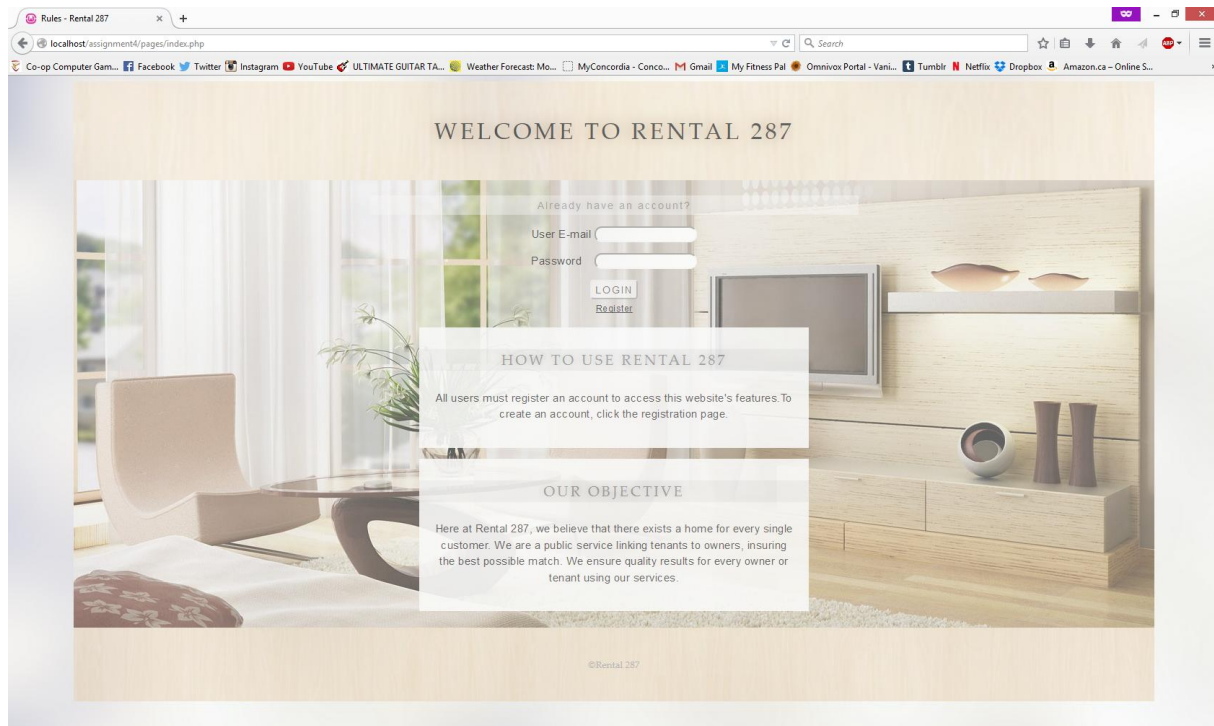


Jhayzle Arevalo
ID: 7344333

ASSIGNMENT 4 - PROJECT DESCRIPTION

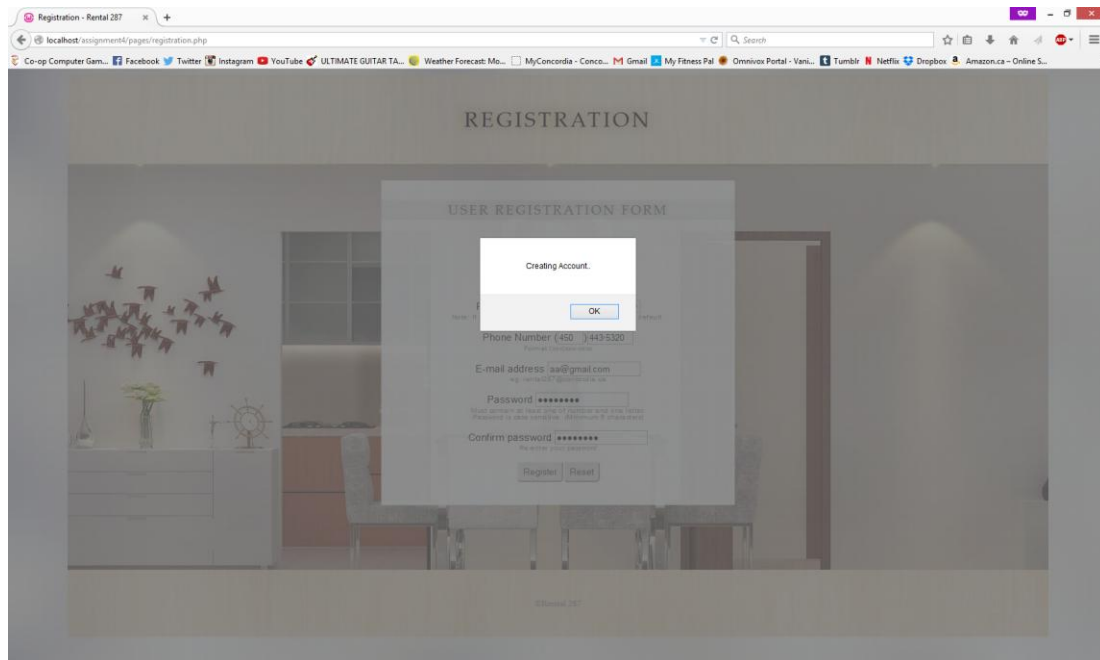
At the root of the file, there is a file called create_rental_7344333.php which allows one to create the database from scratch using phpmyadmin. Included is the creation of the 7 tables and 10 users of the database rental_7344333.

