**Lesson 07 Demo 03**

**react-redux-thunk-api-employee-operation**

**Objective:** To demonstrate the react with redux using thunk module to store and retrieve employee details in json file with help of axios and thunk module. Debug this application with redux dev tool with action when we perform on components.

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

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

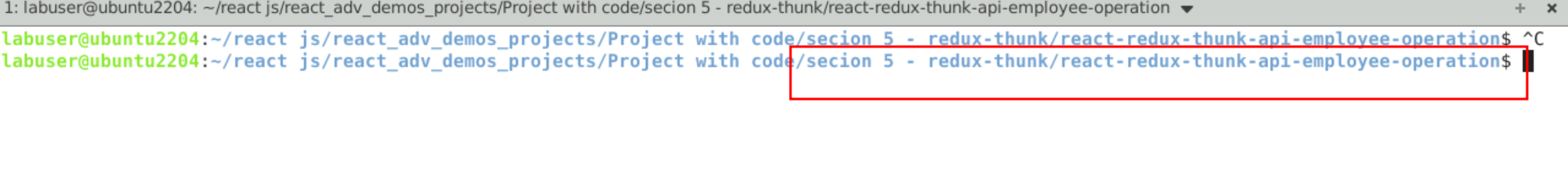
**Note** : All react js project already created with version 18.x with Sample App.js file

**Steps to be followed:**

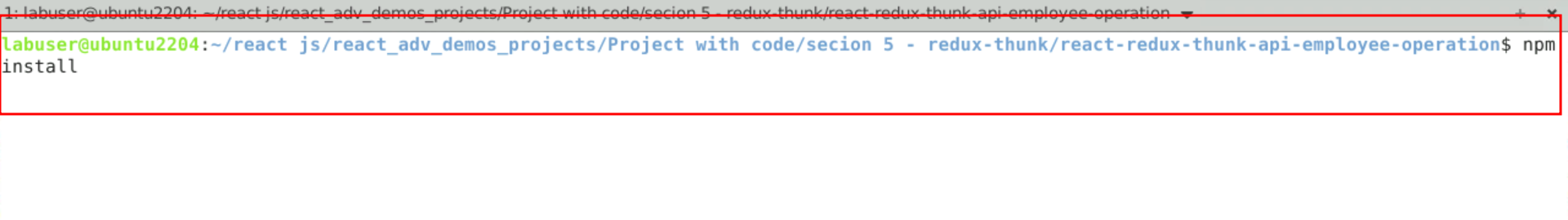
1. Set up for react js project
2. Create employee.json file which contains few static employee details in the form of json file.
3. Create action folder which contains EmployeeAction.js file
4. Create reducers folder which contains employeeReducer.js file
5. Component folder contains AddEmploee.js and DisplayEmployee.js file
6. In index.js file configure store and thunk details.
7. Please install redux dev tools in browser.
8. Test the application.
9. Now debug this application using redux dev tool with browser.

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

1. Open a terminal window inside a React JS pre-created project ie **react-redux-thunk-api-employee-operation**

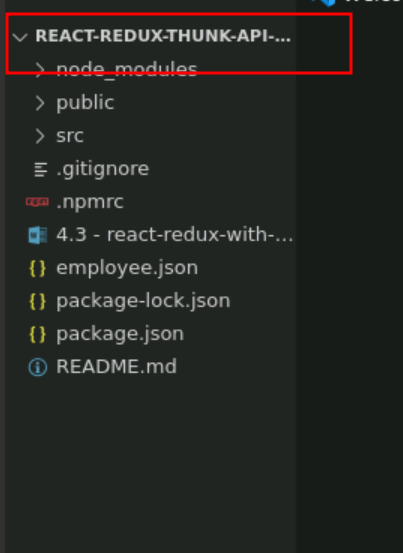


1. Now you need to run the command as **npm install.** This command helps us to installed all required dependencies mention in package.json file in local machine in the form of node\_module folder.

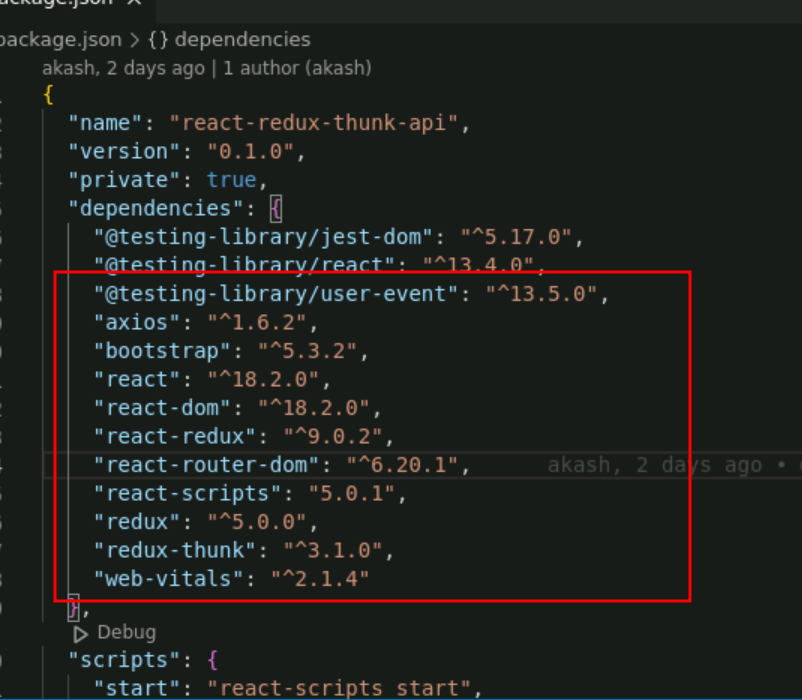


1. Now open **react-redux-thunk-api-employee-operation** folder in VS Code Editor

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

****

1.4 now open package.json file and view external dependencies.

****

**Step 2 : Create employee.json file which contains few static employee details in the form of json file.**

2.1 create the employee.json file which contains static json data.

