SW Engineering CSC 648 / 848 Spring 2019
Dormy Home Rental Services

Team 01(Local)

Zaur Melikov - Team Lead/M2 Editor/Backend Team
Email: zmelikov@mail.sfsu.edu
Ulises Martinez - Frontend Team Lead
Cyrus Riahi - Backend Team Lead
Kevin Reyes - GitHub Master/Frontend Team
Joe Binalinbing - Frontend Team
Siu Chun Kung - Frontend Team
Mahdi Massoodi - Backend Team

Milestone 4

May 7th, 2019

History

May 7th, 2019 (First Submission)

May12th ,2019 (Revised Version)

1. Product Summary

The need for finding a property to live in is a commonality shared among all people. Often finding such a place is plagued with stress sourcing from the difficulty that finding one's ideal home entails. At Dormy, our team looks to disentangle this process by offering a platform for people to find the properties that suit them best.

At Dormy we seek to streamline the home searching process by offering our users helpful features that will significantly reduce the time required to be spent on finding one's ideal place. Through the use of our website, we promise that users shall:

- Be able to log in.
- Be able to register with their name, email, and phone number, then allowing them to accept the website's terms and conditions.
- Be able to access the website's home page.
- Be able to search through the use of a search bar, free search using the zip code of a city, and the use of filters such as an apartment, house, or condo.
- Be able to view search results with thumbnails and images.
- Be able to view an individual listing's details page alongside a map of where it's located.
- Be able to send messages (renter to the landlord).
- Be able to post a new listing upon approval from a Dormy admin.
- Be able to access a dashboard in which they can manage their messages and listings.
- Be able to access an Admin dashboard via special login credentials, which is equipped with added functionality allowing them to approve or block pending listings, block or delete users, and view all listings (Admins Only).

Along with these features, Dormy is also specifically catered to SFSU students, allowing them to see how far each potential property is from

campus. Designed by students, for students, we believe that our product's ability to break down a commonly stressful life situation will fit in perfectly with the already strenuous lifestyle of those attending college. Although our website is tailored to students attending SFSU, anyone in the San Francisco area can take advantage of its features, because with Dormy, living is simplified.

Dormy's website can be accessed at the following URL:
http://52.53.124.134/

2. Usability Test Plan

Usability Test Plan for Search

PART 1

Test Objectives:

The effectiveness of the search functionality shall provide San Francisco State students with the necessary means to filter through listings, so students are able to find their perfect apartment, house, or condo to live in. The search functionality is the most important feature of dormy because it provides students with a reliable and fast way of searching for a place with very little effort. Testing the Search by and obtaining is crucial for the dormy team because it will provide the team with feedback on whether the users are using such tools efficiently, or whether is there anything can be improved to facilitate the user experience and satisfaction. The search input will be tested for functionality. By functionality, we infer that the user will be able to search items, in this case, apartments close to campus by using the following: Category (Dropdown), Search label by zip code and city, or simply using the search bar to look for a specific term in the listing title. First

of all, Category (Dropdown) will be composed of 4 items, which are: All, Apartments, Houses, Condos. These items will filter listings based on listing-type. Second, the search bar will filter the listing by zip code, title, city based on the user input.

Test Description:

System setup:

San Francisco State students will be exposed to our Search bar usability test. Every member of the team is expected to reach out to two SF students to conduct such test that is about two minutes long. The test should be done with a group member present, the reason been is that such member will take note on how many clicks will the student take to reach a goal given. The team member is also responsible to provide a satisfaction test once such task has been completed, but if such task is not completed over three minutes the team shall take note on what did the user did wrong during his steps.

Starting Point:

The usability testing procedure should start at the home page. The SF State student will not be provided any sort of information on how to do it, but he will be asked to do a certain procedure such as: look for a specific type of listing, to filter a specific price range of apartments, and to do distance search to find the closest apartment to campus. The students will also be asked to think of how he feels during the test too, later on, fill up questionnaires or answer a team member's questions related to his satisfaction using the search bar.

