

Lesson 05 Demo 03

Automating Boilerplate Code generation with GitHub Copilot

Objective: To demonstrate the automation of boilerplate code generation in a college management system using GitHub Copilot, specifically focusing on creating standard API endpoints

Tools required: Visual Studio Code, node js, json-server, and GitHub Copilot

Prerequisites: Lesson 05 Demos 01 and 02

Steps to be followed:

1. Input a specific endpoint creation prompt
2. Based upon prompt message whatever code generate follow all these steps and make the project ready.
3. Test the application.

Note: The generative AI tool used in this exercise can produce varied outputs even when presented with similar prompts. Thus, you may get different outputs for the same prompt.

Step 1: Input a specific endpoint creation prompt

1.1 First create the folder with name as **react-demo-api-app**

1.2 Open this folder in vs code.

1.3 Open Visual Studio and provide the following prompt to GitHub Copilot:


“The project is based on React JS using axios module to interact with emp.json file which is running using json-sever module and provide the end point. All the REST endpoints follow camelCasing for the URL paths. The emp.json file is as follows: departments with attribute [id, departmentName] and students with columns [id, StudentName, departmentId]. Create an API store and retrieve department and student details using react js. Please provide these provide all steps in details.”

/help What can you do?

The project is based on React JS using axios module to interact with emp.json file which is running using json-sever module and provide the end point. All the REST endpoints follow camelCasing for the URL paths. The emp.json file is as follows: departments with attribute [id, departmentName] and students with columns [id, StudentName, departmentId]. Create an API store and retrieve department and student details using react js. Please provide these provide all steps in details.

@ 📎 🗣️

The project is based on React JS using axios module to interact with emp.json file which is running using json-sever module and provide the end point. All the REST endpoints follow camelCasing for the URL paths. The emp.json file is as follows: departments with attribute [id, departmentName] and students with columns [id, StudentName, departmentId]. Create an API store and retrieve department and student details using react js. Please provide these provide all steps in details.

 GitHub Copilot

Sure, I'll guide you through the steps to create a React JS application that interacts with a **json-server** to store and retrieve department and student details using axios.

Step 1: Set Up the Project

1. **Create a new React project:** Open your terminal and run:

```
npx create-react-app react-demo-api-app
cd react-demo-api-app
```

2. **Install axios and json-server:**

```
npm install axios json-server
```

Step 2: Set Up the JSON Server

1. **Create **emp.json** file:** In the root of your project, create a file named **emp.json** with the following content:

```
{
```

Step 2. Based upon prompt message whatever code generate follow all these steps and make the project ready.

2.1 Set Up the Project

1. Initialize a new React project:

Note : We will create the react js project using new version with vite framework.

```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_05 - with MERN Using Generative AI\React JS Project using Generative AI>npm create vite@latest react-demo-api-app --template react
```

npm create vite@latest react-demo-api-app --template react

```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_05 - with MERN Using Generative AI>npm create vite@latest react-demo-api-app --template react
Need to install the following packages:
  create-vite@6.3.1
Ok to proceed? (y)
```

```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_05 - with MERN Using Generative AI>npm create vite@latest react-demo-api-app --template react
Need to install the following packages:
  create-vite@6.3.1
Ok to proceed? (y) y
|
* Select a framework:
|   Vanilla
|   Vue
|   > React
|   Preact
|   Lit
|   Svelte
|   Solid
|   Qwik
|   Angular
|   Others
|
```

```
Ok to proceed? (y) y
|
o Select a framework:
|   React
|
* Select a variant:
|   TypeScript
|   TypeScript + SWC
|   > JavaScript
|   JavaScript + SWC
|   React Router v7 ↗
|
—
```

After project created move inside a project directory

```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lo
tive AI>cd react-demo-api-app
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lo
tive AI\react-demo-api-app>
```

