CMPS 356 – Fall 2022
Web Applications Design and Development

Lab 05
React Hooks and Routing

Objective

1. Using outlet contexts (`useOutletContext`)
2. Navigating programmatically (`useNavigate`)
3. Setting protected routes
4. Using multiple layouts
5. Memoizing values using callbacks (`useMemo` and `useCallback`)
6. Prioritizing state updates using transitions and deferred values (`useTransition` and `useDeferredValue`)

Prerequisites

1. React Router tutorial: https://reactrouter.com/en/main/getting-started/tutorial
2. React hooks API reference: https://reactjs.org/docs/hooks-reference.html
3. Debouncing and throttling: https://css-tricks.com/debouncing-throttling-explained-examples

1. Outlet Contexts and Programmatic Navigation

1. Create a new directory `01-routing-navigation` and reuse the code of the last exercise from the previous lab under it.
2. Fetch the list of country facts and store it in the `App` component.
3. Rename your `Gallery` page to `Facts` and display the list of countries in a dropdown list instead of using multiple links.
4. Navigate to the corresponding country and display its facts when the selection changes. Add an extra default option for the index page.
5. Navigate to the `Facts` page as fallback when the country provided (in the path) is not part of the list.
6. Use an outlet context to pass the list of country facts to your outlets.
7. How can the country selection be made more efficient to the end-user?

2. Protected Routes and Multiple Layouts

1. Create a new directory `02-protected-routes` and reuse the code of the previous exercise.
2. Add a `Login` page that allows the end-user to authenticate by providing their username and email address. Store the user state and move your routes to the `App` component.
3. Use a different layout for the login page that does not include a header nor a footer. This requires modifying your routes and creating a `Basic` component for layout.
4. Display the username in the header along with a button to unauthorize (logout). The end-user should be automatically redirected the landing page after logging out.

5. Create a new `Photos` page that is only accessible after the user has authenticated. This requires a new route entry and component, `ProtectedRoute`, to handle such protected pages (also called private pages). Update your navigation bar to link to this new page.
6. Create a protected `Profile` page that allows the end-user to update their username or email address. Add the corresponding route entry and navigation bar link.
7. Update your login/logout button such that the end-user can manually authorize by clicking it.
8. Hide the protected links from the navigation bar when the end-user is not authorized.

3. Value Memoization and Update Deferral

1. Create a new directory `03-memoization-deferral` and an application under it.
2. Create an `Input` component that holds a text value linked to a state that can be set by the end-user through a textbox.
3. Update your component to compute and display the n^{th} prime number based on a property passed to the component.
4. Try changing the text input quickly and check the responsiveness of the overall user interface when dealing with large prime numbers. Use a hook to memoize the n^{th} prime number so that it is reused whenever the component is rendered.
5. Add a textbox to test whether an input number is prime or not. Try changing the text input quickly using large numbers. How responsive is the experience?
6. Use a hook to defer the state associated with the value of the input. Is that sufficient to enhance the responsiveness?
7. Memoize the call using the deferred value of the state instead of the state itself.
8. How can the end-user be kept updated about of the status of the computation? Use a transition.