

American University: CSC 435 Web Programming

Homework Assignment 6: GeekLuv, an online dating website.

Due: April 24th, 2016. This will be our **final** homework assignment. I expect high quality work with no late assignments accepted.

Points: 100pts + Bonus.

Teamwork: maximum 2 people. Team members must provide an evaluation of each other's contribution.

This assignment is about making a simple multi-page "online dating" site that processes HTML forms with PHP. Online dating has become mainstream with popular sites such as Tinder, eHarmony, Match.com, OkCupid, Chemistry, and Plenty of Fish. And actually the website GeekLuv.com exists. We blatantly modified this name. If you don't know, "[Luv](#)" is a CIE color space. Notice helping a geek looking for love is no longer a niche market☺. However, we still try.

Your task for this assignment is to write HTML and PHP code for a **fictional** online dating site for desperate single geeks, called GeekLuv. Please turn in the following files:

- [geekluv.php](#), a front page that links to the other page (partially provided)
- [signup.php](#), a page with a form that the user can use to sign up for a new account.
- [signup-submit.php](#), the page that receives data submitted by [signup.php](#) and signs up the new user
- [matches.php](#), a page with a form for existing users to log in and check their dating matches
- [matches-submit.php](#), the page that receives data submitted by [matches.php](#) and show's the user's matches
- [common.php](#), a file containing any common code used by the above pages

Page Functions and flow

You can download the files from the blackboard. The first is the near finished [geekluv.php](#). I also provide a complete CSS file, [geekluv.css](#) with all of the page layout. Link to the CSS file from all of your pages and use its styles in your code. Feel

Index Page ([Geekluv.php](#)). Images are provided for you in the folder. But feel free to substitute with your own logo image.



where meek geeks meet

Welcome!



[Sign up for a new account](#)



[Check your matches](#)

This page is for single nerds to meet and date each other! Type in your personal information and wait for the nerdly luv to begin! Thank you for using our site.

Results and page (C) Copyright Geekluv Inc.



[Back to front page](#)

The provided **geekluv.php** has a header/footer that links to **signup.php** and **matches.php**. The file's contents are complete, but large parts of it are preated on other pages. The repeated parts should be turned into functions in **common.php** that are called by each page.

The “sign up” link leads to **signup.php**



where meek geeks meet

New User Signup:

Name:	<input type="text"/>
Gender:	<input type="radio"/> Male <input checked="" type="radio"/> Female
Age:	<input type="text"/>
Personality type:	<input type="text"/> (Don't know your type?)
Favorite OS:	<input type="text" value="Windows"/>
Seeking age:	<input type="text" value="min"/> to <input type="text" value="max"/>
<input type="button" value="Sign Up"/>	

This page is for single nerds to meet and date each other! Type in your personal information and wait for the geeky luv to begin! Thank you for using our site.

Results and page (C) Copyright GeekLuv Inc.



[Back to front page](#)

When submitted the form, it leads into the log in page (**signup-submit.php**):

Thank you!

Welcome to NerdLuv, David!

Now [log in to see your matches!](#)

This page is for single nerds to meet and date each other! Type in your personal information and wait for the geeky luv to begin! Thank you for using our site.

Results and page (C) Copyright GeekLuv Inc.

[← Back to front page](#)

Then it will ask you to log in (**matches.php**):

Returning User:

Name:

View My Matches



This page is for single nerds to meet and date each other! Type in your personal information and wait for the geeky luv to begin! Thank you for using our site.

Results and page (C) Copyright GeekLuv Inc.

[← Back to front page](#)

After you log in, you should see (**matches-submit.php**)

Matches for David

Jadzia Dax	
	gender: F
	age: 46
	type: ENFJ
	OS: Mac OS X
Ms Frizzle	
	gender: F
	age: 39
	type: ENTP
	OS: Mac OS X

Notice the there is no choice of gender. So if the user is male, the matches are female. This can be modified in extra credit portion.

Individual page specifications

Sign up form: the signup.php

The page has a header logo, a form to create a new account, and footer notes/images. The forms has the following fields:

Name: A 16 character box for the user to type a name.

Gender: Radio button for Male or Female.

Age: a 6-letter-wide text input box.

Personality type: A 6-character-wide text box A 6-character-wide text box allowing the user to type a Keirsey personality type, such as ISTJ or ENFP. The box should let the user type up to 4 characters. The label has a link to <http://www.humanmetrics.com/cgi-win/JTypes2.asp>.

Favorite OS: A drop-down select box allowing the user to select a favorite operating system. The choices are Windows, Mac, Linux, Windows is selected initially.

Seeking age: Two 6-character-wide text box for the user to specify the range of acceptable ages of the partner. The placeholder text of "min" and "max" are set initially.

Sign up: When pressed, submits the form for processing and go to **signup-submit.php**

Submitting the Sign-up Form (signup-submit.php)

When the user presses "Sign Up," the form should submit its data as a POST to [signup-submit.php](#). (The exact names and values of the query parameter(s) are up to you.) your PHP code should read the data from the query parameters and store it as described below. The resulting page has the usual header and footer and text thanking the user. The text "log in to see your matches!" links to [matches.php](#).

