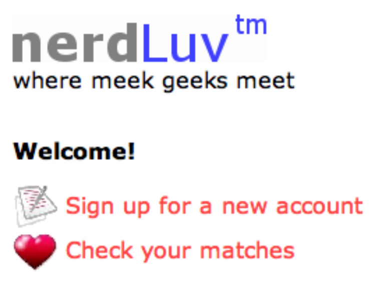
University of Washington, CSE 190 M
Homework Assignment 4: NerdLuv

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 eHarmony, Match.com, OkCupid, Chemistry, and Plenty of Fish. Your task for this assignment is to write HTML and PHP code for a fictional online dating site for desperate single geeks, called **NerdLuv**. Turn in the following files:

- **signup.php**, a page with a form that the user can use to sign up for a new account
- **confirm.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
- **results.php**, the page that receives data submitted by matches.php and show's the user's matches
- **common.php** (optional), a php file containing common code

There are some **provided files** on the web site. The first is a complete version of the site's front page, **index.php**. This front page simply links to your other pages. The other complete provided files are **top.html** and **bottom.html**, which contain common header/footer HTML code that you should include in your other pages. We also provide a complete CSS file **nerdluv.css** with all of the page styles. You should link to this CSS file from all of your pages and use its styles in your code. You should be able to fully style all pages using the styles in **nerdluv.css** only.

Index Page (index.php) and Overall Site Navigation:



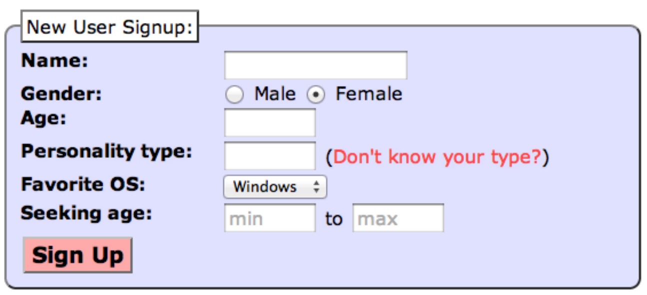
Welcome!

Sign up for a new account

Check your matches

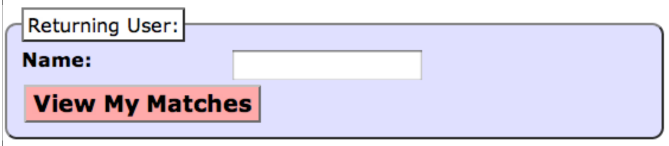
The provided **index.php** page has a header logo, links to **signup.php** and **matches.php**, and footer notes/images. You do not need to modify this file, but you should put it in the same folder with your other files and upload it to Webster with your files.

The "Sign Up" link leads to **signup.php** (left below), and "Check matches" to **matches.php** (right below):



New User Signup:

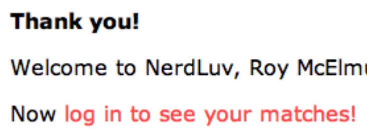
Name:	
Gender:	Male [✓ Female]
Age:	
Personality type:	(Don't know your type?)
Favorite OS:	Windows
Seeking age:	min to max



Returning User:

Name:

When submitted, the Signup page looks like this:

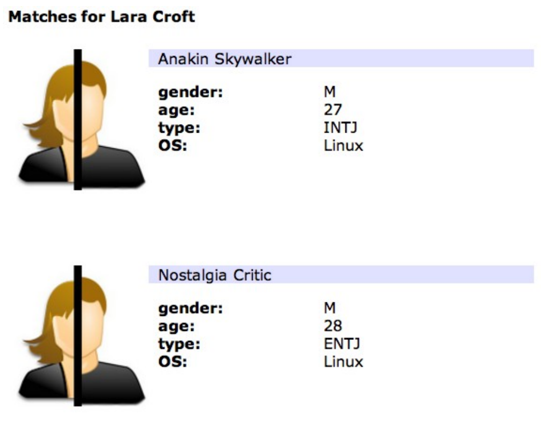


Thank you!

Welcome to NerdLuv, Roy McElmurry!

Now log in to see your matches!

When submitted, the View Matches form looks like this:



