Software Engineering CSC 648/848 Section 01 Fall 2017
Department of Computer Science
SAN FRANCISCO STATE UNIVERSITY

Milestone 4

Gator Realtor

Team 11 - Local

Alexander Tung : alextung94@gmail.com

Sukhjit Singh

Ilya Nemtsov

Nicholas Szeto

Ralph Acosta

Yangshan Huang

Date	Version	Description
12/09/2017	1.0	First draft submitted for review

1) Product Summary

- All visitors to the website will be able to search for home listings using city or zip code.
- The search results can be filtered to suit the customer's needs.
- All users can then find contact information for any listing by look at the listing's details.
- Any customer can register for a buyer's account.
- By registering for an account, a customer is able to contact a seller through a listing's page.
- A history of this conversation can be found in the user's dashboard.
- Sellers can register for an account too if they wish to sell on this website.
- By registering for an account, an agent is able to create listings and later edit or remove them when desired.
- A dashboard for sellers is also present where sellers manage their listings and messages with potential buyers.
- Listings will show necessary information such as address, contact information, and at least one image. These are required by the seller when the listing is created.

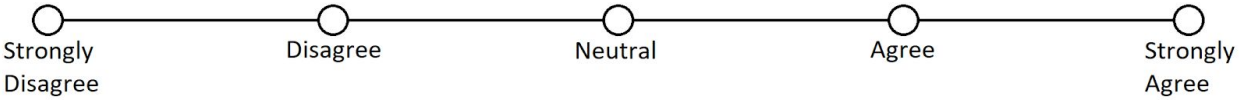
2) Utility Test Plan

Feature to be tested: Search

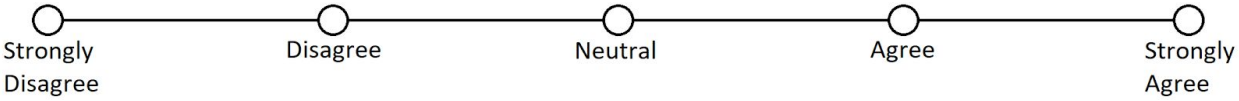
Test Objectives
All users should be able to search and get desired results.
When given a proper query with results, display results.
Results page should display correct results.
Results page should display number of listings found.
When given a empty query, display all.
When given a proper query that has no results, display no results along with suggestions.
Results page should persist search query
When given an improper query, show corresponding error.
Search should work by pressing enter or by clicking on the magnifying glass button.

Test Plan
Starting from https://www.sfsuse.com/fa17g11/ the search bar should be present centered of the user's screen. Type into the search bar '94132' for a proper query. The user should be brought to https://www.sfsuse.com/fa17g11/search with results displayed. Check to see that the number of found listings and the number of listings displayed match up. Check to see that the query was preserved. Search for 'san' as another proper query and recheck previously mentioned features. Search for whitespace and check to see that it displays all results. Search for 'foo' to return 0 results. Search for improper queries to check for errors. Improper queries any string of digits that isn't 5 characters in length.

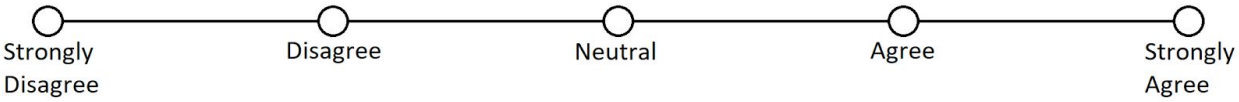
1. Starting the search process was intuitive.



2. The search results were displayed in a commonsensical manner.



3. The search errors guided to a correct search query.



3) QA Test Plan

Feature to be tested: Search

Test Objectives
When given a proper query with results, display results.
Results page should display correct results.
Results page should display number of listings found.
When given a empty query, display all.
When given a proper query that has no results, display no results.
Results page should persist search query
When given an improper query, show corresponding error.
Search should work by pressing enter or by clicking on the magnifying glass button

Hardware required: Two computers. One running MacOS and the other running Windows. Linux is optional.
Software required: Safari on MacOS device, Internet Explorer/Edge on Windows, Chrome on both devices, Firefox on both devices.

