Final Project for Software Engineering

CSC 648/848 Section 02 Spring 2019

Milestone 5

Team 08

Cheryl Fong
::
cfong8@mail.sfsu.edu

Jerry Wong

Chirag Agarwal

Harold Pedroso

Johanna Hadgu

Sharan Lao

Ermias Haile

http://13.52.51.45/

1. PRODUCT SUMMARY

1.1 Our product GatorRent is an accommodation-search application that is intended for the use of the students of San Francisco State University (SFSU) to find housing and for landlords that are looking for renters alike. This application is important as on-campus housing at SFSU is extremely limited and expensive thus making it difficult for both domestic students and international students to find accommodation within their budget and preference.

The GatorRent application will allow a user (users of the GatorRent application includes landlords looking for renters and students looking for housing) to either post houses, apartments, rooms that they are renting out, or for the user to filter through the currently available listings posted on the application. Our application will allow users to find listings based on important factors to consider as a student of San Francisco State University such as distance from the university, rent price. Thus, making GatorRent uniquely tailored for students and landlords as compared to other applications currently on the market and we will market it through societies at San Francisco State University to local students and to incoming international students.

GatorRent is accessible through the following link: http://54.183.44.255/

1.1 Committed Function List

1.1.1 **Users can:**

1.1.1.1 Login or register themselves a user of the website.

1.1.1.2 Easily search and browse for listings.

1.1.1.3 Post a place for rent.

1.1.1.4 Filter their search results by price, rent-type, price, postcode, if pets are allowed and if the rent-type is shared.

1.1.1.5 View additional details about any listing, including a map displaying the location of the listing.

1.1.1.6 Message any landlord about a specific listing and view all received messages on their dashboard.

1.1.1.7 Contact the Admin regarding any concerns through the Contact Us page.

1.1.1.8 **Unique feature:** users can filter listings respective to their distance from SFSU.

1.1.2 **We ensured the proper safety precautions when users are visiting and using this site. Passwords will be encrypted.**

TABLE OF CONTENTS

Software Engineering CSC 648/848 Section 02 Spring 2019

Milestone 1

Team 08

Cheryl Fong
::
cfong8@mail.sfsu.edu

Jerry Wong

Chirag Agarwal

Harold Pedroso

Johanna Hadgu

Sharan Lao

Ermias Haile

REVISION TABLE		
Date	**Version Number**	**Description**
2/28/19	1.0	First draft
3/6/19	1.1	Second draft
3/11/19	1.2	Third Draft

FROZEN STATE

1. EXECUTIVE SUMMARY

The launch of GatorRent, an accommodation-search application is intended for the use of the students of San Francisco State University (SFSU) to find housing and for landlords that are looking for renters alike. This application is important as on-campus housing at SFSU is extremely limited and expensive thus making it difficult for both domestic students and international students to find accommodation within their budget and preference.

The GatorRent application will allow a user (users of the GatorRent application includes landlords looking for renters and students looking for housing etc.) to either post houses, apartments, rooms that they are renting out, or for the user to filter through the currently available listings posted on the application. Our application will allow users to find listings based on e.g. distance from the university, rent price. The design of this application will take into account of the technical experience/familiarity of the user by implementing a simple and straightforward design with large buttons and appropriately sized writing along with the use of images and colours to create a more visual-oriented design for ease of use and accessibility. Thus, making GatorRent uniquely tailored for students and landlords as compared to other applications currently on the market.

Our student start-up team is comprised of current computer science students of SFSU, including domestic and international students of different backgrounds. Our personal experiences with finding off-campus and on-campus housing have contributed largely to the ideation of the intended purpose and design of the GatorRent application with the potential range of users in mind.

2. PERSONAE AND USE CASES

Personae

2.1 Kelly

Kelly is a recently admitted college student from Maine, who's attending SFSU in the fall as a freshman. She needs to find a place to live before classes begin. However, she's finding it hard to find an apartment that's close to campus and accommodating for her as a student. Luckily, she stumbles upon GatorRent**.**

2.2 Andrew

Andrew is an ICU nurse at Zuckerberg San Francisco General. After working 6 years in the field, and experienced the successes and struggles of his career, he decides to pursue his Master of Science in Nursing at San Francisco State University. Recalling the earlier struggles of finding housing in his undergraduate years, Andrew decides to use GatorRent to find an apartment.

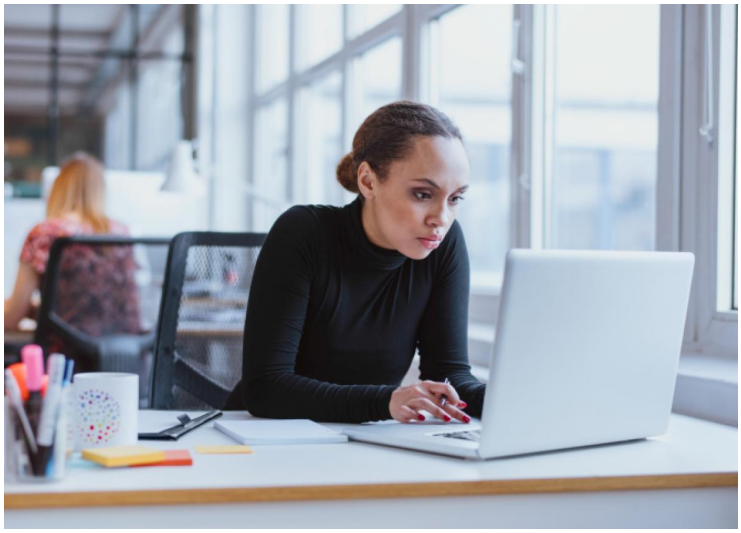
2.3 Max

Max is a junior at San Francisco State University, who recently got accepted to study abroad in Stockholm, Sweden. However, since he is still in the middle of his long-term lease, he realizes that he cannot fulfill his current lease's terms and needs to break it. Max was recently informed by his **landlord** that he needs to find a sufficient replacement. Since Max's

place is relatively close to SFSU, he considers using GatorRent, a site he had used in his sophomore year of university, as a way to find his replacement.

2.4 Jordan

Jordan is a part of the **Admin** team at GatorRent. She recently transferred from a local startup and has spent a sufficient amount of time familiarizing herself with the site.



Use Cases

2.1 Non-registered User (Kelly)

As a **non-registered user**, Kelly is presented with a variety of apartments **listings** available for rent and is able to **browse** through them. Kelly wants to search for apartments that are in close proximity to public transportation, campus and also within her price range. She **filters** her results by **zip code**, **distance,** and **price** and is presented with several apartment listings that are according to her specifications. Now that she found an acceptable apartment, she has solved her problem with GatorRent.

2.2 Registered User (Andrew)

Upon opening the website, Andrew notices that several **listings** are far beyond his price range and are not convenient enough to support his commute to work and school. Therefore, he filters his results according to the **distance** from local transit and **price.** Additionally, GatorRents allows him to accommodate for his cat allergies by **filtering** out renters that allow **pets** to reside in the premises. After finding one apartment that satisfies his conditions, Andrew creates an account and becomes a **registered user,** where he is required to accept the **terms and conditions** of the site. Once this is done, he is able to message the **landlord** and secure an apartment with the help of GatorRent.

2.3 Landlord (Max)

Having logged into GatorRent, Max is able to find options that allow him to post his apartment as a potential place to rent. Once he provides a description of the place, pricing, and images of the **property**, Max is prompted to register or log in before completing the post. Then, Max submits his post for **admin** approval.

2.4 Admin (Jordan)

After logging in to the GatorRent dashboard, Jordan is greeted with a well-organized data list to include **listings** in need of approval and user-abuse reports. As part of her job, during a session of listing evaluation, Jordan comes across a **listing** that did not include photos of the **property**. Jordan denies the listing, contacting the lister and specifying that he'll need to modify his **listing** before it becomes live on the site. After approving several **listings**, she also notices that another **landlord** has been reported by 3 registered users. Having recalled that this specific **landlord** had already received a warning several months prior, Jordan immediately terminates the account.

3. DATA ITEMS AND ENTITIES

Database Name:

Gatorrent-db - This database will contain the following tables below:

The Database Tables:

Table Name	Fields/Columns										
Users	id	username (unique)	password	email							timestamp
Admins	id	username (unique)	password	email							timestamp
Listings	id	user-id (foreign-key)	description		rent-type (house, apartment, room)	rent-type-size	price (per month)	distance from campus	postcode	isPet (bolean - are pets allowed?)	isShared (boolean - is the rent-type shared?)	timestamp
Markers	id	address	lat (lattitude)	long (longitute)	listing-id (foreign-key)						timestamp
Messages	id	subject	body	user-id-sender (foreign-key)	user-id-recipient (foreign-key)	isRead (boolean - is the message read/opened?)					timestamp

Click here to View Image 1.1v Above

Website Meta-Data:

user-session — To keep track of the user's state when using the website.
user favourites — To keep a list of URLs of the user's saved listings.
user-preference — To keep track of user's preferences.
activity-log — A log of entries added or removed from databases.

Glossary:

User/Users	A user of the site. Could be registered or non-registered. If registered, will be stated as “Registered user(s)” otherwise “Non-registered user(s)”.
Registered User	A user who has signed up on the website, i.e. the user’s username, password, and email address will be contained in the Users Table. This user has full access to all user functionalities, such as sending messages to other users.
Non-registered User	A user of the website who did not sign up, i.e. this user does not have an entry on the Users Table. This user is only allowed to browse listings on the website.
Admin/Admins	Website staff who are site usage maintainers, i.e. responsible for handling the appropriateness of user activities by disapproving or deactivating users. This user has full access to all user functionalities.
Student	A user of the site, specifically looking/browsing for listings.
Landlord	A user of the site, specifically looking to rent out their property**.** Has the intention of creating listings.
Filter	The ability/functionality to filter results from the Listings Table.
Listing	An entry on the Listings Table, which includes information on a posted property such as rent-type, its description, rent-type size in square feet, the rent price per month, distance from campus, if pets are allowed, if the rent-type is a shared or not shared occupancy and postcode.
rent-type	Refers to the type of space that is intended to be rented out (e.g. house, room, apartment, etc.).

Registration	An entry added to the Users Table, containing user data during the registration process, such as their username, password, and email address.
Drop-pins	Refers to an entry on the Markers Table, as a visual marker  on Google Maps
Terms and Conditions	The website staff is not responsible for interactions between users outside the realms of the website.

4. FUNCTIONAL REQUIREMENTS

4.1 Non-registered Users:

4.1.1 Shall be able to see/browse all **listing** results on the website, excluding the description of **listing** results.

4.1.2 Shall have the option to register themselves as a **user** of the website.

4.1.3 Shall have the option to use the message function to inform **admins** about site related issues, such as inappropriate images or untrustworthy listings.

4.1.4 Shall be able to **filter listing** results by lowest or highest price, price range, postcode, distance from campus, rent-type, rent-type price, if pets are allowed, and if the rent-type is shared.

4.2 Registered Users:

4.2.1 Shall have access to all the functionalities as a **non-registered user**.

4.2.2 Shall be able to see/browse all **listing** results on the website, excluding the description of **listing** results.

4.2.3 Shall accept all **terms and conditions** during the registration process.

4.2.4 Shall be able to publish a property for rent, i.e. this process creates a **listing** entry with the required data fields as stated in the **Listings Table**.

4.2.5 Shall be able to communicate with other **users** via the website message function.

4.2.6 Shall have the option to view their **username**.

4.2.7 Shall have the option to view and change their **email address**.

4.2.8 Shall have the option to view and change their **user** preferences.

4.2.9 Shall receive notifications and the option to turn it off for new messages sent to them.

4.2.10 Shall have the option to receive notifications for newly added **listing** entries that match their price range.

4.2.11 Shall have the option to provide their price range preference on their **user** profile.

4.2.12 Shall have the option to save and unsave desired **listing** results.

4.2.13 Shall have the option to view all saved **listing** results.

4.2.14 Shall have the option to share a **listing** on social media i.e. on **Facebook** and **Twitter** via their respective **API's**.

4.2.15 Shall have the option to view nearby transit methods via the **Google Transit API**.

4.2.16 Shall have the option to view directions to any location they prefer, with SFSU as a quick selection option, from a **listing** result's location address as a starting point.

4.2.17 Shall have the option to rate any other user, if and only if the two **registered users** (the one rating and the one being rated) have at least sent a message to each other via the message function.

4.2.18 Shall have the option to view the number of **registered users** who have their preferred price preference equal to the **listing** price that this registered user has created.

4.2.19 Shall be able to see available **listing** results as **drop-pins** of locations on the Google Maps, via the **Google Maps API**.

4.3 Administration:

4.3.1 Shall have access to all the functionalities of a **registered user**.

4.3.2 Shall approve or disprove **listing** results.

4.3.3 Shall view website **activity log** and act accordingly to unusual website behaviour.

4.3.4 Shall deactivate **users** for their misuse of the website.

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 email 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 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:

	Trulia	Zillow	Apartments	Nestpick	GatorRent
Listings	+	+	+	+	+
Expanded Apartment Info Page	+	++	+	+	+
User Navigation Bar	++	+	+	-	+
Filter	++	+	+	-	+
Search	+	+	+	+	+
Distance from SFSU filter	-	-	-	-	+
Distance from Transportation options	++	-	-	-	+
Map	++	++	+	-	+
- Not Implemented		+ Implemented		++ Implemented Well	

Our website shall comprise of listings, expanded apartment info pages, user navigation bar, and filters as our basic feature set because they are prevalent features amongst real estate listings and rental properties. By polishing within this feature set, it will allow us to gain a market share over our smaller competition and make strides in becoming a larger company.

By incorporating SFSU specific filters to include distance from campus as well as number of nearby SFSU recommended transportation options, GatorRent is focusing on providing resourceful features that will benefit our niche market: SFSU students. Additionally, these features will allow us to dominate within our niche market, therefore potentially increasing our user-base considerably every semester.

7. HIGH-LEVEL SYSTEM ARCHITECTURE AND TECHNOLOGIES

Framework	Python Flask Bootstrap MySQL-Connector-Python
APIs	Google Maps Facebook Twitter Google Transit
Tools	Docker AWS S3 Bucket
System	Ubuntu 18.04 LTS MySQL 5.7 Server Container Python Flask Web Server Container
Deployment Platform	Amazon Web Services (AWS)
Supported Browsers	Desktop and mobile browsers i.e. Chrome versions: 72.0 and 68.0 Firefox versions: 65.0 to 61.0 Safari versions: 12.0 to 11.0

8. TEAM

Position	Names
Team Leader	Cheryl Fong
Front-End Lead	Jerry Wong
Back-End Lead	Chirag Agarwal
Document Master	Johanna Hadgu | Sharan Lao
Team Member Front End	Ermias Haile | Sharan Lao
Team Member Back End	Harold Pedroso | Johanna Hadgu
GitHub Master	Harold Pedroso

9. CHECKLIST

Status	To Do Item
DONE	Team found a time slot to meet outside of the class. - Every Tues & Thurs at 10am-12pm
DONE	GitHub master chosen. - Master Harold Pedroso
DONE	Team decided and agreed together on using the listed SW tools and deployment server.
ON TRACK	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.)

Software Engineering CSC 648/848 Section 02 Spring 2019

Milestone 2

Team 08

March 22th, 2019

Local

Cheryl Fong
cfong8@mail.sfsu.edu

Jerry Wong

Chirag Agarwal

Harold Pedroso

Johanna Hadgu

Sharan Lao

Ermias Haile

REVISION TABLE		
Date	**Version Number**	**Description**
3/20/2019	1.0	Date submitted for review
3/21/2019	1.1	Revised after Feedback

FROZEN

1. DATA DEFINITIONS V2

User/Users	A user of the site. Could be registered or non-registered. If registered, will be stated as “Registered user(s)” otherwise “Non-registered user(s)”.
Registered User	A user who has signed up on the website, i.e. the user’s username, password, and email address will be contained in the Users Table. This user has full access to all user functionalities, such as sending messages to other users and posting listings.
Non-registered User	A user of the website who did not sign up, i.e. this user does not have an entry on the Users Table. This user is only allowed to browse listings on the website.
Admin/Admins	Website staff who are site usage maintainers, i.e. responsible for handling the appropriateness of user activities by disapproving or deactivating users. This user has full access to all user functionalities.
Student	A user of the site, specifically looking/browsing for listings.
Landlord	A user of the site, specifically looking to rent out their property**.** Has the intention of creating listings.
Filter	The ability/functionality to filter results from the Listings Table.
Listing	An entry on the Listings Table, which includes information on a posted property such as rent-type, its description, rent-type size in square feet, the rent price per month, distance from campus, if pets are allowed, if the rent-type is a shared or not shared occupancy and postcode.
rent-type	Refers to the type of space that is intended to be rented out (e.g. house, room, apartment, etc).

Registration	An entry added to the Users Table containing user data during the registration process, such as their username, password, and email address.
Drop-pins	Refers to an entry on the Markers Table, as a visual marker  on Google Maps
Terms and Conditions	The website staff is not responsible for interactions between users outside the realms of the website.

2. FUNCTIONAL REQUIREMENTS V2

2.1 Priority 1

Unregistered Users:

2.1.1 Shall be able to see/browse and filter all **listing** results on the website.

2.1.2 Shall have the option to register themselves as a **user** of the website.

2.1.3 Shall have the option to use the message function to inform **admins** about site related issues, such as inappropriate images or untrustworthy listings via the contact button on the side footer.

2.1.4 Shall be able to **filter listing** results by lowest or highest price, price range, postcode, distance from campus, rent-type, rent-type price, if pets are allowed, and if the rent-type is shared.

Registered Users:

2.1.5 Shall have access to all the functionalities as a **non-registered user**.

2.1.6 Shall be able to see/browse all **listing** results on the website, including the description of **listing** results.

2.1.7 Shall accept all **terms and conditions** during the registration process.

2.1.8 Shall be able to publish a **listing** for rent, i.e. this process creates a **listing** entry with the required data fields as stated in the **Listings Table**.

2.1.9 Shall be able to communicate with other **users** via the website message function.

2.1.10 Shall be able to check for messages related to posted listings.

Administration:

2.1.11 Shall have access to all the functionalities of a **registered user**.

2.1.12 Shall approve or disprove **listing** results before they go live on the site.

2.1.13 Shall view website **activity log** and act accordingly to unusual website behaviour.

2.1.14 Shall deactivate **users** for their misuse of the website.

2.2 Priority 2

Registered Users:

2.2.1 Shall have the option to view nearby transit methods via the **Google Transit API**.

2.3 Priority 3

Registered Users

2.3.1 Shall have the option to view their **username**.

2.3.2 Shall have the option to view and change their **email address**.

2.3.3 Shall have the option to view and change their **user** preferences.

2.3.4 Shall receive notifications and the option to turn it off for new messages sent to them.

2.3.5 Shall have the option to receive notifications for newly added **listing** entries that match their price range.

2.3.6 Shall have the option to provide their price range preference on their **user** profile.

2.3.7 Shall have the option to save and unsave desired **listing** results.

2.3.8 Shall have the option to view all saved **listing** results.

2.3.9 Shall have the option to return to a draft version of their **listing**.

2.3.10 Shall have the option to label a **listing** as currently occupied, under maintenance and vacant/open.

2.3.11 Shall have the option to share a **listing** on social media i.e. on **Facebook** and **Twitter** via their respective **API's**.

2.3.12 Shall have the option to view directions to any location they prefer, with SFSU as a quick selection option, from a **listing** result's location address as a starting point.

2.3.13 Shall have the option to rate any other user, if and only if the two **registered users** (the one rating and the one being rated) have at least sent a message to each other via the message function.

2.3.14 Shall have the option to view the number of **registered users** who have their preferred price preference equal to the **listing** price that this registered user has created.

2.3.15 Shall be able to see available **listing** results as **drop-pins** of locations on the Google Maps, via the **Google Maps API**.

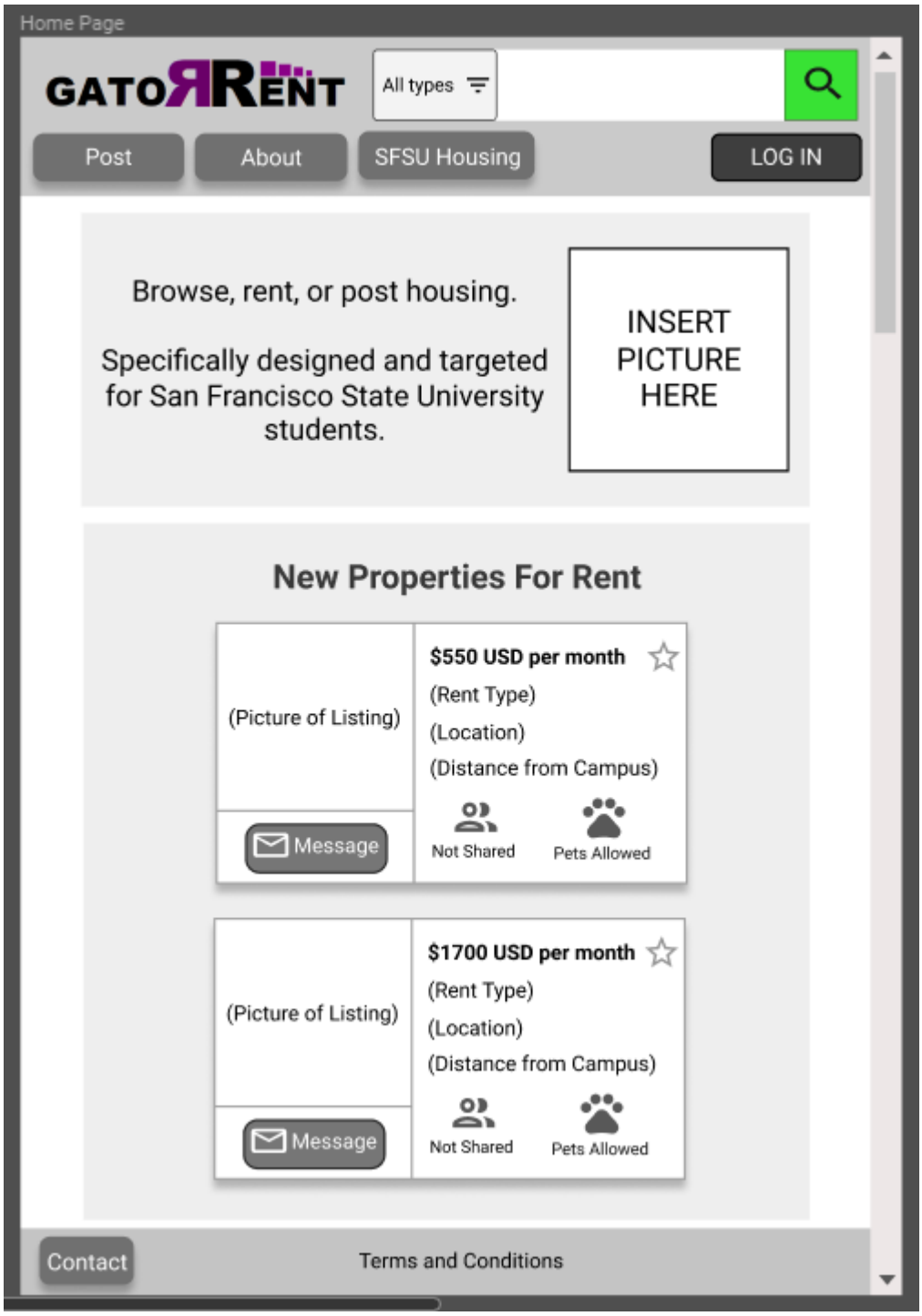
2.3.16 Shall be able to edit or modify their send **messages.**

2.3.17 Shall be able to receive read receipts when their message has been seen by the receiver (isRead function).

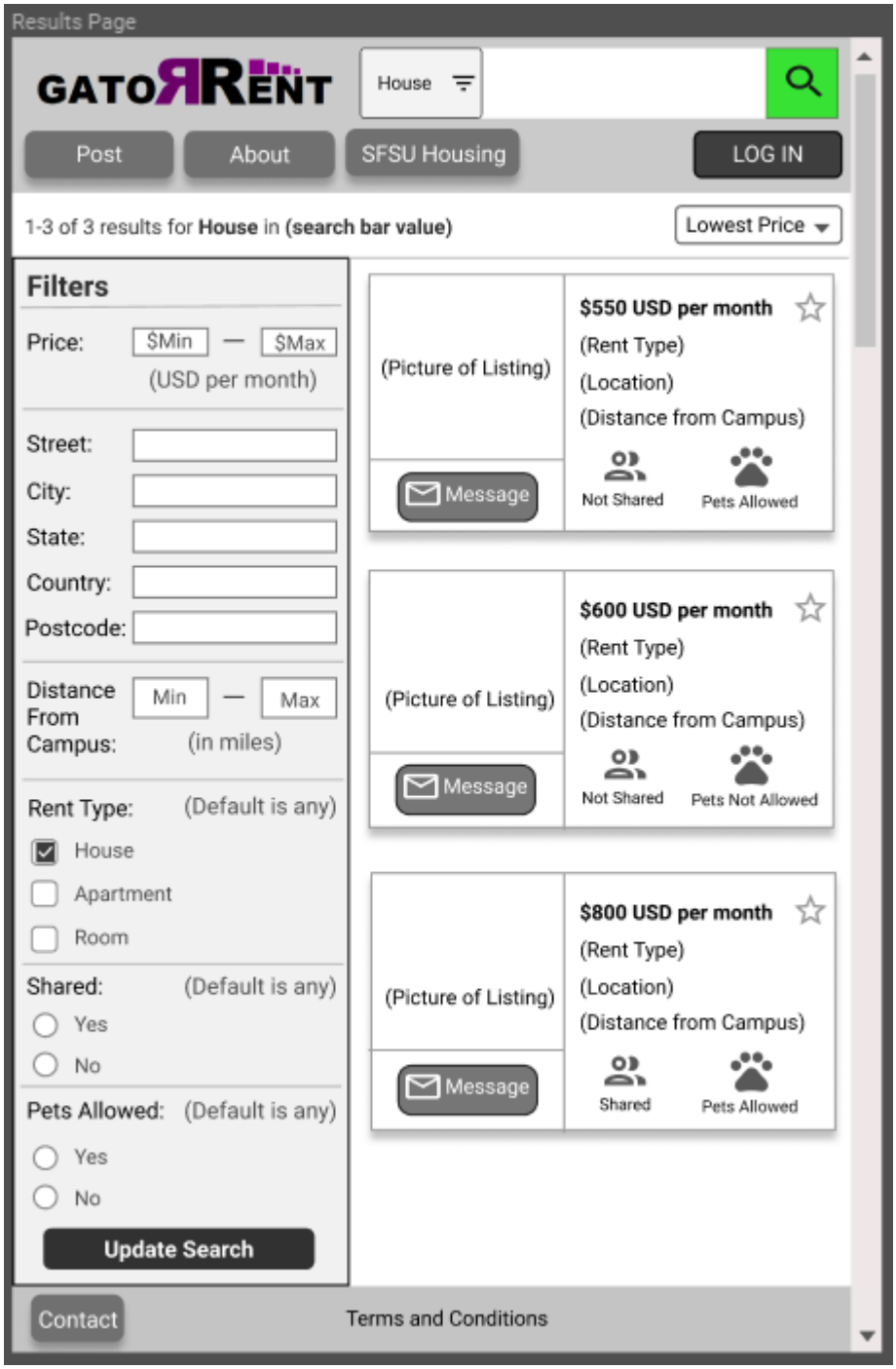
3. UI MOCKUPS AND STORYBOARDS

[Link to Interactive Figma UI Mockups]

