**Lesson 05 Demo 01**

**Airport-search-react-redux-thunk-app**

**Objective:** To demonstrate the react with redux using thunk module to consume rest api from json file. Search particular Airport details using react js application. to store employee details, view employee details as well delete employee information from store.

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

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

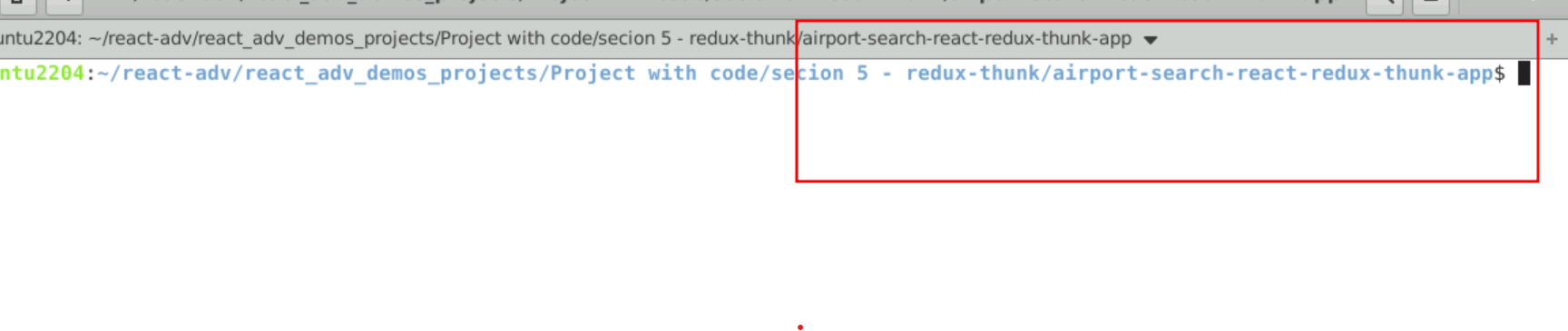
**Note** : All react js project already created with version 18.x with Sample App.js file

**Steps to be followed:**

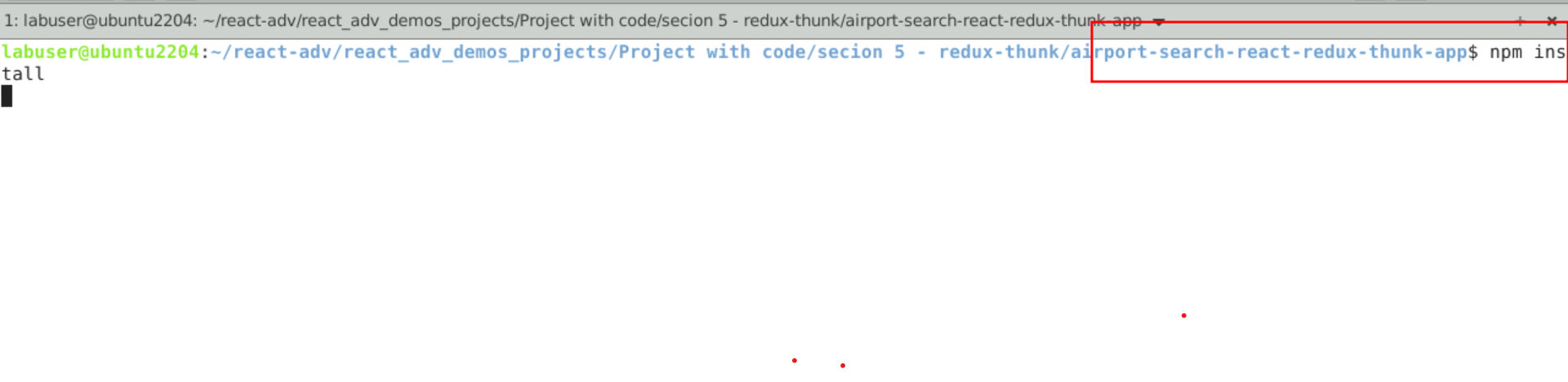
1. Set up for react js project
2. Creating airport.json file which contains json data.
3. Crate service folder which contains two sub folder ie axios and redux.
4. Axios folder contains index.js file
5. Redux folder contains actions as well as reducers folder.
6. Actions folder contains index.js file
7. Reducer folder contains index.js and fetchAPI.js file.
8. Create component folder which contains App.js main component as well as App.css file
9. In index.js file ie outside all folder provide store details.
10. Test the application.

