Zack Griffith

Mike Brennan

Brian Engal

Dr. Tian

4 December 2018

SugarDaddy.com

**Purpose of the project:**

The purpose of the project is to have a wholesome, inviting website where younger people can meet older people and vice versa. Our website allows younger people to become financially stable by meeting older wealthier people. Also, SugarDaddy.com helps older folk who are getting lonely in their older age meet younger people to share their golden years with. We also want to make sure your matches extend beyond the financial to include physical and emotional needs. We are an alternative to existing websites that are for creating financially dependent relationships. We believe that we provide tangible benefits to society through our services.

**Software Requirements:**

* Login page
* If the user does not have an account, they must be able to click on a “Create an account” button that takes them to the registration page.
* If the user has an account, they can login using this page by entering their email and password which takes them to the results.php where they can see their matches.
* Must encrypt password
* Registration page
* Must be connected to database
* Must post data entered to database.
* Asks for: Name, Password, Email, Phone Number, Age, Zip Code, Type of user, Income, Hair Color, Eye Color, Height, Animal preference, Religious or not, Whether you prefer rural or urban living, Can you cook, preferred vacation, Relationship Status, About yourself, Horoscope, favorite cereal, Who are you looking for, and shoe size.
* Submit button takes the user back to login page, and posts data entered to the database.
* Database
* Must store 24 columns of user entered data from registration page
* Must be a cloud database
* Must be remotely accessible
* Results page
* Must output top three matches and show their name, email, about themselves, and their match score.
* Must use algorithm to determine matches based on all the information in database.
* If the user does not like the top three matches, they can pull up the next three matches.
* Must have logout button to end session

**Weekly Status Report:**

**Week 1**

Dr. Tian,

Our group met on Sunday to work on our software engineering final project.

First, we had a meeting to discuss our goals for the following work session. We decided to proceed with our dating website idea. To that end, we:

- Decided to start with a rudimentary profile creation page

- Made decisions on characteristics for a profile, such as height, age, and income

- Created an initial version of an HTML page for profile creation

- Created an initial version of an AWS database to house user data

Goals for the coming week include:

- Connect the HTML to the database with PHP

- Test the profile creation page

- Create an initial version of a matching algorithm

Sincerely,

Michael

**Week 2**

This week,

-Zack worked on improving the registration pages.

-Mike wrote PHP to connect our database to the HTML.

-Brian worked on the CSS for our webpages.

Problems:

-Our decision to create our HTML pages from scratch means that our developers will have a good understanding of the page structure, but progress is slower than it would be with a premade online template.

- Several members are uncomfortable with Github and version control, and we need to work on learning how to use it.

- We still need to write our matching algorithm

Goals for this week:

- Write the matching algorithm

- Define our class structure in PHP.