**employee.json**

{

"employees": [

{

"id": 100,

"name": "Ravi",

"salary": 12000

},

{

"id": 101,

"name": "Ramesh",

"salary": 14000

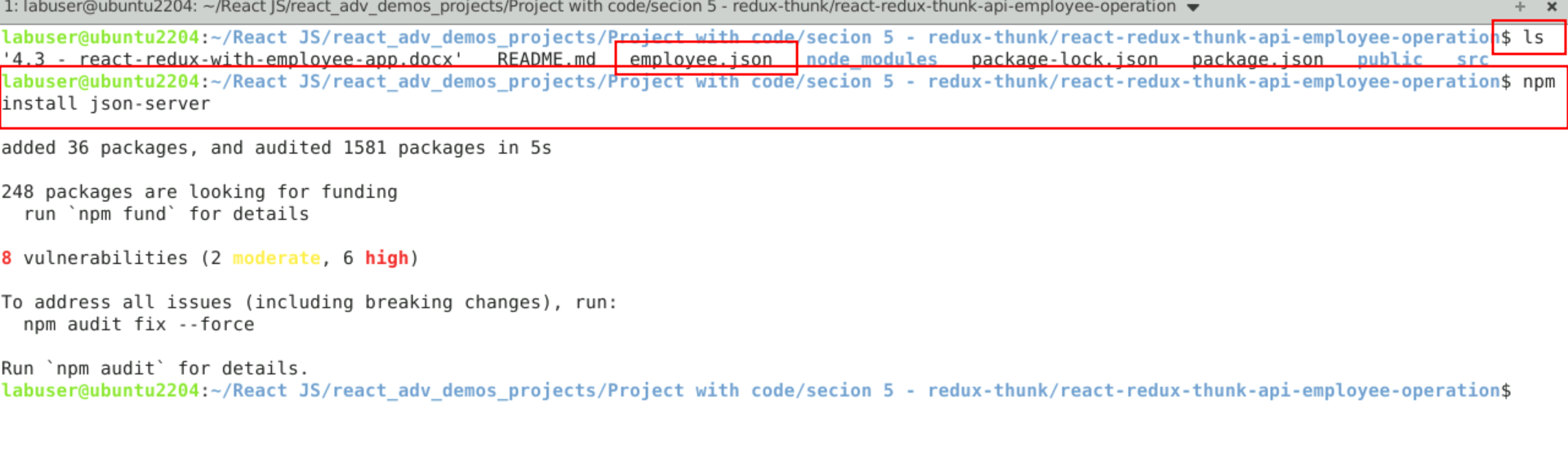
}

]

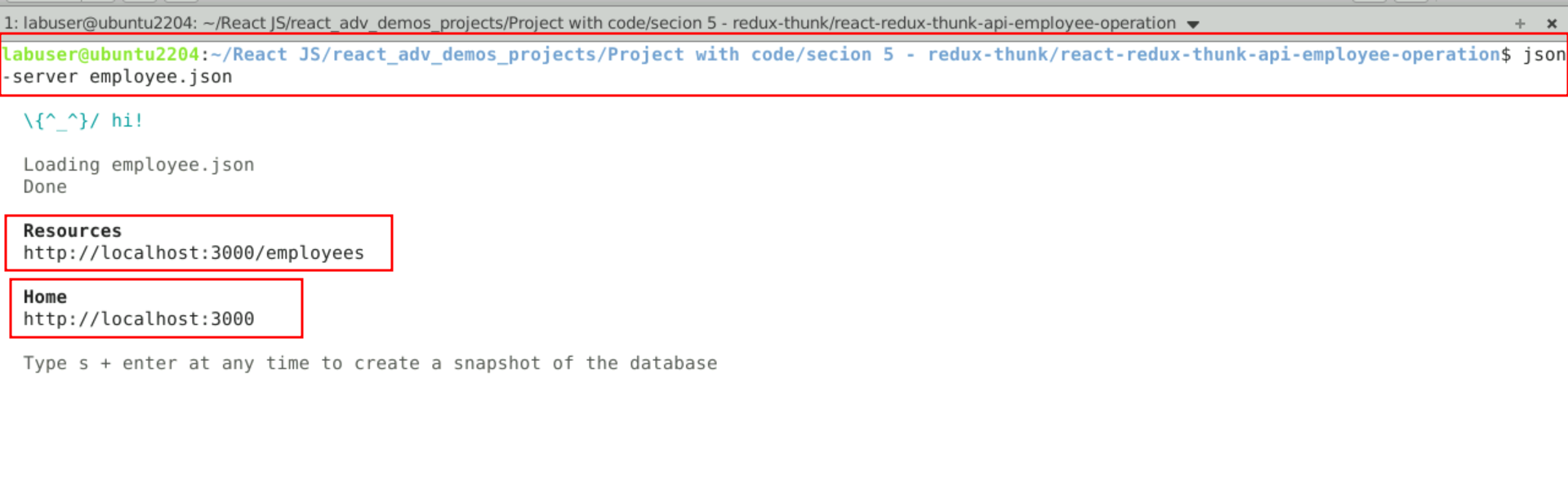
}

2.2 We need to run this file using the node module as **json-server**. So first install json-server module using command as **npm install json-server -g**

Open the folder inside that location employee.json file present.



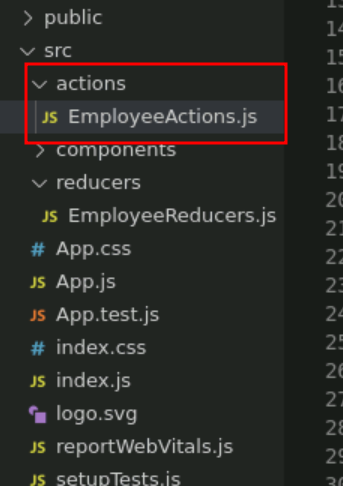
Now you can run the command as json-server employee.json file which provide rest api.



So here <http://localhost:3000/employees> URL provide REST API. You can add and retrieve operation through react JS.

**Step 3:** Create action folder which contains EmployeeAction.js file

3.1 create action folder which contains EmployeeAction.js file



3.2 Inside EmployeeAction.js file contains Action fields and base upon action we can interact with rest api using axios modules. If you want we can write action field in separate file as well as fetchAPI function in another file.

**EmployeeAction.js**

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"})

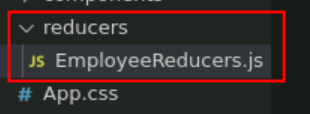
})

}

}

**Step 4 : Create reducers folder which contains employeeReducer.js file**

4.1 Create reducer folder and create employeeReducer.js file



4.2 employeeReducer.js file contains normal function which takes two parameter ie state and action. Base upon action it do the action with help axios with thunk modules.

