**Lesson 07 Demo 03**

**Creating a React Redux Thunk API Employee Operation**

**Objective:** To create the React app with Redux using the Thunk middleware to store and retrieve employee details from a JSON file using Axios

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

**Prerequisites:** Redux DevTools

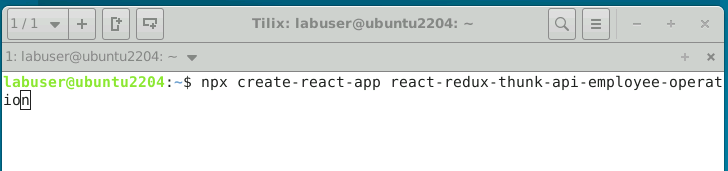
Steps to be followed:

1. Create and set up the React project
2. Create an employee.json file
3. Create actions, components, and reducers
4. Configure the store and Thunk details in the index.js file
5. Run the application
6. Debug the application using the Redux DevTools with the browser

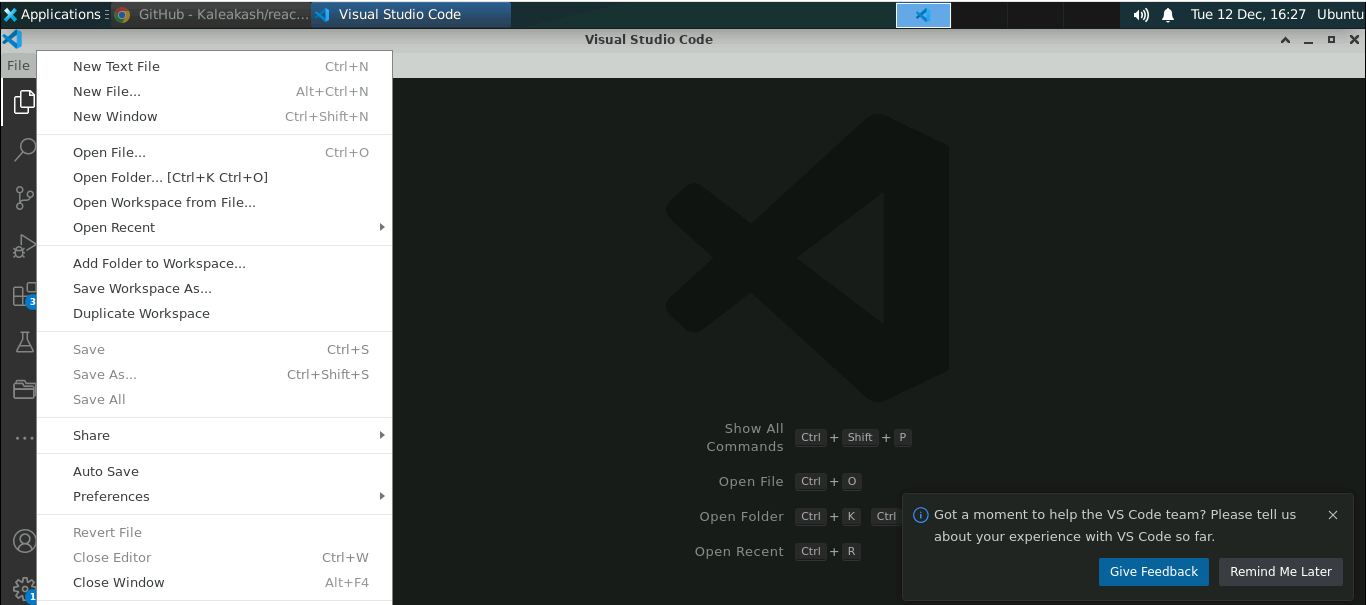
**Step 1: Create and set up the React project**

* 1. Open a terminal window and run the following command to create a React app:

**npx create-react-app react-redux-thunk-api-employee-operation**

****

* 1. Open the created React app folder (**react-redux-thunk-api-employee-operation**) in VS Code by clicking on **File** in the top left corner and selecting **Open Folder**

****

* 1. Click on the **Open** button

**A screenshot of a computer

Description automatically generated**

The project structure appears as follows:

A screenshot of a computer

Description automatically generated

* 1. Inside the project, open the **TERMINAL** and run the following command to install the required dependencies:

**npm install**

**A screenshot of a computer

Description automatically generated**

**Note**: This command helps you to install all the required dependencies mentioned in the **package.json** file in the local machine in the form of a **node\_module** folder.