- Make a functional registration page

**Week 4**

**Testing results**

Testing Results

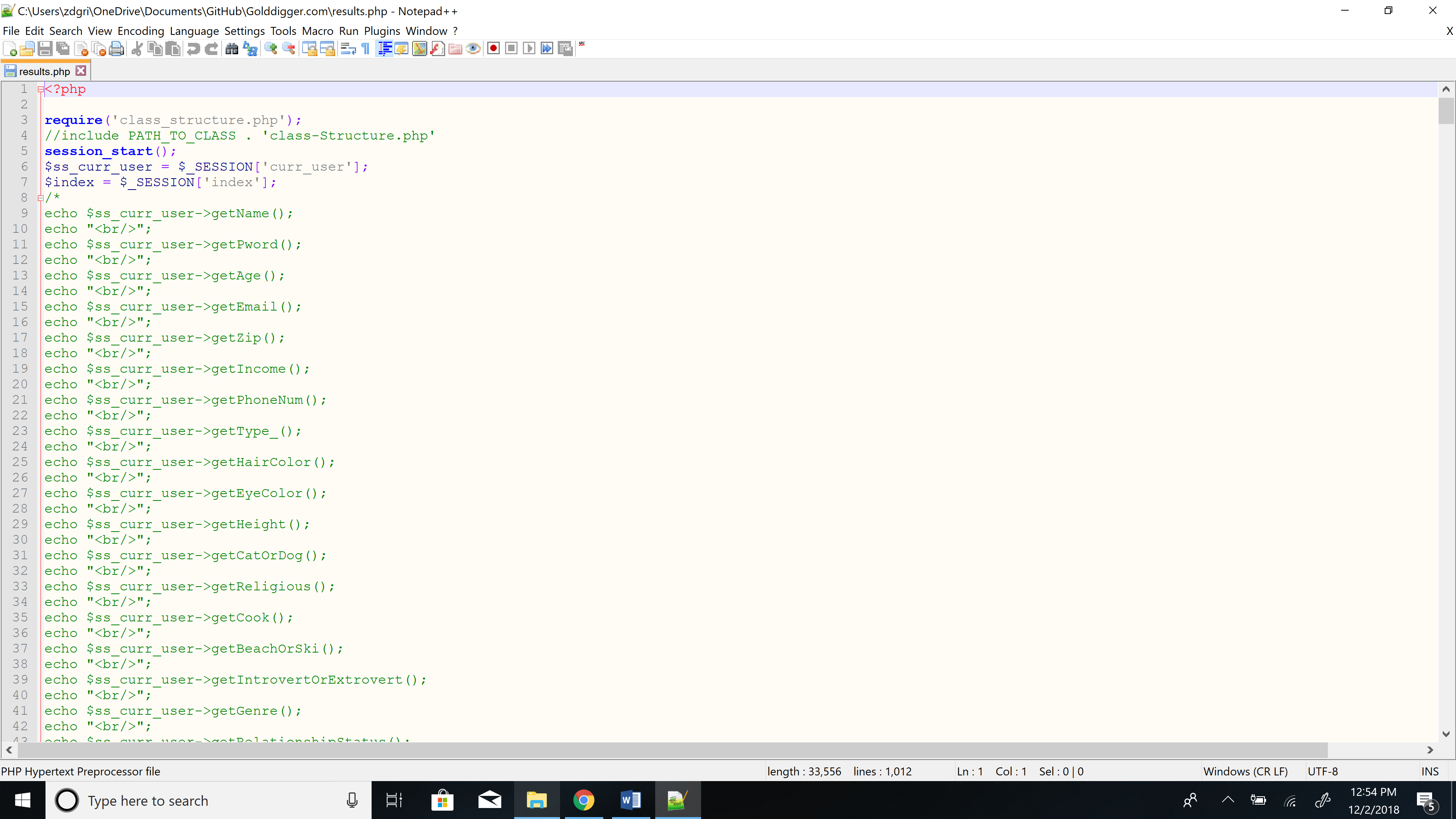
The first tests that we ran were very primitive and were focused on making sure that the right values were being stored in the correct variables. These tests often focused on using PHP echo statements to allow us to better understand what was happening in the background and why specific frontend elements were being displayed the way they were. A large test that we performed in this section was used to test the functionality of the PHP Classes. Some of the time, our code would execute, yet our variables were all set to 0. After researching this issue, we found that the “this” identifier was required to use private variables in getter and setter methods. Our echo statements were crucial at this point in our testing, since our entire program relies on the PHP Classes to function properly.