Test Cases

#	Name	Description	Input	Expected Output	Pass/Fail
1	Proper Query	A proper query	94132	One search result: 265 Winston Drive	
2	Improper Query	A query that should display an error due to a user's mistake such as a typo.	1234	Error message requesting 5 digit zip code	
3	Security Check	A possible input from malicious users that requires sanitation in the backend	Select * from hello;	Error message, same page	

4) Code Review

Coding Style

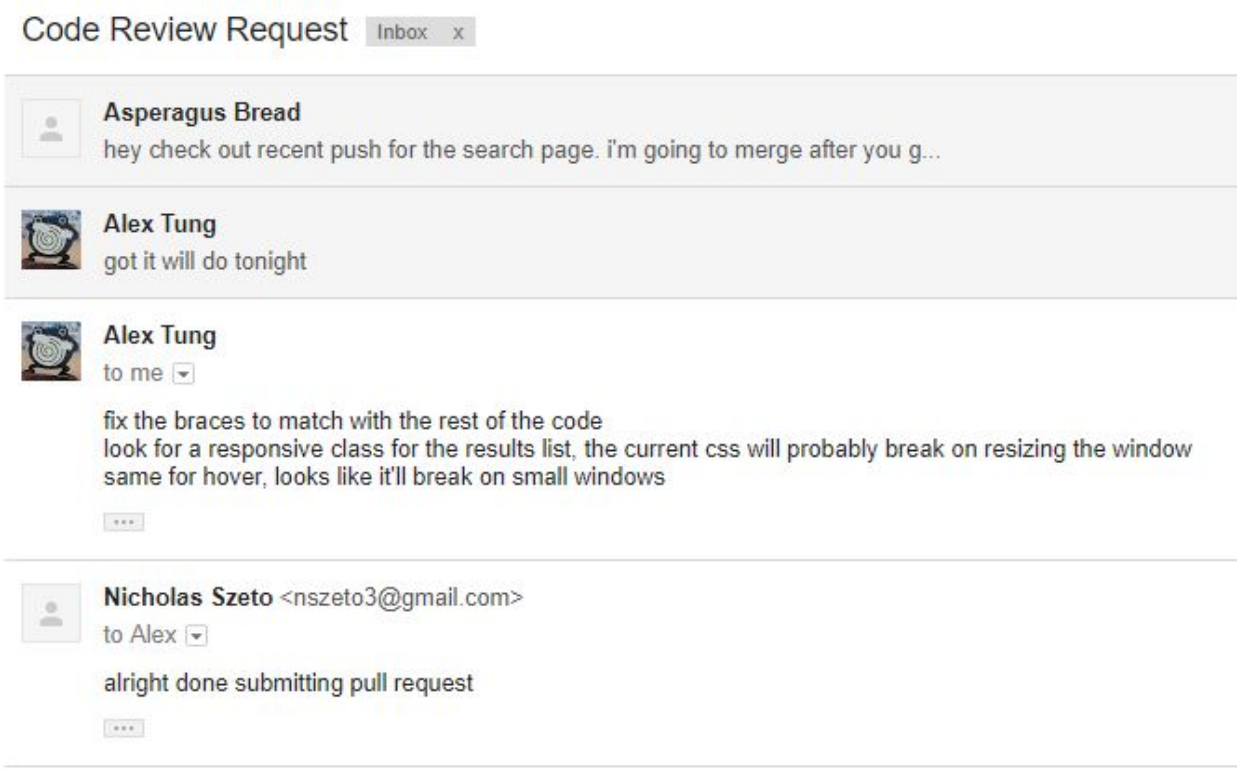
For our coding style we use an indent style where the opening brace of the compound statement is on the same line as the first line of code and the closing brace is flush with the left at the bottom on its own new line. This is sometimes known as K&R. We use white space to enhance readability even when it is not required for functionality. Other than that we follow standard coding conventions such as intuitive naming, line length limiting, and commenting conventions.

```
<div class="container">
    <div class="row">
        <!--Search Bar Start-->
        <div style="padding: 40px; " class="col-md-12">
            <form action="/search" method="POST">
                <div id="custom-search-input">
                    <div class="input-group col-md-12">
                        <input type="text" class="form-control input-lg" placeholder="Search by city or zip"
                               name="searchQuery"/>
                        <span class="input-group-btn">
                        <button class="btn btn-info btn-lg" type="submit" formaction="/search">
                            <i class="fa fa-search"></i>
                        </button>
                    </span>
                    </div>
                </div>
            </form>
        </div>
    </div>
    <!--Search Bar End-->
</div>
<style>
    .img-thumbnail:hover {
        box-shadow: 0px 0px 150px #000000 ;
        z-index: 2;
        -webkit-transition: all 200ms ease-in;
        -webkit-transform: scale(1.5);
        -ms-transition: all 200ms ease-in;
        -ms-transform: scale(1.5);
        -moz-transition: all 200ms ease-in;
        -moz-transform: scale(1.5);
        transition: all 200ms ease-in;
        transform: scale(1.5);
    }
</style>


<!--Results Listings Start-->
<div class="container">

    <!--First Left and Right Card End-->
```

```
    <div class="d-flex flex-row align-items-stretch flex-wrap">
      {{#each results}}
        <div class="col-sm-4">
          <div class="card" style="height:100%">
            <!--Card Image-->
            <img class="img-thumbnail" src="{{imageLink}}" alt="No Image">
            <!--Card Body-->
            <div class="card-body">
              <h4>Property: {{propertyId}}</h4>
              <h6>{{streetAddress}}</h6>
              <h6>{{city}}, {{state}} {{zipcode}}</h6>
              <h6>Built Year: {{buildYear}}</h6>
              <h6>Bedrooms: {{bedrooms}} Bathrooms: {{bathrooms}}</h6>
              <!--<p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>-->
              <a href="#" class="card-link">Card link</a>
              <a href="#" class="card-link">Another link</a>
            </div>
          </div>
        </div>
      {{/each}}
    </div>
  </div>
  <!--Results Listings End-->
```