**EmployeeReducers.js file**

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;

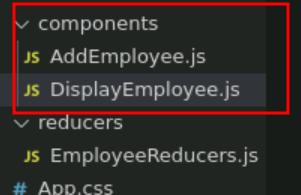
default:return state;

}

}

**Step 5 : Component folder contains AddEmploee.js and DisplayEmployee.js file**

5.1 Now create component folder and inside this folder cerate AddEmployee.js and DisplayEmployee.js file



5.2 AddEmployee.js file. Inside this file create user defined component which contains forms we take employee id ,name and salary and using useSelector and useDispatch function we interact with redux and thunk to store the data in employee.json file.

**AddEmployee.js file**

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;

5.3 DisplayEmployee.js file. Inside this file create user defined component which take the help of useDispatch and useSelector hook interact with redux with thunk to load the data from employee.json file.

**DisplayEmployee.js file**

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;

**Step 6 : In index.js file configure store and thunk details.**

6.1 In index.js file we need to create store and configure reducer and thunk modules details

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

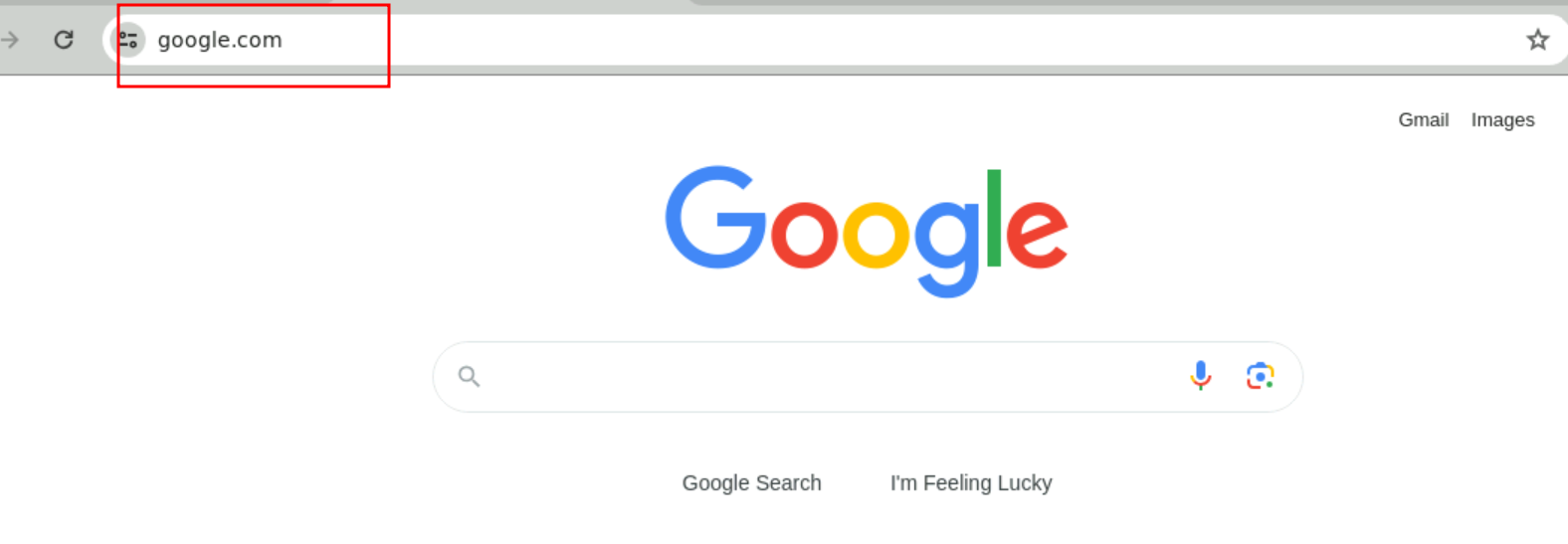
// to log results (for example: reportWebVitals(console.log))

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

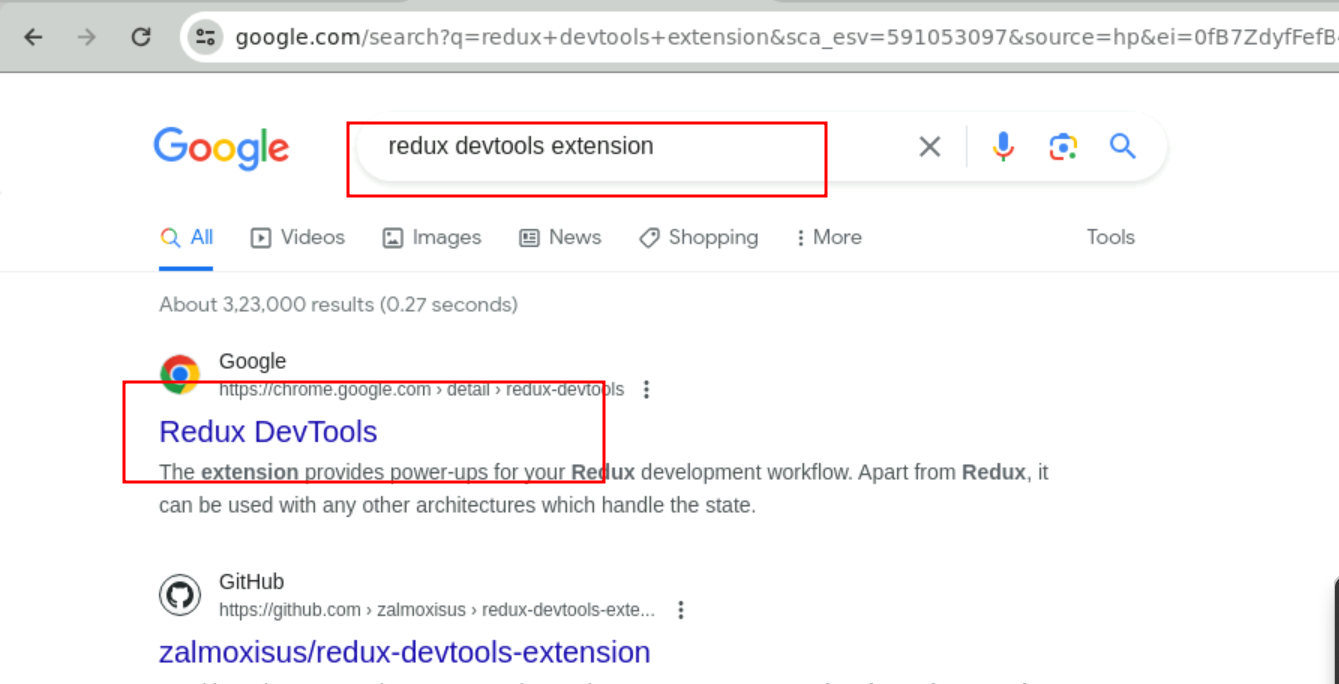
reportWebVitals();