Intended Users:

Every team member is responsible for targeting two SF students to conduct such test and questionnaires. There should be some requirement for such students. For example; students that qualify for such assessment should

be students that commute to school for more than one hour, students that have used a similar service before to look for an apartment such like Zillow or craigslist.

URL: http://52.53.124.134 (Dormy's webpage)

What is to be measured:

San Francisco State students will be tested on comfort and acceptability. So, we will be measuring how the user feels through the process of searching for a specific listing, how does he/she feels with the general search procedure, and also how intuitive the search procedure is. For testing acceptability, we will be asking the user how does the dormy website compares to other websites like Zillow, craigslist, and others. The reason why we will ask how does dormy compares to other websites is to test whether the websites seems simple enough or to the standards of SF State students. This will serve as a measurement for investors to realize that their a viable market of San Francisco State students that could potentially expand to other state campuses.

PART 2

Usability Task Description:

- Describe the task testers do before filling out the Likert questionnaire – a few lines (check class sldies)

TASK	DESCRIPTION
Task	Find apartments which are the closest to campus

Machine state	Dormy home page just loaded
Successful completion criteria	Results came out
Benchmark	Completed in 1 minute

TASK	**DESCRIPTION**
Task	Register an account to post a listing
Machine state	Dormy home page just loaded
Successful completion criteria	Listing is posted
Benchmark	Completed in 3 minute

Questionnaire:

- Provide 3 Lickert scale questions getting user satisfaction after the above task has been performed, in a proper format as it is to be used by reviewer (check class slides) – 3/4 page

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**

The design of GUI is easy to use.	❑	❑	❑	❑	❑
The process of finding houses is intuitive.	❑	❑	❑	❑	❑
Viewing the listing of results is informative.	❑	❑	❑	❑	❑
The process of creating an account is efficient.	❑	❑	❑	❑	❑
The process of posting a listing is easy.	❑	❑	❑	❑	❑

Comments:

3. QA Test Plan

QA Test Plan for Search

PART 1

Test Objectives:

The search feature is one of the most important features on the web application but it is also one that requires a lot of user input and communication between the user and the server. Testing the search feature for Quality Assurance is crucial because it is going to make sure that the user gets a valid and reasonable response when they try to search for something. This portion of the testing will test whether the correct information will be called from the database and displayed for the user. We have to search every edge case to make sure that anything a user inputs gets a valid and useful output. We will test a combination of the settings that can be used to refine the search.

Test Description:

System setup:

One of the dormy members will be testing the system out by inputting very specific input in the search bar and then checking whether the search yields correct results. The tester will input all edge cases and valid input. The input will be predefined and based on how the search actually works. The results are also predetermined because we are checking whether the search feature is working how the team intends for it to work.

Starting Point:

The Quality Assurance testing will start on the dormy homepage where the search is located. The Quality Assurance Tester will have a predetermined set of input that they will use. We will assume that the search bar is fully usable. The tester will first check valid input and make sure the valid output is given by the database, after that there are some edge cases that the tester will check including leaving the search bar blank and typing an invalid input which in this case will be "@###!@$#". The tester will then

check if the results are as expected. The valid input should clearly give us a valid output but the blank search bar should clearly advise the tester that the input is empty; the error test should also clearly tell the tester that the input is invalid.

URL: http://52.53.124.134 (Dormy's webpage)

What is to be measured:

The results will be measured to ensure that the backend database and the front end are working cohesively and correctly with each other. The results will prove that everything is working correctly when saving all the listing and that they are appropriately listed to be used by any other user. The results will show if the search bar can handle any invalid input that a user may put in.

