



**COLLEGE NAME:** MEENAKSHI COLLEGE OF  
ENGINEERING

**COLLEGE CODE:**3114

**YOUR NAME:** ARIVANANTHAM.J

**NAAN MUDHALVAN (NM) ID:**

C243912FBA9215698764EA53EB29B62B

**PHONE NO:**8807194452

**EMAIL ID:** nandhamariyu06@gmail.com

**GIT HUB LINK:** <https://github.com/Ariyu611/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:**

<b>Route</b>	<b>Method</b>	<b>Purpose</b>
/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:

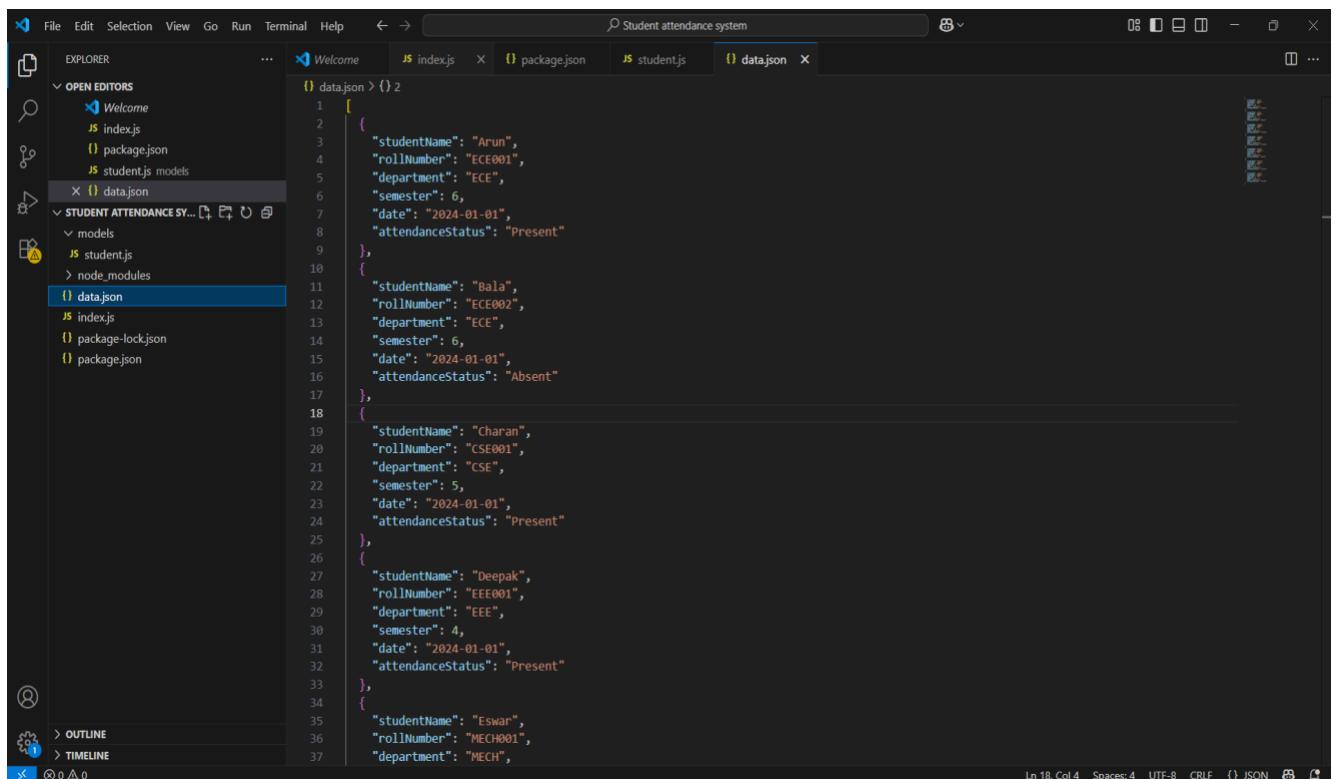
- POST: Add attendance
- PUT: Update attendance
- GET: View attendance

## 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



```
Student attendance system

File Edit Selection View Go Run Terminal Help ⏎ → Explorer ... Welcome JS index.js X package.json JS student.js X data.json

OPEN EDITORS
  Welcome
  JS index.js
  package.json
  JS student.js
  models
  STUDENT ATTENDANCE SY...
    models
      JS student.js
    node_modules
      data.json
      JS index.js
      package-lock.json
      package.json

STUDENT ATTENDANCE SYSTEM
  models
    JS student.js
  node_modules
    data.json
    JS index.js
    package-lock.json
    package.json

data.json > {} 2
1 [
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 Visual Studio Code interface with the title bar "Student attendance system". The left sidebar displays the file structure under "STUDENT ATTENDANCE SYSTEM", including "models", "student.js", "node\_modules", "data.json", "index.js", "package-lock.json", and "package.json". The main editor area shows the "index.js" file content:

```
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 {} JavaScript".

