Final Project for SW Engineering Class
CSC 648-848 Spring 2019

Team 5

Gatorbnb

Peter Le - team lead (ple2@mail.sfsu.edu) local
Jay Sunga - front end lead
Wesley Goldfisher - back end lead
Franky
Mehi Ledwon
Tanya Wong
Anthony Owyeong

URL of the Demo: http://13.57.19.213/

Date: 5/19/19

Project Summary

- Name of product: Gatorbnb
- Url: http://13.57.19.213/
- Committed Functions
    - User registration - an unregistered user shall be able to sign up and create an account
    - User Login - a registered user shall be able to log into their registered account
    - Browsing - all users shall be able to browse through the current listings
    - Filtering - all users shall be able to filter the search results by House Type, number of Bedrooms, number of Bathrooms, Price and Distance from SFSU with a drop list
    - Search- all users shall be able to use free text search for zipcode and city
    - Post - a registered user shall be able to Post a new listing, while an unregistered user may not
    - Message - a registered user shall be able to Message the user who created a listing they're interested in
    - Map - from the selected listing, all users shall be able to view a map of the housing will be available
    - Search - all users shall be able to search for a listing in the search bar
    - Admin- administrators shall be able to deny or approve listing and can delete users from the database

3. Milestone Documents:

Milestone 1:

1. Executive summary:

The process of finding a place to live near the school while studying can be more complicated and a struggle than it has to be. The student is working hard for their college degree and now they have to put in the effort to find a place to live near the school. Is it possible to find a place near the school for an easier commute? Where should they look to find the best deal for their given situation? This project aims to solve this problem by offering a platform for San Francisco State University students to find a place to rent with ease. With us knowing our demographics being students, we can focus on their needs rather than an application using a larger demographic to hinder their ability to offer the best deal.

The problem that we solve is the difficulty of finding a place for San Francisco State students as they are going to school. Having a one-stop place to solve all their searching needs and easily giving them available listings will make their lives easier and reduce the stress of housing. Students can go to our website and search for rooms for rent that will match their need, as we are able to fit their needs and connect them to better sellers. This product is about connecting San Francisco State students to sellers who offer a place to rent. Allowing students and sellers to connect is the current dilemma that students are facing when they have a lot on their plant to search for a place. Showing available listings to students with the ability to select criteria that that would like us to accommodate for them.

This product is designed by a team of talented software engineers attends San Francisco State University. A team of seven students with different backgrounds and specialties are tasked with designing, developing, and maintaining the site. The team is split into two groups, which is the backend and frontend team which is in charge of the functionalities that are designated to that particular section by the team lead to create this product.

2. Persona and Use Cases

2.1 Groups

- San Francisco State Students / Potential Tenants
- Landlords / seller
- Admin

2.2a Persona: Buyer
Major responsibilities: studying yet also having fun
Demographics: 18-25 years of age
Goals and tasks: short on money yet, staying on track with school while enjoying their personal life
Environment: School and work
Skill level in mobile use: well
Reason to use application: to find a place to live and study in

2.2b Persona: Landlord
Major responsibilities: retired
Demographics: 60 -75 years of age
Goals and tasks: making money by renting out house, having time for family, enjoying hobbies
Environment: Family and friends
Skill level in mobile use: below average
Reason to use application: renting out a room to help students looking for a place to rent in

2.2c Persona: Admin
Major responsibilities: maintaining Gatorbnb
Demographics: 23 - 35 years of age
Goals and tasks: work
Environment: Friends and coworkers
Skill level in mobile use: good/ professional
Reason to use application: to maintain the application by monitoring new accounts and postings

2.3 Use Cases

2.3a Unregistered User - Jonathan

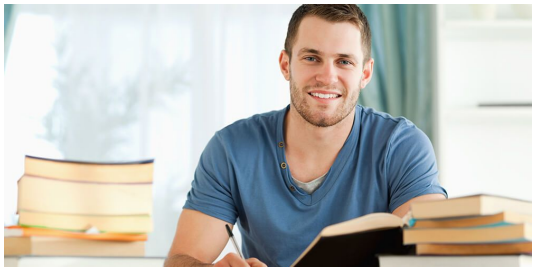
Jonathan is a transfer student looking for a place to live for his first year at San Francisco State University. He uses the Gatorbnb website believing he can quickly find a place to rent. As an unregistered user, he is only able to search for and see postings that's out on the market. To make searches faster to satisfy his needs he will be able to filter his searches by home type, distance, price, animal friendly, fully furnished and the amount of bedrooms. After reviewing the apartment that he likes, he attempts to contact the landlord. He clicks the contact button but a prompt pops up requiring him to log in before he is able to proceed. He realizes that he can't contact the seller of the house that has caught his attention unless he registers for an account.

2.3b Registered User - Jaime

Jaime is working full-time while also maintaining a relationship with her friends. She enjoys family life and work life but has a busy schedule. Jaime's parents are asking for her help; her family has an empty house and her parents want to rent it out so it's being inhabited. Knowing that the house is relatively close to San Francisco State University, Jaime uses Gatorbnb to rent out the house so it caters towards university students. As a first time user, Jaime creates a post for the house. While creating her post, she adds many recent pictures of the house that display various perks and information about the house: a garage to park cars, 3 bathrooms, 5 bedrooms, fully furnished with plenty of kitchen appliances, a washer and dryer and a backyard with a sun patio. After finishing adding all the details about the house she attempts to click the "Post" button but is prompted to register first. To register she must add valid credentials such

as her full name, a username, password and her email (for registration purposes only). As she finishes her registration, she can then finally post her listing. With a registered account, she is now able to login and go to her dashboard where she can find the listing of her house and check for messages of students willing to rent.

2.3c Registered User - Max



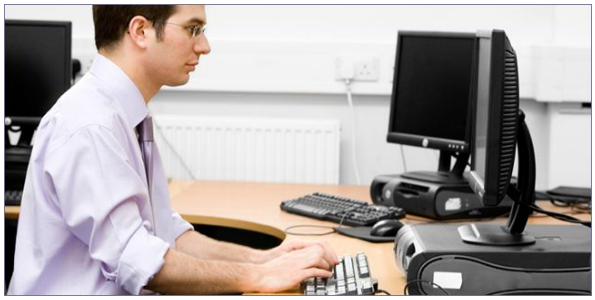
Max is a senior college student at San Francisco State University. He has to balance work and school. He's close to graduating and wants to focus on school work but needs to work in order to pay for his rent. His roommates have already graduated and left, so currently he is living in an empty house with just himself paying full rent. To avoid needing to work overtime to compensate for the lack of roommates, he will use Gatorbnb to post a listing of needing roommates. As a registered user Max is able to login to his already registered account. He creates a posting by clicking the "Post" button. He creates a description that he's looking for roommates willing to room with him. Max will constantly check his messages throughout the next few weeks until he has enough roommates and will be able to delete his post with the "Delete" button that is easily recognizable by its red color. By using the website, he will be able to focus his time on studying without the need to worry about his renting issue.

2.3d Registered User - Tabitha

Tabitha is a kind-hearted middle-aged woman who lives alone. Her children live away from home, so her house is relatively empty. She wants to rent out parts of her house she doesn't use. So she goes onto the website and clicks the "Post" button that is big enough to see for her elderly eyes. As she registers her house up for selling, she is able to add a label "Pet-friendly" which will allow renters to find a pet-friendly house easily through filtering. As a pet lover herself with her 2 cats, a snake, and a chicken she doesn't mind pet owners. In fact, prefers

them and adds that to the listing to encourage pet owners who go to SFSU to rent from her. She is allowed to add all this information for renters to read while adding her postings. By using the website she can use parts of her unused house and potentially be a pet watcher.

2.3e Admin - Josh



Josh is the website admin of Gatorbnb. His main job as an admin is to monitor all registered users accounts and reviewing content. Josh can log in to a dashboard type view which gives him information on all users and their listings in the order of most recent. He can click the registered users tab to give him a list of all registered users with newer ones at the top. Josh will be able to see all the users data and it's up to him to review their information to ensure they're accurate and valid. If Josh sees a spam account he can delete it by clicking to removing it from the database. Josh can also view all new listings that want to be posted by clicking the new listings tab which is only available to admins. Before any listing or post goes public, Josh has to approve that the listing is valid. The administrator dashboard gives him an overview of the database that allows him to review all content without having to have high-level knowledge of the database system itself.

3. List of main data items and entities:

- **Unregistered User**: any person utilizing the website application who does not hold an account on the website or is not logged in at the moment
- **Registered User**: a person who is logged in via an authentication method to utilize the website application and whose main purpose is to browse among listings as well as contact sellers or post possible listings for users to browse among
- **Site Administrator/Admin**: a person utilizing the website application to review listings and to be able to view as well as remove potentially unwanted listings, users, or content
- **Filter/Filter Option**: specifies a user-defined preference about what listings are to be displayed by the following possible guidelines: price, home type, bedrooms, bath distance, and/or type
- **Listings:** a reference to a house, apartment, or property for sale that contains the following detailed information: address, image(s) of location, price, home type, how many bedrooms and bathrooms, and seller contact materials
- **Log in**: references the action of authenticating yourself as a registered user to use the registered user functions of the website
- **Log out**: references the action of signing out of your logged in account to go to a basic unregistered user function of the website
- **Registration**: allows a user to fully utilize the website application and contains username, email and password

4. Initial Functional requirements:

1. **Unregistered Users:**
    a. Unregistered users shall be able to register as renter or seller and log in
    b. Unregistered users shall be able to browse through all of the listings
    c. Unregistered users shall be able to filter the listings by price, beds, baths, home type and pets
    d. Unregistered users shall be able to view the public details of any listing
    e. Unregistered users shall be able to register for an account

2. **Registered Users:**
    a. Registered Users shall be able to do everything an unregistered user is allowed to do
    b. Registered Users shall be able to log in and log out of an account
    c. Registered Users shall be able to contact other Registered Users through the website
    d. Registered Users shall be able to save listings in the website
    e. Registered Users shall be able to post listings
    f. Registered Users shall be able to edit their listings
    g. Registered Users shall be able to remove their listings

3. **Site Administrators:**
    a. Site Administrators shall be able to view all listings on the database
    b. Site Administrators shall be able to view all Registered Users and Sellers on the database
    c. Site Administrators shall be able to view all messages between Registered Users and Sellers
    d. Site Administrators shall be able to approve or reject listings for posting
    e. Site Administrators shall be able to remove any listing on the site
    f. Site Administrators shall be able to remove the account of a Registered User or Seller

5. List of non-functional requirements:

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).
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers
3. Selected application functions must render well on mobile devices
4. Data shall be stored in the team's chosen database technology on the team's deployment server.
5. No more than 50 concurrent users shall be accessing the application at any time
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
7. The language used shall be English.
8. Application shall be very easy to use and intuitive.
9. Google analytics shall be added
10. No e-mail clients shall be allowed
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated.
12. Site security: basic best practices shall be applied (as covered in the class)
13. Before posted live, all content (e.g. apartment listings and images) must be approved by the site administrator
14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
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. (Important so as to not confuse this with a real application).

