This document has a Task Description that explains the code challenge. It contains the details of each functionality and the associated scenarios. If you have any questions, feel free to reach out.

Do not spend too much time on the aesthetics, because we are looking at the code structure and how easy it is to extend functionalities within the codebase.

Task Description

Create a patient search tool. It should have 2 pages: a list page and a detail’s page.
The list page should have a search form and a list while the detail’s page should have the patient's details.

List Page

Search Functionality

1. A search form that takes in a query and returns a patients’ list. A patient can be searched by name, ID or email (use the data in the attached JSON file)
2. It should have two filters: a sex filter (male and female) and an age filter(“18 - 30”, “31 - 45” and “ > 45”)
3. An alphabetical sorting option

Note that the search data should **persist,** that is the data should not be lost, even when you switch components or routes.

Scenario

- A user visits the list page and sees a list of patients along with a search form that comprises: a search query input, a filter (with sex and age) and an alphabetical sorting tool.
- A user starts searching by typing a query inside the search input field.
- When the user stops typing, that is, after a 500 milliseconds delay, the query is used to search the data
- If a result is found, the data it returns should render a list of patient
- If there is no result, the user should see an error message
- If the user deletes every single search query from the input field, it should return the entire users in the list
- When the user selects the age filter, it should return the search result that contains only the selected age range
- When the user selects the sex filter, It should return the search result that contains only the selected sex
- When the user selects the age filter and the sex filter, it should return the search result that contains only the selected sex and selected age range.

List Functionality

1. List should only show the Patient ID and Full name

2. A route to the Patient's detail page

Scenario

- A user visits the app main page, the user should see a list of Patients.
- Each list item should have the patient's Full name and patient's ID
- When the user clicks on a Patient on the list
- The user is directed to the patient's detail page

Details Page

Go back Functionality

1. A go back button

Scenario

- A user visits the Patient's details page
- When the user clicks the go back button, the user is directed back to the list page. The search result, filter and sort data should persist.

Delete Functionality

1. A delete button to trigger delete confirmation modal
2. A delete confirmation modal with a confirm delete and a cancel button

Scenario

- A user visits the Patient's details page, the user should see the complete patient data.
- And it should have a delete button.
- When the user clicks on the delete button, a modal should pop up with a confirmation text: "Are you sure you want to delete `${userID}`"?
- When the user clicks on the confirm Delete button, the Patient is deleted
- And the user is directed back to the main page with the same search result, filter and sort data that were there previously without the deleted Patient.