Inside the 'pages' folder is located all the pages accessible through the website. Homepage.php is to be launched first, which prompts the user to enter the website. After clicking "ENTER", the user enters the index.php page also located in 'pages'.

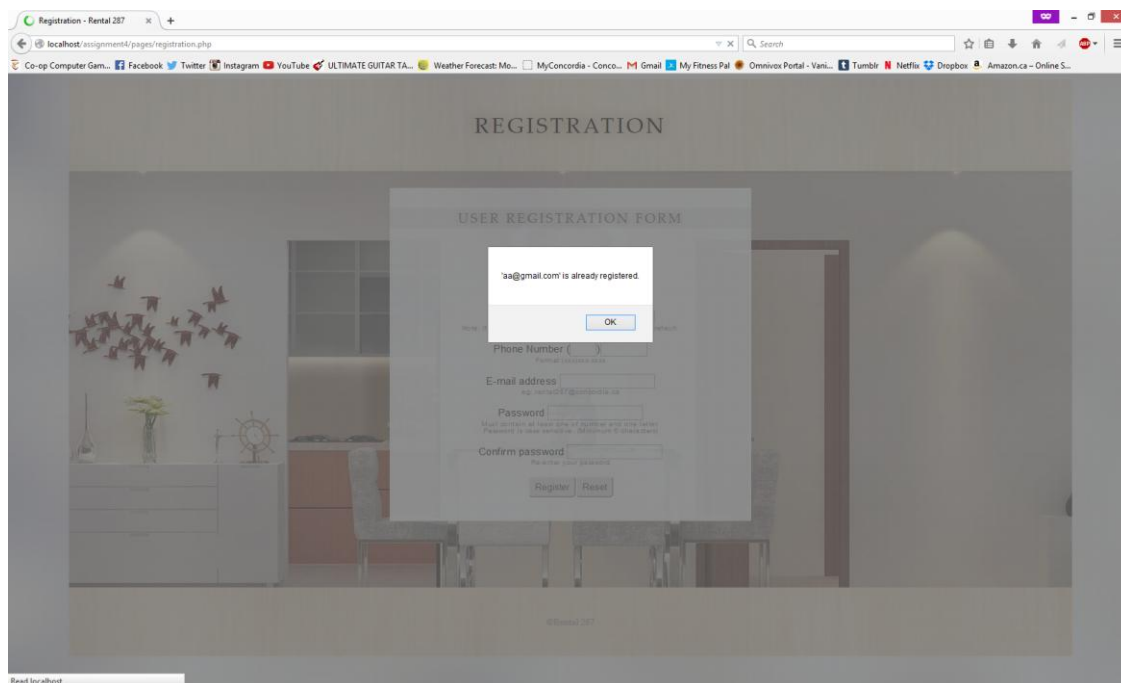


The user can either register for a new account or login using existing data. As you can see above, there is currently no navigation bar. That is because the navigation bar or menu differs from user to user, which I will explain later on. The small form at the top is the loginWidget.php file inside the 'includes' folder. This is in a separate folder because this widget will only appear if a session has not been started.

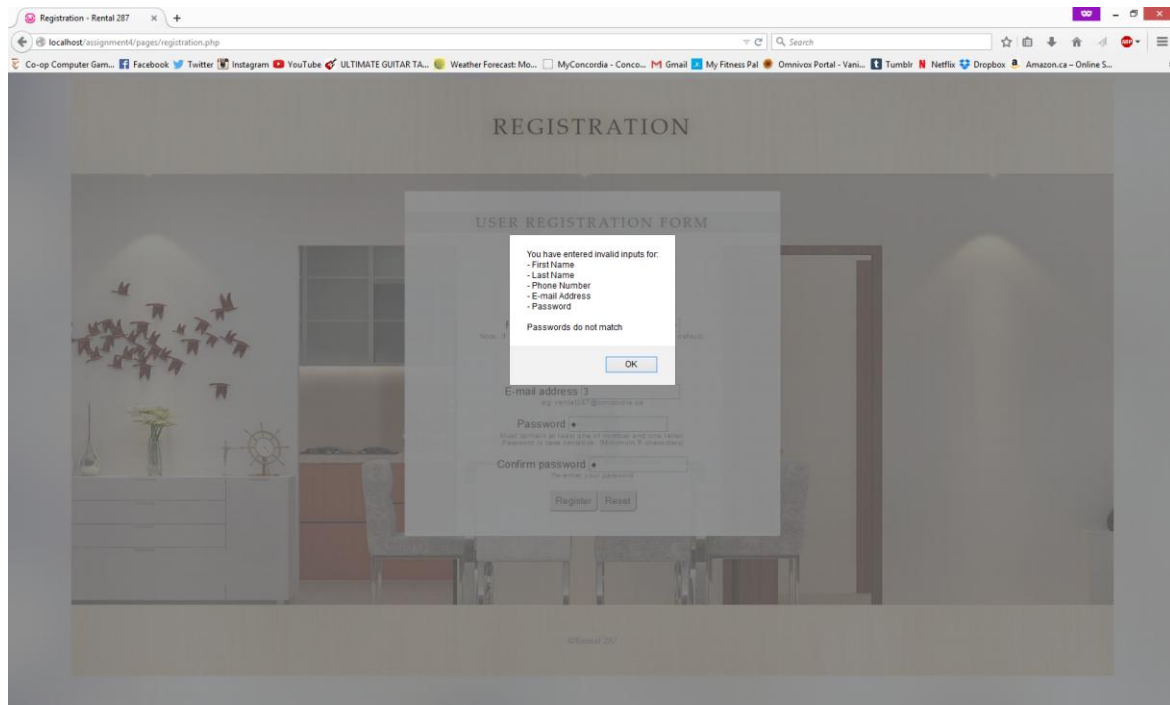
At the registration page, after entering data, alert boxes will appear to show the results:



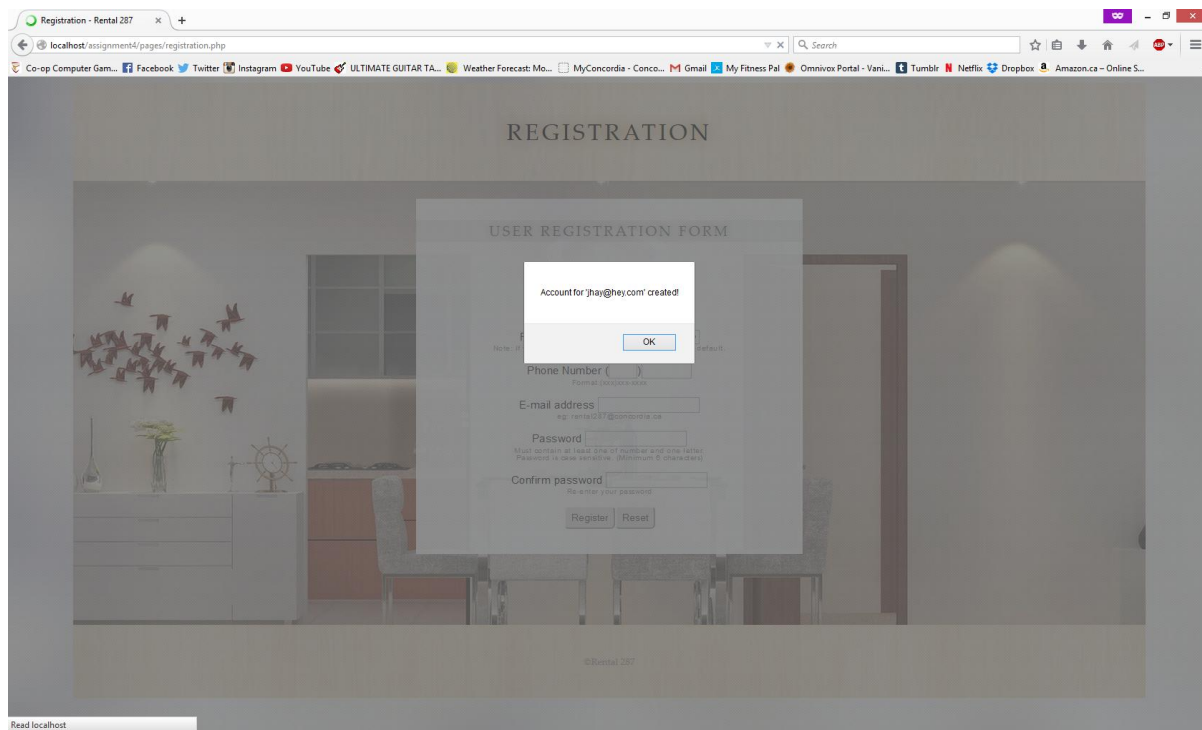
This alert box just tells us that the information was submitted. After clicking 'ok', the user is redirected back to the registration.php page.



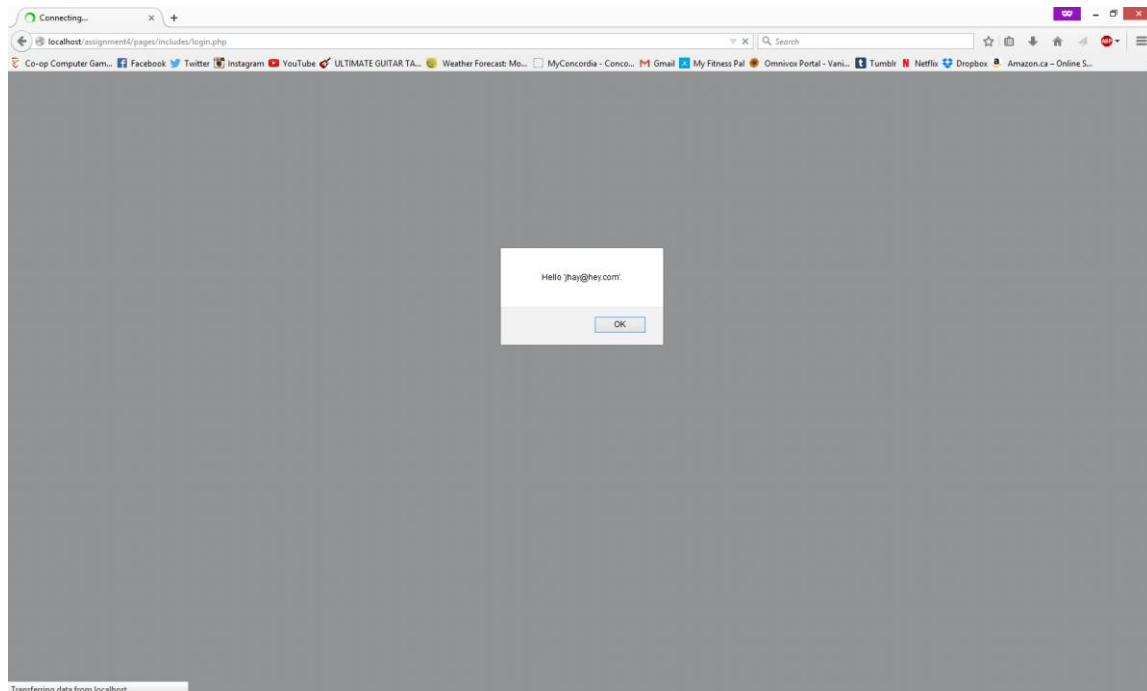
This alert box tells us that the e-mail we tried to use to register has already been registered with another account. Since I did not put a username variable, I use the e-mail address as username which is why this has to be unique. After clicking 'ok', the user is redirected back to the registration.php page.