PART 2:

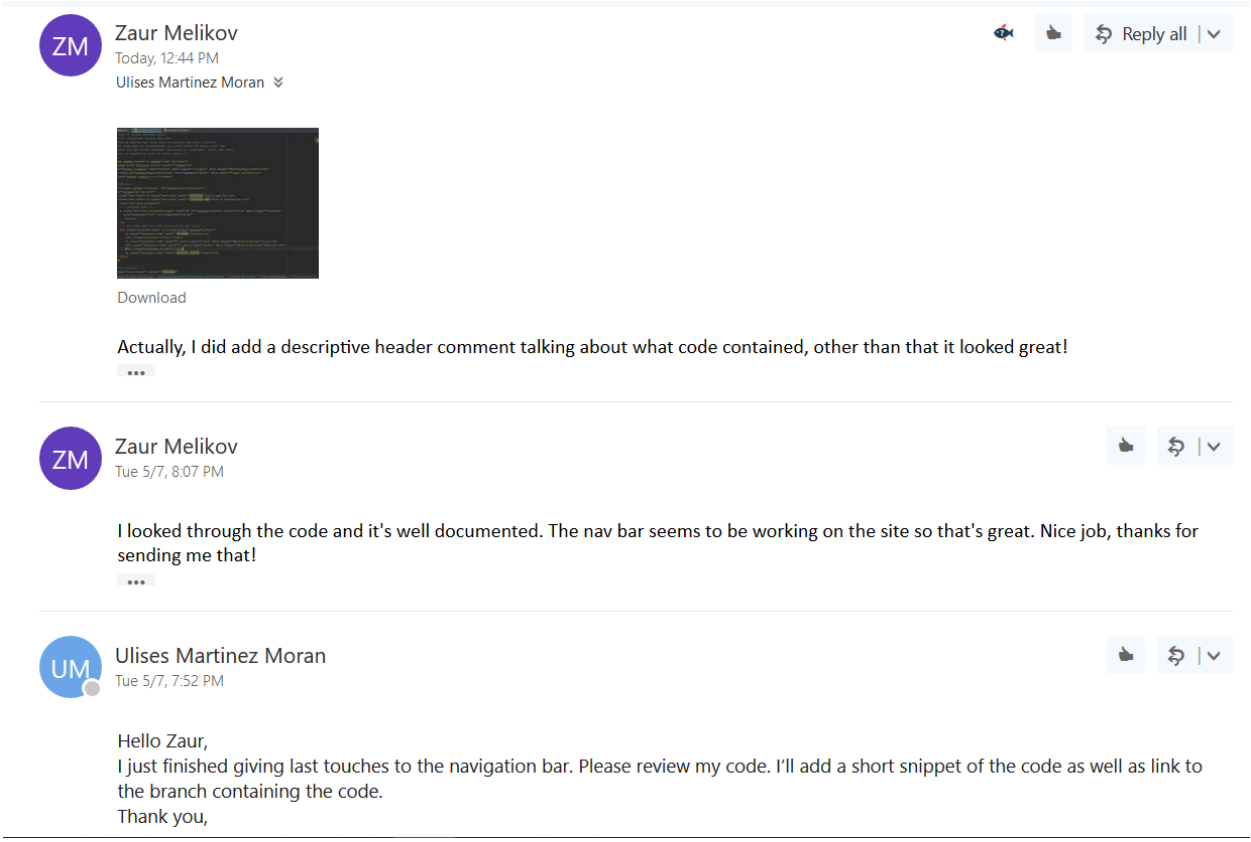
Test #	Test Title	DESCRIPTION	INPUT	EXPECTED OUTPUT	**TEST RESULT**
1	Filter Test	Search function shall return a listing specified on search text and what category is chosen.	Search Text: house Category Option: House	Results shall return a listings of houses. No other category shall be returned.	**PASS**

