**Lesson 09 Demo 04**

**Creating a Rest API Using Next.js**

**Objective:** To create a Next.js server-side component for a REST API that interacts with a file module for efficient data storage and retrieval, accessible across various technologies

**Tools required:** Node.JS and React.JS

**Prerequisites:** None

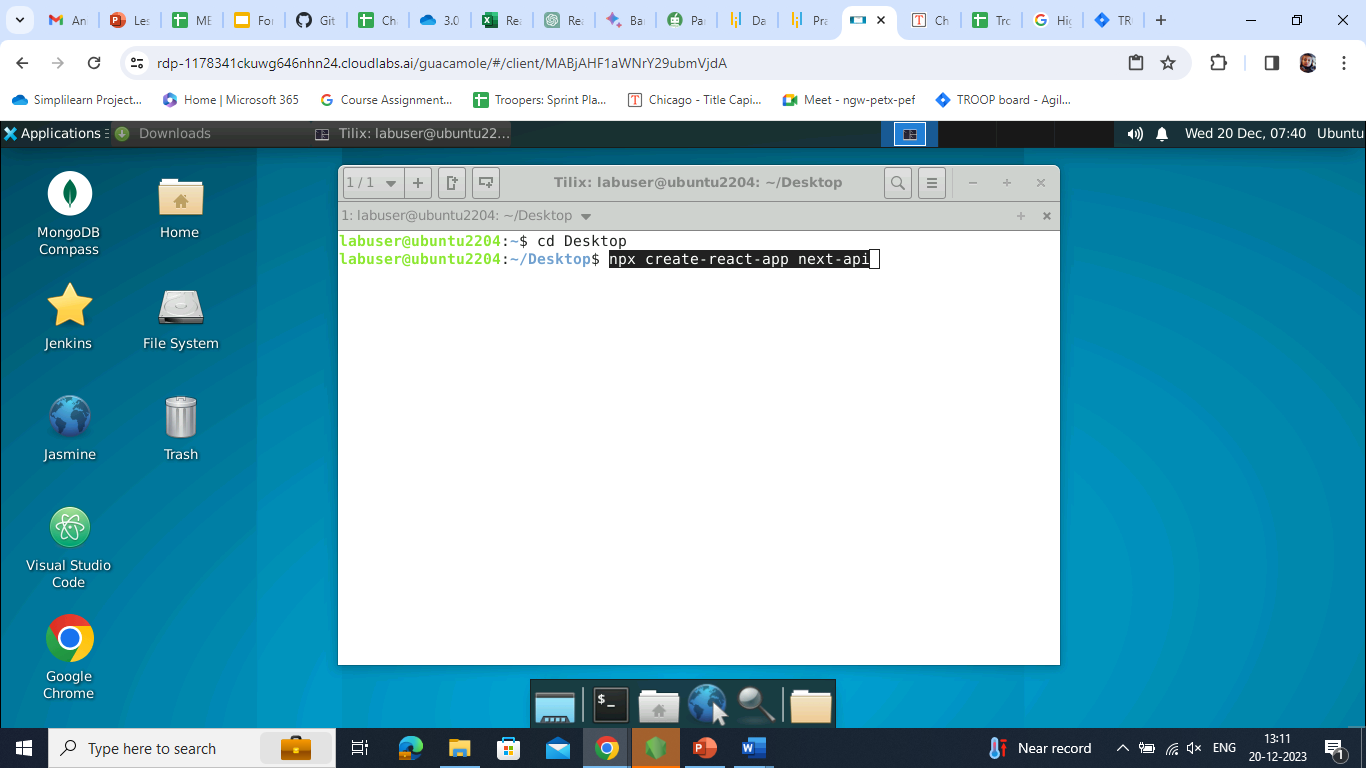
Steps to be followed:

1. Create and set up a Next.js project
2. Create a JSON file to add static data
3. Create a routing component
4. Run the application

