SE 3200: Web Application Development I

Assignment: Authentication

Requirements

Registration

- Define a resource to represent a user. Its attributes should include, at a minimum, first name, last name, email, and encrypted password.
- Using SQLite, create a valid database schema to store records for the user resource and its representative data model.
- As part of your server application, create a RESTful API endpoint that will create a new user record, using the provided data attributes. Be sure to encrypt the user's password using a secure password hashing method, and discard the plain-text password.
- As part of your client application, provide a user interface that allows the user to register by providing the required information, and informs the user upon successful registration.
- At a minimum, validate that the email entered by the user is unique prior to creating a new user record, and provide feedback to the user if it is not. This should be coordinated between the server and client applications, via the same API endpoint described above. The logic to validate uniqueness should be performed by the server API.

Authentication

- As part of your server application, create a RESTful API endpoint that will authenticate a user's provided credentials. If the user cannot be identified, or if the password does not match the previously-encrypted password, then an appropriate error response should be returned. Otherwise, an appropriate success response should be returned that includes basic data attributes for the authenticated user that may be used by the client application.
- As part of your client application, provide a user interface that allows the user to enter their credentials to be authenticated, and informs the user upon either successful or failed authentication.
- Implement a basic in-memory session store to save temporary data related to the user's session, using cookies appropriately to associate the session ID with the user agent. Upon successful authentication, use the session store to persist the user's authenticated state and identity. Do not add superfluous data to the session store.

Basic Authorization

- Restrict at least one other resource represented by your application (different than the user resource) such that it may only be read and managed by an authenticated user. This should include all read and write operations on the resource supported by your server and client.
- The authorization logic should be implemented on both your client and server applications.
    - Your client application should not reveal user interface elements related to a restricted resource until a user has successfully authenticated.
    - Likewise, your server application should not allow unauthenticated access to any API endpoint related to a restricted resource. If unauthenticated access is attempted, an appropriate error response should be returned.
- It is not necessary to discriminate between different users; all authenticated users may be given equal permissions.

General

- Your client and server application should conform to the same standards and conventions as required by previous asignments, including: encapsulation of database logic; proper implementation of REST

standards; appropriate use of HTTP methods, status codes, and headers; correct implementation of common error responses (404, etc.); CORS.

- All data communicated between the client application and the server API should be implemented using Ajax requests.
- You may take liberties to modify your application's features and purpose from that described above, but the overall specifications and structure listed above should still be met.
- Make your application look professional and presentable. Use valid HTML and CSS to structure and style your application.
- No third-party JavaScript or CSS libraries or frameworks may be used without prior instructor permission.

Documentation

- The following items should be clearly detailed and documented in the `README.md` within your Git repository:
  - The names and attributes of each of your resources.
  - The database schema which represents your resources, documented as valid SQLite `CREATE TABLE` queries.
  - All REST endpoints implemented by your API server. Include the name, path, and HTTP method for each.
  - The password hashing method and any relevant parameters used by your application.
- Use Markdown to structure and style the content within your `README.md`.

Submission

1. Submit your project using Git and GitHub. Start by creating a repo for this assignment here.
2. Show your completed assignment to the instructor during class or office hours to receive credit.