6. Competitive analysis:

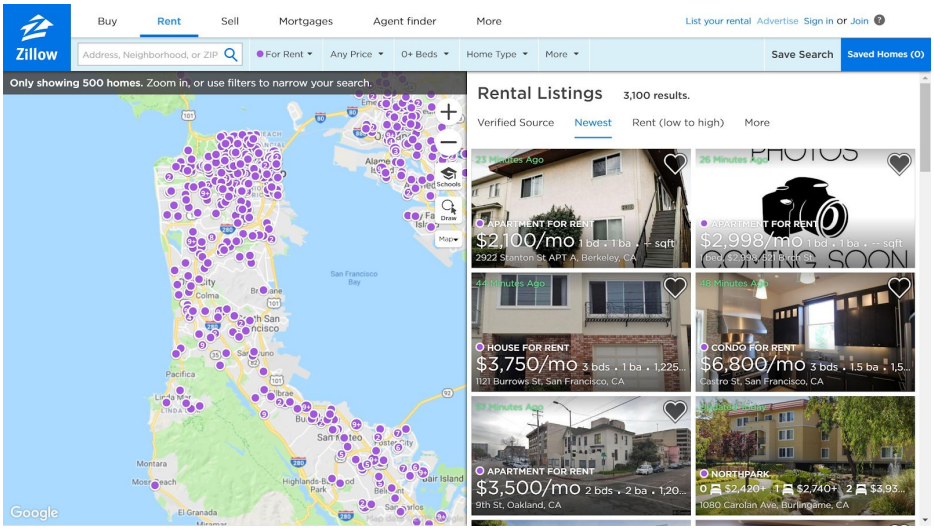
	Zillow	Trulia	Craigslist	SFSU apt roommate fb group	**Gatorbnb**
Design of UI	+	+	-	-	+
Filter Option/ Tags	+	+	-	n/a	+
Map Integration	++	+	+	n/a	+
Contact	-	-	-	+	++
Caters to SFSU students	-	-	-	+	++

++ Superior, + Good, - Not Good

6.1 Competitive Summary and Advantages

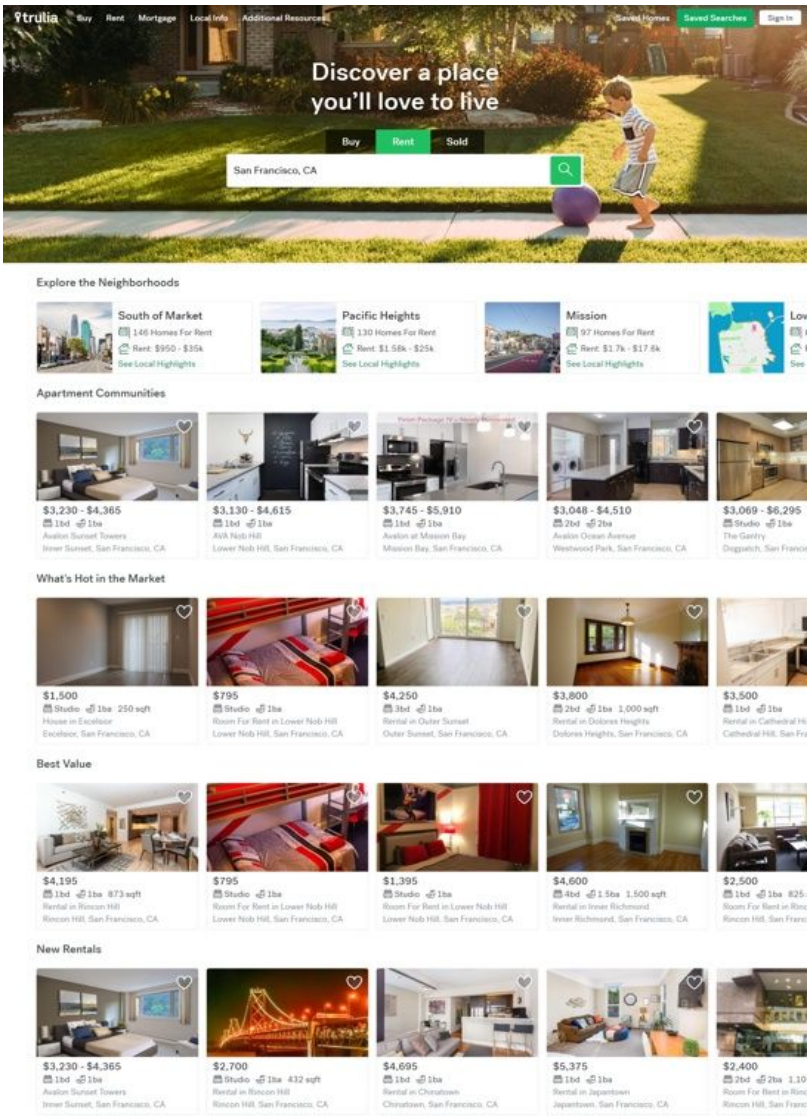
Websites such as Zillow and Trulia offers nice design of user interface, however they don't do well in muc of the other categories. Facebook group and Craigslist offers good contacts and caters well towards SFSU students, but their design and filters lack in that department. Gatorbnb offers good design of user experience, good filter / tags, and map integration. It however excels in being able to connect sellers and buyers, along with being catered towards SFSU students allowing for better results in terms of rooms and comfortability for the user.

6.2a Zillow



Zillow is an online real estate database where users go to for finding listings that are available on the market. They allow tags such as verified source, newest, rent (low to high), rent (high to low), bedrooms, bathrooms, square feet, year built, lot size, zestimate (high to low), zestimate (low to high) as well as a search bar. They include a map showing number of listings in an area that updates when you zoom in. The map has built information when you hover / choose a listing. They allow the ability to request a tour of the listing, save listing, show similar listing, save listing, and description of the listing. Our website differs from Zillow by us being specifically targeting San Francisco State Students, therefore we can have better tags and search features. Our map will have details that will be more a lined with our users.

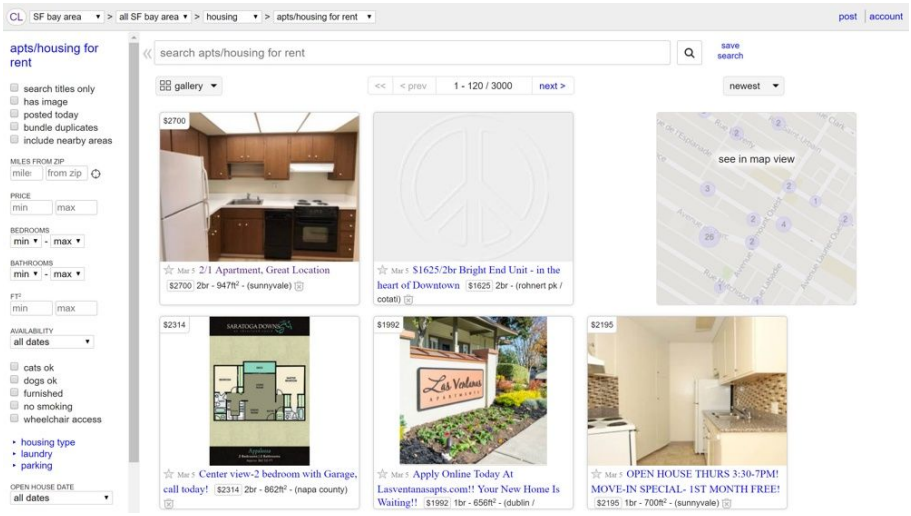
6.2b Trulia



Trulia is a home and neighborhood site for buyers and renters to find locations that fit their needs. They allow tags such as houses for rent, apartments for rent, rooms for rent, specific

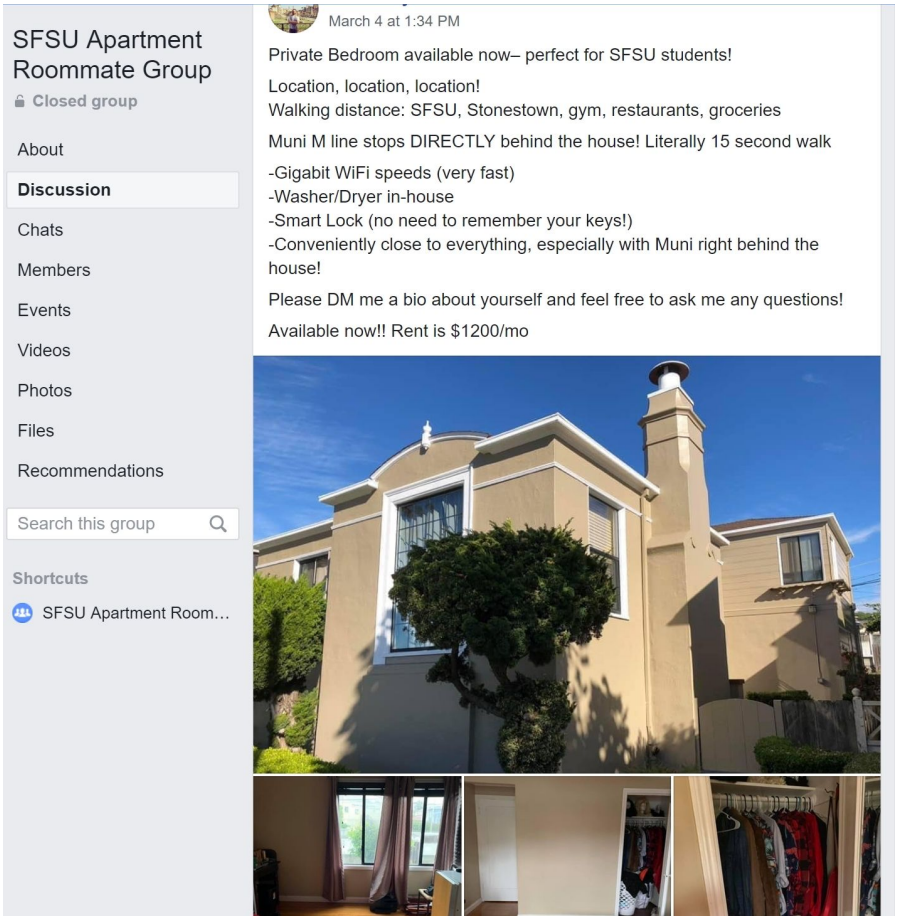
neighborhoods and apartment communities, ‘hot on the market’, ‘best value’, new rentals, and a search bar. The listings have options for map and street view, crime rate in the area, schools, common commute choices, and other local information that a buyer / renter would like to know. Our website differs from Trulia by us being specifically targeting San Francisco State Students, therefore we can have better tags and search features. Our details will be more geared towards the students giving them more control over the information that they want to see.