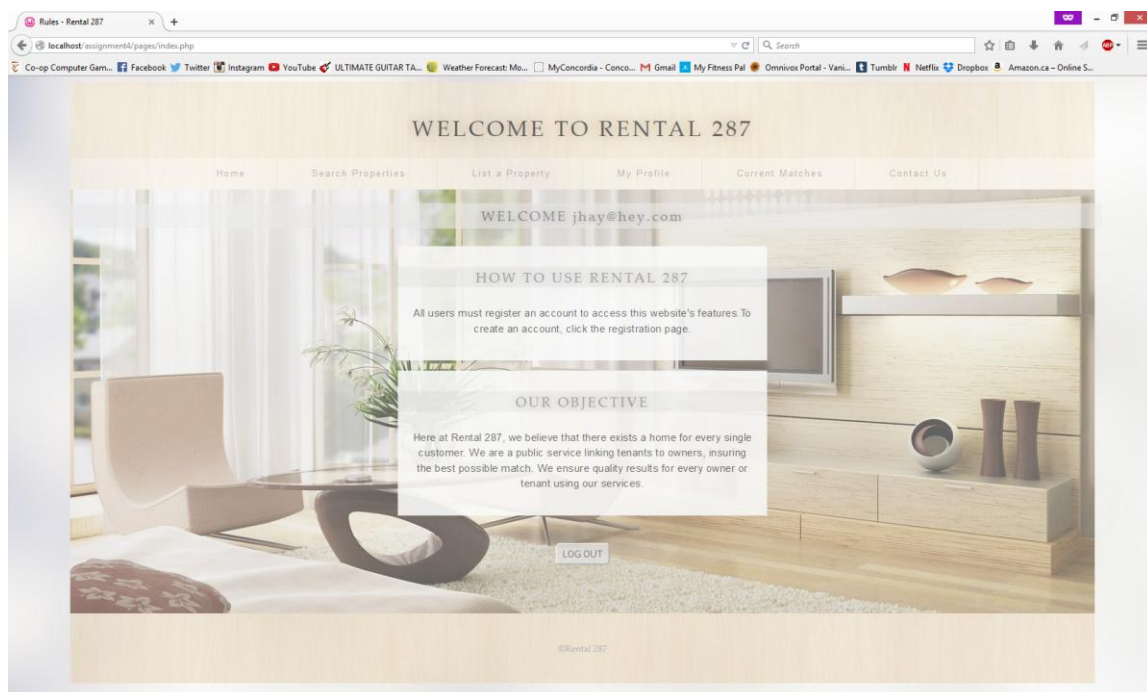
This alert box notifies us what field inputs are incorrect. After clicking okay, the alert box closes and the user can modify the inputs. This alert is created through the JavaScript file functions.js.