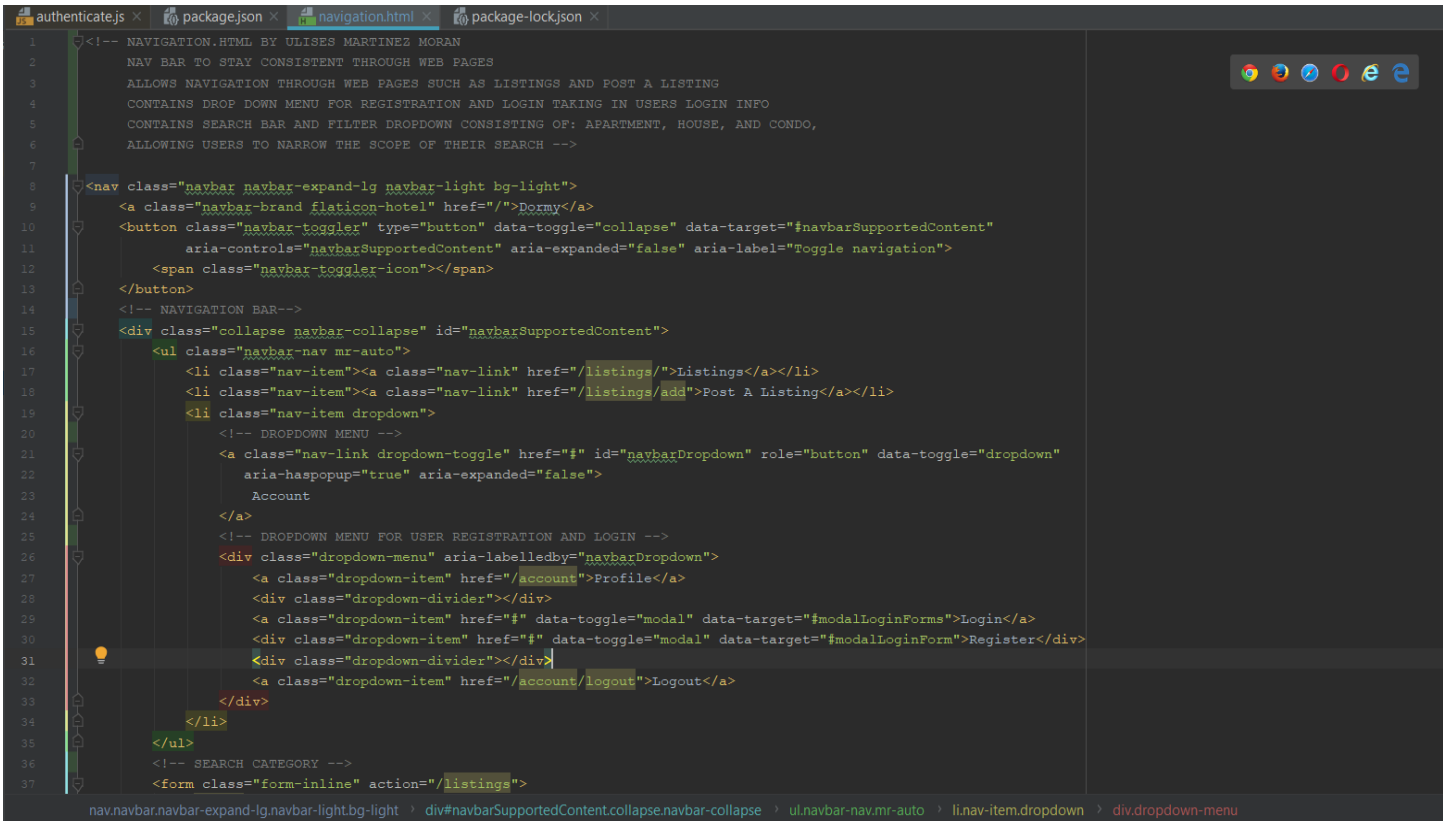
2	Blank Test	Search function shall return all listings based on what category is chosen on the dropdown panel.	Search text: (Blank) Category option: condo	Results shall return a listings of condos. No other category shall be returned.	**PASS**
3	Error Test	Search function shall show an error message when dealing with input that is not recognized by the database	Search Text: @###!@$# Category Option: condo	An error message shall return indicating the inputted text is invalid."You must enter a valid text to continue" shall be shown to remind users to continue the query.	FAIL. No error message is shown. However, a blank page is shown indicating no listing was queried.