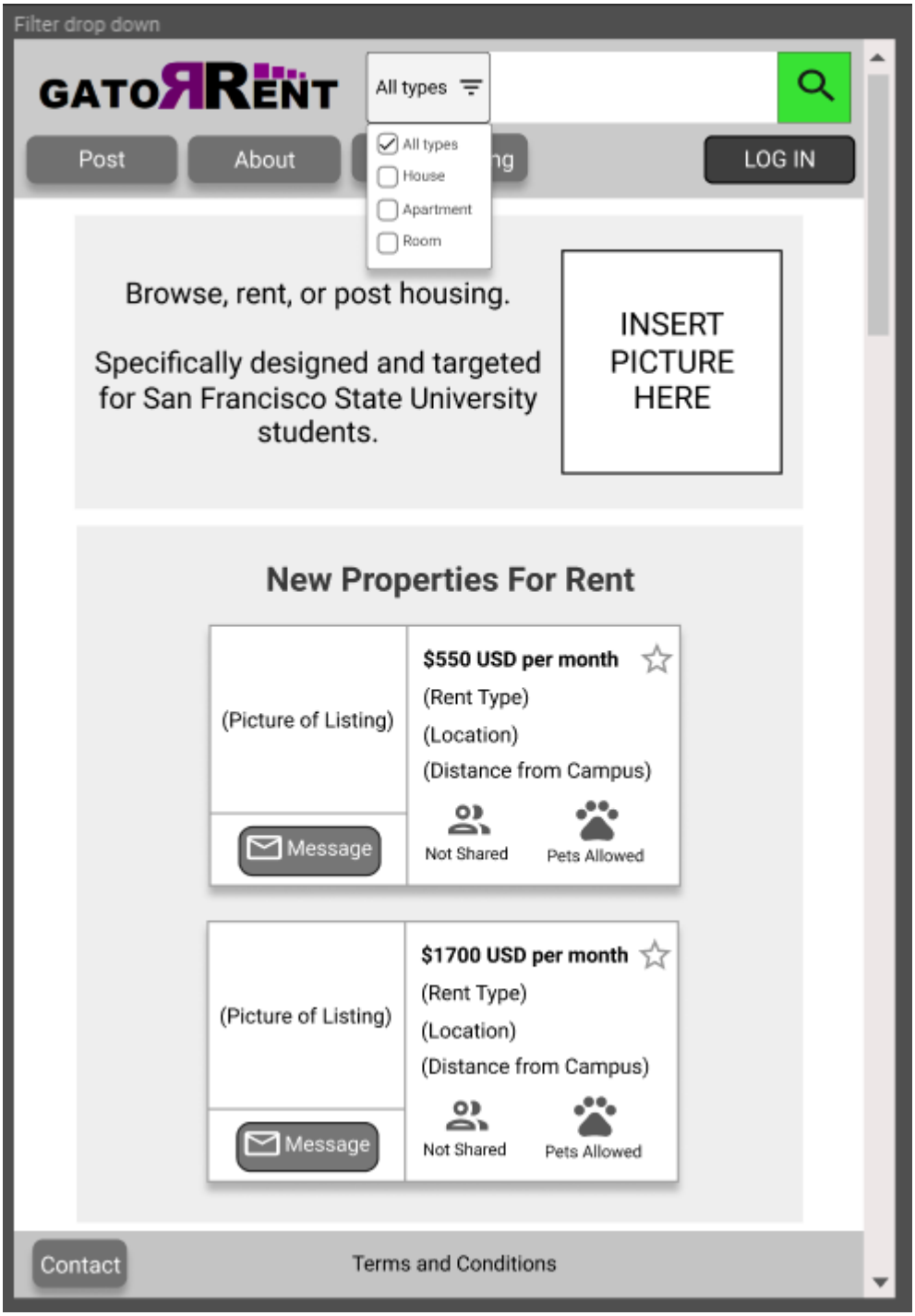
3.1 UI Mockups



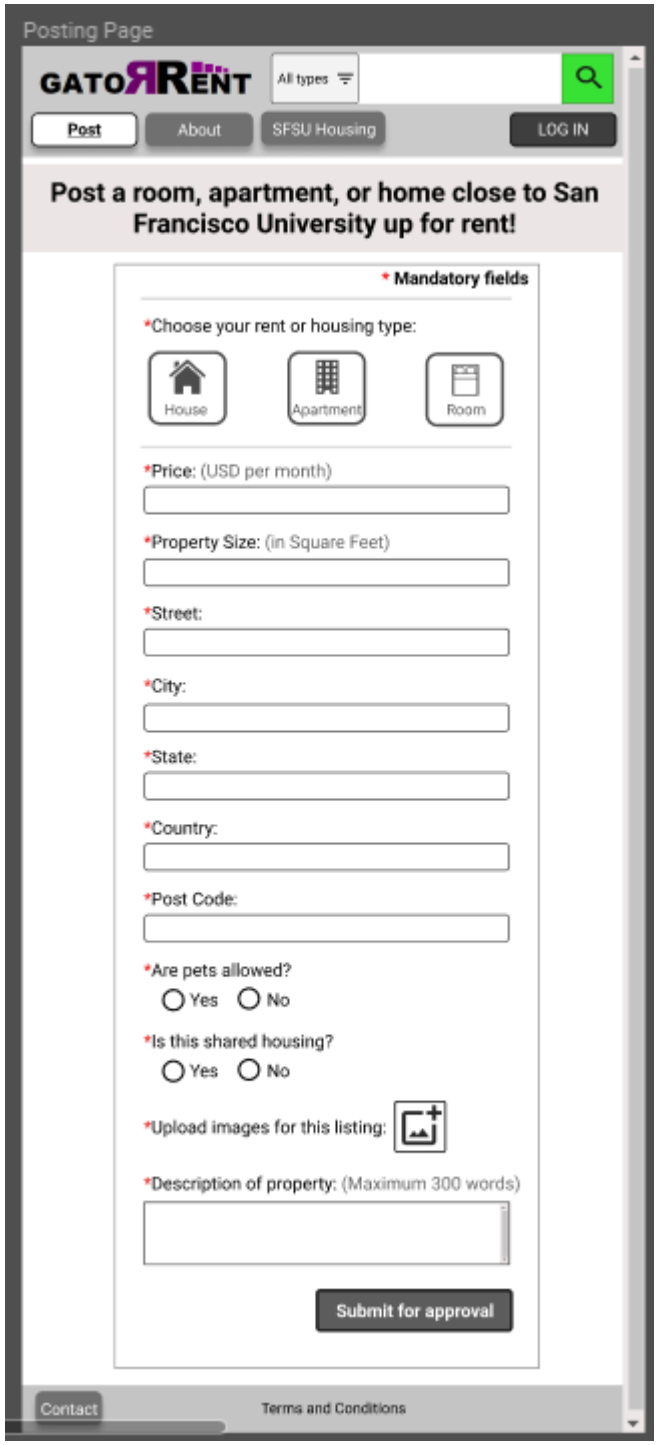
Home Page



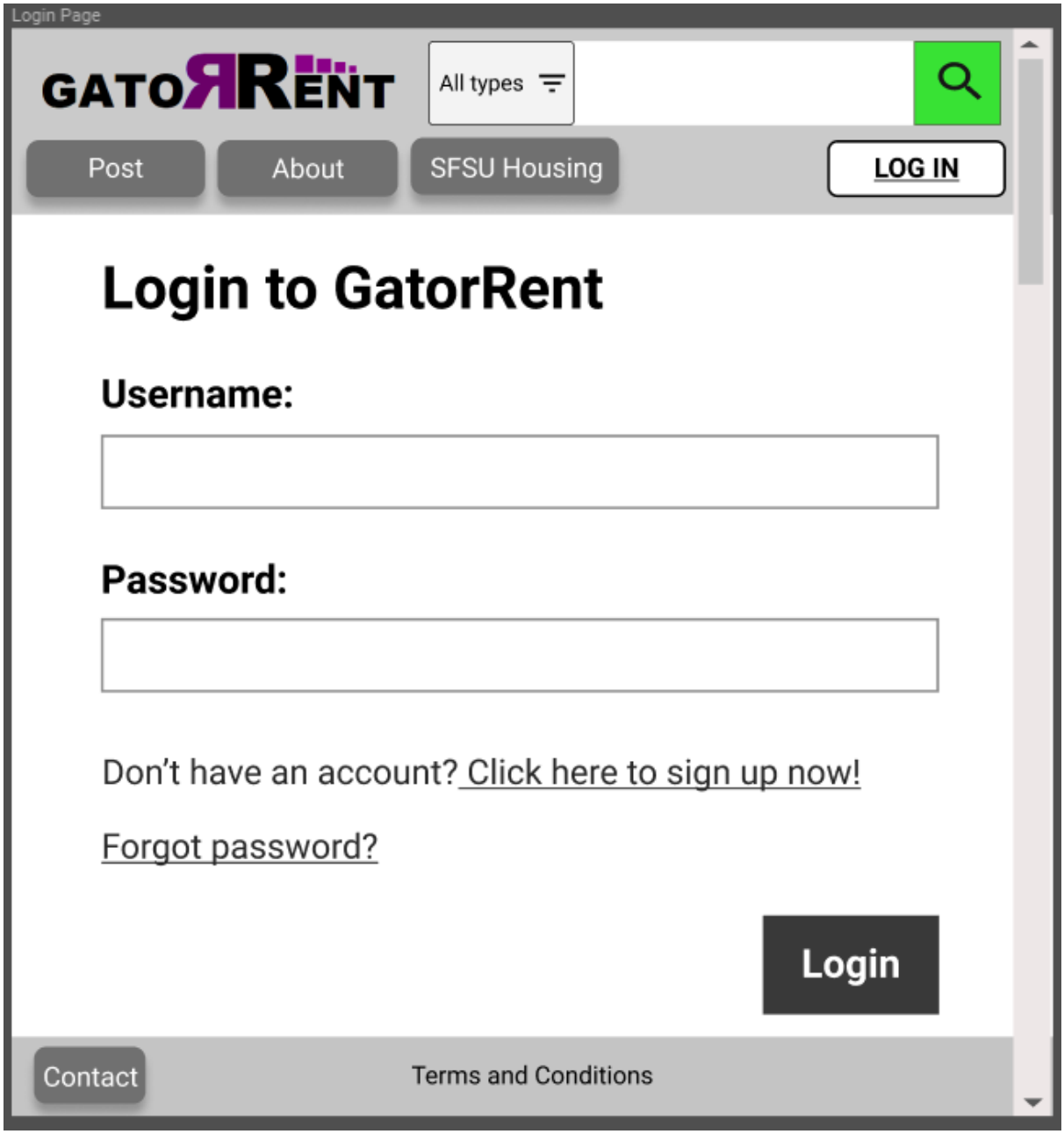
Browsing Page


Filter drop down
GATORRENT
All types
All types
House
Apartment
Room
Post
About
LOG IN
Browse, rent, or post housing.
Specifically designed and targeted for San Francisco State University students.
INSERT PICTURE HERE
New Properties For Rent
(Picture of Listing)
$550 USD per month
(Rent Type)
(Location)
(Distance from Campus)
Message
Not Shared
Pets Allowed
(Picture of Listing)
$1700 USD per month
(Rent Type)
(Location)
(Distance from Campus)
Message
Not Shared
Pets Allowed
Contact
Terms and Conditions

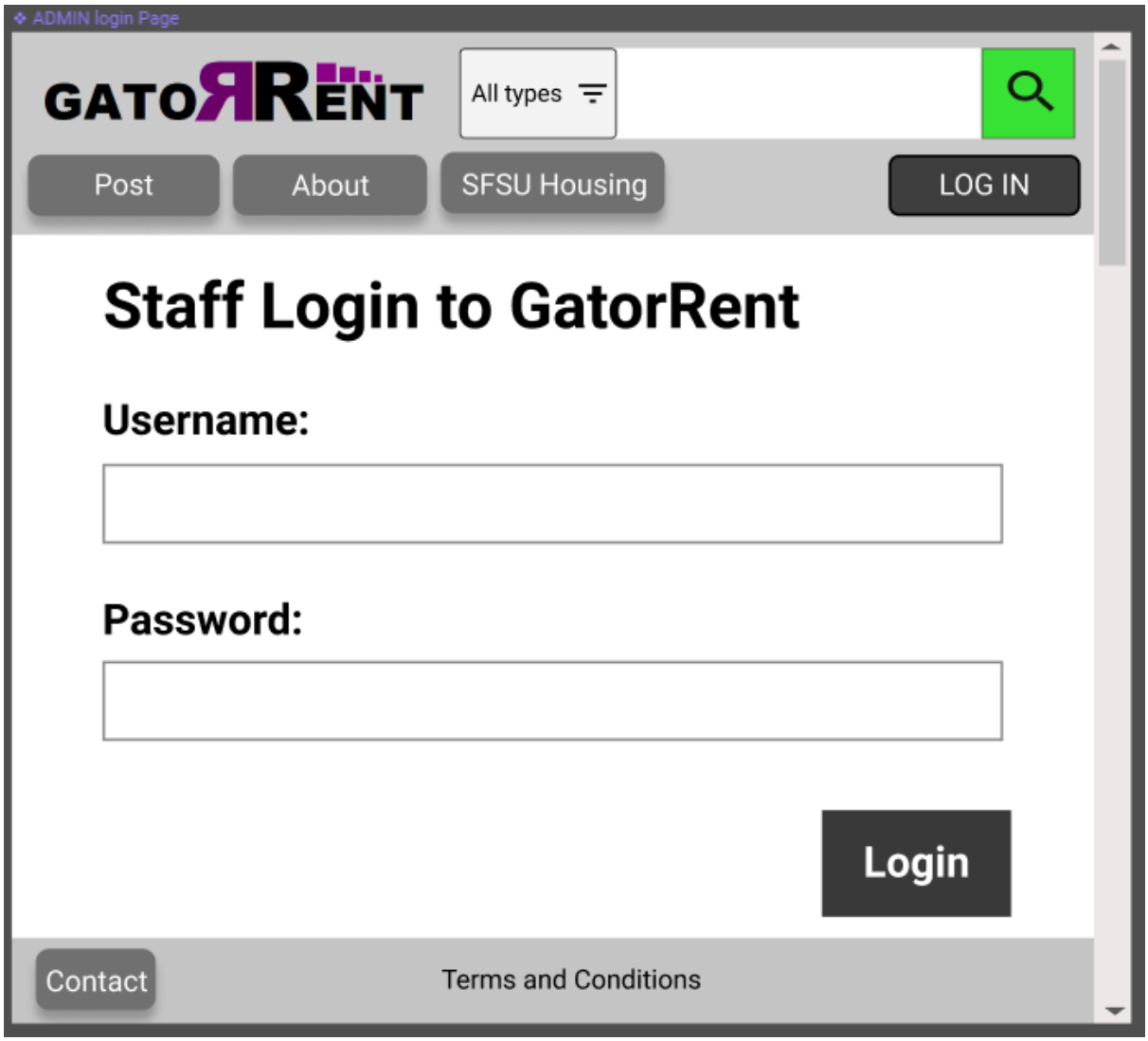
Filter



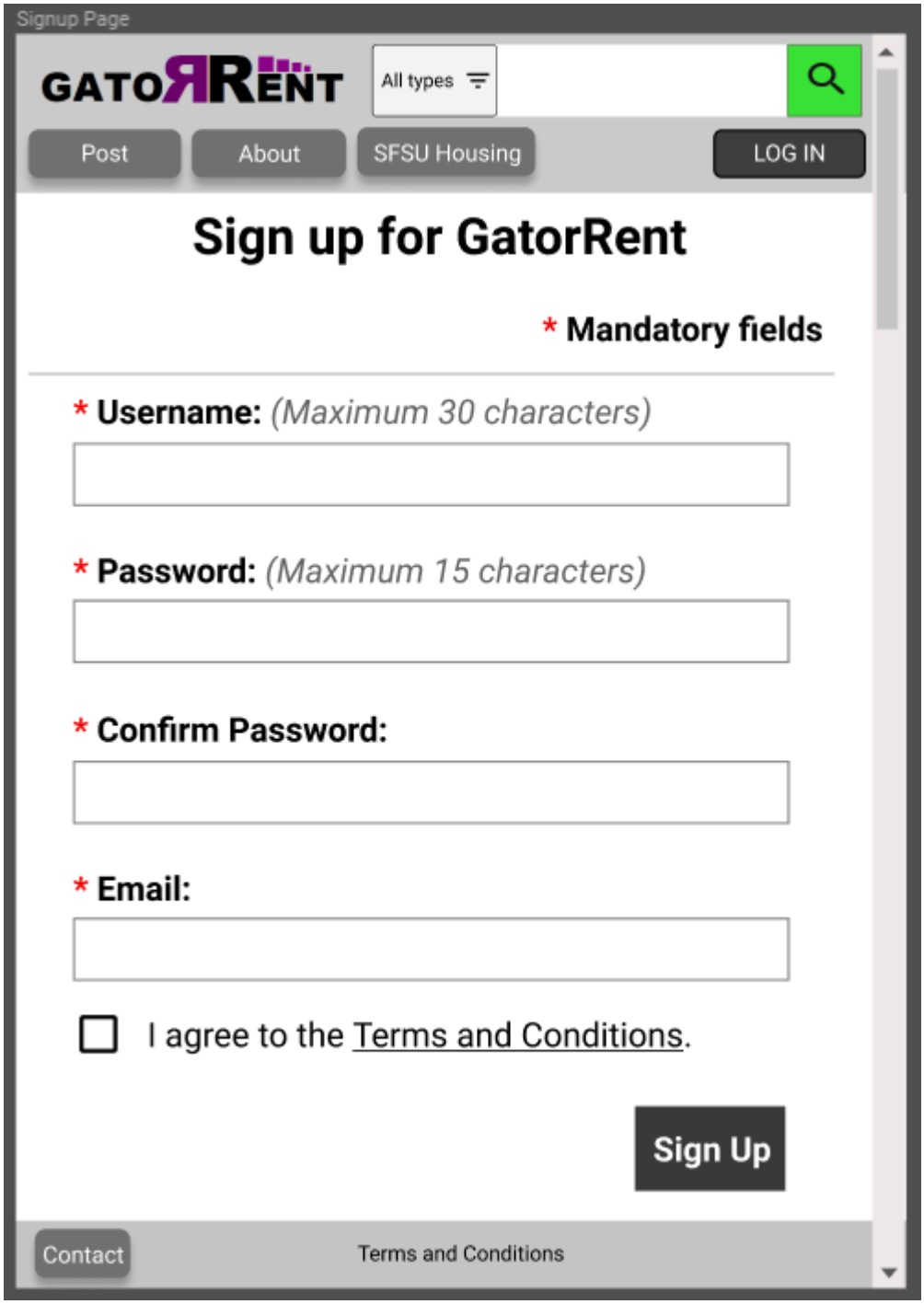
Posting a Listing


Login Page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Login to GatorRent
Username:
Password:
Don't have an account? Click here to sign up now!
Forgot password?
Login
Contact
Terms and Conditions

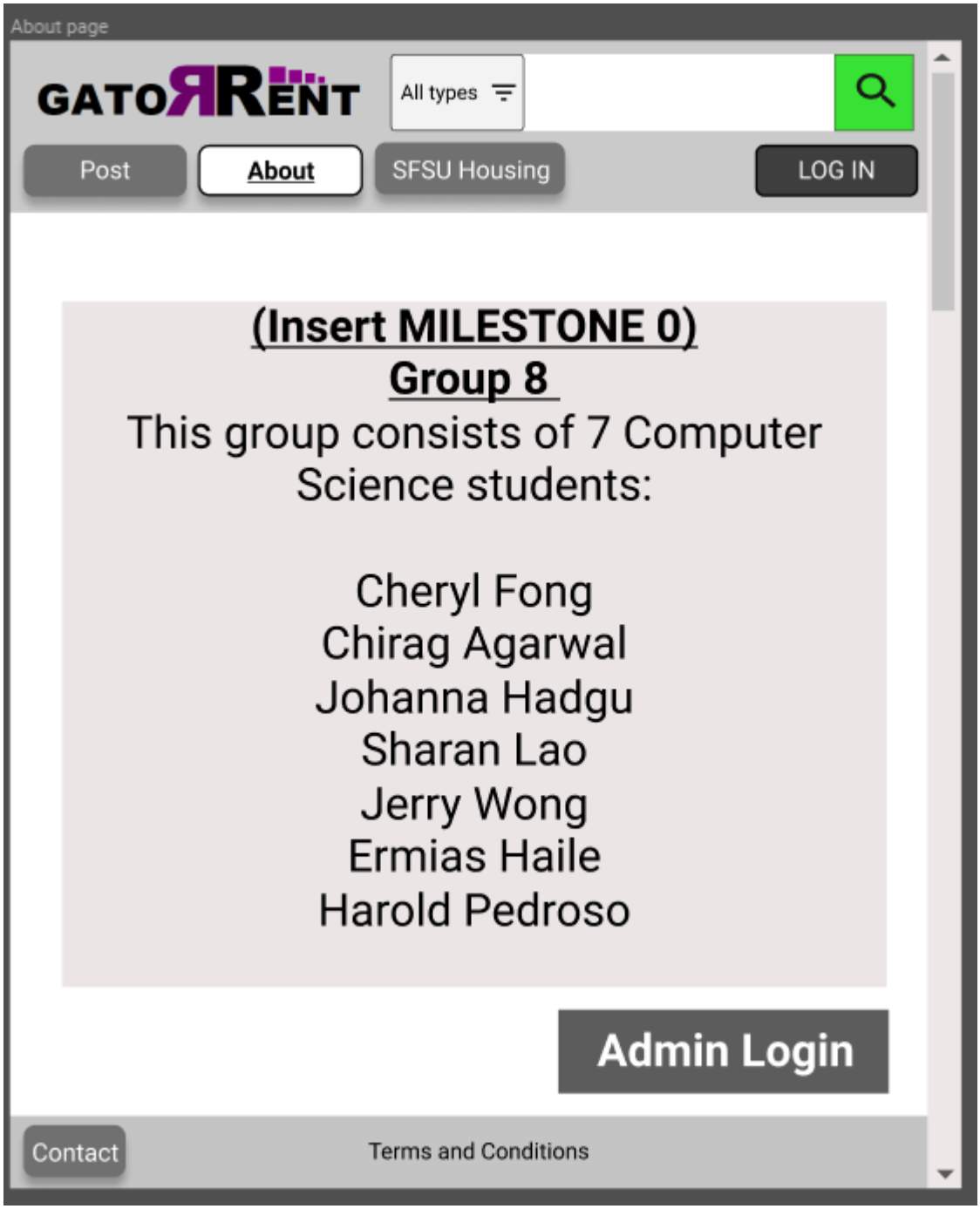
Login Page



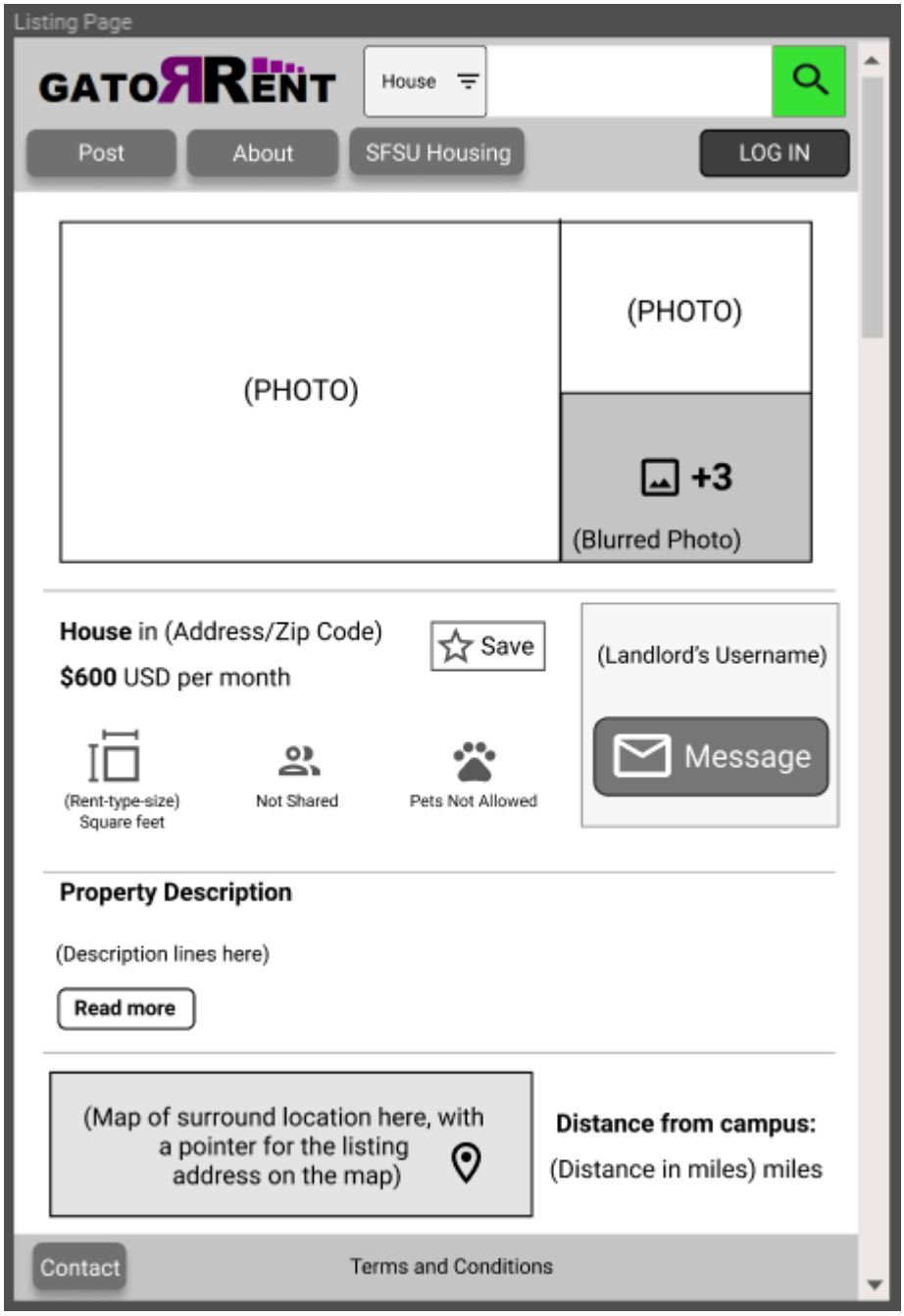
Admin Login Page



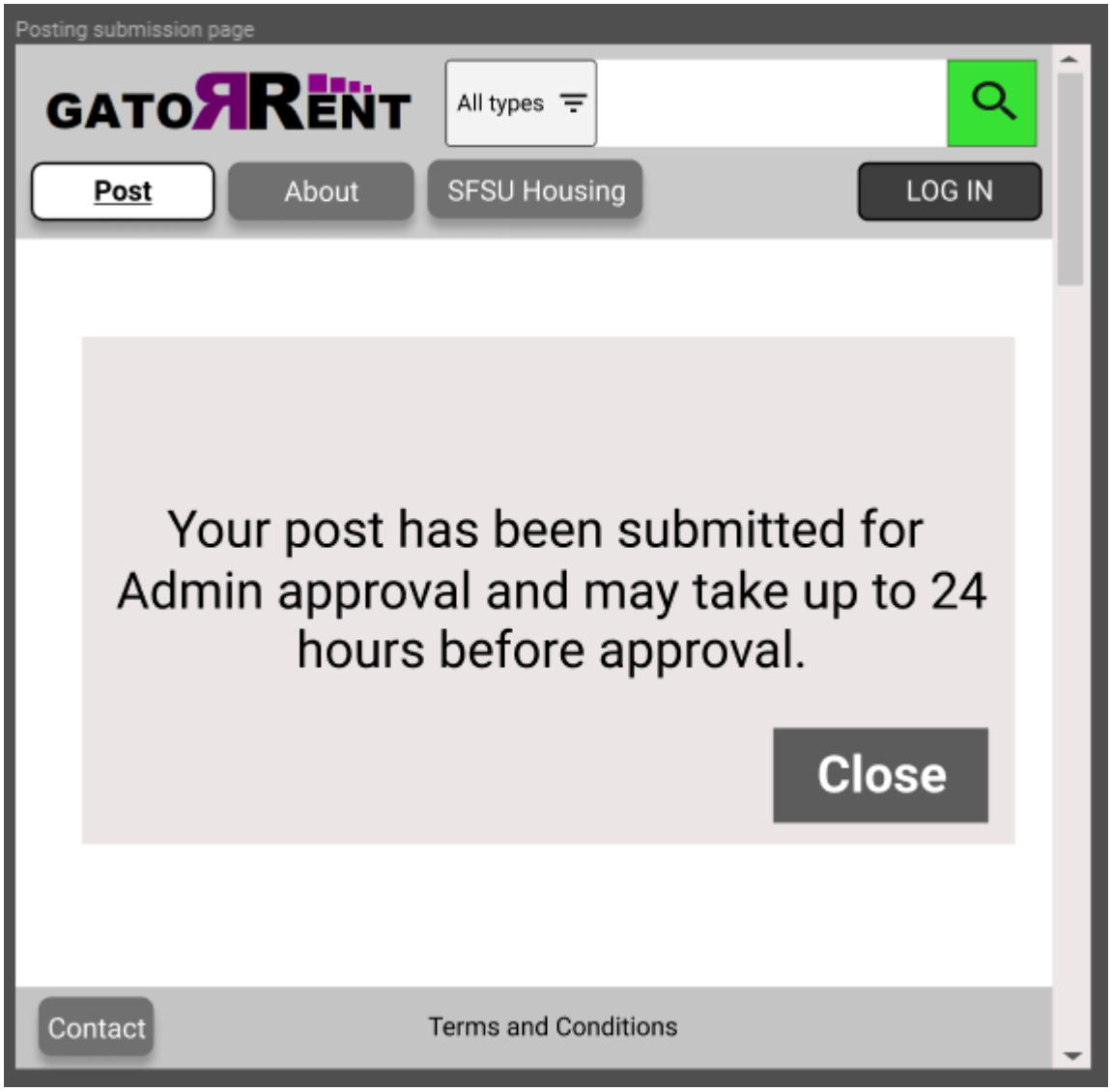
Registration Page


About page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
(Insert MILESTONE 0)
Group 8
This group consists of 7 Computer Science students:
Cheryl Fong
Chirag Agarwal
Johanna Hadgu
Sharan Lao
Jerry Wong
Ermias Haile
Harold Pedroso
Admin Login
Contact
Terms and Conditions

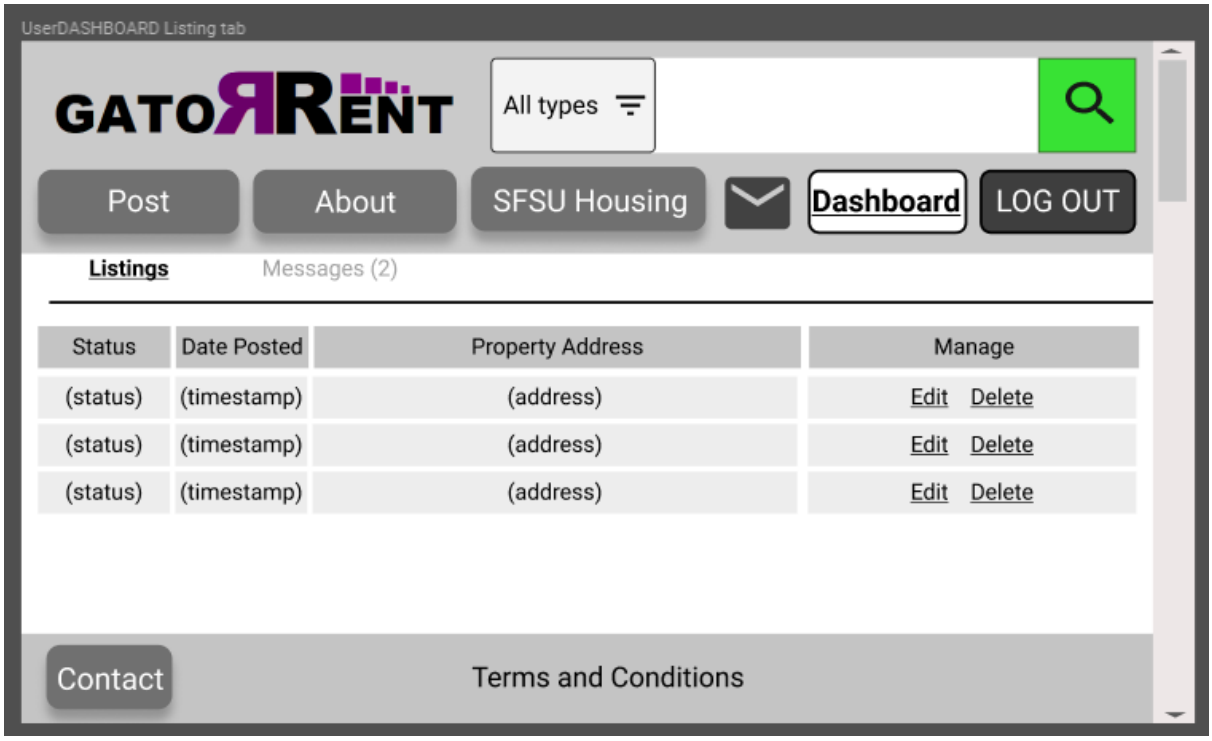
About Page



Selecting a Specific Listing

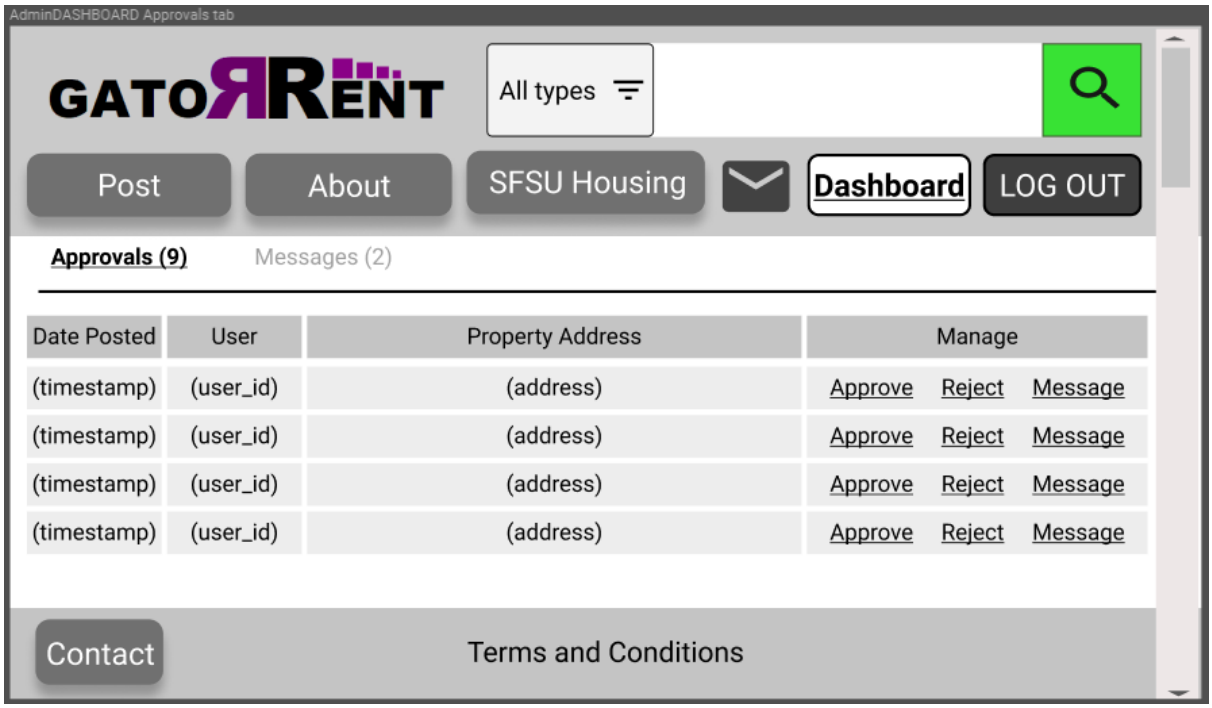


After Submitting a Post

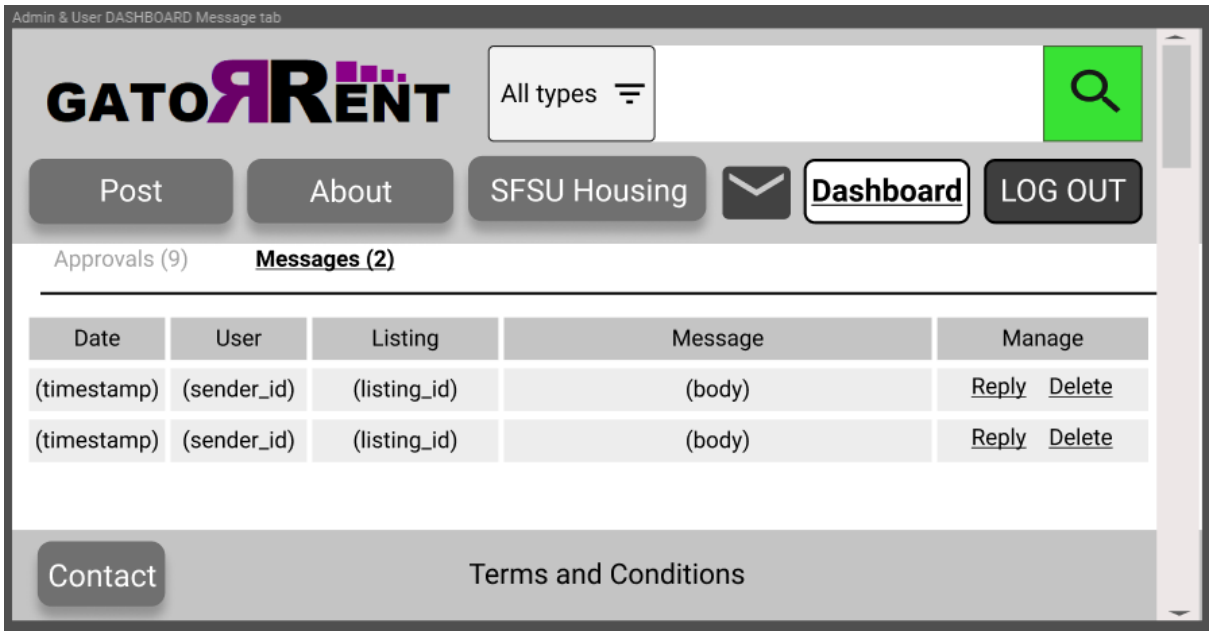


User Dashboard

User Messages



Admin Dashboard (Listing Management)



Admin Dashboard (Messages)

3.2 STORYBOARDS

3.2.1 Unregistered User (Kelly)

As a **non-registered user**, Kelly is presented with a variety of apartments **listings** available for rent and is able to **browse** through them. Kelly wants to search for apartments that are in close proximity to public transportation, campus and also within her price range. She **filters** her results by **zip code**, **distance,** and **price** and is presented with several apartment listings that are according to her specifications. Now that she found an acceptable apartment, she has solved her problem with GatorRent.