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

1. Open a terminal window inside a React JS pre-created project ie **aiport-search-react-redux-thunk-app**

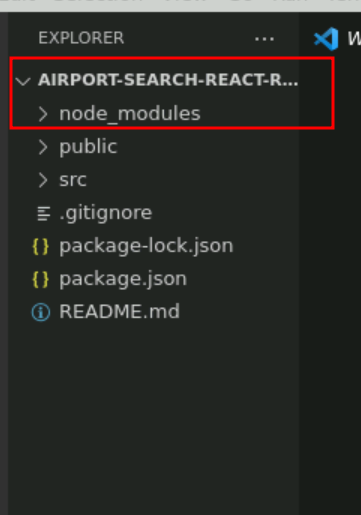


1. Now you need to run the command as **npm install.** This command helps us to installed all required dependencies mention in package.json file in local machine in the form of node\_module folder.

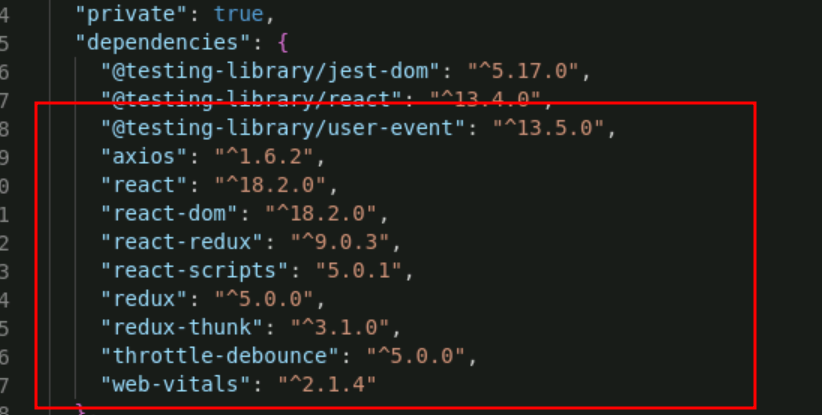


1. Now open **aiport-search-react-redux-thunk-app** folder in VS Code Editor

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



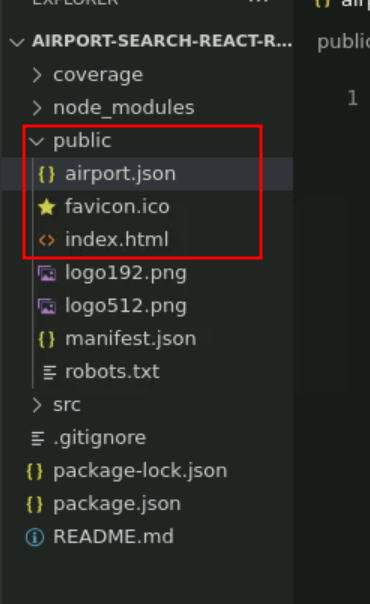
1.4 now open package.json file and view external dependencies.

****

**Step 2:** Creating airport.json file which contains json data.

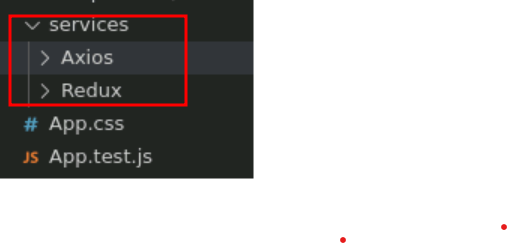
2.1Create or check static json file inside a public folder. Which contains collection of aiport details in the form of json which help to retrieve the using thunk module in react application.

**aiport.json file**



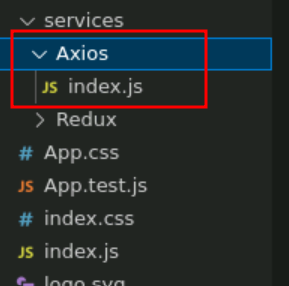
**Step 3: Create service folder which contains two sub folder ie axios and redux.**

3.1 create service folder which contains two sub folder ie axios and redux.



