**Lesson 04 Demo 04**

**Creating a React with Redux Employee Application**

**Objective:** To create a React application with the redux store, view and delete employee information from store

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

**Prerequisites:** None

Steps to be followed:

1. Create and set up the React project
2. Configure the Redux
3. Create a user-defined component
4. Test the application

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

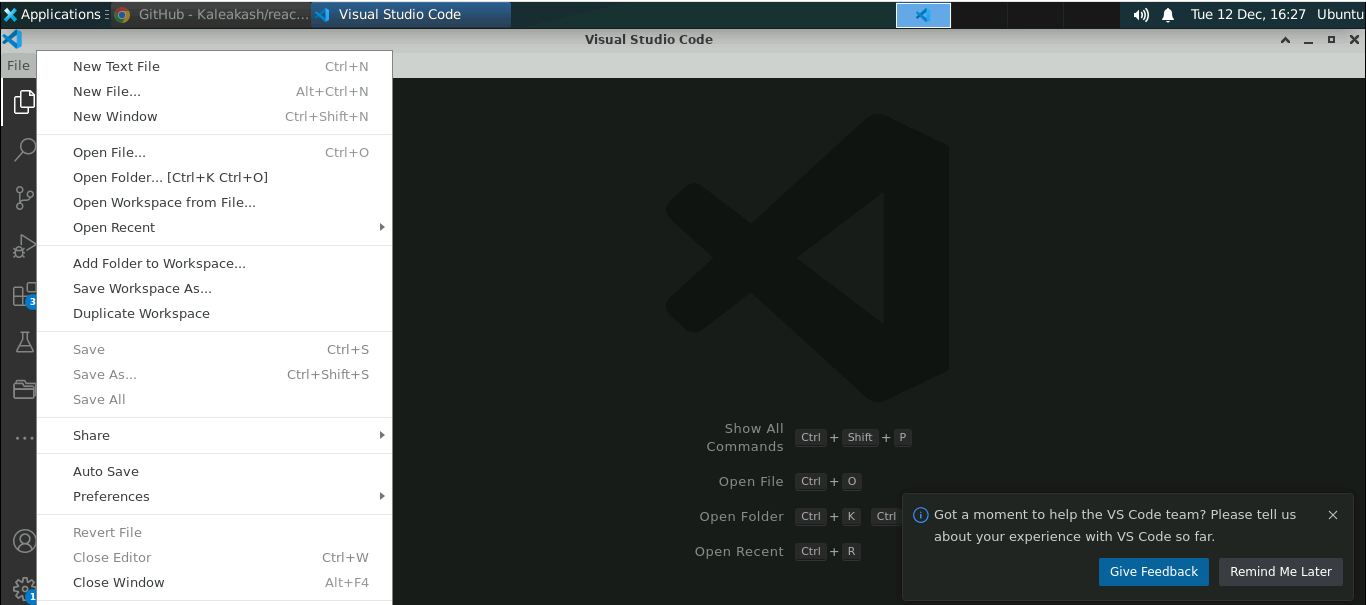
* 1. Open a terminal window and run the following command to create a React application:

**npx create-react-app** **redux-employee-app**

A screenshot of a computer

Description automatically generated

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

****

* 1. Click on **Open** button

**A screenshot of a computer

Description automatically generated**

The folder 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 program

Description automatically generated

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

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

A screenshot of a computer

Description automatically generated

**Step 2: Configure the Redux**

1. Inside the **src** folder, create an **employeeReducer.js** file and enter the following code:

**let initialState = {**

**employees:[**

**{id:100,name:"Ravi",age:21},**

**{id:101,name:"Ramesh",age:22}**

**]**

**}**

**function employeeReducer(state=initialState,action){**

**//Based upon action we can make the changes on the state variable.**

**// which is consider as global state**

**console.log(action);**

**if(action.type=="ADD\_EMPLOYEE"){**

**return {**

**...state,employees:state.employees.concat(action.payload)**

**//...state,employees:[...state.employees,action.payload]**

**}**

**}**

**if(action.type=="DELETE\_EMPLOYEE"){**

**return {**

**...state,employees: state.employees.filter(e=> e.id !== action.payload)**

**}**

**}**

**return state;**

**}**

**export default employeeReducer;**

**A screen shot of a computer

Description automatically generated**

1. Inside the **src** folder, create an **employeeStore.js** file to connect the reducer to the Redux store to make the state variable a global variable

**import { legacy\_createStore as createStore} from 'redux';**

**import reducer from './employeeReducer';**

**const store = createStore(reducer);**

**export default store;**

A screenshot of a computer

Description automatically generated

1. Configure the **store** using the **Provider** in **index.js**

**import React from 'react';**

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

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

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

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

**import store from './employeeStore'**

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

**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();**

A screen shot of a computer

Description automatically generated

**Step 3:** **Create an user-defined component**

1. Inside the **src** folder, create a **DisplayEmployee.js** file and enter the below code. It helps to access global state variables using the **useSelector** hook and displays the records in a table format.