3.2.2 Registered User (Andrew)

Upon opening the website, Andrew notices that several **listings** are far beyond his price range and are not convenient enough to support his commute to work and school. Therefore, he filters his results according to the **distance** from local transit and **price.** Additionally, GatorRents allows him to accommodate for his cat allergies by **filtering** out renters that allow **pets** to reside in the premises. After finding one apartment that satisfies his conditions, Andrew creates an account and becomes a **registered user,** where he is required to accept the **terms and conditions** of the site. Once this is done, he is able to message the **landlord** and secure an apartment with the help of GatorRent.

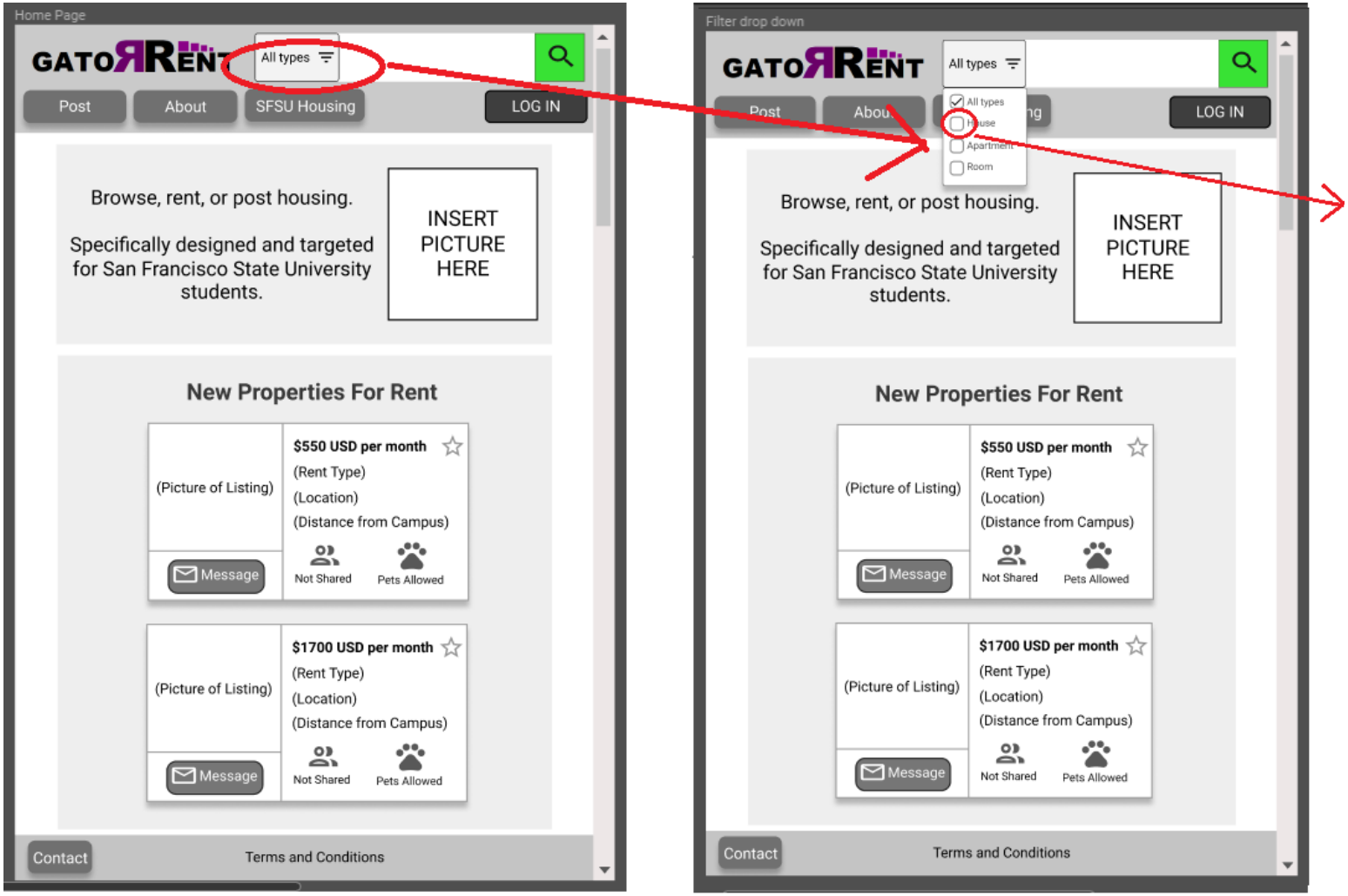
3.2.3 Landlord (Max)

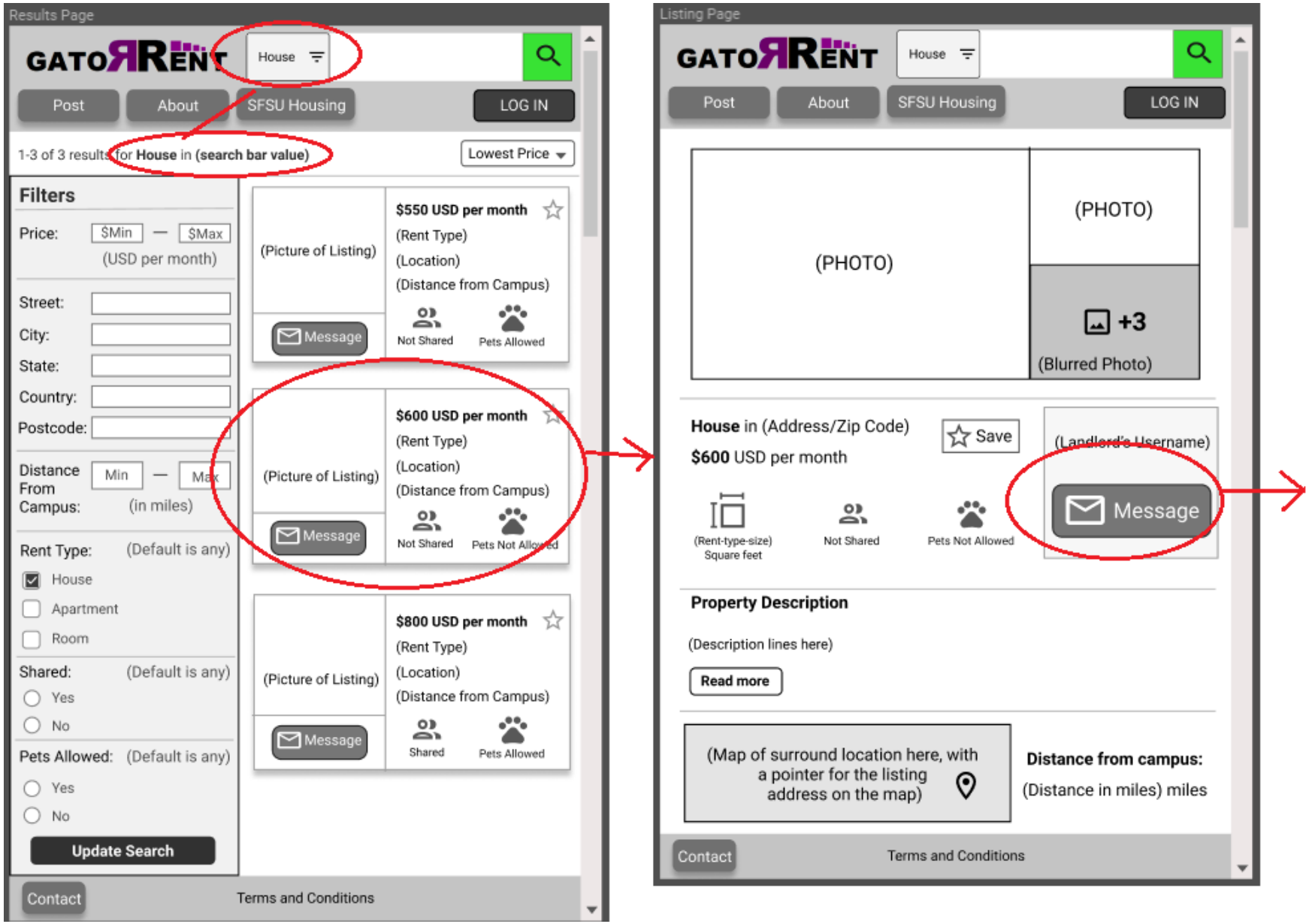
Having logged into GatorRent, Max is able to find options that allow him to post his apartment as a potential place to rent. Once he provides a description of the place, pricing, and images of the **property**, Max is prompted to register or log in before completing the post. Then, Max submits his post for **admin** approval.

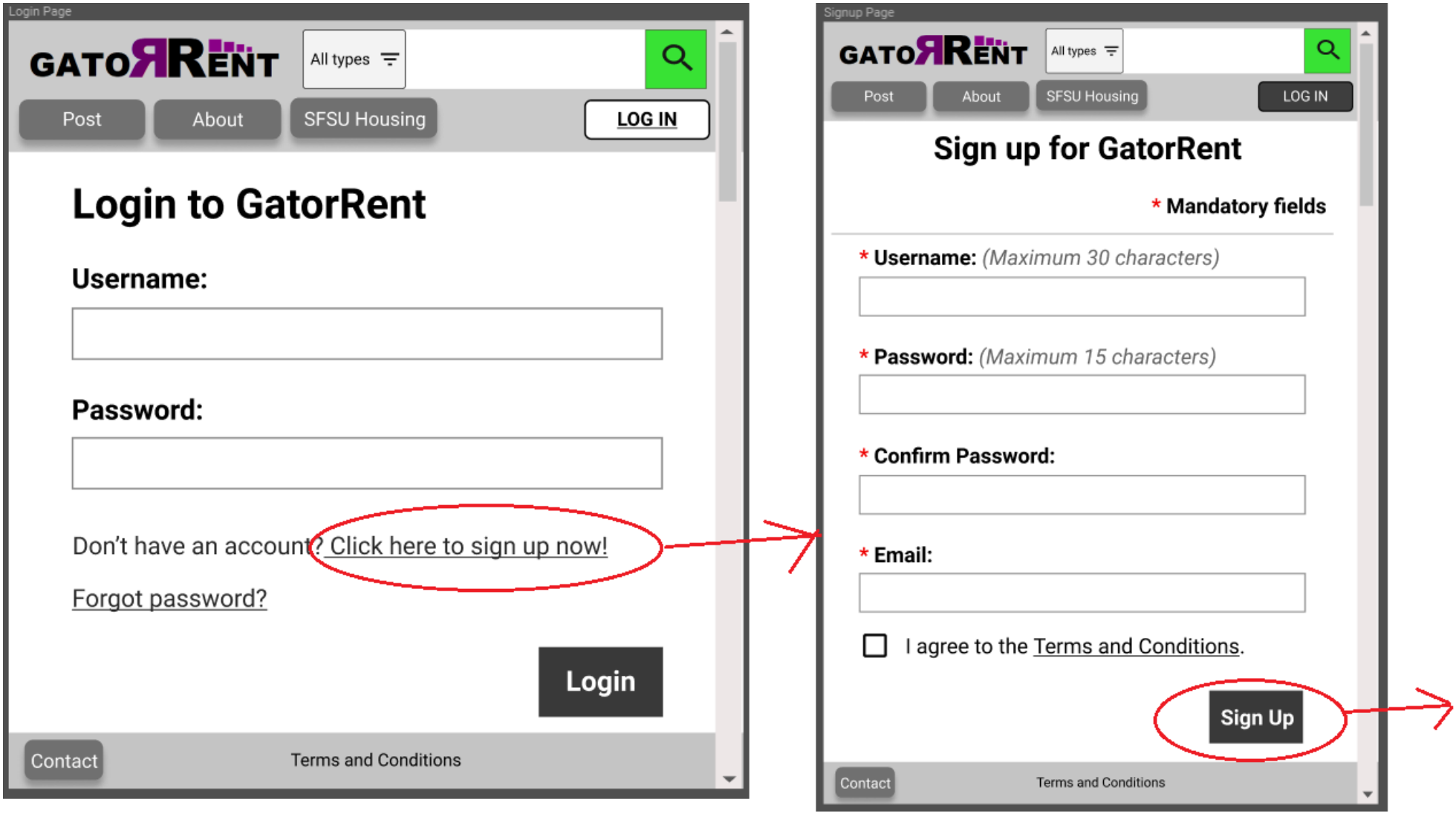
3.2.4 Admin (Jordan)

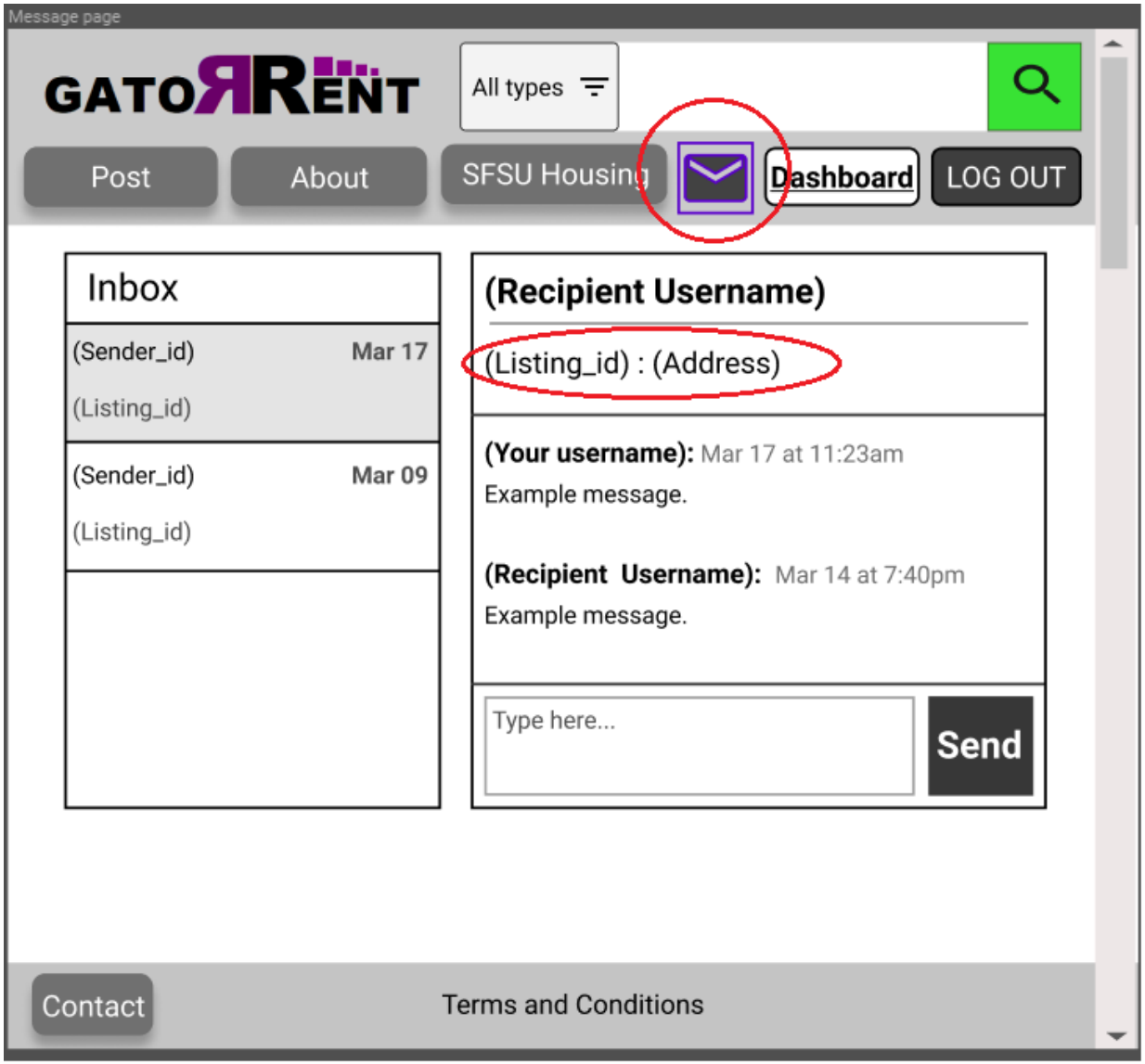
After logging in to the GatorRent dashboard, Jordan is greeted with a well-organized data list to include **listings** in need of approval and user-abuse reports. As part of her job, during a session of listing evaluation, Jordan comes across a **listing** that did not include photos of the **property**. Jordan denies the listing, contacting the lister and specifying that he'll need to modify his **listing** before it becomes live on the site. After approving several **listings**, she also notices that another **landlord** has been reported by 3 registered users. Having recalled that this specific **landlord** had already received a warning several months prior, Jordan immediately terminates the account.

1. **Use Case #1**
    a. Browse, search filter (filters: zip code, DISTANCE, PRICE, allowing pets, no), select, create account (username, password, email) accept Terms and Conditions), message landlord.

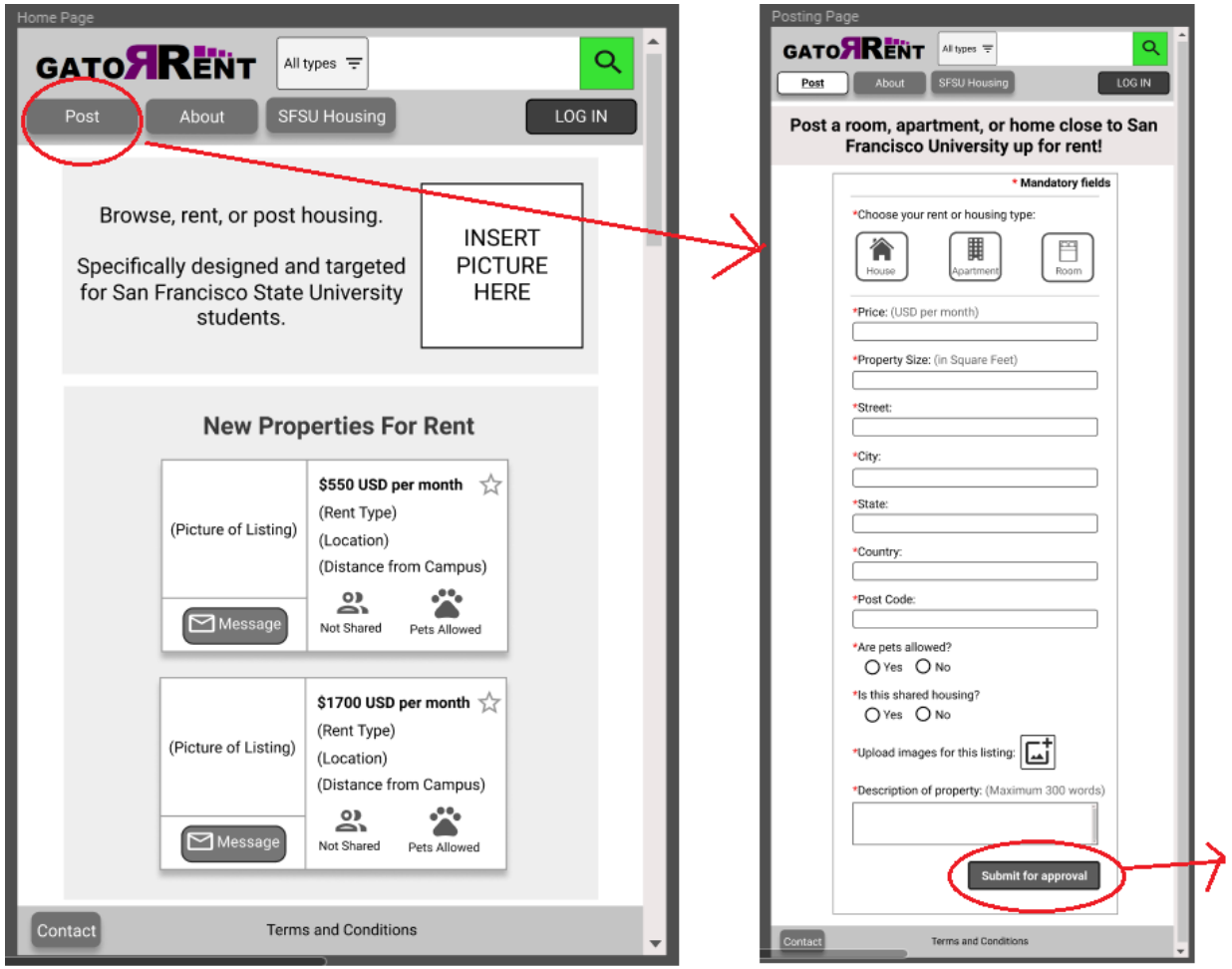


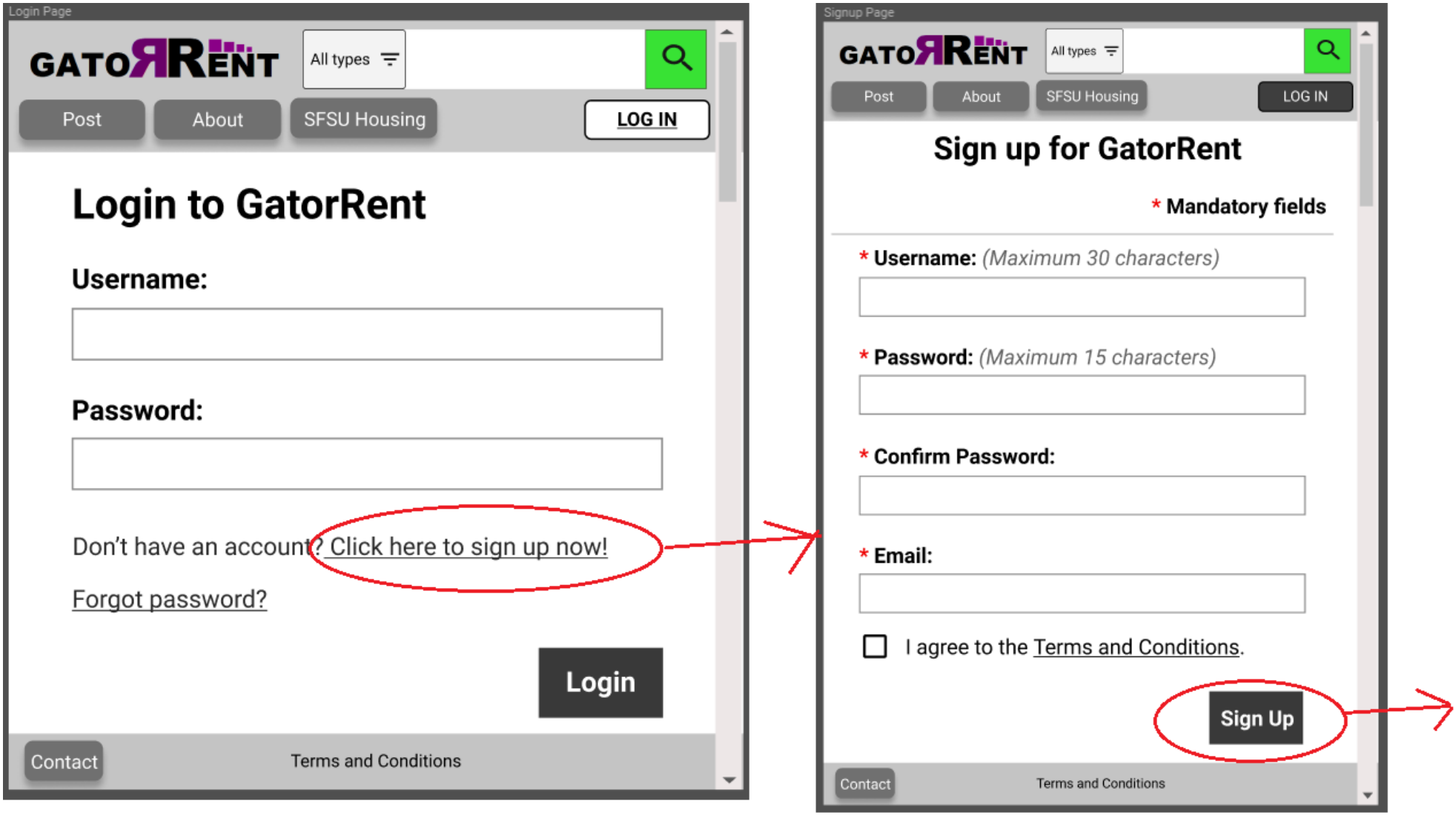

Results Page
GATORRENT
House
Post
About
SFSU Housing
LOG IN
1-3 of 3 results for House in (search bar value)
Lowest Price
Filters
Price: $Min — $Max
(USD per month)
Street:
City:
State:
Country:
Postcode:
Distance From Campus: Min — Max
(in miles)
Rent Type: (Default is any)
House
Apartment
Room
Shared: (Default is any)
Yes
No
Pets Allowed: (Default is any)
Yes
No
Update Search
(Picture of Listing)
$550 USD per month
(Rent Type)
(Location)
(Distance from Campus)
Message
Not Shared
Pets Allowed
(Picture of Listing)
$600 USD per month
(Rent Type)
(Location)
(Distance from Campus)
Message
Not Shared
Pets Not Allowed
(Picture of Listing)
$800 USD per month
(Rent Type)
(Location)
(Distance from Campus)
Message
Shared
Pets Allowed
Contact
Terms and Conditions
Listing Page
GATORRENT
House
Post
About
SFSU Housing
LOG IN
(PHOTO)
(PHOTO)
+3
(Blurred Photo)
House in (Address/Zip Code)
$600 USD per month
Save
(Landlord's Username)
Message
(Rent-type-size)
Square feet
Not Shared
Pets Not Allowed
Property Description
(Description lines here)
Read more
(Map of surround location here, with a pointer for the listing address on the map)
Distance from campus:
(Distance in miles) miles
Contact
Terms and Conditions

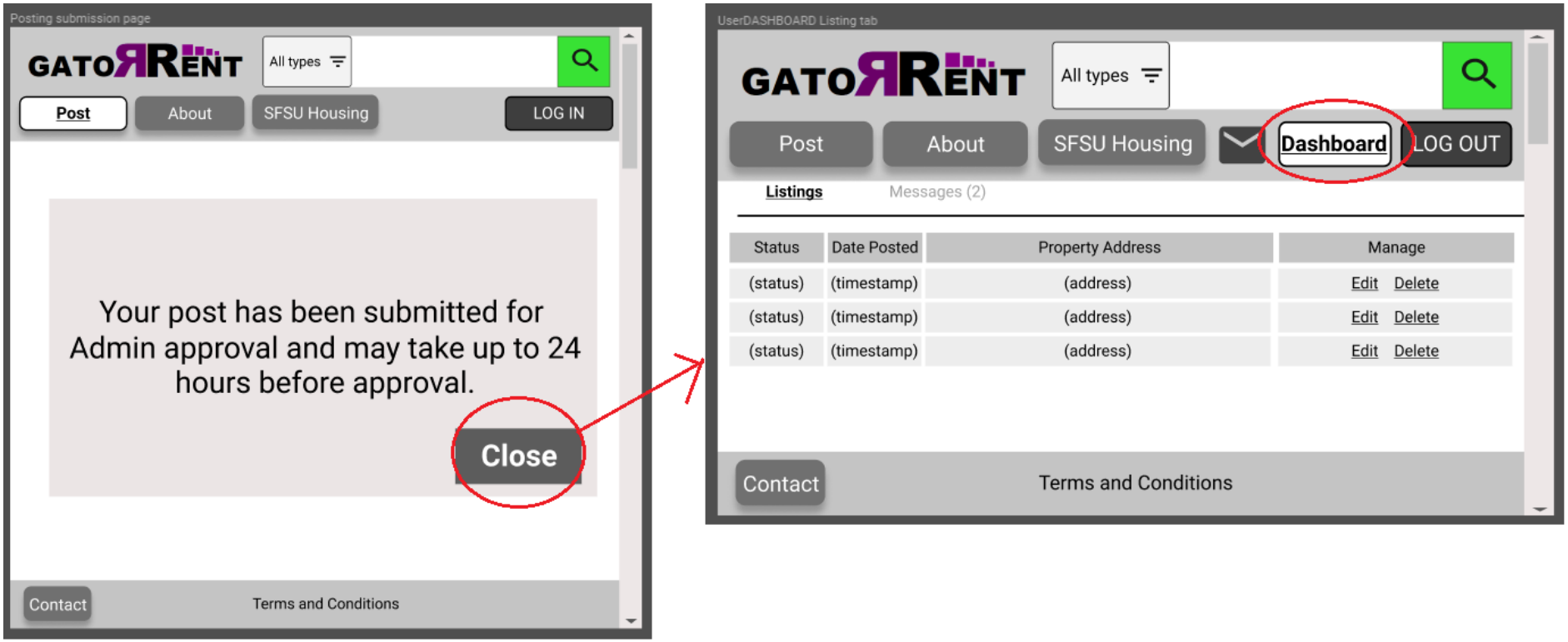

Login Page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Login to GatorRent
Username:
Password:
Don't have an account? Click here to sign up now!
Forgot password?
Login
Contact
Terms and Conditions
Signup Page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Sign up for GatorRent
* Mandatory fields
* Username: (Maximum 30 characters)
* Password: (Maximum 15 characters)
* Confirm Password:
* Email:
I agree to the Terms and Conditions.
Sign Up
Contact
Terms and Conditions


Message page
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Inbox
(Sender_id)
Mar 17
(Listing_id)
(Sender_id)
Mar 09
(Listing_id)
(Recipient Username)
(Listing_id) : (Address)
(Your username): Mar 17 at 11:23am
Example message.
(Recipient Username): Mar 14 at 7:40pm
Example message.
Type here...
Send
Contact
Terms and Conditions

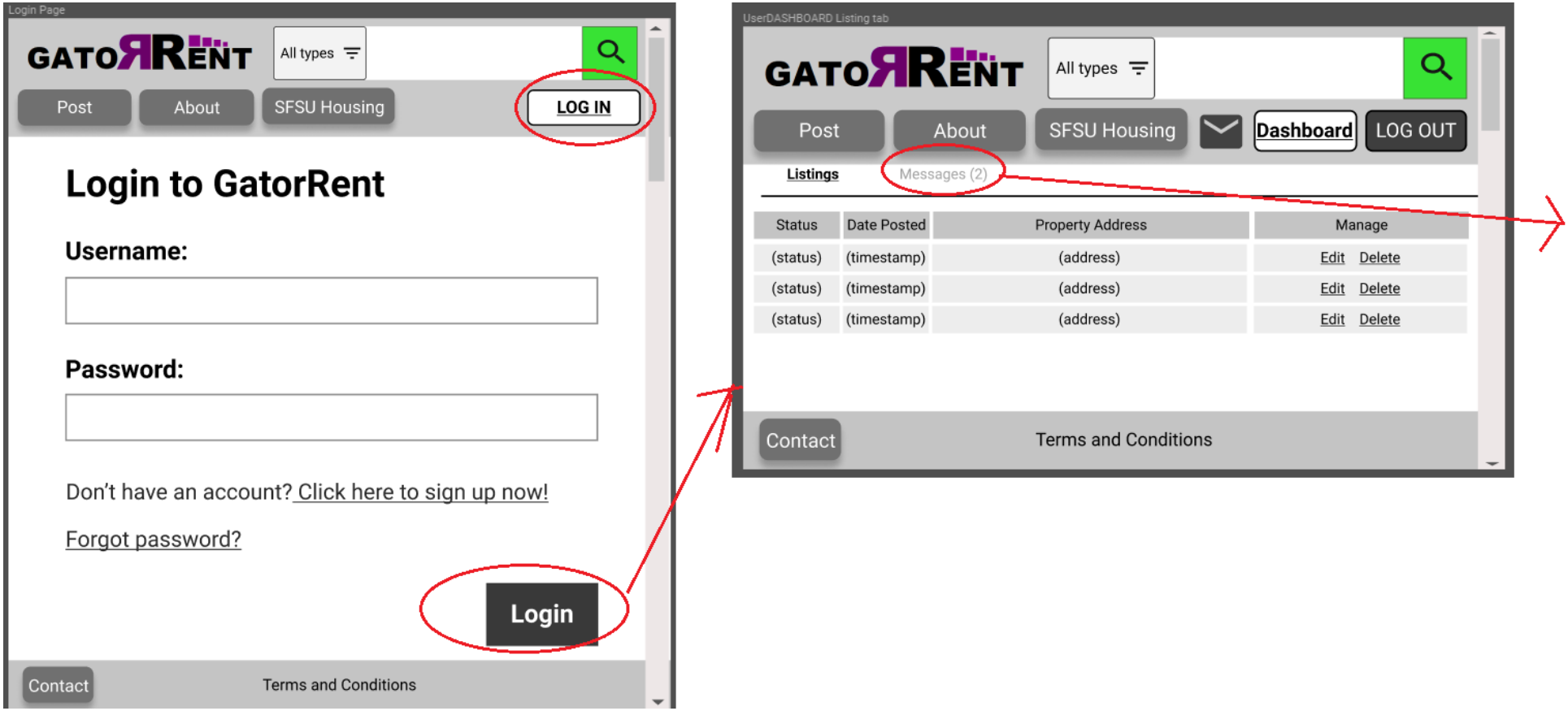
2. **Use Case #2**
    a. Post place for rent (rent-type:house/apartment/room, rent-type-size?insquarefeet, price:permonth, distance:fromcampus, postcode, pets?, shared? description, images), prompted to log in, submit post for admin approval.

