**Lesson 05 Demo 02**

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

**Objective:** To create a React application for airport search with Redux Thunk, enabling users to efficiently search and select airports by name, code, or city

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

**Prerequisites:** None

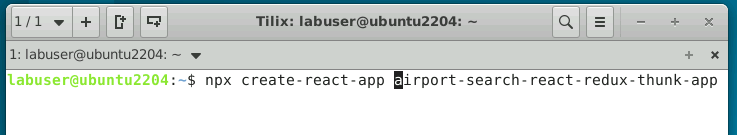
Steps to be followed:

1. Create and set up the React project
2. Create anairport.json file
3. Create a services folder
4. Create a component folder
5. Configure the index.js file
6. 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 airport-search-react-redux-thunk-app**



* 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**

**A screenshot of a computer

Description automatically generated**

* 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 screen shot of a computer

Description automatically generated**

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

**A screen shot 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. In the **airport.json** file, enter the following JSON data:

**A screen shot of a computer

Description automatically generated**

**Step 3: Create a services folder**

1. Inside the **src** folder, create a **services** folder

A screen shot of a computer

Description automatically generated

1. Create two subfolders named **Axios** and **Redux**

A screen shot of a computer

Description automatically generated

1. Inside an **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 => {**

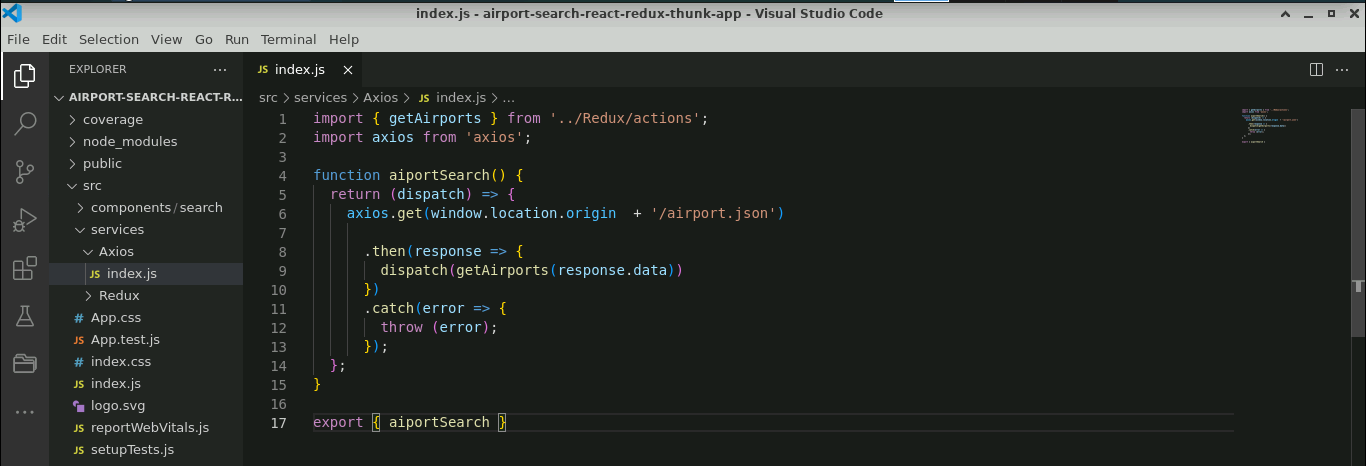
**throw (error);**

**});**

**};**

**}**

**export { aiportSearch }**

****

1. Inside a **Redux** folder, create an **action** and a **reducers** folder

A screenshot of a computer

Description automatically generated

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

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

**switch (action.type) {**

**case "FETCH\_AIRPORTS":**

**return action.payload;**

**default:**

**return state;**

**}**

**}**

**A screen shot of a computer

Description automatically generated**

1. Inside the **reducers** file, create an **index.js** file 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**

**Step 4: Create a component folder**

1. Inside the **src** folder, create a folder named **components/search** that contains **App.js** and **App.css** files

**A screenshot of a computer

Description automatically generated**

1. Modify the **App.js** file that includes functionality for searching and selecting airports, as shown below. It utilizes **React hooks**, **Redux**, and **Axios** for handling asynchronous API requests.