6.2c craigslist



Craigslist is a buy and trade website where users can post anything they want to offer. The have a section where users can search for housing, which offers options such as tags, and search for details / restricts on the place to rent. The contact is dependent on the poster, which is not regulated by craigslist. Our website differs from Craigslist in the sense that it is a housing / real estate website and not just a buy and trade page. This means that we are more geared for offer better search results for the users as well as better communication between the renter and the seller since we regulate the information that is posted by the seller.

6.2d SFSU apartment roommate group on Facebook



SFSU apartment roommate group is a Facebook group where users can post information about housing. Post tends to get buried and all the post on the group looks similar to one another making it difficult to tell the difference between buyers and sellers. The page is not updated since there is no moderator, therefore it's also not as safe to use. Our website will differ from facebook group by us offering better search results rather than a feed. Having the ability to have the same format for each housing / apartment also gives user the peace of mind of not going to a 3rd party website.

7. High-level system architecture and technologies used

1. **Frameworks:**
    a. React
    b. Bootstrap
    c. NodeJS 11.0.0
    d. Npm
    e. Postgres

    f. Express
2. **APIs:**
    a. Google Maps
    b. Google Analytics
3. **Tools and Systems:**
    a. GitHub
    b. pgAdmin 4.2
4. **Supported Browsers:**
    a. Chrome
    b. Mozilla Firefox
5. **Deployment Platforms:**
    a. Ubuntu
    b. Amazon Web Services

8. Team:

- Peter Le: team lead
- Franky: front end lead
- Wesley Goldfisher: back end lead
- Mehi Ledwon: front end
- Tanya Wong: front end
- Anthony Owyeong: back end
- Jay Sunga: back end

9. Checklist: (answer each bullet point with DONE or ON TRACK or ISSUE)

- Team found a time slot to meet outside of the class: DONE
- Github master chosen: DONE
- Team decided and agreed together on using the listed SW tools and deployment server: DONE
- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing: DONE
- Team lead ensured that all team members read the final M1 and agree/understand it before submission: DONE
- Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.): DONE

Milestone 2:

1. Data Definitions V2

- **Unregistered User**: any person utilizing the website application who does not hold an account on the website or is not logged in at the moment
- **Registered User**: a person who is logged in via an authentication method to utilize the website application and whose main purpose is to browse among listings as well as contact sellers or post possible listings for users to browse among
- **Site Administrator/Admin**: a person utilizing the website application to review listings and to be able to view as well as remove potentially unwanted listings, users, or content
- **Filter/Filter Option**: specifies a user-defined preference about what listings are to be displayed by the following possible guidelines: price, home type, bedrooms, bath distance, and/or type
- **Listings:** a reference to a house, apartment, or property for sale that contains the following detailed information: address, image(s) of location, price, home type, how many bedrooms and bathrooms, and seller contact materials
- **Log in**: references the action of authenticating yourself as a registered user to use the registered user functions of the website
- **Log out**: references the action of signing out of your logged in account to go to a basic unregistered user function of the website
- **Registration**: allows a user to fully utilize the website application and contains a username, email, and password
- **Favorites:** allows a user to access bookmarked listings.
- **Post:** allows a user to create a listing for rent.
- **Messages:** allows a registered user to contact another registered user to inquire about a listing.

2. Functional Requirements V2

Priority 1:

1. **Unregistered Users:**
    a. Unregistered users shall be able to register
    b. Unregistered users shall be able to browse through all of the listings
    c. Unregistered users shall be able to view the public details of any listing
    d. Unregistered users shall be able to filter the listings by price, distance, and home type

2. **Registered Users:**
    a. Registered Users shall be able to do everything an unregistered user is allowed to do besides part 1a(register).
    b. Registered Users shall be able to log in and log out of an account
    c. Registered Users shall be able to post listings
    d. Registered Users shall be able to view listings they posted and related messages

3. **Site Administrators:**
    a. Site Administrators shall be able to view all Registered Users on the database
    b. Site Administrators shall be able to remove the account of a Registered User
    c. Site Administrators shall be able to view all listings on the database
    d. Site Administrators shall be able to approve or reject listings for posting
    e. Site Administrators shall be able to remove any listing on the site

Priority 2:

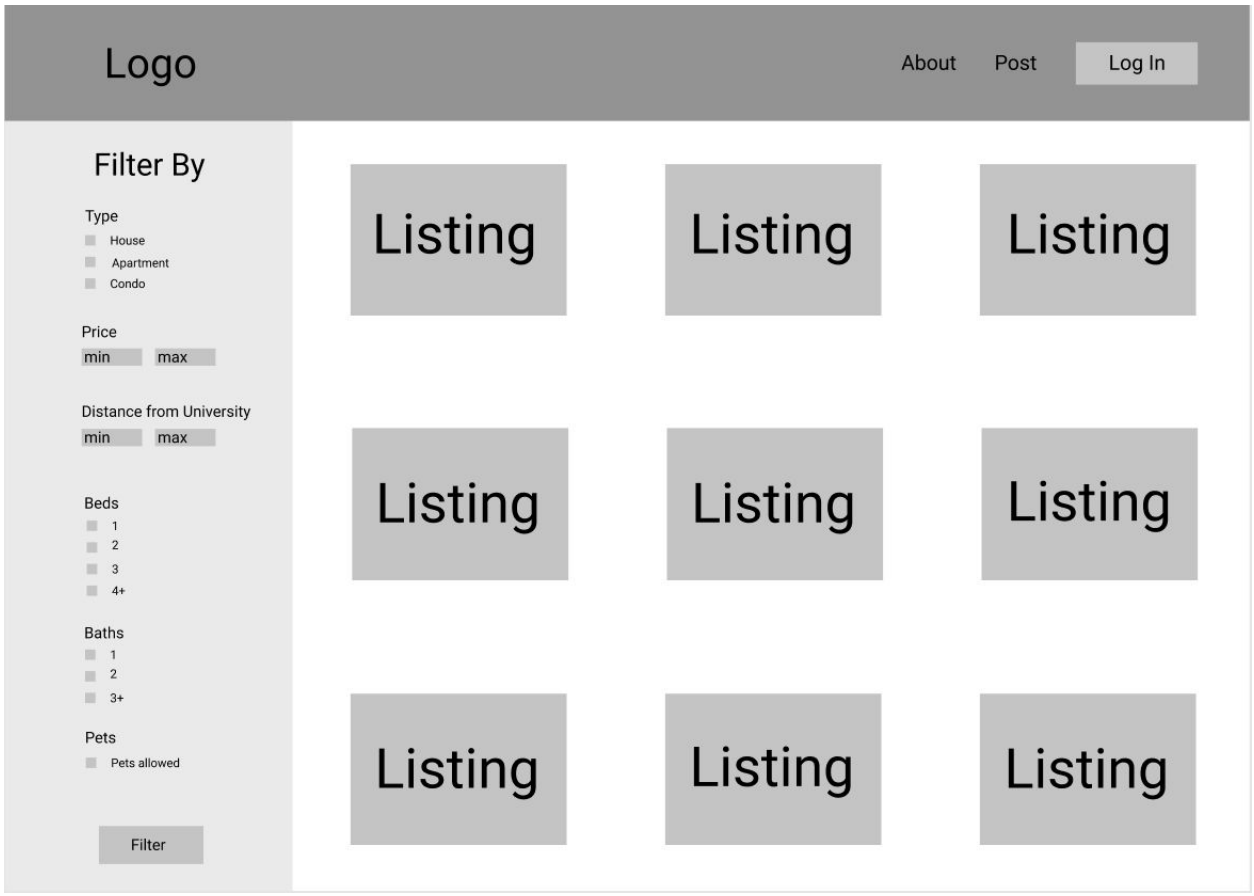
1. **Unregistered Users:**
    a. Unregistered users shall be able to filter the listings by beds, baths, and pets
2. **Registered Users:**
    a. Registered Users shall be able to edit their listings
    b. Registered Users shall be able to remove their listings
3. **Site Administrator:**

Priority 3:

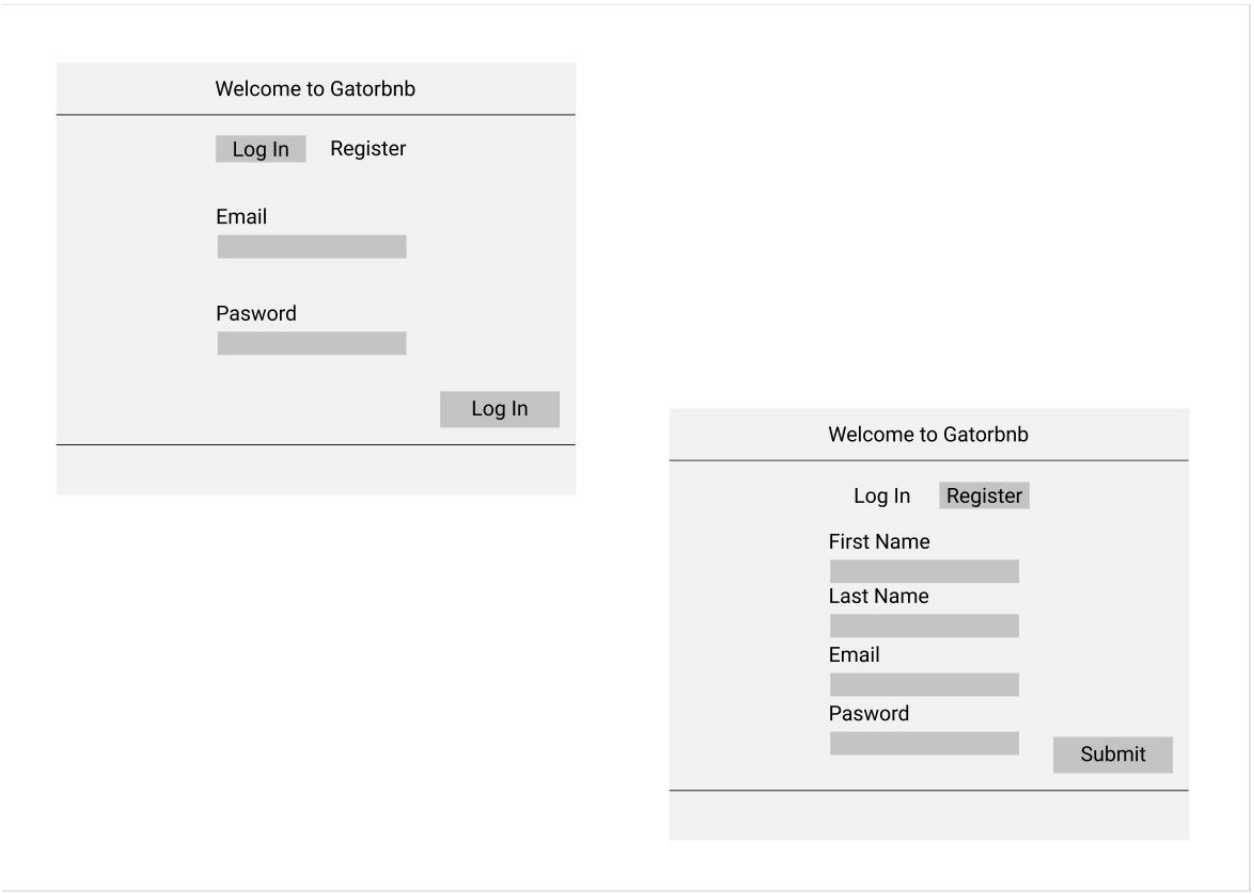
1. **Unregistered Users:**
2. **Registered Users:**
    a. Registered Users shall be able to save listings in the website
3. **Site Administrator:**
    a. Site Administrators shall be able to view all messages between Registered Users and Sellers