Your site's user data is stored in a file [singles.txt](#), placed in the same folder as your PHP files. We will provide you an initial version of this file. The file contains data records as lines in exactly the following format, with the user's name, gender (M or F), age, personality type, operating system, and min/max seeking age, separated by commas:

```
Angry Video Game Nerd,M,29,ISTJ,Mac OS X,1,99
Lara Croft,F,23,ENTP,Linux,18,30
Seven of Nine,F,40,ISTJ,Windows,12,50
```

Your [signup-submit.php](#) code should create a line representing the new user's information and add it to the end of the file. See the PHP `file_put_contents` function in PHP.net for help. Add a `\n` to each line in the file.

In all pages, assume valid data for the file's contents and form submissions. For example, no fields will be left blank or contain illegal characters (such as a comma). No user will resubmit data for a name already in the system.


View Matches Page (matches.php):

The [matches.php](#) page has a header logo, a form to log in and view the user's matches, and footer notes/images. You must write the HTML for the form. The form has one field:

- Name: A label and 16-letter box for the user to type a name. Initially empty. Submit to the server as a query parameter name.

When the user presses "View My Matches," the form submits its data as a GET request to [matches-submit.php](#). The name of the query parameter sent should be name, and its value should be the encoded text typed by the user. For example, when the user views matches for Rosie O Donnell, the URL should be:

- [matches-submit.php?name=Rosie+O+Donnell](#)



This page is for single nerds to meet and date each other! Type in your personal information and wait for the geeky luv to begin! Thank you for using our site.

Results and page (C) Copyright GeekLuv Inc.

[← Back to front page](#)

Viewing Matches (matches-submit.php):

When viewing matches for a given user, [matches-submit.php](#) should show a central area displaying each match. Write PHP code that reads the name from the page's name query parameter and finds which other singles match the given user.

The existing singles to match against are records found in the file [singles.txt](#) as described previously. You may assume that the name parameter is passed and will be found in the file. Below the banner should be a heading of "Matches for (name)". Below this is a list of singles that match the user. A "match" is a person with all of the following qualities:

- The opposite gender of the given user; (this needs to be extended for LGBT dating for extra features).
- Of compatible ages; that is, each person is between the other's minimum and maximum ages, inclusive;
- Has the same favorite operating system as this user; However, Mac OS can be compatible with Linux.
- Shares at least one personality type letter in common at the same index in each string.

For example, INTP and ESFP have 1 in common (P).

As you find each match, output the HTML to display the matches. I have user.png and userm.png

for female and male matches. Each match has an icon image to show the gender, and their age, personality type, and OS.

Matches for David



Jadzia Dax

gender: F
age: 46
type: ENFJ
OS: Mac OS X



Ms Frizzle

gender: F
age: 39
type: ENTP
OS: Mac OS X

Styling:

The styles you need are already given to you in [geekluv.css](#), but you still need to use proper tags and class attributes to make sure they are applied. Be mindful of the styles on forms and form controls. On the course web site are several screenshots of the various pages. Make sure that your form has the same width, colors, fonts, borders, etc. as in these examples. If you choose the right tags to represent your form, it should match. Make sure that form fields line up in columns by using a strong tag or column class so that each text label floats to the left and is 11em wide.

In [matches-submit.php](#), the matches are displayed in a div with class of match. First is a paragraph containing an image of the match, shown with a width of 150px, and the person's name to the right. The paragraph has a light blue background color. The section with the match's gender, age, etc. must be represented as an unordered list (ul).

Uploading and testing:

You can first test your scripts in your localhost in AMMPs. Change the permission of single.txt so that PHP can write to it. We demonstrated this in class. Look up for chmod r+x in terminal.

You can then upload all your files to a webserver. You can find a free webhost here:

<http://www.free-webhosts.com/free-php-webhosting.php>

Suggested Debugging:

- Use print and print_r statements to track down bugs. print_r(\$_GET) to see query parameters.
- Based on [nerdluv.php](#), write [matches.php](#) and [matches-submit.php](#) to work properly for existing users.
- First output everyone as potential matches first and then check the gender, age, etc. Focus on getting the correct File I/O.

Extra features:

Students can earn up to 10% of the total grades if they complete extra features.

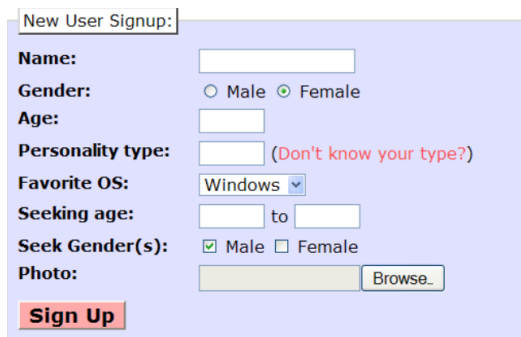
1. Extend the page to LGBT dating. This includes a new form field for gender preference and the ability to change the matches accordingly.