**Step 4 : Axios folder contains index.js file**

4.1 create index.js fie inside a Axios folder.



4.2 index.js file contains code which is responsible to interact with airport.json file present inside public folder with help of axios third party library. Using airportSearch function we can load statis airport details present in airport.json file. Please copy paste the below code in index.js file part of Axios folder.

index.js

import { getAirports } from '../Redux/actions';

import axios from 'axios';

function aiportSearch() {

return (dispatch) => {

axios.get(window.location.origin + '/airport.json')

.then(response => {

dispatch(getAirports(response.data))

})

.catch(error => {

throw (error);

});

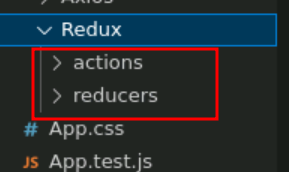
};

}

export { aiportSearch }

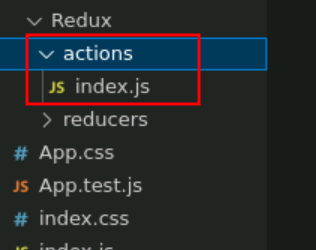
**Step 5 : Redux folder contains actions as well as reducers folder.**

5.1 insider Redux folder you need to create the actions and reducer folder.



**Step 6 : Actions folder contains index.js file**

6.1 Inside Action folder create the index.js file



6.2 The index file part of actions folder which contains action to do by reducer with help of thunk modules. Which return type of action and payload.

index.js

export function getAirports(data) {

return {

type: "FETCH\_AIRPORTS",

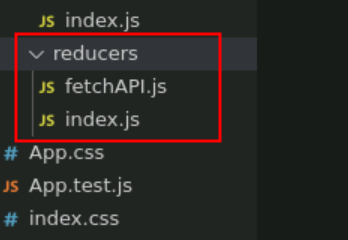
payload: data

};

}

**Step 7 : Reducer folder contains index.js file**

7.1 Inside reducer folder create index.js and fetchAPI.js file



7.2 fetchAPI.js file is type of reducer file which is responsible to take two parameter is state and action. And base upon action it will do some operation on state. In this example we are not doing any changes in state variable. We are loading the data from airport.json file.

**fetchAPI.js**

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

switch (action.type) {

case "FETCH\_AIRPORTS":

return action.payload;

default:

return state;

}

}

7.3 create index.js file which combine all reducer function depending upon our requirements. Which file interact with reducer file and provide all reducer function details to store.

**index.js**

import { combineReducers } from 'redux';

import fetchAPI from '../reducers/fetchAPI';

const allReducers = combineReducers({

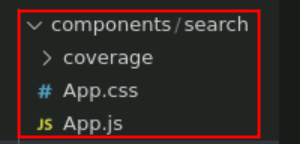
fetchAPI

});

export default allReducers;

**Step 8 : Create component folder which contains App.js main component as well as App.css file**

8.1 Now create component folder which contains App.js and App.css file.



8.2 Now create App.js file which is main component in the project which take the help of useSeletor() and useDispatch() hook to interact with reducer.

App.js

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

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

import { debounce } from 'throttle-debounce';

import \* as API from '../../services/Axios';

import './App.css';

function App() {

const [airports, setAirports] = useState([]);

const [airport, setAirport] = useState({});

const [resultAvailable, setResult] = useState(false);

const [loading, setLoading] = useState(true);

const [selected, setSelected] = useState(false);

const results = useSelector(state => state.fetchAPI);

const dispatch = useDispatch();

useEffect(() => {

if (!results.data) {

fetch();

} else {

if (results.data.length > 0) {

fetched();

}

}

if (airport.city !== undefined) {

changeTitle();

}

});

let fetch = () => {

dispatch(API.aiportSearch());

}

let fetched = () => {

setLoading(false);

}

let changeTitle = () => {

document.title = `Searched Airport - ${airport.airport}`;

}

let searchAirports = debounce(500, (input) => {

setSelected(false);

let data = [...results.data];

if (input.length > 1) {

setAirports(data.filter(e => e.airport.toLowerCase().includes(input.toLowerCase()) || e.city.toLowerCase().includes(input.toLowerCase()) || e.iata.toLowerCase().includes(input.toLowerCase())));

setResult(true);