**Step 7 :** Please install redux dev tools in browser.

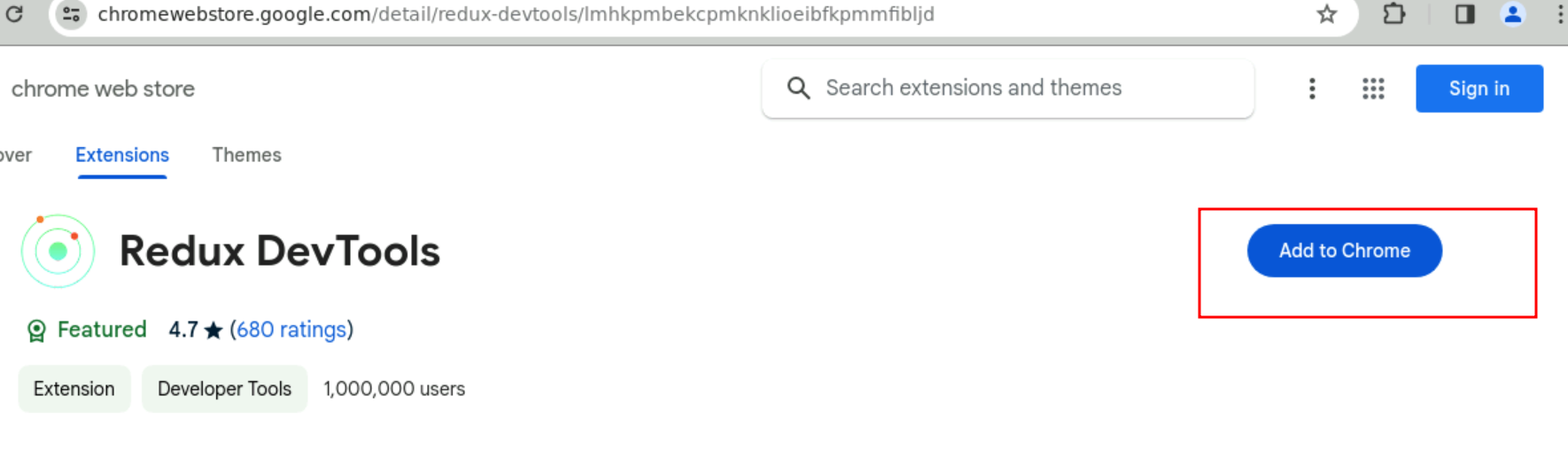
7.1 Open the chrome browser

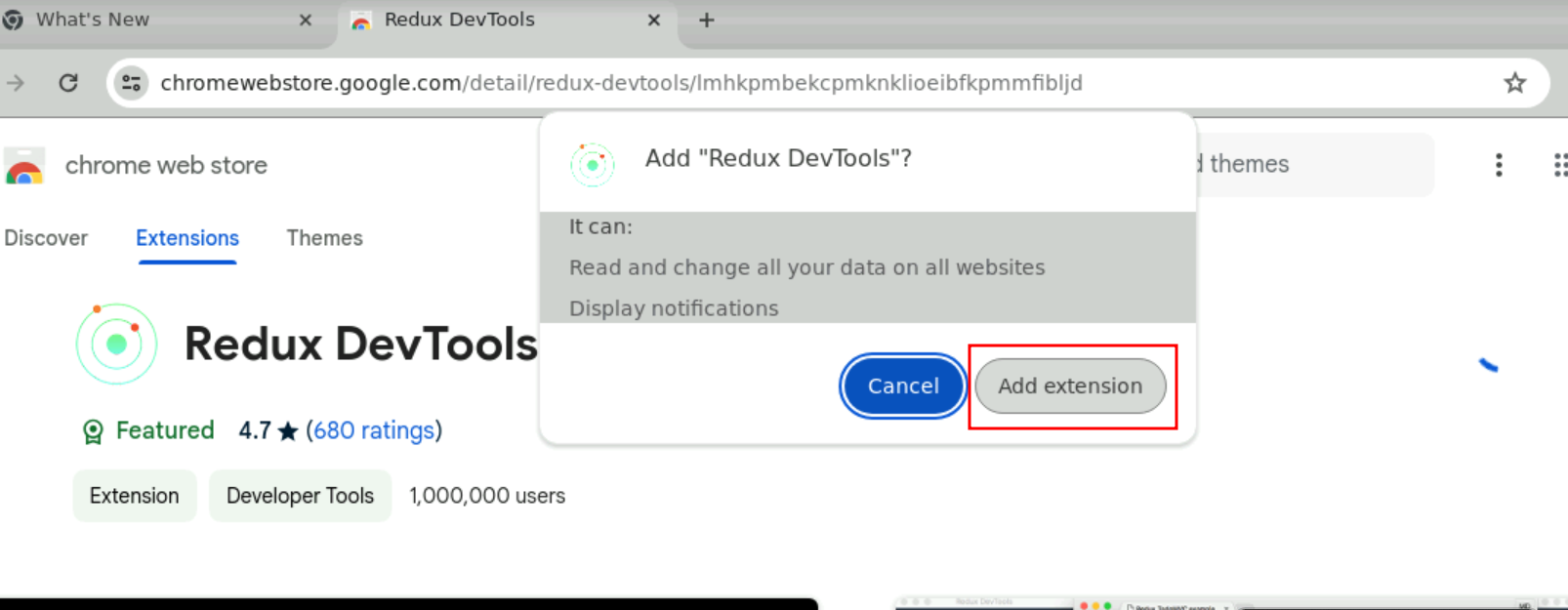


7.2 **search redux dev tool plugins**



7.3 add this plugin to browser.