4. Code Review

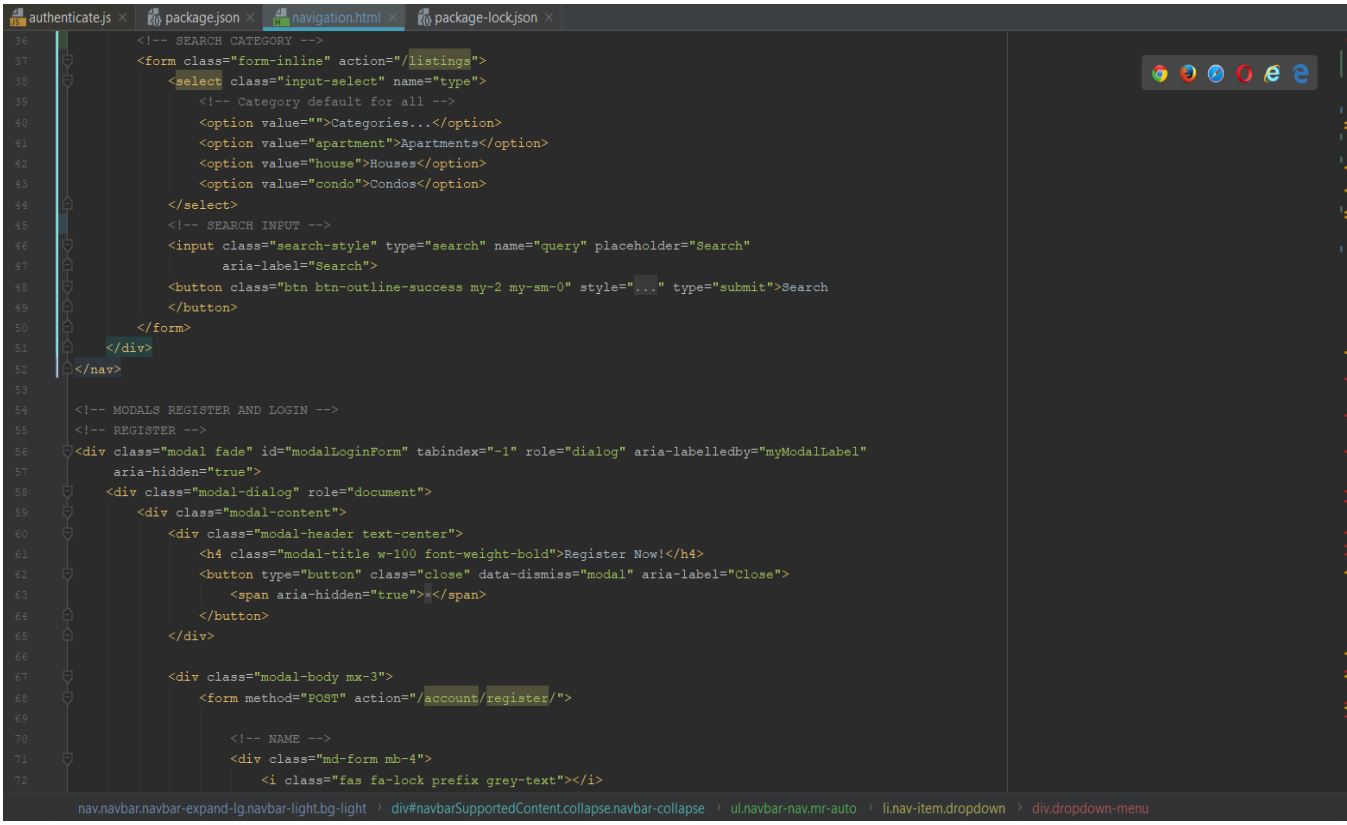
At the beginning of this course our team decided on which languages we would use to construct our website. After a group discussion we decided on using a combination of languages, which consisted mostly of using html, css, and javascript. Before coding began our team agreed upon the strict enforcement of version control through the use of multiple branches within Github. Enforcement consisted of a Github master, and a mutual understanding of the importance of communication between group members when it came

