CS 295R
Lab 6 – Blog Application
Routing and Post Page

The objective of this lab is to introduce you to navigation or routing in a React application. You will be introduced to React Router, one of the most popular open-source routing libraries for React. After watching Section 13 of Modern React and experimenting with 2 React Router tutorials, you'll add routing to the Blog application from the previous lab. You will also complete the Post page for the application.

Routes the App Must Support

Link	Goes To	URL	Notes
Brand Icon on NavBar	Home page	http://localhost:3000/ or http://localhost:3000/index	
Edit User Profile on NavBar	EditUser page	http://localhost:3000/user	User information from UserContext
Add Post on NavBar	EditPost page	http://localhost:3000/posts/new	
More … link on PostCard	Post page	http://localhost:3000/posts/# where # is the id of the post	Post could be added to state or you could get the post from PostsContext based on the id
Edit Post on PostCard	EditPost page	http://localhost:3000/posts/edit/# where # is the id of the post	Post could be added to state or you could get the post from PostsContext based on the id
Edit Post on NavBar on Post page	EditPost page	http://localhost:3000/posts/edit/# where # is the id of the post	Post could be added to state or you could get the post from PostsContext based on the id
	Error page or Home page	Anything else	

Setting Up

1. Add react-router-dom to the project.
2. You may encounter an error when you install react-router-dom. You can eliminate the error by npm install --D @babel/plugin-proposal-private-property-in-object

Adding Routing

3. Add the BrowserRouter in index.js. Render the BrowserRouter inside both contexts but outside the App component.

```
root.render(
  <UserProvider>
    <PostsProvider>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </PostsProvider>
  </UserProvider>
);
```

4. Create the Layout component. It should display the NavBar component followed by the Outlet component. The Outlet component will contain the page determined by the route.
5. Add Routes to App.js.
    a. The / Route should display the Layout element. Nested inside that
    b. The index Route should display the Home page.
    c. The user Route should display the EditUserProfile element when a user is defined. Otherwise, the app should navigate to the / route.
    d. The posts/:id Route should display the Post page.
    e. The posts/new Route should display the EditPost page when a user is defined. Otherwise, the app should navigate to the / route.
    f. The posts/edit/:id Route should display the EditPost page when a user is defined and the user is the user who created the post. Otherwise, the app should navigate to the / route.
    g. Anything else should display an error page. Or the home page.
6. Create a first version of the Post page in the pages folder. Display a heading on the page as well as the id from the route. You'll finish this page later in the lab.
7. Create a first version of the EditPost page in the pages folder. Display a heading on the page. Because this page will be used for both editing a post and creating a new post, display a different heading depending on whether the page is being used for editing or creating. This information is available from the route. You'll create this page in the next lab.
8. Create a first version of the EditUserProfile page in the pages. Display a heading on the page as well as some identifying information about the user who is logged in.

9. Replace anchor elements with Link or NavLink elements in the NavBar component.
    a. Replace the brand anchor element. It should link to /.
    b. Replace the edit post anchor element that appears only when you're on the EditPost page. It should link to /posts/edit/:id. It should be displayed only when the logged in user is the user who created the post. The post might be part of the state that is available in the route because you're on the EditPost page. Otherwise, it can be retrieved from the PostsContext based on the id of the post which is available in the route.
    c. Replace the new post anchor element. It should link to /posts/new.
10. Replace anchor elements with Link in the PostCard component.
    a. Replace the edit post anchor element. It should link to /posts/edit/:id. It should be displayed only when the logged in user is the user who created the post. The post is available as a prop because you're in the PostCard component.
    b. Replace the more … anchor element. It should like to /posts/edit/:id. The post is available as a prop because you're in the PostCard component.
11. Test. All links in the application as well as the back and forward buttons in the browser. Make sure you test when a user is not logged in, with a logged in user and after a user has logged out. Make sure you test manually editing the url in the address bar too.

Creating the Post Page

12. Create the PostHeader component.
    a. It will receive post as a prop.
    b. It should display some identifying information about the post.
13. Finish the Post page component.
    a. Import either useLocation or useParams from react-router-dom.
        i. If your post page uses the id from the route to get the post from the PostsContext, you'll use useParams.
        ii. If you added post as the state prop in the Link component in the PostCard component, you'll use useLocation. (specifically the state property of the location object returned from calling useLocation)
    b. It should render the PostHeader component passing the post as the post prop.
    c. It should render the properties of the post that were not rendered in the header.
        i. If you want your users to be able to format the content of their posts using html, you'll need to
            1. install html-react-parser in the project.

2. import parse from html-react-parser at the top of the file.
3. use the parse function in your JSX to render properties of the post that might contain html.

```
<div className="m-2">{parse(post.content)}
```

14. Test.
15. Create a production version of the app. Deploy the production version to citstudent.
    a. You already have .env file for the project. You already set up an instance of json-server for this application on citweb.
    b. Change the value of the environment variable REACT_APP_SERVER_URL to use the port on citweb.lanecc.net.
    c. Create a production version of the app in the usual way.
    d. Test the application from the build folder on your machine. This version will be using the instance of json-server running on citweb.
    e. Deploy the contents of the build folder to citstudent and test the application again.