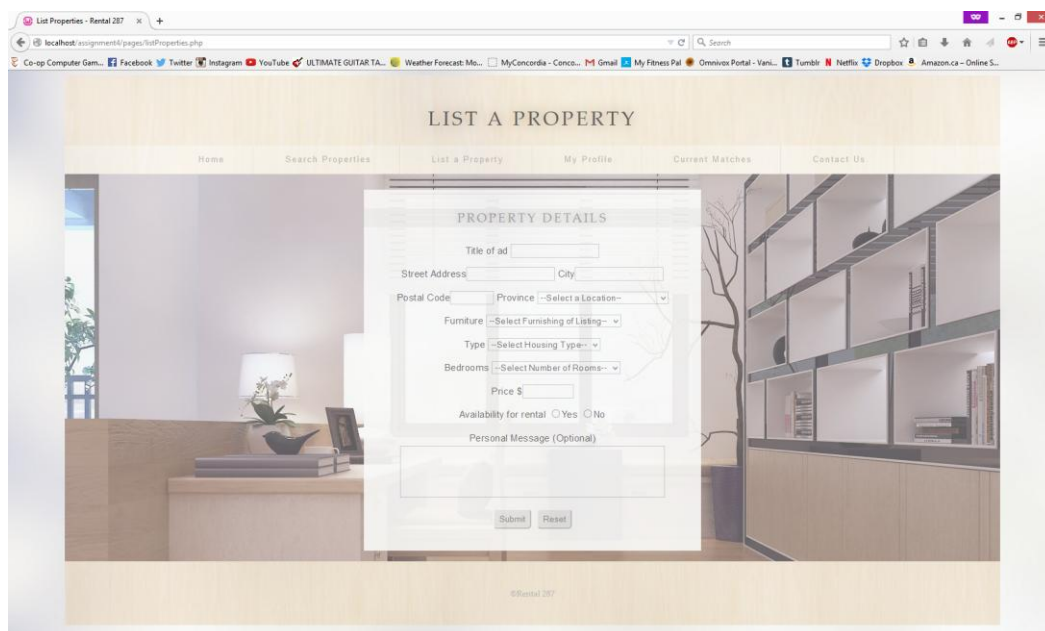
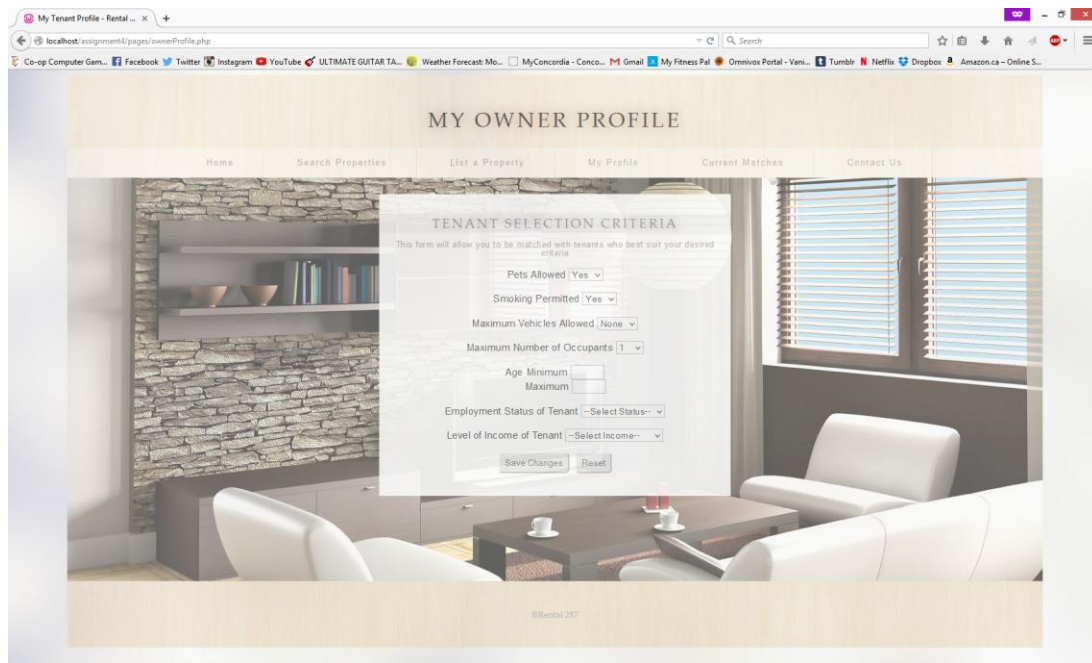
This alert box tells us that the account is successfully created. After clicking 'ok', it redirects you to the index.php page so that you can login with your new credentials. All credentials are stored in the database table called 'users'.



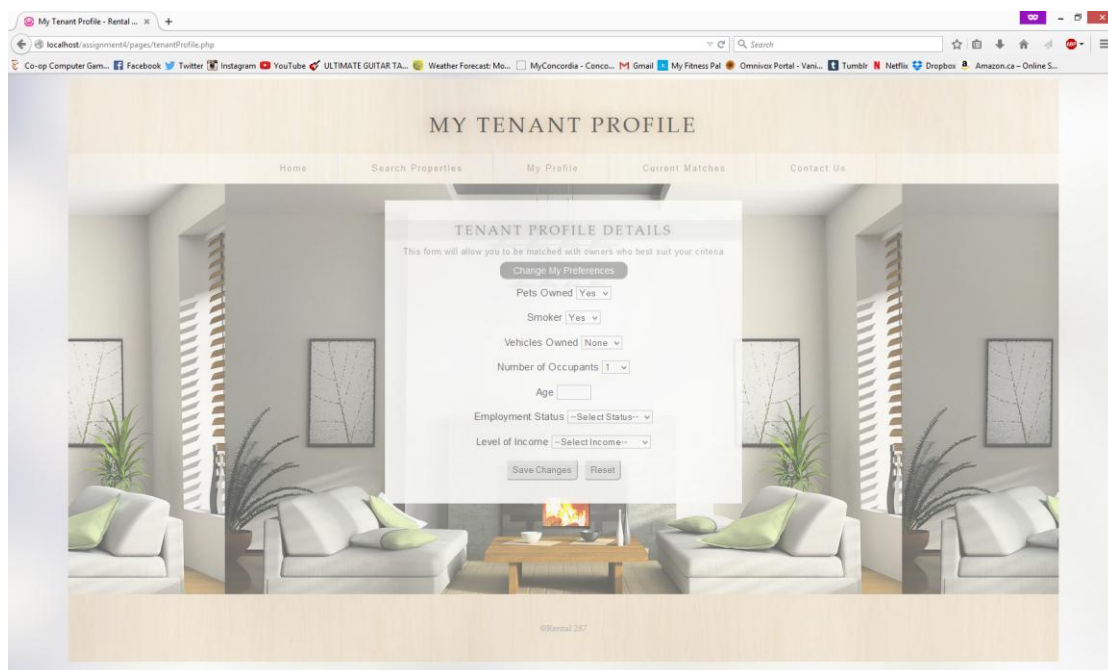
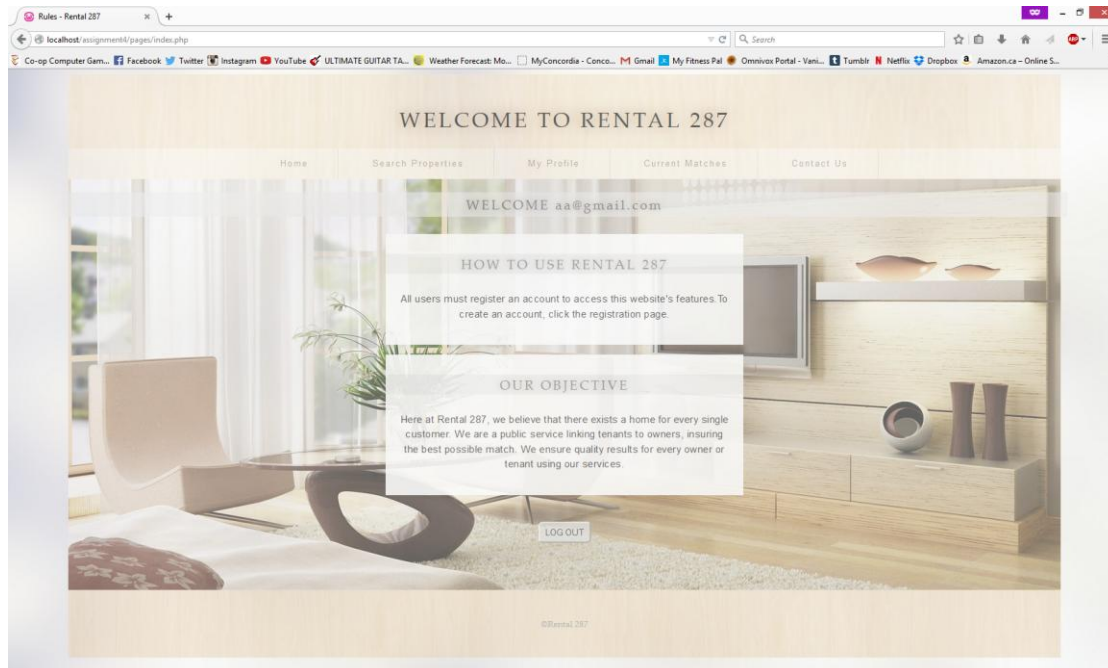
At the index page, submitting the form for user-email and password activates the login.php file inside the 'includes' folder. This file is mainly php code therefore it is not used as a page. The login.php file starts a session for the user, checks if a field is left blank, if the login was successful and if it wasn't due to a wrong e-mail and/or password. It displays the errors through alert boxes. After any of the three results, the user is redirected to index.php.

