Programming Assignment 5

This assignment is not required and will be extra credit for those who want it. However, implementing authentication and authorization system can be a good thing to put on a professional resume or portfolio as a CS major.

The goal of this assignment is for you to gain experience implementing a basic authentication and authorization system. During your last programming assignment you implemented the basic form handling, but we left out the access control and session storage piece of the system. This access control piece will be the primary focus of this assignment. In addition, there will be one minor front end change.

As always, you and potentially one group member will be responsible for implementing the described changes. You will earn points for each implemented access rule as well as for the front end change.

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
- Please be sure to include both group members names on your submission

Back End Endpoint

GET - /user/logout

This endpoint should delete the user's session and redirect them to `/user/login`. There is no front end page for this endpoint.

Front End Changes

Currently your front end should be fairly robust with an alert and error system. The only update will be to the navbar.

Now that the system knows who the current user is, we are able to display the correct information in the navbar. Wether or not the user is logged in will change the layout of the navbar.

When the user is logged in

The navbar should have the following links

- My Account - /user/:user_id (this link should actually work and take them to their account page)
- New User - /user/new
- Logout - /user/logout

When the user is not logged in

The navbar should have the following links

- Login - /user/login

Access Control Changes

As it stands our server is comprised of four endpoints. Most with two implemented requests GET and POST.

- GET, POST: /user/login
- GET: /user/logout
- GET, POST: /user/new
- GET, POST: /user/:user_id

Due to the small number of possible requests, I will enumerate the behavior of each. In practice, enumerating access control patterns across all possible endpoint and method combinations may not be possible.

GET, POST - /user/login

The user should only be able to navigate to this page if they are not logged in.

If a user is logged in they should be redirected to their account page `/user/:user_id` (for both GET and POST requests)

GET - /user/logout

The user should only be able to logout if they are logged in.

If a user is not logged in they should be redirected to the login page.

GET, POST - /user/new

The user should only be able to create a new user if they are logged in.

If a user is not logged in they should be redirected to the login page.

GET - /user/:user_id

The user should only be able to view a user's account if they are logged in.

If a user is not logged in they should be redirected to the login page.

POST - /user/:user_id

The user should only be able to update their own account page.

If a user is not logged in they should be redirected to the login page.

If the user is trying to update an account other than their own they should get an access error.

Rubric

total 50 pts

- total 30 pts access management
    - 6 pts: GET, POST - /user/login
    - 6 pts: GET - /user/logout
    - 6 pts: GET, POST - /user/new

    - 6 pts: GET - /user/:user_id
    - 6 pts: POST - /user/:user_id
- total 15 pts session management
    - 5 pts: login page creates session
    - 5 pts: logout page deleted session
    - 5 pts: correct arguments set on session cookie
- total 5 pts front end
    - 5 pts: navbar changes and link redirects to current user's account