Matches for Lara Croft

Anakin Skywalker

gender:	M
age:	27
type:	INTJ
OS:	Linux

Nostalgia Critic

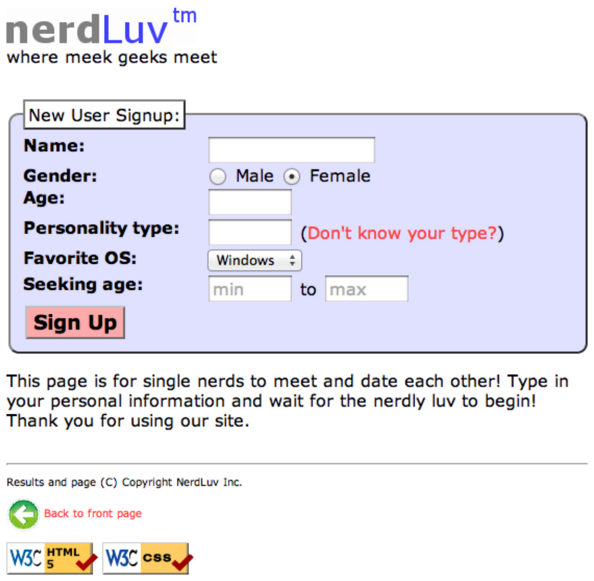
gender:	M
age:	28
type:	ENTJ
OS:	Linux

The details about each page's contents and behavior are described on the following pages.
Screenshots in this document are from Mac OS X in Chrome, which may differ from your system.

Sign-Up Page (signup.php):

The **signup.php** page has a header logo, a **form** to create a new account, and footer notes/images. You must write the HTML code for the form. The form should contain the following labeled fields:

- **Name:** A 16-character box for the user to type a name.
- **Gender:** Radio buttons for the user to select a gender of Male or Female. When the user clicks the text next to a radio button, that button should become checked. Initially Female is checked.
- **Age:** A 6-letter-wide text input box for the user to type his/her age in years. The box should allow typing up to 2 characters.
- **Personality type:** 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 OS X, and Linux. Initially "Windows" is selected.
- **Seeking age:** Two 6-character-wide text boxes for the user to specify the range of acceptable ages of partners. The box should allow the user to type up to 2 characters in each box. Initially both are empty and have placeholder text of "min" and "max" respectively. When the user starts typing, this placeholder text disappears.
- **Sign Up:** When pressed, submits the form for processing as described below.


nerdLuv tm
where meek geeks meet

New User Signup:

Name:	
Gender:	○ Male [● Female]
Age:	
Personality type:	(Don't know your type?)
Favorite OS:	Windows
Seeking age:	min to max

Sign Up

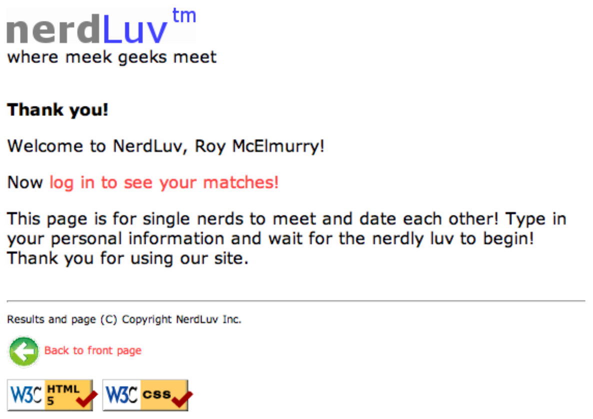
This page is for single nerds to meet and date each other! Type in your personal information and wait for the nerdly luv to begin! Thank you for using our site.

Results and page (C) Copyright NerdLuv Inc.

Back to front page

Submitting the Sign-Up Form (confirm.php):

When the user presses "Sign Up," the form should **submit** its data as a POST to **confirm.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**.


nerdLuv tm
where meek geeks meet

Thank you!

Welcome to NerdLuv, Roy McElmurry!

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 nerdly luv to begin! Thank you for using our site.

Results and page (C) Copyright NerdLuv Inc.

Back to front page

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 **confirm.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 book Chapter 5 or the lecture slides.

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.

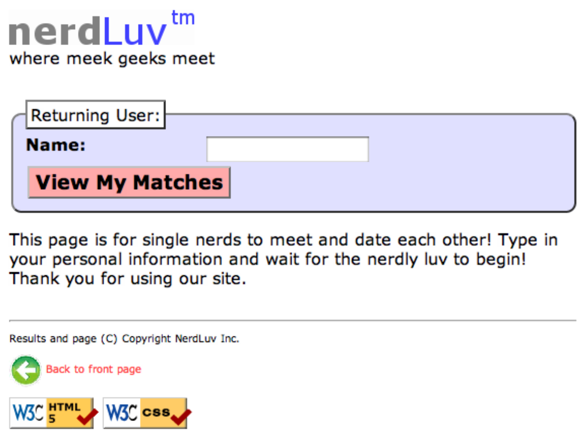
View Matches Page (matches.php):

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 **results.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:

- **results.php?name=Rosie+O+Donnell**


nerdLuv tm
where meek geeks meet

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 nerdly luv to begin! Thank you for using our site.