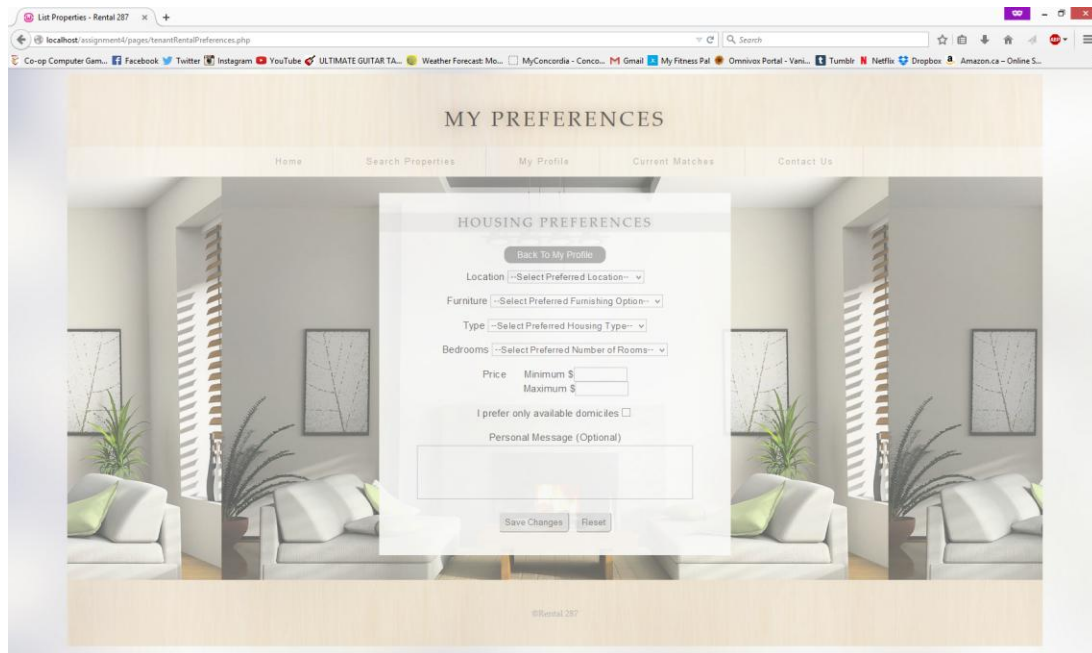


If the login was successful, when redirected to the index.php page, it displays the user's e-mail at the top. You can see that the navigation bar has now appeared. Inside the menu.php file located in the 'includes' folder, there is code dictating which navigation menu should appear based on the user type of the user in current session. The current user above is an owner type profile, which allows her to see the ownerMenu.php as the navigation menu. It includes listProperties.php page and ownerProfile.php. When clicking 'My Profile', it redirects the user to ownerProfile.php. Both files are in the 'page' folder because they are pages.



In the case below, the user is a tenant. You can see that the 'List a Property' tab has now disappeared because the menu is now the tenantMenu.php and after clicking 'My Profile', the user will be redirected to tenantProfile.php. On this page, he can access tenantPreferences.php by clicking the button "Change my Preferences". Both of these files are in 'pages' folder because they are pages.