**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>**

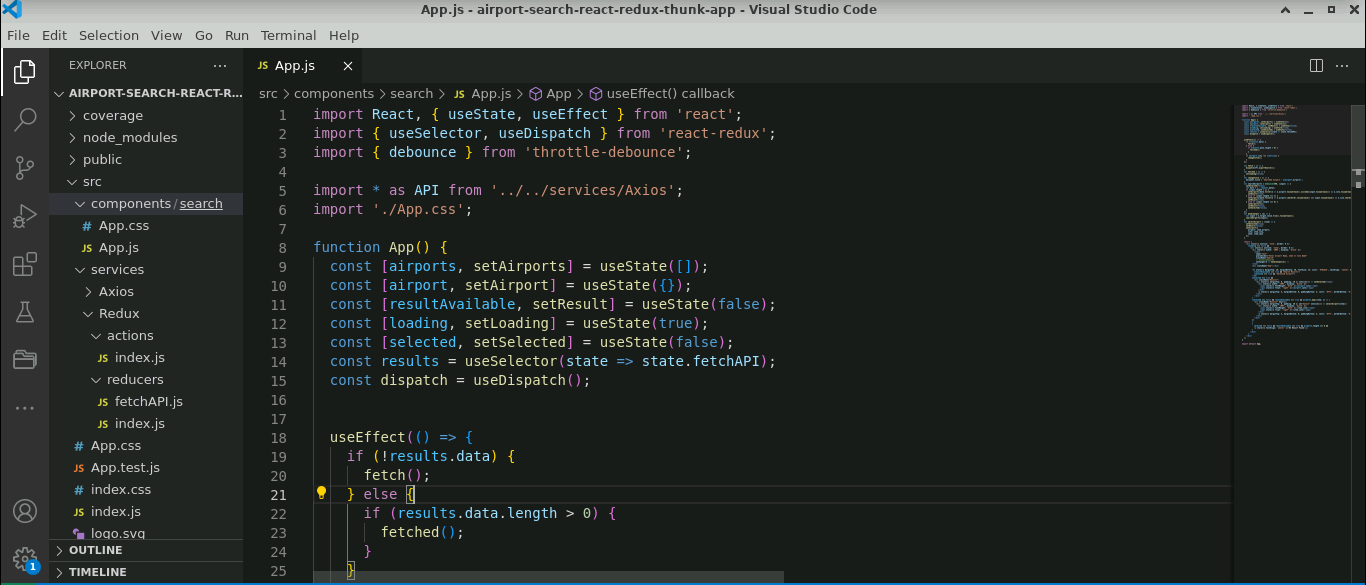
**}**

**</div>**

**);**

**}**

**export default App;**

****

1. Modify the **App.css** file as shown below. It has the **CSS** code that defines styles for the search functionality in the React component.

**.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 screen shot of a computer

Description automatically generated**

**Step 5: Configure the index.js file**

1. Inside the **src** folder, configure the store and thunk details inside the **index.js** file, as shown below:

**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>**

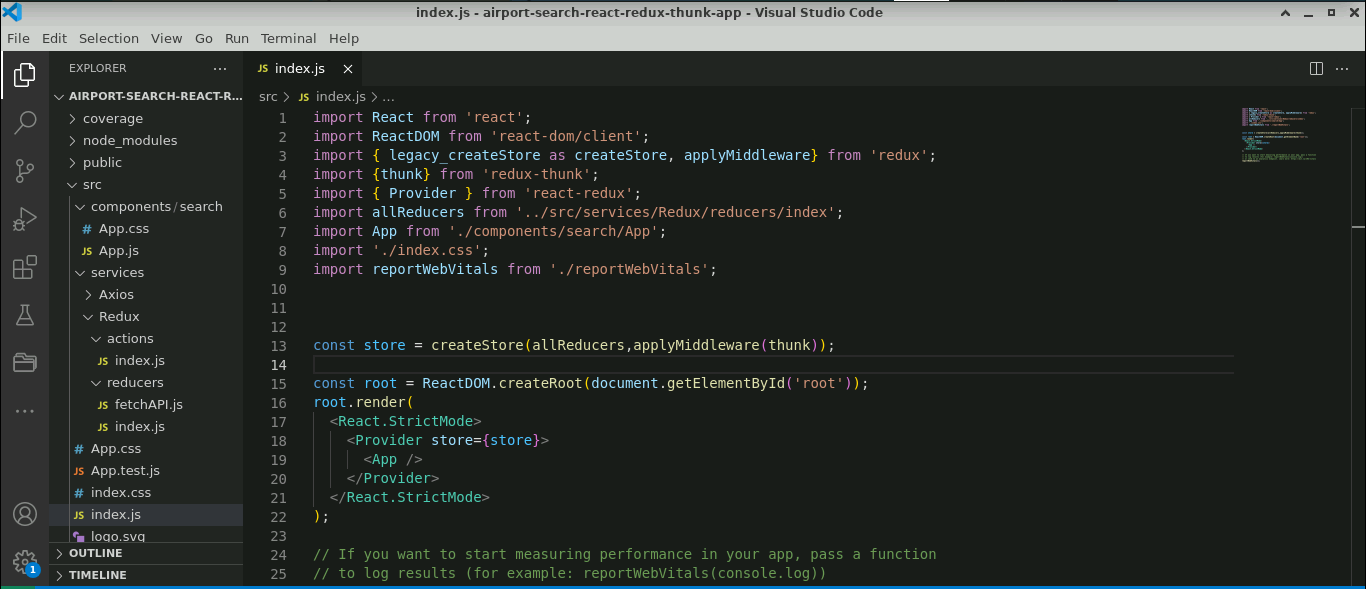
**);**

**// 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 6: Test the application**

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

**npm start**

**A screenshot 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** in the tab (refer to the above screenshot) to get the airport details

A screenshot of a computer

Description automatically generated

* 1. Enter the **Code** in the tab

A white screen with orange text

Description automatically generated

* 1. Enter the **City Name** in the tab

A screenshot of a computer

Description automatically generated

With this, you have successfully created a React application for airport search, utilizing Redux Thunk, and Axios for efficient state management and asynchronous API requests.