**Step 1: Create and set up a Next.js project**

1. Open a terminal window to run the following command to create a **next-api** project:

**npx create-react-app** **next-api**

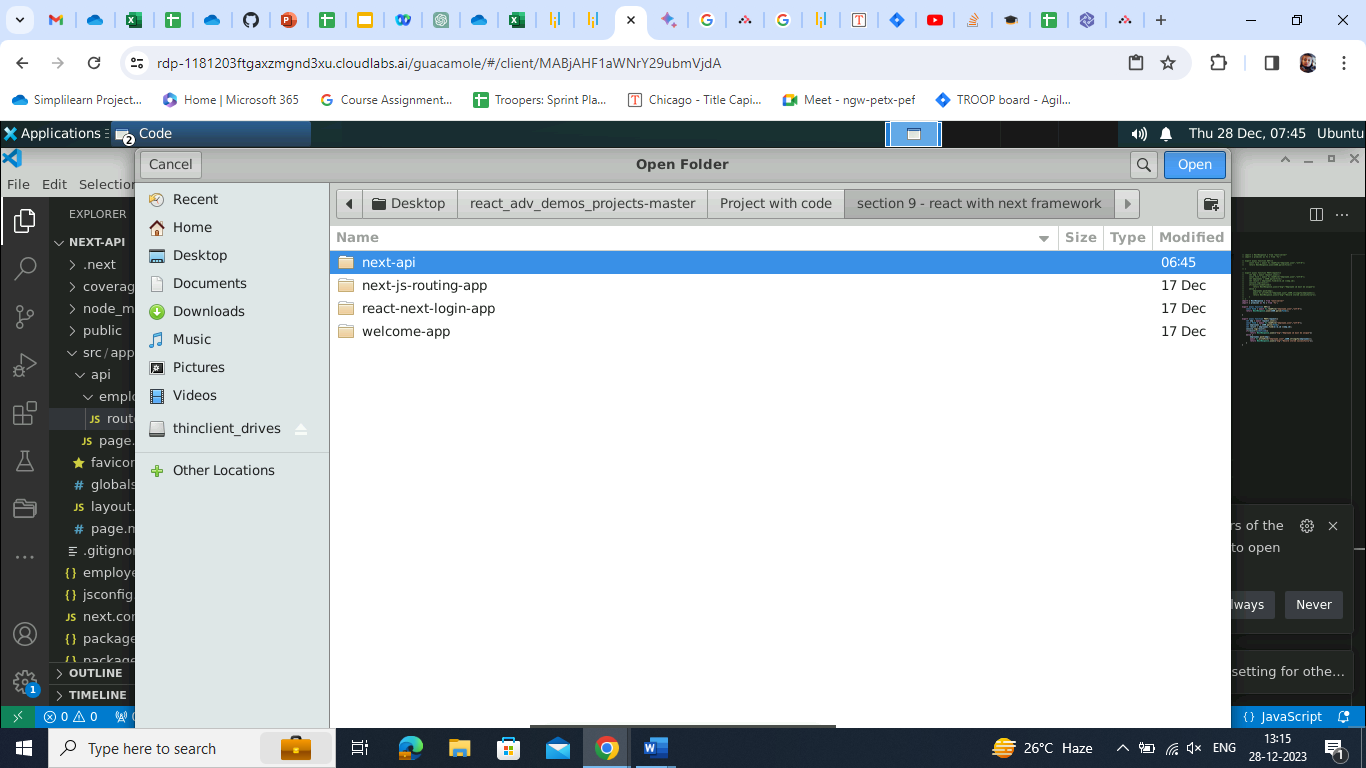
****

1. Open the created React application folder (**next-api**) in VS Code by clicking on **File** in the top left corner and selecting **Open Folder**

A screenshot of a computer

Description automatically generated

1. Select the **next-api** folder and click on **Open** button

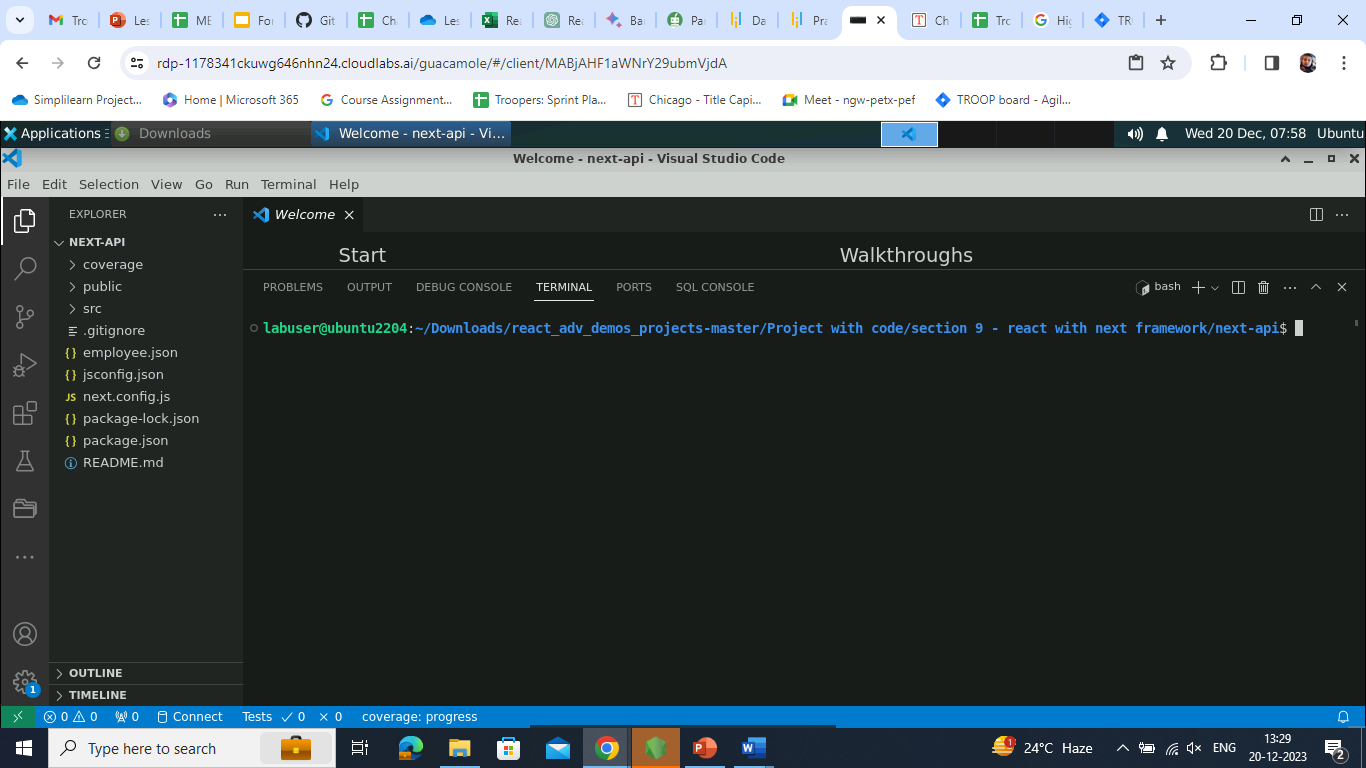


The folder structure appears as follows:

A screenshot of a computer

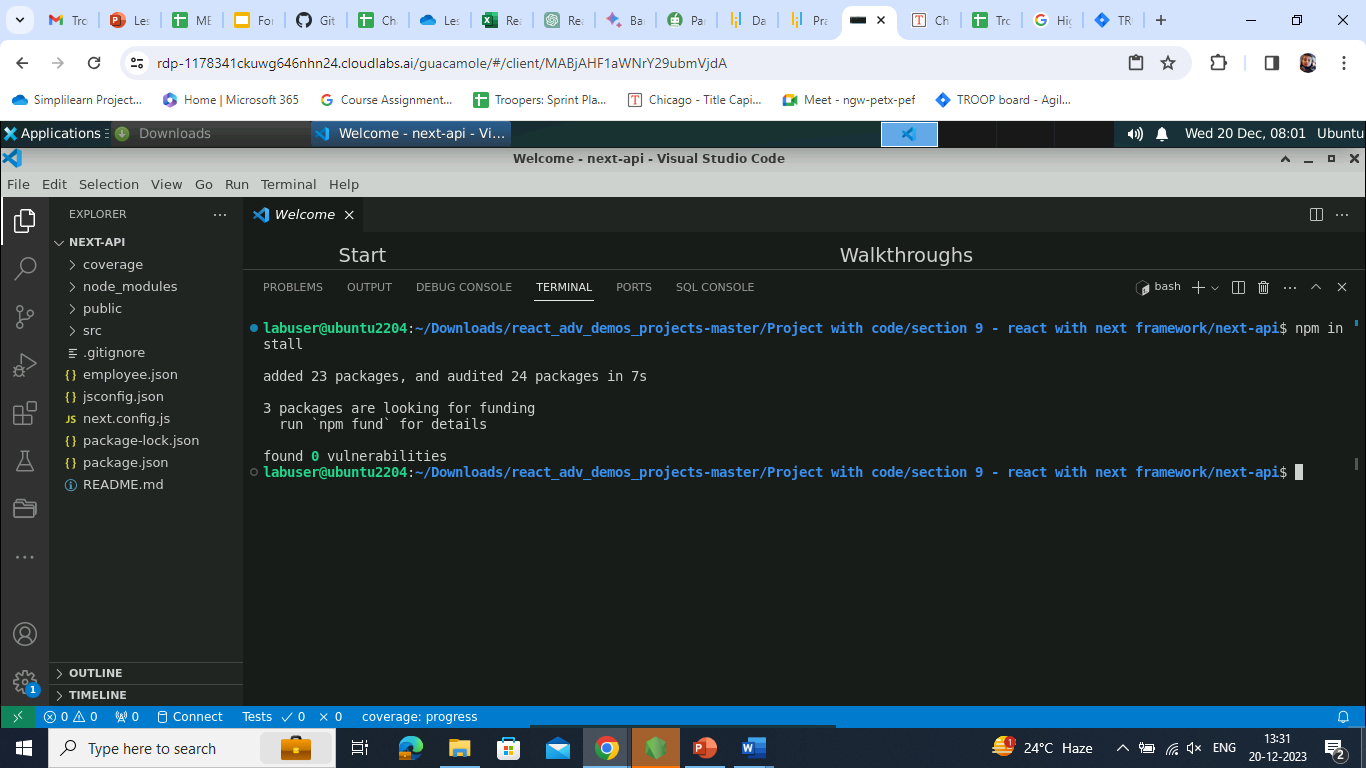
Description automatically generated

1. Open the **TERMINAL** window **inside the next-api** folder



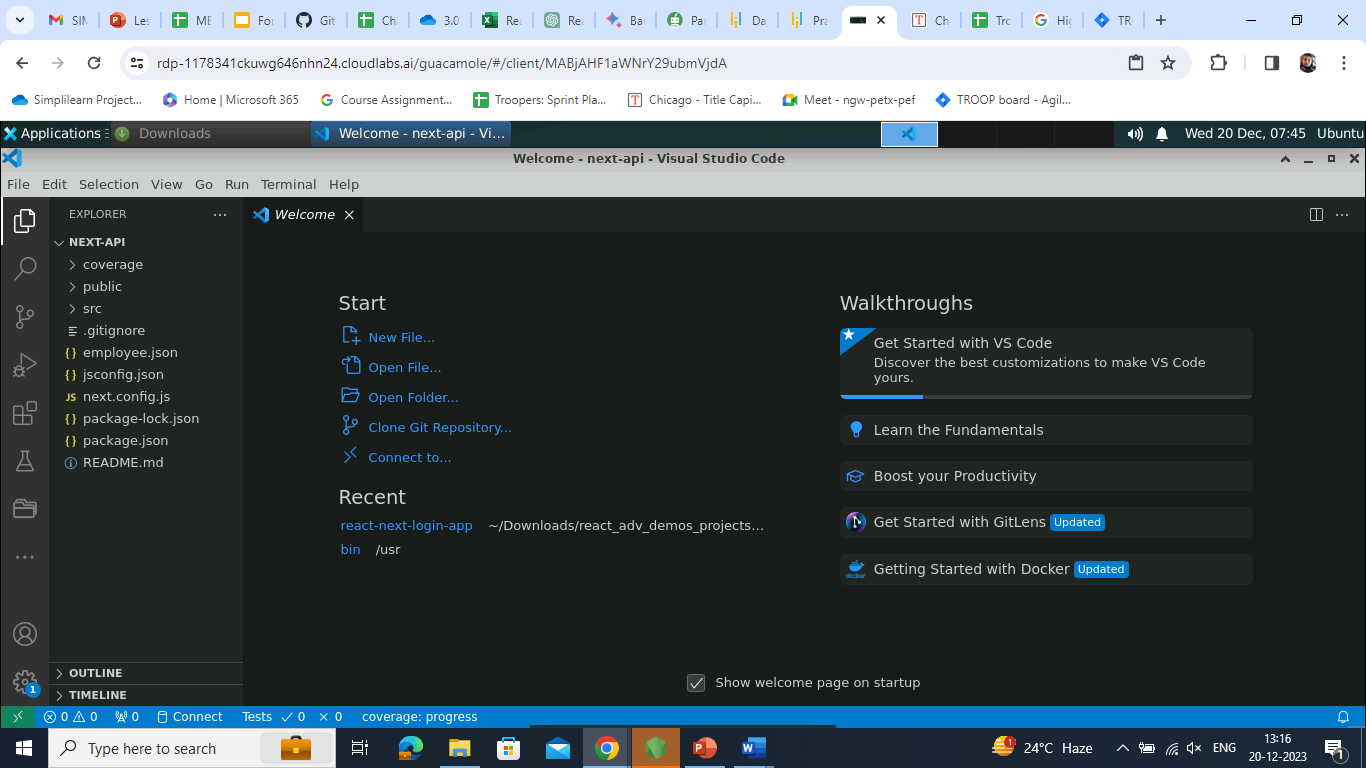
1. Run the following command to download the dependencies:

**npm install**

****

**Step 2: Create a JSON file** **to add static data**

1. Create an **employee.json** file