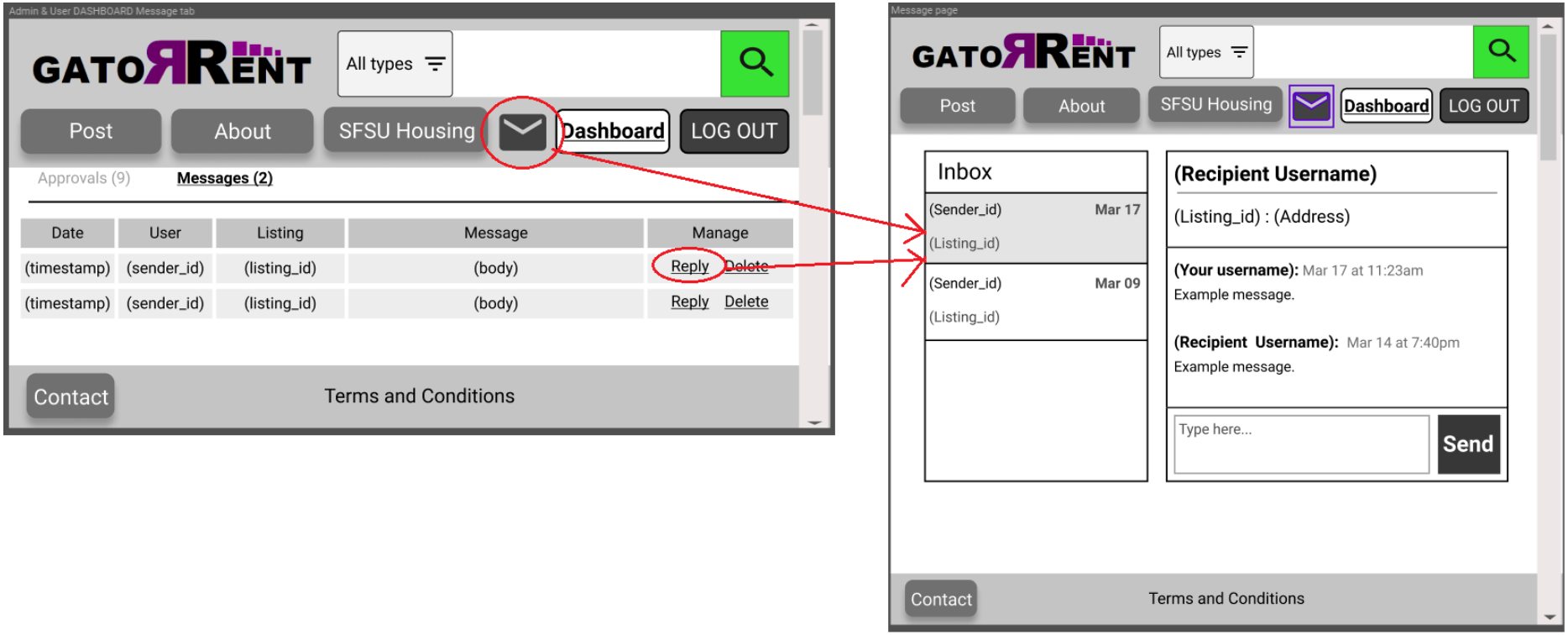



Login Page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Login to GatorRent
Username:
Password:
Don't have an account? Click here to sign up now!
Forgot password?
Login
Contact
Terms and Conditions
Signup Page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Sign up for GatorRent
* Mandatory fields
* Username: (Maximum 30 characters)
* Password: (Maximum 15 characters)
* Confirm Password:
* Email:
I agree to the Terms and Conditions.
Sign Up
Contact
Terms and Conditions

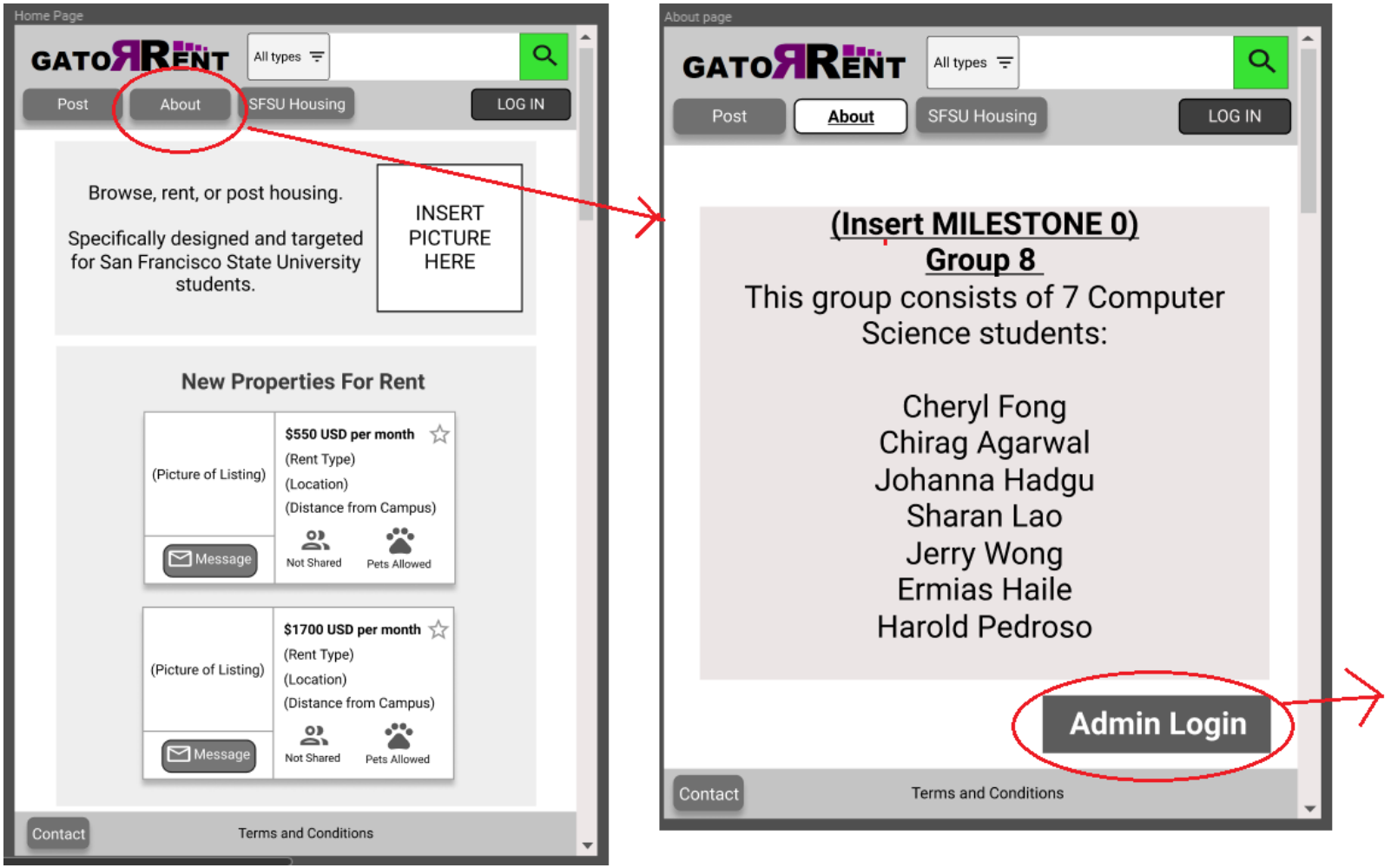

Posting submission page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Your post has been submitted for Admin approval and may take up to 24 hours before approval.
Close
Contact
Terms and Conditions
UserDASHBOARD Listing tab
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Listings
Messages (2)
Status
Date Posted
Property Address
Manage
(status)
(timestamp)
(address)
Edit
Delete
(status)
(timestamp)
(address)
Edit
Delete
(status)
(timestamp)
(address)
Edit
Delete
Contact
Terms and Conditions

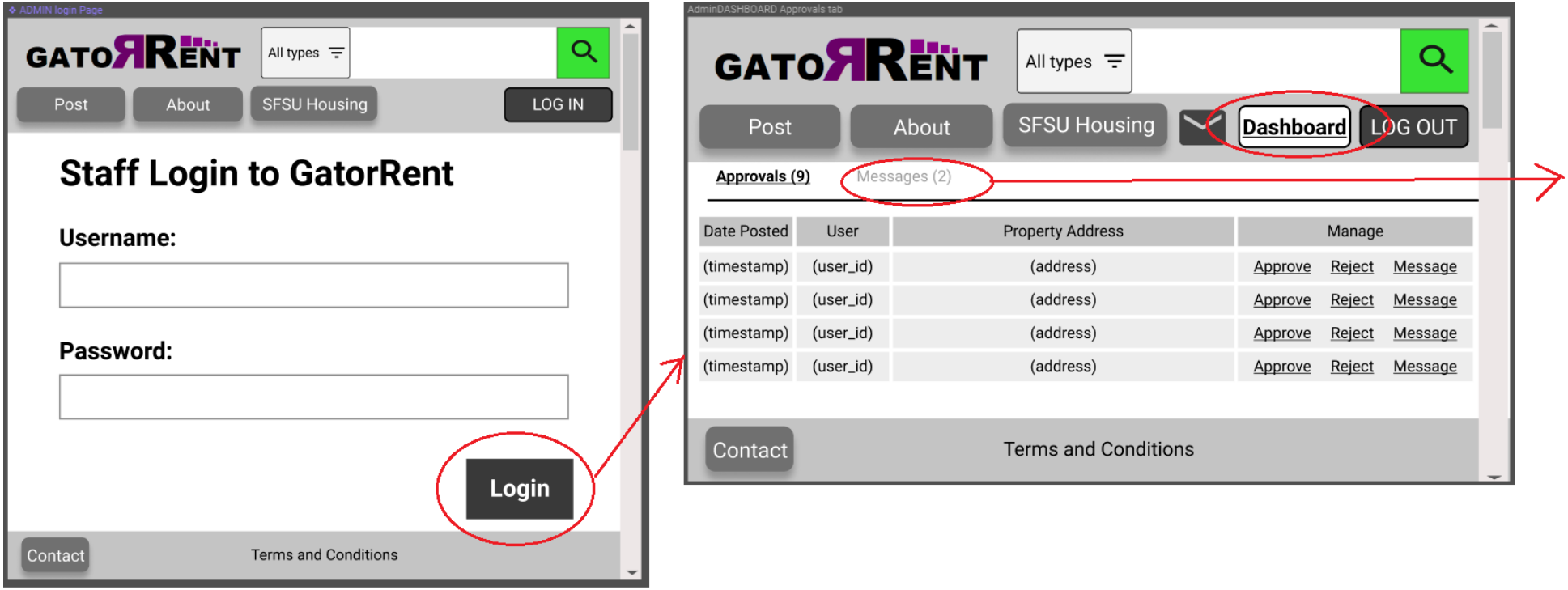
3. **Use Case #3**
    a. User login dashboard and messaging

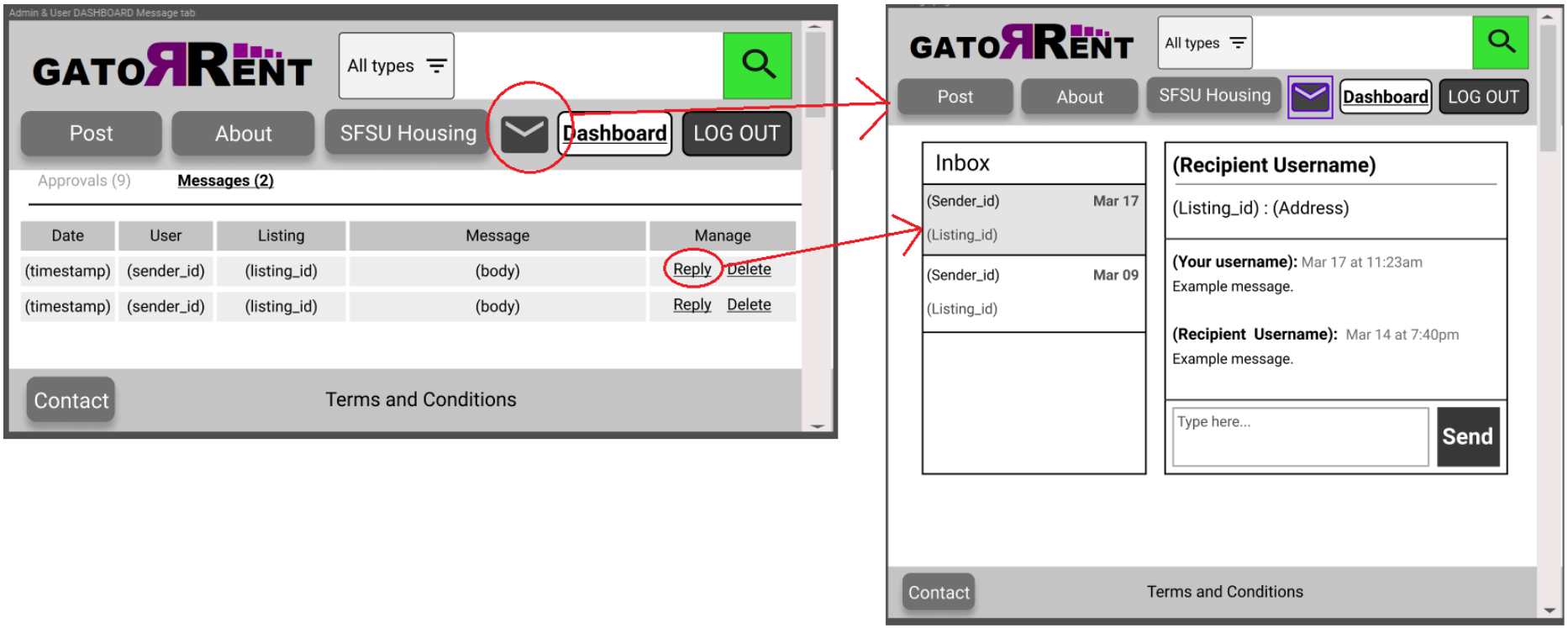



Admin & User DASHBOARD Message tab
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Approvals (9)
Messages (2)
Date
User
Listing
Message
Manage
(timestamp)
(sender_id)
(listing_id)
(body)
Reply
Delete
(timestamp)
(sender_id)
(listing_id)
(body)
Reply
Delete
Contact
Terms and Conditions
Message page
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Inbox
(Sender_id)
Mar 17
(Listing_id)
(Sender_id)
Mar 09
(Listing_id)
(Recipient Username)
(Listing_id) : (Address)
(Your username): Mar 17 at 11:23am
Example message.
(Recipient Username): Mar 14 at 7:40pm
Example message.
Type here...
Send
Contact
Terms and Conditions

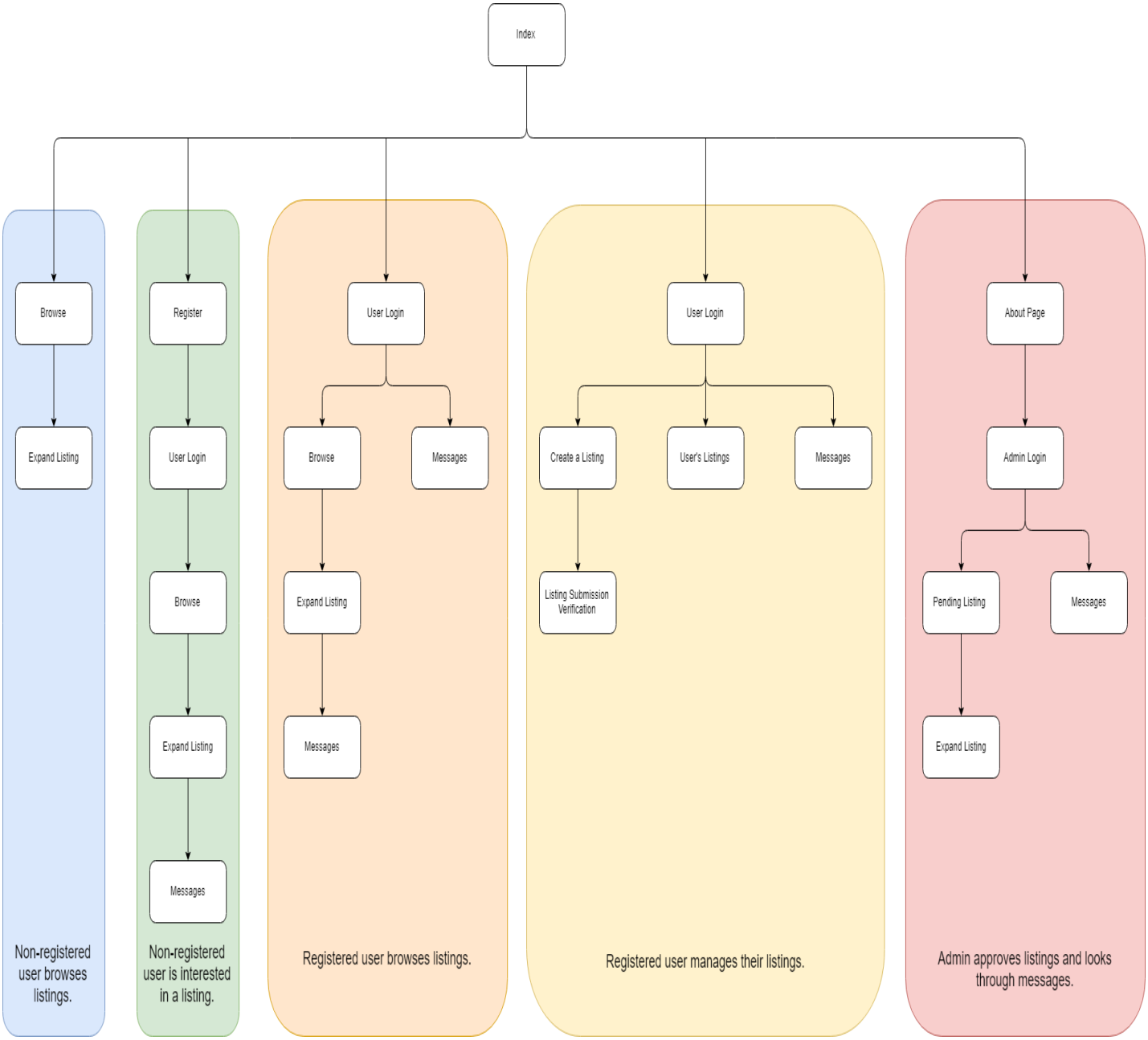
4. **Use Case #4**
    a. Admin logging in, gatorrent dashboard layout, show data list that includes listings that need approval + user-abuse reports




ADMIN login Page
GATORRENT
All types
Post
About
SFSU Housing
LOG IN
Staff Login to GatorRent
Username:
Password:
Login
Contact
Terms and Conditions
AdminDASHBOARD Approvals tab
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Approvals (9)
Messages (2)
Date Posted
User
Property Address
Manage
(timestamp)
(user_id)
(address)
Approve
Reject
Message
Contact
Terms and Conditions


Admin & User DASHBOARD Message tab
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Approvals (9)
Messages (2)
Date
User
Listing
Message
Manage
(timestamp)
(sender_id)
(listing_id)
(body)
Reply
Delete
(timestamp)
(sender_id)
(listing_id)
(body)
Reply
Delete
Contact
Terms and Conditions
GATORRENT
All types
Post
About
SFSU Housing
Dashboard
LOG OUT
Inbox
(Sender_id)
Mar 17
(Listing_id)
(Sender_id)
Mar 09
(Listing_id)
(Recipient Username)
(Listing_id) : (Address)
(Your username): Mar 17 at 11:23am
Example message.
(Recipient Username): Mar 14 at 7:40pm
Example message.
Type here...
Send
Contact
Terms and Conditions

3.3 Use Case Flow Chart



[Expand Image for Use Case Flow Chart]

4. HIGH LEVEL ARCHITECTURE, DATABASE ORGANIZATION

4.1 Database Organization (continued on 4.6):

4.1.1 We have a MySQL database which stores all the information in the form of different table entries. There are various tables such as a user table consisting user information, listing table consisting posting details and several others tables explained in the schema.

4.1.2 Database Schema:

4.1.2.1 The database contains six tables:

i. **User:** This table will contain information on all registered users, such as their **id**, **username**, **password**, **email**, and **user_type**. The password will be encrypted. Each entry creation is marked with a timestamp.

ii. **Listing:** This table will contain information of all listings such as a listing's **id**, **user_id** (foreign key to use table), its **description**, **rent-type_id** (foreign key to rent-type table), **price** (monthly rent amount), **distance_campus** (distance away from campus). Additional, columns that specify if the listing allows pets, and if its a shared property is recorded under the i**sPet** and **isShared** columns respectively. **Status** column specifies if the listing is still a draft, if the listing is currently open (unoccupied), under maintenance, or occupied and thumbnail. This table also contains the listing information, such as the **street**, **city**, **state**, **country**, and the coordinates of the location i.e. **lat** for latitude and **long** for longitude. Each entry creation and updates are marked with a timestamp.

iii. **Rent-type:** This table serves as a point for reference for various **type**(s) of rentable properties to the **Listing** table such as room, apartment and house (where rent-type_id is the foreign key to listing table).

iv. **Images**: This table will contain the image information in the form of the path of that file.

v. **Message:** This table will contain all the information pertaining the messages sent users of the website. Each message entry will contain the **body** of the message, the **sender_id** (id of the user who sent the message), and **listing_id** (id of the listing entry that this message is inquiring about). Each entry creation is marked with a timestamp.

[continued on Section 4.6]

4.2 Media Storage:

4.2.1 Images will be stored on the file system where the Images table where each entry will contain paths pointing to a folder of images belonging to a Listing entry. Images can be in the format, png, gif, jpeg or jpg.

4.3 Search/Filter Architecture and Implementation:

4.3.1 Search: The search will be implemented by using the LIKE statement combined with the '%' wildcard or pattern matching on strings. MySQL uses B-trees to queries the specified terms.

4.3.2 Implementation: For example, users can filter listing results to show results that matches the street field that they input, and where pets are allowed: SELECT * FROM listing WHERE MATCH (street) AGAINST ('keyword1) AND listing.isPet=1

4.4 API Endpoints:

4.4.1 Get all listing results, newest to oldest e.g. ip-address/listings/

4.4.2 Get listing results based on user query e.g. ip-address/listings?<query-string>

4.4.3 Get all listings that need approval e.g. ip-address/listings-to-approve/

4.5 New SW Tools/Frameworks:

4.5.1 As per discussion with Professor Anthony Souza, we are allowed to use an S3 Bucket to store images on AWS. However, we have decided as a team that we will use the Images table instead as described on item 4.1.2.1.iv above.

4.5.2 We will also use Google Map and Google Analytics APIs.

4.6 Visual Representation of Database Organization

Gatorrent-db – This database will contain the following tables displayed below:

USER		
Column	Type	Attributes and Constraints
id	int	PRIMARY KEY IDENTITY(1,1) NOT NULL
username(unique)	varchar(30)	UNIQUE NOT NULL
password	varchar(15)	NOT NULL
email	varchar(40)	NOT NULL
user_type	tinyint	
timestamp	timestamp	DEFAULT CURRENT_TIMESTAMP NOT NULL

LISTING		
Column	Type	Attributes and Constraints
id	int	PRIMARY KEY IDENTITY(1,1) NOT NULL
user_id	int	FOREIGN KEY(user_id) REFERENCES users(id)
description	text	
rent-type_id	int	FOREIGN KEY(rent-type_id) REFERENCES rent-type(id)
price	double	NOT NULL
distance_campus	float	
isPet	boolean	NOT NULL
isShared	boolean	NOT NULL
status	varchar	
street	varchar	
city	varchar	
state	varchar	
country	varchar	
postcode	int	
lat	float(10.6)	NOT NULL
long	float(10.6)	NOT NULL
isApproved	boolean	DEFAULT 0

image_id	int	FOREIGN KEY (image_id) REFERENCES images(id) NOT NULL
thumbnail_path	varchar	NOT NULL
timestamp	timestamp	DEFAULT CURRENT_TIMESTAMP NOT NULL

RENT-TYPE		
Column	Type	Attributes and Constraints
id	int	PRIMARY KEY IDENTITY(1,1) NOT NULL
type	varchar(40)	UNIQUE NOT NULL

IMAGES		
Column	Type	Attributes and Constraints
id	int	PRIMARY KEY IDENTITY(1,1) NOT NULL
image_path	varchar	NOT NULL

Message		
Column	Type	Attributes and Constraints
id	int	PRIMARY KEY IDENTITY(1,1) NOT NULL
body	text	
sender_id	int	FOREIGN KEY (sender_id REFERENCES user(id) NOT NULL

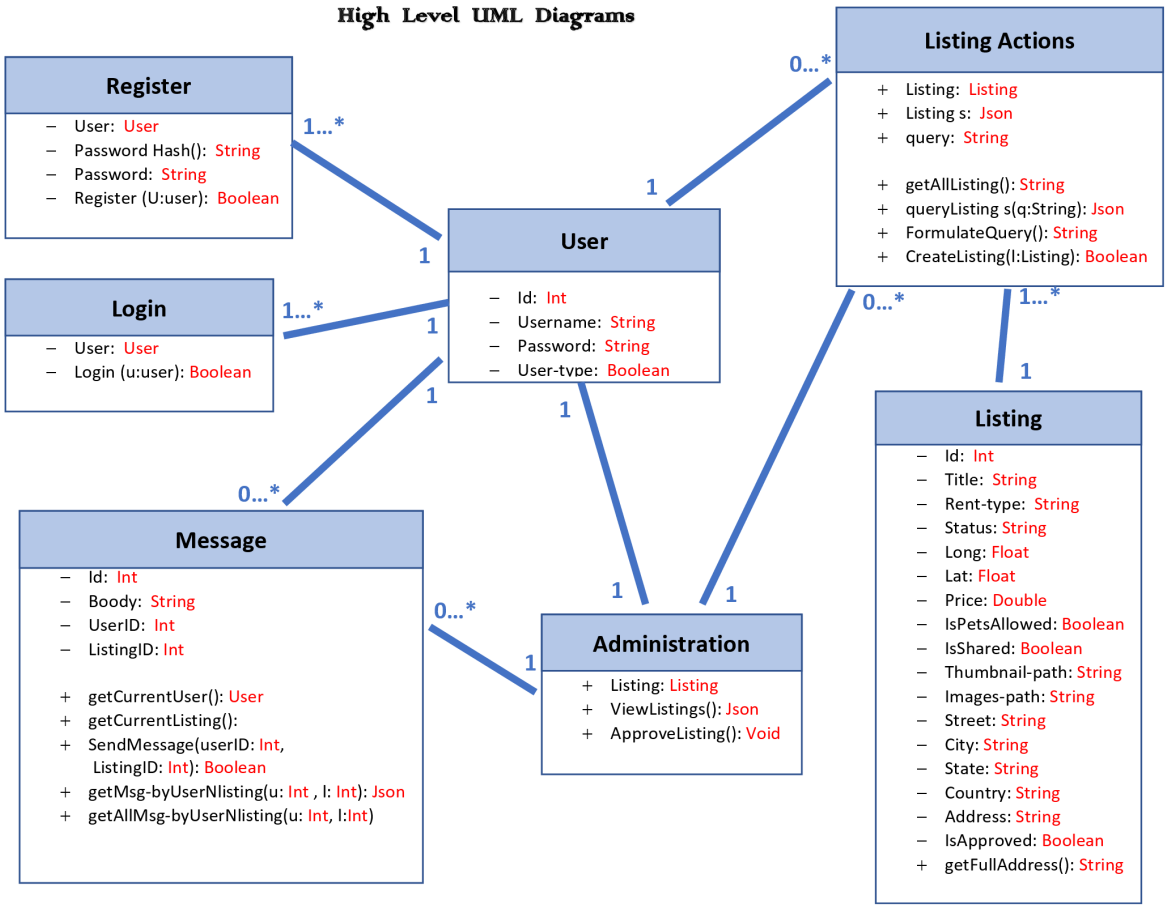
listing_id	int	FOREIGN KEY (listing_id REFERENCES listing(id) NOT NULL
timestamp	timestamp	DEFAULT CURRENT_TIMESTAMP NOT NULL

4.6.1 Website Meta-Data:

user-session — To keep track of the user's state when using the website
user-favourites — To keep a list of URLs of the user's saved listings
user-preference — To keep track of a user's preferences
activity-log — A log of entries added or removed from databases

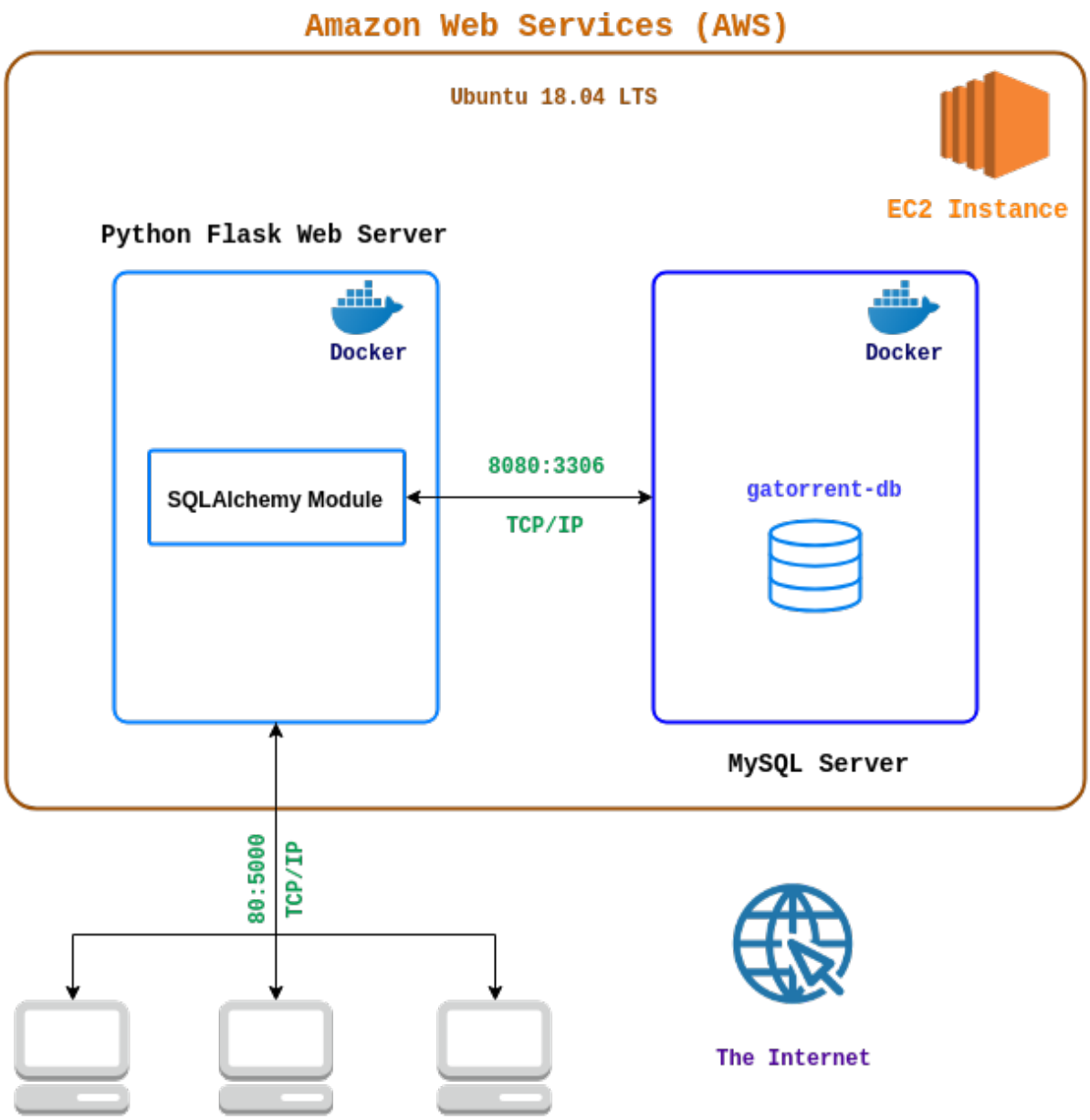
5. HIGH LEVEL UML DIAGRAMS

5.1. High Level UML Class Diagram:



[Expand image for High Level UML Class Diagram]

5.2. UML Component and Deployment Diagram:



6. ACTUAL KEY RISKS

6.1 Skill Risks

As a group, we have varying skills. Yet a majority of the group are first being exposed to using Git as a command line utility for version control and GitHub in this course. Therefore, one of the key issues that have arisen is ensuring that everyone is comfortably adjusting to using Git and GitHub. Some are still confused with the concept of branches, pull requests, and pushing the correct content to the right branches. In order to resolve this, our team leader has provided excellently well-documented instructions to the group repository that clearly outlines the correct commands necessary for execution, as well as the reason why we are doing it.

