Technical Assignment: React Web Application

Objective: Create a Single Page Application (SPA) that showcases your React skills, particularly in hooks, lifecycles, state management, Webpack optimization, and web performance.

Task Description:

SPA Features:

- The application should have at least 3 routes/pages: Home, User List Page and User Details Page.
- User List Page:
    - Display a list of users with the following columns: Name, Email, Age, and Actions.
    - Pagination: Limit to 10 users per page.
    - Sort: Provide sorting options for Name and Age.
    - Search: Include a search bar to search users by name.
- User Details Page:
    - Display detailed information of a selected user: Full Name, Email, Age, Address, and Profile Picture.
    - Provide a back button to return to the user list page.
- The Home page should showcase some dynamic content fetched from an API like JSONPlaceholder, where users should be able to add, delete, and list items (e.g., tasks, notes, etc.).

TypeScript:

- Ensure the entire application is written using TypeScript.
- Utilize strong typing, including interfaces and types wherever necessary.
- Implement utility and generic types where appropriate.

React Features:

- Use React hooks effectively (useState, useEffect, useMemo, etc.).
- Demonstrate side effects handling with useEffect.
- Use React Router for navigation.
- Add animations/transitions when navigating between pages.

State Management:

- Choose a state management strategy or tool (e.g., Context API, Redux, Recoil).
- Implement a theme switcher (light/dark mode) using the chosen state management tool.

Performance:

- Implement code-splitting using React.lazy() and Suspense.

- Use useMemo or useCallback where necessary to showcase your ability to prevent unnecessary renders or recalculations.
- Ensure no render-blocking resources are present.
- Optimize images or any assets you use.

Webpack:

- Customize the Webpack config (You might start with Create React App but eject it for customization).
- Ensure the final bundle is optimized in size.
- Implement tree-shaking if necessary.
- Set up Webpack to show a bundle size report.

Server-Side Rendering (Optional but a plus):

- Implement server-side rendering for the application to showcase initial load performance enhancement.

Extras:

- Use styled-components or any CSS-in-JS solution of your choice.
- Ensure the application is responsive.
- Implement error boundaries in the application for better error handling.

Documentation:

- Include a README.md detailing how to run and build the project, and any design decisions you made.
- Discuss how you ensured the application's performance and any tools or techniques you used.

Evaluation Criteria:

1. Code Quality: Consistent code style, clear namings, modularity, and usage of best practices.
2. Functionality: All features should work without errors.
3. Performance: No unnecessary re-renders, efficient data fetching, and optimized asset loading.
4. Responsiveness: The application should be fully responsive across devices.

Bonus

1. Tests: Quality and coverage of unit tests

Submission:
Submit your application as a Git repository (e.g., GitHub, GitLab, Bitbucket). Ensure to include all source files and documentation. Make sure to commit regularly so we can track your development process.