and run the command as **npm install**

it helps to install required react and vite framework dependencies

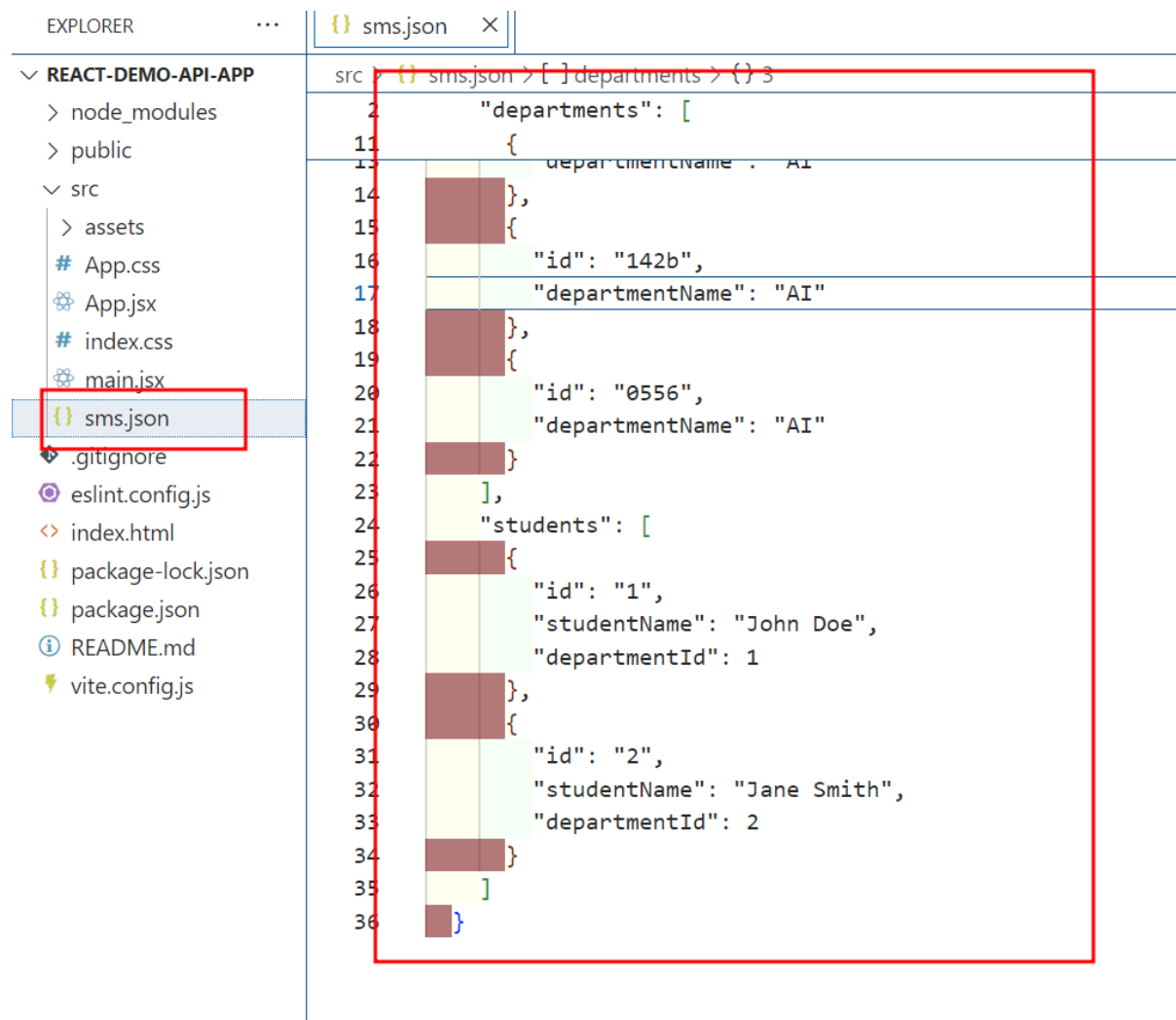
2.3 Now need to installed required external dependencies.