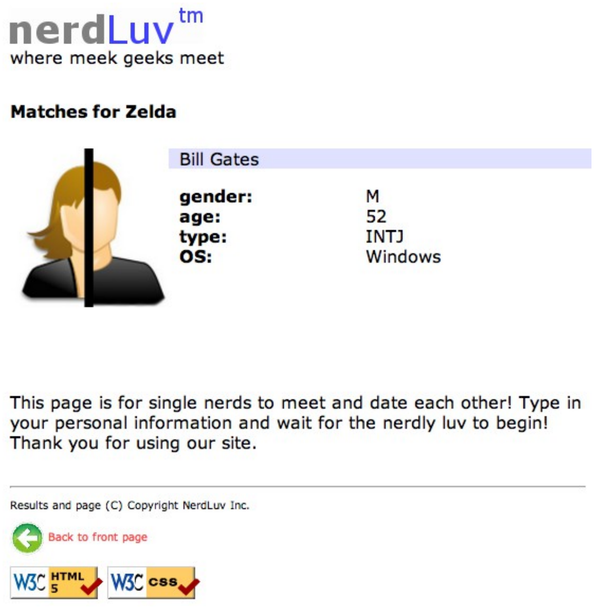
Results and page (C) Copyright NerdLuv Inc.
Back to front page

Viewing Matches (results.php):

When viewing matches for a given user, **results.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;
- 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;
- Shares **at least one personality type letter in common** at the same index in each string.
  For example, I<u>S</u>T<u>P</u> and E<u>S</u>F<u>P</u> have 2 in common (S, P).


nerdLuv tm
where meek geeks meet

Matches for Zelda

Bill Gates

gender:	M
age:	52
type:	INTJ
OS:	Windows

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 NerdLuv Inc.
Back to front page

As you find each match, output the HTML to display the matches, in the order they were originally found in the file. Each match has the image **user.jpg** below, the person's name, and an unordered list with their gender, age, personality type, and OS.

http://www.cs.washington.edu/education/courses/cse190m/12su/homework/4/images/user.jpg

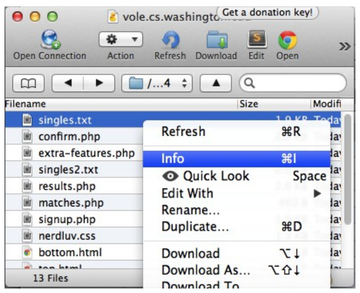
Styling:

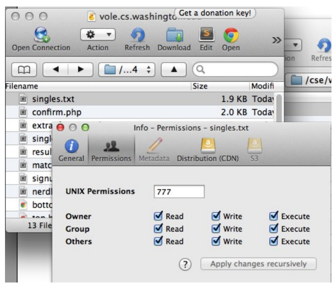
The styles you need are already given to you in **nerdluv.css**, but you still need to use proper tags and `class` attributes to make sure they are applied. Be mindful of the styles on forms and form controls. On the course web site are several screenshots of the various pages. Make sure that your form 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 **results.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:

Upload all files to Webster to test them; include **index.php** , **top.html**, and **bottom.html** even if you won't modify them. You must change **permissions** on **singles.txt** so that PHP can write to it. In Cyberduck, right-click **singles.txt** and choose **Info...**. Enable **Group Write** by checking the appropriate box to the right. These screenshots were taken on Mac OS X, so your interface may differ slightly.





Suggested Development Strategy and Hints:

- Based on **index.php**, write **matches.php** and **results.php** to work properly for existing users.
  - Write an **initial version** that outputs *every* person, even ones who aren't compatible "matches." This way you can debug your file I/O, styles, etc. Then add checks like gender, age, and OS. Focus on the PHP code and behavior first, as opposed to style details (CSS is not an emphasis of this assignment).
- Write **signup.php** and **confirm.php**. If you finish the match page you'll understand forms, making the signup page easier. This is tough; there are more parameters to manage, and you must write to a file.

Use **debug `print` and `print_r` statements** to track down bugs. For example, you can `print_r($_GET);` or `$_POST` to see the query parameters submitted. Use **Chrome Dev Tools** and also **View Source** to find HTML output problems.

Recall that form controls must have `name` attributes. Sometimes you must also add a `value` to affect how data is sent. Test a form by setting its `action` to **http://webster.cs.washington.edu/params.php**, which prints debug info.

Implementation and Grading:

Your HTML output for all pages must pass the W3C **HTML validator**. (Not the PHP source code itself, but the HTML output it generates.) Do not use HTML tables. Since we are using HTML **forms**, choose proper form controls and set their attributes accordingly. Properly choose between GET and POST requests for sending data.

Your PHP code should not cause errors or warnings. Do not use 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.

Some HTML sections are shared redundantly between your PHP pages, found in the provided files **top.html** and **bottom.html**. Include these files as appropriate in your other pages using the PHP `include` function.

A major grading focus is **redundancy**. Use **functions, parameters/return, included files/code**, loops, variables, etc. to avoid redundancy. If you have PHP code you want to share between multiple pages, you may turn in an optional file named **common.php** containing this code. 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, similar to CSE 14x. Put a descriptive comment header at the top of each file, **each function**, and each section of complex 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.

Please do not place a solution to this assignment online on a publicly accessible web site. Part of your grade will come from successfully uploading your files to the **Webster** server in the directory:

- https://webster.cs.washington.edu/*your_uwnetid*/hw4/