**Lesson 09 Demo 03**

**Next-api**

**Objective:** To demonstrate the next js to create rest api ie server side component which any other technologies can consume. These next api interact with file module to store and retrieve.

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

**Prerequisites:** HTML, CSS, JavaScript ES5/ES6, Basic React Concept

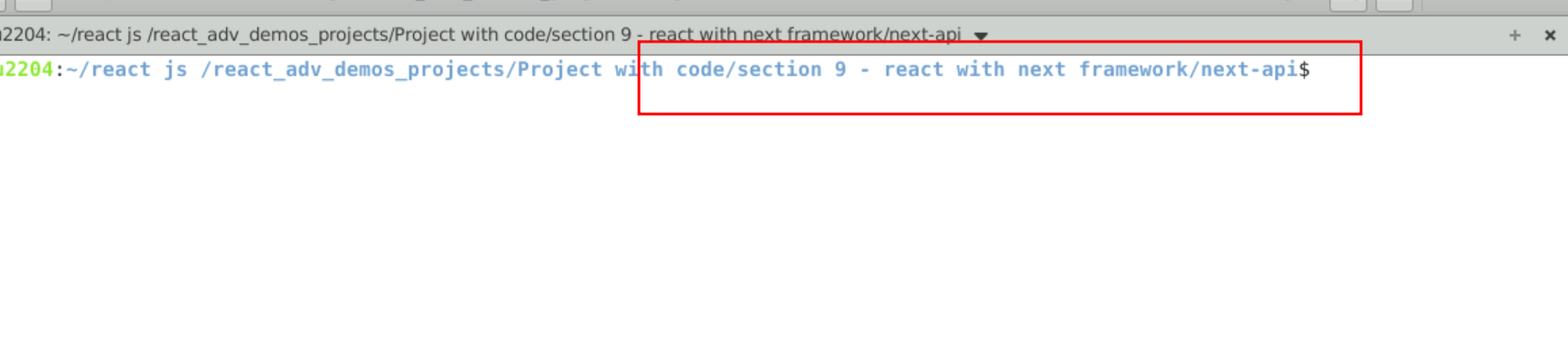
**Note** : All react js project already created with version 18.x with Sample App.js file

**Steps to be followed:**

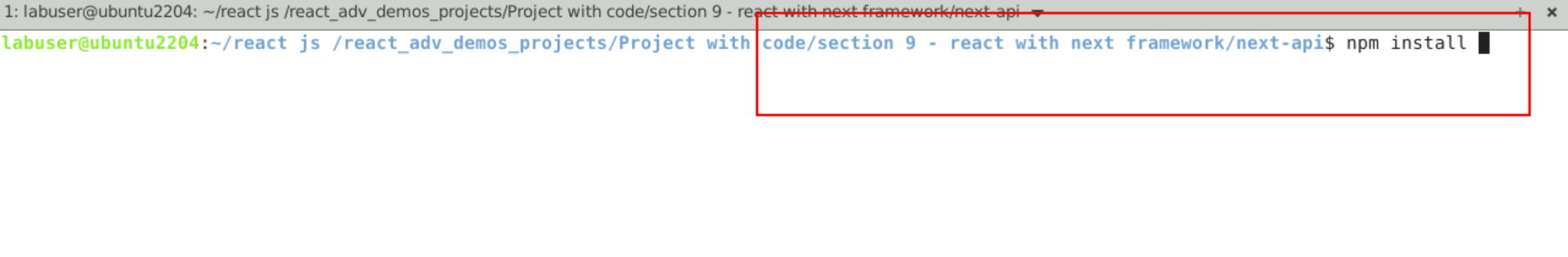
1. Set up for next js project
2. Create employee.json file
3. Create more than one component
4. Now we run the application.

