San Francisco State University

Software Engineering

CSC 648-848 SUMMER 2020

GatorHub

Milestone 04

Team 03

Megha Babariya (mbabariya@mail.sfsu.edu) (Team Lead)
Abraham Zepeda (Github Master)
Nathalia Sainez (Front-end Lead)
Raymond Kuang (Back-end Lead)
Yaxin Deng
Tania Nemeth
Harsh Saxena

History Table:

Version No.	Date	Comments
01	08/03/2020	Initial Document

1. Product Summary :

Our product, GatorHub, is a custom web application built exclusively for SFSU students and faculty to help connect them to the best housing. The following is an itemized list of all major committed functions. The functions described in this list will be delivered in our final project.

1. Search: Search listings based on different parameters such as zip code, property type, and whether a user is looking to rent or buy.
2. Filter: Filter results based on attributes such as distance to SFSU, minimum Price etc.
3. Google Maps API: Showcases distance from the estate to SFSU campus.
4. Registration: Users shall be able to register for an account.
5. Login/Logout: Users shall be able to to login and logout of their account.
6. Compose a Listing: Users shall be able to post a listing.
7. Edit Listing: Users shall be able to edit or remove their listing.
8. Display Listing: When a user clicks a listing, it will direct the user to a full view of that listing.
9. Contact Listing Owner: Ability to contact seller via button on listing.
10. Admin: Admin approval is required when a user wants to add a listing.

2. Usability Test Plan:

2.1 Usability Test Plan for Search bar - Test Objectives

Effectiveness of the search functionality shall provide San Francisco State students with the tools to filter through the listings so that they can find the apartments, condos, homes, or studios that they need and want to live in. The search feature is the most important feature that we offer as it gives a quick result of listings that the students are looking for. Search should be simple and intuitive to use, and the task of searching for a place to call home shall be reliable and fast, with the whole search process requiring very little effort and having high user efficiency.

Searching cuts browsing time significantly by allowing users to get the results that they want and the website also allows search by category of living space which further narrows the scope of a given search. In addition to correct functionality, the search and category fields must be persistent for ease of use. Correctly functioning search and results are vital to the project and must be thoroughly tested.

The search input will be tested for functionality. By functionality, we mean that the user will be able to search items, in this case, apartments close to campus by using the following: Property (category dropdown), search by zip code, city, or simply using the search bar to look for a specific field in the listing.

Property (category dropdown) will be composed of 4 items, which are: All (Property) by default, Apartments, Studios, Condos, and Houses. These items will filter listings based on listing- type. Second, the search bar will filter the listing by zip code, based on the user input.

2.2 Usability Test Plan for 'Search' - Test Plan (background and setup)

System Setup: GatorHub website supports Google Chrome and Mozilla Firefox browsers and one of our teammates will use either browser to test the system by inputting various specific inputs in the search bar and making sure that the correct results are displayed. The tester will use edge cases and input only those fields that are intended to work in the search bar. These will be based on the functionality of the search bar and we will determine that the results agree with our expectations. For example, searching by zip work must work and display listings with that zip code.

Starting Point: The search bar functionality is present in the navbar at the top of every webpage. In addition, it functions the same regardless of which page the user is searching from. Home page of the site is reachable at http://54.151.14.7/views/index.php

Intended Users: The search bar is implemented for all users looking to find a place to live. The user base for this website is focused towards students at SFSU ages 18-30. Expected computer skills are average but should not be technically demanding. Searching is intended to be the primary way for users to find the living spaces that they need or want. Users (landlords) looking to post a living space can also search for similar items to use as an example when creating their own listings.

Evaluation: Efficiency in effort and design will be tested in this usability plan. Task should be done with minimal clicks and instructions. User satisfaction will be surveyed after the usability tests to measure ease of use and website clarity. San Francisco State students will be tested on comfort and acceptability.

What are we measuring: We will be measuring how the user feels through the process of searching for a specific listing and how does s/he feels with the general search procedure, and also how intuitive the search procedure is.

