Assignment on react js axios

Objective: Develop a React application that fetches and displays user data from the JSONPlaceholder API using Axios.

Assignment Tasks:

1. **Setting up the project:**
   - Create a new React application using Create React App.

npx create-react-app user-fetcher cd user-fetcher

   - Install Axios.

npm install axios

2. **Developing the Components:**
   - Create a **UserList** component:
     - This component should fetch a list of users from **https://jsonplaceholder.typicode.com/users** when mounted.
     - Display each user's name in a list.
   - Create a **UserDetails** component:
     - This component should accept a user ID as a prop.
     - When a user ID is passed, it should fetch that user's details from **https://jsonplaceholder.typicode.com/users/{userId}** and display their details (name, username, email, etc.).
3. **User Interaction:**
   - Modify the **UserList** component:
     - Add a button or clickable element next to each user's name.
     - When clicked, the user's details should be displayed below the list using the **UserDetails** component.
4. **Styling:**
   - Add some basic styling to your components using CSS or any CSS-in-JS library of your choice.
5. **Error Handling:**
   - Implement error handling for the Axios requests. If there's an error in fetching data, display a user-friendly error message.
6. **Extra Credit (Optional):**
   - Implement a loading spinner or message that displays while fetching data.
   - Implement pagination for the user list, given that the API might contain many users.

Submission:

Submit the following:

1. The source code of your React application.
2. Upload code into git and share url
3. A **README.md** file with instructions on how to set up and run your application.
4. Any additional notes or documentation about your application.

Evaluation Criteria:

1. Code clarity and organization.
2. Proper usage of React principles (state, props, lifecycle methods, etc.).
3. Proper handling of asynchronous operations with Axios.
4. User interface and error handling.
5. Bonus: implementation of the extra credit tasks.

Tips:

- Keep your components small and focused on a single responsibility.
- Use modern React features like hooks (**useState**, **useEffect**, etc.) for better code organization.
- Remember to handle edge cases (e.g., what if the API returns no users?).