3. UI Mockups and Storyboard

Unregistered Home Page -

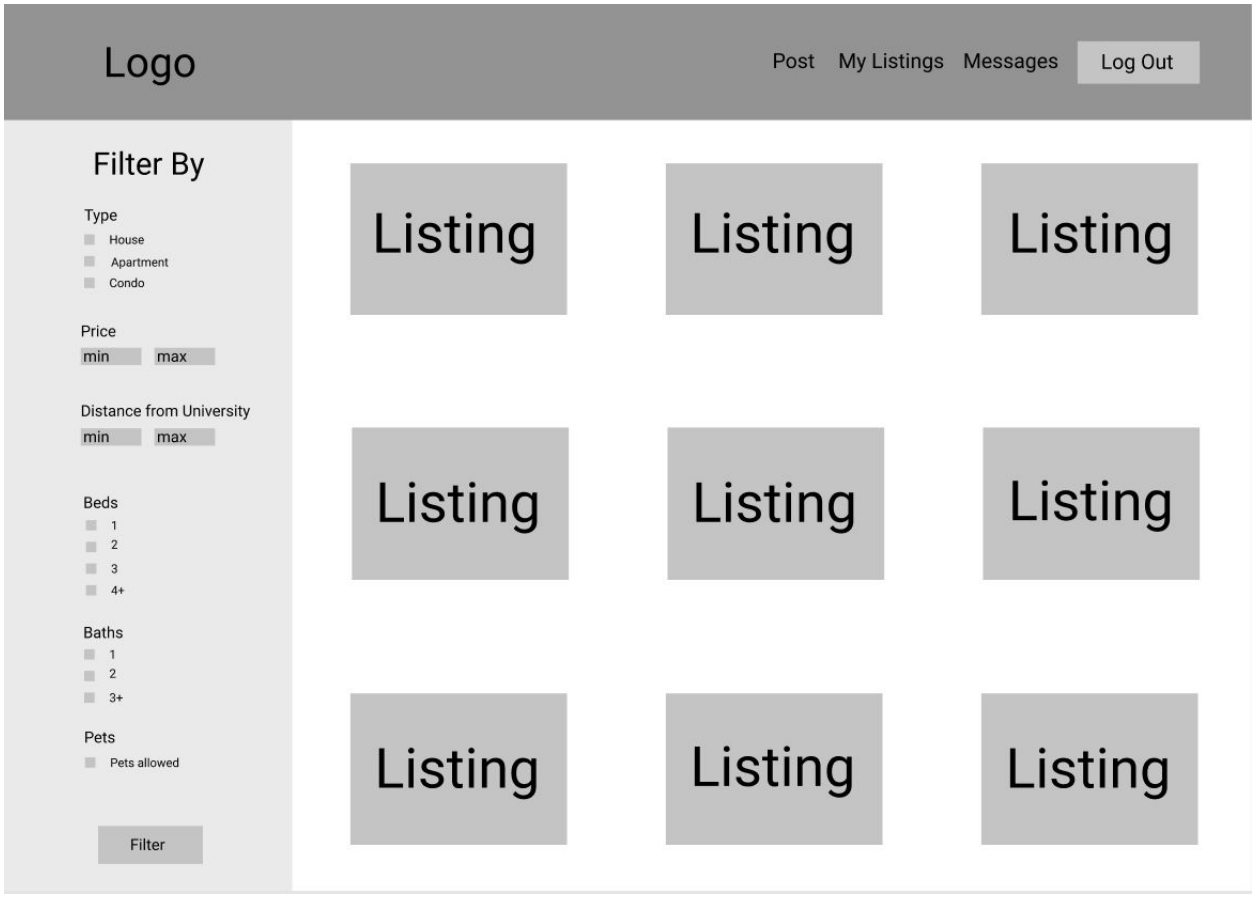


Login/Register (Pop up/Modal on Unregistered Home Page) -

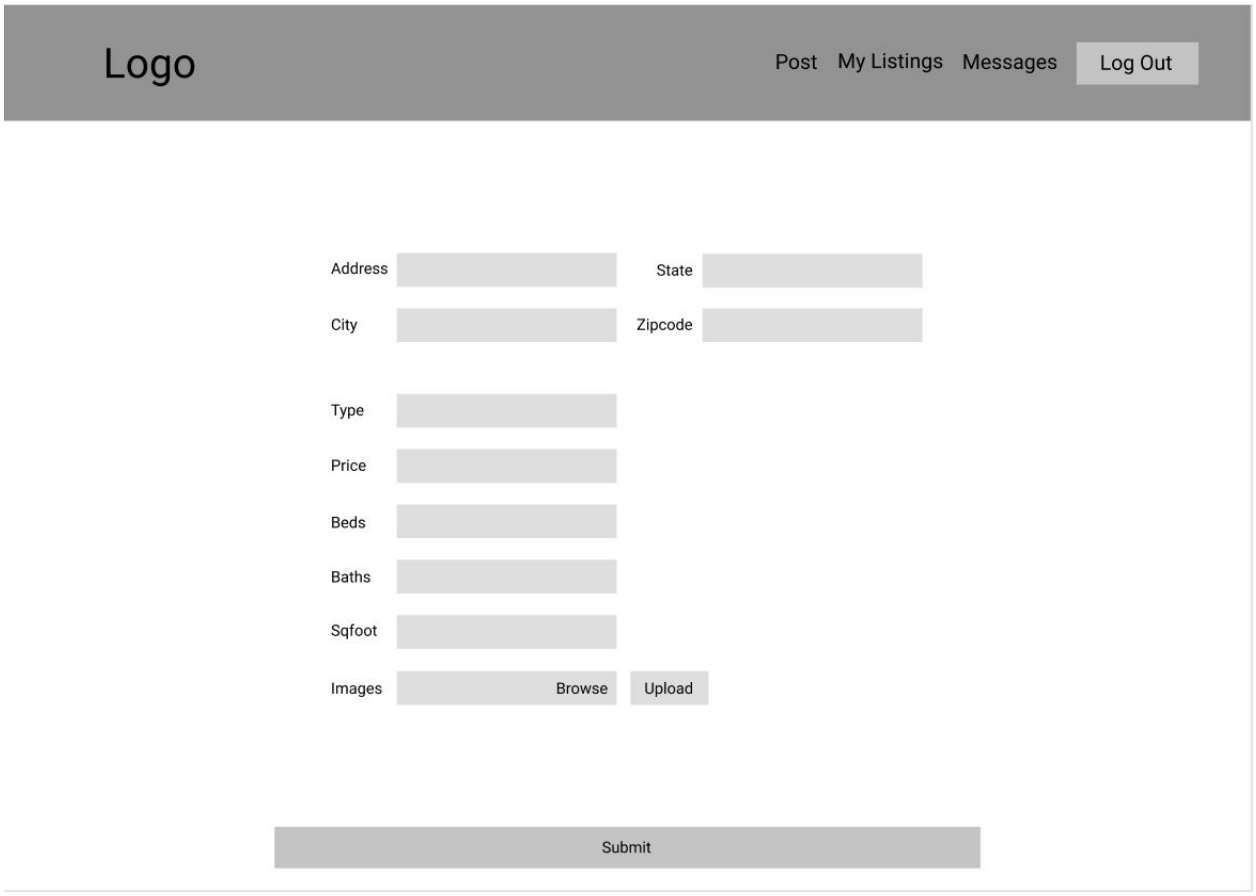


Registered Home Page

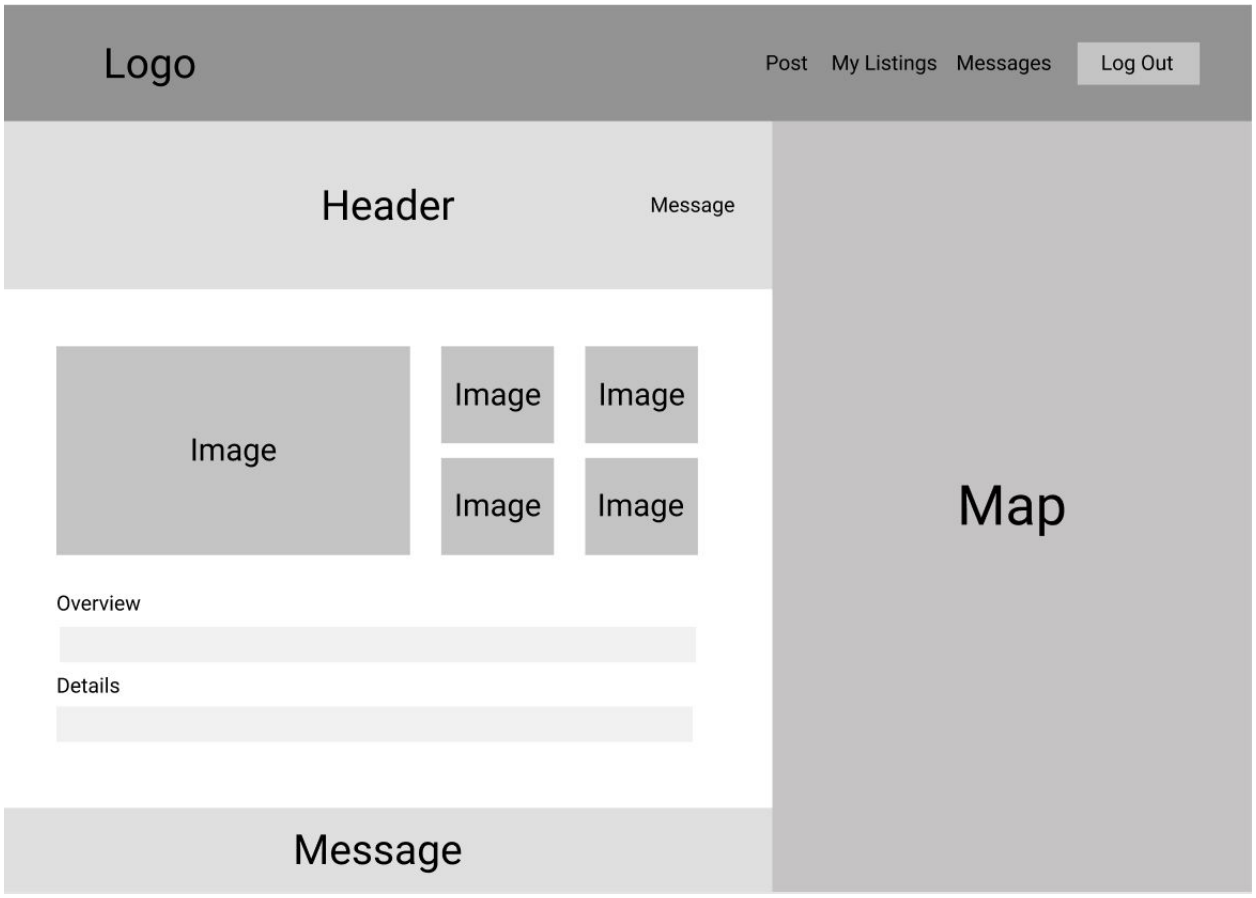
- Users shall be able to navigate to post, my listings, and messages at all times



Post -

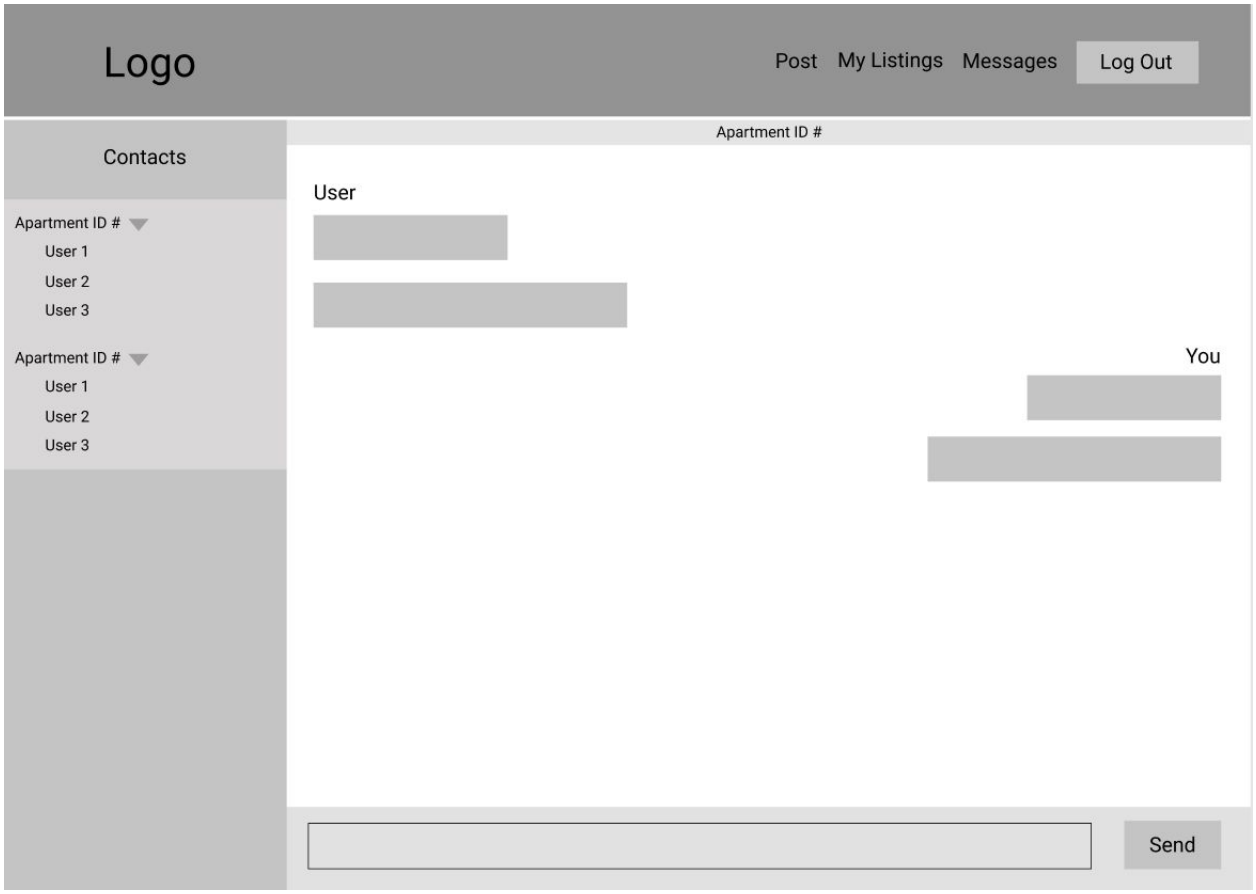

Logo
Post
My Listings
Messages
Log Out
Address
State
City
Zipcode
Type
Price
