**Lesson 10 Demo 03**

**React-redux-testing**

**Objective:** To demonstrate the react redux testing redux module responsible to add the names and display the names.

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

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

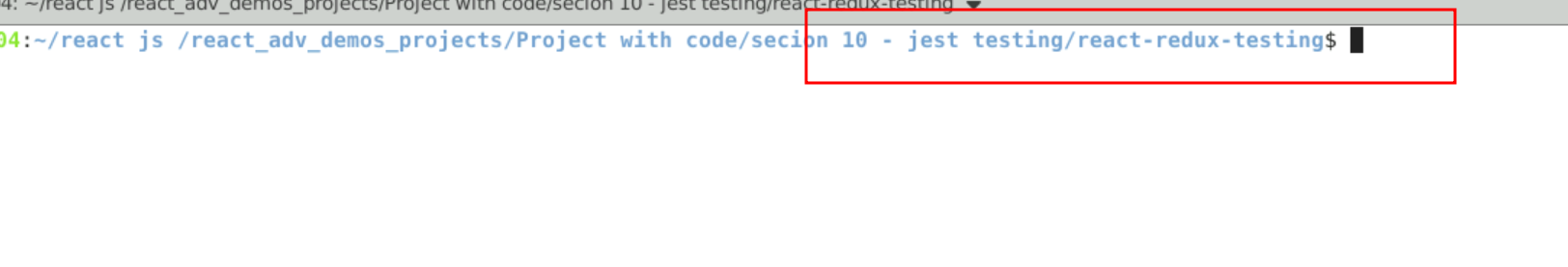
**Note** : All react js project already created with version 18.x with Sample App.js file

**Steps to be followed:**

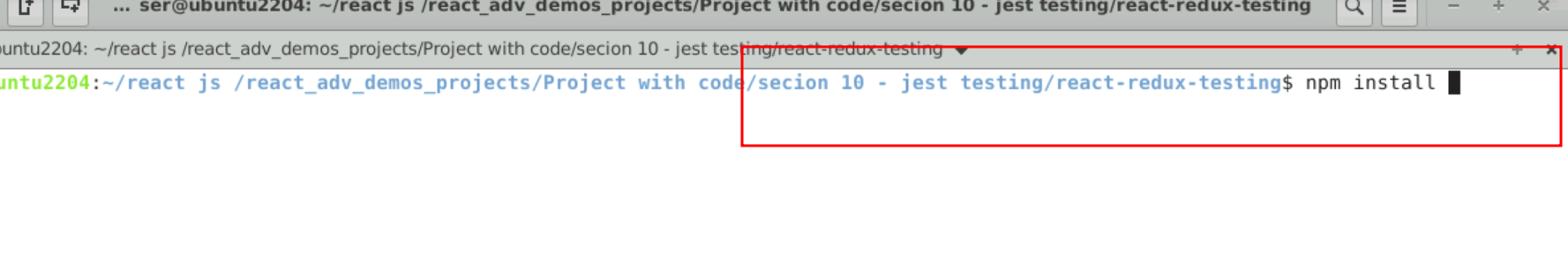
1. Set up for react jest js project
2. verify testing dependencies in package.json file
3. create slice file
4. create store file
5. create user defined component ie OfficeCharacter
6. create App.test.js file
7. index.html file configure bootstrap features.
8. Now we test the application.

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

1. Open the terminal window inside the **react-redux-testing** folder



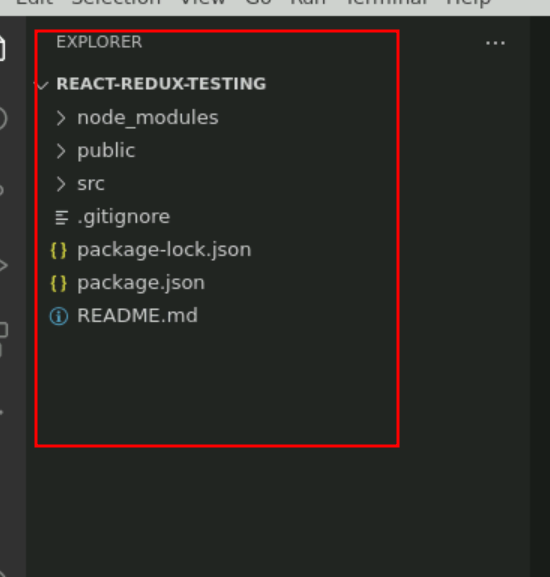
1.2 run npm install to download the dependencies.