The screenshot shows the Visual Studio Code interface with the title bar "Student attendance system". The left sidebar displays the file structure under "STUDENT ATTENDANCE SYSTEM", including "models", "student.js", "node\_modules", "data.json", "index.js", "package-lock.json", and "package.json". The main editor area shows the "package.json" file content:

```
1 {
2   "name": "student-attendance-system",
3   "version": "1.0.0",
4   "description": "Student Attendance Management System",
5   "main": "index.js",
6   "Debug": {
7     "scripts": {
8       "start": "node index.js"
9     },
10    "dependencies": {
11      "express": "^4.18.2",
12      "mongoose": "^7.6.0"
13    }
14 }
```

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

The screenshot shows the Visual Studio Code interface with the title bar "Student attendance system". The left sidebar displays the "EXPLORER" view, which lists files and folders: index.js, package-lock.json, package.json, student.js, models, data.json, and index.js again. The "OPEN EDITORS" tab at the top has tabs for index.js, package-lock.json, package.json, student.js, and data.json. The main editor area shows the content of package-lock.json:

```
{} package-lock.json > ...
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/ytDIdwUfLcLoKgdfJe1Fa8uw9seJYvd0mnP9jBwgWm69HkOixX6My9yvGf1MbhcS6lVmrihy4jm2g==",
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-W609buLVRVme693XkfzHeIV6nJGGz98uCPfeX11ELMLXVeKYZ9m15fAMsaUPBHylGFsVRcMsCksQRrZV9BYA==",
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-CijJvcRtIgzaHCYXw7dqENMRjhGzLYK05Mj9oyktqV8uVT8fd2BF0B7StuwBE3kj2Z+4LyPmfwIxgwLAIA=="

```

The screenshot shows the Visual Studio Code interface with the title bar "Student attendance system". The left sidebar displays the "EXPLORER" view, which lists files and folders: Welcome, index.js, package.json, student.js, models, data.json, index.js, package-lock.json, and package.json. The "OPEN EDITORS" tab at the top has tabs for Welcome, index.js, package.json, student.js, and data.json. The main editor area shows the content of student.js:

```
models > JS student.js > ...
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 );

```

## **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

Home Workspaces API Network

WorkspaceForTestingAPI By Sachin Vishwakarma

Get / GET

GET http://localhost:3000/attendance

Body JSON

```
1
2 {
  "studentName": "arivu",
  "rollNumber": "EEE007",
  "department": "EEE",
  "semester": 6,
  "date": "2024-02-01",
  "attendanceStatus": "Present"
}
```

200 OK 187 ms 1.66 KB Save Response

Runner Start Proxy Cookies Vault Trash

The screenshot shows a Postman collection named 'Postman Test Collection' under the workspace 'WorkspaceForTestingAPI'. It contains a single GET request to 'http://localhost:3000/attendance'. The request body is a JSON object with student details. The response is a 200 OK status with a response time of 187 ms and a size of 1.66 KB. The response body contains two student records, each with an \_id, studentName, rollNumber, department, semester, date, and attendanceStatus.

Home Workspaces API Network

WorkspaceForTestingAPI By Sachin Vishwakarma

POST / GET

POST http://localhost:3000/attendance

Body JSON

```
1
2 {
  "studentName": "arivu",
  "rollNumber": "EEE007",
  "department": "EEE",
  "semester": 6,
  "date": "2024-02-01",
  "attendanceStatus": "Present"
}
```

200 OK 189 ms 417 B Save Response

Runner Start Proxy Cookies Vault Trash

This screenshot is identical to the one above, showing a POST request to 'http://localhost:3000/attendance' instead of a GET request. The response is a 200 OK status with a response time of 189 ms and a size of 417 B. The response body contains a single student record with an \_id, studentName, rollNumber, department, semester, date, and attendanceStatus.

MongoDB Compass - Arivanantham/attendanceDB.students

Connections Edit View Collection Help

Compass

My Queries

Data Modeling

CONNECTIONS (1)

Search connections

Arivanantham

admin

attendanceDB

students

config

local

startup.log

attendanceDB mongosh:Arivanantham admin students config startup.log + Open MongoDB shell

Arivanantham > attendanceDB > students

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,_l_date_l	REGULAR	36.9 kB	0 (since Mon Jan 05 2026)	UNIQUE COMPOUND	READY