Beds
Baths
Sqfoot
Images
Browse
Upload
Submit

Listing -


Logo
Post
My Listings
Messages
Log Out
Header
Message
Image
Image
Image
Image
Image
Overview
Details
Map
Message

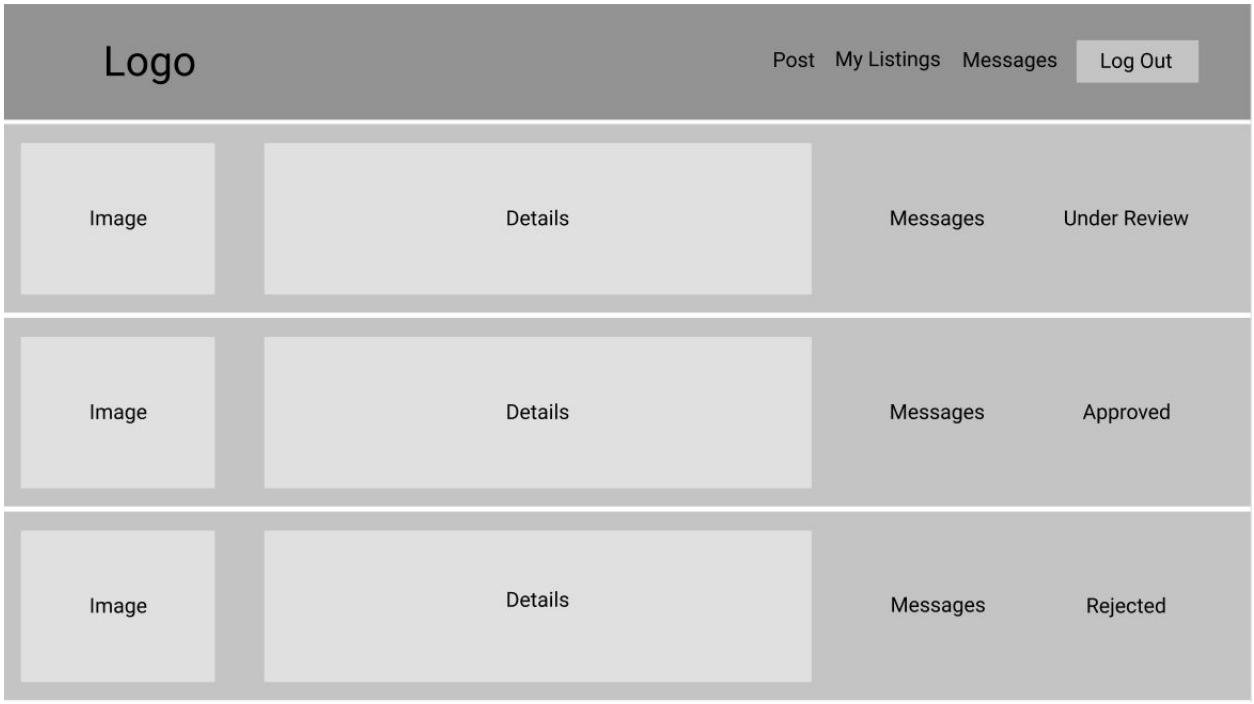
Messages

- Users shall be able to access message at all times through the navigation bar.
- Contacts and messages are grouped by apartments. They are identified by ID for now.
- Users shall be able to view all messages from users associated by apartment.



