SW Engineering CSC648/848 Summer 2020

House-a-Gator

Team 02

Milestone 4

08/04/2020

Team members:

Team Lead	Raviteja Guttula (rguttula@mail.sfsu.edu)
Front End Lead	Swetha Govindu
Front End Developer	Henry Meier
Front End Developer	Kevin Zhou
Front End Developer	Troy Turner
Back End Lead	Ashwini Uthirakumar
Back End Developer & Github Master	Fiona Senchyna

History:

Date Submitted:	08/04/2020
Date Revised:	08/05/2020

Table of Contents:

1. Product summary:

Our team has created an apartment and housing website for renting and selling University real estate called House-a-Gator. It is specifically focussed on San Francisco State University students and faculty. The website design is centered around the SFSU community. Guest users can browse the listings, registered users can post listings, however only those registered with an @mail.sfsu.edu or @sfsu.edu email address can buy or rent on the site. The organized practices will allow the users to see the location of a property relative to the university campus. The website user interface is very user friendly and naturally invokes carefree campus life. We care very much about making SFSU a more affordable place to study and work for anyone who wants to study here. The product is marketable as an easy to use and effective way to list housing. It takes the hassle out of finding a place to live.

To access the website go to http://ec2-54-80-160-88.compute-1.amazonaws.com. One unique feature about House-a-gator is that the houses can be filtered based on the distance from SFSU campus.

These are the features:

- All users shall be able to search based on title and description
- All users shall be able to filter listings based on home type
- All users shall be able to filter listings based on distance from SFSU campus
- All users shall be able to filter listings based on listing type (Rent or Sell).
- Guest users shall be able to register
- Registered users shall be able to post new listings
- Registered users shall have a personalized dashboard to check listings
- Registered users shall have a personalized dashboard to check the messages received.
- Admin shall be required to approve listings in 24 hours, before it goes live
- Admin shall be able to delete flagged listings
- Admin shall be able to delete flagged accounts and users

2. Usability test plan

The major function that we have selected to be tested for usability is the **search** functionality.

2.1 Test objectives:

Users would be presented with a website on a browser environment known to be functional. They will be asked to perform a series of tasks emulating common usage patterns. Afterwards, they will be instructed to fill out a survey form on their ease of usage.

The objective here is to test the "**search**" functionality extensively. Since we are closer to the release of the product, we will perform the **Validation** test. On the House-a-gator website, the "search" function should return the listings matching the search query.

2.2 Test background and setup:

The user will be instructed to use "search" functionality without any details on the specific process.

Hardware setup:
macOS catalina version 10.15.2
Processor : 2.6 GHz 6-Core Intel Core i7
Memory : 16 GB

Software setup:
Google chrome (Version 84.0.4147.105), Firefox(79.0)

House-a-gator website is hosted on an AWS EC2 instance, here is the link http://ec2-54-80-160-88.compute-1.amazonaws.com . Listings on this website can be browsed by anyone and only registered users should be able to post listings. Only those registered with @mail.sfsu.edu or @sfsu.edu email should buy/rent. The usability test can be performed by

either one of the above mentioned users. Users should be able to perform search operations equally well, regardless of being a guest user or registered user.
The usability metric to be evaluated will be **user satisfaction**. After performing the task, users should participate in a survey and rate the product depending upon the usability.

2.3 Usability Task description:

- Go to House-a-gator website
- Search for an "house"
- Check to see if 12 listings are returned
- Once completed please fill-in the likert questionnaire for feedback

Task	**Description**
Task: Find an "house"	Search for an "house"
Successful completion criteria	Listings obtained
Benchmark	~ 10 secs

Effectiveness is the completeness with which users achieve goals. To measure the effectiveness of our product, we would calculate the percentage of users who were able to use the "search" functionality successfully to the total number of users who performed the same task.

Efficiency is defined as the resources spent by the user in order to ensure accurate and complete achievement of their goals. To gauge the efficiency of our product we would measure the time taken, number of clicks performed, and the number of screens visited by the users to achieve their goal.

