



COLLEGE NAME: MEENAKSHI COLLEGE OF
ENGINEERING

COLLEGE CODE:3114

YOUR NAME: ARIVANANTHAM.J

NAAN MUDHALVAN (NM) ID:
C243912FBA9215698764EA53EB29B62B

PHONE NO:8807194452

EMAIL ID: nandhamarivu06@gmail.com

GIT HUB LINK: <https://github.com/Arivu611/crud.git>

The Student Attendance System

1. Introduction:

- Brief description of the project:

The Student Attendance System is a web-based application designed to help teachers and educational institutions efficiently record, manage, and track student attendance.

- Problem statement:

Manual attendance systems are time-consuming and prone to errors. This system automates attendance recording, reducing errors and saving time.

- Purpose of the system:
 - Easy attendance management
 - Quick access to attendance records
 - Reports and updates on student attendance
-

2. Objectives:

- To create an easy-to-use system for teachers and administrators.
- To ensure attendance is accurately recorded daily.
- To provide real-time tracking of student attendance.
- To reduce paper-based errors and save administrative time.

3. Features of the System:

- **User Interface:** Simple dashboard for teachers and admins
- **Add Student Attendance:** Mark attendance as Present/Absent
- **Update Attendance:** Ability to correct mistakes using PUT requests
- **Prevent Duplicates:** Ensures a student's attendance for a day cannot be entered twice
- **Reports:** View student attendance history
- **Database:** MongoDB stores all attendance records securely

4. Technologies Used:

- **Frontend:** HTML, CSS, JavaScript (optional framework if used)
- **Backend:** Node.js, Express.js
- **Database:** MongoDB (for storing student and attendance data)
- **Tools:** VS Code, Postman for API testing, Node Package Manager (npm)

5. System Design:

5.1 Architecture Diagram (optional: you can draw and paste)

- Client → Frontend → Backend → Database (MongoDB)

5.2 Database Design:

- **Collection:** Students
 - rollNumber (String, unique per day)
 - date (Date)
 - status (String: Present/Absent)

5.3 Routes:

Route	Method	Purpose
/attendance	POST	Add new attendance
/attendance	PUT	Update existing attendance
/attendance	GET	Fetch all attendance records

6. Implementation Steps:

6.1 Setup Environment:

1. Install Node.js and npm
2. Install MongoDB
3. Install project dependencies: express, mongoose

6.2 Create Server:

- Create index.js
- Connect to MongoDB
- Add middleware: express.json()

6.3 Create Routes:

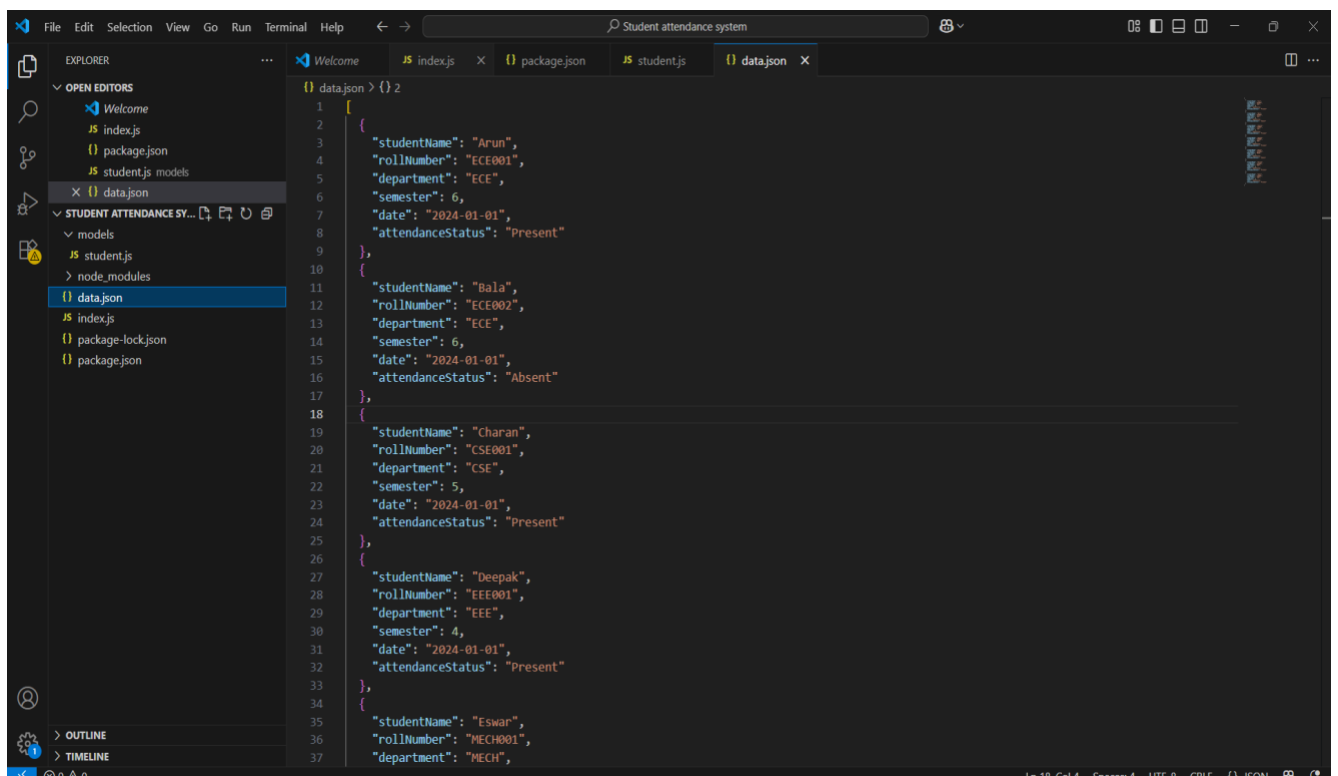
- POST: Add attendance
- PUT: Update attendance
- GET: View attendance

