

University of Washington, CSE 154

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:

- **home.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

There are some **provided files** on the web site. The first is a mostly complete version of the site's front page, **home.php**. We also provide a complete CSS file **nerdluv.css** with all of the page styles. Link to this CSS file from all of your pages and use its styles in your code. You can fully style all pages using the styles in **nerdluv.css** only.

Home Page (home.php) and Overall Site Navigation:



Welcome!



[Sign up for a new account](#)



[Check your matches](#)

The provided **home.php** has a header/footer and links to **signup.php** and **matches.php**. This file's contents are complete, but large parts of it are repeated 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** (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:

Seeking age: to

Sign Up

When submitted, the Signup page looks like this:

Thank you!

Welcome to NerdLuv, Marty Stepp!

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


Returning User:


Name:

View My Matches

When submitted, the View Matches form looks like this:

Matches for Lara Croft

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

	Marty Stepp
gender:	M
age:	30
type:	ISTJ
OS:	Linux

The details about each page's contents and behavior are described on the following pages. Screenshots in this document are from Windows in Firefox, 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 contains 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 Keirse 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.

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 book Chapter 5 or the lecture slides. You will have to add a line break (`\n`) after each line of data that you add to the file.



New User Signup:

Name:

Gender: Male Female

Age:

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

Favorite OS:

Seeking age: to

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](#)



Thank you!

Welcome to NerdLuv, Marty Stepp!

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](#)



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](#)

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;
- 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, ISTP and ESFP have 2 in common (S, P).

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](#), the person's name, and an unordered list with their gender, age, personality type, and OS.

<https://webster.cs.washington.edu/images/nerdluv/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 [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`).



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 nerdluv to begin! Thank you for using our site.

Results and page (C) Copyright NerdLuv Inc.

[Back to front page](#)



Matches for Lara Croft

Anakin Skywalker

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

Nostalgia Critic

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

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

Results and page (C) Copyright NerdLuv Inc.

[Back to front page](#)



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 right.

Add PHP code in [signup-submit.php](#) and [matches-submit.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:

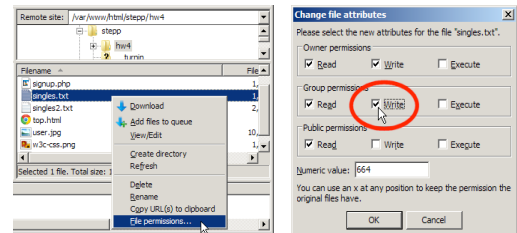
- The **name** must not be blank (both pages).
- The **age** submitted must be an integer and must be between 0 and 99 inclusive.
- The **gender** submitted must be male or female. (No other values such as "robot" are allowed.)
- The Keirsey **personality type** must be a 4-letter string whose letters 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 #2) 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.

Uploading and Testing:

Upload all files to Webster to test them. You must change **permissions** on [singles.txt](#) so that PHP can write to it; else you will see "Permission Denied". In FileZilla, right-click [singles.txt](#) in the right pane and choose **File Permissions...**. Enable **Group Write** by checking the box shown at right.



Suggested Development Strategy and Hints:

- Based on [home.php](#), write [matches.php](#) and [matches-submit.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 [signup-submit.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 **print** and **print_r** statements to track down bugs. For example, `print_r($_GET);` or `print_r($_POST);` to see query parameters submitted. Use **Firebug** and **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.

Extra Features for CSE Majors:

CSE majors must complete some additional requirements **specified in a separate document on the course web site**. If non-majors want to complete some extra features, they can earn extra late days for doing so.



Error! Invalid data.

We're sorry. You submitted invalid user information. Please go back and try again.

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

Results and page (C) Copyright NerdLuv Inc.

[Back to front page](#)



Implementation and Grading:

All of your HTML, CSS, and PHP code should follow the guidelines in our **style guide** on the class web site. Your HTML output for all pages must pass the W3C **HTML validator**. (*Not the PHP source code itself, but the HTML output it generates. To validate, View Source on your pages in the browser and copy/paste this into the W3C validator manually.*) 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. 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 (plus the section on regular expressions). 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, similar to CSE 14x. 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. Your source files should have proper indentation, but it's okay if their HTML output (View Source) does not.

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/

Copyright © Marty Stepp / Jessica Miller, licensed under Creative Commons Attribution 2.5 License. All rights reserved.