} else if (input.length === 1) {

setAirports(data.filter(e => e.airport.charAt(0).toLowerCase() === input.toLowerCase() || e.city.charAt(0).toLowerCase() === input.toLowerCase() || e.iata.charAt(0).toLowerCase() === input.toLowerCase()));

setResult(true);

} else if (input.length === 0) {

setAirports([]);

setResult(false);

setSelected(false);

}

});

let handleInput = (e) => {

let input = e.target.value.trim().toLowerCase();

searchAirports(input);

}

let selectAirport = (item) => {

setSelected(true);

setResult(false);

setAirport({

airport: item.airport,

city: item.city,

iata: item.iata

});

}

return (

<div style={{ outline: 'none', border: 0 }}>

{loading === false &&

<div style={{ outline: 'none', border: 0 }}>

<div style={{ width: '100%', display: 'block' }}>

<input

type="text"

placeholder="Enter Airport Name, Code or City Name"

className="Search"

// value={keyword}

onChange={e => handleInput(e)} />

</div>

<div className="Gap"></div>

<h5 style={{ marginTop: 10, marginBottom: 10, fontSize: 15, color: '#f0ad4e', textAlign: 'center' }}>

{resultAvailable === true && "Search Results"}

{selected === true && "Selected Airport"}

</h5>

{selected === true &&

<div className="Results">

<div style={{ marginTop: 0, padding: 10 }} onClick={() => setSelected(true)}>

<div style={{ width: '100%', display: 'block' }}>

<span style={{ fontWeight: 'bold' }}>{airport.city}</span>

<span style={{ float: 'right' }}>{airport.iata}</span>

</div>

<p style={{ marginTop: 5, marginBottom: 0, paddingBottom: 5, color: '#777', borderBottom: '0.5px solid #9997' }}>{airport.airport}</p>

</div>

</div>

}

{selected === false && resultAvailable === true && airports.map((item, i) => (

<div className="Results" key={i}>

<div style={{ marginTop: 0, padding: 10 }} id="Select" onClick={() => selectAirport(item)}>

<div style={{ width: '100%', display: 'block' }}>

<span style={{ fontWeight: 'bold' }}>{item.city}</span>

<span style={{ float: 'right' }}>{item.iata}</span>

</div>

<p style={{ marginTop: 5, marginBottom: 0, paddingBottom: 5, color: '#777', borderBottom: '0.5px solid #9997' }}>{item.airport}</p>

</div>

</div>

))

}

{

selected === false && resultAvailable === true && airports.length === 0 &&

<p style={{ textAlign: 'center' }}>No Result Found</p>

}

</div>

}

</div>

);

}

export default App;

8.3 Now create App.css file which contains css rules.

.Search {

font-size: 16px;

position: absolute;

top: 0;

right: 0;

left: 0;

display: block;

margin-right: auto;

margin-left: auto;

}

.Search:focus {

border: 2px solid #666666;

outline: none;

}

.Gap {

border: 0px;

background-color: #fff;

}

@media only screen and (max-width: 767px) {

.Search {

width: 92%;

padding: 4%;

border: 0px;

box-shadow: 0px 2px 4px -1px rgba(0, 0, 0, 0.2), 0px 4px 5px 0px rgba(0, 0, 0, 0.14), 0px 1px 10px 0px rgba(0, 0, 0, 0.12);

}

.Search:focus {

border: 0px;

outline: none;

}

.Gap {

margin-bottom: 55px;

}

.Results {

width: 100%;

border: 0px;

}

}

@media only screen and (min-width: 768px) {

.Search {

width: 75%;

padding: 15px;

margin-top: 10px;

border: 2px solid #999999;

border-radius: 4px;

}

.Gap {

margin-bottom: 65px;

}

.Results {

width: 75%;

margin-top: 10px;

border: 0px;

display: block;

margin-right: auto;

margin-left: auto;

}

}

#Select:hover {

background-color: #eeeeee

}

**Step 9. In index.js file ie outside all folder provide store details.**

9.1 Now in main file ie index.js file which responsible to render App component to browser. Inside this file we need to configure store and thunk details.

import React from 'react';

import ReactDOM from 'react-dom/client';

