COMP 484L Spring 2016 **Lab 4**

Due Friday, May 20th 2016 23:55 – **Friday of *FINALS* week** via Moodle

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

----Students MAY perform this assignment in groups of <=2 people. Such groups must submit only one single lab4.zip (under either members' Moodle account) BUT their lab4.txt MUST name all students on the team (or they will not receive credit)!

Purpose: You are going to learn how to create a basic website using PHP and MySQL.

Requirements: To complete this project you will write and submit one zip file containing all project files. The file will be called `lab4.zip`

There should be **at least one** PHP file which integrates with the web-server, which will be named `lab4.php`

You will also write and submit one PHP file which is invoked every half hour by a CRON job named `mailer.php`

You will also use mysqldump to provide a dump of your entire database. This file will be `lab4.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 `lab4.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
- a description of **EXACTLY** which extra credit parts were completed (if any)
- a description of any known bugs in the program

To recap, you will be turning in at least **four** files within `lab4.zip`
lab4.php, mailer.php, lab4.mysql, and ***lab4.txt***

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

If desired, jQuery can be used on the client side.

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).

In this lab, you will implement a web page that allows a logged-in user to enter an email address (in a textfield), a date and time to the nearest half-hour (using drop-down menus with <select>), and a message (in a textfield). This message and timestamp will be saved in a database on the server (and later sent to the email at the specified time). This web page is created using a file called `lab4.php`

The site must have a section where a new user may register with a username and password. Before using the form, which schedules messages to be sent, a user must provide the correct username and password (with which they registered). For the basic project, you may store the user's password in the database in plain text. **NOTE: although this is acceptable in your Lab 4, this should never be done on a live website. (see Extra Credit part a)**

Every half hour, a Cron job on the server will execute another PHP script at command line. This script is called `mailer.php`. This script will examine the database on the server and send every message whose timestamp is less-than-or-equal-to the current time. You must configure your system to execute this script by creating a CRON job. NOTE: `mailer.php` is **NOT** a web script…it should ONLY execute at command line. You can find details on creating a CRON job here: http://www.thesitewizard.com/general/set-cron-job.shtml

Hint: you can use your CSUN gmail account to send email. An example of doing this in PHP can be found here:
http://stackoverflow.com/questions/712392/send-email-using-gmail-smtp-server-from-php-page

The look and the feel of the site are entirely up to you. The site must function as specified.

EXTRA CREDIT
You will receive **up to 7%** extra credit **on your FINAL COURSE GRADE** for implementing additional features of Lab #4 as follows…

a) **(2% extra credit)** 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 transmitted nor 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.
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 they match, the user entered the right password, and will be logged in.

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

Use SHA-512 as the hash function. You can use the PHP method hash() to accomplish hashing: http://php.net/manual/en/function.hash.php

b) **(1% extra credit)** Ensure password strength by only allowing a user to register if they select a password with at least 8 characters, which is a combination of letters, numbers, and special characters like +, -, $, *, etc. The password must have at least one letter, one number, and one special character. This should be checked **BOTH** on the client side (using JavaScript) and the server side (in PHP).

c) (1% extra credit) Make sure that there is no way a user can attack your program using SQL injection by validating all inputs which are stored in the database. Make sure none of these inputs have any data except that which is expected. For instance, username should be a single alphanumeric word. The email address and timestamp should also be properly formatted. You may use regular expressions and/or any other PHP method to check that nothing illegal is inserted into the database.

d) (3% extra credit) 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, login, and message-scheduling to be called in response to user events by JavaScript or jQuery code running in the browser.

******REMEMBER: if you perform any extra credit, you must tell me about it in** `lab4.txt`
or…
IT WILL NOT BE GRADED!!!

Suggestions:

- Your MySQL database should consist of at least two tables: **user** and **message**.
  - Table **user** has three columns: userid (an auto-incrementing public key), username (**must** be unique for every row – in other words different users must have different usernames), and password. Those students doing extra credit part a should also add a fourth column for the salt. (An even more advanced secure approach is to mix the salt and the hashed password together in the password field…only your app knows which characters are which, and an attacker wouldn't know which part is password and which is salt unless they get your source code.)
  - Table **message** has 6 columns: msgid (an auto-incrementing public key), userid (foreign key matching the userid in the **user** table), email address, message, a timestamp (date & time), and *sent* (a Boolean, true if the message has been sent already, false otherwise).
    - When a message is first created by the user, in the **message** table its column *sent* should be initialized to FALSE
    - When the `mailer.php` script sends each message, it should update the column *sent* to TRUE

Grade Breakdown:
10%: Submission instructions were followed, and a minimum of 4 files are submitted, named according to spec

10%: JavaScript and PHP code has proper formatting, proper use of functions, and comments as appropriate

20%: Web page renders properly in Mozilla Firefox 17+

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

30%: PHP code and database abide by spec

Cheating: This project (like all projects in this class) 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 source code then both students will receive an F in the course. Note a person who gives his code to another student also fails the class (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.