Moreover, familiarity with using Python and the web microframework Flask is part of our team's skills risk. For example, the usage of built in template tags and filters, various python libraries e.g. boto, manipulating and retrieving information in the form of JSON from the database.

One way to resolve our skills risks is to spend a sufficient amount of our own individual time to research the concepts we are struggling with, whether it's watching YouTube videos online or finding online textbooks. If the confusion still persists, it's also important to ask team members for assistance.

6.2 Schedule Risks

The team has collectively agreed to meet on Thursday mornings. However, we have experienced several issues with meeting on other days due to conflicting schedules. We found the best possible solutions are to first, prioritize the time allocated in class for working on the group project while being respectful of everyone's time. Second, to make strides in being a collaboratively distributed, yet effective team. If neither of these are possible, we plan to utilize online services such as Skype, Google Hangouts, or Zoom to meet.

6.3 Technical Risks

One of the key technical risks our team faces is understanding the working components of the development environment particularly on Docker usage.

Docker was chosen as part of the development environment and as well as the deployment foundation because it is one of the leading components in modern web microservices and for software build/test environments. Milestone 0 was a great opportunity to introduce the idea of containerization and learn about industry web technologies. It was a refresher to some and a brand-new topic to others. All in all, the team got to learn about a hot industry trend and potentially add a new skill to their bag of tricks to show off to their future employers.

To help ease the transition in using Docker, installation instructions were given to the team and conducted together as a team activity. There are parts of the installation that individual team members did not understand, and understanding

what is going on when the development environment is launched (i.e. understanding log outputs) may come across as difficult. The type of commands to use and tweaking the environment to the preference of the individual can seem very unfamiliar at this point.

To address these risks, detailed documentation has been placed on the team's GitHub repository for quick reference. Additionally, there's an abundance of online resources to take advantage of and as well as a welcoming Docker user community to reach out to for help via Docker's GitHub and Slack (and even on IRC). Overall, with patience and practice, using Docker will eventually become second nature.

Another technical risk that the team faces is the usage of Amazon Web Services (AWS). AWS offers an easy to use interface, however, it will take time to get use to and understand the technical terms used by AWS.

In order to reduce the severity of these risks, it is up to the team members to put time into reading, research and be willing to ask for help.

6.4 Teamwork Risks

The prominent risks that we face as a team are inconsistent usage of our dedicated group Slack channel and GitHub pull requests and issues. One way to mitigate these risks is to have all team members install the Slack app on their mobile devices to get notifications instantly and check our team's GitHub repository regularly. The best solution is to tackle the problem that arises as soon as possible.

Additionally, it's key that we address the individual(s) who aren't putting the time into their tasks and/or group activities in a respectful manner. Asking questions and offering help whenever and wherever need is highly encouraged among team members. The goal is to maintain a level of open-mindedness, transparent communication and dedication to ensure productivity and to mitigate confusion and miscommunication.

6.5 Legal/Content Risks

Any legal or content issue that may arise is the presence of users that may be attempting to use this website for malicious purposes. For example, users may post fake listings and bait users into renting a fake property offline. Our message feature will allow users to contact the site administrators about any suspicious listings. Messages containing such reports will arrive at the admin dashboard. It will be up to the site administrators to dispute such claims and take appropriate action such as taking a listing off the browsing results and/or ban a user by placing the user's email on a blacklist.

Additionally, any acts committed by registered users who violate the terms and conditions agreement will be acted upon accordingly by the site administrators.

7. PROJECT MANAGEMENT

In order to promote efficient workflow that is both timely and productive, we have collectively decided to use the GitHub Projects Board as an alternative to Trello. It is comprised of issues, pull requests, and notes that are organized according to the user. It will allow us to keep track of tasks and bugs as well as enhancements to our project. For this specific project, the Projects Board will allow us to assign certain individuals with specific tasks and deadlines. It will also enable us to organize our tasks on a higher level. Once an individual has been assigned a task, they will receive an email notification, alerting them of the task description, deadline, and people working on this issue. Although our team has been using slack as a means of communication, the Projects Board provides a clear and concise outline for the group to reference throughout the semester.

All in all, for Milestone 2 and future tasks, our Projects Board will allow us to reduce the amount of confusion on what needs to be done by whom. Every assigned task, whether closed or opened will remain on GitHub, allowing us to keep a log of who did what. Additionally, we will be able to add labels to help organize our issues as well as include any dependencies amongst current or future tasks. This will be helpful as our tasks get increasingly greater in number as the days progress and promote efficient workflow.

Software Engineering CSC 648/848 Section 02 Spring 2019

Milestone 3

Team 08

May 8th, 2019

Cheryl Fong
cfong8@mail.sfsu.edu

Jerry Wong

Chirag Agarwal

Harold Pedroso

Johanna Hadgu

Sharan Lao

Ermias Haile

FROZEN STATE

1. UI AND FUNCTIONALITY FEEDBACK

1.1 Home Page:

1.1.1 We need to add some SFSU related words

1.1.2 Our main image should be replaced by a couple of recently posted listings.

1.1.3 We need to move the “About” and “Post” to the left side of the navigation bar.

1.2 Search (including search field validation):

1.2.1 It’s important that we resolve our persistence issues and ensure search validations

1.3 Search Results:

1.3.1 Instead of displaying the SQL query, we should display “Show 1 out of 1” when displaying the results of a recent search.

1.3.2 Our search results page should have clickable images.

1.3.3 When displaying all of the results after a search has been conducted, display “Showing 55 results” instead of “showing all results”.

1.3.4 All the cards should be the same size and aligned (have the line around the boxes)

1.4 Filtering:

1.4.1 Replace the input field boxes to ranges of radio buttons e.g. “less than $500, 501 – 1000”

1.4.2 We need to remove property type from the side panel (distance and prices is number 1, 2-3 filter only).

1.4.3 For usability, flip the filter and clear button positions on the side panel.

1.5 Search Details and Maps:

1.5.1 When clicking on the “more info” button, we should make the main apartment image smaller and put the map of the listing to the right of the listing image.

1.5.2 The “more info” page should include “distance” and the “property title”

1.5.3 Put distance and the property title on the page.

1.6 Messaging/Contact seller/User:

1.6.1 It shouldn’t lead to dashboard because they might not have even registered yet.

1.6.2 A reply function is not needed. Users should only be able to see the messages and contact them.

1.6.3 When hitting contact the seller it should have the apartment info pre-populated, let them enter their contact information in another box, cancel(left) and send(right) button.

1.7 Data Upload:

1.7.1 We need to increase the sizing of the font of the posting

1.7.2 Make sure to maintain the alignment of the test and only have a single column.
1.7.3 We need to Mark the mandatory fields using '*' and '* Mandatory field'.
1.7.4 The posting form should have a cancel button
1.7.5 After submitting the form, users should be presented with a message, indicating that it may take up to 24 hours for the post to be reviewed

1.8 User and Admin Dashboards:

1.8.1 The login page needs to:
1.8.1.1 Have larger text
1.8.1.2 Have an implemented captcha
1.8.1.3 Have a link for a user who has forgotten their password.
1.8.2 The log in button that is located in the login page needs to have a link for a user to register on the site. It should be located under the button, shifted to the right
1.8.3 We need to make sure that after logging in, the user's dashboard should have a "Welcome (Username)" displayed.
1.8.4 To improve the layout of the navigation, move it from the side to the top center of the page.
1.8.5 When a user is clicking to obtain additional info on a listing, the listing should open to another tab.
1.8.6 In the user/admin dashboard, include another tab that lists (pending listings). If we have time, make the listings sortable.
1.8.7 When user is viewing all of their messages in dashboard, remove the "reply" button.

1.9 Brief Review of Coding, GitHub, Databases, etc.

1.9.1 For branches, it is recommended to have a development branch to test everything before it gets merged onto Master.
1.9.2 It's important to make separate branches for each feature being developed and name specifically, not "pages-sprint", more like "ui-resultspage"
1.9.3 Our commit messages need work. Commit messages such as "Update result pages" is not a good message. Instead, we should rewrite it as "update results page to add new buttons" or "update results page to make buttons clickable." Use keywords so the commit messages can easily be browsed and searched through.
1.9.4 Also, instead of writing "remove unnecessary lines" change the commit message to "remove unnecessary lines in results page"

1.10 DB tables and data in them using the tool of your choice (e.g. whatever your site admin would use):

1.10.1 Copy images that you own because you don't want to rely on someone else's server that might change its path. Have like 20 images of our own.
1.10.2 Make sure passwords are hashed/properly protected. Do not store as plain text.

2. PROJECT STATUS

2.1 Teamwork:

2.1.1 Currently, we are working on communication and contributing our assigned tasks on time.

2.2 Risks:

2.2.1 We currently have no risks. All have been addressed and resolved appropriately.

2.3 Coding Practices:

2.3.1 Our tables will consist of consistent basic database table names.

2.3.2 Our GitHub comments need to be descriptive and specific.

2.3.3 Our variable names need to be easy to understand, if they aren't, include a comment where the variable is defined, stating what the purpose of that variable is.

2.3.4 Our GitHub branches should be specific. A single branch should not consist of multiple, large changes to the site.

2.4 Digital content (e.g. images, video):

2.4.1 Status: Complete

2.4.2 Chosen formats: .jpg

2.4.3 Availability: Free to use

2.4.4 Resolution: 3715 x 2362

2.5 Addressing Site Security and Safe Coding Practices:

2.5.1 Site security and safe coding practices have been ensured. We are protecting the user's personal information by protecting and encrypting passwords via the Bcrypt library.

2.5.2 The site is being protected from spam, automated accesses and extractions with reCaptcha

2.5.3 To secure our images, we used the **secure_filename()** function, as recommended by the **Flask Documentation Release 0.9 Dev**, to secure a filename before we stored it.

2.6 Usage of proper SE code management practices

2.6.1 Code submissions are being reviewed through pull requests. Once a branch that is comprised of a feature is thought to be complete, a pull request is made. The individual who makes the pull request is required to tag, at minimum, 1 team member to review and test the code on their local device. The number of members tagged to test depend on the size of the change. Once this change has been reviewed, tested and confirmed to have no conflicts with the master branch, per the approval of the GitHub Master, it is merged.

2.7 Other:

2.7.1 Based on the feedback we have received; we are now focusing on fixing up the UI pages. If we can't do admin page, drop it and do the demo using workbench or PHP or MySQL.

2.7.2 We will optimize the user features first. An example of this would be prioritizing the distance filter over the admin features.

3. LIST OF P1 FEATURES COMMITTED FOR DELIVERY

3.1 Login & Registration:

3.1.1 Users shall be able to log in to their account with some form of captcha.

3.1.2 Users shall be able to register as a user of the website with some form of captcha and acceptance of terms.

3.2 Home Page:

3.2.1 Shall greet the user for visiting the site and state the purpose of the site.

3.2.2 Shall display recently added listings.

3.3 Searching Functionality:

3.3.1 Users shall be able to filter depending on listing type.

3.3.2 Users shall be able to filter according to zip code.

3.3.3 Users shall be able to filter listing results by lowest or highest price, price range, postcode, distance from campus, rent-type, rent-type price, if pets are allowed, and if the rent-type is shared.

3.3.4 Users shall be able to search by text (up to 40 alphanumeric characters) in the search bar.

3.4 Search Results:

3.4.1 Users shall be able to view search results with images as thumbnails.

3.5 Details page with Map:

3.5.1 When clicking for more info regarding a specific listing, users shall be able to see more details about the property and a map.

3.6 Messaging:

3.6.1 Users shall be able to message the landlord inquiring about a listing.

3.6.2 Users shall be able to access messages regarding their own properties.

3.6.3 Users shall be able to message the Admin through the Contact Us page, regarding any concerns.

3.7 Posting:

3.7.1 Shall be able to post a listing for rent.

3.7.2 Shall be able to upload multiple images.

3.8 User Dashboard:

3.8.1 Users shall be able to view their listings.

3.8.2 Users shall be able to view all received messages regarding their listings.

3.9 Admin Dashboard:

3.9.1 Users shall be able to access their dashboard via a special login id.

3.9.2 Admin users shall approve or disprove listing results before they go live on the site.

3.9.3 Admin users shall delete any listing.

3.9.4 Admin users can block or delete any user for misuse.

3.10 Additional Priority 1 Tasks Specified by the CEO:

3.10.1 Shall work on two browsers (Chrome 74.0.3729.131, Safari 12.1).

3.10.2 Pages shall be responsive and render well on mobile (e.g. the home, search results, details and maps).

3.10.3 Passwords shall be encrypted in the database.

Software Engineering CSC 648/848 Section 02 Spring 2019

Milestone 4

Team 08

May 8th, 2019

Cheryl Fong
cfong8@mail.sfsu.edu

Jerry Wong

Chirag Agarwal

Harold Pedroso

Johanna Hadgu

Sharan Lao

Ermias Haile

REVISION TABLE		
Date	**Version Number**	**Description**
05/08/19	1.0	Date submitted for review
5/19/2019	1.1	Revised after Feedback

FROZEN STATE

1. PRODUCT SUMMARY

1.1 Our product GatorRent is an accommodation-search application that is intended for the use of the students of San Francisco State University (SFSU) to find housing and for landlords that are looking for renters alike. This application is important as on-campus housing at SFSU is extremely limited and expensive thus making it difficult for both domestic students and international students to find accommodation within their budget and preference.

The GatorRent application will allow a user (users of the GatorRent application includes landlords looking for renters and students looking for housing) to either post houses, apartments, rooms that they are renting out, or for the user to filter through the currently available listings posted on the application. Our application will allow users to find listings based on important factors to consider as a student of San Francisco State University such as distance from the university, rent price. Thus, making GatorRent uniquely tailored for students and landlords as compared to other applications currently on the market and we will market it through societies at San Francisco State University to local students and to incoming international students.

GatorRent is accessible through the following link: http://54.183.44.255/

1.2 Committed Function List

1.2.1 **Users can:**

1.2.1.1 Login or register themselves a user of the website.

1.2.1.2 Easily search and browse for listings.

1.2.1.3 Post a place for rent.

1.2.1.4 Filter their search results by price, rent-type, price, postcode, if pets are allowed and if the rent-type is shared.

1.2.1.5 View additional details about any listing, including a map displaying the location of the listing.

1.2.1.6 Message any landlord about a specific listing and view all received messages on their dashboard.

1.2.1.7 Contact the Admin regarding any concerns through the Contact Us page.

1.2.1.8 **Unique feature:** users can filter listings respective to their distance from SFSU.

1.2.2 We ensured the proper safety precautions when users are visiting and using this site. Passwords will be encrypted.

2. USABILITY TEST PLAN

2.1 Test Objectives:

The main purpose of this usability test is to review the effectiveness, efficiency, and user satisfaction of our site. It is designed to observe and interpret the user's interactions with our product, assess its strengths and weaknesses, and allow us to take the proper steps to improve the user experience for future use. In this test plan, the main function being tested and analysed for usability is the search function.

The effectiveness of the site will be measured by the user's ability to correctly retrieve the needed results. For this specific test plan, the user will be provided with a task description that consists of three simple tasks to complete. If the user is able to successfully find the correct listings, based on the parameters of the task, the site is then deemed effective.

The efficiency of the site will be quantitatively measured by the length of time it takes the user to complete the task, in relation to accuracy and completeness. It will be influenced by the user's ability to adjust to the layout and search function provided on the site.

And lastly, the satisfaction of the site will be studied by the user's comfort and acceptability while completing a series of given tasks. Once the user has completed a task, they will be required to complete a questionnaire provided at the end of this section. This allows us to properly assess if our product is satisfying our targeted users.

2.2 Test Description:

2.2.1 System Startup & URL to be Tested

The system startup requires a computer, either a laptop or desktop, with access to the internet. The device must contain either a Windows 7 and up, Mac OS or Linux operating system and the type of browser to be utilized is either Chrome (version 74.0.3729.131) or Safari (version 12.1).

The URL of the system to be tested is found on http://54.183.44.255/results with the objective of searching for a listing. In the test plan, it is already assumed that the provided URL to be tested has already been loaded and that the site already consists of several posted listings.

2.2.2 Starting Point

The starting point will be a Chrome (version 74.0.3729.131) or Safari (version 12.1) web browser, set to the provided URL of the system being tested, http://54.183.44.255/results.

2.2.3 Intended Users

While this site is open to anyone, the intended users for this usability test plan are current or prospective SFSU students that are looking to find or provide a place to rent. They are average consumers, with some to little familiarity with computers and smartphones. More specifically, they are first-time users, naturally unregistered, who are not familiar with the layout, implementation, or effectiveness of the site, therefore allowing us to retrieve unbiased and accurate feedback on the usability of our search function.

2.2.4 What is Measured

Overall, we will measure the effectiveness, efficiency, and user satisfaction. However, for this specific test plan, we will mainly focus on user satisfaction. After asking users to perform a series of tasks, we will require the user to complete a Lickert scale on their experience with completing the tasks. Although the measure of satisfaction is subjective, our Lickert scale questions will allow us to acquire measurable and productive feedback on our site

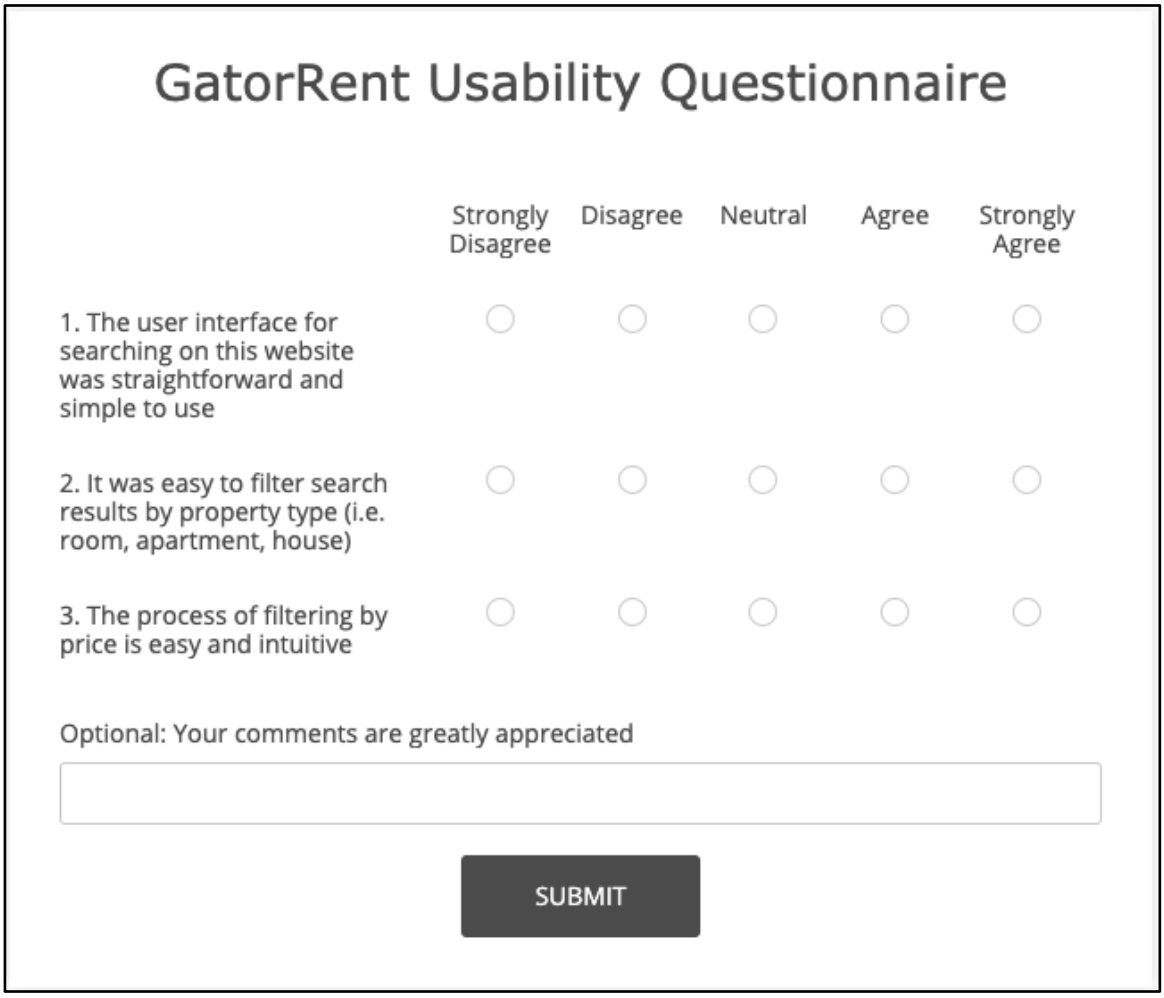
2.3 Usability Task Description

Before filling out the Likert questionnaire, the task testers will have to complete the following 3 tasks:

1. From the webpage, search and find a house.
2. From the webpage, search and find a room.
3. From the webpage, search and find an apartment for less than $700 per month.

TASK	**MACHINE STATE**
Task	Perform a search on GatorRent
Machine State	GatorRent webpage successfully loaded
Successful Completion Criteria	Listings that satisfy the search are successfully found
Benchmark	1 minute & 30 seconds

2.4 Questionnaire



GatorRent Usability Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. The user interface for searching on this website was straightforward and simple to use	○	○	○	○	○
2. It was easy to filter search results by property type (i.e. room, apartment, house)	○	○	○	○	○
3. The process of filtering by price is easy and intuitive	○	○	○	○	○

Optional: Your comments are greatly appreciated

SUBMIT

3. QA TEST PLAN

3.1 Test Objectives

3.1.1 The objective of this test is to check if the search bar is providing the correctly desired results.

3.2 HW and SW Setup

3.2.1 The hardware and software setup consist of a laptop with a Chrome browser (74.0.3729.131) with http://54.183.44.255/results opened.

3.3 Feature to be Tested

3.3.1 The feature to be tested in this plan is the search bar.

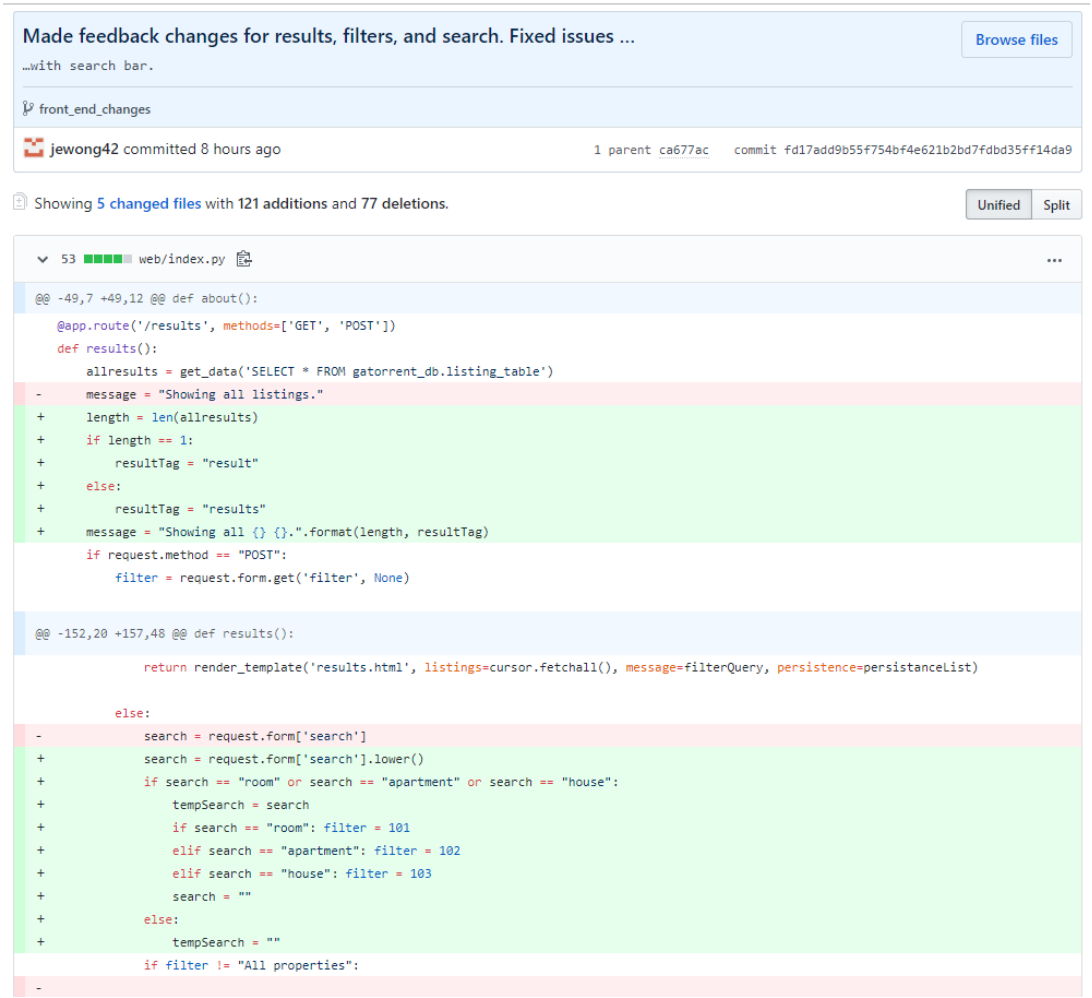
TEST #	TEST TITLE	TEST DESCRIPTION	TEST INPUT	EXPECTED CORRECT OUTPUT	TEST RESULTS	
					CHROME 74.0.3729.131	**SAFARI** 12.1
1	**Search Bar Property**	Open the webpage: http://54.183.44.255/results Change the dropdown search bar located near the top of the page to "All properties". In the search bar type "apartment". Click the blue "Search" button.	**Search bar drop down filter:** All properties **Search bar:** "apartment	3 Listings that fall under the "apartment" category **3 apartment results**	FAIL	FAIL
2	**Search Bar Location**	Open the webpage: http://54.183.44.255/results Change the dropdown search bar located near the top of the page to "All properties". In the search bar type "San". Click the blue "Search" button.	**Search bar dropdown filter:** All properties **Search:** "San"	5 Listings that fall under the category "san" **5 listings results [3 apartments, 1 house, 1 room]**	PASS	PASS
3	**Search Bar Validation**	Open the webpage: http://54.183.44.255/results Change the dropdown search bar located near the top of the page to "All properties". In the search bar type "12344?". Click the blue "Search" button.	**Search bar dropdown filter:** All properties **Search:** "12344?"	**Alert:** "Search must not contain special characters and must be less than 40 characters."	FAIL	FAIL

4. CODE REVIEW

4.1 Coding Style

4.1.1 As part of using VSCode or Atom for development, Pylint is the suggested default code analysis and quality check. Python Coding Style, written and published by Mozilla is a good summary of coding style best practices being followed right now. This is the complete Style Guide as specified by Python.org, developers are expected to glanced through and try to enforce rules as stated in this guide.