1. Enter the following code inside the **employee.json** file:

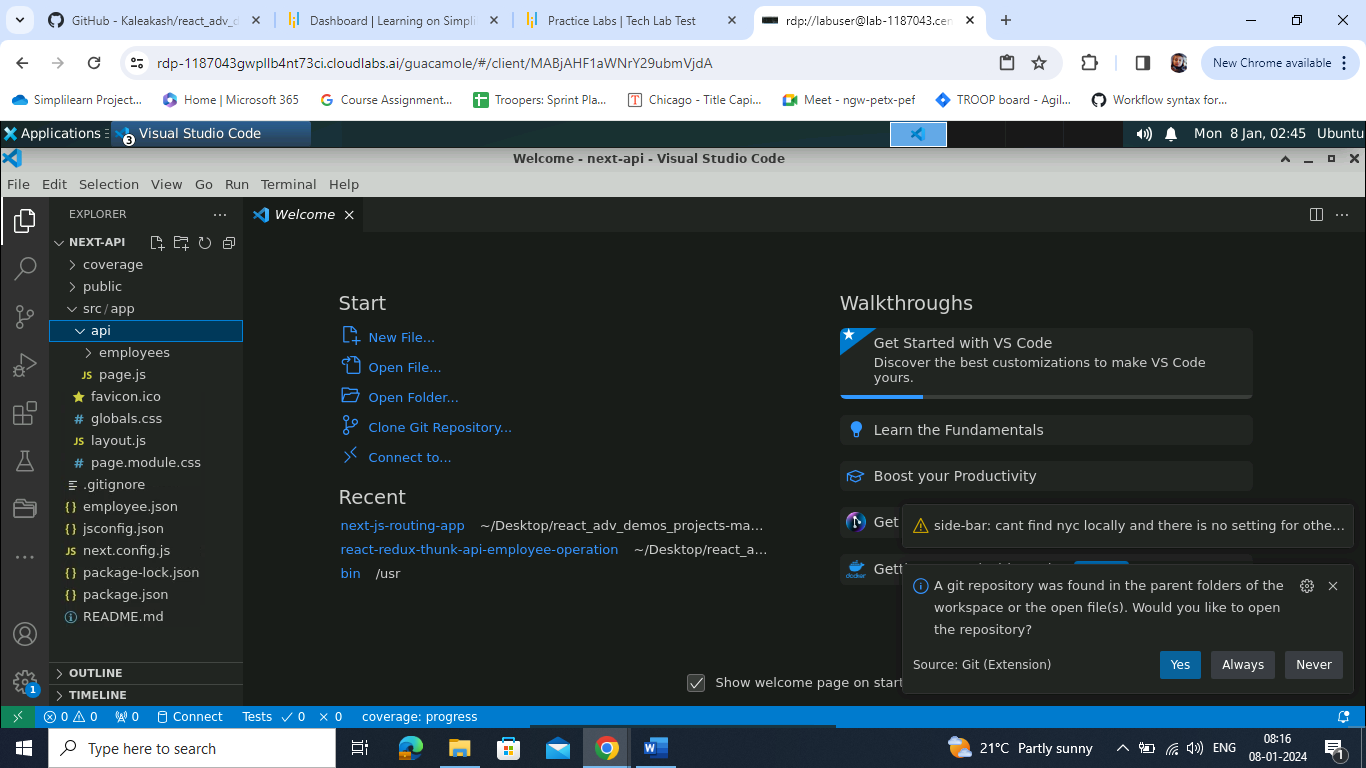
**[{"id":101,"name":"Ravi","salary":34000},{"id":102,"name":"Raju","salary":36000},{"id":103,"name":"Ramesh","salary":39000},{"id":104,"name":"Raj","salary":32000},{"id":111,"name":"Raju","salary":59000},{"id":112,"name":"Raju","salary":59000}]**