to working on sections of our project and pulling/pushing to and from branches. As far as coding styles are concerned, our biggest point of focus as a team was to emphasize that everyone documented their code through the use of header and inline comments in order to make their code easily navigable. Besides the documentation found within everybody's respective code, we supplemented further details about our code by having a constantly active discord channel which allowed all team members to talk to one another and clear up misunderstandings at a moments notice. The use of Discord turned out to be one of the biggest factors that allowed our team to succeed and work together. Overall teamwork was productive and we avoided any major issues by speaking to one another daily and rallying together to address any minor hiccups throughout the semester.





```
<!-- NAVIGATION.HTML BY ULISES MARTINEZ MORAN
     NAV BAR TO STAY CONSISTENT THROUGH WEB PAGES
     ALLOWS NAVIGATION THROUGH WEB PAGES SUCH AS LISTINGS AND POST A LISTING
     CONTAINS DROP DOWN MENU FOR REGISTRATION AND LOGIN TAKING IN USERS LOGIN INFO
     CONTAINS SEARCH BAR AND FILTER DROPDOWN CONSISTING OF: APARTMENT, HOUSE, AND CONDO,
     ALLOWING USERS TO NARROW THE SCOPE OF THEIR SEARCH -->

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand flaticon-hotel" href="/">Dormy</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <!-- NAVIGATION BAR-->
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item"><a class="nav-link" href="/listings/">Listings</a></li>
            <li class="nav-item"><a class="nav-link" href="/listings/add">Post A Listing</a></li>
            <li class="nav-item dropdown">
                <!-- DROPDOWN MENU -->
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"
                   aria-haspopup="true" aria-expanded="false">
                    Account
                </a>
                <!-- DROPDOWN MENU FOR USER REGISTRATION AND LOGIN -->
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" href="/account">Profile</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="#" data-toggle="modal" data-target="#modalLoginForms">Login</a>
                    <div class="dropdown-item" href="#" data-toggle="modal" data-target="#modalLoginForm">Register</div>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="/account/logout">Logout</a>
                </div>
            </li>
        </ul>
        <!-- SEARCH CATEGORY -->
        <form class="form-inline" action="/listings">
```

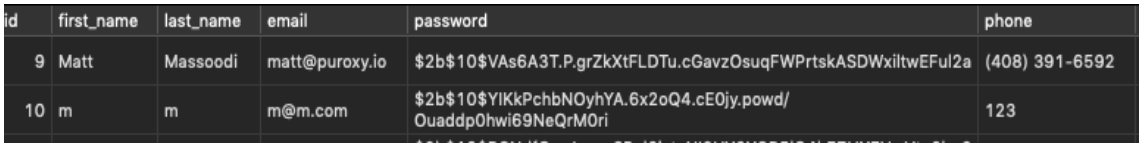


```
            <!-- SEARCH CATEGORY -->
            <form class="form-inline" action="/listings">
                <select class="input-select" name="type">
                    <!-- Category default for all -->
                    <option value="">Categories...</option>
                    <option value="apartment">Apartments</option>
                    <option value="house">Houses</option>
                    <option value="condo">Condos</option>
                </select>
                <!-- SEARCH INPUT -->
                <input class="search-style" type="search" name="query" placeholder="Search"
                       aria-label="Search">
                <button class="btn btn-outline-success my-2 my-sm-0" style="..." type="submit">Search
                </button>
            </form>
        </div>
    </nav>

    <!-- MODALS REGISTER AND LOGIN -->
    <!-- REGISTER -->
    <div class="modal fade" id="modalLoginForm" tabindex="-1" role="dialog" aria-labelledby="myModalLabel"
         aria-hidden="true">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header text-center">
                    <h4 class="modal-title w-100 font-weight-bold">Register Now!</h4>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">×</span>
                    </button>
                </div>

                <div class="modal-body mx-3">
                    <form method="POST" action="/account/register/">

                        <!-- NAME -->
                        <div class="md-form mb-4">
                            <i class="fas fa-lock prefix grey-text"></i>
```

5. Self-check on best practices for security

a. List major assets you are protecting
   i. Hiding landlords address from public word, only showing distance from SFSU campus.
   ii. Messages between students and landlords is kept private, not even admins can view this content.
   iii. We are protecting images by storing them locally on the disk and storing the path name as a string in the database. It's best practice to store files on a storage server and access it using an API.