4.1.2 Code related to the feature used in the QA and usability test:



Made feedback changes for results, filters, and search. Fixed issues ... Browse files

...with search bar.

front_end_changes

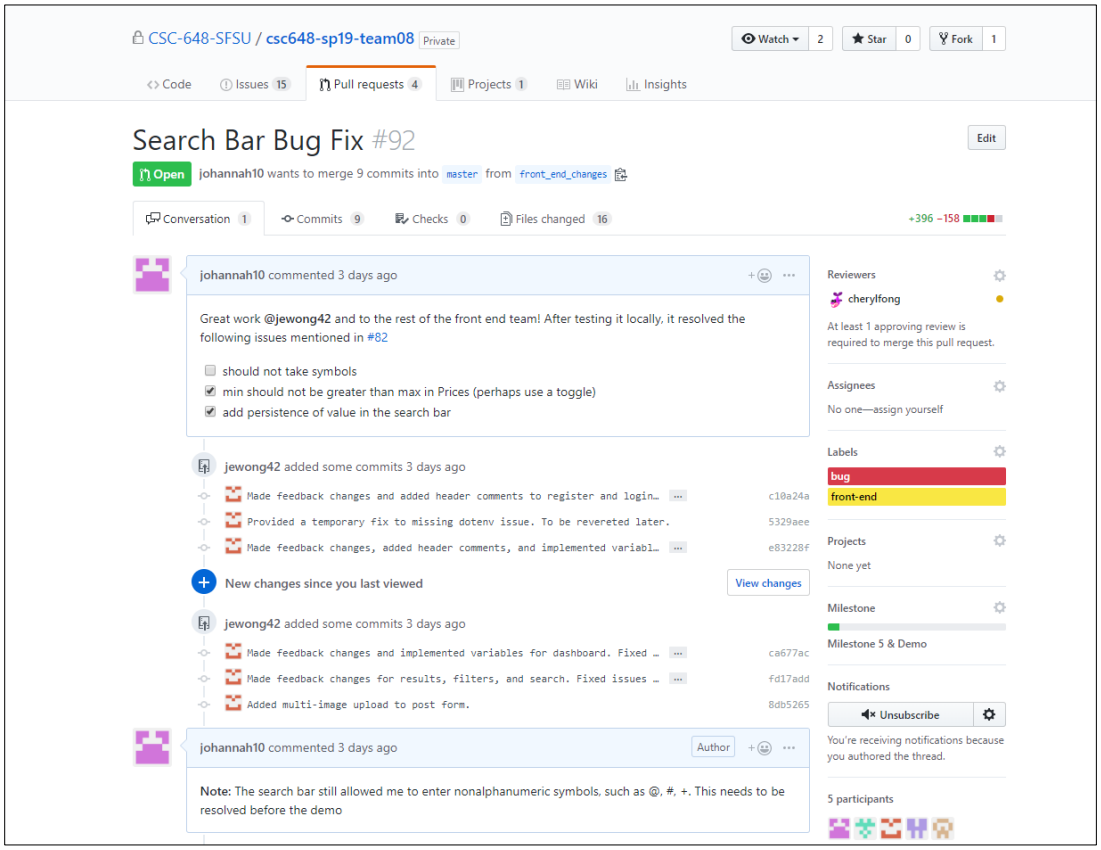
jewong42 committed 8 hours ago 1 parent ca677ac commit fd17add9b55f754bf4e621b2bd7fdbd35ff14da9

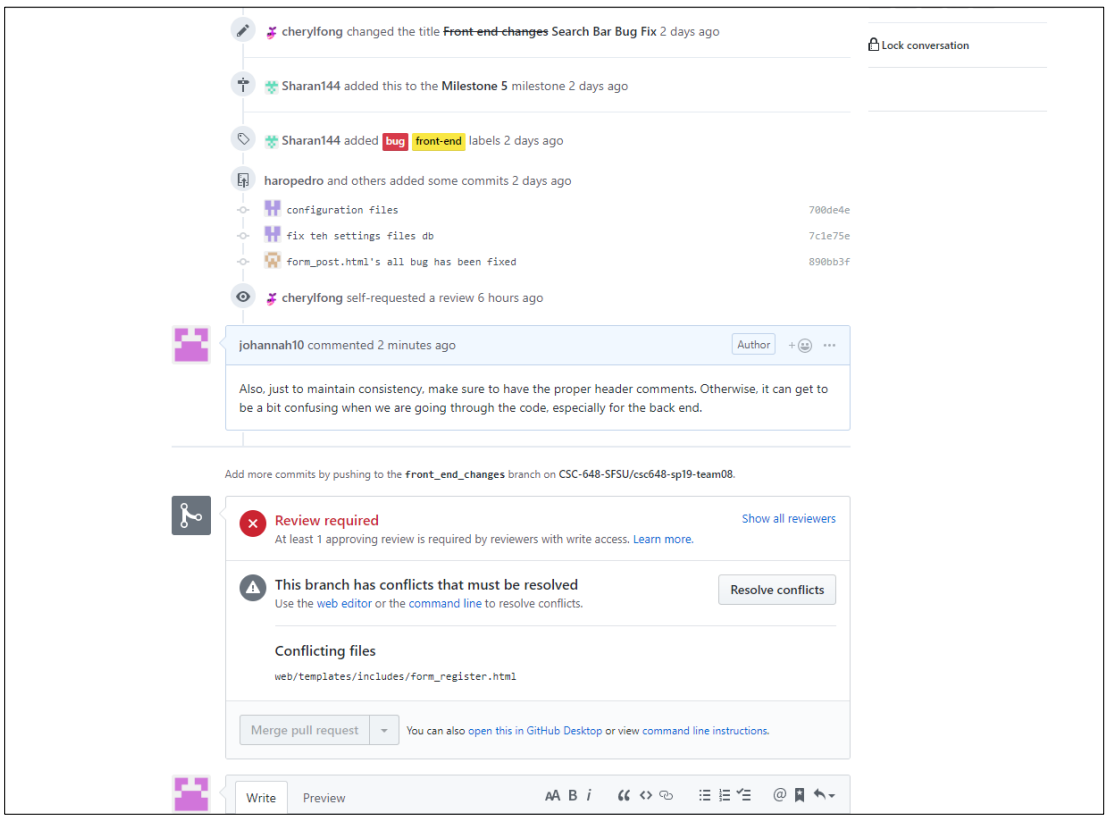
Showing 5 changed files with 121 additions and 77 deletions. Unified Split

53 web/index.py

```
@@ -49,7 +49,12 @@ def about():
    @app.route('/results', methods=['GET', 'POST'])
    def results():
        allresults = get_data('SELECT * FROM gatorrent_db.listing_table')
-       message = "Showing all listings."
+       length = len(allresults)
+       if length == 1:
+           resultTag = "result"
+       else:
+           resultTag = "results"
+       message = "Showing all {} {}.".format(length, resultTag)
        if request.method == "POST":
            filter = request.form.get('filter', None)

@@ -152,20 +157,48 @@ def results():
                return render_template('results.html', listings=cursor.fetchall(), message=filterQuery, persistence=persistanceList)

            else:
-               search = request.form['search']
+               search = request.form['search'].lower()
+               if search == "room" or search == "apartment" or search == "house":
+                   tempSearch = search
+                   if search == "room": filter = 101
+                   elif search == "apartment": filter = 102
+                   elif search == "house": filter = 103
+                   search = ""
+               else:
+                   tempSearch = ""
                if filter != "All properties":
-
```

```
-
                searchresults = get_data("SELECT * FROM gatorrent_db.listing_table WHERE CONCAT_WS('', street, city, state, country, postcode) L
-
-               filterToString = get_data("SELECT DISTINCT type FROM gatorrent_db.rent_type_table WHERE rent_type_id = {}".format(filter))
-               message = "Showing all results for '{}' in {}".format(search, filterToString)
+               filterName = get_data("SELECT DISTINCT type FROM gatorrent_db.rent_type_table WHERE rent_type_id = {}".format(filter))[0]
+               filterToString = filterName[0]
+               length = len(searchresults)
+               if length == 1:
+                   resultTag = "result"
+               else:
+                   resultTag = "results"
+               if search == "":
+                   message = "Showing {} {} in {}".format(length, resultTag, filterToString)
+                   search == tempSearch
+               elif search == "house" or search == "room" or search == "apartment":
+                   message = "test"
+               else:
+                   message = "Showing {} {} for '{}' in {}".format(length, resultTag, search, filterToString)
+               if tempSearch != "":
+                   search = tempSearch
                return render_template('results.html', listings=searchresults, filter=filter, search=search, message=message, persistence=None)
            else:
-
                searchresults = get_data("SELECT * FROM gatorrent_db.listing_table WHERE CONCAT_WS('', street, city, state, country, postcode) L
-
-               message = "Showing all results for '{}'".format(search)
-               return render_template('results.html', listings=searchresults, message=message, persistence=None)
+               length = len(searchresults)
+               if length == 1:
+                   resultTag = "result"
+               else:
+                   resultTag = "results"
+               if search == "":
+                   message = "Showing all {} {}.".format(length, resultTag)
+               else:
+                   message = "Showing {} {} for '{}'".format(length, resultTag, search)
+               if tempSearch != "":
+                   search = tempSearch
+               return render_template('results.html', listings=searchresults, message=message, search=search, persistence=None)
       return render_template('results.html', listings=allresults, message=message, persistence=None)


  # Auth pages
```


CSC-648-SFSU / csc648-sp19-team08 Private
Watch 2
Star 0
Fork 1
Code
Issues 15
Pull requests 4
Projects 1
Wiki
Insights
Search Bar Bug Fix #92
Edit
Open
johannah10 wants to merge 9 commits into master from front_end_changes
Conversation 1
Commits 9
Checks 0
Files changed 16
+396 −158
johannah10 commented 3 days ago
Great work @jewong42 and to the rest of the front end team! After testing it locally, it resolved the following issues mentioned in #82
should not take symbols
min should not be greater than max in Prices (perhaps use a toggle)
add persistence of value in the search bar
jewong42 added some commits 3 days ago
Made feedback changes and added header comments to register and login… c10a24a
Provided a temporary fix to missing dotenv issue. To be revereted later. 5329aee
Made feedback changes, added header comments, and implemented variabl… e83228f
New changes since you last viewed
View changes
jewong42 added some commits 3 days ago
Made feedback changes and implemented variables for dashboard. Fixed … ca677ac
Made feedback changes for results, filters, and search. Fixed issues … fd17add
Added multi-image upload to post form. 8db5265
johannah10 commented 3 days ago
Author
Note: The search bar still allowed me to enter nonalphanumeric symbols, such as @, #, +. This needs to be resolved before the demo
Reviewers
cherylfong
At least 1 approving review is required to merge this pull request.
Assignees
No one—assign yourself
Labels
bug
front-end
Projects
None yet
Milestone
Milestone 5 & Demo
Notifications
Unsubscribe
You're receiving notifications because you authored the thread.
5 participants


cherylfong changed the title ~~Front end changes~~ Search Bar Bug Fix 2 days ago
Lock conversation
Sharan144 added this to the Milestone 5 milestone 2 days ago
Sharan144 added bug front-end labels 2 days ago
haropedro and others added some commits 2 days ago
configuration files 700de4e
fix teh settings files db 7c1e75e
form_post.html's all bug has been fixed 890bb3f
cherylfong self-requested a review 6 hours ago
johannah10 commented 2 minutes ago
Author
Also, just to maintain consistency, make sure to have the proper header comments. Otherwise, it can get to be a bit confusing when we are going through the code, especially for the back end.
Add more commits by pushing to the front_end_changes branch on CSC-648-SFSU/csc648-sp19-team08.
Review required
At least 1 approving review is required by reviewers with write access. Learn more.
Show all reviewers
This branch has conflicts that must be resolved
Use the web editor or the command line to resolve conflicts.
Resolve conflicts
Conflicting files
web/templates/includes/form_register.html
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Write
Preview

5. SELF-CHECK ON BEST PRACTICES FOR SECURITY

5.1 Major Assets We're Protecting

5.1.1 The major assets we are protecting are relative to the user's personal information. User Confidentiality is a vital element that is being taken into account during the implementation of this site, therefore passwords are protected and encrypted via the Bcrypt library. Passwords are hashed before being stored on the database.

5.1.2 To protect our site from spam, automated accesses and extractions of data from websites, we are using reCAPTCHA. This form of captcha provides protection while also letting valid users navigate through our website with ease.

5.1.3 To secure our images, we used the **secure_filename()** function, as recommended by the **Flask Documentation Release 0.9 Dev**, to secure a filename before we stored it.

5.1.4 While we currently haven't done so, we plan to validate our search bar input. To provide protection against potential SQL injection, form inputs are validated via Bootstrap's Form Validation. Functions that handle SQL and user input have taken into account the vulnerabilities, as stated in W3Schools .

form_filters.html

```
6  -->
7
8  <div class="container py-3">
9     <h3 class="text-center">Filters</h3>
10    <form class="form-group" method="post" action="{{ url_for('results') }}">
11       <ul class="list-group py-3">
12          <li class="list-group-item justify-content-between">
13             <!-- Price range -->
14             <!-- Required data: 'minPrice', 'maxPrice' -->
15             <div class="container">
16                <h6>Price range</h6>
17                <div class="form-inline py-2 mx-2 justify-content-between">
18                   <input type="number" class="form-control" name="minprice" placeholder="min" style="width:40%"
19                   {% if minPrice %} placeholder={{minPrice}} {% endif %}>
20                   <small class="mx-2">-</small>
21                   <input type="number" class="form-control" name="maxprice" placeholder="max" style="width:40%"
22                   {% if maxPrice %} placeholder={{maxPrice}} {% endif %}>
23                </div>
24             </div>
25          </li>
26          <li class="list-group-item justify-content-between">
27             <!-- Location -->
28             <!-- Required data: 'city', 'zipcode' -->
```

This is an example of how filter inputs are validated. In line 21, **`type="number"`** specifies that only numbers can be typed into the field.

```
<!-- Login Form for the Website -->
<form class="w-25" data-toggle="validator" role="form" method="post">
   <h3>Register for GatorRent</h3>
      <div class="form-group my-4">
         <div class="form-group">
            <label>Username <small class="text-muted">(Maximum 30 characters)</small></label>
            <input type="text" id="username" maxlength="30" class="form-control" placeholder="Enter username" required autofocus>
         </div>
         <div class="form-group">
            <label>Password <small class="text-muted">(Maximum 15 characters)</small></label>
            <input type="text" id="inputPassword" maxlength="15" class="form-control" placeholder="Enter password" required>
          </div>
          <div class="form-group">
            <label>Confirm Password</label>
            <input type="text" id="inputPasswordConf" maxlength="15" class="form-control" data-match="#inputPassword" placeholder="
         </div>
         <div class="form-group">
            <label>Email</label>
            <input type="email" id="email" class="form-control" placeholder="Enter email" required >
         </div>
         <div class="form-check">
            <input type="checkbox" class="form-check-input" id="agree">
```

In the image above, line 12 restricts the user from having a password that is more than 15 characters. This requirement is set so that an SQL injection of more than 15 characters will not work.

6. SELF-CHECK: ADHERENCE TO ORIGINAL NON-FUNCTIONAL SPECS

#	REQUIREMENT	STATUS
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).	ON TRACK
2.	Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers.	ON TRACK
3.	Selected application functions must render well on mobile devices.	ON TRACK
4.	Data shall be stored in the team's chosen database technology on the team's deployment server.	ON TRACK
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.	ON TRACK
7.	The language used shall be English.	DONE
8.	Application shall be very easy to use and intuitive.	ON TRACK
9.	Google analytics shall be added.	ON TRACK
10.	No email clients shall be allowed.	DONE
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)	ON TRACK
13.	Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator.	ON TRACK
14.	Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.	ON TRACK

15.	**The website shall <u>prominently</u> display the following <u>exact</u> 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

Software Engineering CSC 648/848 Section 02 Spring 2019

Milestone 5

Team 08

Cheryl Fong
::
cfong8@mail.sfsu.edu

Jerry Wong

Chirag Agarwal

Harold Pedroso

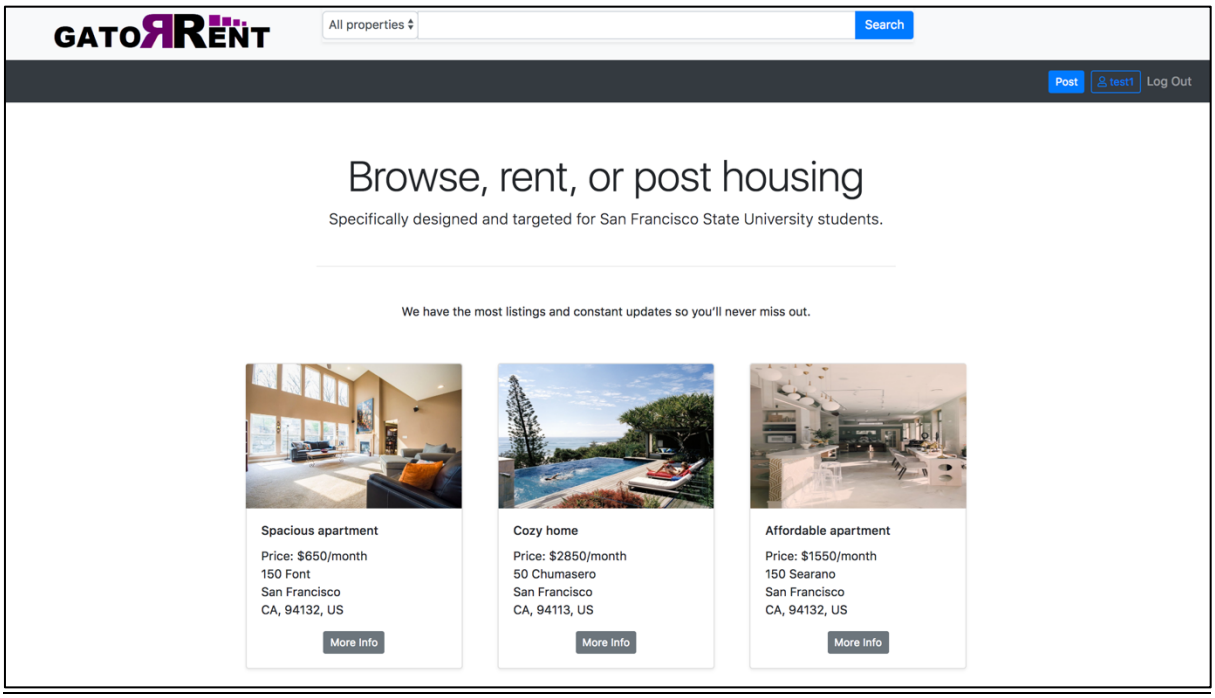
Johanna Hadgu

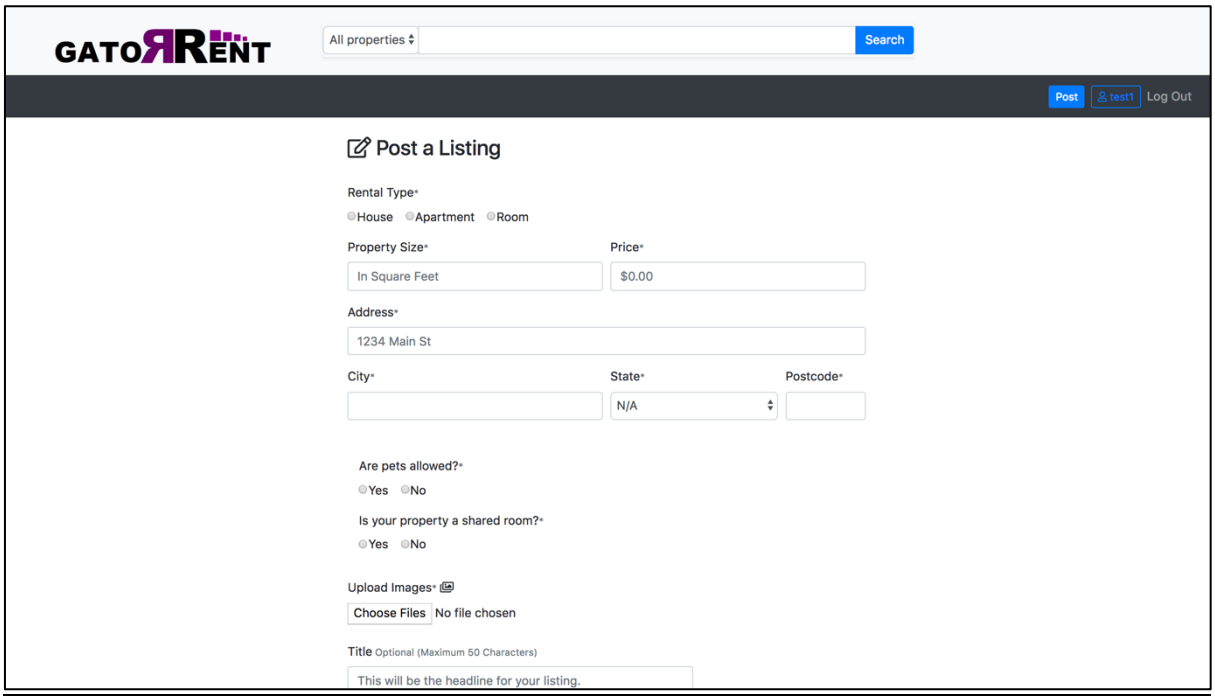
Sharan Lao

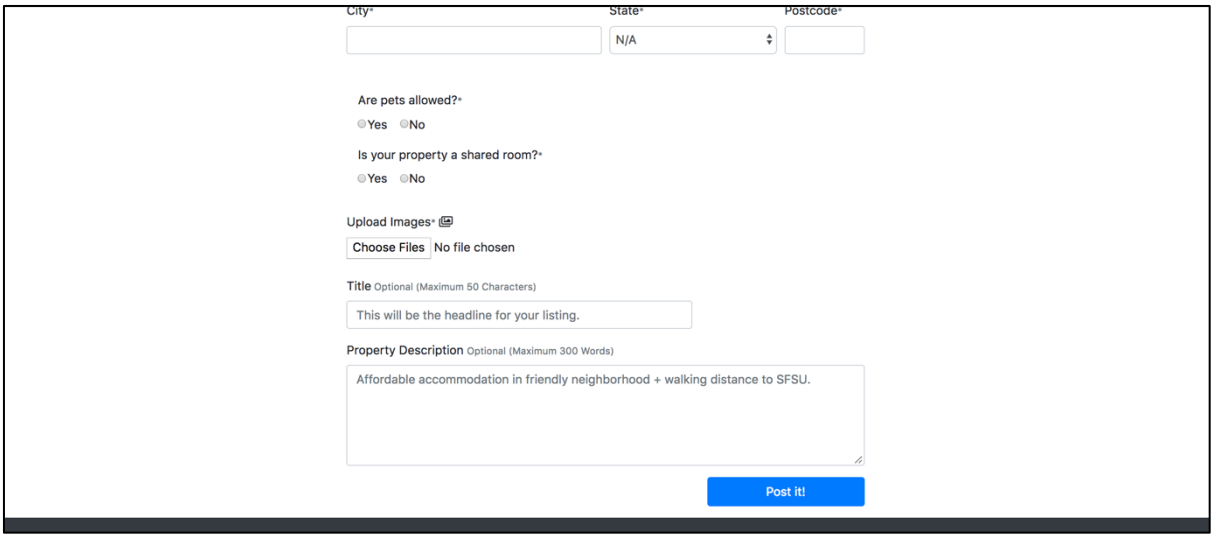
Ermias Haile

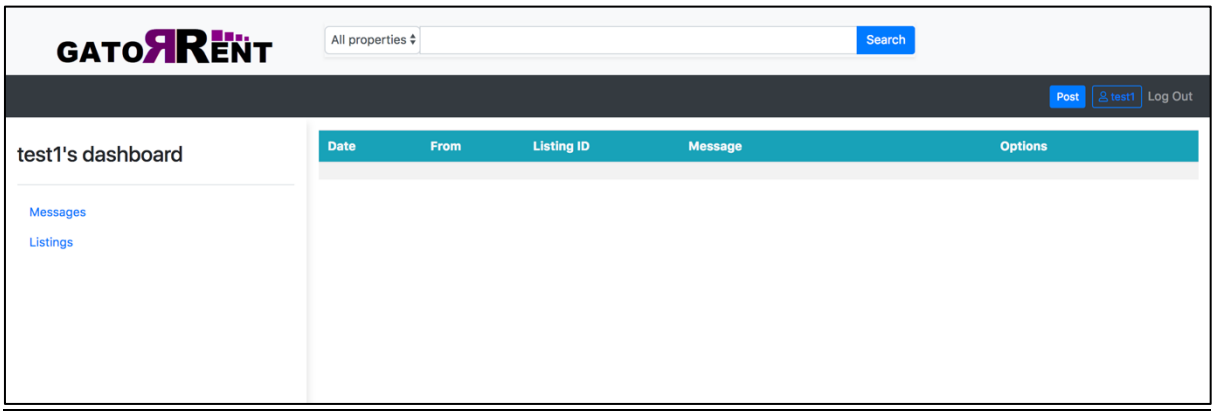
FROZEN STATE

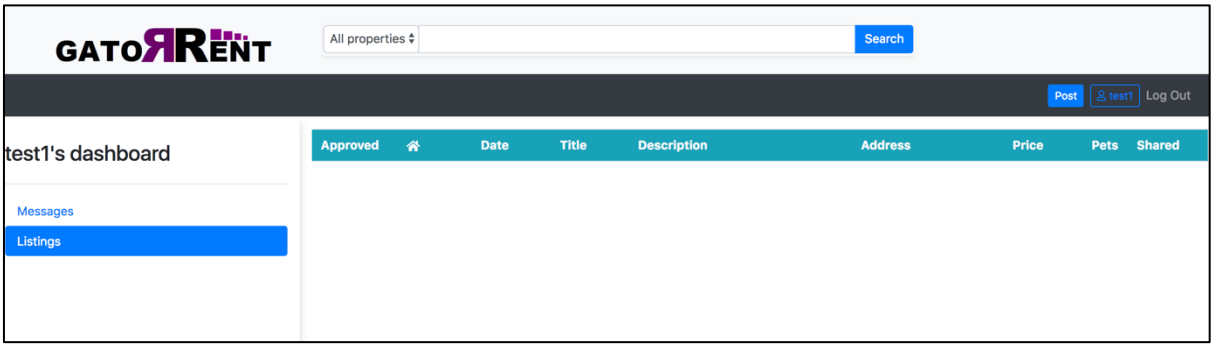
1. PRODUCT SCREENSHOTS

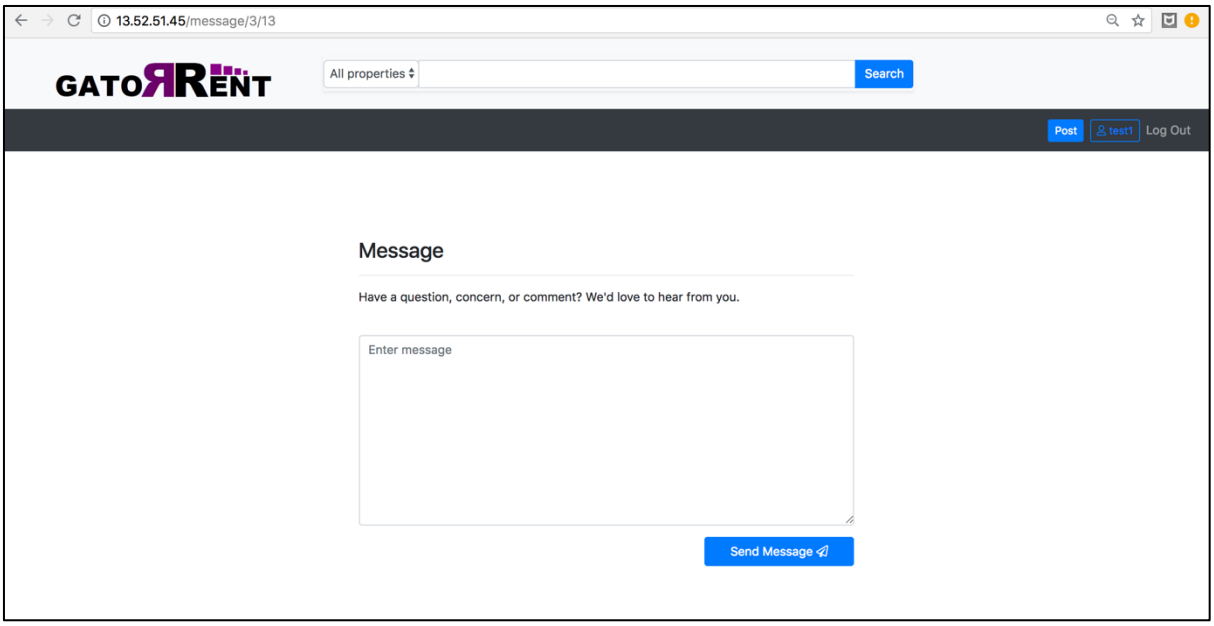


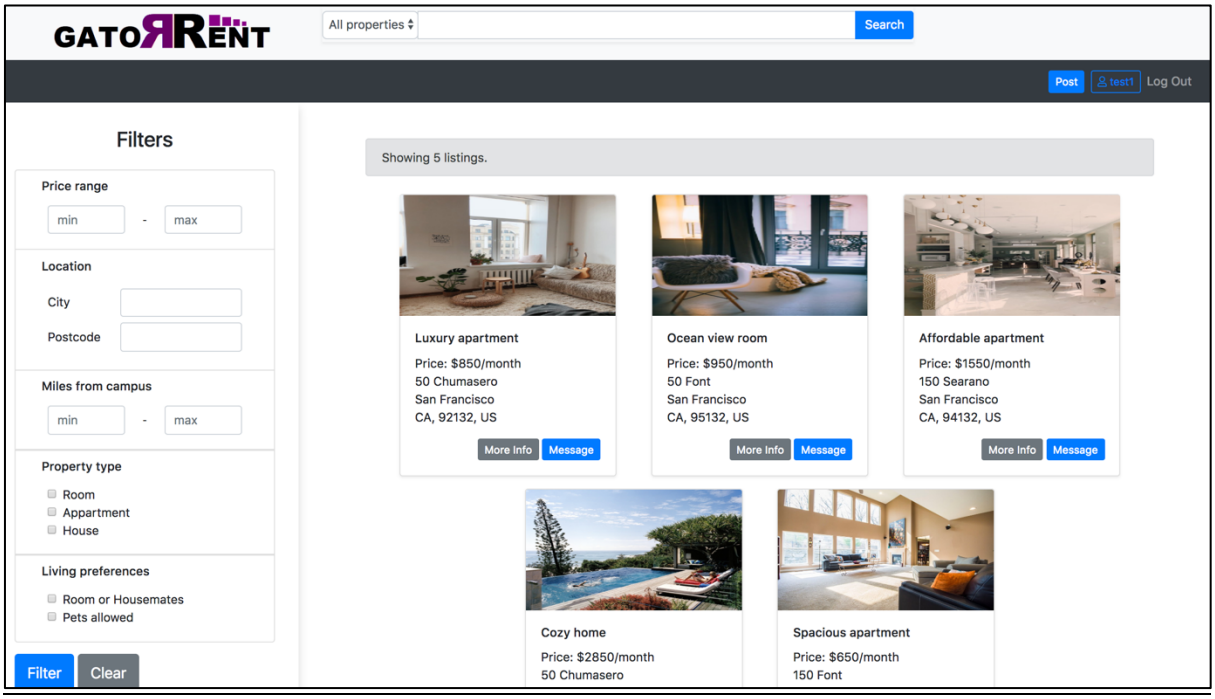


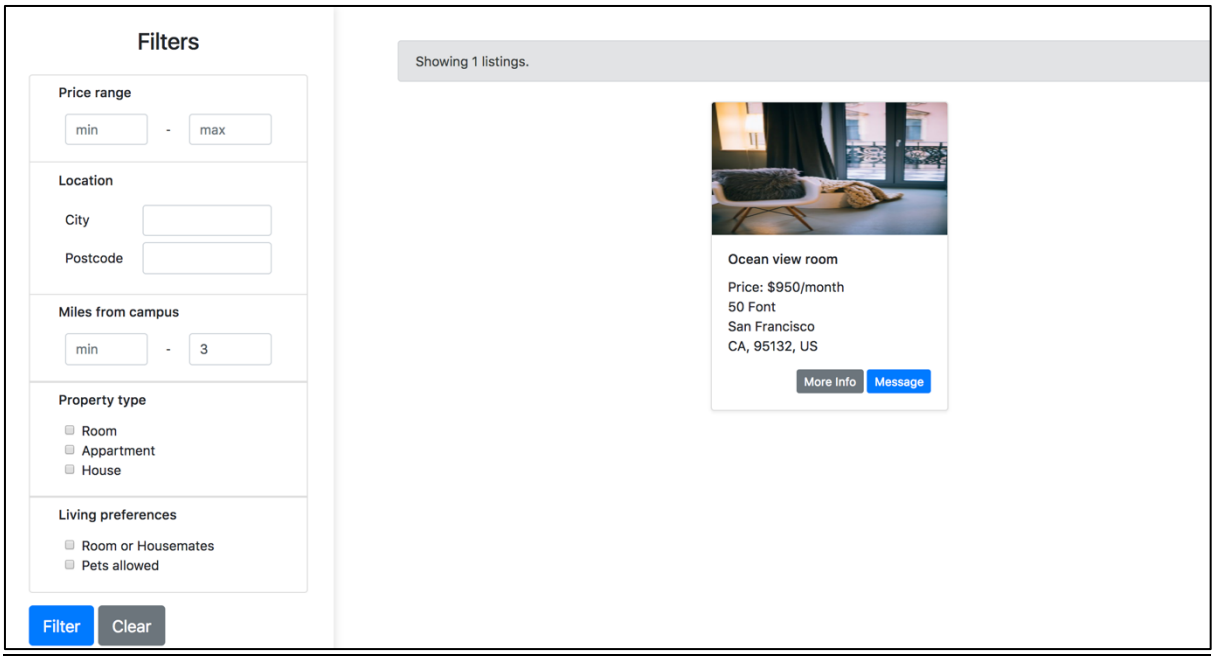

City*
State*
Postcode*
N/A
Are pets allowed?*
Yes
No
Is your property a shared room?*
Yes
No
Upload Images*
Choose Files
No file chosen
Title Optional (Maximum 50 Characters)
This will be the headline for your listing.
Property Description Optional (Maximum 300 Words)
Affordable accommodation in friendly neighborhood + walking distance to SFSU.
Post it!