6.4 Create Student Schema:

- Fields: rollNumber, date, status
- Unique index: rollNumber + date

6.5 Test API:

- Using Postman or browser fetch requests
- Ensure duplicate entries are handled
- Validate JSON data



The screenshot shows a Visual Studio Code editor window with a project named "Student attendance system". The Explorer sidebar on the left shows the file structure, including "data.json" which is currently selected. The main editor area displays the content of "data.json", which is a JSON array of five student attendance records. The records are for students Arun, Bala, Charan, Deepak, and Eswar, each with their roll number, department, semester, date, and attendance status.

```
1  {} data.json > {} 2
2  {
3    "studentName": "Arun",
4    "rollNumber": "ECE001",
5    "department": "ECE",
6    "semester": 6,
7    "date": "2024-01-01",
8    "attendanceStatus": "Present"
9  },
10 {
11   "studentName": "Bala",
12   "rollNumber": "ECE002",
13   "department": "ECE",
14   "semester": 6,
15   "date": "2024-01-01",
16   "attendanceStatus": "Absent"
17 },
18 {
19   "studentName": "Charan",
20   "rollNumber": "CSE001",
21   "department": "CSE",
22   "semester": 5,
23   "date": "2024-01-01",
24   "attendanceStatus": "Present"
25 },
26 {
27   "studentName": "Deepak",
28   "rollNumber": "EEE001",
29   "department": "EEE",
30   "semester": 4,
31   "date": "2024-01-01",
32   "attendanceStatus": "Present"
33 },
34 {
35   "studentName": "Eswar",
36   "rollNumber": "MECH001",
37   "department": "MECH",
```

The screenshot shows the VS Code editor with the 'index.js' file open. The Explorer sidebar on the left shows the project structure: 'OPEN EDITORS' (index.js, package.json, student.js, data.json), 'STUDENT ATTENDANCE SYSTEM' (models, student.js, node_modules, data.json, index.js, package-lock.json, package.json), and 'OUTLINE' and 'TIMELINE' tabs at the bottom. The main editor area displays the following JavaScript code:

```
1 const express = require("express");
2 const mongoose = require("mongoose");
3 const Student = require("./models/student");
4 const data = require("./data.json");
5
6 const app = express();
7 app.use(express.json());
8
9
10 mongoose.connect("mongodb://127.0.0.1:27017/attendanceDB")
11 .then(() => console.log("MongoDB connected"))
12 .catch(err => console.log(err));
13
14
15 app.get("/insert", async (req, res) => {
16   try {
17     await Student.insertMany(data);
18     res.send("Sample attendance data inserted");
19   } catch (err) {
20     res.status(400).send(err.message);
21   }
22 });
23
24 /* CREATE */
25 app.post("/attendance", async (req, res) => {
26   try {
27     const student = new Student(req.body);
28     await student.save();
29     res.send(student);
30   } catch (err) {
31     res.status(400).send(err.message);
32   }
33 });
34
35 /* READ ALL */
36 app.get("/attendance", async (req, res) => {
37   const students = await Student.find();
```