2.3 Usability Test Plan for 'Search' - Usability Task Description

Open a Google Chrome or Firefox browser and navigate to the project page:

http://54.151.14.7/views/index.php

Perform the following Tasks:

1. Search for ALL listings
2. Search for ONLY apartment listings
3. Search for ONLY house listings that are for Rent
4. Search for ONLY condos that are for Sale

Review the search results for each search and verify that only relevant results are being shown. Correct results indicate successful completion of task criteria. Expected benchmark for full completion of the task is 1 - 2 minutes.

Effectiveness - would be measured by the amount of clicks the tester takes to perform each search in the task. By reviewing the click count of search, we can examine how effective each action is in our search functionality.

Efficiency - would be measured by the amount of time taken and confusion caused by the usability task. A large amount of time taken would indicate that the interface for searching is unclear for first-time users to perform the given searches. If any one of those tasks takes more time than the others, we can pinpoint which functionality in our search needs to be reviewed.

Satisfaction - would be measured by deploying the following survey after the task is completed where we 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 the reviewer:

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Search bar and drop down menu were easy to use (good GUI design)					
Search results were informative and clearly displayed (reliable results)					
Searching was prefered to browsing all listings (convenient)					

There can also be a comments section along with the survey, where the user can put down how he felt about the whole search procedure. This survey results will signal to us the overall satisfaction of the product, specifically the search bar, and give us more insight into both its usability, marketing and future success among SFSU users that are looking for a place to live.

3. QA Test Plan:

3.1 Test objectives - What is being tested

We are testing the functionality of using the website's search, and testing it for bugs. We will test input validation where it should only take up to 40 characters. The search and dropdown listing category selected after a search should also remain persistent. Search and category menu should be fully functional from any page in the website. Only correct and relevant results should be displayed for each tested search. Empty searches should return all found listings.

3.2 HW and SW setup (http://54.151.14.7/views/index.php)

Our website should function on any machine connected to the internet that can run:

1. Google Chrome or 2. Firefox browsers.

We must also support the latest two versions of Google Chrome and Firefox. Browsers must be updated to ensure this. No further software required for testing.

3.3 Feature to be tested

This QA test will test the functionality of the search function in all of the major pages: home page, search result page, rent out page, listing detail page and user account page. The test will test for the following expected Functionality:

1. The search result page shall show all listings that match the search query (the search query being either ZIP CODE (5 digit number), PROPERTY (apartment, studio, condominium, or house), TYPE (rent/buy), as well as the number of found listings.
2. All results should be readable and openable to the listings detail page.

3. The search query shall not disappear from the search box after searching.
4. All listings shall include a listing title, listing details, listing price, ZIP code, number of bedrooms, pets allowed, and distance from the apartment to SFSU along with a map showing the Origin ‘O’ (listing address) and destination ‘D’ (SFSU address) pins.
5. All results shall be the same across both browsers.

3.4 Test Cases

3.4.1 Test 1 – Searching by ZIP CODE: Search for the ZIP code “94122” from all of the Testing Pages. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers (Google Chrome, Firefox).

3.4.2 Test 2 – Searching by PROPERTY: Search for the property “House” from all of the Testing Pages. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.4.3 Test 3 – Searching by TYPE: Search for the option to “Rent” from all of the Testing Pages. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.5 Test Results

Test #	Test Field Search	Test Description	Test Input	Expected Output	Test Results
1	Zip Code	Search listings in a specific zip code	“94122”	Listings in 94122 that are apartments, studios, condos, or houses for rent and sale	Pass/Pass
2	Property	Search listings that are Houses	"House"	Listings that are Houses for rent and sale in any Zip code	Pass/Pass
3	Type	Search listings that are for Rent	"Rent"	Listings for rent that are apartments, studios, condos, or houses in any Zip code	Pass/Pass

4. Code Review:

Coding Style: Careful indentation, consistent use of spaces over tabs, clear and concise commenting in the style of "/* text */" or "// text" and starting brackets on the same line as function call (Egyptian braces).