Satisfaction aims at subjective thoughts of the user. A survey is then taken by the users to rate how they felt about usability.

Questionnaire:

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I found search bar on home page					✓
I found searching for listings is simple					✓
I found search results to be easy to read and understand					✓

Other comments:

3. QA test plan:

3.1 Test objectives:

The objective of QA test is to test the "search" functionality in House-a-gator website for correctness and permanence of specs and ensure that the search returns the expected number of listings matching the search query. We test the "search" functionality by entering a search query that matches the title or description of the listing. This is accomplished with the use of %like. This test also involves cross browser testing.

3.2 HW and SW setup:

Hardware setup:
macOS catalina version 10.15.2
Processor : 2.6 GHz 6-Core Intel Core i7
Memory : 16 GB

Software setup:
Google chrome (Version 84.0.4147.105), Firefox

3.3 Feature to be tested:

This QA test is created to test the "search" feature of House-a-gator available in the home page of the website.

"Search" functionality can be used to search by:

- Text/Keyword (eg. "house")

We have the following types of test cases

- Functional test cases -> to ensure the functionality works as expected
- Negative test cases -> Appropriate message has to be displayed when Invalid search terms are entered.

The search results are returned in the same page and the total number of listings are visible at the top. To view the complete details about a listing, click anywhere on the listing tile and the details regarding the listing opens in a new tab. Results are the same across all the major browsers (Firefox and Chrome in this case).

3.4 QA Test plan:

No	Title	Description	Input	Expected correct output	Test results for Firefox	Test results for Chrome
1	Test test search	Testing text search input and %like function	Enter “house” in the search field	Get 12 listings, all containing the string “house” in title or description of the listing	PASS	PASS
2	Test negative case	Testing negative case for search functionality	Enter “-1” in the search field	Alert pops up indicating that only alphanumeric characters of max length 40 are allowed	PASS	PASS
3	Test SQL injection	Test SQL injection for search functionality	Enter ' OR '1'='1 in the search field	Get appropriate error message	PASS	PASS

4. Code Review:

4.1 Coding style:

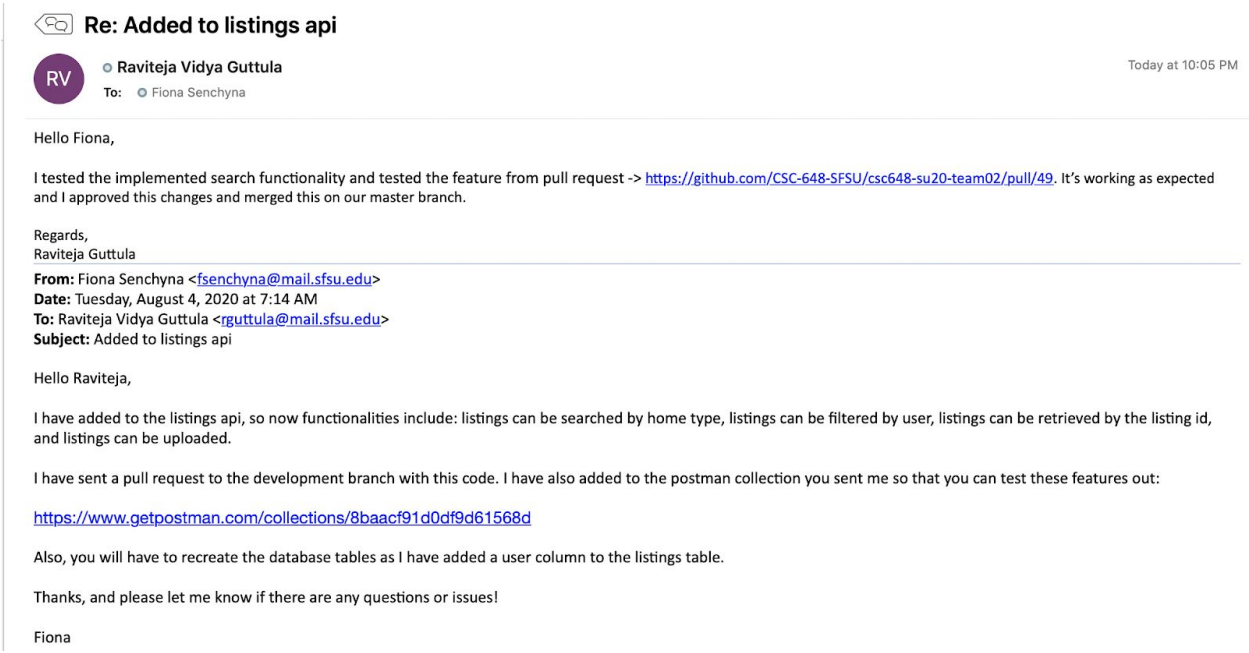
We have followed a variety of coding styles for coding both front-end pages and back-end API’s.