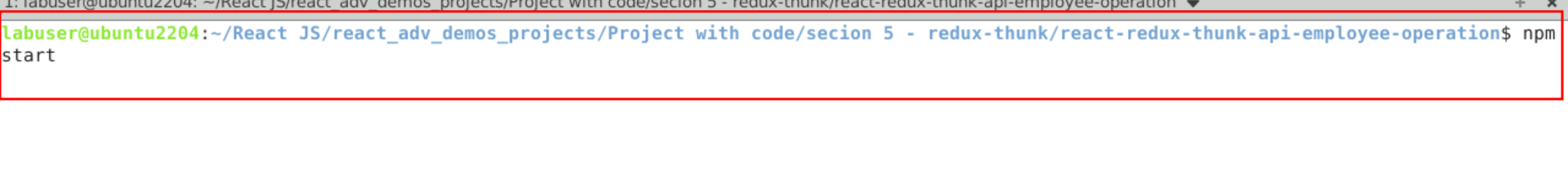




Then restart the browser.

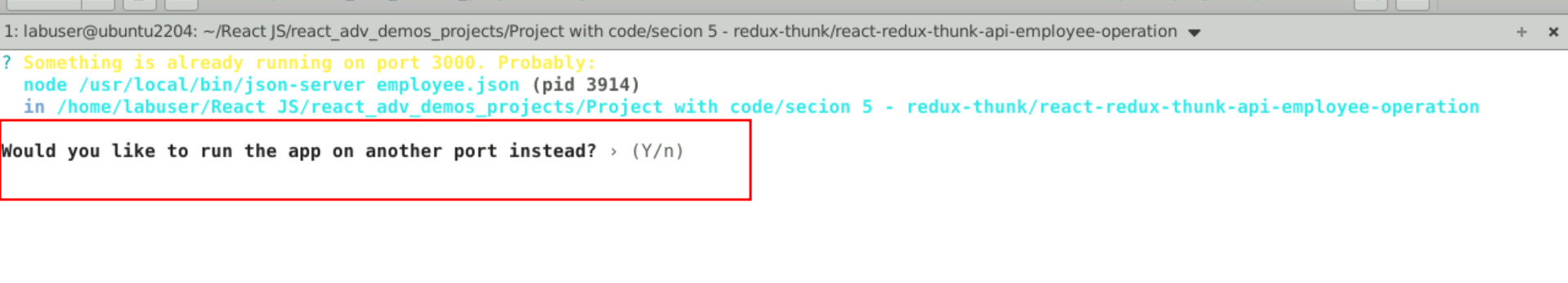
**Step 8 Now we run the application using npm start**