Peer review was conducted on the search feature mentioned in usability testing section 2. The snippet from index.php is included in the document and feedback from index as well as listing_search.php (file is 250+ lines so did not paste it).

Snippet from Index.php prior to peer review:

```
<!-- Search -->
<form method="POST" action="/views/listings/result.php">
  <div class="row">
    <div class="col-lg-12">
      <div class="row">

        <!-- Zip Code -->
        <div class="col-lg-6 col-md-3 col-sm-12 p-0 m-1">
          <input type="text" name="zip_code_listing" class="form-control search-slt"
            placeholder="search by zip code">
        </div>


        <!-- Property -->
        <div class="col-lg-2 col-md-3 col-sm-12 p-0 m-1">
          <select class="form-control search-slt" id="property_type"
            name="property_type_listing">
            <option value="">Property</option>
            <option value="apartment">Apartment</option>
            <option value="studio">Studio</option>
            <option value="condonimium">Condominium</option>
```

```
                <option value="house">House</option>
            </select>
        </div>

        <!-- Type -->
        <div class="col-lg-2 col-md-3 col-sm-12 p-0 m-1">
            <select class="form-control search-slt" id="type"
                name="payment_type_listing">
                <option value="">Type</option>
                <option value="rent">Rent</option>
                <option value="buy">Buy</option>
            </select>
        </div>

        <!-- Submit btn -->
        <div class="col-lg-1 col-md-5 col-sm-5 p-0 m-1">
            <input type="submit" class="btn btn-dark">
        </div>
    </div>
  </div>

</form>
```

Feedback for index.php:

```
<!--
    CODE REVIEW:

    Appears to be a well-written search form. It makes good use of boostrap's breakpoints, is well styled
    and uses things like the placeholder attribute on input. Indentation follows coding style and is readable,
    Descriptive/well-placed comments made each aspect of search clear.
-->

<!-- Search -->
<form method="POST" action="/views/listings/result.php">
    <div class="row">
        <div class="col-lg-12">
            <div class="row">

                <!-- Zip Code -->
                <div class="col-lg-6 col-md-3 col-sm-12 p-0 m-1">
                    <input type="text" name="zip_code_listing" class="form-control search-slt"
                           placeholder="search by zip code">
                </div>
```

Feedback for listing_search.php:

```php
/*
    CODE REVIEW:
    Repetitiveness could be minimized, maybe put all of the search property names in an
    array and loop through that? There's some additional complexity added there by booolean arguments
    (since they need to be converted from true/false to 1/0), but that can be addressed with some type inspection.

    There is a nice use of console.log in a script tag, makes debugging this module easier. The use of $_SERVER['document_root']
    is clever as it prevents relative imports (PHP has strange import semantics). Implementing this prevents difficult-to-debug
    issues further down the line.

    The comment at the top that mentions that this module does something on load which is useful & makes it clear that stuff
    is being done when the module is imported.Overall could use slightly more commenting but seems to generally following
    coding style and is clean/readable.

*/

// retrieve form values
if (isset($_POST['title_listing'])) {
    $title = $_POST['title_listing'];
}

if (isset($_POST['city_listing'])) {
    $city = $_POST['city_listing'];
}
```