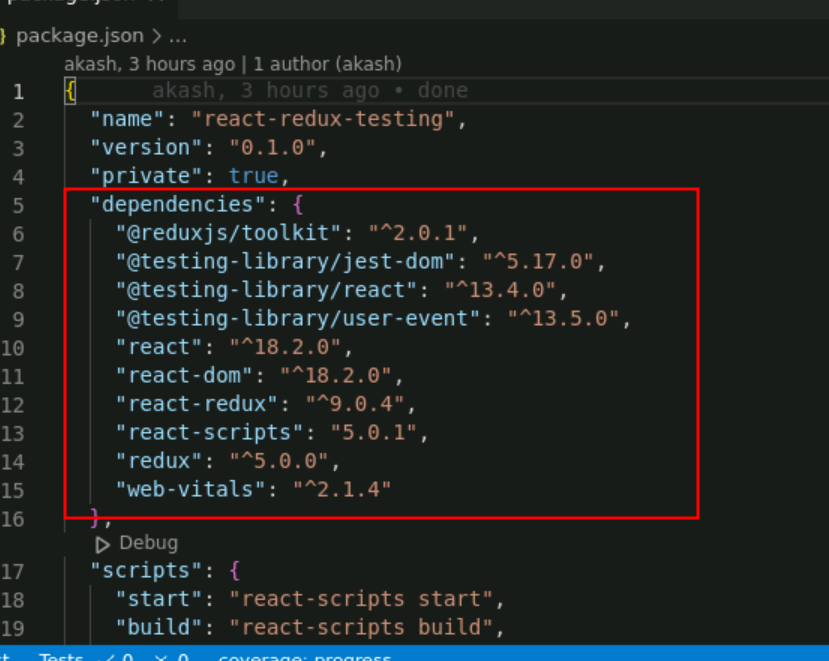
* 1. Open the project in VS code

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

* 1. view the project structure



**Step 2 : verify testing dependencies and redux in package.json file**

****

**Step 3: create slice file**

3.1 create characterSlice.js file which provide add and remove functionality names.

**characterSlice.js**

import { createSlice } from "@reduxjs/toolkit";

const charactersSlice = createSlice({

name: "characters",

initialState: [],

reducers: {

add(state, action) {

state.unshift(action.payload);

},

remove(state, action) {

state.splice(state.indexOf(action.payload), 1);

}

}

});

export default charactersSlice;

**Step 4 : create store file**

4.1 create buildStore.js file . this file provide the store details with initial state values.

import { configureStore } from "@reduxjs/toolkit";

import charactersSlice from "./charactersSlice";

export default function buildStore() {

const store = configureStore({

reducer: charactersSlice.reducer,

preloadedState: ["Michael Scott", "Dwight Schrute", "Jim Halpert"]

});

return store;

}

**Step 5 : create user defined component ie OfficeCharacter**

5.1 In App.js file create the : This file responsible to create OfficeCharacter component which is responsible to add , delete and display names with help of useSelector and useDispach hook. App component configure with store.

import { useState } from "react";

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

import charactersSlice from './charactersSlice';

import buildStore from "./buildStore";

import "./index.css";

const store = buildStore();

const actions = charactersSlice.actions;

export default function App() {

return (

<Provider store={store}>

<OfficeCharacters />

</Provider>

);

}

export function OfficeCharacters() {

const dispatch = useDispatch();

const characters = useSelector((state) => state);

const [newCharacter, setNewCharacter] = useState("");

function add(e) {

e.preventDefault();

dispatch(actions.add(newCharacter));

setNewCharacter("");

}

function deleteCharacter(character) {

dispatch(actions.remove(character));

}

return (

<div className="container">

<form onSubmit={add} className="form-group">

<div className="row">

<div className="col-2">

<label htmlFor="newCharacter">New Names</label>

</div>

<div className="col-3">

<input

type="text"

id="newCharacter"

value={newCharacter}

onChange={(e) => setNewCharacter(e.target.value)}

className="form-control"

/>

</div>

<div className="col-3">

<button className="btn btn-success">Add Names</button>

</div>

</div>

</form>

<ol>

{characters.map((character, i) => (

<li key={i} data-testid="character">

<div className="row offset-3">

<div className="col-3"> <span data-testid="name">{character}</span>{" "}</div>

<div className="col-3"> <button type="button" onClick={() => deleteCharacter(character)} className="btn btn-danger">

Delete

</button></div>

</div>

</li>

))}

</ol>

</div>

);

}

This file testing with valid emailed and password as well as invalid emailed and password.

**Step 6 : create App.test.js file**

6.1 inside this file we are taking the help of testing library to do the testing with help of jest of react redux store.

**App.test.js**

import { render, screen, within, fireEvent } from "@testing-library/react";

import { Provider } from "react-redux";

import buildStore from "./buildStore";

import { OfficeCharacters } from "./App";

import charactersSlice from "./charactersSlice";

describe("OfficeCharacters", () => {

  function renderOfficeCharacters() {

    const store = buildStore();

    render(

      <Provider store={store}>

        <OfficeCharacters />

      </Provider>

    );

    return {

      newCharacter: screen.getByLabelText("New Names"),

      addButton: screen.getByText("Add Names"),

      getCharacters() {

        return screen.getAllByTestId("character").map((item) => ({

          name: within(item).getByTestId("name").textContent,

          deleteButton: within(item).getByText("Delete")

        }));