npm install axios json-serve

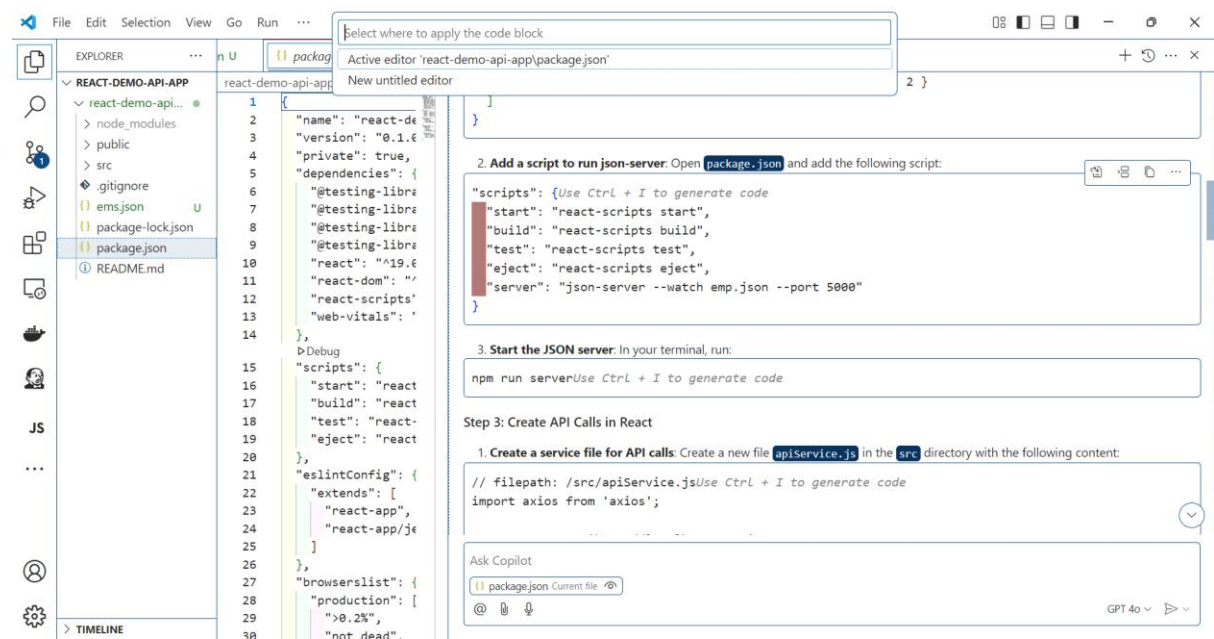
2.4 Now we will create the **sms.json** file

```
{
  "departments": [
    { "id": 1, "departmentName": "Computer Science" },
    { "id": 2, "departmentName": "Mathematics" }
  ],
  "students": [
    { "id": 1, "studentName": "John Doe", "departmentId": 1 },
    { "id": 2, "studentName": "Jane Smith", "departmentId": 2 }
  ]
}
```

Apply this sample file in editor.



2.5 apply generate package.json sever code to existing package.json code



2.6 Start the JSON server: In your terminal, run:

```
1 {
2   "name": "react-demo-api-app",
3   "private": true,
4   "version": "0.0.0",
5   "type": "module",
6   "scripts": {
7     "dev": "vite",
8     "build": "vite build",
9     "lint": "eslint .",
10    "preview": "vite preview",
11    "serve": "json-server --watch sms.json --port 5000"
12  },
13  "dependencies": {
14    "react": "^19.0.0",
15    "react-dom": "^19.0.0"
16  },
17  "devDependencies": {
18    "@eslint/js": "^9.21.0",
19    "@types/react": "^19.0.10",
20    "@types/react-dom": "^19.0.4",
21    "@vitejs/plugin-react": "^4.3.4",
22    "eslint": "^9.21.0",
23    "eslint-plugin-react-hooks": "^5.1.0",
24    "eslint-plugin-react-refresh": "^0.4.19",
25    "globals": "^15.15.0",
26    "vite": "^6.2.0"
27  }
28 }
```

```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_05 - with MERN Using Generative AI\React JS Project using Generative AI\react-demo-api-app>npm run serve
> react-demo-api-app@0.0.0 serve
> json-server --watch sms.json --port 5000

--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :5000
Press CTRL-C to stop
Watching sms.json...

(. ^ o ^.)

Index:
http://localhost:5000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:5000/departments
http://localhost:5000/students
```

2.7 create API call in react js.

apiService.js

// filepath: /src/apiService.js

```

import axios from 'axios';

const API_URL = 'http://localhost:5000';

export const getDepartments = async () => {
  const response = await axios.get(`${API_URL}/departments`);
  return response.data;
};

export const getStudents = async () => {
  const response = await axios.get(`${API_URL}/students`);
  return response.data;
};

export const addDepartment = async (department) => {
  const response = await axios.post(`${API_URL}/departments`, department);
  return response.data;
};

export const addStudent = async (student) => {
  const response = await axios.post(`${API_URL}/students`, student);
  return response.data;
};

```

Apply this code to editor and save with file name as **apiService.js** inside src folder.

2.8 Create Components to Display Data

Create a Departments component: Create a new file **Departments.jsx** in the src directory with the following content:

Department.jsx

```

// filepath: /src/Departments.jsx

import React, { useEffect, useState } from 'react';
import { getDepartments } from './apiService';

const Departments = () => {
  const [departments, setDepartments] = useState([]);

  useEffect(() => {
    const fetchDepartments = async () => {
      const data = await getDepartments();

```



```

    setDepartments(data);
  };
  fetchDepartments();
}, []);

return (
  <div>
    <h2>Departments</h2>
    <ul>
      {departments.map(department => (
        <li key={department.id}>{department.departmentName}</li>
      ))}
    </ul>
  </div>
);
};

export default Departments;