**Step 1: Set up for next js project**

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



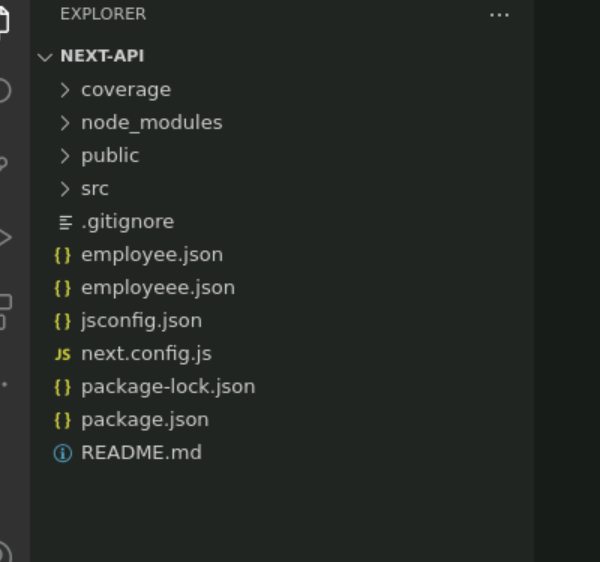
1.2 run npm install to download the dependencies.



* 1. Open the project in VS code

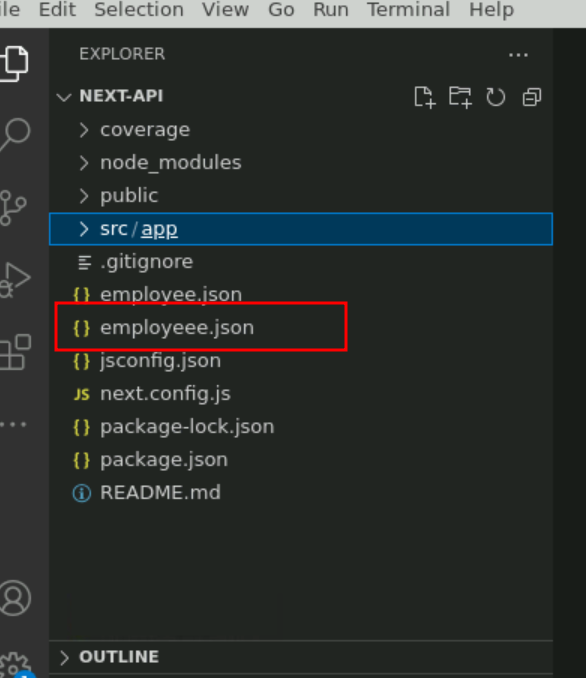
Note: short cut to open write **code .**

* 1. view the project structure



**Step 2 : Create employee.json file**

2.1 This file contains few static employee.json data.

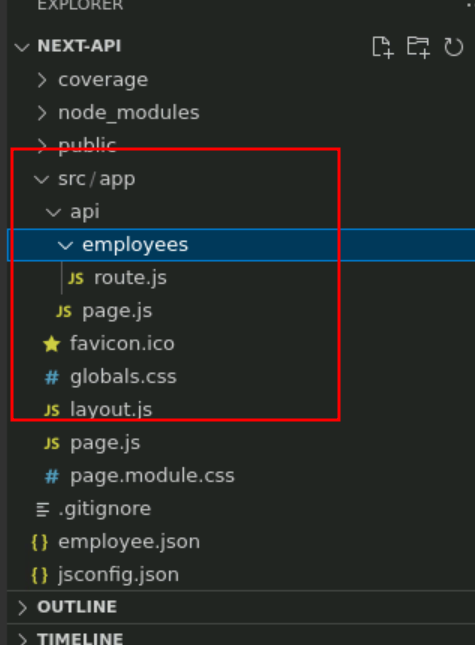
****

2.2 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}]

**Step 3: create more than one component**

3.1 Now create the folder as api and inside the folder create the employees folder which contains route.js file(file name must be route) which is responsible to provide Get, Post, Put and Delete http method behaviour.



3.2 route.js file part of employees folder which contains two method ie Get or post.

Get method is responsible to read the data form employee.json file and produce the data in json format to client. Post method is consume data from frontend technologies and store the data in employee.json file.

route.js

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"});