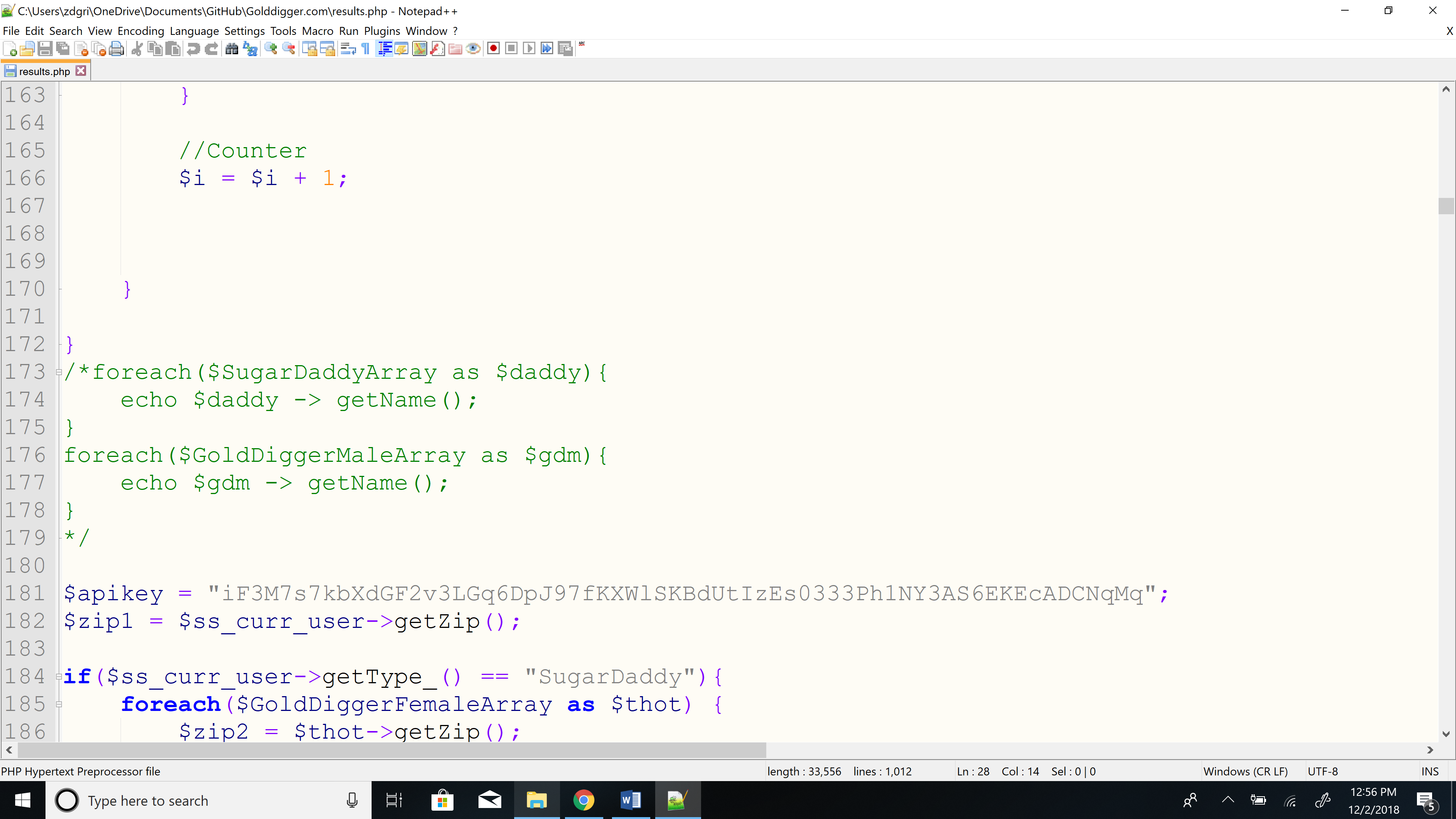
Another test that we ran was debugging the PHP Sessions that we had created. After a user logs in, we wanted to obtain all their information through a PHP Session since the user would travel through several pages. To test that the PHP Sessions were working properly, we created a Testing account and attempted to display the account information on another page after the PHP Session was created. This allowed us to further test that our getters and setters were functioning in the way that we thought they were. At this point we realized that one of our methods was not functioning. We were able to determine the bug and immediately fixed the problem.

The next test that we ran was sorting our matched users by a score in ascending order. This allowed us to check if the sorting algorithm was functioning properly as we wanted to display the highest matched people and then work our way down to the lower scored users. When running our sorting function initially, we realized that it was sorting in reverse order and displaying users with the lowest scores first. After fixing our sorting function, we were able to print the users in the correct order.

For the next few tests, we ran our webpage with an attached cloud database to simulate the user experience on our site. For each of the four user account types, we created users that fit into several categories and would display on our matching results page. Once a user logs in, they are taken to this matching page and we display the top three results. If the user decides that they are not interested in the users displayed on the page, the next three results will be displayed when the user clicks on the “Not Interested Button”. Once this button is clicked by the user, three more results will be shown. This action may be done as many times as needed and will continue displaying new, unseen users per that session.

Throughout our entire project, we were able to create two files that were used as testing. The first one was test.php. This file was used to test retrieval of user data from the database as well as testing user creation through PHP classes. After the user objects were created, we tested that we could call the object’s methods and display the correct results. This file was crucial in our correct implementation on the LoginPage.php because the same code was used when beginning our user sessions in PHP. The second file that was used heavily for testing was the ziptest.php file. This file was used to test the functionality of the Google Maps API with two zip codes and returning the distance. At first, we had trouble parsing the output since it came in JSON format, so we had to learn how to process the output. Once we were able to process the output, we were able to view the correct results for the distance between two zip codes.



We used these echo statements to show that the database was connected and that the class functions worked for retrieving and outputting information to the user. 

This for Each loop was used in testing to show that not only our syntax was correct but that the arrays could be used in our code for the purposes we needed it.