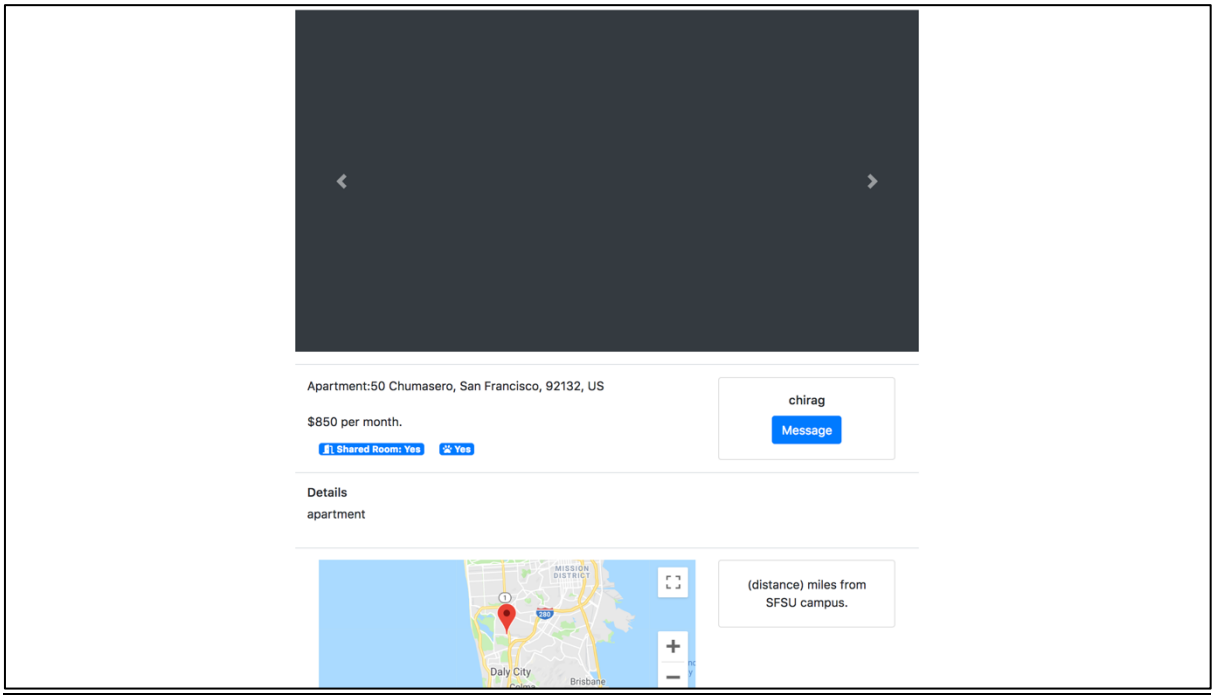

GATORRENT
All properties
Search
Post
test1
Log Out
test1's dashboard
Messages
Listings
Date
From
Listing ID
Message
Options


GATORRENT
All properties
Search
Post
test1
Log Out
test1's dashboard
Messages
Listings
Approved
Date
Title
Description
Address
Price
Pets
Shared


13.52.51.45/message/3/13
GATORRENT
All properties
Search
Post
test1
Log Out
Message
Have a question, concern, or comment? We'd love to hear from you.
Enter message
Send Message

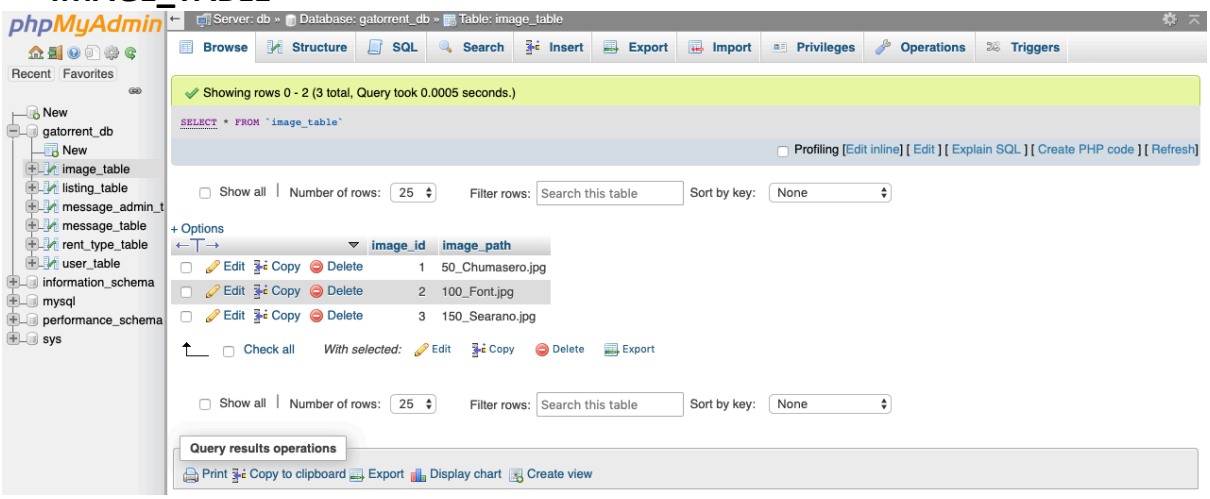

GATORRENT
All properties
Search
Post
test1
Log Out
Filters
Price range
min
max
Location
City
Postcode
Miles from campus
min
max
Property type
Room
Appartment
House
Living preferences
Room or Housemates
Pets allowed
Filter
Clear
Showing 5 listings.
Luxury apartment
Price: $850/month
50 Chumasero
San Francisco
CA, 92132, US
More Info
Message
Ocean view room
Price: $950/month
50 Font
San Francisco
CA, 95132, US
More Info
Message
Affordable apartment
Price: $1550/month
150 Searano
San Francisco
CA, 94132, US
More Info
Message
Cozy home
Price: $2850/month
50 Chumasero
Spacious apartment
Price: $650/month
150 Font


Filters
Price range
min
-
max
Location
City
Postcode
Miles from campus
min
-
3
Property type
Room
Appartment
House
Living preferences
Room or Housemates
Pets allowed
Filter
Clear
Showing 1 listings.
Ocean view room
Price: $950/month
50 Font
San Francisco
CA, 95132, US
More Info
Message

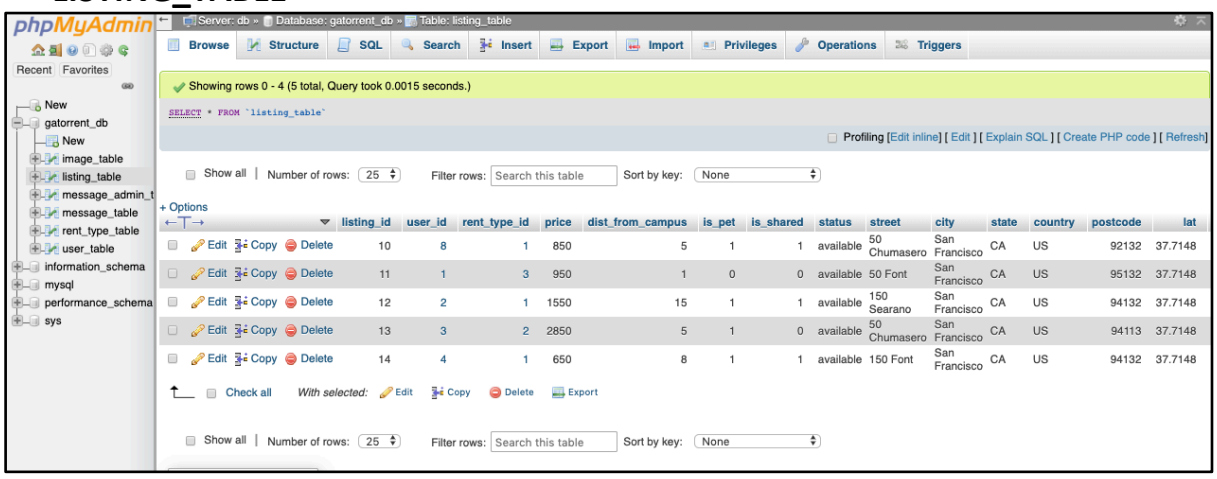

Apartment:50 Chumasero, San Francisco, 92132, US
$850 per month.
Shared Room: Yes
Yes
chirag
Message
Details
apartment
(distance) miles from SFSU campus.
Daly City
Brisbane

2. KEY DB TABLES SCREENSHOTS

IMAGE_TABLE



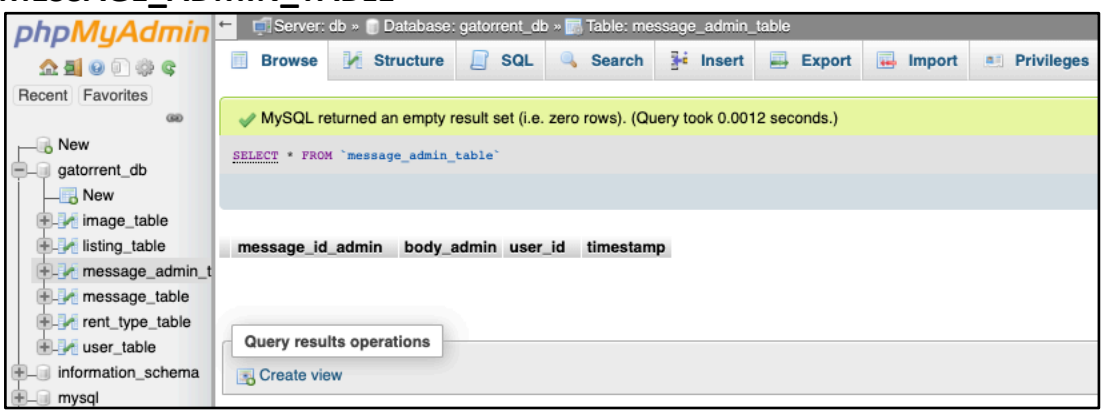
LISTING_TABLE



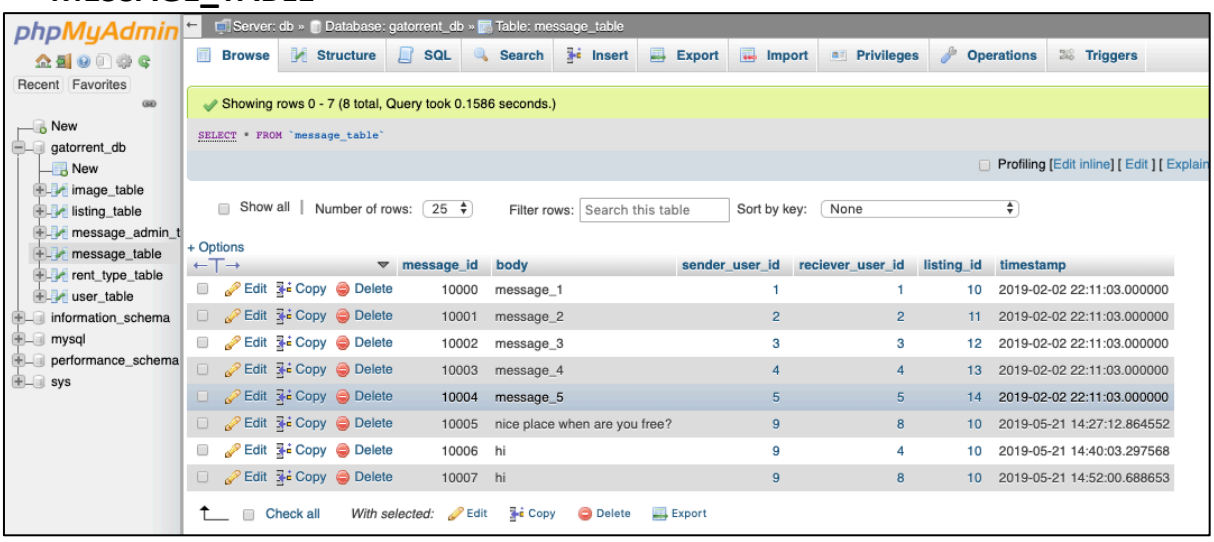
LISTING_TABLE (continued)

lat	long	isApproved	image_id	thumbnail_path	description	title	timestamp
37.7148	-122.474	1	1	https://marketplace.canva.com/MADGyVipG3g/4/screen...	apartment	Luxury apartment	2019-05-21 14:25:58
37.7148	-122.474	1	1	https://marketplace.canva.com/MADGybsz64k/4/thumbn...	room	Ocean view room	2019-04-04 01:11:31
37.7148	-122.474	1	3	https://marketplace.canva.com/MADGvrjQHI8/7/screen...	apartment	Affordable apartment	2019-04-04 01:11:07
37.7148	-122.474	1	1	https://marketplace.canva.com/MAAgcMpQ-oY/1/screen...	house	Cozy home	2019-04-04 01:10:46
37.7148	-122.474	1	2	https://marketplace.canva.com/MADGvkJj6KQ/7/screen...	apartment	Spacious apartment	2019-04-04 01:10:17

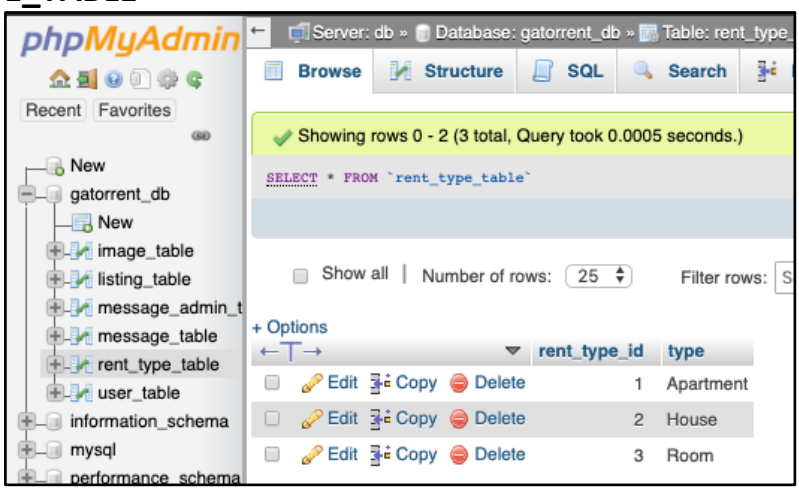
MESSAGE_ADMIN_TABLE



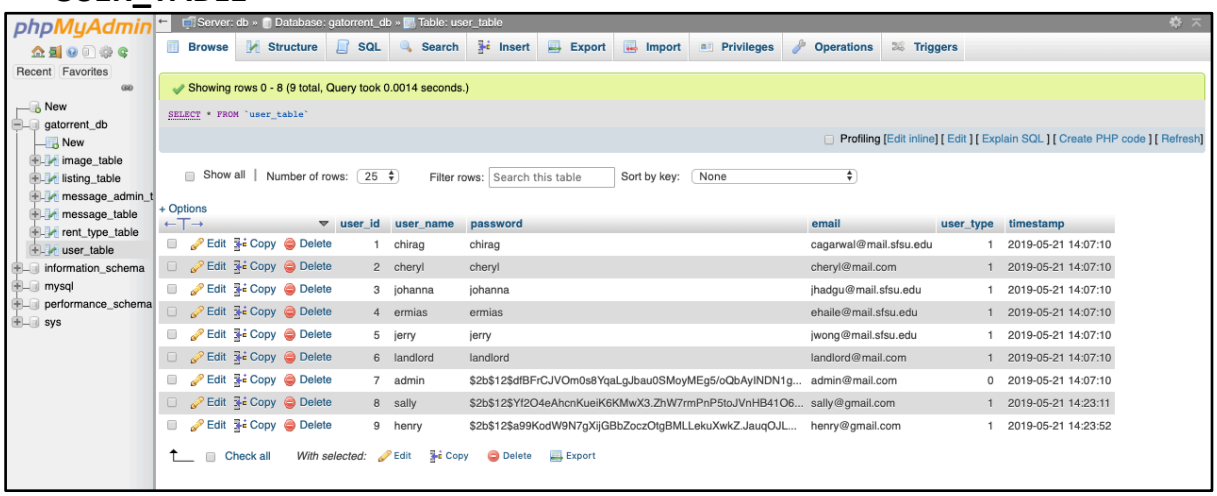
MESSAGE_TABLE



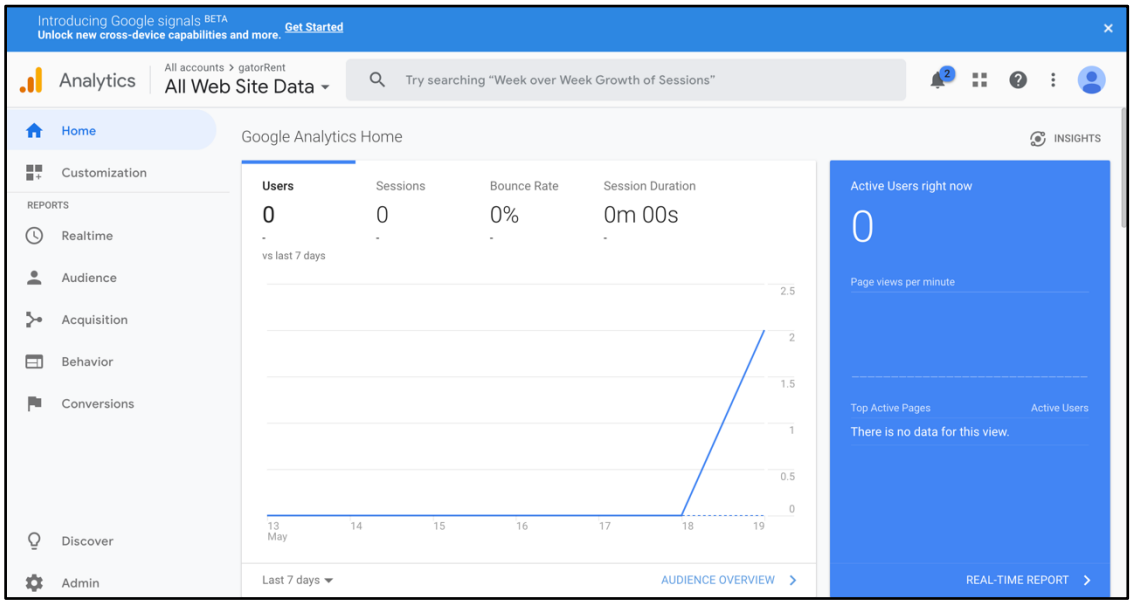
RENT_TYPE_TABLE

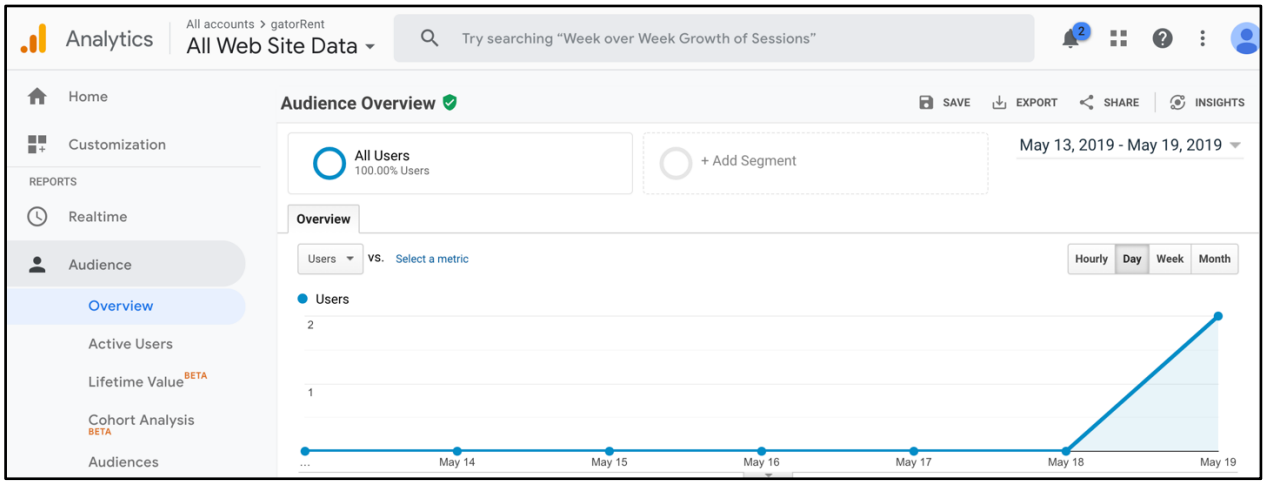


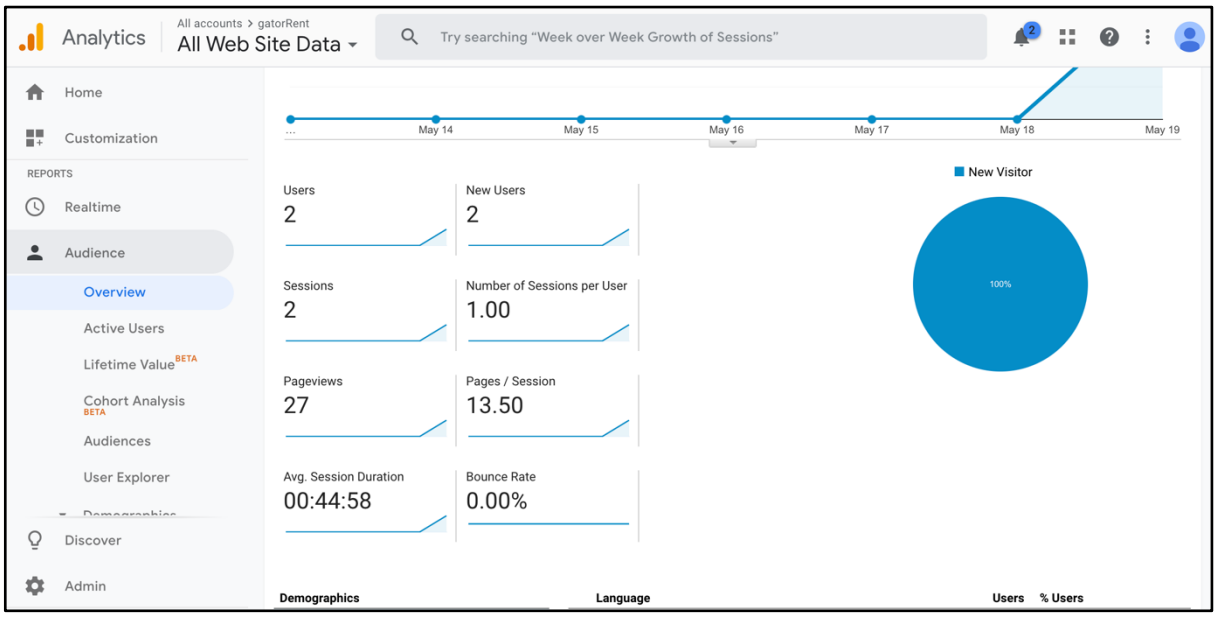
USER_TABLE


phpMyAdmin
Recent Favorites
New
gatorrent_db
New
image_table
listing_table
message_admin_t
message_table
rent_type_table
user_table
information_schema
mysql
performance_schema
sys
Server: db » Database: gatorrent_db » Table: user_table
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Showing rows 0 - 8 (9 total, Query took 0.0014 seconds.)
SELECT * FROM `user_table`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
user_id user_name password email user_type timestamp
Edit Copy Delete 1 chirag chirag cagarwal@mail.sfsu.edu 1 2019-05-21 14:07:10
Edit Copy Delete 2 cheryl cheryl cheryl@mail.com 1 2019-05-21 14:07:10
Edit Copy Delete 3 johanna johanna jhadgu@mail.sfsu.edu 1 2019-05-21 14:07:10
Edit Copy Delete 4 ermias ermias ehaile@mail.sfsu.edu 1 2019-05-21 14:07:10
Edit Copy Delete 5 jerry jerry jwong@mail.sfsu.edu 1 2019-05-21 14:07:10
Edit Copy Delete 6 landlord landlord landlord@mail.com 1 2019-05-21 14:07:10
Edit Copy Delete 7 admin $2b$12$dfBFrCJVOm0s8YqaLgJbau0SMoyMEg5/oQbAylNDN1g... admin@mail.com 0 2019-05-21 14:07:10
Edit Copy Delete 8 sally $2b$12$Yf2O4eAhcnKueiK6KMwX3.ZhW7rmPnP5toJVnHB41O6... sally@gmail.com 1 2019-05-21 14:23:11
Edit Copy Delete 9 henry $2b$12$a99KodW9N7gXijGBbZoczOtgBMLLekuXwkZ.JauqOJL... henry@gmail.com 1 2019-05-21 14:23:52
Check all With selected: Edit Copy Delete Export

3. GOOGLE ANALYTICS PLOT OF OUR WWW SITE

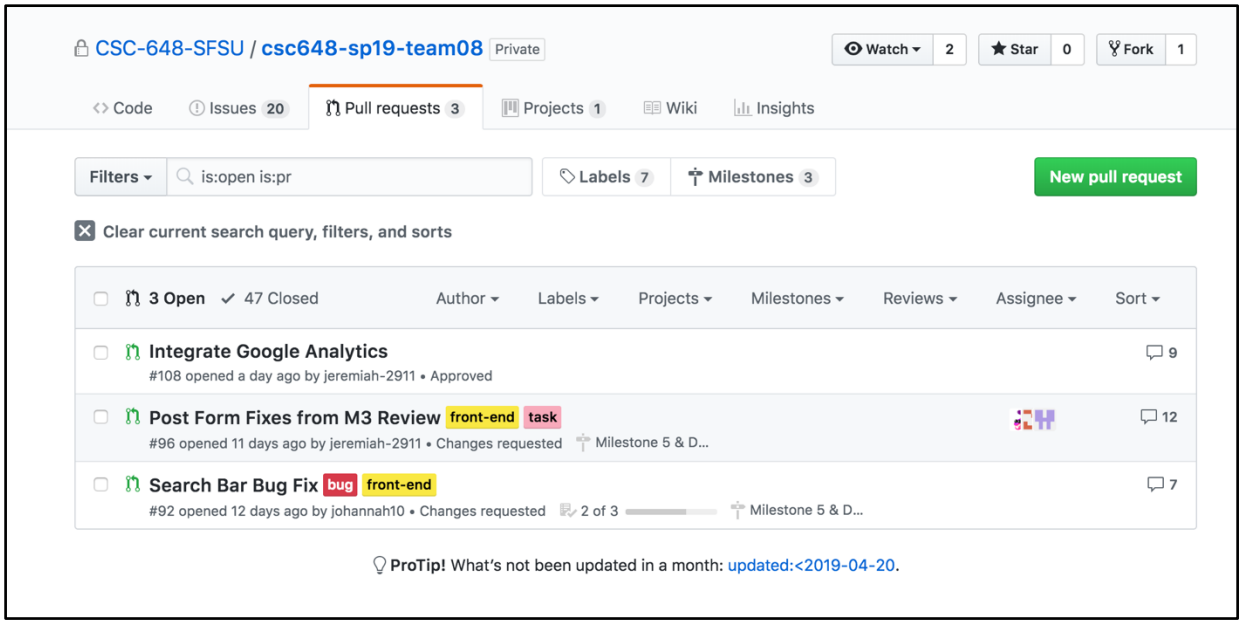




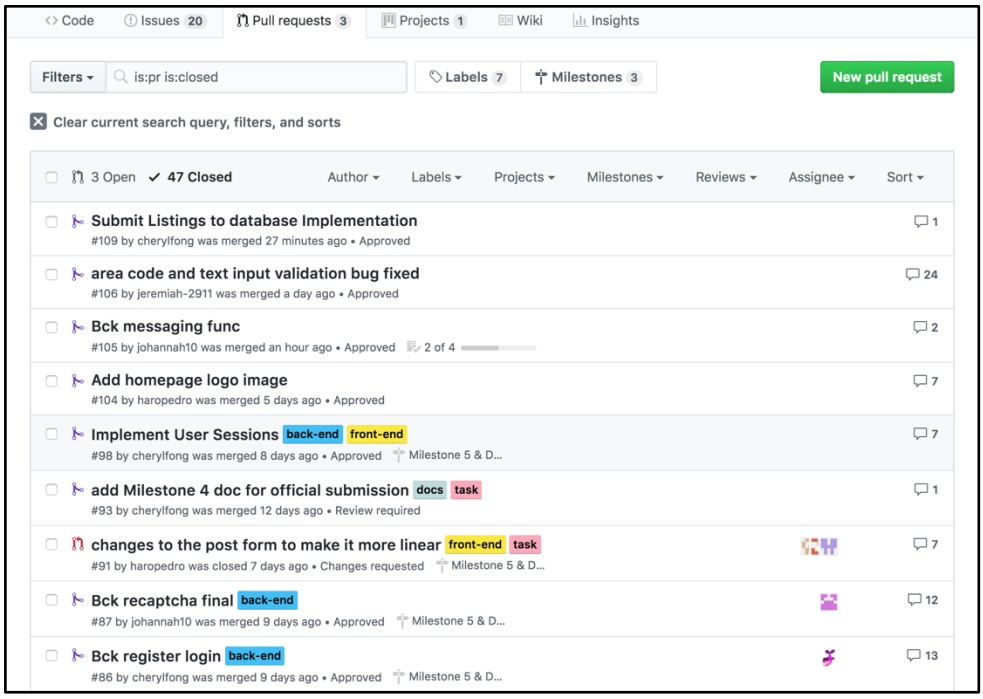


4. SCREENSHOTS OF OUR TASK MANAGEMENT

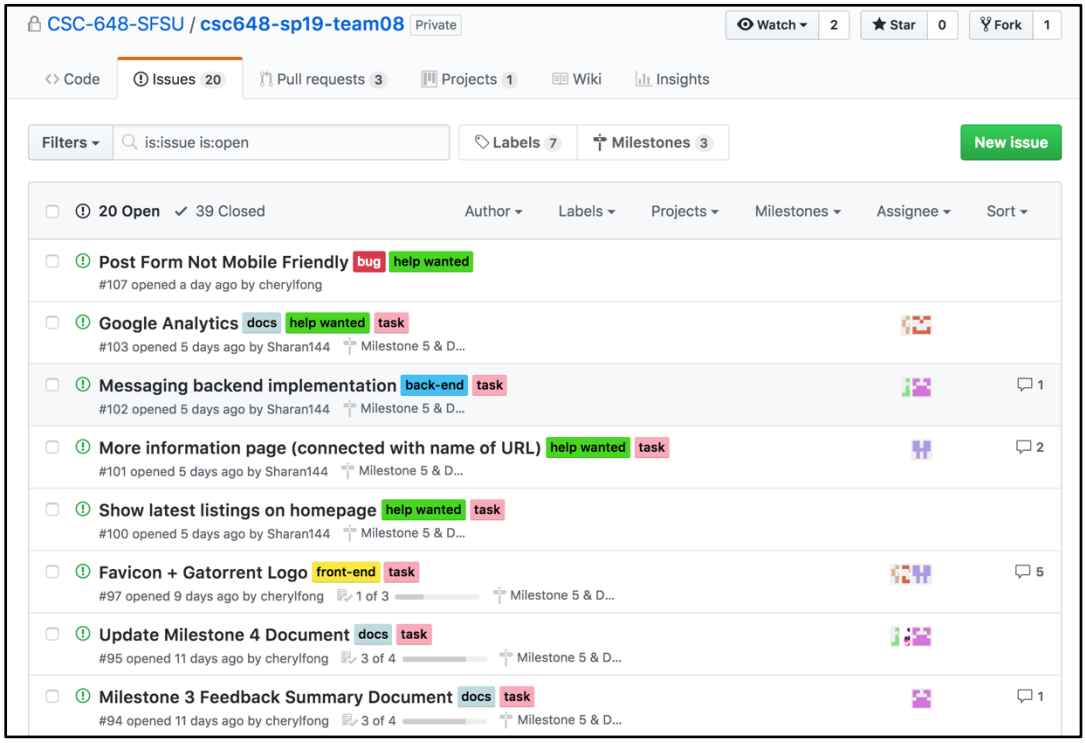
OPEN PULL REQUESTS



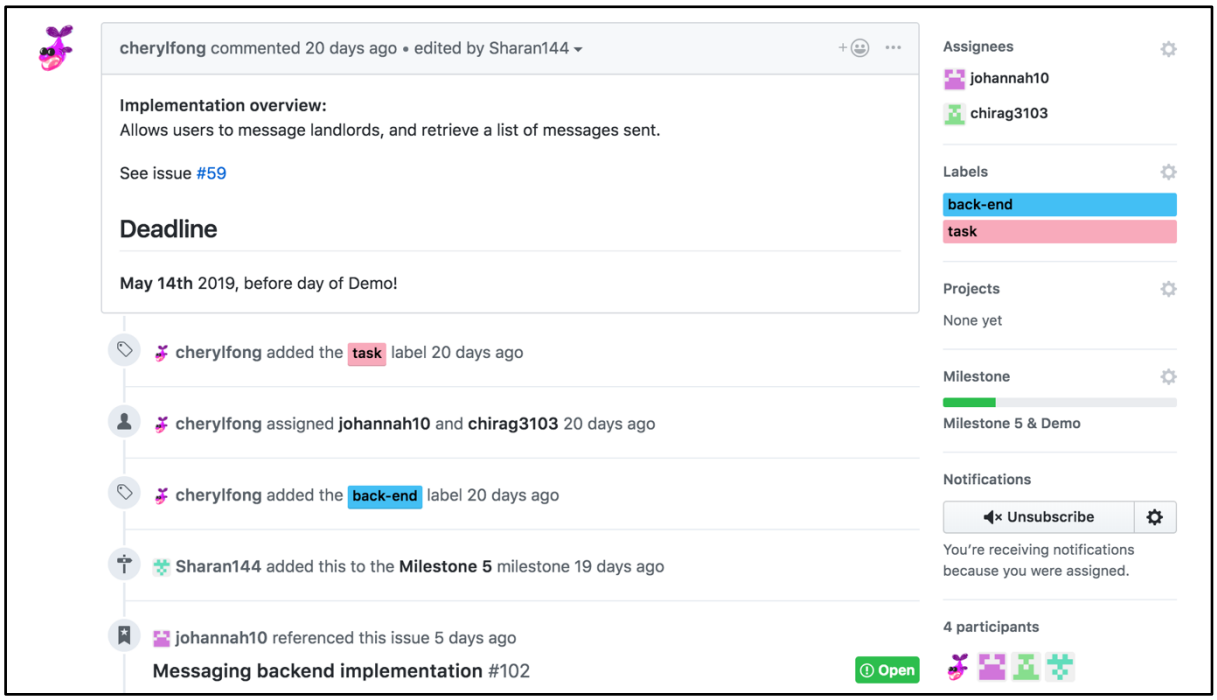
CLOSED PULL REQUESTS



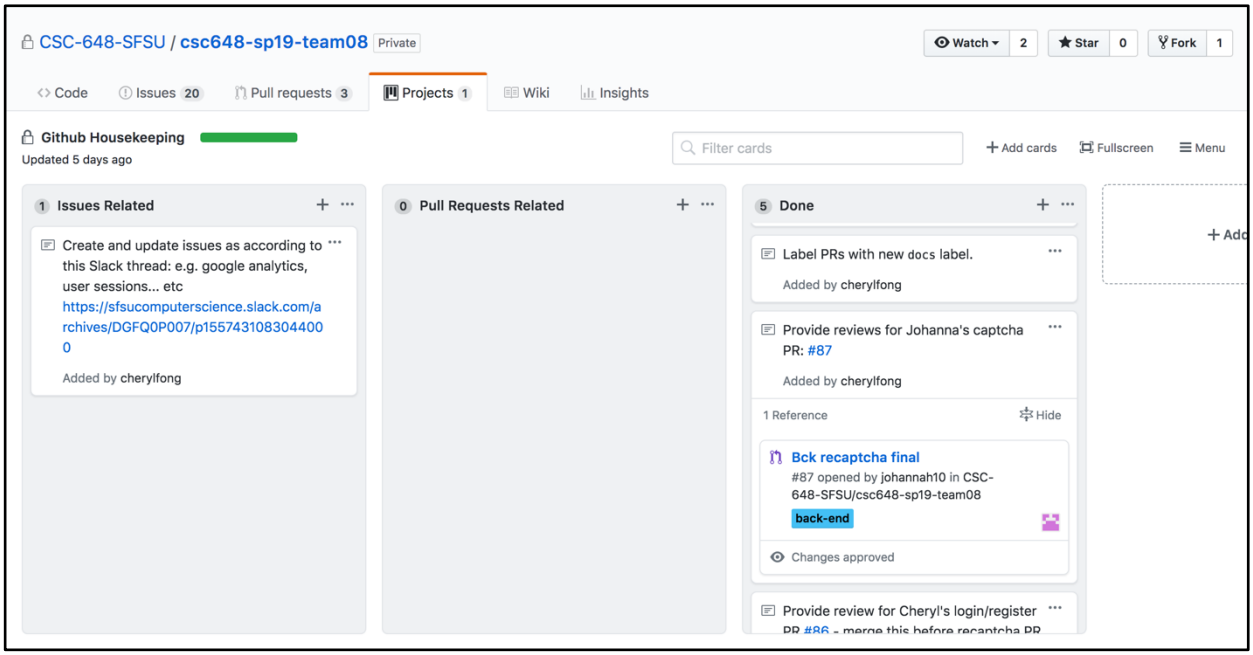
OPEN ISSUES



DELEGATING ASSIGNMENTS WITH DUE DATES



PROJECTS


CSC-648-SFSU / csc648-sp19-team08
Private
Watch 2
Star 0
Fork 1
Code
Issues 20
Pull requests 3
Projects 1
Wiki
Insights
Github Housekeeping
Updated 5 days ago
Filter cards
Add cards
Fullscreen
Menu
1 Issues Related
Create and update issues as according to this Slack thread: e.g. google analytics, user sessions... etc https://sfsucomputerscience.slack.com/archives/DGFQ0P007/p1557431083044000
Added by cherylfong
0 Pull Requests Related
5 Done
Label PRs with new docs label.
Added by cherylfong
Provide reviews for Johanna's captcha PR: #87
Added by cherylfong
1 Reference
Hide
Bck recaptcha final
#87 opened by johannah10 in CSC-648-SFSU/csc648-sp19-team08
back-end
Changes approved
Provide review for Cheryl's login/register

5. TEAM MEMBER CONTRIBUTIONS

TEAM LEAD: CHERYL FONG

Reply all | Delete | Junk | Block ···

Re: Milestone 5 - Team Member Contributions

From: Cheryl Fong
Sent: Thursday, 16 May 2019 7:23:19 AM
To: Johanna Negassi Hadgu; Jerry Wong; Chirag Madhukar Agarwal; Ermias Resom Haile; Harold Pedroso; Sharan Gi Ching Lao
Subject: Re: Milestone 5 - Team Member Contributions

Cheryl Fong (Team Lead)

Contributions

1. How I helped with documentation:
    a. Milestone 1
        i. Help complete *3. Data Items and Entities*
        ii. Help complete *3.1 Glossary*
        iii. Helped defined *4. Functional Requirements*
        iv. Completed *7. High-level System architecture and Technologies*
    b. Milestone 2
        i. Helped with *1. Data Definitions*
        ii. Helped set up *2. Functional requirements prioritizing*
        iii. Elaborated and finalize *4. High-Level Architecture Database Organization*
        iv. Created *5.2 UML Component and Deployment Diagram*
        v. Helped elaborate content for *6.1 Skill, 6.2 Schedule and 6.3 Technical Risks*
    c. Milestone 3
        i. Recorded and Uploaded Milestone 3 Review recording
    d. Milestone 4
        i. Gave input for *4.1 Coding Style*
        ii. Cleaned up and gave input for *5.1 Major Assets Protected*

Github Submissions (Top 3 Important PRs)

1. Implement User Sessions - Purpose: To allow users to log in and define sections of the website that requires registration/login.
2. Project Setup and Documentation for development stack setup - Purpose: To define the foundation of the current website and provide documentation on how to set and setup locally. For Milestone 0.
3. Cleaned vertical-prototype branch - Purpose: This pull request contained changes necessary to set up our vertical prototype. It was cleaned up as the original branch had confusing commits. Co-authored with Jerry Wong.

