SW Engineering CSC648/848 Section 01 Fall 2017
FriscoHousing
Team 14

John Lazzarini, Team Lead, jlazzari@mail.sfsu.edu
Jonah Manarang
Ivan Villamar
Eric Gumba
Prakash Gurung

Milestone 4
12/7/2017

Date submitted for review	12/7/2017
Date revised after feedback and frozen	12/17/2017

Product Summary

Product name: FriscoHousing

Commited functions:

- Users can search for a listing by city or zip code, from any page, regardless of registration status.
- The search function will validate input based on search type -- zip code searches will return an error to invalidate alphanumerical input, and city searches will return an error to invalidate numerical input.
- Unregistered users can browse the site freely, except where certain pages (agent dashboard and associated pages, agent contact) are restricted to account access.
- Users may register as realtors, who shall have access to a dashboard page from which they can review or delete their posted listings, review their messages, or post new listings.
- Users may register as buyers, who shall have the ability to contact an agent from one of the agent's listing pages to express interest in that listing.

What makes our platform special:

FriscoHousing manages to deliver the essential features of a real estate browsing website, despite having been developed by an initially inexperienced skeleton crew of five. Scheduling issues meant that we could only meet once per week outside of slack, and yet the product works as expected for all intents and purposes after just one semester.

URL: https://sfsuse.com/fa17g14/

QA test plan

Test Objectives: The goal of the QA test plan is to test one of the major functions of our product which is the ability to search for housing. Along with the sell dashboard, this is one of the primary features of the website indicated by its persistence throughout all of the pages of the website.

HW and SW setup:
Hardware: MacBook Pro 2016
Software: Mozilla Firefox and Google Chrome.

Feature to be tested:
The feature we're testing is the search bar.

Task 1:

1. Open Google Chrome.
2. Under address bar, type: https://sfsuse.com/fa17g14/
3. Click on drop-down menu to the left of the search bar. Click on zip code or city.
4. Press "Search" to the right of search bar.
5. Verify that five results are returned.
6. Open Mozilla Firefox and repeat steps 2-5.

Task 2:

1. Open Google Chrome.
2. Under address bar, type: https://sfsuse.com/fa17g14/
3. Click on drop-down menu to the left of the search bar. Click on city.
4. Enter "san francisco"
5. Press "Search" to the right of search bar.
6. Verify that 1 result(s) are returned all related to "San Francisco".
7. Open Mozilla Firefox and repeat steps 2-6.

Task 3:

1. Open Google Chrome.
2. Under address bar, type: https://sfsuse.com/fa17g14/
3. Click on drop-down menu to the left of the search bar. Enter zipcode.

4. Press “Search” to the right of search bar.
5. Verify that 2 result(s) are returned with the first two digits under the zipcode column starting with “92”.
6. Open Mozilla Firefox and repeat steps 2-5.

Google Chrome / Firefox

Number	Description	Test Input	Expected Output	PASS/FAIL
1	Test to see if inputting nothing will output entire list of database.	nothing	5 items	PASS
2	Test to see if inputting San Francisco will output 1 item.	San Francisco	1 items with San Francisco under “city” column.	PASS
3	Test to see if inputting 95 will output only two items.	95	2 items with the first two digits being 95 under “zipcode” column	PASS

Usability Test_Plan

Test Objectives:
For our usability test, we are testing the the feature of ease of use and accuracy. We select the “Search” feature to analyze our test plan.

Test Plan:

System Setup:
User will need access to the internet and a web browser. The web browser has to be running the latest version installed. The web browser supported: Chrome, Safari, and Firefox. User need to type our URL to access our website and navigate to the search box.

Starting Point: https://sfsuse.com/fa17g14/

Tasks:
1. From houses in San Francisco
2. Find houses in San Jose
3. Find houses using the zip code 94131

Intended Users:
The intended users for the website are SFSU students who are looking to buy and sell real estate to other SFSU students.

Completion Criteria:
Users perform a search using a valid keyword with an appropriate selected dropdown menu.

Questionnaire Form:
The following questions will determine the usability of the search function.

1. It was easy to locate search function on the website from any page.

○ Strongly Agree

○ Agree

○ Neutral

◯ Disagree

◯ Strongly Disagree

2. It was easy to use search function and find the results to my satisfaction.

◯ Strongly Agree

◯ Agree

◯ Neutral

◯ Disagree

◯ Strongly Disagree

3. It was easy to narrow down the search results by using the category filter.

◯ Strongly Agree

◯ Agree

◯ Neutral

◯ Disagree

◯ Strongly Disagree

Comments: (Optional)

