**Lesson 07 Demo 02**

**Creating an Airport Search React Redux Thunk Application**

**With Debugger**

**Objective:** To create a React application with Redux and Thunk middleware, consuming a REST API to retrieve details based on inputs such as airport name, code, or city name input in the Redux store

**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 airport.json file
3. Create components and services
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**

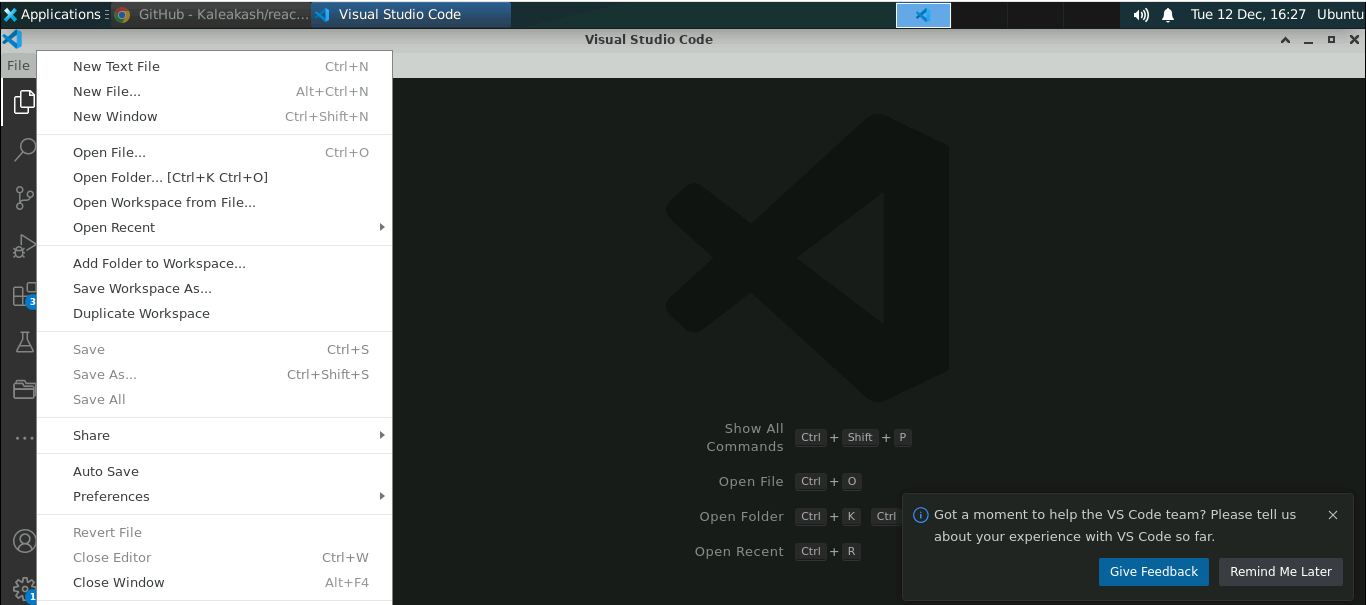
* 1. Open a terminal window to run the following command to create a React application:

**npx create-react-app airport-search-react-redux-thunk-app**

A screenshot of a computer

Description automatically generated

* 1. Open the created React application folder (**airport-search-react-redux-thunk-app**) 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 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: Create an airport.json file**

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

A screenshot of a computer

Description automatically generated

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

A screen shot of a computer

Description automatically generated

* 1. Inside the **airport.json** file, add the following JSON data, as shown below:

A screen shot of a computer

Description automatically generated

**Step 3: Create components and services**

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

A screen shot of a computer

Description automatically generated

1. Inside the **services** folder, create two subfolders named **Axios** and **Redux**

**A computer screen with a black x

Description automatically generated**

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

**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 => {**

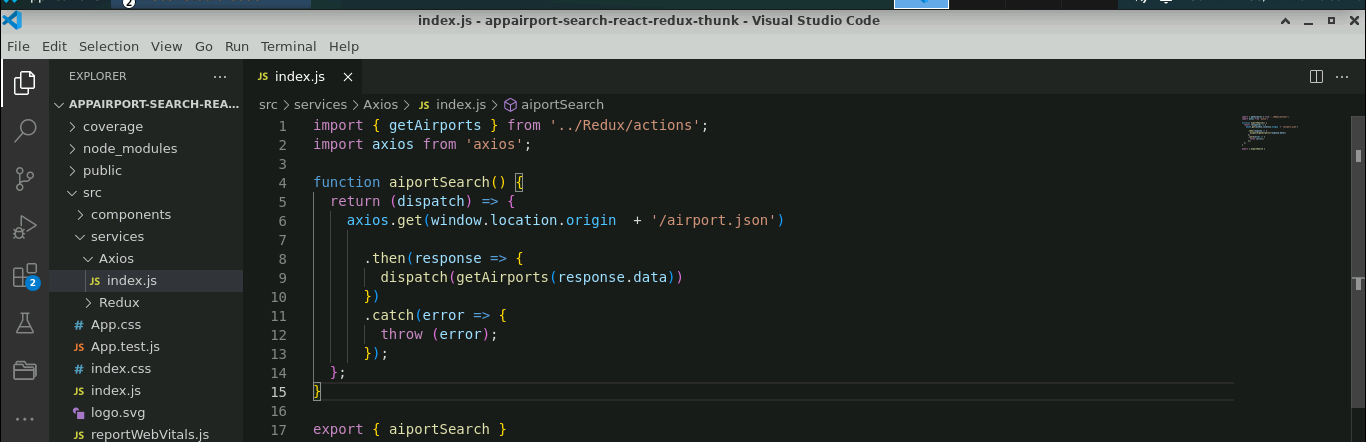
**throw (error);**

**});**

**};**

**}**

**export { aiportSearch }**

****

* 1. Inside the **Redux** folder, create two subfolders named **actions** and **reducers**

**A screen shot of a computer

Description automatically generated**

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

**export function getAirports(data) {**

**return {**

**type: "FETCH\_AIRPORTS",**

**payload: data**

**};**

**}**

**A screenshot of a computer

Description automatically generated**

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

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

**switch (action.type) {**

**case "FETCH\_AIRPORTS":**

**return action.payload;**

**default:**

**return state;**

**}**

**}**

**A screenshot of a computer

Description automatically generated**

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

**import { combineReducers } from 'redux';**

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

**const allReducers = combineReducers({**

**fetchAPI**

**});**

**export default allReducers;**

**A screenshot of a computer

Description automatically generated**

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

**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;**

**A screenshot of a computer screen

Description automatically generated**

* 1. Inside the **reducers** folder, create a file named **App.css** and enter the following code. It 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**

**}**

**A screenshot of a computer

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 React from 'react';**

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

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

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

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

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

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

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

**//import \* as serviceWorker from './serviceWorker';**

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

**const enhancer = composeEnhancers(**

**applyMiddleware(thunk),**

**// other store enhancers if any**

**);**

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

**ReactDOM.render(**

**<React.StrictMode>**

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

**<App />**

**</Provider>**

**</React.StrictMode>,**

**document.getElementById('root')**

**);**

**// If you want your app to work offline and load faster, you can change**

**// unregister() to register() below. Note this comes with some pitfalls.**

**// Learn more about service workers: https://bit.ly/CRA-PWA**

**//serviceWorker.register();**

A screen shot of a computer

Description automatically generated

**Step 5: Run the application**

* 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:

**A screenshot of a computer

Description automatically generated**

* 1. Enter the **Airport Name**, **Code,** or **City Name** to get the details

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 **>>** and choose **Redux** from the options

A computer screen shot of a computer screen

Description automatically generated

The output appears as shown below:

A screenshot of a computer

Description automatically generated

* 1. Click on **payload**

A screenshot of a computer

Description automatically generated

The output appears as shown below:

A screenshot of a computer

Description automatically generated

* 1. Click on **State** to get the global state details

A screenshot of a computer

Description automatically generated

With this, you have successfully created a React application with Redux and Thunk middleware, consuming a REST API to retrieve details based on airport name, code, or city name input in the Redux store while utilizing React Redux DevTools for debugging and ensuring application robustness.