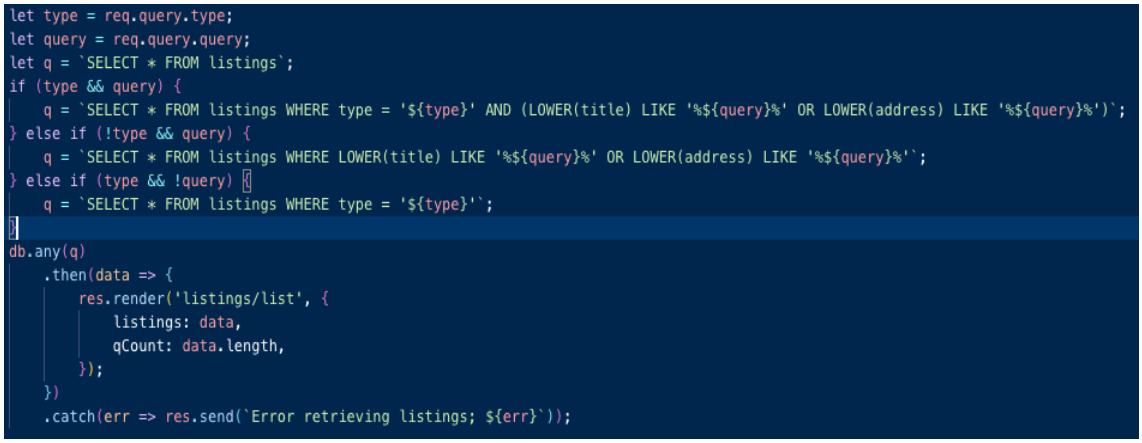
b. Confirm that you encrypt PW in the DB
   i. One-way encrypt user account passwords during user registration using bcrypt.
   ii.



id	first_name	last_name	email	password	phone
9	Matt	Massoodi	matt@puroxy.io	$2b$10$VAs6A3T.P.grZkXtFLDTu.cGavzOsuqFWPrtskASDWxiltwEFul2a	(408) 391-6592
10	m	m	m@m.com	$2b$10$YIKkPchbNOyhYA.6x2oQ4.cE0jy.powd/Ouaddp0hwi69NeQrM0ri	123

c. Confirm Input data validation
   i. Search input validation consists of lowering the query text and doing SQL % LIKE on the title and address columns. This consists of the title of the listing and the address. The search query will **always** return an array of listings based on the type and query string, even if none was specified. In the HTML, we force the user to input a search query of 40 characters or less.

ii.



```
let type = req.query.type;
let query = req.query.query;
let q = `SELECT * FROM listings`;
if (type && query) {
    q = `SELECT * FROM listings WHERE type = '${type}' AND (LOWER(title) LIKE '%${query}%' OR LOWER(address) LIKE '%${query}%')`;
} else if (!type && query) {
    q = `SELECT * FROM listings WHERE LOWER(title) LIKE '%${query}%' OR LOWER(address) LIKE '%${query}%'`;
} else if (type && !query) {
    q = `SELECT * FROM listings WHERE type = '${type}'`;
}
db.any(q)
    .then(data => {
        res.render('listings/list', {
            listings: data,
            qCount: data.length,
        });
    })
    .catch(err => res.send(`Error retrieving listings; ${err}`));
```

6. Self-check: Adherence to original Non-functional specs

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO). **DONE.**

2. Application shall be optimized for standard desktop/laptop browsers. **DONE.**

3. Selected application functions must render well on mobile devices.

   IN PROGRESS.

4. Data shall be stored in the team's chosen database technology on the team's deployment Server. **DONE.**

5. No more than 50 concurrent users shall be accessing the application at any time.

DONE.

6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.

   IN PROGRESS.

7. The language used shall be English.

   DONE.

8. Application shall be very easy to use and intuitive.

   DONE.

9. Google analytics shall be added.

   IN PROGRESS.

10. No email clients shall be allowed.

    IN PROGRESS.

11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated.

    DONE.

12. Site security: basic best practices shall be applied (as covered in the class).

    DONE.

13. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator.

    IN PROGRESS.

14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.

    DONE.

15. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2019. For Demonstration Only" at the top of the WWW page.

    IN PROGRESS.