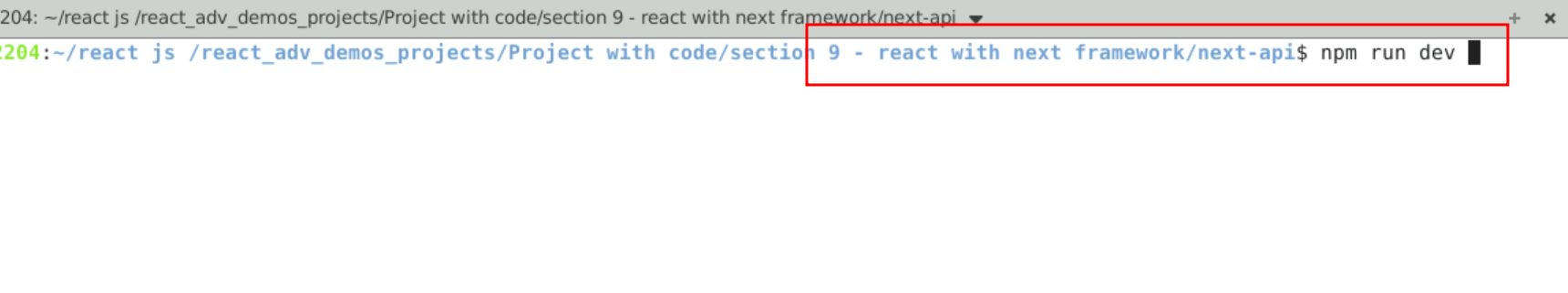
}

}

**Step 3 :Now we run the application.**

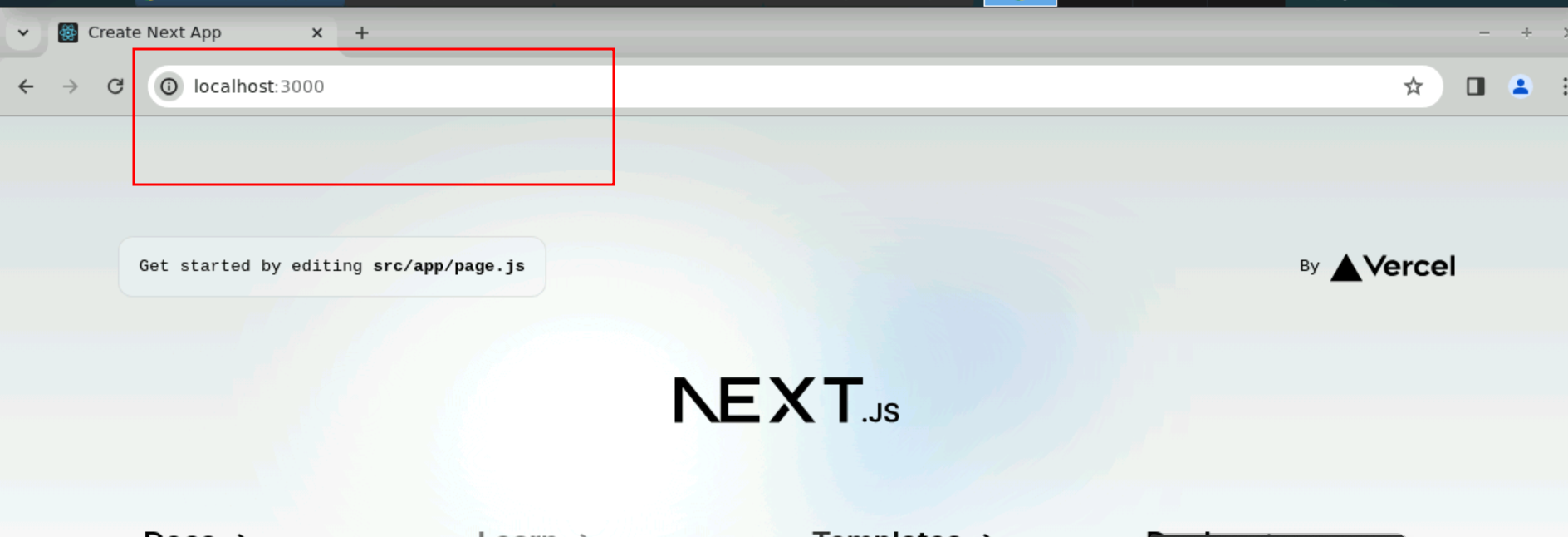
3.1 open the terminal inside the project and run the application we need to use the command as