My Listing

- Users will be redirected to the corresponding apartment messages



4. High-level Architecture, Database Organization

Database Schema:

Users:

Name	Data Type	Default Value	NULL
user_id	uuid	Auto Increment	No
first_name	varchar(30)		No
last_name	varchar(30)		No
email	varchar(30)		No
password	char(32)	This will take in a MD5 encryption hash	No

Listings

Name	Data Type	Default Value	NULL
listing_id	uuid	Auto Increment	No
user_id	uuid		No
address	varchar(255)		No
state	varchar(2)		No
city	varchar(30)		No
zipcode	int(6)		No
date_created	date	Current Timestamp	No
description	text		Yes
bedroom	int(2)		No
bathroom	int(2)		No
distance	float		No
squarefoot	int(10)		No

price	decimal(7,2)		Yes
image	varchar(255)	NULL	Yes
confirm_listing	boolean	FALSE	NO

Filter_Tags_Relation:

Name	Data Type	Default Value	NULL
listing_id	uuid		No
tag_id	uuid		No

Filter_Tags:

Name	Data Type	Default Value	NULL
tag_id	uuid	Auto Increment	No
tag_name	varchar(255)		Yes

Favorite_listings:

Name	Data Type	Default Value	NULL
user_id	uuid		No
listing_id	uuid		No

Message:

Name	Data Type	Default Value	NULL
message_id	uuid	Auto Increment	No
sending_user	uuid		No
listing_id	uuid		No
message	varchar(255)		No

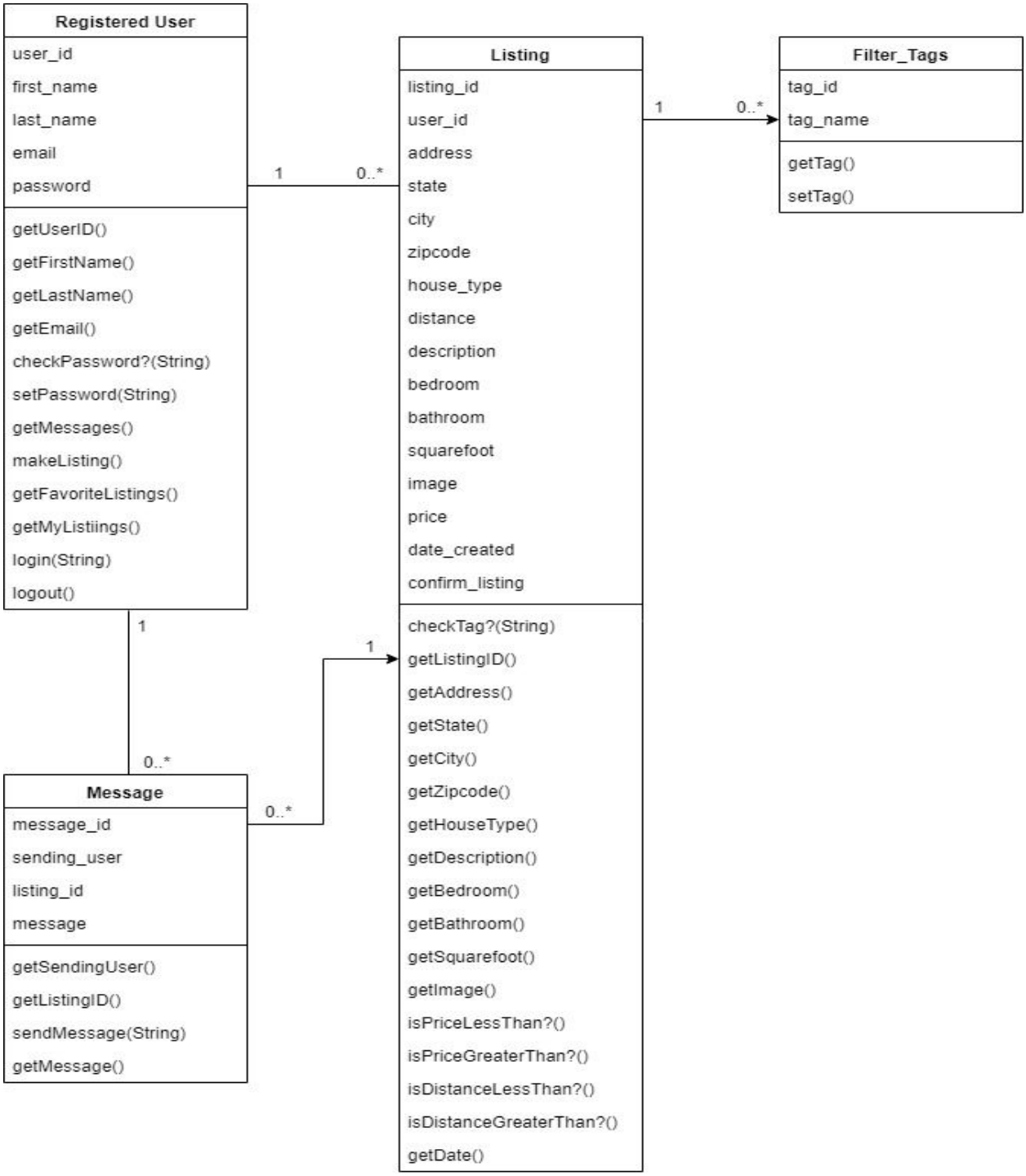
Media Storage:
All images and video/audio will be kept in file system format with the database containing the file pathname pointing to the raw media.

Search/filter Architecture and Implementation:

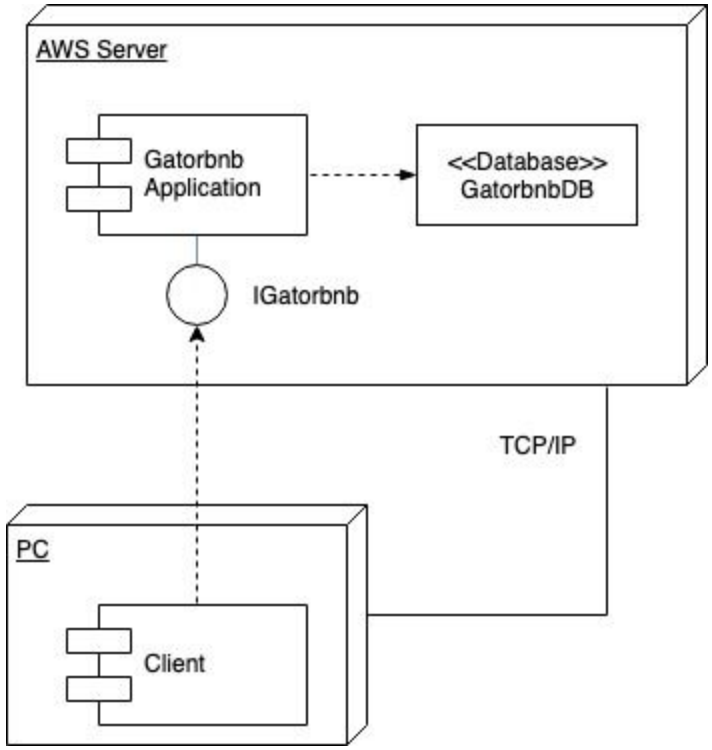
- In the event that the user has not specified any search filters, the page will display all listings organized by the most recent
- If the user enters a location like address, state, city, or zip code then all database listings will be returned with a like value to these locations. This will be implemented using the "LIKE%" operator to return all search results of the search for a specified location.
- If the user wants to filter tags like the number of bedrooms, the number of bathrooms, the type of house, or any other miscellaneous filter tags than all the listings will be sorted and filtered according to the precise equality of the columns values that the listings have in the database
- If the user enters in max price then all the listings that are less than or equal to that price will be returned and if the user enters in the minimum price then all listings greater than or equal to that price will be returned
- If the user enters in the max distance then all listings that are less than or equal to that distance from SFSU will be returned and if the user enters in the minimum distance then all listings that are greater than or equal to that distance from SFSU will be returned
- If no options are found given the user-defined search parameters than an error message will be displayed and they will be directed to search for a different category

5. High-level UML Diagrams

a) High-level UML class diagram



b) UML component and deployment diagrams



6. Key Risks:

Current skill risks the team has been getting familiar with node js/express js and javascript. Some of the team members have basic knowledge of node js and express js but most have little or no knowledge at all. To solve this risk early the team has decided to find tutorials and research on our own individual time to familiarize ourselves, along with reviewing the tutorial that the TA provided. With javascript, the team members who aren't proficient in it will practice when they have time, and .

There is a schedule risk because we have to create the Gaterbnb website within the given amount of time provided. Within the few months remaining, not only will the website need to be created but it must also be functional with the services promised. With all of us being students with other classes we must juggle familiarizing ourselves with the current skill risks, other homework from different classes, working on our website and also other extracurricular activities or our jobs. To address this issue we must work on time management, where we each dedicate enough time for each particular activity. Each day we will work a certain amount of hours on each task to ensure procrastination doesn't happen. Also with the help from Trello, we can focus on a particular task first rather than just absent-mindedly working on everything.

Technical issues may appear while working on the website such as not knowing how to implement a certain segment. With everyone working on their own part of the website, it'll be difficult trying to fix bugs that may appear. To solve this as soon as possible, we will ask each other for help after trying to find the solution by ourselves first. An answer to this would be communication. Constant updates from every team member will give everyone a sense of where they're at and how close they are to completing their own tasks. Unresolved issues will be fixed first before moving on.

Along with the technical issues, a possible legal risk that may occur is finding examples online when stuck on a certain segment. With the internet being a great source for help when stuck, it provides a risk of legal consequences. To prevent any illegal misuse, we will use the examples we find as guidance and give credit when credit is due. To prevent any copyright issues with images, we'll use what is given to us by the professor and find free images from Google's advanced filter.

7. Project Management

Starting with milestone 2, the front end and back end could start to work independently from each other because the planning stage is nearly done. There are still dependencies between the front end and the back end, however, there is far less as we move forward to this situation. The way that we managed milestone 2 is that each person would take a section that would be related to their position in the group. Another thing that was implemented was the use of Trello, since it made it easier to see where people are at and what they need to do. Tasks that involve the front end also now gets assigned to the front end lead who will distribute the work and check the status of the work. The same thing applies to the back end and the back end when it involves the structure of the project.

Milestone 3:

1. UI and functionality feedback (P1 functions only)

Home page - functions well and display looks good.

Search - good searching, smooth,and nice validation. don't have complex filter or search. Only do zip code or city search

Search results - search results look good. Enforce minute character length of search bar

Filtering - have a more descriptive name for the filters.

Search Details and maps - search results looks good, map looks good too.

Messaging/contact seller/user (if applicable) - not implemented yet at this time

Data Upload - functioning

dashboards (user, admin) - make more consistent with web page design

UI responsiveness (resize the browser) - resize looks good

Performance (e.g. display of results list) - results displays at decent speed.

2. ***Brief review of coding, github, database etc.***

Naming convention looks fairly consistent.
Git organization looks good, feature branches are being used.
GitHub commit message looks good. It seems like a few members have done little contributions.
Little to no comments, few header comments found. Coding style looks fine.
Framework followed, express and react deployed and working.
DB table looks fine, likes the idea of having photos and listings be separate tables that references each other. Table names good.
None that need attention

3. Project status – be ready to verbally explain status of the issues below

1. *Teamwork*: Teamwork is moving along. There isn't any major conflicts.
2. *Risks*: No major risk at the moment
3. Coding practices: Consistent naming convention throughout the code.

4. *Usage of proper SE code management practices*: Uses dev branch and branch off based of features.
5. *How did you address site security and safe coding practices:* Used passport js for authentication, also validation on some fields. Used pg-promise to sanitize and defend against SQL injection.
6. *Digital content* (e.g. images, video): Used file system to store photos. Is a public static directory.