**import { useSelector } from "react-redux";**

**import DeleteEmployee from "./DeleteEmployee";**

**function DisplayEmployee() {**

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

**let employeeRecord = employees.map((e,index)=>**

**<tr key={index}>**

**<td>{e.id}</td>**

**<td>{e.name}</td>**

**<td>{e.age}</td>**

**<td><DeleteEmployee id={e.id}/></td>**

**</tr>**

**)**

**return(**

**<div>**

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

**<table border="1">**

**<thead>**

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

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

**<th>Age</th>**

**<th>Delete</th>**

**</thead>**

**<tbody>**

**{employeeRecord}**

**</tbody>**

**</table>**

**</div>**

**)**

**}**

**export default DisplayEmployee;**

**A screen shot of a computer

Description automatically generated**

1. Inside the **src** folder, create an **AddEmployee.js** file and enter the below code. It helps to add the employee details with the **useDispatch** hook.

**import { useState } from "react";**

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

**function AddEmployee() {**

**//useDispatch is pre-defined hook**

**// which help to pass action and payload (data)**

**// to reducer**

**let [employee,setEmployee]=useState({id:0,name:"",age:0.0})**

**let dispatch = useDispatch();**

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

**let addEmployee = (event)=> {**

**event.preventDefault();**

**//console.log(employee)**

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

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

**dispatch({type:"ADD\_EMPLOYEE",payload:employee})**

**}else {**

**alert("Employee id must be unique")**

**}**

**setEmployee({id:0,name:"",age:0.0})**

**}**

**return(**

**<div>**

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

**<form onSubmit={addEmployee}>**

**<label>Id</label>**

**<input type="number" name="employee.id" value={employee.id} onChange={(event)=>**

**{**

**setEmployee(e=>{return {...e,id:event.target.value}})**

**}}/><br/>**

**<label>Name</label>**

**<input type="text" name="employee.name" value={employee.name} onChange={(event)=>**

**{**

**setEmployee(e=>{return {...e,name:event.target.value}})**

**}}/> <br/>**

**<label>Age</label>**

**<input type="number" name="employee.age" value={employee.age} onChange={(event)=>**

**{**

**setEmployee(e=>{return {...e,age:event.target.value}})**

**}}/><br/>**

**<input type="submit" value="Add Employee" />**

**<input type="reset" value="reset"/>**

**</form>**

**</div>**

**)**

**}**

**export default AddEmployee;**

A screen shot of a computer program

Description automatically generated

* 1. Inside the **src** folder, create a **DeleteEmployee.js** file and enter the below code. It helps to delete the employee details using the employee id with the help of the **dispatch** function.

**import { useDispatch } from "react-redux";**

**function DeleteEmployee({id}){**

**let dispatch = useDispatch();**

**return(**

**<div>**

**<td><input type="button" value="Delete"**

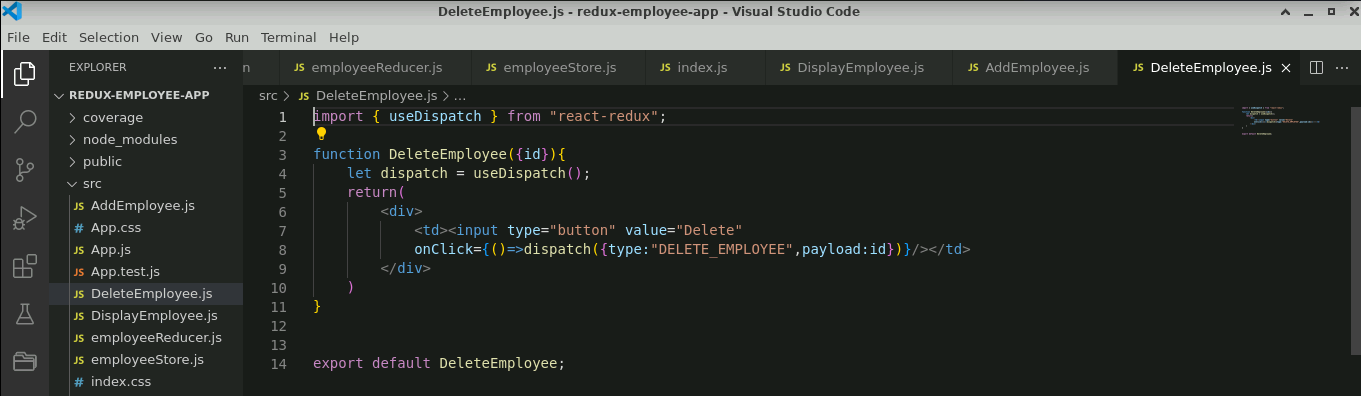
**onClick={()=>dispatch({type:"DELETE\_EMPLOYEE",payload:id})}/></td>**