5. Self-Check on best practices for security:

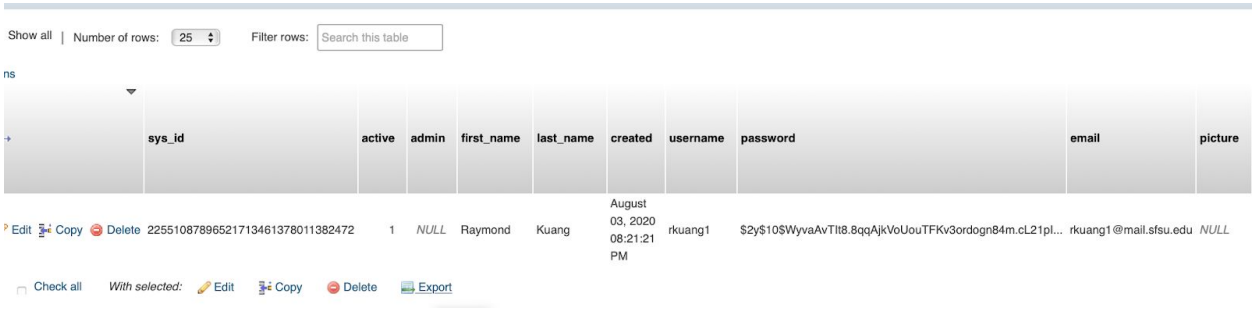
1. **List of major assets you are protecting**
    - User personal information
    - Images on server
    - Database integrity

2. **List of major threats for each asset above**
    - User personal information
    - Images on server
    - Database integrity

3. **For each asset say how you are protecting it**
    - **User personal information**: password encryption using php's password_hash() function
    - **Images on server**: file name encryption using php's password_hash() function
    - **Database integrity**: using PDO which allows for positional and named placeholders to prevent SQL injections, also using php's addslashes() to escape characters to further protect against SQL injections

4. **Confirm that you encrypt PW in the DB**

- User passwords are encrypted before being sent out to the database. Here you can see a screenshot showing the encrypted user passwords in the database:



Passwords are hashed in the API before inserting or updating a record

```
59    function insert()
60    {
61        // has the user's password before inserting the new record
62        $this->password = password_hash($this->password, algo: PASSWORD_DEFAULT);
```

```
97    function update()
98    {
99        // has the user's password before updating the current record
100       $this->password = password_hash($this->password, algo: PASSWORD_DEFAULT);
```

5. **Confirm Input validation (list what is being validated and what code you used**
    - User's email and confirm email must match
    - Email must be an official SFSU one
    - User's password and confirm password must match
    - Search is restricted to 40 alphanumeric characters

We have implemented a listener and once the submit button on the form is clicked, the following code is used for input validation of the registration form:

```
function validate(e) {

    //validate email is SFSU
    if (document.getElementById( elementId: "email_register").value.split('@')[1] !== 'mail.sfsu.edu') {
        alert('you must register with an SFSU email address');
        e.preventDefault();
        return false;
    }

    // validate emails match
    if (document.getElementById( elementId: "email_register").value !== document.getElementById( elementId: "email_confirm_register").value) {
        alert('emails do not match, please reverify your email');
        e.preventDefault();
        return false;
    }


    // validate passwords match
    if (document.getElementById( elementId: "password_register").value !== document.getElementById( elementId: "password_confirm_register").value) {
        alert('passwords do not match, please re-enter your password');
        e.preventDefault();
        return false;
    }
}
```

Snippet from index.php that restricts search entry to 40 characters or less:

```
<div class="col-lg-6 col-md-3 col-sm-12 p-0 m-1">
    <input type="text" name="zip_code_listing" class="form-control search-slt" maxlength="40"
           placeholder="search by zip code">
</div>
```

Snippet from result.php that restricts search entry to 40 characters or less:

```
<div class="col-lg-6 col-md-3 col-sm-12 p-0 m-1">
    <input type="text" name="zip_code_listing" class="form-control search-slt" maxlength="40"
           placeholder="search by zip code">
</div>
```

6. Self- Check: Adherence to original non-functional specs:

Non-functional Specs	Done/ Not Done
1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0.	Done
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	Done
3. Selected application functions must render well on mobile devices	Done
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.	Done
7. The language used shall be English (no localization needed)	Done
8. Application shall be very easy to use and intuitive.	Done
9. Google analytics shall be used	Done
10. No email clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application	In Progress
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	Done

12. Site security: basic best practices shall be applied (as covered in the class) for main data items.	Done
13. Media formats shall be standard as used in the market today.	Done
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, Summer 2020. For Demonstration Only" at the top of the WWW page. (Important so as to not confuse this with a real application).	Done