Code Review

-variable names are straightforward and easy to understand what is being stored in them
- just by reading the code you can easily tell that the var pattern is used for some sort of validation and that maxlength is for checking the length
-function does not seem to have a name, I don't know if it needs one or not

```
 $(fuction() {

          $('#searchBy').change(function() {

               var optionValue = $('option:selected', this).attr('value');



               var placeholder = (optionValue === 'City' ? 'Enter a City' : optionValue ===
         'ZipCode' ? 'Enter a ZIP code' : 'Enter a city or ZIP code');
               $('#textsearch').attr('placeholder', placeholder);



               var maxlength = (optionValue === 'City' ? '20' : optionValue === 'ZipCode' ?
         '5' : '20');
               $('#textsearch').attr('maxlength', maxlength);



               var pattern = (optionValue === 'City' ? '[A-z \s]{0,}' : optionValue ===
         'ZipCode' ? '[0-9 \d]{0,}' : '[A-z0-9 \s]{0,}');
               $('#textsearch').attr('pattern', pattern);



               var title = (optionValue === 'City' ? 'Use letters from A-z' : optionValue ===
         'ZipCode' ? 'Use digits from 0-9' : 'Use letters from A-z or digits from 0-9');
               $('#textsearch').attr('title', title);

          });

         });
```

};

-unlike the previous function reviewed this function seems to have a name
-the name of the function is straightforward and I immediately have some sort of idea what it does
-camelcase is consistently used in both of the functions
-variable names again are helpful and straightforward as to what they contain

```
function
setPicturePriceAndAd
dress() {

var housePhoto = localStorage.getItem("housePhoto");

var houseAddress = localStorage.getItem("address");

var housePrice = localStorage.getItem("price");

let strHTML = "";

strHTML +=
"<img class=\"card-img-top\" src=" + housePhoto + " alt=\"Card image cap\"> "
+
"<div class=\"card-block\">" +
"<h4 class=\"card-title\">" +
"</i>" + houseAddress + "</h4>" +
"<p class=\"card-text\">" + housePrice + "</p>" +
"</div>";

$("#housePhotoBuy").html(strHTML);

};
```

Code Review Correspondence:

12:02 PM

Jonah Nickolas Manarang

to John

```
$(functio
n() {
        $('#searchBy').change(function() {
        var optionValue = $('option:selected', this).attr('value');

        var placeholder = (optionValue === 'City' ? 'Enter a City' : optionValue ===
        'ZipCode' ? 'Enter a ZIP code' : 'Enter a city or ZIP code');
        $('#textsearch').attr('placeholder', placeholder);

        var maxlength = (optionValue === 'City' ? '20' : optionValue === 'ZipCode' ? '5' :
        '20');
        $('#textsearch').attr('maxlength', maxlength);

        var pattern = (optionValue === 'City' ? '[A-z \s]{0,}' : optionValue === 'ZipCode' ?
        '[0-9 \d]{0,}': '[A-z0-9 \s]{0,}');
        $('#textsearch').attr('pattern', pattern);

        var title = (optionValue === 'City' ? 'Use letters from A-z' : optionValue ===
        'ZipCode' ? 'Use digits from 0-9' : 'Use letters from A-z or digits from 0-9');
        $('#textsearch').attr('title', title);
        });
        });
```

```
functionsetPicturePriceAndAdd
ress() {

        var housePhoto = localStorage.getItem("housePhoto");

        var houseAddress = localStorage.getItem("address");

        var housePrice = localStorage.getItem("price");

        let strHTML = "";

        strHTML +=
        "<img class=\"card-img-top\" src=" + housePhoto + "
        alt=\"Card image cap\"> " +
        "<div class=\"card-block\">" +
        "<h4 class=\"card-title\">" +
        "</i>" + houseAddress + "</h4>" +
        "<p class=\"card-text\">" + housePrice + "</p>" +
        "</div>";

        $("#housePhotoBuy").html(strHTML);

        };
```

12:25 PM

John Lazzarini
<j.lazzarini.work@gmail.com>

to Jonah

Jonah,

Thanks for the submission -- I like that you inlined the conditionals, it helps keep the code concise.

Variable and function names are in camel case and your naming conventions clearly indicate their usages, so I'm going to go ahead and approve this submission.

Best,

John

Self Check Best Security Practices

Database fields that store confidential or sensitive data such as passwords, social security numbers, etc. must be stored using encryption techniques. In our web application, we used the MySQL functions AES_ENCRYPT() and AES_DECRYPT() to encrypt and decrypt user passwords. They implement encryption and decryption of data using the official AES (Advanced Encryption Standard) symmetric-key algorithm, previously known as “Rijndael,” which uses the same key for both encrypting and decrypting the data.

Additionally, our Node web application is protected from “SQL Injection Attacks” where the attacker enters SQL commands into Web form input fields or URL query strings to try to manipulate the SQL statement being sent to and from the database. It is a common web hacking technique to destroy or misuse your database. The methods used in our application to avoid SQL injection attacks are as follows:

- Use of parameterized queries (escaping query values provided by the user using the ‘?’ placeholder in our query and mysql.escape() method) as opposed to using string concatenation.
- Limiting the amount of characters in Web form input fields with the maxlength attribute.
- Validating text input for improper characters with the pattern attribute using regex.

- No displaying of error messages that contain information about the database or actual source code.

Self Check Non-Functional Specs

<u>High-level non-functional specifications (HOW the app is delivered and other constraints) that MUST be adhered to</u>

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.**
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. **DONE.**
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.**
10. No more than 50 concurrent users shall be accessing the application at any time, **IN PROGRESS.**
11. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users, **DONE**.
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, **IN PROGRESS**.
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.**