```

2.9 Create a Students component: Create a new file Students.js in the src directory with the following content:

Students.jsx

```

// filepath: /src/Students.jsx
import React, { useEffect, useState } from 'react';
import { getStudents } from './apiService';

const Students = () => {
  const [students, setStudents] = useState([]);

  useEffect(() => {
    const fetchStudents = async () => {
      const data = await getStudents();
      setStudents(data);
    };
    fetchStudents();
  }, []);

  return (
    <div>
      <h2>Students</h2>
      <ul>
        {students.map(student => (
          <li key={student.id}>{student.studentName}</li>

```

```

    )}
  </ul>
</div>
);
};

export default Students;

```

2.10 Integrate Components in App

Update App.jsx: Open src/App.jsx and update it to include the new components:

App.jsx

```

// filepath: /src/App.jsx

import React from 'react';
import Departments from './Departments';
import Students from './Students';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>React Demo API App</h1>
        <Departments />
        <Students />
      </header>
    </div>
  );
}

export default App;

```

2.11 : If you need to add Add operation on Student and Department ask for the prompt message.

“please provide the code for add Student and deparment must be react js component”

2.12 **Create AddDepartment.jsx:** Create a new file AddDepartment.jsx in the src directory with the following content:

AddDepartmentx.js

```

// filepath: /src/AddDepartmentx.js

```

```

import React, { useState } from 'react';
import { addDepartment } from './apiService';

const AddDepartment = () => {
  const [departmentName, setDepartmentName] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    const newDepartment = { departmentName };
    await addDepartment(newDepartment);
    setDepartmentName("");
  };

  return (
    <div>
      <h2>Add Department</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          value={departmentName}
          onChange={(e) => setDepartmentName(e.target.value)}
          placeholder="Department Name"
          required
        />
        <button type="submit">Add Department</button>
      </form>
    </div>
  );
};

export default AddDepartment;

```

2.13 Create AddStudent Component

Create AddStudent.jsx: Create a new file AddStudent.js in the src directory with the following content:

```

// filepath: /src/AddStudent.jsx

import React, { useState, useEffect } from 'react';
import { addStudent, getDepartments } from './apiService';

const AddStudent = () => {
  const [studentName, setStudentName] = useState("");
  const [departmentId, setDepartmentId] = useState("");
  const [departments, setDepartments] = useState([]);

```

```

useEffect(() => {
  const fetchDepartments = async () => {
    const data = await getDepartments();
    setDepartments(data);
  };
  fetchDepartments();
}, []);

const handleSubmit = async (e) => {
  e.preventDefault();
  const newStudent = { studentName, departmentId: parseInt(departmentId) };
  await addStudent(newStudent);
  setStudentName("");
  setDepartmentId("");
};

return (
  <div>
    <h2>Add Student</h2>
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        value={studentName}
        onChange={(e) => setStudentName(e.target.value)}
        placeholder="Student Name"
        required
      />
      <select
        value={departmentId}
        onChange={(e) => setDepartmentId(e.target.value)}
        required
      >
        <option value="">Select Department</option>
        {departments.map(department => (
          <option key={department.id} value={department.id}>
            {department.departmentName}
          </option>
        ))}
      </select>
      <button type="submit">Add Student</button>
    </form>
  </div>
);
};

```

```
export default AddStudent;
```

2.14 Integrate Components in App

Update [App.jsx](#): Open [App.jsx](#) and update it to include the new components:

```
// filepath: /src/Appx.js

import React from 'react';
import './App.css';
import Departments from './Departments';
import Students from './Students';
import AddDepartment from './AddDepartment';
import AddStudent from './AddStudent';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>React Demo API App</h1>
        <AddDepartment />
        <AddStudent />
        <Departments />
        <Students />
      </header>
    </div>
  );
}

export default App;
```

Step 3 : Run this application

npm run dev **(this command you need to run on another terminal)**

```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_05 - with MERN Using Generative AI\React JS Project using Generative AI\react-demo-api-app>npm run dev
> react-demo-api-app@0.0.0 dev
> vite

1:04:57 PM [vite] (client) Re-optimizing dependencies because lockfile has changed

VITE v6.2.0 ready in 714 ms
  → Local:   http://localhost:5173/
  → Network: use --host to expose
  → press h + enter to show help
```

You can see the output on browser as

The screenshot shows a web browser at localhost:5173 displaying the 'React Demo API App'. The app is enclosed in a red rectangular border and contains the following sections:

- React Demo API App** (Main Title)
- Add Department**
 - Input field: Department Name
 - Button: Add Department
- Add Student**
 - Input field: Student Name
 - Dropdown menu: Select Department
 - Button: Add Student
- Departments**
 - List of departments: Computer Science, Mathematics, AI, AI, AI
- Students**
 - List of students: John Doe, Jane Smith