**8.1** Now run the application usiing command as **npm start**

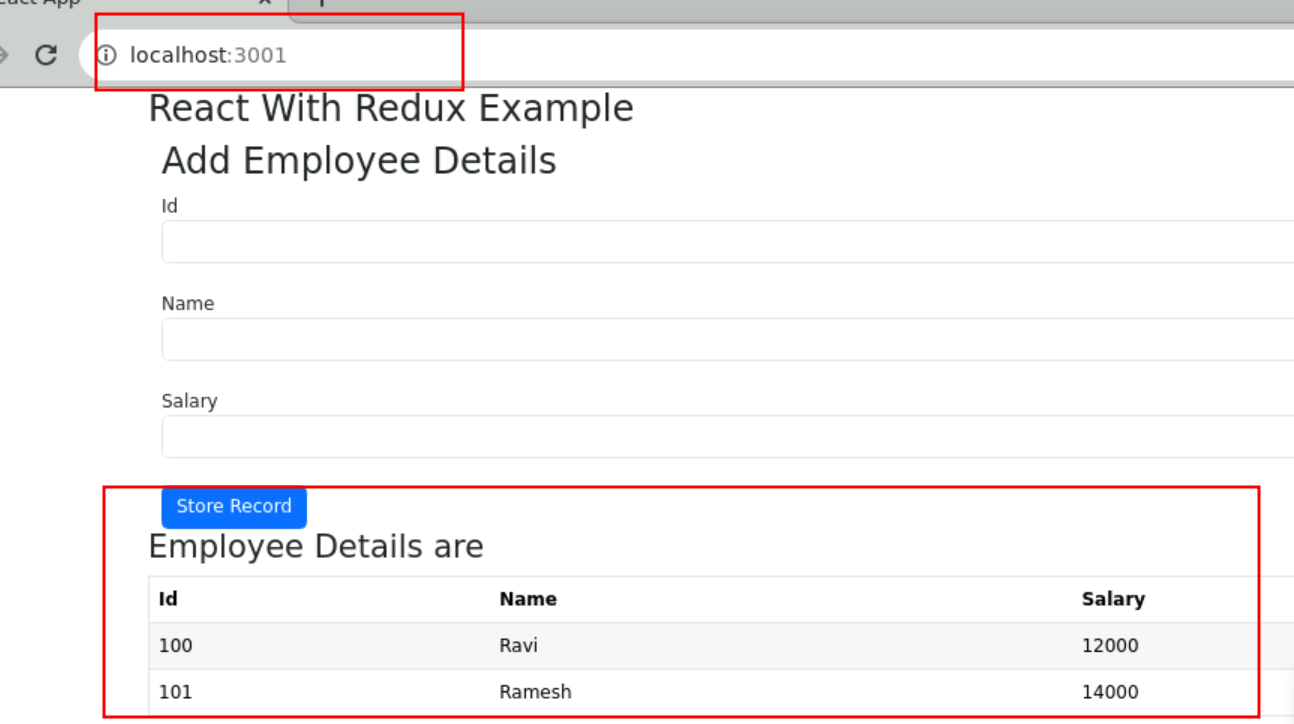


**8.2**  Now you can view the output on browser. You can see the search text field.

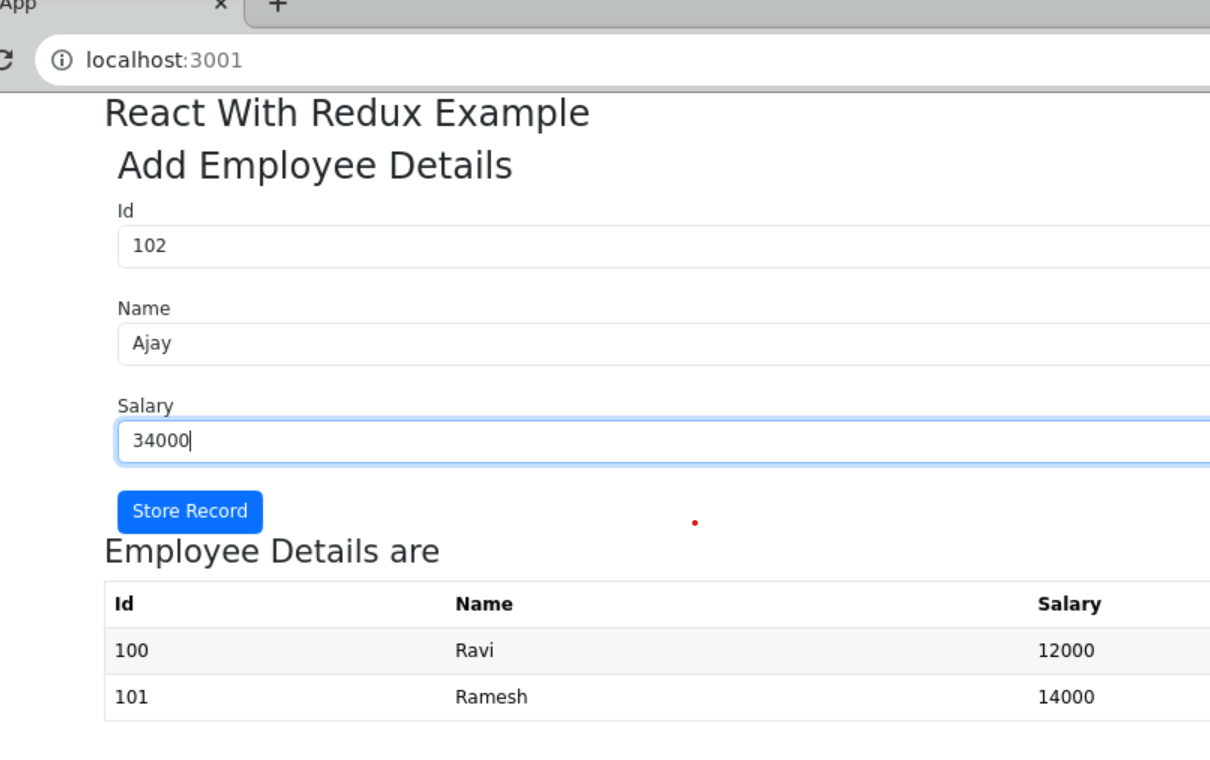
**8.3** It may ask you to store this application in another port number. Because by default json-server also run on default port number 3000 as well as react js also run on default port number 3000. So already json-server running employee.json file on port number 3000. So give yes it will run on 3001 port number.



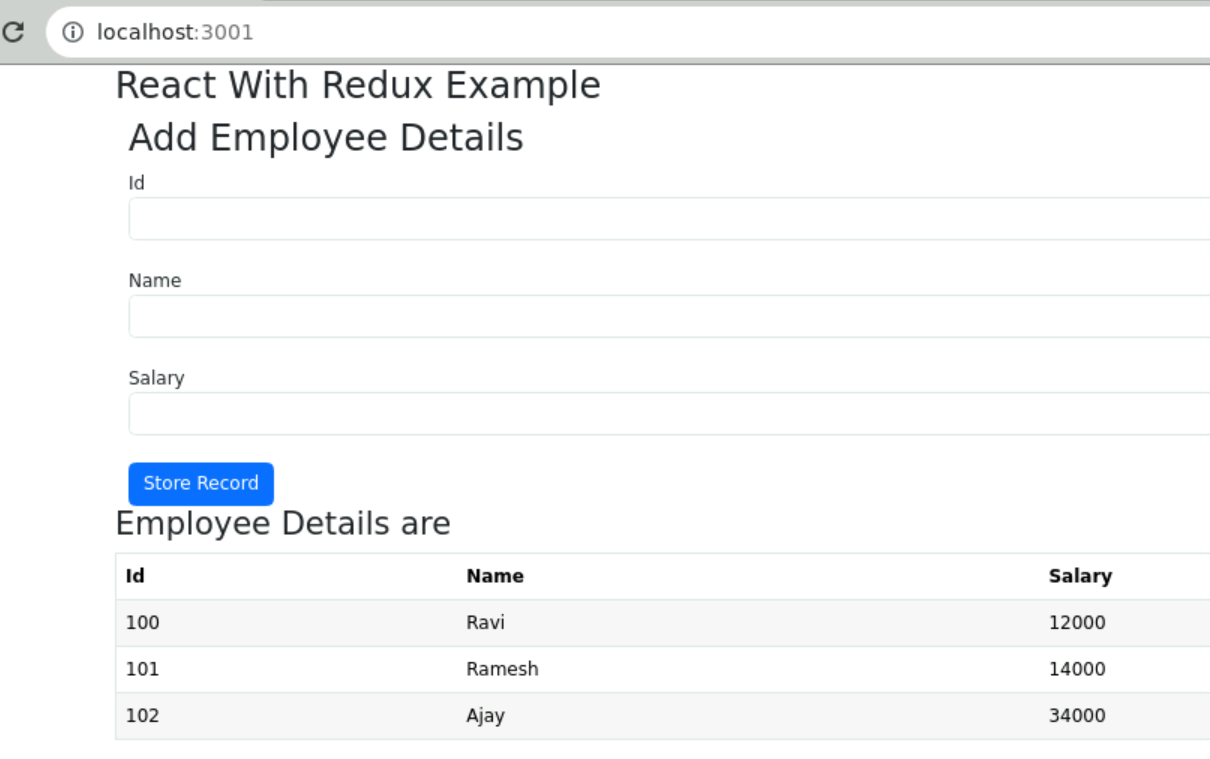
8.4 after running this application on browse



8.5 now store the records.