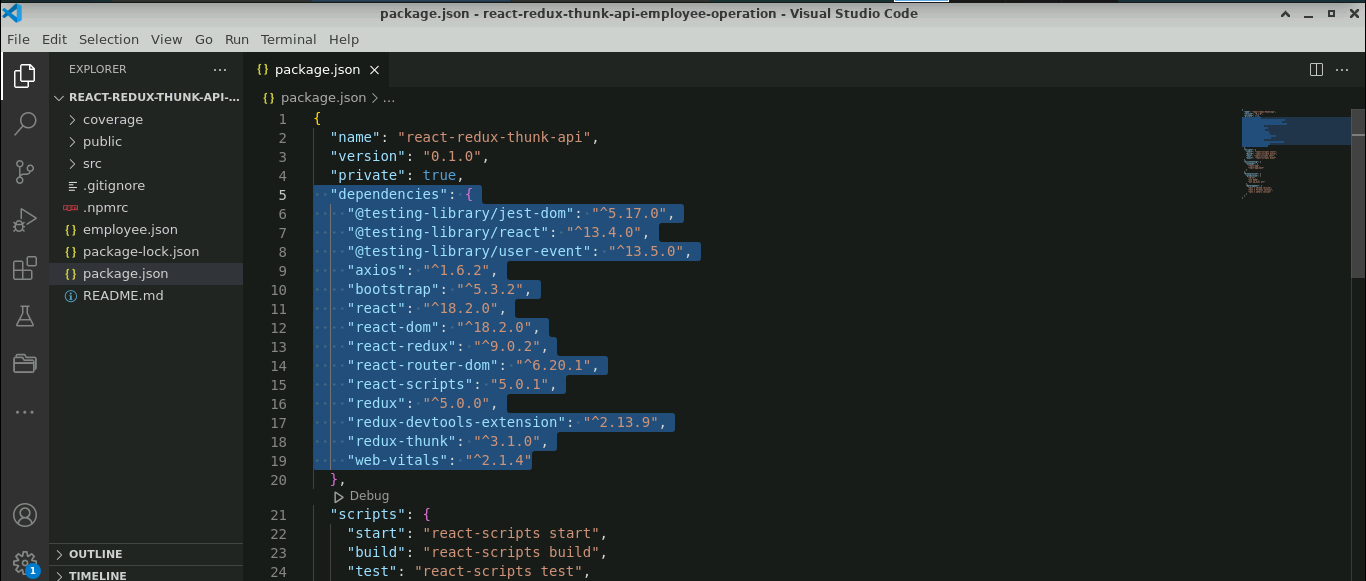
* 1. You may occasionally encounter errors relating to Redux and Redux DevTools extension versions. Run the following command to debug such errors:

**npm install - -force**

A screenshot of a computer program

Description automatically generated

* 1. Open the **package.json** file and view the external dependencies



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

1. Right-click on the **src** folder and select **New File**

**A screenshot of a computer

Description automatically generated**

1. Create a file named **employee.json**

**A screen shot of a computer

Description automatically generated**

* 1. Inside the **employee.json** file, enter the following code:

**{**

**"employees": [**

**{**

**"id": 100,**

**"name": "Ravi",**

**"salary": 12000**

**},**

**{**

**"id": 101,**

**"name": "Ramesh",**

**"salary": 14000**

**},**

**{**

**"id": 102,**

**"name": "Rajesh",**

**"salary": 16000**

**},**

**{**

**"id": 103,**

**"name": "Rajesh",**

**"salary": 45000**

**},**

**{**

**"id": 104,**

**"name": "Akash",**

**"salary": 45000**

**},**

**{**

**"id": "105",**

**"name": "Lokesh",**

**"salary": "56000"**

**},**

**{**

**"id": "106",**

**"name": "Raju",**

**"salary": "67000"**

**},**

**{**

**"id": "108",**

**"name": "Raju",**

**"salary": "45000"**

**},**

**{**

**"id": "110",**

**"name": "Raju",**

**"salary": "45000"**

**},**

**{**

**"id": "111",**

**"name": "Raju",**

**"salary": "45000"**

**},**

**{**

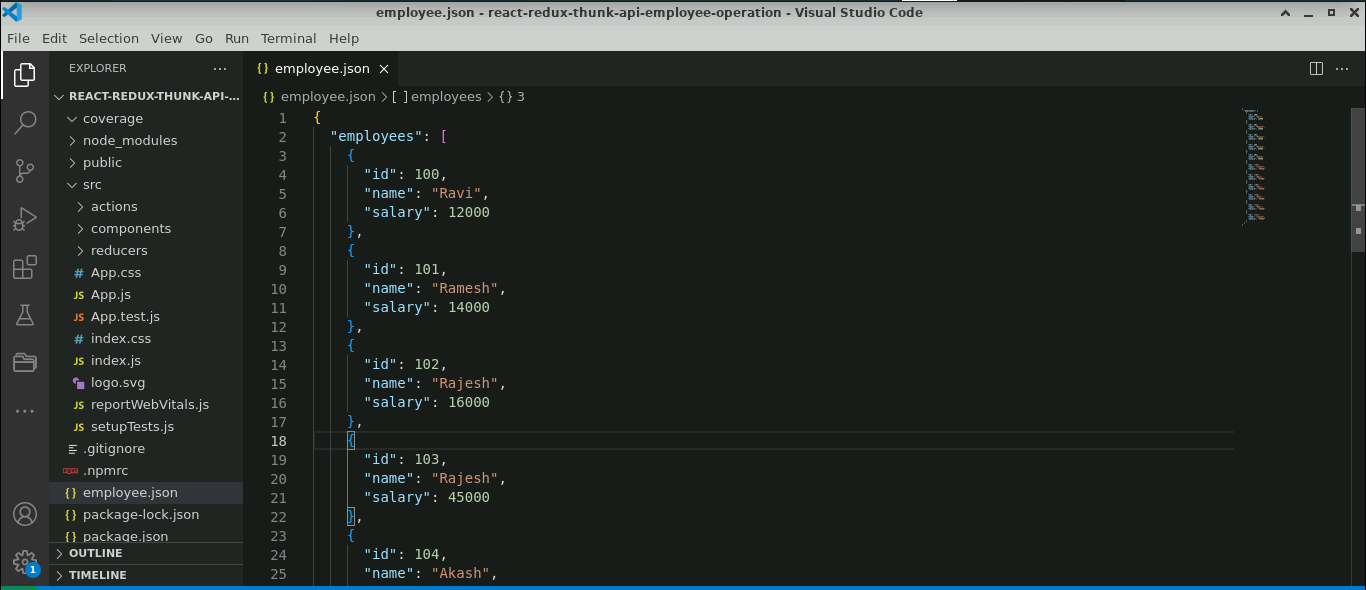
**"id": "112",**

**"name": "Lokesh",**

**"salary": "45000"**

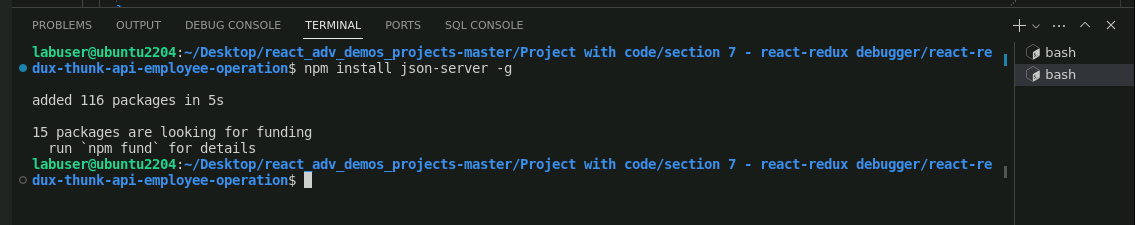
**}**

**]}**

****

* 1. Install the **json-server** module by executing the following command:

**npm install json-server -g**



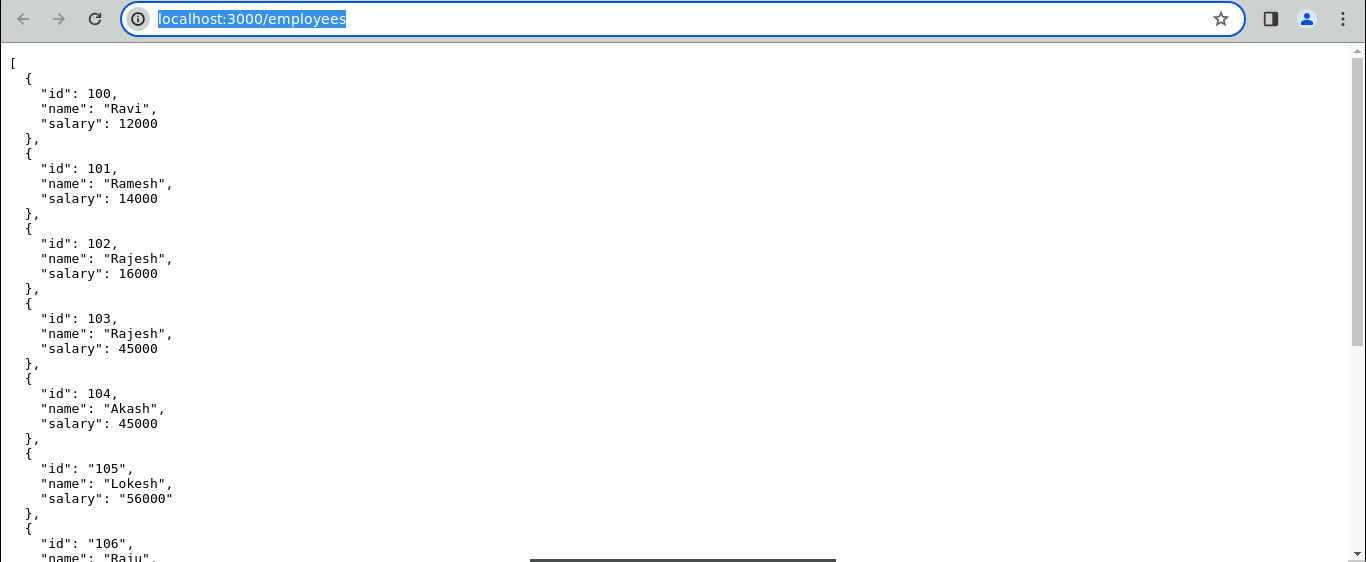
* 1. Execute the following command to start the JSON server:

**json-server employee.json**

**A screen shot of a computer

Description automatically generated**

* 1. Run <http://localhost:3000/employee> (refer to the above screenshot) on the browser and it shows the employees data



**Step 3: Create actions, components, and reducers**

1. Inside the **src** folders, create folder named **actions**, **components,** and **reducers**

**A black and white screen with a black x

Description automatically generated**

1. Inside the **actions** folder, create a file named **EmployeeActions.js** and enter the following code:

**import axios from "axios";**

**export const ADD\_EMPLOYEE="Add Employee details";**

**export const LOAD\_EMPLOYEE = "Load Employee details";**

**export const EMPLOYEE\_ERROR="Employee Error Generated"**

**export function addEmployee(data){**

**return function(dispatch){**

**axios.post("http://localhost:3000/employees", data.payload)**

**.then(function(response){**

**console.log(response)**

**dispatch({type:ADD\_EMPLOYEE, payload: response.data})**

**})**

**.catch(function(err){**

**dispatch({type:EMPLOYEE\_ERROR, payload: "There was an error."})**

**})**

**}**

**}**

**export function loadEmployee() {**

**return function(dispatch){**

**axios.get("http://localhost:3000/employees").then(function(response){**

**dispatch({type:LOAD\_EMPLOYEE, payload: response.data})**

**})**

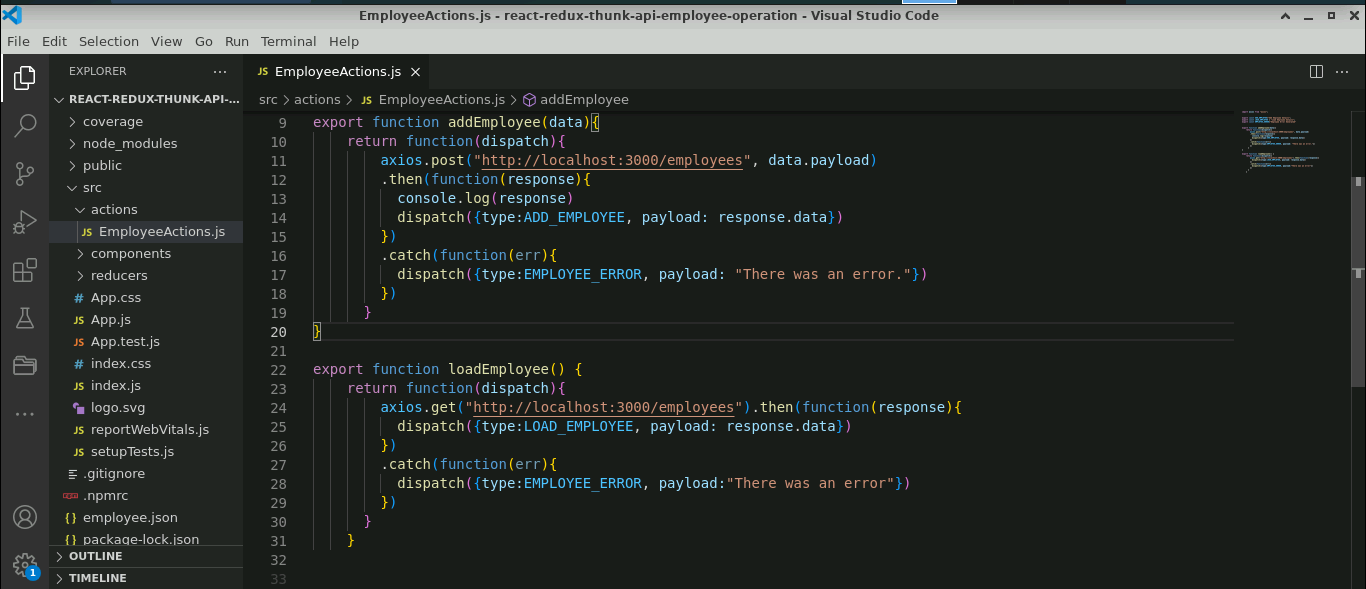
**.catch(function(err){**

**dispatch({type:EMPLOYEE\_ERROR, payload:"There was an error"})**

**})**

**}**

**}**

****

* 1. Inside the **components** folder, create a file named **AddEmployee.js** and enter the following code:

**import React, { useState } from 'react';**

**import {connect, useDispatch} from 'react-redux';**

**import {addEmployee,ADD\_EMPLOYEE, loadEmployee} from '../actions/EmployeeActions';**

**import "../../node\_modules/bootstrap/dist/css/bootstrap.min.css"**

**function AddEmployee() {**

**let [employee,setEmployee]=useState({});**

**let dispatch = useDispatch();**

**let storeEmployeeInfo=async (event) => {**

**event.preventDefault();**

**let result = await dispatch(addEmployee({type:"ADD\_EMPLOYEE",payload:employee}));**

**console.log(result);**

**setEmployee({id:"",name:"",salary:""})**

**}**

**return(**

**<div className="container">**

**<h2>Add Employee Details</h2>**

**<form onSubmit={storeEmployeeInfo} className="form-group col-md-12">**

**Id <input type="text" name="id" value= {employee.id} onChange={**

**(event)=>setEmployee({...employee,"id":event.target.value})**

**}**

**className="form-control"**

**/><br/>**

**Name <input type="text" name="name" value= {employee.name} onChange={**

**(event)=>setEmployee({...employee,"name":event.target.value})**

**} className="form-control"/>**

**<br/>**

**Salary <input type="text" name="salary" value= {employee.salary} onChange={**

**(event)=>setEmployee({...employee,"salary":event.target.value})**

**} className="form-control"/>**

**<br/>**

**<input type="submit" value="Store Record" className="btn btn-primary"/>**

**</form>**

**</div>**

**)**

**}**

**export default AddEmployee;**

**A screenshot of a computer program

Description automatically generated**

* 1. Inside the **components** folder, create a file named **DisplayEmployee.js** and enter the following code:

**import React, { useEffect } from 'react';**

**import {useDispatch, useSelector} from 'react-redux';**

**import { loadEmployee } from '../actions/EmployeeActions';**

**function DisplayEmployee() {**

**let employees = useSelector(gs=>gs.employee);**

**let dispatch = useDispatch();**

**console.log(employees);**

**useEffect(()=> {**

**const loadData=()=> {**

**dispatch(loadEmployee());**

**}**

**loadData();**

**},[])**

**return(**

**<div>**

**<h3>Employee Details are </h3>**

**<table border="1" className="table table-striped">**

**<thead>**

**<tr>**

**<th>Id</th>**

**<th>Name</th>**

**<th>Salary</th>**

**</tr>**

**</thead>**

**<tbody>**

**{employees.map((ele,i)=>**

**<tr key={i}><td>{ele.id}</td><td>{ele.name}</td><td>{ele.salary}</td></tr>)**

**}**

**</tbody>**

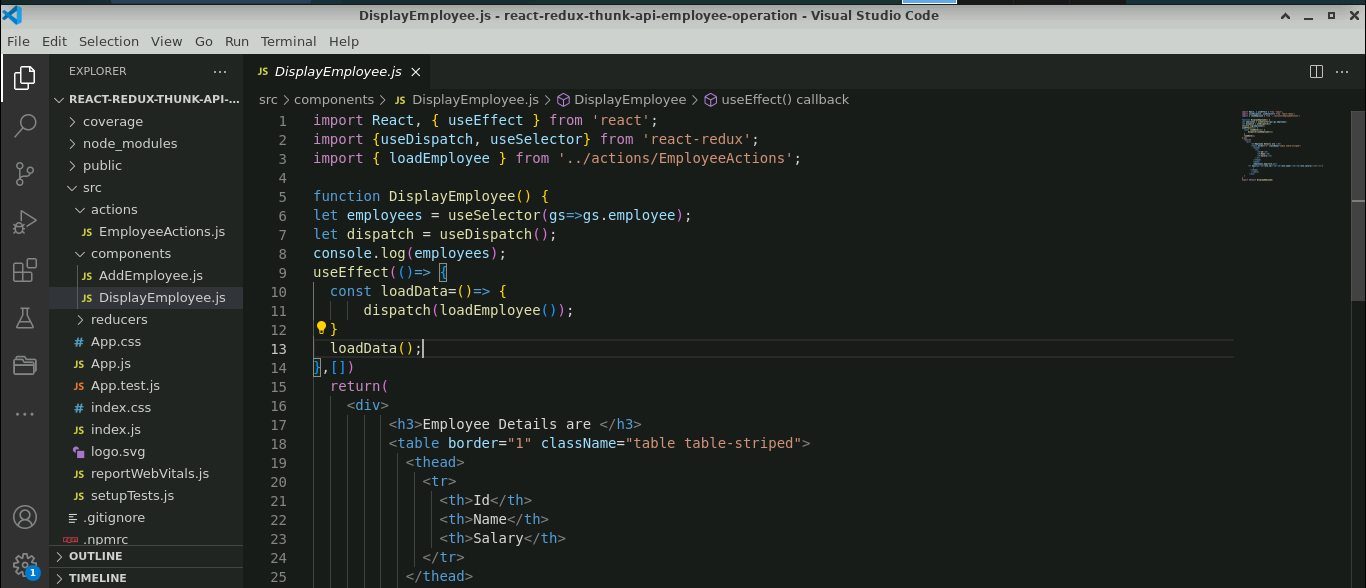
**</table>**

**</div>**

**)**

**}**

**export default DisplayEmployee;**

****

* 1. Inside the **reducers** folder, create a file named **EmployeeReducers.js** and enter the following code:

**import {ADD\_EMPLOYEE,LOAD\_EMPLOYEE,EMPLOYEE\_ERROR} from '../actions/EmployeeActions';**

**export default function employeeReducer(state=[],action) {**

**console.log("Action is "+action)**

**switch(action.type){**

**case LOAD\_EMPLOYEE:**

**return action.payload;**

**case ADD\_EMPLOYEE:**

**return [...state, action.payload]**

**case EMPLOYEE\_ERROR:**

**return action.payload;**

**default:return state;**

**}**

**}**

**A screen shot of a computer program

Description automatically generated**

**Step 4: Configure the store and Thunk details in the index.js file**

1. In the **index.js** file, create a **store** and configure the **reducer** and **Thunk** module details as shown below:

**import ReactDOM from 'react-dom/client';**

**import './index.css';**

**import App from './App';**

**import reportWebVitals from './reportWebVitals';**

**import React from 'react';**

**import { legacy\_createStore as createStore,combineReducers,applyMiddleware, compose } from 'redux';**

**import {Provider} from 'react-redux'**

**import {thunk} from "redux-thunk"**

**import employeeReducer from './reducers/EmployeeReducers';**

**const composeEnhancers = (window && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_) || compose;**

**const enhancer = composeEnhancers(**

**applyMiddleware(thunk),**

**// other store enhancers if any**

**);**

**const allReducer = combineReducers({**

**employee:employeeReducer**

**})**

**const store = createStore(allReducer,enhancer);**

**const root = ReactDOM.createRoot(document.getElementById('root'));**

**root.render(**

**<React.StrictMode>**

**<Provider store={store}><App /></Provider>**

**</React.StrictMode>**

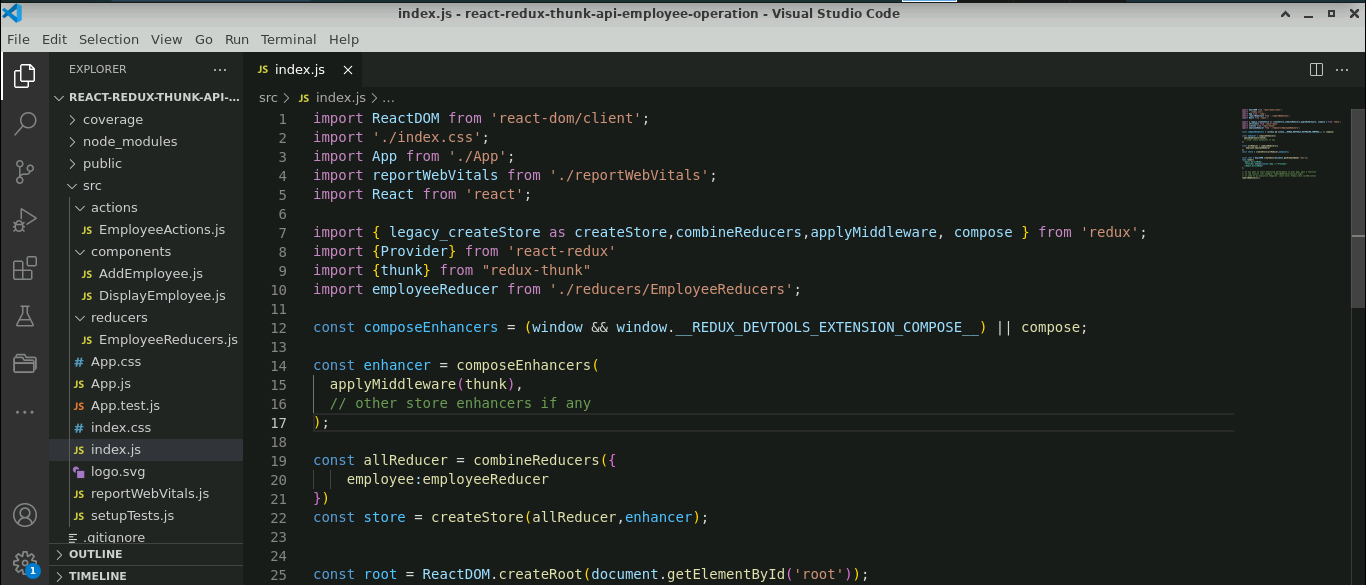