The status bar at the bottom indicates 'Ln 9, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

The screenshot shows the VS Code editor with the 'package.json' file open. The Explorer sidebar on the left shows the project structure: 'OPEN EDITORS' (index.js, package.json, student.js, data.json), 'STUDENT ATTENDANCE SYSTEM' (models, student.js, node_modules, data.json, index.js, package-lock.json, package.json), and 'OUTLINE' and 'TIMELINE' tabs at the bottom. The main editor area displays the following JSON code:

```
1 {
2   "name": "student-attendance-system",
3   "version": "1.0.0",
4   "description": "Student Attendance Management System",
5   "main": "index.js",
6   "scripts": {
7     "start": "node index.js"
8   },
9   "dependencies": {
10    "express": "^4.18.2",
11    "mongoose": "^7.6.0"
12  }
13 }
```

The status bar at the bottom indicates 'Ln 13, Col 2', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JSON'.

This screenshot shows the Visual Studio Code editor with the `package-lock.json` file open. The Explorer sidebar on the left shows the project structure, including `models`, `studentjs`, `data.json`, and `package-lock.json`. The main editor displays the contents of `package-lock.json`, which is a JSON file defining the project's dependencies and their versions. The file includes the project name "student-attendance-system", version "1.0.0", and lists dependencies such as `express` and `mongoose`. It also includes the resolved paths for these dependencies.

```
1 {
2   "name": "student-attendance-system",
3   "version": "1.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "student-attendance-system",
9       "version": "1.0.0",
10      "dependencies": {
11        "express": "^4.18.2",
12        "mongoose": "^7.6.0"
13      },
14    },
15    "node_modules/@mongodb-js/saslprep": {
16      "version": "1.4.4",
17      "resolved": "https://registry.npmjs.org/@mongodb-js/saslprep/-/saslprep-1.4.4.tgz",
18      "integrity": "sha512-p7X/ytJD1dwUffL/CL0hkgdfe1Fa8uw9seJVvdOmpP9JBW6W4M69Hk0IXS6W9yrvGf1MbhcS6lVmrhy4jm2g==",
19      "license": "MIT",
20      "optional": true,
21      "dependencies": {
22        "sparse-bitfield": "^3.0.3"
23      },
24    },
25    "node_modules/@types/node": {
26      "version": "25.0.3",
27      "resolved": "https://registry.npmjs.org/@types/node/-/node-25.0.3.tgz",
28      "integrity": "sha512-W609bulVRVmeW693xKfZHeIV6nJGGZ98uCPfeXI1ELMLXVeKY29m15FAMSaUPBHYLGFsVRCMmScksQOrZV9BYA==",
29      "license": "MIT",
30      "dependencies": {
31        "undici-types": "~7.16.0"
32      },
33    },
34    "node_modules/@types/webidl-conversions": {
35      "version": "7.0.3",
36      "resolved": "https://registry.npmjs.org/@types/webidl-conversions/-/webidl-conversions-7.0.3.tgz",
37      "integrity": "sha512-C1j3CtE1YwDfz1YwFfzH0L11L1ELMLXVeKY29m15FAMSaUPBHYLGFsVRCMmScksQOrZV9BYA==",
```

