Front-End Test Exercise

Goal

The goal of this exercise is to implement an *Admin* dashboard to manage the *Advisors* of a company. You should be able to obtain the *Advisors* list based on their average income amounts. Additionally, you will be able to create, update, and delete advisors.

Here are the design assets for the app.

The experience includes 3 core pages:

- Starting page to enter an income amount.
- *Advisor* list view with the result of the previous search.
- *Advisor* detail view.

And one shared component:

- Modal to create/edit *Advisors*.

Technical requirements

- You will develop the application using NextJS.
- Only use vanilla JavaScript, ReactJS, and vanilla CSS.
- You must complete the test without using any additional third-party library.
- You must consume the data dynamically via AJAX from the provided API.
    - `technical-server` (make sure to read the README file)

Extras: TypeScript and Unit Testing might be requested based on your previous experience.

Main logic

1. You create the first view with 1 input for the *Admin* to type an income value.
    a. 5 digits are the required length (input validation).
    b. A button to trigger the search.
2. Create a second view to render the search result.
    a. You must filter the pulled *Advisors* data and find the ones with the closest income value, based on the input. The search must filter pulled *Advisors* data by an average income that matches the specified threshold: from income -10,000 to income +10,000.
    b. Include a CTA to add a new *Advisor*.
3. Create a third view to render *Advisor* details.
    a. You land here by clicking on an *Advisor* from the previous list.
    b. Include an action button to edit the current *Advisor*.
4. Create a modal interface to edit an *Advisor*.

Final details

1. List a maximum of 10 *Advisors* per page, adding an option to paginate and show the next 10 *Advisors*.
2. Include controls to order *Advisors* (ascendent and descendent) by name and income.
3. In case of no found *Advisors*, an error message has to be shown to the *Admin*: “No available Advisors based on the provided income. Please try a different income value.”.
4. Code the logic for an action button to delete *Advisors* from their detail view.
5. The URL for the *Advisors* list must be shareable, meaning if you share the URL to someone else they should be able to access the app with all the current states you have (income, sort, page, etc) and see the exact same result.

Responsive Design

Make the modal component to create/edit *Advisors* responsive for mobile and desktop devices.

Test coverage

Write a short test that covers the flow to edit an *Advisor*.

Your final touch

Based on the information you already know about the experience and the pieces previously required, what do you think is missing in the experience? How can the interaction or the app be improved? Add a feature, UI element, or interaction that fixes or improves how the app is structured or how the *Admin* interacts with it.