**);**

**// If you want to start measuring performance in your app, pass a function**

**// to log results (for example: reportWebVitals(console.log))**

**// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals**

**reportWebVitals();**

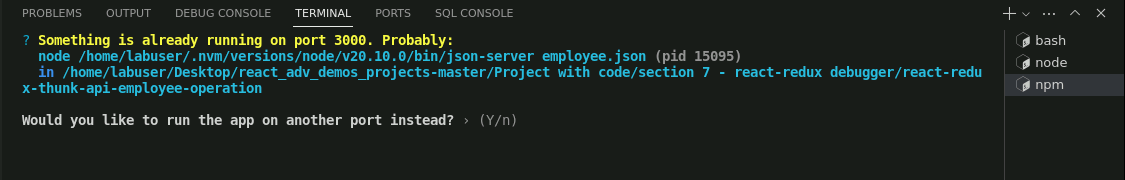


**Step 5: Run the application**

* 1. Open the terminal and run the below command to execute the application:

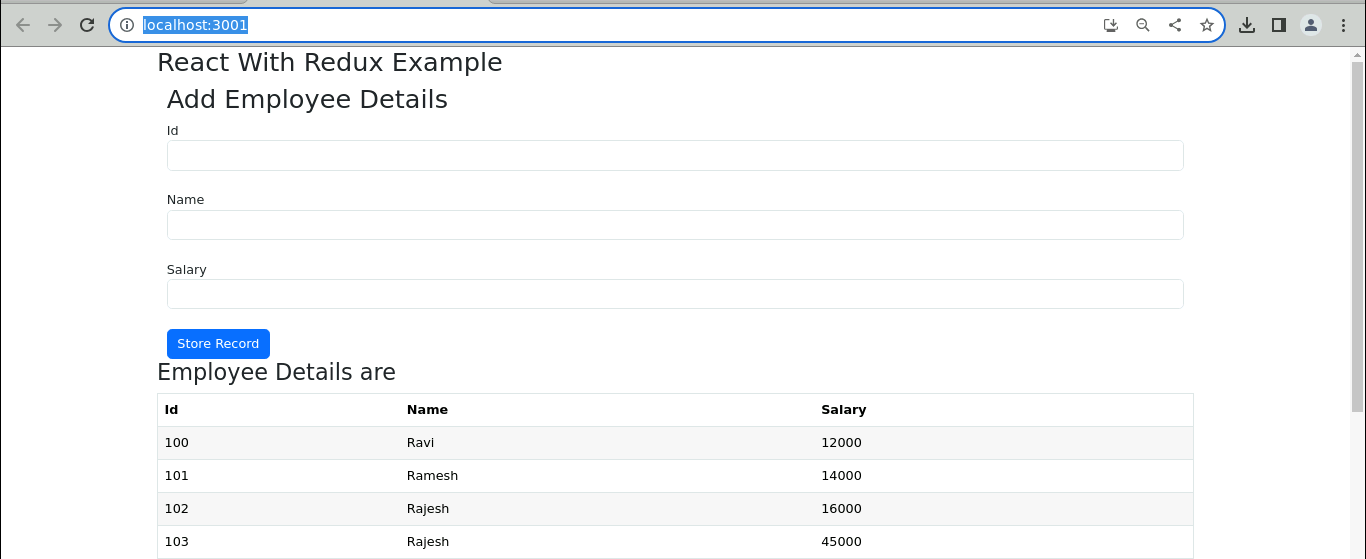
**npm start**





**Note:** The program may ask the user to save it on a different port number. This is because the default port number for json-server and React is running on 3000. Since the json-server is already running on port 3000 for the employee.json file, please type **'Y'** when the following question is raised: **Would you like to run the app on another port instead?**

The output appears as shown below:



* 1. Under **Add Employee Details** page, enter the **Id**, **Name,** and **Salary** and click on **Store Record** button

A screenshot of a computer

Description automatically generated

The output appears as shown below:

A screenshot of a computer

Description automatically generated

**Step 6: Debug the application using the Redux DevTools with the browser**

* 1. In the output screen, right-click and select **Inspect**

A screenshot of a computer

Description automatically generated

* 1. Navigate to the **Elements** tab, click on **>>** (right-side corner), and choose **Redux** from the options