This is my current contribution metric for "commits" as shown on Github Contributors from Jan 21 to May 15 2019.

FRONT END LEAD: JERRY WONG

Reply all | Delete | Junk | Block ···

RE: Milestone 5 - Team Member Contributions

JW

Jerry Wong
Wed 5/15/2019 4:13 PM
Cheryl Fong; Johanna Negassi Hadgu; Chirag Madhukar Agarwal; Ermias Resom Haile; Harold Pedroso; Sharan Gi Ching Lao

Contributions

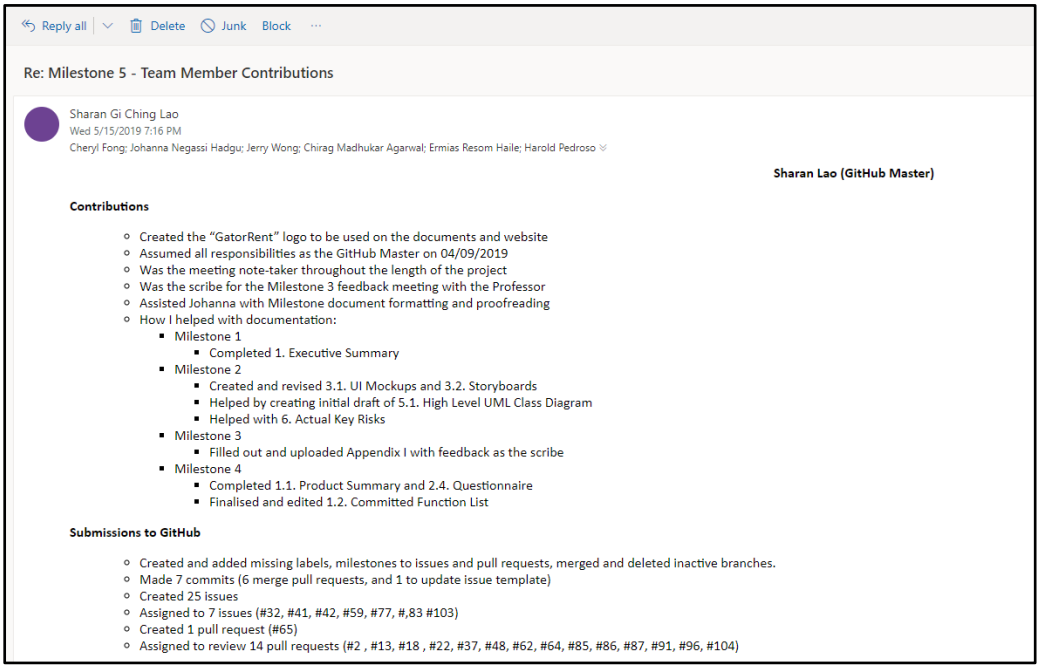
- Created vertical prototype (search results) to include the page, search bar functionality, and the returning of results.
- Established render architecture; created two bases and the header, navigation, and page footer includes.
- Created login, register, search results, listing info, and user-dashboard pages.
- Documentation contributions include competitive analysis, contributions to mock-up, and UML diagram.
- Created initial team-about page.

Front-End Lead

- Enforced usage of Bootstrap and Jinja2 engine.
- Delegated pages and assigned issues to front-end team.

From: Cheryl Fong
Sent: Wednesday, May 15, 2019 2:23 PM
To: Johanna Negassi Hadgu; Jerry Wong; Chirag Madhukar Agarwal; Ermias Resom Haile; Harold Pedroso; Sharan Gi Ching Lao
Subject: Re: Milestone 5 - Team Member Contributions

GITHUB MASTER: SHARAN LAO



Reply all | Delete | Junk | Block

Re: Milestone 5 - Team Member Contributions

Sharan Gi Ching Lao
Wed 5/15/2019 7:16 PM
Cheryl Fong; Johanna Negassi Hadgu; Jerry Wong; Chirag Madhukar Agarwal; Ermias Resom Haile; Harold Pedroso

Sharan Lao (GitHub Master)

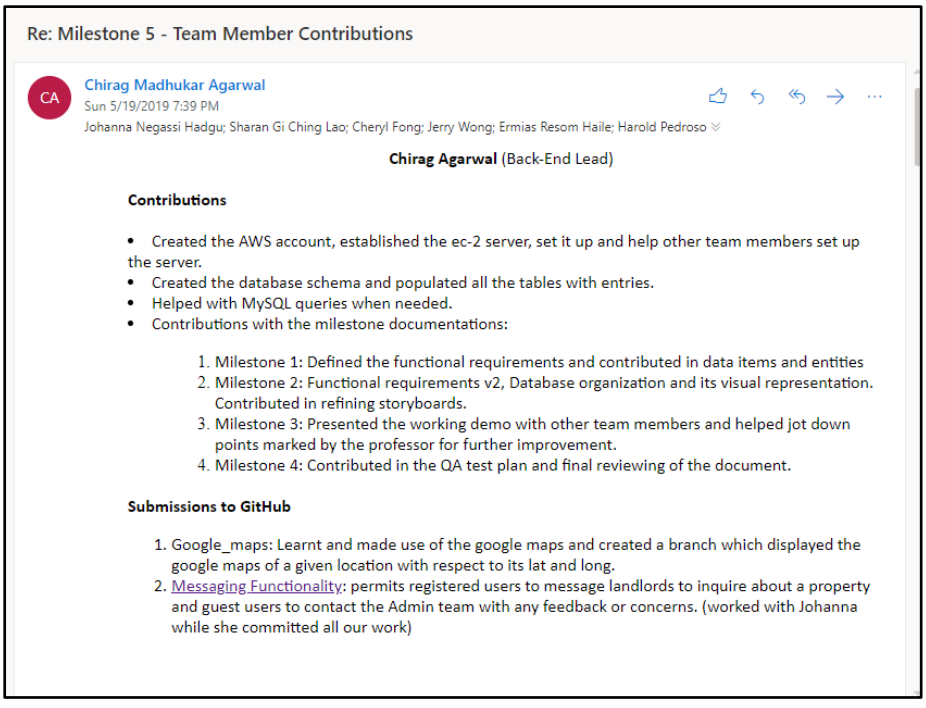
Contributions

- Created the "GatorRent" logo to be used on the documents and website
- Assumed all responsibilities as the GitHub Master on 04/09/2019
- Was the meeting note-taker throughout the length of the project
- Was the scribe for the Milestone 3 feedback meeting with the Professor
- Assisted Johanna with Milestone document formatting and proofreading
- How I helped with documentation:
  - Milestone 1
    - Completed 1. Executive Summary
  - Milestone 2
    - Created and revised 3.1. UI Mockups and 3.2. Storyboards
    - Helped by creating initial draft of 5.1. High Level UML Class Diagram
    - Helped with 6. Actual Key Risks
  - Milestone 3
    - Filled out and uploaded Appendix I with feedback as the scribe
  - Milestone 4
    - Completed 1.1. Product Summary and 2.4. Questionnaire
    - Finalised and edited 1.2. Committed Function List

Submissions to GitHub

- Created and added missing labels, milestones to issues and pull requests, merged and deleted inactive branches.
- Made 7 commits (6 merge pull requests, and 1 to update issue template)
- Created 25 issues
- Assigned to 7 issues (#32, #41, #42, #59, #77, #,83 #103)
- Created 1 pull request (#65)
- Assigned to review 14 pull requests (#2 , #13, #18 , #22, #37, #48, #62, #64, #85, #86, #87, #91, #96, #104)

BACK END TEAM LEAD: CHIRAG AGARWAL



Re: Milestone 5 - Team Member Contributions

CA

Chirag Madhukar Agarwal
Sun 5/19/2019 7:39 PM
Johanna Negassi Hadgu; Sharan Gi Ching Lao; Cheryl Fong; Jerry Wong; Ermias Resom Haile; Harold Pedroso

Chirag Agarwal (Back-End Lead)

Contributions

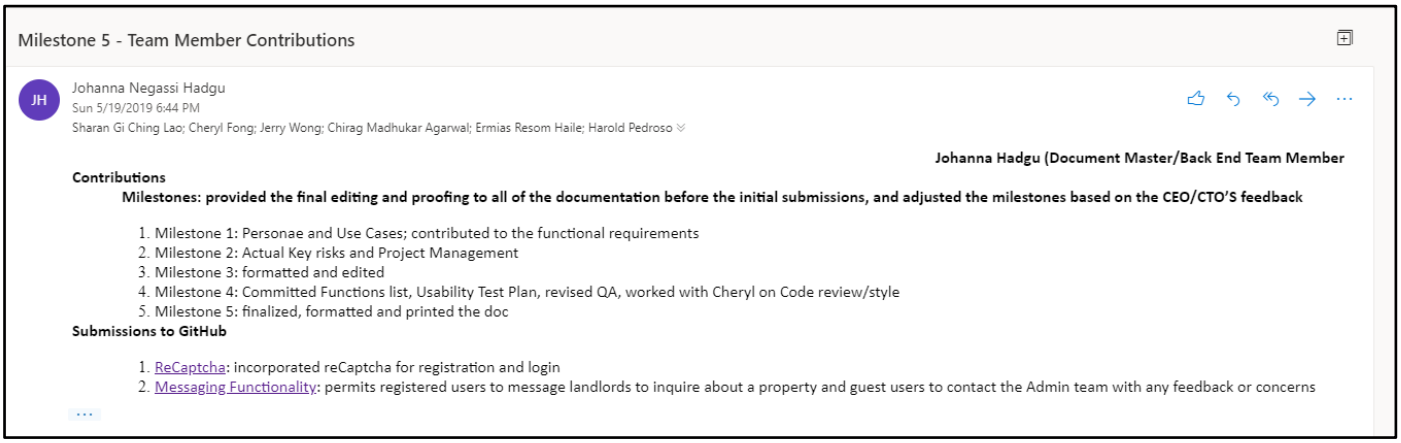
- Created the AWS account, established the ec-2 server, set it up and help other team members set up the server.
- Created the database schema and populated all the tables with entries.
- Helped with MySQL queries when needed.
- Contributions with the milestone documentations:

1. Milestone 1: Defined the functional requirements and contributed in data items and entities
2. Milestone 2: Functional requirements v2, Database organization and its visual representation. Contributed in refining storyboards.
3. Milestone 3: Presented the working demo with other team members and helped jot down points marked by the professor for further improvement.
4. Milestone 4: Contributed in the QA test plan and final reviewing of the document.

Submissions to GitHub

1. Google_maps: Learnt and made use of the google maps and created a branch which displayed the google maps of a given location with respect to its lat and long.
2. Messaging Functionality: permits registered users to message landlords to inquire about a property and guest users to contact the Admin team with any feedback or concerns. (worked with Johanna while she committed all our work)

DOCUMENT MASTER/BACK END TEAM MEMBER: JOHANNA HADGU



Milestone 5 - Team Member Contributions

JH

Johanna Negassi Hadgu
Sun 5/19/2019 6:44 PM
Sharan Gi Ching Lao; Cheryl Fong; Jerry Wong; Chirag Madhukar Agarwal; Ermias Resom Haile; Harold Pedroso

Johanna Hadgu (Document Master/Back End Team Member

Contributions

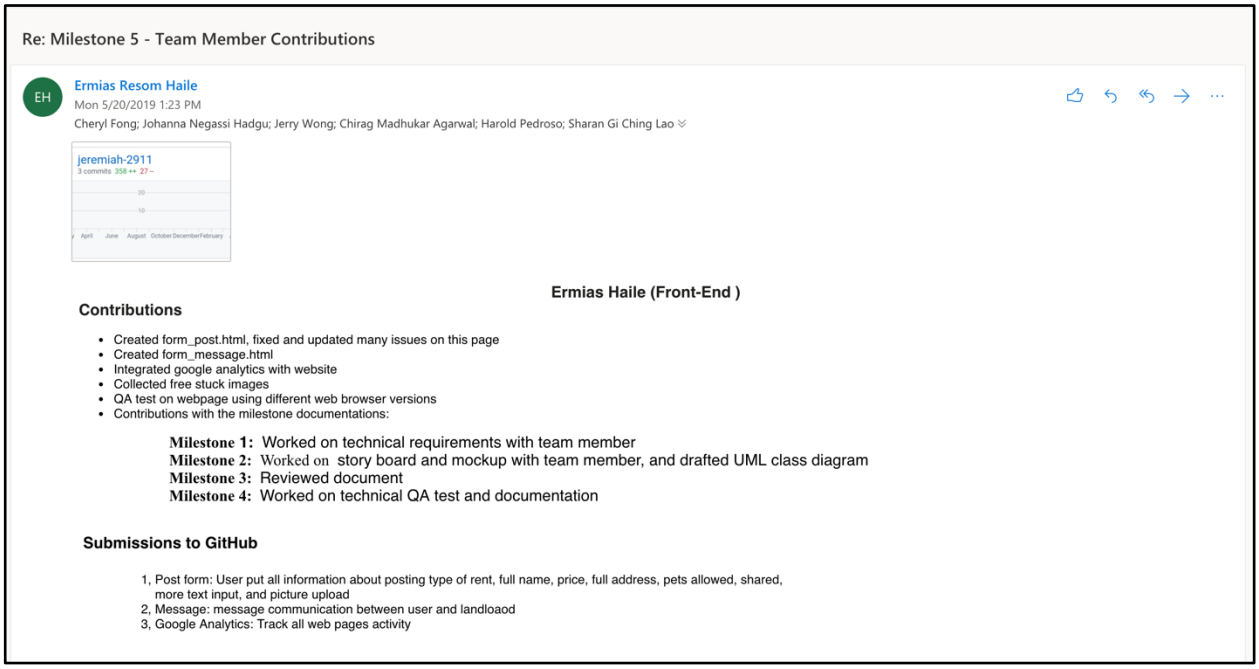
Milestones: provided the final editing and proofing to all of the documentation before the initial submissions, and adjusted the milestones based on the CEO/CTO'S feedback

1. Milestone 1: Personae and Use Cases; contributed to the functional requirements
2. Milestone 2: Actual Key risks and Project Management
3. Milestone 3: formatted and edited
4. Milestone 4: Committed Functions list, Usability Test Plan, revised QA, worked with Cheryl on Code review/style
5. Milestone 5: finalized, formatted and printed the doc

Submissions to GitHub

1. ReCaptcha: incorporated reCaptcha for registration and login
2. Messaging Functionality: permits registered users to message landlords to inquire about a property and guest users to contact the Admin team with any feedback or concerns

FRONT END TEAM MEMBER: ERMIAS HAILE



Re: Milestone 5 - Team Member Contributions

EH

Ermias Resom Haile
Mon 5/20/2019 1:23 PM
Cheryl Fong; Johanna Negassi Hadgu; Jerry Wong; Chirag Madhukar Agarwal; Harold Pedroso; Sharan Gi Ching Lao

Ermias Haile (Front-End)

Contributions

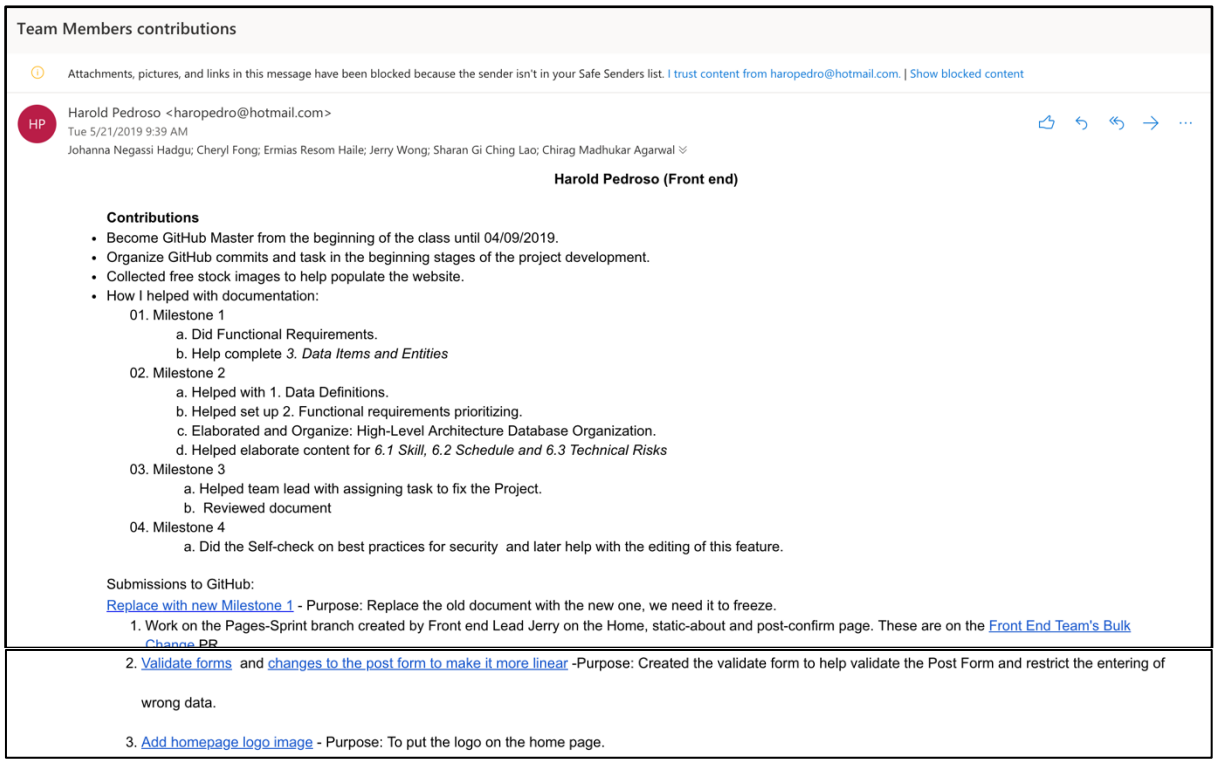
- Created form_post.html, fixed and updated many issues on this page
- Created form_message.html
- Integrated google analytics with website
- Collected free stuck images
- QA test on webpage using different web browser versions
- Contributions with the milestone documentations:

Milestone 1: Worked on technical requirements with team member
Milestone 2: Worked on story board and mockup with team member, and drafted UML class diagram
Milestone 3: Reviewed document
Milestone 4: Worked on technical QA test and documentation

Submissions to GitHub

1, Post form: User put all information about posting type of rent, full name, price, full address, pets allowed, shared, more text input, and picture upload
2, Message: message communication between user and landloaod
3, Google Analytics: Track all web pages activity

FRONT END TEAM MEMBER: HAROLD PEDROSO



Team Members contributions

Attachments, pictures, and links in this message have been blocked because the sender isn't in your Safe Senders list. I trust content from haropedro@hotmail.com. | Show blocked content

HP

Harold Pedroso <haropedro@hotmail.com>
Tue 5/21/2019 9:39 AM
Johanna Negassi Hadgu; Cheryl Fong; Ermias Resom Haile; Jerry Wong; Sharan Gi Ching Lao; Chirag Madhukar Agarwal

Harold Pedroso (Front end)

Contributions

- Become GitHub Master from the beginning of the class until 04/09/2019.
- Organize GitHub commits and task in the beginning stages of the project development.
- Collected free stock images to help populate the website.
- How I helped with documentation:
  - 01. Milestone 1
    - a. Did Functional Requirements.
    - b. Help complete *3. Data Items and Entities*
  - 02. Milestone 2
    - a. Helped with 1. Data Definitions.
    - b. Helped set up 2. Functional requirements prioritizing.
    - c. Elaborated and Organize: High-Level Architecture Database Organization.
    - d. Helped elaborate content for *6.1 Skill, 6.2 Schedule and 6.3 Technical Risks*
  - 03. Milestone 3
    - a. Helped team lead with assigning task to fix the Project.
    - b. Reviewed document
  - 04. Milestone 4
    - a. Did the Self-check on best practices for security and later help with the editing of this feature.

Submissions to GitHub:

Replace with new Milestone 1 - Purpose: Replace the old document with the new one, we need it to freeze.

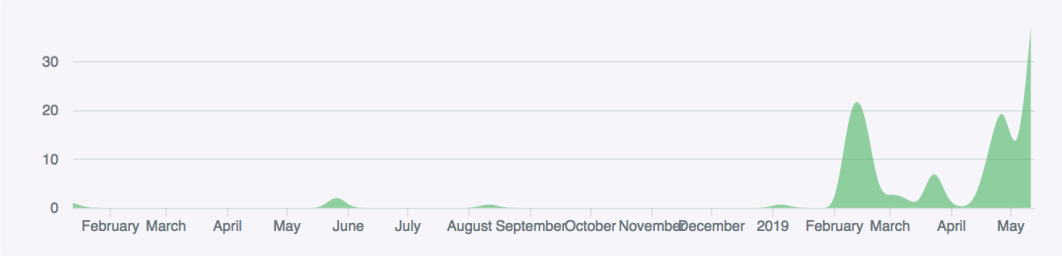
1. Work on the Pages-Sprint branch created by Front end Lead Jerry on the Home, static-about and post-confirm page. These are on the Front End Team's Bulk Change PR
2. Validate forms and changes to the post form to make it more linear -Purpose: Created the validate form to help validate the Post Form and restrict the entering of wrong data.
3. Add homepage logo image - Purpose: To put the logo on the home page.

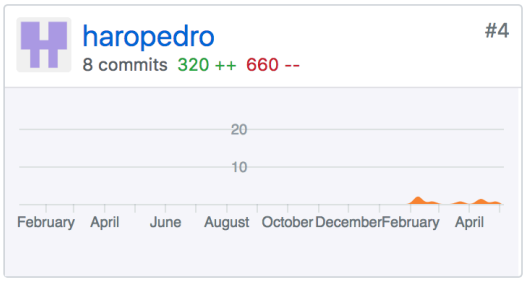
COMMITS

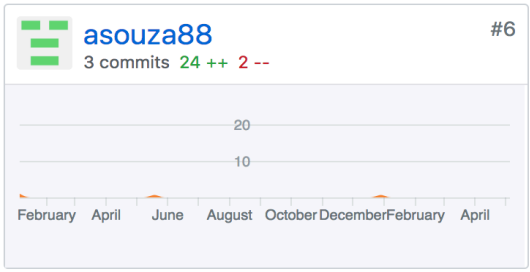
Jan 21, 2018 – May 21, 2019

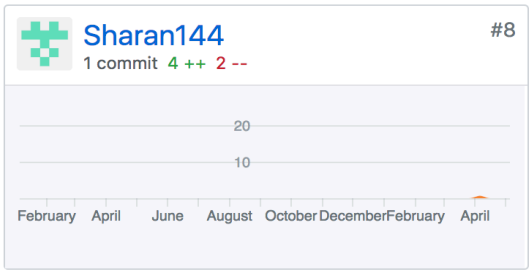
Contributions: **Commits**

Contributions to master, excluding merge commits

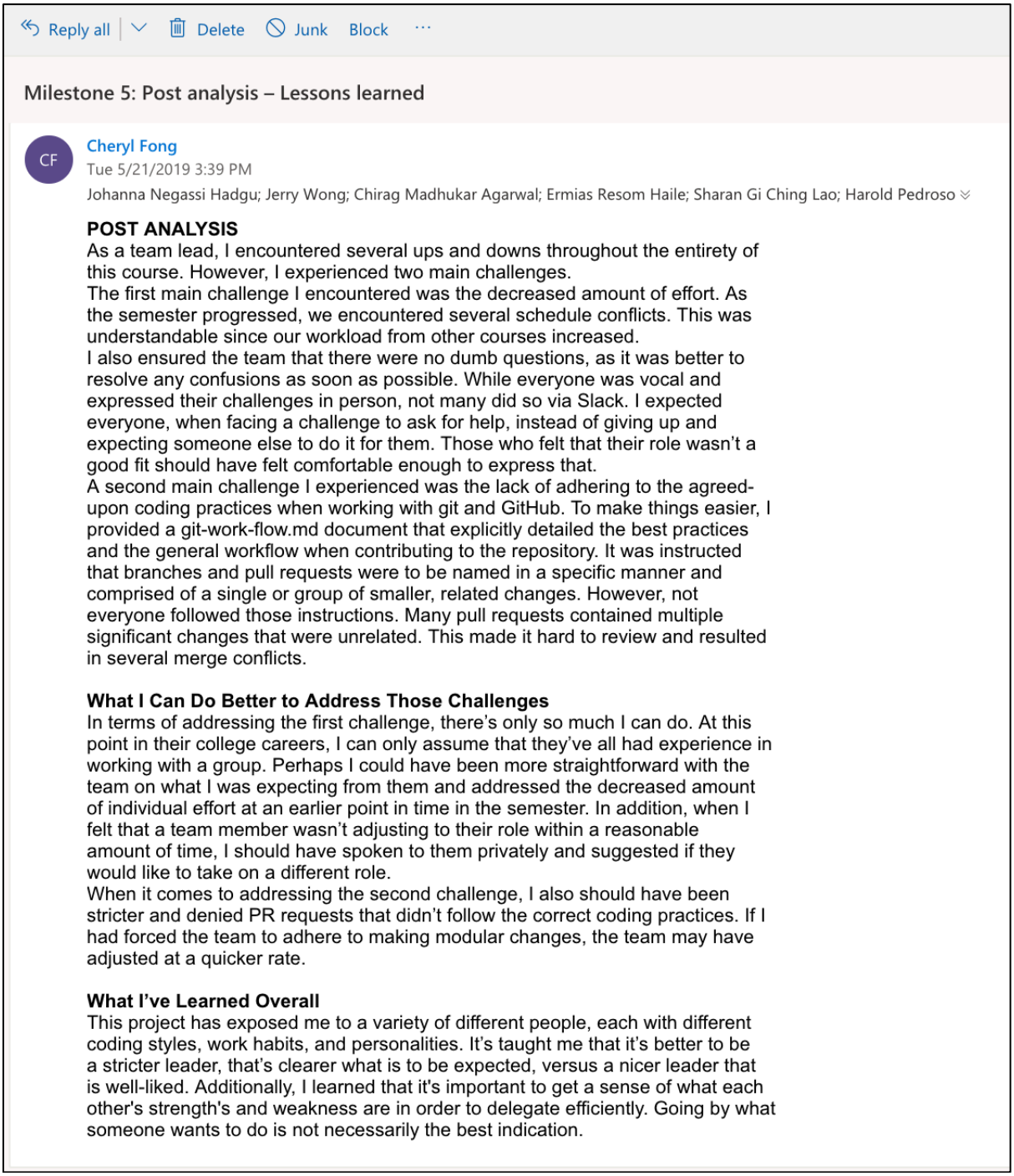








6. POST ANALYSIS



Reply all | Delete | Junk | Block | ···

Milestone 5: Post analysis – Lessons learned

CF

Cheryl Fong
Tue 5/21/2019 3:39 PM
Johanna Negassi Hadgu; Jerry Wong; Chirag Madhukar Agarwal; Ermias Resom Haile; Sharan Gi Ching Lao; Harold Pedroso

POST ANALYSIS
As a team lead, I encountered several ups and downs throughout the entirety of this course. However, I experienced two main challenges.
The first main challenge I encountered was the decreased amount of effort. As the semester progressed, we encountered several schedule conflicts. This was understandable since our workload from other courses increased.
I also ensured the team that there were no dumb questions, as it was better to resolve any confusions as soon as possible. While everyone was vocal and expressed their challenges in person, not many did so via Slack. I expected everyone, when facing a challenge to ask for help, instead of giving up and expecting someone else to do it for them. Those who felt that their role wasn't a good fit should have felt comfortable enough to express that.
A second main challenge I experienced was the lack of adhering to the agreed-upon coding practices when working with git and GitHub. To make things easier, I provided a git-work-flow.md document that explicitly detailed the best practices and the general workflow when contributing to the repository. It was instructed that branches and pull requests were to be named in a specific manner and comprised of a single or group of smaller, related changes. However, not everyone followed those instructions. Many pull requests contained multiple significant changes that were unrelated. This made it hard to review and resulted in several merge conflicts.

What I Can Do Better to Address Those Challenges
In terms of addressing the first challenge, there's only so much I can do. At this point in their college careers, I can only assume that they've all had experience in working with a group. Perhaps I could have been more straightforward with the team on what I was expecting from them and addressed the decreased amount of individual effort at an earlier point in time in the semester. In addition, when I felt that a team member wasn't adjusting to their role within a reasonable amount of time, I should have spoken to them privately and suggested if they would like to take on a different role.
When it comes to addressing the second challenge, I also should have been stricter and denied PR requests that didn't follow the correct coding practices. If I had forced the team to adhere to making modular changes, the team may have adjusted at a quicker rate.

What I've Learned Overall
This project has exposed me to a variety of different people, each with different coding styles, work habits, and personalities. It's taught me that it's better to be a stricter leader, that's clearer what is to be expected, versus a nicer leader that is well-liked. Additionally, I learned that it's important to get a sense of what each other's strength's and weakness are in order to delegate efficiently. Going by what someone wants to do is not necessarily the best indication.