**A computer screen shot of a computer screen

Description automatically generated**

**Step 3: Create a routing component**

1. Create a folder and name it as **api**



1. Inside the **api** folder, create the **employees** subfolder that contains the **route.js** file

A computer screen shot of a computer screen

Description automatically generated

1. Write the following code in the **route.js** file:

**import { NextResponse } from "next/server"**

**import { promises as fs } from 'fs';**

**export async function GET(){**

**const file = await fs.readFile("employee.json","utf-8");**

**return NextResponse.json(JSON.parse(file));**

**}**

**export async function POST(request){**

**let emp = await request.json();**

**const file = await fs.readFile("employee.json","utf-8");**

**let employees = JSON.parse(file);**

**let result = employees.find(e=>e.id ==emp.id);**

**console.log(result);**

**if(result!=undefined){**

**return NextResponse.json({"msg":"Employee id must be unique"})**

**}else {**

**employees.push(emp);**

**await fs.writeFile("employee.json",JSON.stringify(employees));**

**return NextResponse.json({"msg":"Record stored successfully"});**

**}**

**}**

A computer screen shot of a computer screen

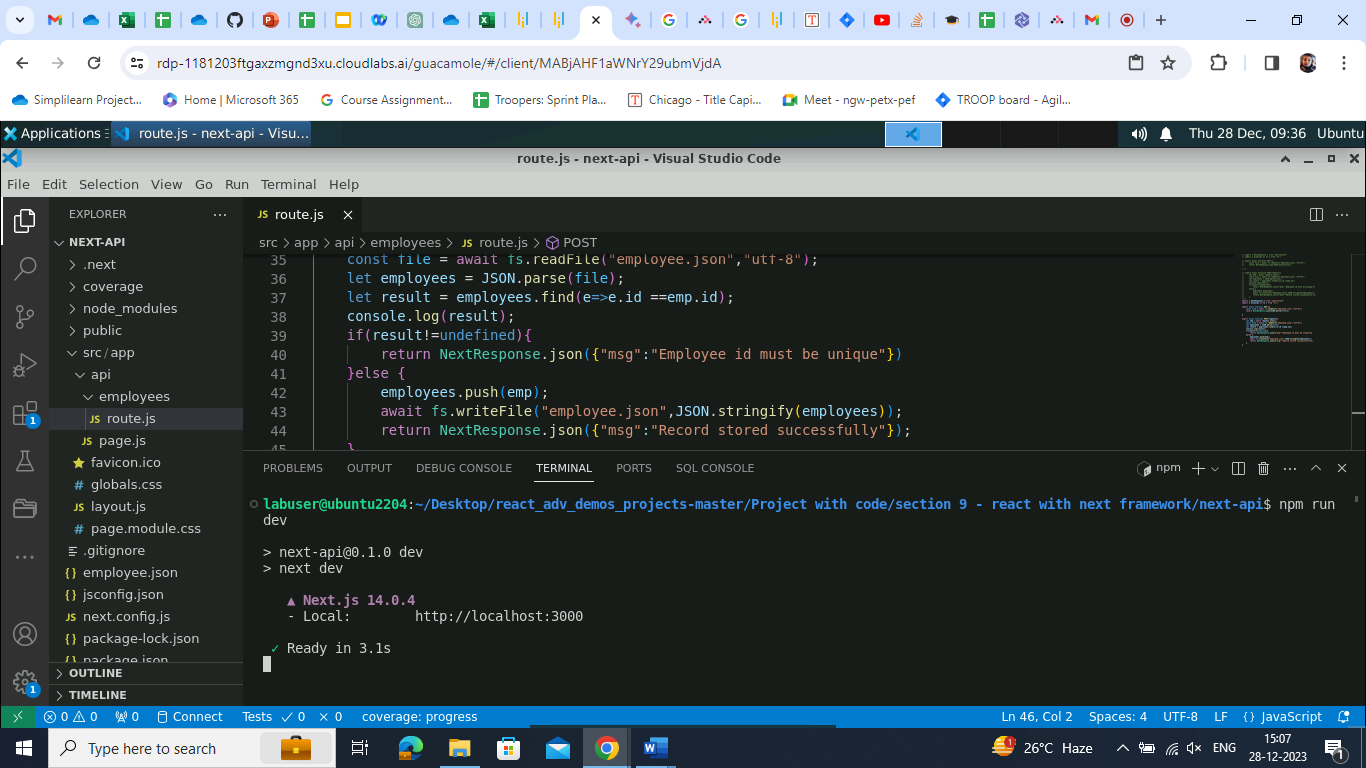
Description automatically generated

**Note**: The GET method reads the data from the **employee.json** file and produces the data in JSON format for the client. POST method consumes data from frontend technologies and stores the data in the **employee.json** file.

**Step 4: Run the application**

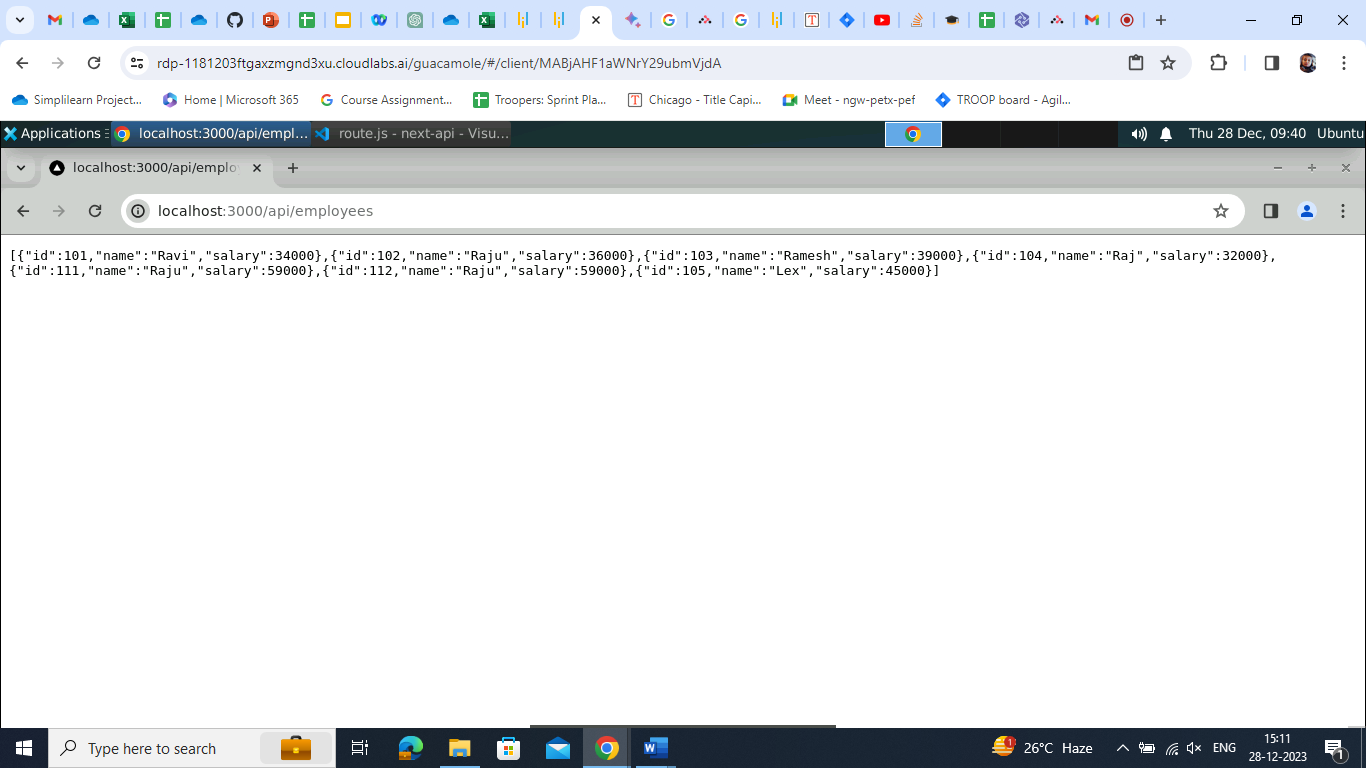
1. Open the **TERMINAL** inside the project and run the application with the following command:

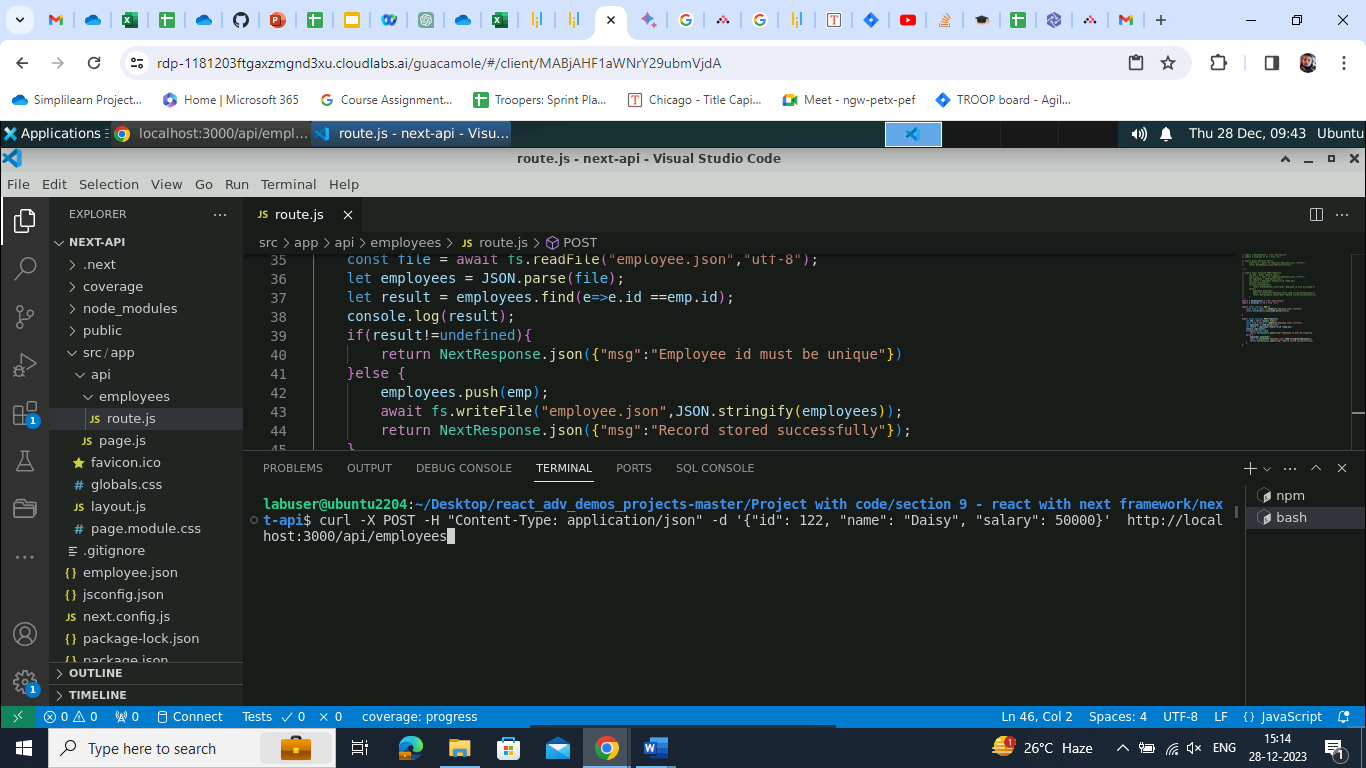
**npm run dev**

****

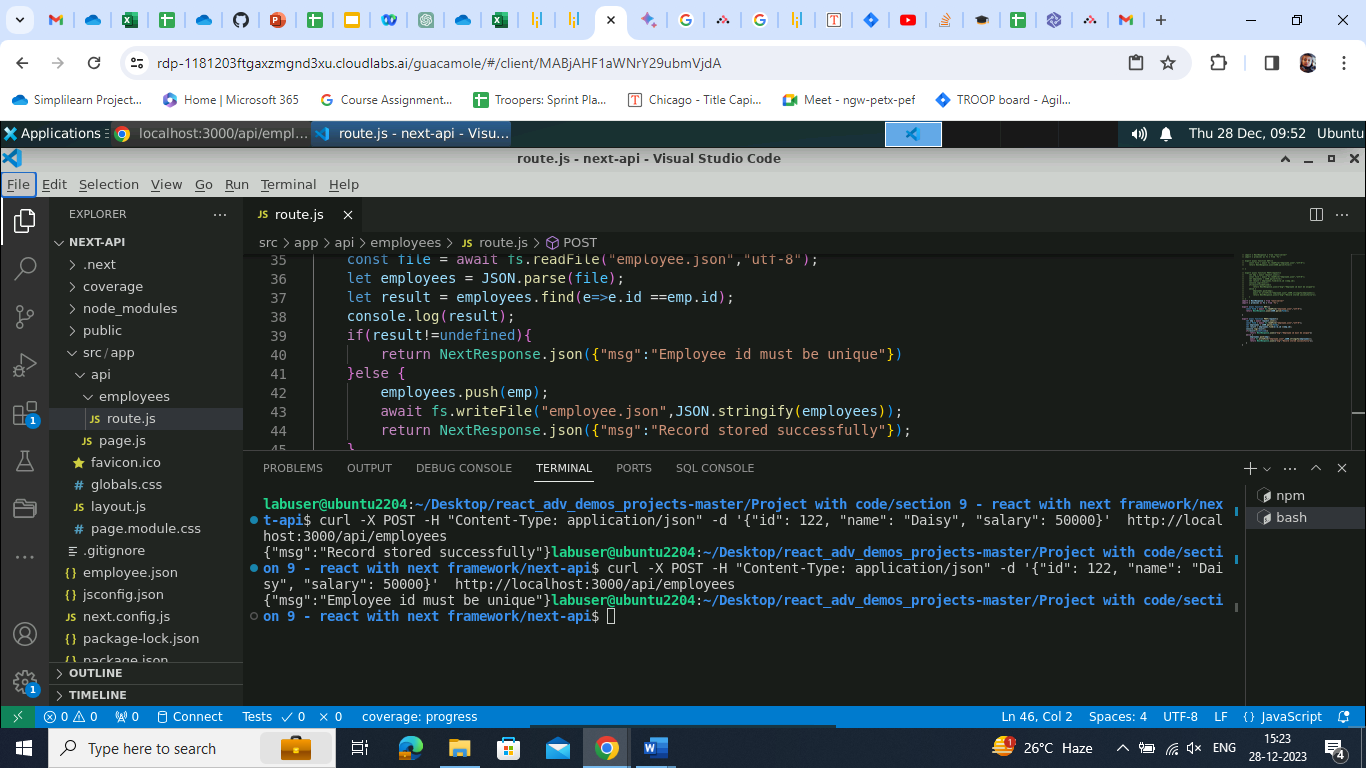
1. Access the GET method through the browser using the following URL:

[**http://localhost:3000/api/employees**](http://localhost:3000/api/employees)



1. Use the following curl command to test the POST method:  
     
   **curl -X POST -H "Content-Type: application/json" -d '{"id": 122, "name": "Daisy", "salary": 50000}'** [**http://localhost:3000/api/employees**](http://localhost:3000/api/employees) ****

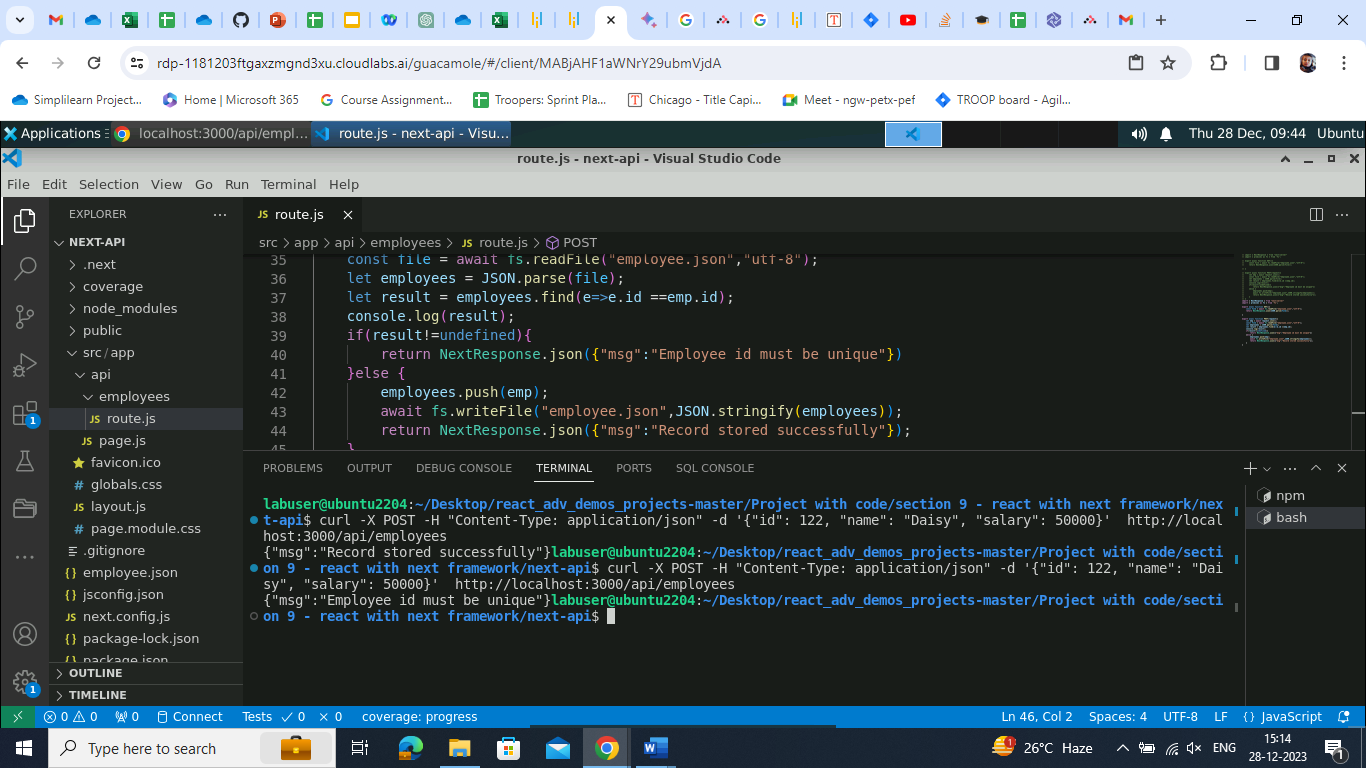
The output will display the **Record stored successfully**, as shown in the image below:



Given below is the recently added data on the web page:  
  
A computer screen with a white screen

Description automatically generated

1. Re-run the same command



It shows that the **Employee** **id must be unique.** Therefore, the record is not stored.

You have now created a Next.js server-side component for a REST API, seamlessly interacting with a file module for efficient data storage and retrieval, accessible across diverse technologies.