For all four forms located inside `ownerProfile.php`, `listProperties.php`, `tenantProfile.php` and `tenantPreferences.php`, creating and submitting a valid entry adds an entry to the corresponding tables on the database (`owner_preferences`, `listings`, `tenant_details` and `tenant_preferences`).

The two last tables are dedicated for the matching algorithm. I did not have much time to implement this because I was not time efficient and succeeded solely in building the algorithm. I did not manage to put the users' values in arrays through database, but manually by hard-coding the arrays. In the page below called `matches.php`, this shows all the users names in the database and their types, and also their ID's. Then, it displays how they ranked each tenant if they are an owner or vice versa, inside an array of integers. All the tenants are grouped in one array, and each tenant has an array of integers to rank the owners. Same goes for the owners, but with an array of integers ranking the tenants.

The page describes step by step how the matching algorithm works, and displays the results at the end. The user can use his matching Tenant or Owner ID with his match through the list of names with corresponding ID's displayed.

I am sorry for my formatting as I did not have much time to fix up the look of this page because I hadn't made it beforehand.

The `matches.php` page can be accessed by both owners and tenants. The `contact.php` and `searchProperties.php` pages should in theory also be accessed by both user types, allowing either to send a message or to look at properties, regardless of their preferences set. Their functionalities have not been implemented in this project as they serve no purpose for now.

Home - Rental 287

localhost/assignment4/pages/matches.php

Co-op Computer Gam... Facebook Twitter Instagram YouTube ULTIMATE GUITAR TA... Weather Forecast: Mo... MyConcordia - Conco... Gmail My Fitness Pal Omnivox Portal - Vani... Tumblr Netflix Dropbox Amazon.ca - Online S...

MATCHING RESULTS

Home Search Properties My Profile Current Matches Contact Us

ALL TENANTS

User ID: 1
Name: Ali Anderson

User ID: 2
Name: Bob Barkley

User ID: 3
Name: Carl Clooney

User ID: 4
Name: Dom Darling

User ID: 5
Name: Eric Eary

User ID: 1
Tenant ID: 1
Given Ranking of Owners: 3,2,5,1,4

User ID: 2
Tenant ID: 2
Given Ranking of Owners: 1,2,5,3,4

User ID: 3
Tenant ID: 3
Given Ranking of Owners: 4,3,2,1,5

User ID: 4
Tenant ID: 4
Given Ranking of Owners: 1,3,4,2,5

User ID: 5
Tenant ID: 5
Given Ranking of Owners: 1,2,4,5,3

Home - Rental 287

localhost/assignment4/pages/matches.php

Co-op Computer Gam... Facebook Twitter Instagram YouTube ULTIMATE GUITAR TA... Weather Forecast: Mo... MyConcordia - Conco... Gmail My Fitness Pal Omnivox Portal - Vani... Tumblr Netflix Dropbox Amazon.ca - Online S...

ALL OWNERS

User ID: 6
Name: Fred Fiddle

User ID: 7
Name: George Gillian

User ID: 8
Name: Hyde Hellner

User ID: 9
Name: Ilana Iancu

User ID: 10
Name: Jillian Jonas

User ID: 11
Name: Jhay Arevalo

User ID: 6
Tenant ID: 1
Given Ranking of Owners: 3,5,2,1,4

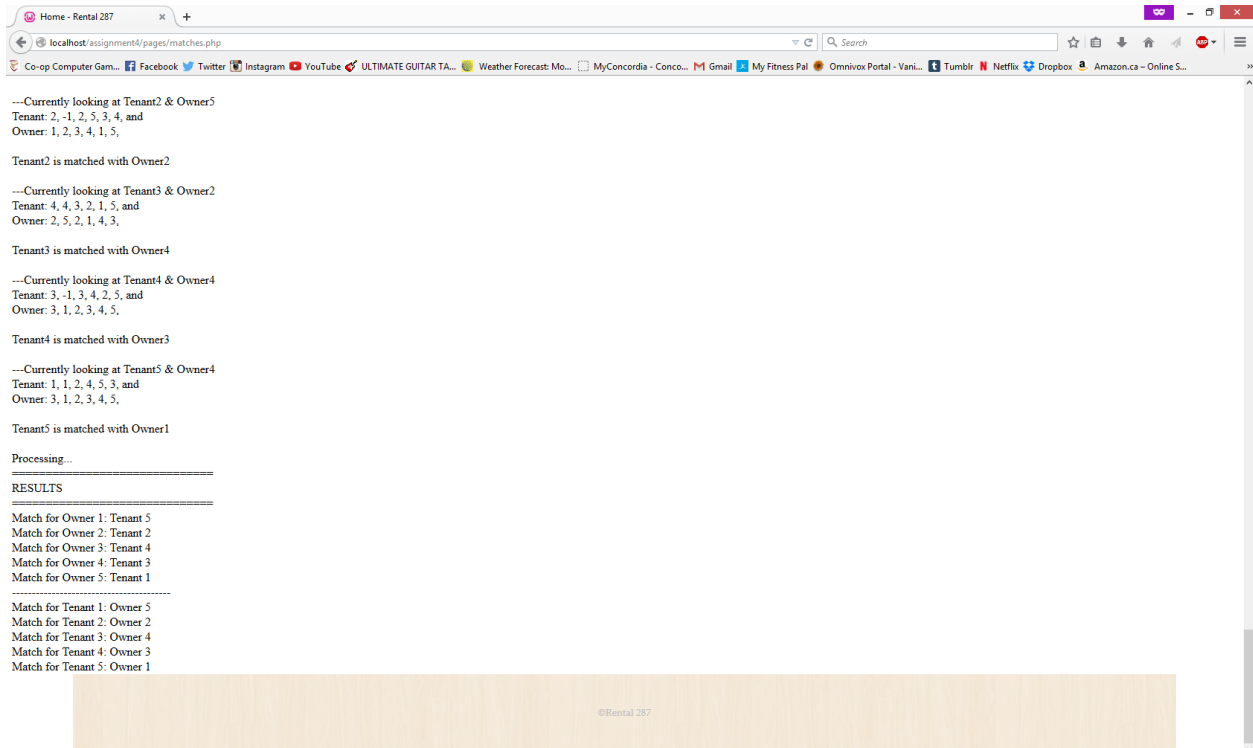
User ID: 7
Tenant ID: 2
Given Ranking of Owners: 5,2,1,4,3

