CSCI 466 – Database
Spring 2017
Lab 8 (due Friday, April 21st at 5pm)
100 pts

For this lab, you're going to revise your NerdLuv php front-end to communicate with a web service via HTTP requests (both GET and POST) and JSON. You will also write the NerdLuv web service (using PHP with PDO to connect to a MySQL database).

The handout from AutoLab includes two files:

1) The MySQL table and initial insert statements are in *data.sql*. You should run these statements on your MySQL database on your VM
2) The specifications for the web service are in the *nerdluv.php*. You may add additional functions as you see fit, but be sure to implement the functions whose headers are included in that file. Be especially sure to return the JSON exactly as specified in the comments in this file.

Details

- You must use prepared statements in PDO to interact with the database
- Store your db username and login in the separate .txt file as specified in the nerdluv.php coments
- Match the JSON exactly to what is specified
- Use the php function file_get_contents to call the web service from your -submit files (refer to the stackoverflow entries on my resources page for details)
- If an error occurs in nerdluv.php, return an HTTP 400 error. Your front end php does not need to handle this error, it can behave as the forms currently do in the case of an error (display no matches, etc.)

Approach
First implement the web service and test it thoroughly. You can test by calling the .php file directly (without using the forms). Once you're sure the web service is working properly, modify the forms to utilize the web service via the file_get_contents function. Remember to utilize the error log in /var/log/apache2 of your vm to debug your PHP.

Grading Criteria will include:

- Your HTML output for all pages must pass the W3C **HTML validator**. (Not the PHP source code itself, but the HTML output it generates.)
- Your PHP code should not cause errors or warnings. Minimize use of the global keyword, use indentation/spacing, and avoid lines over 100 characters.
- Use **functions, parameters/return, included files/code**, loops, variables, etc. to avoid redundancy.
- For full credit, reduce the amount of large chunks of PHP code in the middle of HTML code. Replace such chunks with **functions** declared at the top or bottom of your file. You will also lose points if you use PHP print or echo statements in your forms. Insert dynamic content into the page using PHP **expression blocks**, <?= ... ?> , as taught in class.

- Another grading focus is PHP **commenting**. Put a descriptive comment header at the top of each file, **each function**, and each section of PHP code.
- **Format your HTML and PHP code** similarly to the examples from class. Properly use whitespace and indentation. Do not place more than one block element on a line or begin a block element past the 100th character.
- Part of your grade will also come from successfully uploading your files to the web. You should place your files into your public web space in a subdirectory named Lab8, so that it is possible to navigate to your page in the browser.

Submitting

Submit the following files as a single .zip to AutoLab:

- signup.php – this file should remain unchanged from the previous lab
- signup-submit.php – alter this file to call the web service (using POST) to add the new user
- matches.php – this file should remain unchanged
- matches-submit.php – alter this file to call the web service (using GET) and display the matches (if any)
- nerdluv.php – the web service
- db.txt – the include file with your db login info (so that I can test)

Be sure to upload your code to a Lab8 directory on your vm.