4. List of P1 features committed for delivery

- Login
- Registration
- Search
    - Filter for searching type
    - Free search for zip code and city
    - At least two filters
- Search results
    - Shows results, each with image as thumbnails
- Detail page with map
- Message page
    - Simple renter to landlord. Not developed complete messaging
- Persistence posting
- User dashboard
    - listings
    - Received messaging
- Admin dashboard
    - Allow approval of listing (approve or deny)
    - Access to listings(can delete)
    - Access to users (can delete)

Milestone 4:

1. Project Summary

- Name of product: Gatorbnb
- Url: http://13.57.19.213/
- Committed Functions
    - User registration - an unregistered user shall be able to sign up and create an account
    - User Login - a registered user shall be able to log into their registered account
    - Browsing - all users shall be able to browse through the current listings
    - Filtering - all users shall be able to filter the search results by House Type, number of Bedrooms, number of Bathrooms, Price and Distance from SFSU with a drop list
    - Search- all users shall be able to use free text search for zipcode and city
    - Post - a registered user shall be able to Post a new listing, while an unregistered user may not
    - Message - a registered user shall be able to Message the user who created a listing they're interested in
    - Map - from the selected listing, all users shall be able to view a map of the housing will be available
    - Search - all users shall be able to search for a listing in the search bar
    - Admin- administrators shall be able to deny or approve listing and can delete users from the database

2. Usability Test Plan

Test Objectives:

The selected function being tested is the search function. Gatorbnb relies heavily on the search function because it allows one of our many intended audiences, SFSU students, to be able to search for a place to live. By utilizing the search function, every user whether they're registered or unregistered are able to find listings based on the information they've entered within the search bar. The search bar searches the parameters of a city or zip code and sends

them to a listings page. Since the search function is being relied upon, it is important to ensure it is functioning correctly to create a greater user experience. Users who have difficulties using the search function on Gatorbnb will be unsatisfied, equating to a lost in profit in a real-world situation. So in order to create a well functioning website that satisfies every user, the main function being used (search) should be the function being tested to prevent any difficulties or problems.

Test Description:

Url: http://13.57.19.213/

System Setup:

The system is setup where testers are asked to use our search function. Testers will be instructed to use a computer, regardless if it's Windows or Linux, that has access to the internet. They may choose to use either Google Chrome or Safari as the browser.

Starting Point:

The starting point would be the home page of the website. After their initial search, they will be brought to another page which lists the listings based on their search results. After that, one more search will be given, to not only test the search function of the landing page, but also the listing page's search function.

Intended Users:

The intended users for the test are people who use a computer for an adequate amount of time, between the ages of 18-45. This will give us feedback from many different perspectives on whether the search function is easy to use or if it was difficult.

What's Measured:

For this test, the level of difficulty of using the search function is being tested. Testers should have a clear understanding of what searching is and how to search. What is also tested is if testers input the correct searchable arguments and whether invalid searches are handled correctly. Lastly responsiveness is also being tested. Testers will be able to observe whether they got listings based on their arguments and how long it took for them to receive their results.

Usability Task Description:

Testers are given the task to search for listings within San Francisco. They may either search for all listings within San Francisco by typing “San Francisco” in the search bar, or by a zip code within San Francisco for a more specific result.

Questionnaire:

Please, check one

	Strongly Disagree	Disagree	Neither agree or disagree	Agree	Strongly Agree
1) The parameters of search was clearly stated					
2) Search was easy to use and understand					
3) My search results were correct					

Comments:

3. QA Test Plan

Test Objectives:

- To ensure the search function operates correctly where the searched for listings are being displayed
- Results are obtained whether the tester were able to accomplish the given instructions by marking pass or fail.

Hardware and Software Setup

1. Testers will access a computer connected to the internet and use either Google Chrome or Safari
2. Testers will go to http://13.57.19.213/
3. Url: http://13.57.19.213/

Feature to be tested:

- The Search function will be the feature to be tested since it's one of the most utilized features. By typing a city or a zip code, listings will be displayed based on their input.

QA Test Plan

QA Test Number	QA Test Title (search by:)	QA Test Description	QA Test Input	QA Expected Output	QA Results Chrome	QA Results Safari
1	City	Type San Francisco on the search bar	San Francisco	Get 15 results of listings that are located in San Francisco	Pass	Pass
2	Zip Code	Type 94122 on the search bar	94122	Get 2 listings with the zip code of 94122	Pass	Pass
3	Case sensitivity	Type SaN fRanCiSc o on the search bar	SaN fRanCiSc o	Get all 15 results of listings in San Francisco	Pass	Pass
4	Whitespa ce	Type sanfrancis co in the search bar	sanfrancis co	Return the message “No matching results. Please check your spelling or enter a valid ZIP code”	Pass	Pass

4. Code Review

a.

From: ple2@mail.sfsu.edu
To: aownyeong@mail.sfsu.edu

Hello Anthony,

Thanks for letting me review your code, here are my thoughts.

For the variable naming convention, it is camel case just like how you did. The style of the coding is to separate the subfunctions if it is possible to pull out the make it as modular as possible. Separates the files by the functionality of the functions within in and by the routes / object that it is related to.

- Peter

b. Copy and paste email:

To: ple@mail.sfsu.edu
From :aownyeong@mail.sfsu.edu

Hello Peter, please Review my code.

```
/**
 * get function for search in listings route
 * @params queue | the search queue to search db
*/
router.get('/search/', (req, res) => {
 let { queue } = req.query
 const { type, beds, baths, priceMax, distanceMax } = req.query


 console.log( queue, type, beds, baths, priceMax, distanceMax )
```

```
          queue = queue.replace("+", " ")
          // create queue for database call
          let dbQueue = null
          if( !isNaN(queue) ) {
            dbQueue = `SELECT * FROM listings WHERE zipcode=$1`
          } else {
            dbQueue = `SELECT * FROM listings WHERE
LOWER(city)=LOWER($1)`
          }
          // database call for the search results
          db.any(dbQueue, [queue])
          .then(data => {
            console.log(data)
          // filter
            // @TODO move this function to module
            const filteredListings = data.filter(listing => {
              console.log(beds + " " + listing.bedroom)
              if( ( type && listing.housing_type !== type) )
                console.log('invalid type')
              else if( ( beds && listing.bedroom < beds) )
                console.log('invalid bedroom')
              else if( ( baths && listing.bathroom < baths) )
                console.log('invalid bathroom')
              else if( ( priceMax && listing.price > priceMax) )
                console.log('invalid price')
              else if( ( distanceMax && listing.distance > distanceMax) )
                console.log('invalid distance')
              else
                return listing
```

```
  })
  console.log('fitered stuffs here')
  console.log(filteredListings)

  console.log( "LENGTH: ", filteredListings.length)

  res.send(filteredListings)
 })
 .catch(error => {
  console.log(error)
  res.sendStatus(204)
 })
})
```

Comments: In line comments

function is robust.

A bit long, so need to pull out functions to make it more readable.

5. Self-check on best practices for security

- For our application, the major assets we are protecting is the name, email, images, and the user's password. Personal information like user data is stored solely in the users table in the database and the password specifically will be encrypted via bcrypt, a password hashing function, to protect this personal information from being stolen and then exploited for malicious use. Images are stored in a file system with hashed path. Also the database cannot be controlled without the routes we provide. Images are stored with the relative path name to their file placement and not the absolute. An internal function handles the pathname to point to images.
- All the passwords in the database are hashed into a 64-bit char using the bcrypt password hashing function. We are encrypting the password in the database with 8 rounds of salt. This ensures the protection of the passwords in the users database against rainbow tables and someone trying to access password data maliciously.

- For input validation, we require that nothing over 40 alphanumeric only characters will be accepted in the search input validator. The code we used for the search bar input is as follows:

```
<Input
   style={inputStyle}
   size='large'
   action={{ icon: 'search' }}
   name="search"
   placeholder='Enter a city or ZIP code'
   value={this.props.queue}
   onChange={this.props.changeQueue}
   maxLength="40"
/>

getListings = (value) => {
    Var regex = /^[a-z\d-_\s]+$\
    if(regex.test(value)) {
        axios.get(`/api/listing/search/${value}`)
    }
}
```

6. Self-check on Non-functional specs

a. 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**
b. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers **DONE**
c. Selected application functions must render well on mobile devices **DONE**
d. Data shall be stored in the team's chosen database technology on the team's deployment server. **DONE**
e. No more than 50 concurrent users shall be accessing the application at any time **ON TRACK**
f. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
g. The language used shall be English. **DONE**
h. Application shall be very easy to use and intuitive. **DONE**

i. Google analytics shall be added. **ON TRACK**
j. No e-mail clients shall be allowed. **DONE**
k. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. **DONE**
l. Site security: basic best practices shall be applied (as covered in the class) **DONE**
m. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator **ON TRACK**
n. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **DONE**
o. 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. (Important so as to not confuse this with a real application). **ON TRACK**

4. Product Screen Shots

PRINTED ON DIFF STITS. . .

5. Screenshot of key DB tables

Table "public.listings"

Column	Type	Collation	Nullable	Default
listing_id	uuid		not null	uuid_generate_v4()
user_id	uuid		not null	
address	character varying(255)		not null	
city	character varying(255)		not null	
state	character varying(255)		not null	
zipcode	integer		not null	
date_created	timestamp with time zone		not null	now()
description	text			
bedroom	integer		not null	
bathroom	integer		not null	
squarefoot	integer		not null	
price	double precision		not null	
confirmation	boolean		not null	false
distance	double precision			
housing_type	character varying(255)		not null	

Indexes:

Table "public.users"

Column	Type	Collation	Nullable	Default
user_id	uuid		not null	uuid_generate_v4()
firstname	character varying(255)		not null	
lastname	character varying(255)		not null	
email	character varying(255)		not null	
password	character varying(255)		not null	
is_admin	boolean		not null	false

Indexes:

Table "public.tag"

