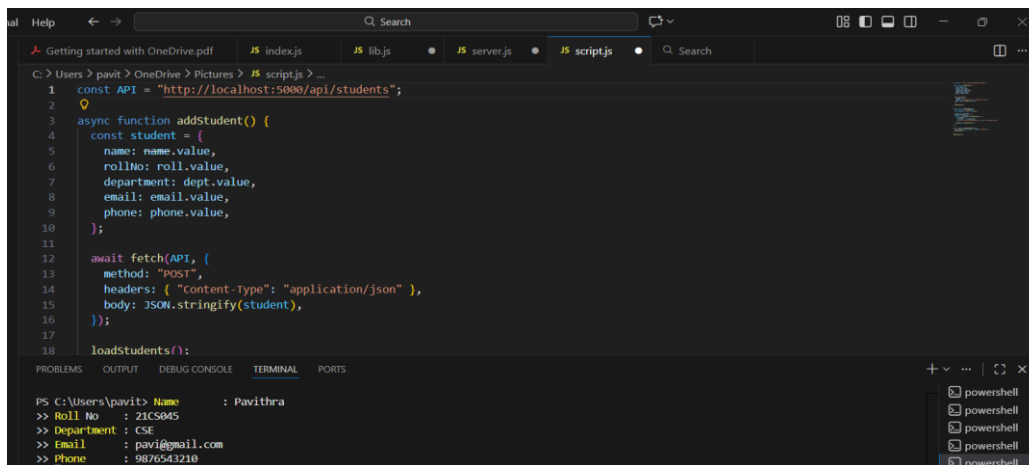
BACKEND :



The screenshot shows a Visual Studio Code editor window with the file `server.js` open. The code is written in JavaScript and sets up an Express server with MongoDB integration. The code includes comments for MongoDB connection and routes. The server is configured to listen on port 5000.

```
1  const express = require("express");
2  const mongoose = require("mongoose");
3  const cors = require("cors");
4
5  const app = express();
6  app.use(express.json());
7  app.use(cors());
8
9  // MongoDB connection
10 mongoose
11   .connect("mongodb://127.0.0.1:27017/studentDB")
12   .then(() => console.log("MongoDB Connected"))
13   .catch(err => console.log(err));
14
15 // Routes
16 app.use("/api/students", require("./routes/studentRoutes"));
17
18 app.listen(5000, () => {
19   console.log("Server running on port 5000");
20 });
21
```

FRONTEND:



The screenshot shows a Visual Studio Code editor window with the file `script.js` open. The code defines a constant `API` and an `addStudent` function. The `addStudent` function uses `fetch` to send a POST request to the API. Below the code, a terminal window is open, showing the output of a script that prompts the user for student information.

```
1  const API = "http://localhost:5000/api/students";
2
3  async function addStudent() {
4    const student = {
5      name: name.value,
6      rollNo: roll.value,
7      department: dept.value,
8      email: email.value,
9      phone: phone.value,
10   };
11
12   await fetch(API, {
13     method: "POST",
14     headers: { "Content-Type": "application/json" },
15     body: JSON.stringify(student),
16   });
17
18   loadStudents();
19 }
20
21 loadStudents();
```

Terminal Output:

```
PS C:\Users\pavit> Name : Pavithra
>> Roll No : 21CS045
>> Department : CSE
>> Email : pavi@gmail.com
>> Phone : 9876543210
```