To do this, we'll add a new dimension to our user data. In the singles2.txt file we will now store a new piece of data that we'll call the "seek gender(s)" at the end of each line; this piece of data represents the gender(s) that the person is willing to date. The value will be either M (male), F (female), or MF (male and female). For example, if Oldspice Guy is a straight male, Alan Turing is a gay male, Nostalgia Critic is a bisexual male, and Rosie O'Donnell is a lesbian female, then their lines in the input file would look like this (notice the very last value on each line):

```
Oldspice Guy, M,36,ENTJ,Windows,1,99,F
Alan Turing,M,41,ESTP,Mac OS X,31,50, M,
Nostalgia Critic,M,28,ENTJ,Linux,12,79,MF
Rosie O'Donnell,F,46,ENFJ,Windows,30,50,F.
```

You need to download the extended version of singles.txt that includes this "seek gender" for every existing user. If you want to complete this extra feature, go get this new file and use it in your program

Modify your signup.php by adding a new line for Seek Gender(s): to the form, so that new users can sign up as straight, gay, or bisexual.



New User Signup:

Name:

Gender: ☐ Male ☒ Female

Age:

Personality type: (Don't know your type?)

Favorite OS:

Seeking age: to

Seek Gender(s): ☒ Male ☐ Female

Photo:

2. Robust page with form validation

If the user submits invalid data that doesn't match the above criteria to **signup-submit.php**, or if the user submits an empty name to matches-submit.php, do not show any matches or sign up the user. Instead display an error message of your choice, such as the one shown at left.

Add PHP code in **signup-submit.php** and **matchessubmit.php** that tests all query parameters submitted for validity. Use PHP regular expressions to do this. Specifically, you

must check the following aspects of each query parameter that is submitted using proper strict regular expressions that allow only these patterns:

- The name must not be blank (both pages).
- The age submitted must be an integer and must be between 0 and 99 inclusive.
- The gender submitted must be male or female. (No other values such as "robot" are allowed.)
- The Keirsey personality type must be a 4-letter string whose letters come from the Keirsey personality dimensions: I or E for the 1st letter, N or S 2nd, F or T 3rd, and J or P 4th.
- The favorite operating system must come from the choices provided.
- The seeking min/max ages submitted must be integers, must be between 0 and 99 inclusive, and the minimum seeking age must be less than or equal to the maximum.
- (if you do Extra Feature #1) The seeking gender(s) must have at least one of the two boxes checked.

Also make your pages robust against the following simple basic input errors:

- Re-submitting to **signup-submit.php** a person who is already in the file.
- Trying to view matches on **matches-submit.php** for a person who isn't in the file.

3. User photos.

When the user views his/her matches, instead of displaying the default image of user.jpg, instead display the proper image for each individual user, as shown at left. The file names exactly match the users' names, except that the file names are all lowercase and with spaces replaced by dashes.

But the system won't have photos for new users that sign up (**For example, Ted Cruz**). So you should modify **signup.php** to allow new applicants to submit photos. To do uploading of photos, add a file input box to **signup.php** that allows the user to browse for a file to upload. Use an input tag with type of file. (See screenshot below.) When **signup-submit.php** receives a POST, it will save this photo onto the server into a subdirectory named images/ using the naming system described (such as ted_cruz.jpg) and show it when subsequent users search for matches. You may assume that every user submits a valid JPEG image file and that the images/ folder already exists.

When saving a file into a directory like **images/**, you may get permission issues. Make sure that your images/ folder on the server has both "Write" and "Execute" permissions enabled for all users. You may also want to add the line below to your code after saving any user's new image, so that when you save a new image file to Webster, it is given full permission so that you can move/delete it. Without this line, only the apache user that PHP runs as will be able to delete any newly added user image files.

Turn in your work

1. Upload your work on a web host that can host php scripts. You can find some free

ones here:

<http://www.free-webhosts.com/free-php-webhosting.php>

2. Upload all your file as a .zip onto blackboard.
3. Learn Git and upload your work on to GitHub. (<https://github.com/>) Then I can check out your work directly to my computer.

Grading scheme:

To get a B and above, your page should function without any error. The user can go back to the front page when needed. Returning users will be remembered.

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 material from the first four weeks of class and the first six book chapters.

Use variables liberally; for example, when accessing data from arrays, store each element of data as its own variable with a meaningful name, which makes the code easier to understand than arbitrary indexes like `$a[7]`.

A major grading focus is redundancy. Some HTML sections are repeated or shared, and you may also have PHP code statements that are repeated. Use functions, parameters/return, included files/code, loops, variables, etc. to avoid redundancy. If you have HTML or PHP code that is shared or redundant between multiple pages, place it into functions in [common.php](#). You can include your [common.php](#) in your other pages.

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. Insert dynamic content into the page using PHP expression blocks, `<?= ... ?>`, as taught in class.

Another grading focus is PHP commenting. We expect more comments here. Please describe the purpose of your code.