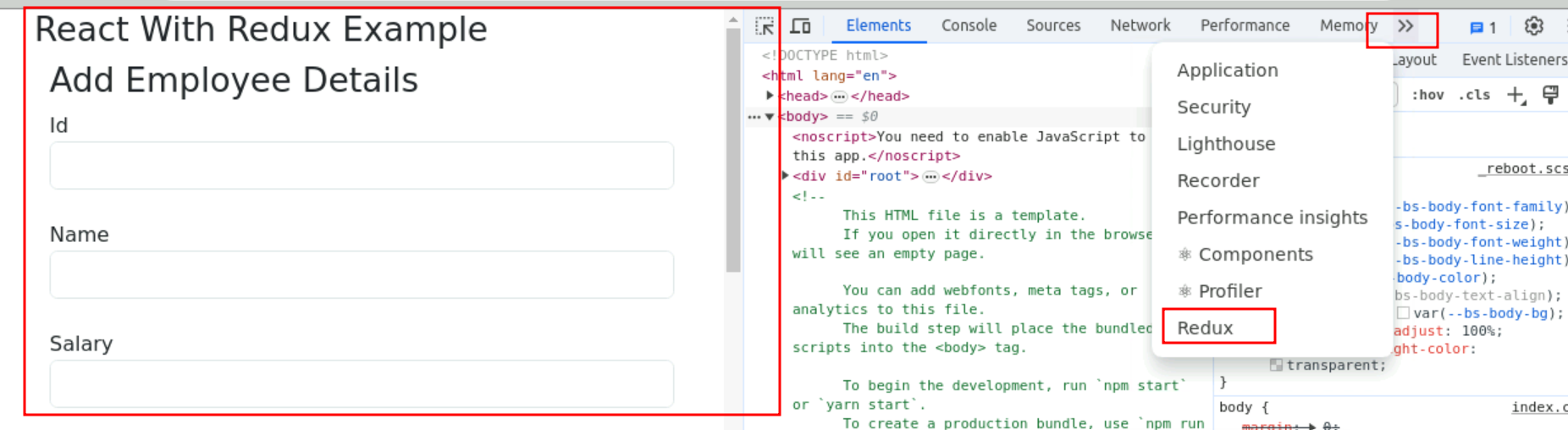


8.6 after store the records.

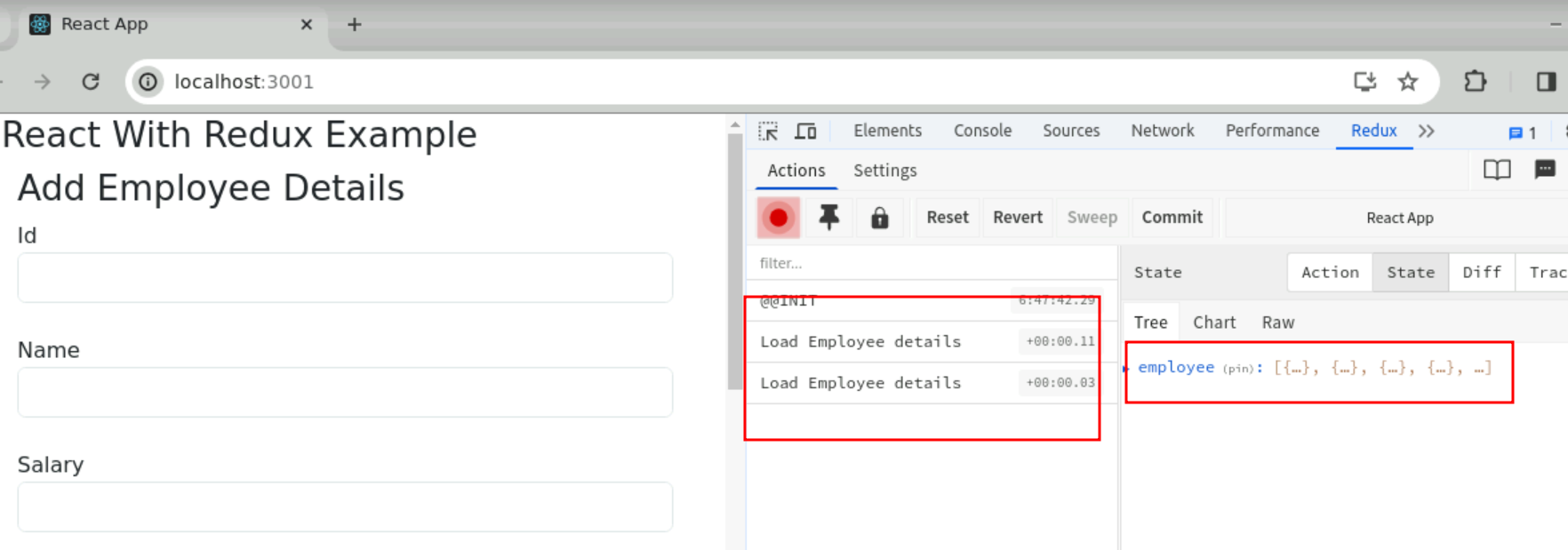


**Step 9: Now debug this application using redux dev tool with browser.**

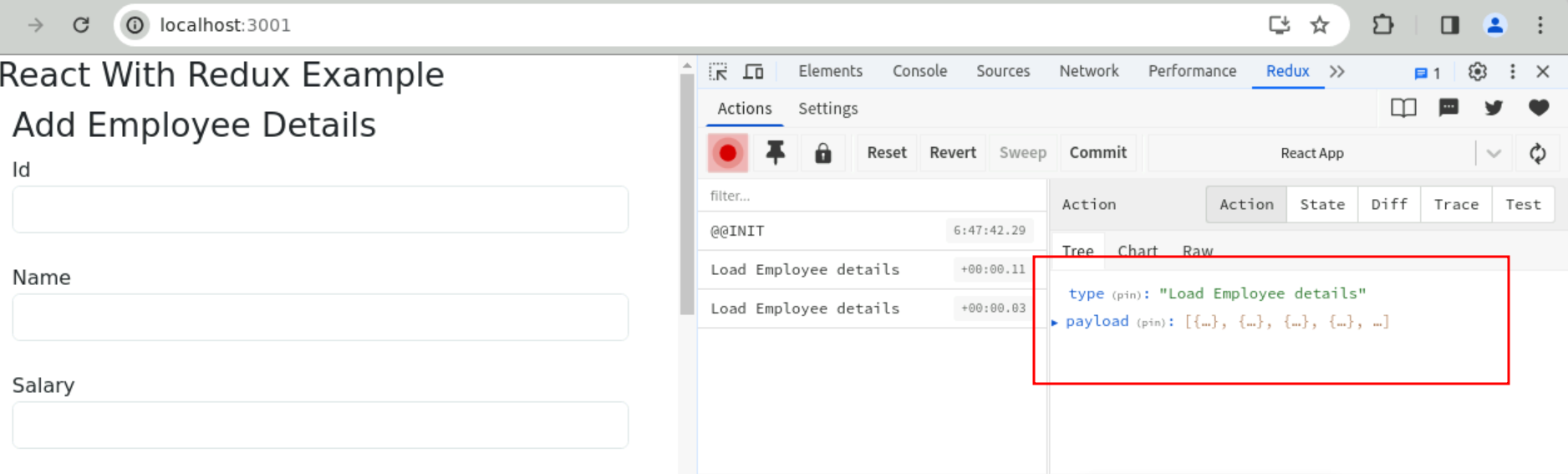
9.1 In browser write cntr + shift + I to open browser redux dev tool.



