Assignment 2: Login with Encryption & Testing Marks: 10

Format: Submit zipped folder containing all files through Blackboard. BE SURE TO PUT YOUR NAME & SECTION IN THE FILENAME: e.g. *Assign2-YourName-A.zip*

BE SURE TO INCLUDE THE VALID USERNAME & PASSWORD IN YOUR SUBMISSION TEXT

Purpose: To implement JavaScript encryption with CryptoJS, and unit testing of your functions with the Jasmine testing framework.

REQUIREMENTS: Using your login code from Lab 1…

1. In addition to the onsubmit function, you will create two custom functions:
    a. **md5Encrypt** (encrypt a string with MD5 and return a string)
    b. **checkLogin** (validate user input)
2. The empty div should be used for a successful login message ("Welcome back!") as well as error messages.
3. Incorporate Jasmine testing to test your two custom functions. Your spec file should thoroughly test for the following specifications:

FUNCTIONAL SPECIFICATIONS

The **md5Encrypt** function should return a string 32 characters long.

The **checkLogin** function should return true if the username and the password match a known username and matching password.

The **checkLogin** function should return "Invalid Username or Password." if the username input does not match a known username; or the password input does not match a known password; or a valid username is input with an invalid password, or an invalid username is input with a valid password.

The **checkLogin** function should return “No username entered.” if the username is an empty string.

The **checkLogin** function should return “No password entered.” if the password is an empty string.

If you need some hints, scroll down…

To Get Started…

- In your Assign2-YourName folder copy your lab 1 login page files and unpack your Jasmine-standalone zip file. On the JavaScript page inside the onload function, create three functions.
- Your **onsubmit** function will get the values from the form, then pass them as *parameters* to your **checkLogin** function. Your

checkLogin function will call **md5Encrypt** when it compares the user input to your hard-coded encrypted password.

- Remember, if you are using <form> and not just grabbing the <input/> by id, you need to return false at the end of your onsubmit function so that you can output to the same page. Otherwise, the page will refresh on submit and your form values will be gone.

Your Functions…

- The role of a function, generally speaking, is to perform a task apart from the main flow of logic, then return to the main flow with the result. Neither of the above functions should access the DOM. That should be handled by the main flow. So, the main flow goes to the DOM and gets values; your functions process the values and return values to the main flow; then the main flow outputs to the DOM.
- Paste your previous validation for empty fields into checkLogin. You will most likely need to modify it-especially if it was accessing the DOM.
- In checkLogin, normally we would go to a database and check for a matching username/password there. For this assignment, check against a hard coded user/pass.
- Leave your output logic until after the functions have run, then output to the html page.

Testing…

- Remember that you are not putting your whole js file into the src folder, you are pasting the two functions into their own new js file for testing with Jasmine.

- Don't forget to include CryptoJS in the specRunner.html file just above the <!-- include source files here... →.
- For testing your encrypt function, use Jasmine's toMatch() matcher with a regular expression (a word character x 32).

Your final submission structure should look like this…

assign-2-MyName-1.zip

- **assign2.html**
- **assign2.js**
- **md5.js**
- **jasmine**
  - **lib**
    - **jasmine-X.x.x**
    - **md5.js**
  - **src**
    - **login.js**
  - **spec**
    - **loginSpec.js**
  - **SpecRunner.html**
  - **MIT.LICENSE**