- Back-end: Coded entirely in Python following the PEP guidelines to create classes, functions, schemas. Docstrings, header and inline # comments are used in the code. Code is being formatted with Black, a code formatter for python.

- Front-end: Proper indentations are used throughout the html files. The css and javascript files are kept external. All the sections of the website are logically ordered and best practices are being followed to ensure fast render times.

4.2 Peer review:

Below is the peer review for “search” functionality.



Re: Added to listings api

RV **Raviteja Vidya Guttula** Today at 10:05 PM
To: Fiona Senchyna

Hello Fiona,

I tested the implemented search functionality and tested the feature from pull request -> https://github.com/CSC-648-SFSU/csc648-su20-team02/pull/49. It’s working as expected and I approved this changes and merged this on our master branch.

Regards,
Raviteja Guttula

From: Fiona Senchyna <fsenchyna@mail.sfsu.edu>
Date: Tuesday, August 4, 2020 at 7:14 AM
To: Raviteja Vidya Guttula <rguttula@mail.sfsu.edu>
Subject: Added to listings api

Hello Raviteja,

I have added to the listings api, so now functionalities include: listings can be searched by home type, listings can be filtered by user, listings can be retrieved by the listing id, and listings can be uploaded.

I have sent a pull request to the development branch with this code. I have also added to the postman collection you sent me so that you can test these features out:

https://www.getpostman.com/collections/8baacf91d0df9d61568d

Also, you will have to recreate the database tables as I have added a user column to the listings table.

Thanks, and please let me know if there are any questions or issues!

Fiona

5. Self-check on best practices for security:

- List major assets you are protecting
  - ❏ User data
  - ❏ Listings
  - ❏ Media
  - ❏ Messages
  - ❏ Database

- List major threats for each asset above
  - ❏ Personal details about the user and password could get stolen
  - ❏ Listings data can be scraped
  - ❏ Images are stored in a filesystem

- ❏ Unauthorised access to the messages

- For each asset say how you are protecting it
  - ❏ User authenticated before successful login
  - ❏ Ratelimit is applied on the API for search functionality
  - ❏ Images are stored in a filesystem
  - ❏ Login is required to view the messages for the user and the session is validated to prevent unauthorised access.

- Confirm that you encrypt PW in the DB
  - ❏ Yes, password is encrypted and stored in the DB

- Confirm Input data validation
  - Yes, the search bar input is restricted to 40 alphanumeric characters and an appropriate message is displayed if the user enters an invalid input.

6. Adherence to original Non-functional specs:

Non-functional specs	**DONE/ON TRACK/ ISSUE**
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
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	DONE
Selected application functions must render well on mobile devices	ON TRACK
Data shall be stored in the team's chosen database technology on the team's deployment server.	DONE

No more than 50 concurrent users shall be accessing the application at any time	ON TRACK
Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	DONE
The language used shall be English (no localization needed)	DONE
Application shall be very easy to use and intuitive	DONE
Google analytics shall be used	ON TRACK
No e-mail 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	DONE
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	DONE
Site security: basic best practices shall be applied (as covered in the class) for main data items	ON TRACK
Media formats shall be standard as used in the market today	DONE
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	DONE
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