Basic Routes Lab (CodeGrade)

- Due Mar 10 by 11:59pm
- Points 1
- Submitting an external tool
- Available until Mar 11 at 11:59pm

This assignment was locked Mar 11 at 11:59pm.

(https://github.com/learn-co-curriculum/react-hooks-react-router-routes-lab-v6) (https://github.com/learn-co-curriculum/react-hooks-react-router-routes-lab-v6/issues/new)

Learning Goals

- Use `createBrowserRouter` to create a client-side router.
- Use `RouterProvider` to include the router in your app.
- Use the `<NavBar>` component to allow client-side navigation.
- Use `errorElement` to set up router error handling.

Introduction

In this lab we are going to build out a Movie application that has routes for a Home Page, Actors Page, Movie Page, and Directors Page. Our goal is to provide routes and links for these 4 pages.

Let's work through this one component at a time.

Setup

Our `src` folder contains the following JavaScript files:

```
src/
├── components/
    ├── MovieCard.js
    ├── NavBar.js
    ├── NavBar.css
└── pages/
    ├── Actors.js
    ├── Directors.js
    ├── Home.js
    ├── Movie.js
├── index.css
├── index.js
├── routes.js
```

You'll need to fill out these various files to get your app up and running. You're also free to make your own new components when you feel its warranted! (Look out for repetitive code, or code that seems like it deserves its own new component.)

To start up the lab, first run `npm install` , as per usual. Then run `npm run server` to start your `json-server` and `npm start` to open the application in the browser.

routes.js

You'll be adding the routes you create to this file and saving them within the `routes` variable. You'll need to provide routes for `/` , `/directors` , `/actors` , and `/movie` . The `/movie` route should also include a URL parameter called `id` . Don't forget that you'll need to import components into this file!

index.js

Our `index.js` file is currently broken. (It's not rendering anything!) You'll need to update it to provide routing to our application using `createBrowserRouter` and `RouterProvider` .

Components

NavBar

This component needs to render three `NavLink` components. They will be for `/` , `/directors` , and `/actors` , in this order (test checks for this). The `NavLink` for `/` should render `Home` , `directors` should render `Directors` , and `actors` should render `Actors` . Each page should render the NavBar.

MovieCard

This component is already set up to render the title of one movie. You'll need to pass it the appropriate props to render a movie's title. You'll also need to use a `Link` component from `react-router-dom` that uses dynamic routing to link a user to the `Movie` page, using the movie id as a parameter.

Pages

Home

This component should render on the `/` route. It should display the text `Home Page` in an `<h1>` . It should also render a list of movies using `MovieCard` components.

Movie

This component should render on the `/movie` route. You will need to include a URL parameter of `id` on that route.

The component will display information about one specific movie. It should display the movie's title in an `<h1>` tag, the movie's time in a `<p>` tag, and each of the movie's genres within its own `<span>` tag.

You'll need to use the `useParams` hook to get URL parameter data about which movie you want to render, then use that data to fetch and render the appropriate movie.

Directors

This component should render on the `/directors` route. It should display the text `Directors Page` in an `<h1>` , and render a new `<article>` element for each director in our array of directors. The `<article>` should contain the director's name in an `<h2>` and a `<ul>` with a list of their movies.

Actors

This component should render the text `Actors Page` in an `<h1>` , and render a new `<article>` element for each actor in our array of actors. The `<article>` should contain the actor's name in an `<h2>` and a `<ul>` with a list of their movies.

> Note: The tests will count how many `<article>` s are nested inside your `Directors` and `Actors` components. So to get tests to pass, you must create *exactly one* `<article>` for each director or actor, and no additional nested `<article>` s in those components.

ErrorPage

You'll need to create a new component within the `pages` folder for our `ErrorPage` . This page should display our `NavBar` component, along with the text "Oops! Looks like something went wrong." in an `<h1>` .

> Note: Even when all of your tests are passing, you will see a `console.warn` message indicating that the route the test file is using — `bad-route` — doesn't match any routes.

Resources

- **[React Router](https://reactrouter.com/en/main) (https://reactrouter.com/en/main)**