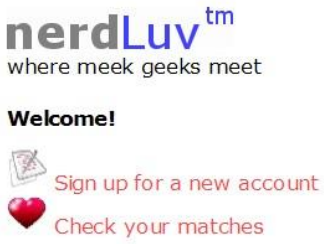
NerdLuv Web Site

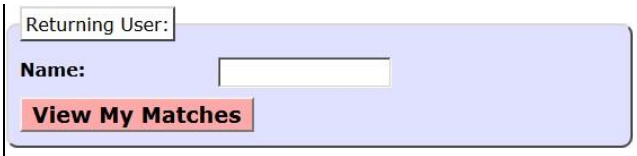
This assignment is about making a simple multi-page "online dating" site that processes HTML forms with Node.js. Your task for this assignment is to write HTML and JavaScript code for a fictional online dating site for single geeks, called **NerdLuv**.

There are some **provided files** on the web site. The first is a complete version of the site's front page, **index.html**. This front page simply links to your other pages.

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



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

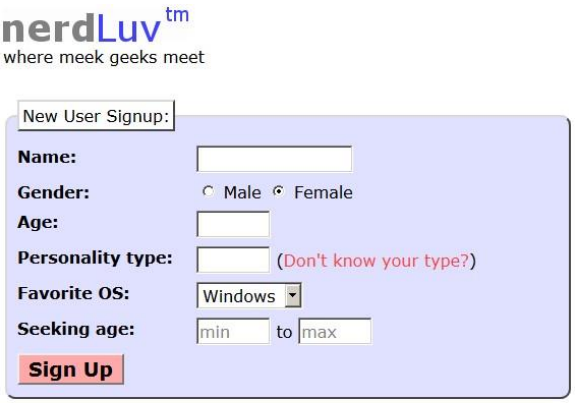


When submitted, the Signup page looks like this:

When submitted, the View Matches form looks like this:

Sign-Up Page (signup.html):

The **signup.html** page has a **form** to create a new account. 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..
- **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.

Submitting the Sign-Up Form (signup-submit.php):

When the user presses "Sign Up," the form should **submit** its data to **signup-submit.js**. (The exact names and values of the query parameter(s) are up to you.) Your Javascript code in node.js should read the data from the query parameters and store it as described below. The resulting page thanks the user. The text "log in to see your matches!" links to **matches.js**.

Your site's user data is stored in a file **singles.txt**, placed in the same folder as your web site 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.js** code should create a line representing the new user's information and add it to the end of the file.

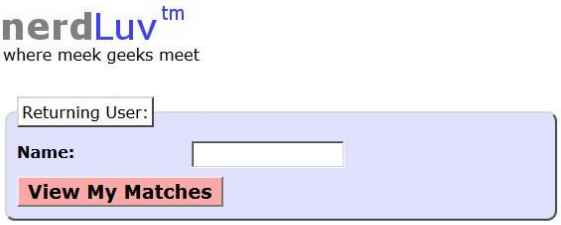
In all pages, **assume valid data** for the file's contents and form submissions. For example, no fields will be left blank or contain illegal characters (such as a comma). No user will resubmit data for a name already in the system.

View Matches Page (matches.js):

The **matches.js** page has a header logo, a **form** to log in and view the user's matches. 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 to **matches-submit.js** The name of the query parameter sent should be `name`, and its value should be the encoded text typed by the user.



Viewing Matches (matches-submit.js):

When viewing matches for a given user, **matches-submit.js** should show a central area displaying each match. Write JavaScript 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.