5) Self-check on best practices for security

We are encrypting passwords in the database.
We are validating inputs on the search bar using a javascript library.
Example:

```
if(!validator.isNumeric(query)) {
    searchByCity(query)
}
else {
    if(!validator.isPostalCode(query, 'US')) {
        return response.send('Invalid Zip Code')
    }
    else {
        searchByZipcode(query)
    }
}
```

6) Self-check: Adherence to original non-functional specs

1. Application shall be developed and deployed using class provided deployment stack - *DONE*
2. Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks must be explicitly approved by Anthony Souza on a case by case basis. - *DONE*
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class - *DONE, on Google now*
4. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. -*ON TRACK*
5. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed - *DONE*
6. Data shall be stored in the MySQL database on the class server in the team's account - *DONE*
7. Application shall provide real-estate images and optionally video - *DONE*
8. Maps showing real-estate location shall be required - *DONE*
9. Application shall be deployed from the team's account on AWS - *DONE, on Google now*
10. No more than 50 concurrent users shall be accessing the application at any time - *DONE*
11. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. -*ON TRACK*
12. The language used shall be English. - *DONE*
13. Application shall be very easy to use and intuitive. No prior training shall be required to use the website. - *DONE*
14. Google analytics shall be added - *ON TRACK*
15. Messaging between users shall be done only by class approved methods and not via e-mail clients in order to avoid issues of security with e-mail services. - *DONE*
16. Pay functionality (how to pay for goods and services) shall not be implemented. - *DONE*
17. Site security: basic best practices shall be applied (as covered in the class) - *DONE*
18. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development - *DONE*
19. The website shall prominently display the following text on all pages "*SFSU Software Engineering Project, Fall 2017. For Demonstration Only"*. (Important so as to not confuse this with a real application). - *DONE*