PIC 40A: Homework 8 (due 12/6 at 10pm)

Like on previous homeworks, it is important that you meet the following requirements.

- You must upload your files to **Gradescope** before the deadline.
- You must upload your files to the **PIC server** in the appropriate directory before the deadline.
- Both submissions must be **identical** (down to the character).

  Never make changes to the PIC server submission after the deadline.

  (We can see when a file was last modified.)
- You must tell us (me and the grader) your **PIC username**.
- You must **validate** your HTML using https://validator.w3.org/.

  Ideally, with PHP, you should check that, after removing the PHP parts, the remaining HTML validates.

In this assignment you will submit nine files...

1. `README.txt`. This will contain your PIC username.
2. `login.php`, `login.js`, `index.php`, `h_password.txt`, and `username.js`.

   These are files that you made on previous homeworks. No changes are necessary.
3. `merch.php` and `merch.js`.

   This file will build upon the file of the same name that you submitted on a previous homework.
4. `money.php`. A script for writing a credit information to `credit.db`.

As mentioned above, you should submit all files to Gradescope before the deadline.
You should also submit the files to the PIC server. Save them in the directory

`/net/laguna/???...???/your_username/public_html/HW8`

(in the folder `HW8` within `public_html`). **You'll also need to create a folder called "sessions" with 755 permissions.** We should all be able to test your live webpage at

www.pic.ucla.edu/~your_username/HW8

Now, I am just left to tell you what I want these files to achieve. See the next page!

New Feature

You will create a database for storing user information. In this database you will store the user's username and credit. Writing to the database will happen via an AJAX request after every credit update. You'll be able to login with a specific username on different computers (or browsers) and see the current credit.

credit.db

The file `credit.db` will be a database for storing the user information. It will not exist in your original submission (delete it for grading). You will create a table with with user's username and current credit with the appropriate corresponding types. It will be updated so there shouldn't be multiple entries with the same username.

merch.php

The HTML of this page should look identical to `merch.html`. You will implement SQL in PHP to do the following...

- Open a database `credit.db`. This will be created when `merch.php` is visited by a user for the first time.
- Create a table consisting of users which will store their username and credit with the appropriate corresponding types.
- Get the information from the table of the username associated with the current user.
- If the user is visiting for the first time, you'll find the entry won't exist. Then insert a new entry with the username and credit of 20.
- If the user is returning to the page, then their credit can be obtained from the table.

Now that you now have the user's credit, inject this into the relavent `<span>` element. You might find `number_format` useful for formatting the number.

Tip: You will have format strings properly in PHP since SQL needs things to be specifically formatted. It can be helpful to use PHP sandbox to test your strings are formatted correctly.

merch.js

In order to make the credit update appear in `credit.db`, we need to change `merch.js` two ways...

1. Before starting this assignment, you should have a function that looks something like

   ```
   function update_credit() {
     credit_para.innerHTML = `Your credit : $${credit.toFixed(2)}`;
   }
   ```

   This is the function that is called every time the credit should be updated (either by a valid coupon code or purchase). This function should do more...

   - It should make an AJAX request.
   - The AJAX request should have its method set to POST and it should send the user's username and the user's credit to the PHP file `money.php`.
   - `money.php` should update the database `credit.db` with the information it receives.
   - **After all this has been done successfully,** the credit displayed on the merch page should be updated.

2. Another update to this code is to change how you initially assign your JavaScript variable `credit`. In the homework 4, its set as `20`, but now it should be whatever is injected into the page by `merch.php`.

money.php

This page is formatted like a plain text document and

- Takes the POST request information from the AJAX request and updates the credit database.
- Landing on the page directly won't produce a warning but should print a nice text message `Either the user or credit was not posted`.

Grading

Two points for each of the following...

- After going to `merch.php`, for the first time ever you should see the credit displaying $20.00.
- Make a valid purchase. The credit should change and remain the same even if refreshing the page.
- Numbers on the page and in the sales receipt should come out formatted correctly as the standard USD $0.00.
- Add a valid coupon code. The credit should change and remain the same even if refreshing the page.
- Go back to the login page. Login with a different username. After going to `merch.php`, you should see the credit displaying $20.00.
- Make a valid purchase. The credit should change and remain the same even if refreshing the page.
- Open Safari (or another browser or on another computer) login with one of those usernames. See the credit is as expected. Make a purchase or add a coupon code.
- `credit.db` should not have multiple entries with the same username.
- Go to `money.php` directly and see a text statement rather than a warning.
- All files are correctly submitted on time and `credit.db` is deleted.