# CS 3380 Lab Assignment 8

**Directions** : This assignment must be submitted via Blackboard by **Wednesday, April 8 at 11:59 PM**. You must submit **all** of your PHP files (bundled as a tar or zip file) through Blackboard prior to the deadline. You must also host your page at the following link:

`https://babbage.cs.missouri.edu/~pawprint/cs3380/lab8/index.php`

If a TA has to host your page on their own account, you will lose points. Host your page at the right address and **test it**!

**Goals:**

1. Develop a PHP script that allows a user to log in and view information about themselves.
2. Use the `$_SESSION` array to keep track of whether a user is logged in and what their username is.
3. Use the `$_POST` array to accept, validate, and process user input.
4. Use the provided SQL file to set up and utilize a database. You may download the file by using the command: `wget http://klaricm.babbage.cs.missouri.edu/ss15/cs3380/lab8/lab8.sql`

**Details:** A common problem with websites is how to have users log in to a webpage. Doing this requires you to use basic database connectivity–to validate whether a user exists in the database, or to check that they've entered the right password, etc.–as well as to address some basic security concerns. You will need to create a web site consisting of several pages.

1. First, read and understand the content of the lab8.sql file provided for you. Use the \i command in psql to execute the content of this file. The tables contained within will form the basis of this lab assignment.
2. The front page of the site (**index.php**) should feature a login/register prompt. If the user is already logged in, they should be redirected to the next page automatically. **This should not require any action** on the part of the user. Redirection needs to be done using PHP! You may use the `header()` function for this. The login prompt should be done using an HTML form. If the user enters a correct username/password, redirect them to **home.php**. If they do not, you should keep them on this page and display an error message.
3. Upon a user's first visit to your site, they will have to register for a user account by accessing the **registration.php** page. This page should contain a form that requests the user's desired username and password. The action of the form should be a PHP page (or a portion of the registration.php page) that adds a record for the new user account. An error message should be displayed if the user account was not able to be added (with a link that allows the user to return to the registration page). The user account creation process should create a salt value that is used to hash the user's password. This salt value should be a SHA1 hash of a random number. (The hashed password stored in the database for a user should be the SHA1 hash of the concatenation of a user's salt and the entered plaintext password.) Finally, upon a successful user account creation, the user should be logged in and redirected to the **home.php** page described below.
4. The next page (**home.php**) is only visible to users who are logged in. Again, if users are not logged in, they should be automatically redirected to index.php. If the user is logged in, this page should display their username, the date and IP of when they registered for an account, and their user description, if any. It should also include a link to your third page, **update.php**. In addition, this page should contain a table listing all of the dates and IP addresses of the user's login history. Finally, somewhere on the page should be a **logout** link.

5. The update page (**update.php**) is only visible to users who are logged in. The page should display the username and their description. In addition, it should contain an HTML form which has two things inside of it: a textarea which the user can type their description into, and a "submit" button. When the user clicks the "submit" button, the database should be updated by setting the user's description to the contents of the textarea. You can have the action of the form (i.e. when the submit button is pressed) go to a fourth page if you wish, or back to update.php. The choice is yours, but in either case, the user should end up back on **home.php** at the end of the update process. You do not need to validate their input for the new description–you may assume it's short enough and contains valid characters.

6. Finally, you will need a **logout.php** page. This should be a short script which logs the user out and sends them back to **index.php**. There should not be any actual HTML on this page–the user will never physically view it. It is a script only.

**Example:** There is an example page available to view, with working examples of each page, at

`http://klaricm.babbage.cs.missouri.edu/ss15/cs3380/lab8/index.php`

This will show you the structure of the page and how you move between pages. Since this assignment is about PHP and database interaction, you are allowed to use some or all of the HTML that you see on the example page.

### Requirements

1. Use the `lab8.log` table to record information about any activity related to a user's account: registration, login, logout, incorrect password during login, user description updated, etc.

2. If any part of your submission is vulnerable to SQL injection, you will lose **half the points** for this lab.

3. If any passwords are stored in plaintext in your PostgreSQL database, you will **automatically receive a 0** for this lab. Hash the passwords you receive!

4. If any passwords are hashed but not salted, you will lose **a quarter of the points** for this lab. You must apply both hashing and salting.

5. Use `$_POST`, not `$_GET`, for your login/update scripts. A GET request stores information in the URL of a webpage. You don't want your user's password to be visible in the address bar of the browser or in web server logs.

6. **All user login information should only be sent to the server via HTTPS**, to encrypt their personal information. This means that the index.php page, as well as any other pages you may use as part of the login process, must only be accessible via HTTPS. If a user attempts to access the page via HTTP, you should redirect them to an HTTPS version of the page using the header() function. (You can check if the connection uses HTTPS via the $_SERVER['HTTPS'] variable. Read the PHP documentation for how to use the header function. )

### Other Important Notes

1. Write modular, easy-to-read-and-understand code. Functions are suggested–the login function in particular will be reusable for future labs/homework and your final project.

2. Follow good programming style and write modular, easy-to-read-and-understand code. Your code should be well documented, i.e. you should have numerous PHP comments. There will be point deductions for unreadable, uncommented, or poorly commented code; this is at the grader's discretion.

3. Before you write a function yourself, consider that PHP may already have a built-in function. A very large number of functions are built in to PHP and available on Babbage.

**Relevant PHP Info**

1. header() – Useful for sending the user to a different webpage (redirection).
2. isset() – Tells you whether a variable has been set, such as a submit button.
3. sha1() – Gives you the hashed form of a string. This function exists natively in PHP.
4. Consider writing a PHP file that defines a function which connects to your database (and returns a handle to the connection). It will be much easier to simply include that file in your pages and call the function, than it will be to duplicate the connection code everywhere.