React Test

<u>Time Duration: 3 Hours</u>

Task:

The task is to build a simple React application that interacts with an API to display and manage data. Follow the instructions below to complete the task:

1. <u>Design a mockup:</u>
    a. Create a responsive design for a web page that displays a list of items.
    b. Each item should include a id, name, email, image.
    c. The design should adapt to different screen sizes, providing an optimal user experience on both desktop and mobile devices.
    d. The link provided contains only the desktop mockup. You have the flexibility to optimize and customize the mobile design according to your own preferences.
    e. Adobe XD Mockup Link: https://xd.adobe.com/view/7a3edbc8-9088-4bc9-9bb2-7bae9ca53db3-2f6d/

2. <u>API Integration:</u>
    a. Make an API call to "https://reqres.in/api/users?page=1" to retrieve a list of items.
    b. Display the fetched data in the designed mockup.
    c. Handle any potential errors during the API request.

3. <u>CRUD Operations Using Redux:</u>
    a. **Implement functionality to create a new item and add it to the list:**
        i. Dispatch an action to update the Redux store with the new item data.
        ii. In the Redux reducer, update the state by adding the new item to the list.
        iii. After updating the Redux store, save the updated list to the local storage using localStorage.setItem().

    b. **Implement functionality to update an existing item's details and reflect the changes in the UI:**
        i. Dispatch an action to update the Redux store with the modified item data.
        ii. In the Redux reducer, find the item in the list based on its unique identifier and update its details.
        iii. After updating the Redux store, save the updated list to the local storage using localStorage.setItem().

    c. **Implement functionality to delete an item from the list and update the UI accordingly:**
        i. Dispatch an action to update the Redux store by removing the item from the list.
        ii. In the Redux reducer, filter out the item from the list based on its unique identifier.
        iii. After updating the Redux store, save the updated list to the local storage using localStorage.setItem().

By following this approach, you can ensure that the state is managed through Redux for global storage and persisted in local storage for data persistence across page reloads.

4. <u>Additional Functionality - Sorting:</u>
    a. Implement a sorting feature that allows users to sort the list of items based on a specific field (*id, name and email*).
    b. Update the list dynamically based on the selected sorting options.

Evaluation Criteria:

You will be evaluated based on the following criteria:

- Correctness of the implemented functionalities (mockup design, API integration, CRUD operations, Redux implementation, sorting feature).

- Proper organization and structure of the React components and Redux setup.
- Effective use of React best practices and patterns.
- Code readability, maintainability, and modularity.
- Handling of edge cases and error scenarios.
- Responsiveness of the design across different devices.
- Overall user experience and attention to detail.

Note: You may use any additional libraries or packages as needed, but clearly specify them in your code. Good luck!