COMP 484L Summer 2021 **Lab 2: Full Stack Web Development**

Due Tuesday, August 24th 2021 23:55 to Canvas, *or as a **live demo on Zoom** in Breakout Room during lab section or office hours before that date.*

---- No late submissions can be accepted for this assignment!!! ----

This assignment is to be worked on ***INDIVIDUALLY*** ! No team submissions allowed.

PURPOSE

You are going to learn how to create a basic full-stack website using PHP and MySQL, as well as learn to use AJAX with JSON (extra credit).

REQUIREMENTS

To complete this project you will perform a demo live during a lab section on Zoom (or in office hours), in a breakout room. If you demo a **fully working** Lab 2, you will receive immediate full credit. *(If you successfully demo on Zoom, you do not need to submit anything to Canvas, and your grade shall appear on Canvas right after the demo!)*

Otherwise you will create and submit one zip file containing all project files. The file will be called `lab2.zip`

There should be ***at least one*** PHP file which launches the main site, which will be named `lab2.php`

You will also use mysqldump to provide a dump of your entire database. This file will be `lab2.mysql`

Instructions for using mysqldump can be found here:
http://thingsilearn.wordpress.com/2008/08/04/mysqldump-to-text-file/
and here:
http://www.howtogeek.com/howto/mysql/backup-mysql-database-to-a-file/

Finally, you must write a short "readme" file named `lab2.txt`, containing…

- a description of all files you are turning in (especially if there are extra PHP files),
- a description of how much of the project is complete, including extra credit
- a description of any known bugs in the program

To recap, you will be turning in at least **three** files within `lab2.zip`:
`lab2.php`, `lab2.mysql`, and `lab2.txt`

You MAY use *more PHP files* to implement your site if you want. However, **you MUST include these three files with your submission!**

YOU MAY USE A WYSIWYG EDITOR! This editor can be used to generate the HTML markup of your document. Any CSS, PHP, or JavaScript must be subsequently added to this file by hand (i.e. manually editing the HTML file).

BASIC SITE FUNCTIONALITY

In this lab, you will implement a web page that allows a user to register or log in to a site. This web page is created using a file called `lab2.php` and others if needed.

The page must have a section where a new user can register with a username and password. There should also be another section where returning users can log in. Once a user logs in, the server will return a page with the following:

- welcome message to the user
- a button that changes the background color of the current page
    - (This functionality is largely up to you. You can toggle between multiple different colors with each button click if you like.)
- a logout button or link that takes them back to the login/registration/start page

The look and feel and navigation of the site is up to you, and can be kept very basic for the purpose of this assignment. You may use the PHP session object (session_start() and the $_SESSION superglobal, etc) if desired, but this is not required.

(In general, you may wish to brush up on the best practices of secure PHP code: https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet – This is not a requirement of Lab 2, but generally good to know about.)

The registration page may be based on the one you used in Lab 1. If so, you may wish to disable/remove the inputs that are not needed on this lab (like studentID and the textarea). In this lab, all that is needed for user registration is a username and password.

For the **basic** project Lab 2, *you may store the user's password in the database in plain text*. See **Extra Credit** below for a *best-practice* approach to authentication.

THE ADMINISTRATOR USER

Finally, there should be a special user in the database named Administrator. No user may register with this username. You can add Administrator to the database manually, with a password of your choice.

When a user logs in successfully as Administrator, the page they see contains a table listing all the users of the site by username and password, in sorted order by username (alphanumeric ascending sorted order). (Obviously this is insecure, and for the extra credit, you would display the *"hashed password"* on this page instead of a plaintext password. Also, you should be using MySQL query with ORDER BY clause to sort the usernames, rather than sorting them in your code.)

You are being evaluated on the site's functionality rather than its appearance. The site must function as specified.

EXTRA CREDIT

You will receive **up to 8%** extra credit on top of your COMP 484/L weighted average for implementing additional features of Lab 2 as follows…***and you will only receive this extra credit if the rest of the lab is also working!***

For 3% extra credit *on COMP 484/L final grade*:

Store a salted hash of the user's password (alongside the salt) on the server rather than the plaintext password. **This is important behavior for real websites…**the user's plaintext password need never be stored. Instead, a salt (random number) is pre-pended to the password and the resulting string is run through a hash function to create a hashed-password. The salt and the hashed password should be stored in the database, often in the same "password" column.

When a user wants to log in, the site fetches the salt for that username and prepends it to the user's entered password, before running this through the hash function to create the hashed-enteredpassword. Then it compares this hashed-enteredpassword to the hashed-password in the database. If these hashes match, the user submitted the correct password, and will be logged in.

A discussion of this behavior can be found here: http://crackstation.net/hashing-security.htm

For modern PHP, a best-practice is to use the newer Secure Password Hashing API, which provides the hashing algorithm, the salt, and the hashed password all in the same string. **You can write this whole string to the *password* column of the database (see DATABASE SUGGESTION below).**

The API for password_hash() can be found here, and is encouraged:
https://gist.github.com/nikic/3707231

Alternatively, you can also use the crypt() function with CRYPT_BLOWFISH. (See the PHP manual at php.net for details on crypt().)

For 5% extra credit *on COMP 484/L final grade*:

Make the website a single main page that never refreshes nor submits to any other page. This can be accomplished by using a PHP/JSON web service API to allow registration and login functionality to be executed on the server in response to user events. This is handled by JavaScript or jQuery (or other frontend library) code running in the browser. Instead of submitting normally, the registration, login, and administrator (list of users) pages will be using AJAX calls to communicate with the server.

******REMEMBER: if you perform any extra credit, you must tell me about it in `lab2.txt` or… *IT WILL NOT BE GRADED!!!***

DATABASE SUGGESTION

- Your MySQL database should consist of a table named **Users**
    - Table **Users** has three columns: *userId* (an auto-incrementing **primary** key), *username* (**must** be unique for every row – in other words different users must have different usernames), and *password*.
    - Hint: you created this database and this table in Homework 2

GRADING

Up to 100% (plus 8% extra credit on your final COMP 484/L score) will be provided for successful live demos on Zoom, in which the instructor will quickly assess your project execution and also your code. ***Be prepared to show both the code and the working website in the breakout room during the demo!*** The instructor may give you a score of 100% (with extra credit if earned) on the spot, or ask you to make fixes and re-submit or re-demo.

OR, for Canvas Submissions:

10%: Submission instructions were followed, and a minimum of 3 files are submitted, named according to spec

15%: JavaScript and PHP code has proper formatting, proper use of functions, and comments as appropriate

30%: Web page contains all required content, behaving according to spec

45%: PHP code and database abide by spec

<= 8% weighted average points in COMP 484/L: extra credit, if attempted/earned

CHEATING

This project is an individual project. You can discuss this project with other students. You can explain what needs to be done and give suggestions on how to do it. You cannot share source code. If two projects are submitted which show significant similarity in their source code then both students will receive an F on the assignment. Note a person who gives his code to another student also fails the assignment (you are facilitating the dishonest actions of another).

Source code that is copied from websites without citation will also count as cheating, and the same consequences apply.