**</div>**

**)**

**}**

**export default DeleteEmployee;**



**Step 4: Test the application**

1. In **App.js** file, import **AddEmployee** and **DisplayEmployee**

**import AddEmployee from './AddEmployee';**

**import './App.css';**

**import DisplayEmployee from './DisplayEmployee';**

**function App() {**

**return (**

**<div>**

**<AddEmployee></AddEmployee>**

**<DisplayEmployee></DisplayEmployee>**

**</div>**

**);**

**}**

**export default App;**

A screenshot of a computer

Description automatically generated

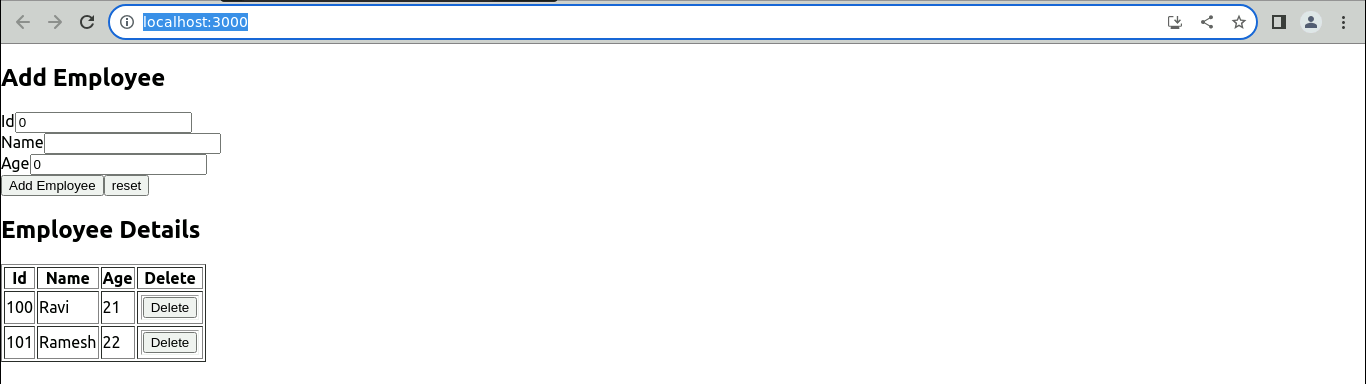
* 1. Open the terminal and run the below command to execute the application:

**npm start**

A screen shot of a computer

Description automatically generated

The output appears as shown below:



* 1. Under the **Add Employee** page, add the **Id**, **Name,** and **Age,** and then click on **Add Employee** to add the employee details

A white rectangular object with a black border

Description automatically generated

The employee details appear as shown below:

A screenshot of a computer

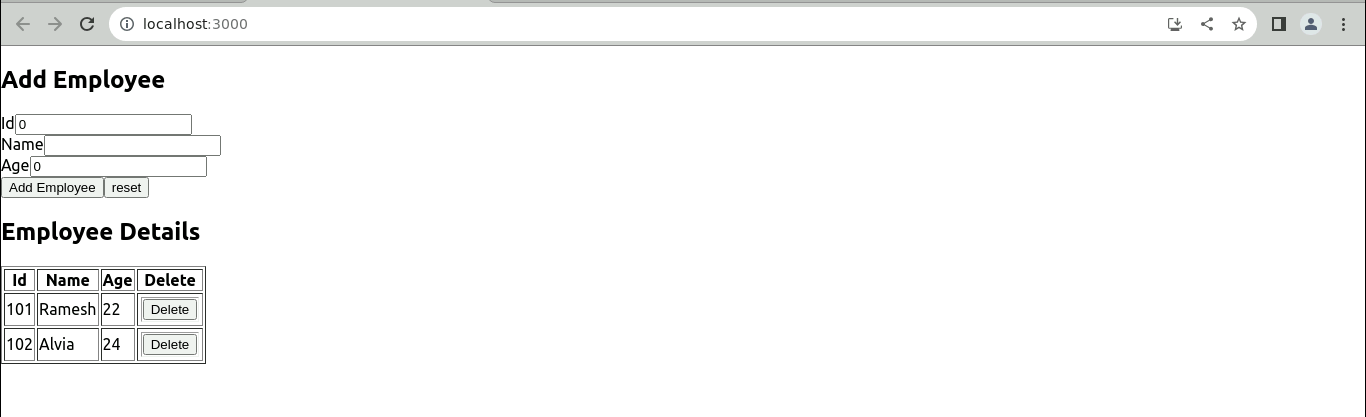
Description automatically generated

* 1. Under the **Employee** **Details** page, select employee detail and click on **Delete** to delete the employee detail

A screenshot of a computer

Description automatically generated

The output appears as shown below:



With this, you have successfully created a React application with a Redux store, view, and delete employee details from the store to manage employee details.