      }

    };

  }

  it("should add a character", () => {

    const { newCharacter, addButton, getCharacters } = renderOfficeCharacters();

    const pam = "Pam Beesly";

    // verify pam is NOT in the initial list

    expect(

      getCharacters().find((character) => character.name === pam)

    ).toBeFalsy();

    // add pam

    fireEvent.change(newCharacter, { target: { value: pam } });

    fireEvent.click(addButton);

    // verify pam is first in the list

    expect(

      getCharacters().findIndex((character) => character.name === pam)

    ).toBe(0);

  });

  it("should delete a character", () => {

    const { getCharacters } = renderOfficeCharacters();

    const jim = "Jim Halpert";

    const deleteJim = getCharacters().find(

      (character) => character.name === jim

    ).deleteButton;

    // delete character

    fireEvent.click(deleteJim);

    // verify character is NOT in list

    expect(

      getCharacters().find((character) => character.name === jim)

    ).toBeFalsy();

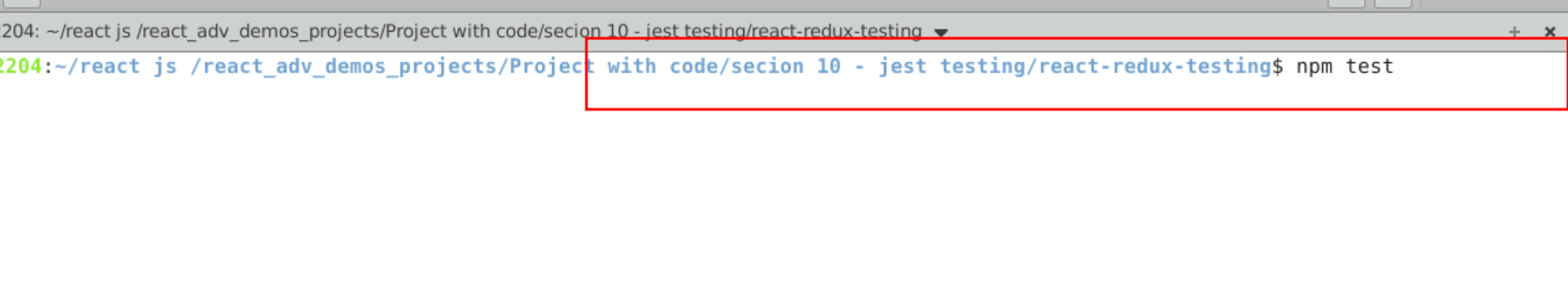
  });

});

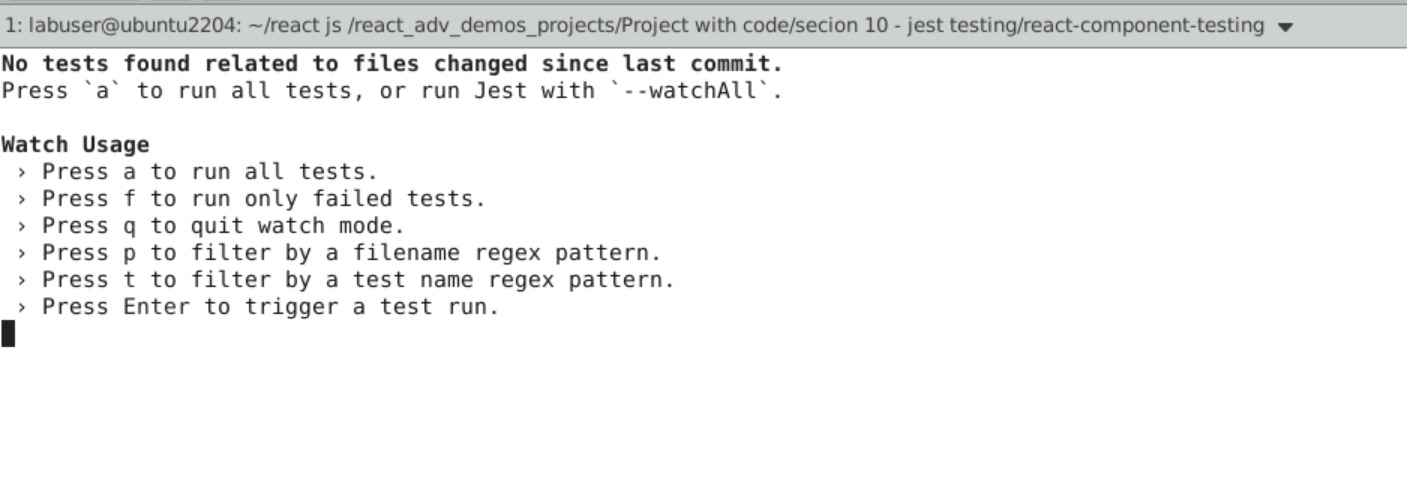
Step 7 : Now we test the application.

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

**npm test**

****

7.2 You can view the output on console.



Press a button to run all test a