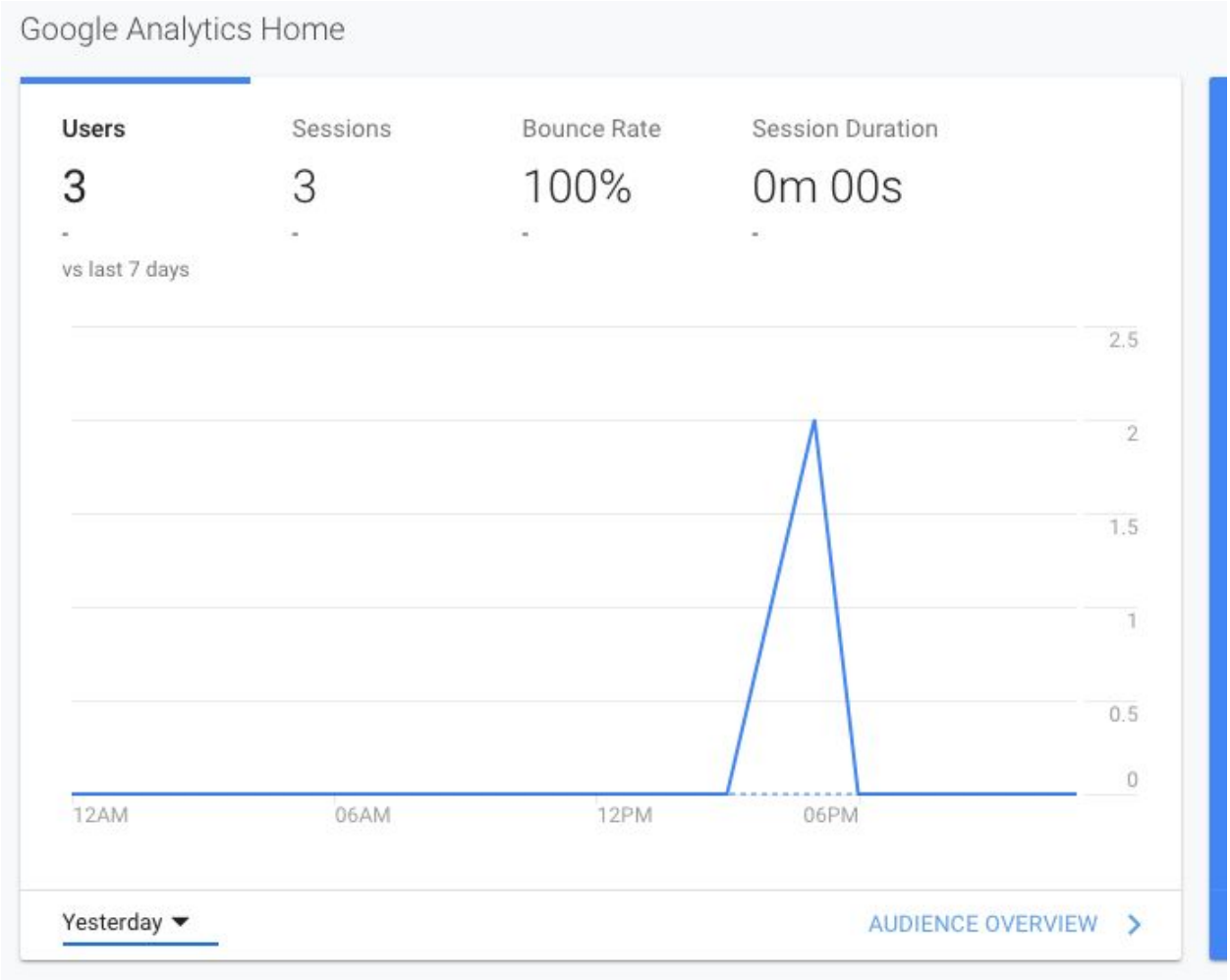
Column	Type	Collation	Nullable	Default
tag_id	uuid		not null	uuid_generate_v4()
tag_name	character varying(255)		not null	

Indexes:

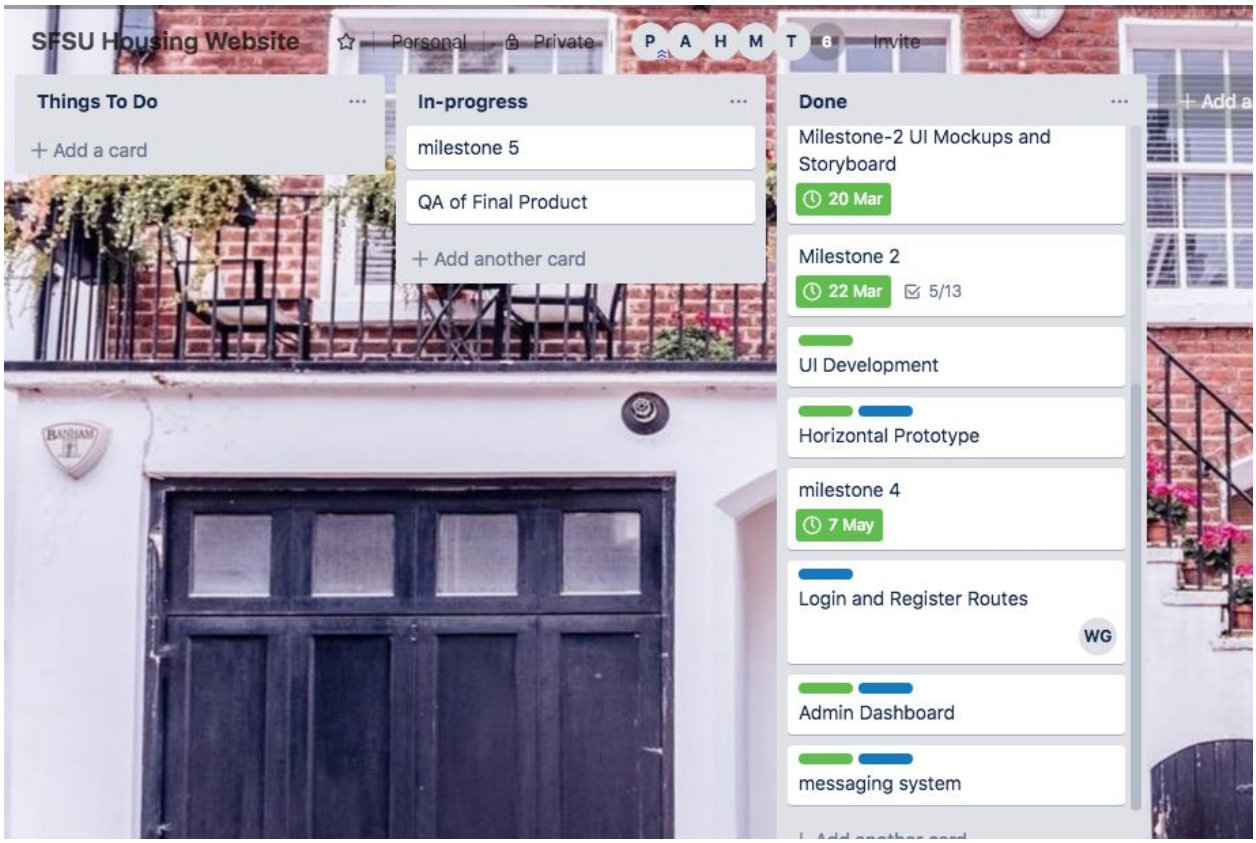
```
                                Table "public.message"
   Column   |           Type           | Collation | Nullable |      Default
------------+--------------------------+-----------+----------+--------------------
 message_id | uuid                     |           | not null | uuid_generate_v4()
 owner_id   | uuid                     |           | not null |
 message    | character varying(255)   |           | not null |
 timestamp  | timestamp with time zone |           | not null | now()
```

```
                    Table "public.session"
 Column |           Type           | Collation | Nullable | Default
--------+--------------------------+-----------+----------+---------
 sid    | character varying(255)   |           | not null |
 sess   | json                     |           | not null |
 expire | timestamp with time zone |           | not null |
```

6. Google analytics



7. Screenshot of Task Management



8. Team member contribution

(Raw copy from email thread)

- **Peter Vinh Le <ple2@mail.sfsu.edu**

  Hello team,

  This is the thread for Team member contribution. Follow the following format:

  Peter Le:

  Contribution:

  - team lead

  - database design

  - database migration / implementation

  - manage / setting up production server and dev environment

  - set up vertical prototype / design

  - API endpoints / queries

  - API authentication

  - API sessions

  - photo file system / upload photos

  Github commits:

  - 98 commits

- **Hoang Dong <hoangmdong@gmail.com>**

  Hoang Dong,

  Contribution

  -Post Listing

  -Milestone contributor

  -Brought snacks

  Github commits:

  -1

Thanks!

- Anthony Owyeong <anthony.owy@gmail.com>

Anthony Owyeong,

Contribution:

- Database design

- Environment tester

- Disclaimers and welcoming messages

- Milestones contributor

Github commits:

- 6

- Jawynson Pranz Tizon Sunga <Jsunga1@mail.sfsu.edu>

Jay Sunga:

Contribution:

- front end lead

- landing page, home, filter, search, navbar, results, listing details, dashboard, and all the admin pages

- connected all the pages to the API

- fixed all the bugs/issues on the front end

- managed all the routes

Github commits:

- 63 commits

- Wesley Goldfisher <wgoldfisher1@gmail.com>

Wesley Goldfisher:

Contribution:

- **Backend team lead**
- **Database design**
- **Database implementation**

- **API endpoints**
- **API authentication / registration**
- **Admin authentication**
- **Update Admin routes**

Github commits:

- **11 commits**

- Tanya Ting Yan Wong <twong6@mail.sfsu.edu>

Tanya Wong:

Contribution:

- front end member

- UI design

- create an account page

GitHub commits:

- 2 commits

- Mehi 707 <mehiberi@gmail.com>

Mehi Ledwon:

Contribution:

- front end team member

- UI design

- website logo

- login page

Github commits:

- 3 commits

9. Post Analysis:

A main challenge that occured while working on this project was working in a huge team. Many of our previous team work contained at most four people in a team but for this project it was a team of seven. With a big team, communication can get confusing because everyone has a different schedule. When communicating via group messaging, slack or discord not everyone would be available at the same time so messages can be lost or unread. To address this problem in the future, having a certain time everyday—for example 8:30 p.m every week night—will allow every team member to speak if they had any issues or just as a status update to keep everyone informed on their progress.

Conflicting schedules was another main issue. With a team size larger than what we're familiar with, having meetings where everyone was available became difficult. Each team member had their own schedule whether it's because of school, work or even personal causing meeting dilemmas. To approach this particular issue, asking for each member's schedule and choosing a time where everyone can meet would be useful. Then both the backend and frontend teams can meet on their own and eventually collaborate with each other.

Management became one of the main issues when creating this project—before using Trello. With the help of Trello, everyone was able to focus on their own tasks. Everyone was able to see who was currently still working on their own task or if they finished it; this gave everyone a rough idea on what still needed to be done. In the future, having each team member use a task managing application from the very beginning can solve the problem of not knowing what to do next. Trello also helped with prioritizing what should be done first. In our project, our functions were listed as P1, P2 and P3 telling us what functions to work on first. If the backend was working on the login function, the frontend would work on the login page allowing the different teams work together creating a more smooth workflow.