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

**import {thunk} from 'redux-thunk';**

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

**import allReducers from '../src/services/Redux/reducers/index';**

import App from './components/search/App';

import './index.css';

import reportWebVitals from './reportWebVitals';

**const store = createStore(allReducers,applyMiddleware(thunk));**

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

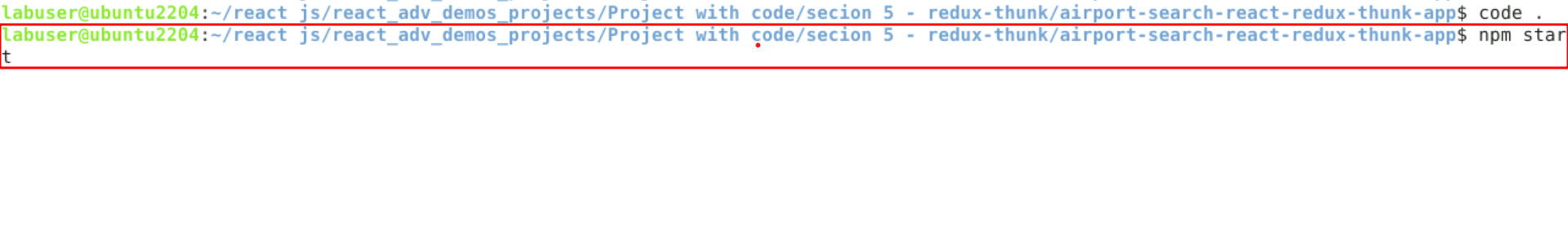
// to log results (for example: reportWebVitals(console.log))

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

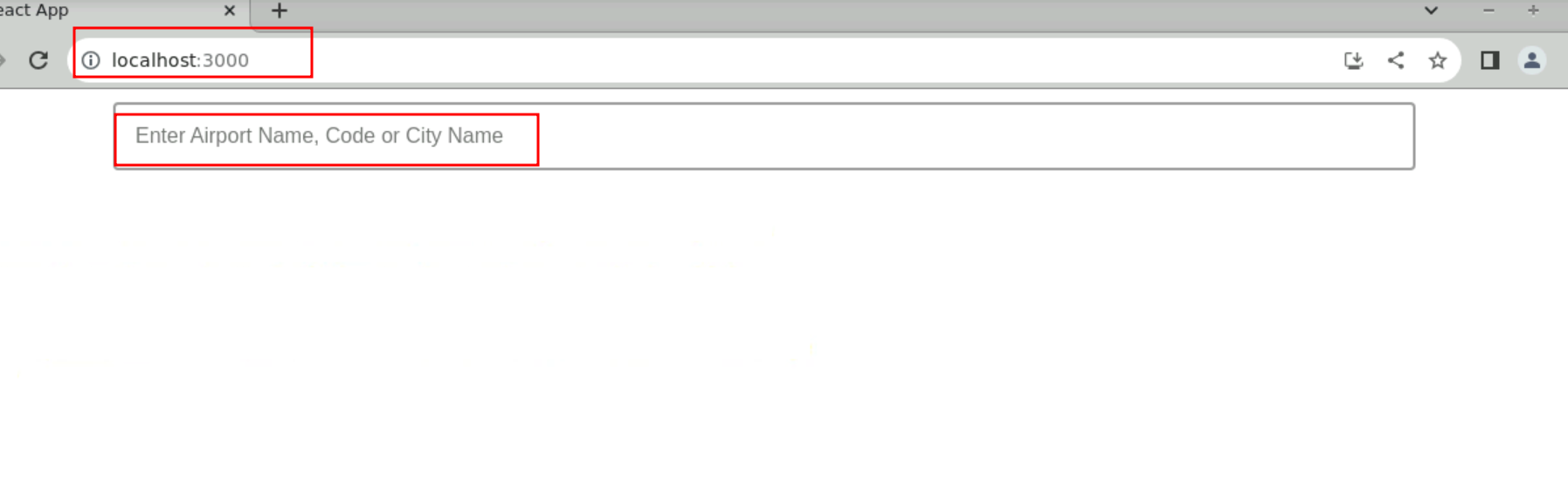
reportWebVitals();

**Step 10 Now we run the application using npm start**

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



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

**10.3** Please write aiport name or code or city name to get the details.