User ID: 8
Tenant ID: 3
Given Ranking of Owners: 4,3,5,1,2

User ID: 9
Tenant ID: 4
Given Ranking of Owners: 1,2,3,4,5

User ID: 10
Tenant ID: 5
Given Ranking of Owners: 2,3,4,1,5

MATCHING



The screenshot shows a web browser window with the address bar displaying 'localhost/assignment04/pages/matches.php'. The page content is as follows:

```
--Currently looking at Tenant2 & Owner5
Tenant: 2, -1, 2, 5, 3, 4, and
Owner: 1, 2, 3, 4, 1, 5,

Tenant2 is matched with Owner2

--Currently looking at Tenant3 & Owner2
Tenant: 4, 4, 3, 2, 1, 5, and
Owner: 2, 5, 2, 1, 4, 3,

Tenant3 is matched with Owner4

--Currently looking at Tenant4 & Owner4
Tenant: 3, -1, 3, 4, 2, 5, and
Owner: 3, 1, 2, 3, 4, 5,

Tenant4 is matched with Owner3

--Currently looking at Tenant5 & Owner4
Tenant: 1, 1, 2, 4, 5, 3, and
Owner: 3, 1, 2, 3, 4, 5,

Tenant5 is matched with Owner1

Processing...
=====
RESULTS
=====
Match for Owner 1: Tenant 5
Match for Owner 2: Tenant 2
Match for Owner 3: Tenant 4
Match for Owner 4: Tenant 3
Match for Owner 5: Tenant 1
=====
Match for Tenant 1: Owner 5
Match for Tenant 2: Owner 2
Match for Tenant 3: Owner 4
Match for Tenant 4: Owner 3
Match for Tenant 5: Owner 1
```

At the bottom of the page, there is a light blue footer bar with the text '©Rental 287'.

How the algorithm works:

There are two 2D arrays which include the ranking stored inside tables `owner_ranks_tenants` and `tenant_ranks_owners`.

Inside `owner_ranks_tenants` table, there are 5 entries with different owner ID's and different rank orders for the tenants. For one single owner, the first index (index 0) of the array stores the owner's current match. It is initialized at 0 because the owner starts off with no match. Then, the next 5 indexes correspond to the ranking of the tenants from highest to lowest. This in theory should be taken from `ranking_tenants` parameter inside the `owner_ranks_tenants` table.

Then, all 5 owners are stored in an array called `$owner`. This array consists of 5 arrays of size 6 → the match and ranking of tenants for each owner.

This is all done the same way for the tenants, replacing `owner_ranks_tenants` table with `tenant_ranks_owners` and storing all the values in an array called `$tenant`.

A Boolean variable `someoneWasRejected` is made to determine whether or not to end the program. The program runs until `someoneWasRejected` is false i.e. while there is still a tenant with no match. In this case, I have made the tenant the "male figure". This implies that the tenant "proposes" his match onto the desired owner.

Since the program will end when all tenants have a match, we check if all tenants are matched first. If they all are, then the program is terminated and everyone is matched to someone.

The first round of proposals, the tenant is most likely to be accepted by the owner because the owner does not currently have a match, unless 2 tenants ask the same owner. Then, the owner will choose the tenant with the highest rank, and reject the other one, making the Boolean `someoneWasRejected` true.

In other words, if the tenant is the only one to ask an owner with no current match, he automatically gets a yes from the owner. If not, then the one with the highest rank for the owner is chosen. To get this “rank”, I have made a function called `getRank` which takes in 2 parameters: the first parameter is the ID to be ranked (an integer) and the 2nd is an array of the ranking. Inside this function, an integer is returned which corresponds to the index of the first parameter within the array of the ranking.

If a tenant’s index within the owner’s rank array is smaller than another’s, then this means that he is ranked higher. The first tenant is then accepted and the second rejected, which makes `someoneWasRejected` true.

If a tenant after all these cases was still rejected, it turns the Boolean `someoneWasRejected` true which repeats the whole process.

This whole function is in a do-while loop which stays within the loop if `someoneWasRejected` = true. At the top of the loop, `someoneWasRejected` is always set to false because we are starting a new evaluation. If someone is rejected along the way, then it is turned to true allowing the function to loop properly.

At the end, 2 for loops are used to display the results of the matching for both owners and tenants, iterating through both arrays and showing the current match of all tenants and owners (the integer situated at index 0 for each array). These numbers correspond to the tenant and owner ID’s listed inside the `owner_ranks_tenants` and `tenant_ranks_owners` tables under `orank_id` and `trank_id`.

Listed previously, the user can link this id to a person’s first name and last name.