Programming Assignment 3

The goal of this assignment is to take the auth website you previously built using HTML, CSS, and Bootstrap and turn it into a React app. By the end of this assignment your auth website will be fully functional in the browser.

Your group is tasked with turning your three basic pages into bootstrap components. You are not allowed to turn the website into three large components for each page and call it there. You earn points for this assignment by implementing specific parts of the page functionality. For each of the pages, I will give you a list of requirements that the page must fulfill. These requirements will largely be form submit behaviors.

A few reminders about how groups work in CS316

- When you decide on a group you must inform me of your decision
- You may not switch groups throughout the semester

Notes on Persisting Data

This assignment involves a lot of managing client side state in the browser. While there are ways to store all of this information in a way that persists, you are not required to get this working. This means that when you manually refresh your website you will lose all state information.

If your app is losing state data from just clicking the nav bar you are probably doing something wrong. React Router should not refresh when routing to a new page. But it can if the feature is turned on. So just make sure you have that turned off.

I also have a design recommendation when implementing your state logic: try to centralize all of your state management into a couple functions. This will make it a lot easier to ensure that every React component is interacting with the state variables correctly. It will also make it easier to add in the extra credit at the end if you want.

Page Descriptions

As an extension of the last programming assignment you will be adding functionality to each of the pages you previously designed. I'll have the required features itemized by page like in PA2.

Navbar

Currently your navbar should contain three links to the account, login, and sign up pages.

Navbar - User Login Status

Add a spot where the current logged in user's username/email is displayed. You can either add it inline with the rest of the header or put it on it's own line above or below the existing navbar. It should not be displayed if the user isn't logged in.

Navbar - Routing Works

Use the react-routing-dom npm library to implement a multi page react app. It should have the following routes

- /login -> Login Page
- /signup -> Sign Up Page
- /user -> Account Page

Login Page

Currently your login page should just be a page with a form on it.

Login Page - Translated to React

Convert this page into React components. Be sure to try and break up the components into small chunks that make sense given the application.

Login Page - Works

You should be able to log in to your website. This means that when you type a email/password combo in your account list you should be able to log in.

On successful login redirect to the account page.

The login status doesn't need to persist across page loads.

Sign Up Page

Currently your login page should be a page with a form on it.

Sign Up Page - Translated to React

Convert this page into React components. Be sure to try and break up the components into small chunks that make sense given the application.

Sign Up Page - Works

Assuming that the form validation passes, when submitted this form should add an account to your list of hard coded accounts. This means that after signing up using this form you should be able to log in using those credentials.

Do not worry about any edge cases like duplicate users/etc. That will be covered in the next assignment.

On successful sign up redirect to the account page.

The updated account list does not need to persist across page loads.

Account Page

The account page currently should include a form with a fake user's information.

Sign Up Page - Translated to React

Convert this page into React components. Be sure to try and break up the components into small chunks that make sense given the application.

Account Page - Works

Assuming that the user is logged in, the account page should fill in the current user's information on the form. I do not care what happens if a user tries to go to this page while not being logged in. In addition to displaying the current user's information, this page should allow you to update the user's information when the form is submitted. The following fields should have the ability to be updated.

- username
- email
- phone

The updated account info does not need to persist across page loads. It can revert back to the original value.

Rubric

- total 25 pts: login page
    - 10 pts: translated to React
    - 15 pts: works
- total 25 pts: sign up page
    - 10 pts: translated to React
    - 15 pts: works
- total 25 pts: account page
    - 10 pts: translated to React
    - 15 pts: works
- total 20 pts: navbar
    - 10 pts: login status
    - 10 pts: routing works (must not reload)
- total 5 pts: footer
    - 5 pts: translated to React

to earn full credit for translated to React your React app should use sufficiently small components

Extra Credit Opportunity

You can earn 5 points extra credit on this assignment by making your client side data persist in the browser across refreshes. I recommend finding a library that allows you to read/write data to one of the following:

- localStorage
- IndexDB

There are a ton of libraries out there that will make this extra credit much easier than it might seem at first glance.