This screenshot shows the Visual Studio Code editor with the `studentjs.js` file open. The Explorer sidebar on the left shows the project structure, including `models`, `studentjs`, `data.json`, and `package-lock.json`. The main editor displays the contents of `studentjs.js`, which is a JavaScript file defining a Mongoose schema for a student. The schema includes fields for `studentName`, `rollNumber`, `department`, `semester`, `date`, and `attendanceStatus`. The `attendanceStatus` field is an enum with values "Present" and "Absent". The file also includes a comment about preventing duplicate attendance for the same student and date, and an index definition for the schema.

```
1 const mongoose = require("mongoose");
2
3 const studentSchema = new mongoose.Schema({
4   studentName: {
5     type: String,
6     required: true
7   },
8   rollNumber: {
9     type: String,
10    required: true
11  },
12  department: {
13    type: String,
14    required: true
15  },
16  semester: {
17    type: Number,
18    required: true,
19    min: 1
20  },
21  date: {
22    type: Date,
23    required: true
24  },
25  attendanceStatus: {
26    type: String,
27    enum: ["Present", "Absent"],
28    required: true
29  }
30 });
31
32 /* Prevent duplicate attendance for same student & date */
33 studentSchema.index(
34   { rollNumber: 1, date: 1 },
35   { unique: true }
36 );
37
```

8. Testing and Screenshots:

- Show screenshots of Postman testing POST, PUT, GET
 - Show sample attendance records in MongoDB Compass (optional)
 - Explain any errors and how they were fixed (like duplicate key errors, JSON syntax errors)
-

9. Conclusion:

- Summarize the benefits:
 - Accurate, fast, and automated attendance system
 - Reduced manual effort and errors
 - Easy tracking and reporting of student attendance
 - Future improvements:
 - Add login system for teachers
 - Generate monthly or yearly attendance reports
 - Add notifications for absent students
-

10. References:

- Node.js Documentation
- MongoDB Documentation
- Express.js Documentation
- Tutorials or blogs you followed

WorkspaceForTestingAPI By Sachin Vishwakarma

GET GET http://localhost:3000/attendance

Overview Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "studentName": "arivu",
3   "rollNumber": "EEE007",
4   "department": "EEE",
5   "semester": 6,
6   "date": "2024-02-01",
7   "attendanceStatus": "Present"
8 }
9
```

Body Cookies Headers (7) Test Results (1/2) 200 OK • 187 ms • 1.66 KB Save Response

JSON Preview Visualize

```
1 [
2   {
3     "_id": "695bc635e2bae43fe6ff15c2",
4     "studentName": "Arun",
5     "rollNumber": "ECE001",
6     "department": "ECE",
7     "semester": 6,
8     "date": "2024-01-01T00:00:00.000Z",
9     "attendanceStatus": "Present",
10    "__v": 0
11  },
12  {
13    "_id": "695bc635e2bae43fe6ff15c3",
14    "studentName": "Bala",
15  }
16 ]
```

Online Find and replace Console Runner Start Proxy Cookies Vault Trash

WorkspaceForTestingAPI By Sachin Vishwakarma

POST GET http://localhost:3000/attendance

Overview Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "studentName": "arivu",
3   "rollNumber": "EEE007",
4   "department": "EEE",
5   "semester": 6,
6   "date": "2024-02-01",
7   "attendanceStatus": "Present"
8 }
9
```

Body Cookies Headers (7) Test Results (1/2) 200 OK • 189 ms • 417 B Save Response

JSON Preview Visualize

```
1 {
2   "studentName": "arivu",
3   "rollNumber": "EEE007",
4   "department": "EEE",
5   "semester": 6,
6   "date": "2024-02-01T00:00:00.000Z",
7   "attendanceStatus": "Present",
8   "_id": "695bc933e2bae43fe6ff15d09",
9   "__v": 0
10 }
```

Online Find and replace Console Runner Start Proxy Cookies Vault Trash

MongoDB Compass - Arivanantham/attendanceDB.students

Connections Edit View Collection Help

Compass

My Queries

Data Modelling

CONNECTIONS (1)

Search connections

Arivanantham

admin

attendanceDB

students

config

local

startup_log

attendanceDB

mongosh: Arivanantham

admin

students

config

startup_log

+

Arivanantham

attendanceDB

students

Open MongoDB shell

Documents 8

Aggregations

Schema

Indexes 2

Validation

Create Index

Refresh

VIEWING

INDEXES

SEARCH INDEXES

Name & Definition	Type	Size	Usage	Properties	Status
> _id_	REGULAR	36.9 kB	5 (since Mon Jan 05 2026)	UNIQUE	READY
> rollNumber_1_date_1	REGULAR	36.9 kB	0 (since Mon Jan 05 2026)	UNIQUE COMPOUND	READY