**npm run dev**

****

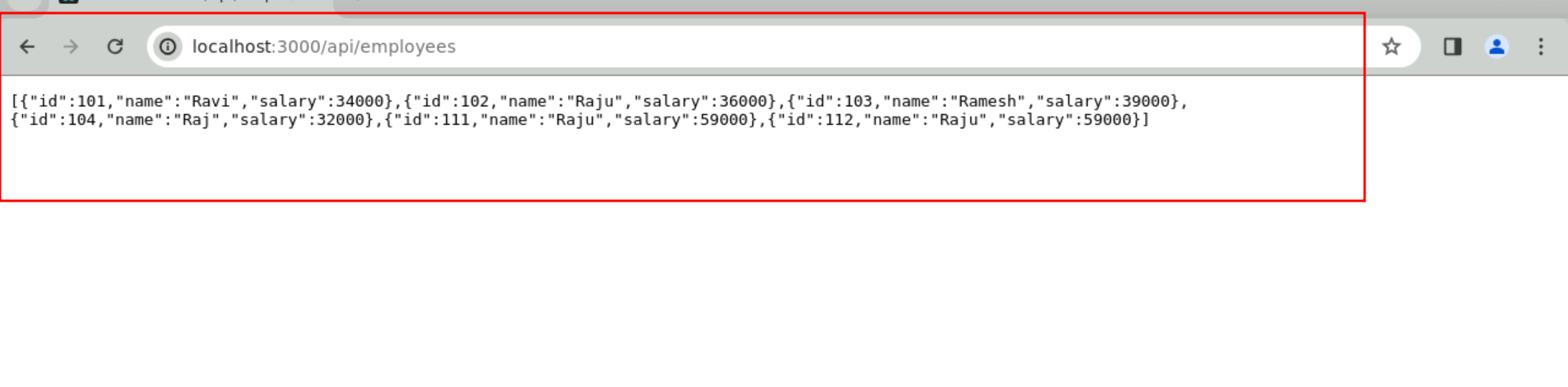
3.2 Now open the browser and write the URL as

<http://localhost:3000>



3.3 you can access the get method through browser

<http://localhost:3000/api/employees>

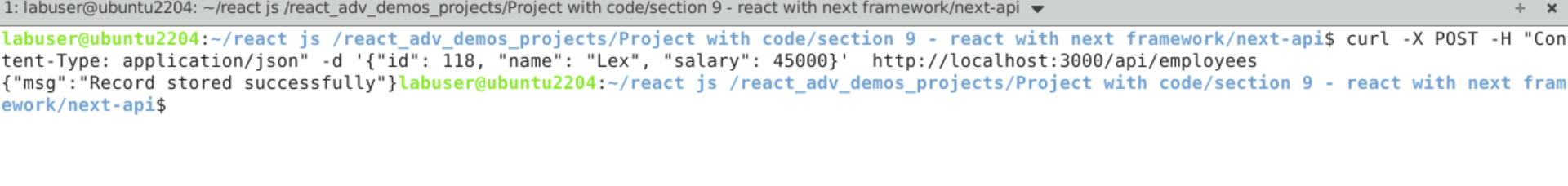


3.4 post method you can test using post man client or curl command

So here we are testing through curl command.

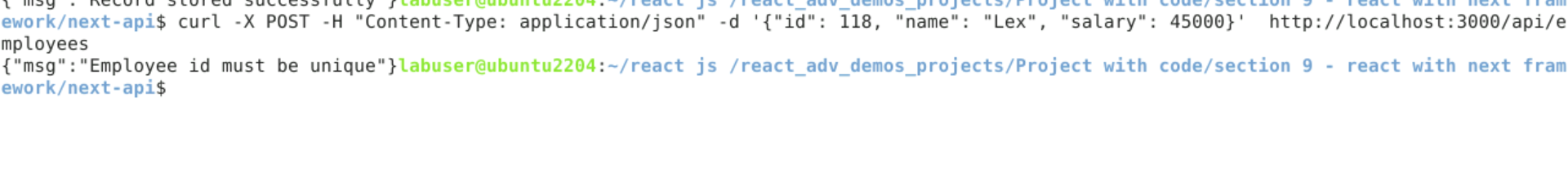
Open the command prompt and hit below URL

**curl -X POST -H "Content-Type: application/json" -d '{"id": 112, "name": "Lex", "salary": 45000}'** [**http://localhost:3000/api/employees**](http://localhost:3000/api/employees)



Here you can see the output as record stored successfully

Once again hit same command.



Id must be unique so record didn’t store.

3.5 Now you can test using get method through browser to view new records.

