Contact Management App with Charts and Maps Project Overview

Introduction

This project is a contact management application combined with a COVID-19 dashboard. The app is built using ReactJS, TypeScript, TailwindCSS, React Router v6, and React Query (TanstackQuery). It allows users to manage contacts and view COVID-19 data through interactive charts and maps.

Objectives

Contact Management:

Add new contacts. Display a list of all contacts. View, edit, and delete contact details. Store contact data using Redux.

COVID-19 Dashboard:

Display a line graph showing case fluctuations. Show a map with markers indicating country-specific COVID-19 data.

Project Structure

The main components involved are:

ContactList:

Displays a list of contacts with options to view, edit, or delete each contact.
ContactForm:

Provides a form to create or edit contacts.

ContactDetails: Displays detailed information about a specific contact.
Components and Their Functionality

1. **ContactList Component**

Path: components/ContactList.tsx

Display the list of contacts with options to view, edit, and delete them.

Key Parts:

useDispatch and useSelector: These hooks are used to interact with the Redux store. useDispatch is used to dispatch actions (e.g., delete or edit a contact), and useSelector is used to select state from the Redux store.

useNavigate: This hook from React Router is used to programmatically navigate between routes.

Contacts Mapping: The list of contacts is retrieved from the Redux store and mapped to display each contact with its details, along with Edit and Delete buttons.

2. ContactForm

Component

Path: components/ContactForm.tsx

Purpose:

Provide a form to create a new contact or edit an existing contact.

Key Parts:

State Management: useState is used to manage form data.

useEffect: Used to pre-fill the form if a contact is being edited.

Form Submission: The form handles both adding a new contact and updating an existing one. The handleSubmit function manages form submission and dispatches the appropriate Redux actions.

3. ContactDetails Component

Path: components/ContactDetails.tsx

Purpose:

Display detailed information about a specific contact.

Key Parts:

1)**useParams:** This hook retrieves the id parameter from the URL to identify which contact to display.

2)useSelector: Used to select the list of contacts from the Redux store. Conditional Rendering: If the contact is not found, a message is displayed. Otherwise, the contact's details are shown

Redux Actions and Reducers Actions

Path: actions/contactAction.tsx

Purpose: Define actions for adding, editing, and deleting contacts.

Key Parts: Action Creators: Functions that return action objects, e.g., addContact, editContact, deleteContact.

Reducers

Path: reducers/contactReducer.tsx

Purpose:

Handle the state changes based on dispatched actions.

Key Parts:

1)Initial State: The initial state of the contacts list.

2)Reducer Function: A function that updates the state based on the action type.

Routing

Path: App.tsx

Purpose:

Define routes for the application using React Router.

Key Parts:

Routes: Define paths for different components, e.g., /contacts for ContactList, /contacts/:id for ContactDetails, and /contact/create for ContactForm.

Contact Reducer

The contactReducer manages the state related to contacts, including the list of contacts and the selected contact for editing. It handles actions for adding, editing, and deleting contacts.

Actions The reducer handles three types of actions:

1)ADD_CONTACT: Adds a new contact to the list of contacts.

2)EDIT_CONTACT: Edits an existing contact in the list of contacts.

3)DELETE_CONTACT: Deletes a contact from the list of contacts.

Routes:

The Routes component from react-router-dom defines the different routes in the application:

1) /: The home page, rendered by the HomePage component.

2) /home: Another path for the home page, also rendered by the HomePage component.

3) /dashboard: The dashboard page, rendered by the Dashboard component.

4) /contacts: The contact list page, rendered by the ContactList component.

5) /contacts/:id: The contact details page, rendered by the ContactDetails component, which uses the id parameter.

6) /contact/create: The contact form page, rendered by the ContactForm component.

Charts and Graphs

Dashboard Component Overview

This React component, Dashboard, fetches and displays global COVID-19 statistics, country-specific COVID-19 data, and historical COVID-19 data using charts and maps. The component uses react-query to fetch and cache data from APIs, react-leaflet to display maps, and chart.js to create a line chart for historical data.

Fetching Data and Handling API Responses

The fetchData function is used to fetch data from the specified URL. Here's how it works:

Async Function:

The function is defined as async, allowing the use of await to handle asynchronous operations.

Fetching Data:

1)const response = await fetch(url);
(Fetches the data from the given URL.)

2)if (!response.ok) { throw new Error('Network response was not ok'); }:
(Checks if the response is OK. If not, it throws an error.)

Return Data: return response.json();: Parses the response as JSON and returns it.

Using react-query

react-query is used to fetch and cache the data:

1)Global Data:

useQuery<WorldData>('worldData', () =>
fetchData('https://disease.sh/v3/covid-19/all'));: Fetches global COVID-19 statistics.

2) Country Data:

useQuery<CountryData[]>('countryData', () =>
fetchData('https://disease.sh/v3/covid-19/countries'));

(Fetches country-specific COVID-19 statistics.)

3)Graph Data:

useQuery<GraphData>('graphData', () =>
fetchData('https://disease.sh/v3/covid-19/historical/all?lastdays=all'));

(Fetches historical COVID-19 data.)