A screenshot of a computer

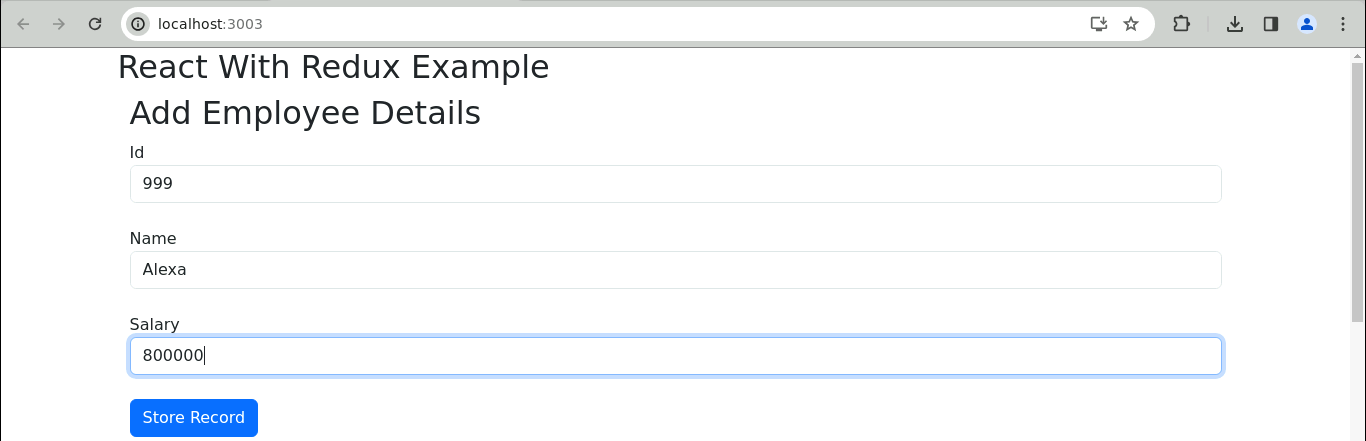
Description automatically generated

The output appears as shown below, and the user can access employee details using the Redux DevTools.

A screenshot of a computer

Description automatically generated

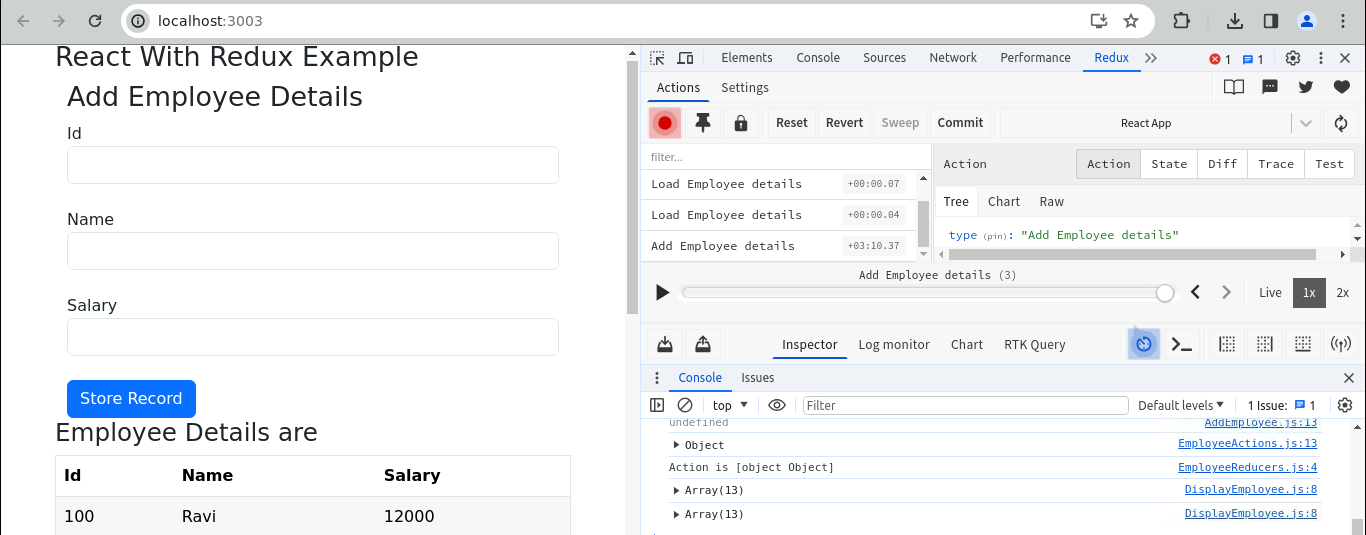
* 1. Enter the employee information and verify it in Redux **DevTools**



A screenshot of a computer

Description automatically generated

The output appears as shown below:



* 1. Click the **play** button to initiate the recording operation and observe the time taken for the execution of that operation

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The output appears as shown below:

A screenshot of a computer

Description automatically generated

* 1. Navigate to the **State** located in the right-hand corner to view the state information

A screenshot of a computer

Description automatically generated

With this, you have successfully created a React app with Redux using the Thunk middleware to store and retrieve employee details from a JSON file using Axios and integrate Redux DevTools for efficient state inspection and debugging.