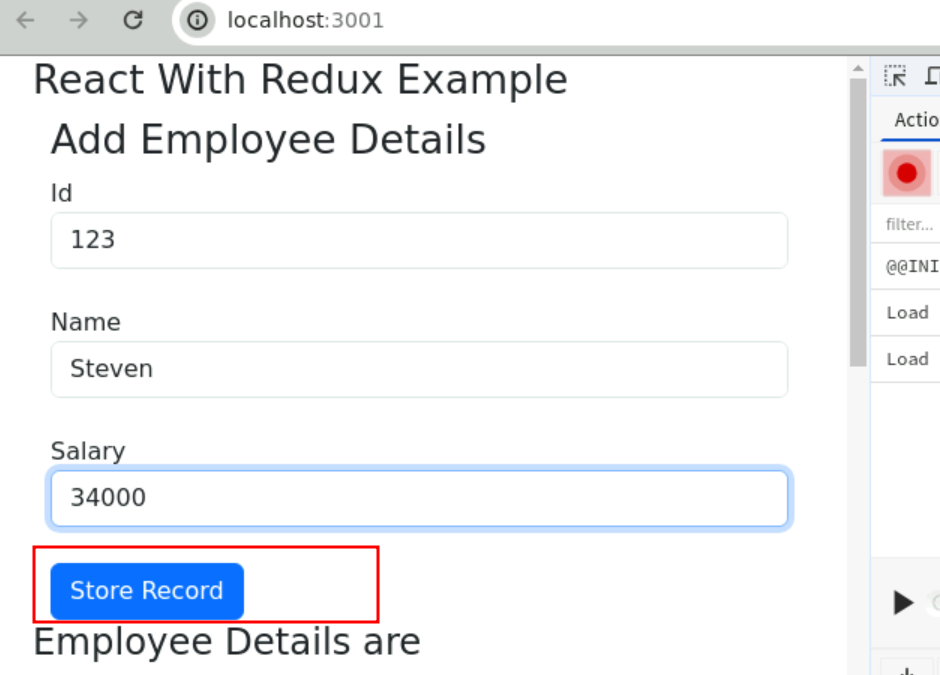
9.2 now you can check all employee details in redux dev tools



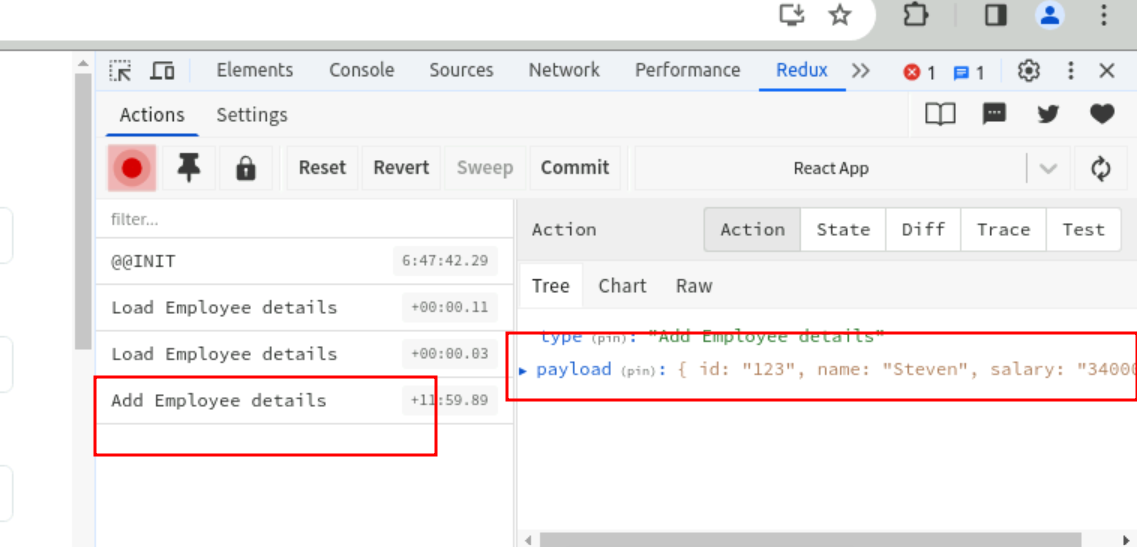
9.3 You can see the state details



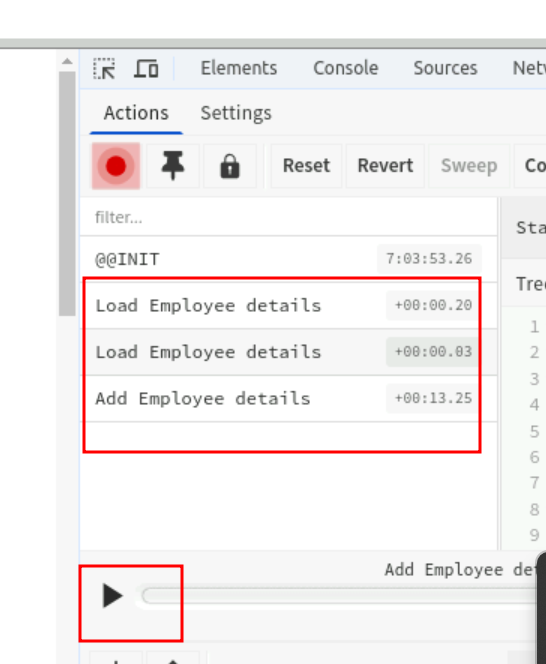
9.4 Now you can add the employee details and check the details in redux dev tools.



9.5 Now redux tool got the details about action as add employee details and payload the employee information



9.6 You can play the operation recoding and time taken to do that operation



9.6 you can see the state information in different format

