Lab 7

Due Dec 11, 2023 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

CS-554 Lab 7

React/NextJS GraphQL Client for Lab 3

For this lab, you will create either a React or NextJS (You can choose which you want to use) front-end for your lab 3's GraphQL Server.

Objective

You must create a complete front-end application that uses **Apollo Client (https://www.apollographql.com/docs/react/)** to connect to your GraphQL server. You must utilize **ALL** of your queries and mutations from lab 3! You can use the GraphQL Client **lecture code (https://github.com/stevens-cs546-cs554/CS-554/tree/master/graphql/react_client)** as reference

Routes You Will Need for Your Frontend.

/

This route will explain the purpose of the site.

/authors

This route will display a list of results from the `authors` query. The list of authors will have some information on each author (make sure you have the first name, last name, and number of books of each author). Each author will also have an edit and delete button, to run the editAuthor and deleteAuthor mutations. The route should also have a button called "Add Author" that handles the addAuthor mutation. The add author and edit buttons should show a modal where the user can fill in the relevant information for the mutations. You should also be able to click the author's name to be sent to the /authors/:id route to display that author's details. For Add/Edit all of the same constraints for the fields apply that applied to your GraphQL server.

/authors/:id

This route will display information on an individual user through calling the `getAuthorById` query. The individual author page will have all information on the author from the query, including the number of books and a list of their books with a limit of 3 (meaning, if they wrote four books, their first three are shown). You should also be able to edit or delete the author from this page so it will also have an edit and delete button, to run the editAuthor and deleteAuthor mutations.

/books

This route will display a list of results from the `books` query. The list of books will have some information on each book (make sure you have the title, genre, and price of each book). Each book will also have an edit and delete button, to run the editBook and deleteBook mutations. The route should also have a button called "Add Book" that handles the addBook mutation. The add book and edit buttons should show a modal where the user can fill in the relevant information for the mutations. You should also be able to click the book's title to be sent to the /books/:id route to display that book's details. For Add/Edit all of the same constraints for the fields apply that applied to your GraphQL server.

/books/:id

This route will display information on an individual user through calling the `getBookById` query. The individual book page will have all information on the book from the query, including the author's first and last name. You should also be able to edit or delete the book from this page so it will also have an edit and delete button, to run the editBook and deleteBook mutations.

/search

The search route should allow for all three search queries (booksByGenre, booksByPriceRange, and searchAuthorsByName) to be executed and the relevant search results displayed. A user should be able to select a specific search query, type their query into the search bar, and get only the results from their selected search query. For example, if a user wants to search for horror books, they should be able to select the "booksByGenre" query, type in "horror", and see all books with the genre of "Horror". (You can either display this search form on every page like in the navigation, or have its own page). **Important Note:** Since booksByPriceRange takes in a min and max price, if the user selects this search type, the search form will change from one input (used for a single search term for booksByGenre and searchAuthorsByName) to a form that has two inputs, one for min price and one for the max price. If no input is supplied for min price, you will default the min to be 0 when you run the query.

HTML, Look, and Content

Besides the specified settings, as long as your HTML is valid, the general look and feel is up to you. Feel free to re-use the same general format as one of your previous labs, if you'd like.

I do, however, recommend using **Material UI (https://mui.com/material-ui/)** to make your life easier if you need any UI elements.

I will strongly recommend that you include ways to easily navigate your website. Do not require the user to press the browser's back button to go back to the previous page, for example, and make sure that you have a navbar to navigate the website easily.

General Requirements

1. Remember to submit your `package.json` file but **not** your `node_modules` folder.
2. Remember to run HTML Validator!
3. Remember to have fun with the content.
4. Remember to practice usage of async / await!