1 Project 4

Due Date: 11/29 by 11:59p

Important Reminder: As per the course *Academic Honesty Statement*, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

1.1 Aims of This Project

The aims of this project are as follows:

- To build a authentication web application which uses a authentication web service.
- To generate HTML on the server.

1.2 Specifications

Set up your gitlab project so that it contains a `web-auth` subdirectory within the top level `submit` directory. Cloning your project, setting the current directory to your `web-auth` subdirectory and then running `npm install` followed by

```
./index.js [ -d|--ssl-dir SSL_DIR ] PORT WS_URL
```

should result in a web server listening for HTTPS requests on local port `PORT`.

Your web server must act as a front-end for a web-service running on the URL specified by `WS_URL` which meets the specifications for *Project 3*.

The program takes the following option:

`-d | --ssl-dir` The path to the directory containing SSL credential files `key.pem` and `cert.pem`. If not specified, the value should default to the directory from which the server was started.

The program should provide an authentication web application with the following web pages:

Login The login page must provide the following suitably labelled input controls:

- An input field for an email address.
- An input field for a password.
- A *Login* submit button.
- A link to the *Registration* page.

When the page is submitted using the *Login* button, the program should validate the input fields:

- None of the input fields should be empty.
- The email address should look like a reasonable email address.

If the validation succeeds, the program should use the login web service at `WS_URL` to set up a login session for the browser which submitted the request and display the *Account* page in the browser.

If either the login web service or the validation fails, the program should redisplay the login page with suitable error messages. All user input except for the password should be retained; the password field should be cleared out.

Registration The registration page must provide the following suitably labelled input controls:

- An input field for a first name.
- An input field for a last name.
- An input field for an email address.
- An input field for a password.
- An input field for password confirmation.
- A *Register* submit button.
- A link to the *Login* page.

When the page is submitted using the *Login* button, the program should validate the input fields:

- None of the input fields should be empty.
- The email address should look like a reasonable email address.
- The password should consist of at least 8 characters none of which is a whitespace character and at least one of which is a digit.
- The value for the password confirmation field must match the password field.

If the validation succeeds, the program should use the registration web service at `WS_URL` to create a registration corresponding to the submitted information and set up a login session for the browser which submitted the request and display the *Account* page in the browser.

If either the registration web service or the validation fails, the program should redisplay the registration page with suitable error messages. All user input except for the password fields should be retained; the password fields should be cleared out.

Account The account page should display suitably labelled first name and last name from the registration corresponding to the current login session.

It should also contain a `Logout` button such that clicking that button terminates the current login session and displays the *Login* page.

You may choose any URLs for the above pages. You are **required** to support a `/` URL such that:

- Any attempt to access `/` using a browser which does not have a valid login session will result in display of the *Login* page.
- Any attempt to access `/` using a browser which has a valid login session will result in display of the *Account* page.
- Any attempt to access the URL chosen for the *Account* page should result in a display of the *Login* page if there is no valid login session.

Your web application must meet the following additional requirements:

- All requests should use https.
- Leading and trailing whitespace should be ignored for all input fields.
- A reasonable email address must look like *user*`@`*domain* where *user* and *domain* can be arbitrary non-empty strings.

1.3 Hints

You may do the project on any machine which has compatible nodejs and mongodb installations. However, it is entirely your responsibility to make sure that the code works on your gcloud vm before submission.

[Note that if you set up your gcloud vm as instructed, the only ports which are open are the HTTP (80) and HTTPS (443) ports; to run your server on the HTTPS port (443), you will need to start your web server using `sudo`.]

The following steps are merely advisory:

1. Review material you will need for this project; the material for URLs, HTTP, web-services, authentication, cookies, HTML and templates.
2. Decide how you will maintain login sessions within the browser.
3. Decide on which external modules you wish to use using the criteria given for *Project 3*.
4. Create and set up the certificate files necessary for your server.
5. Verify that any http-client library (like axios) you may be using can be set up to accept the self-signed certificate used for the web-services running at `WS_URL`.

6. Adapt the options.js module provided for *Project 3* to this project. Verify its operation by running via the command-line.
7. Choose URLs for your login, registration and account pages.
8. Create templates for your pages. Verify that `GET` requests to the corresponding URLs display the pages.
9. Implement the registration page.
   (a) Add the necessary validations for your registration page. When validations fail, set up the page to be redisplayed with user input retained and suitable error message added.
   (b) Call the registration web service when all validations succeed. Set up your code to redisplay the registration page with a suitable error message if the registration web service fails. If it succeeds ensure that the *Account* page is displayed.
10. Implement the login page in a similar manner.
11. Implement the *Accounts* page to display the first and last name of the registration corresponding to the current login session.
12. Implement the required behavior for the `/` URL.
13. Implement the functionality of the *Logout* link on the *Account* page.
14. Test to ensure that you have met all the project specifications.

1.4 Project Submission

Submit your project using gitlab:

- Submit your project files in a top-level `submit/web-auth` directory of your `cs480w` or `cs580w` project in gitlab.
- You should **not** submit executables or object files.
- Make sure that you provide a `README` file containing minimally your Name and B-number. It may contain any other comments which you would like read by the grader.

If your project is incomplete on the due date, please add a file called `.LATE` in your directory so that it will not be graded. Once the project has been completed late, please remote that file and email the grader who has been assigned to your course: *CS 480W*, *CS 580W*.