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

**Team 13**
**Amari Bolmer - Team Lead - amarirules@gmail.com**
**Brian Ho, Simon Tan, Adeel Bhatti,**
**Kim Wang, Kurtis Hoang, Sushil Kumar Plassar**

**Demo URL:** http://18.223.29.166/
**May 21, 2019**

## 2) Product summary:

LiveGator

LiveGator

LiveGator is a rental website targeted towards helping San Francisco State University students find somewhere to live. SFSU students will easily be able to find a place specifically to their needs, whether its is based on price, size or distance from SFSU.

Committed Functions:

- Users shall be able to sign up, log into, and log out of the website.
- Users shall be able to use the search bar to find all available listings.
- Users shall be able to filter listings by price and size.
- Users shall be able to sort listings by price and size.
- Sellers shall be able to post and delete listings.

LiveGator is uniquely focused on students having the ability to search for rooms, housing, or apartments with the option to sort by distance to campus of San Francisco State University. The function to sort by distance shall give students peace of mind to make the decision of where to buy or rent easy as a click of a button. We shall employ an admin function that will ensure quality control over post to ensure content that is posted is not malicious in content.

## 3) Milestone documents – M1-M4

# Milestone 1

## March 5, 2019

## Table of Contents

| | |
|---|---|
| Executive Summary | 3 |
| Personae and Main Use Cases | 4 |
| List of Main Data Items and Entities | 8 |
| Initial List of Functional Requirements | 9 |
| List of Non-functional Requirements | 10 |
| Competitive Analysis | 11 |
| High-level System Architecture and Technologies Used | 12 |
| Teams | 13 |
| Checklist | 14 |

## 1. Executive Summary

Do you think finding somewhere to live around SFSU is unnecessarily difficult? Are you tired of commuting to school from the East Bay? Have you ever wanted to know what your commute options are before moving to a new home? If you said yes to any of these questions, you need LiveGator in your life.

LiveGator is for SFSU students that are looking for somewhere to live near campus. We believe that students should not have to worry about having a roof over their heads while they're struggling with the challenges of school. Currently there is no application on the market targeted at helping SFSU students find housing near campus. We think this is a huge problem because there are over 1000 people waiting for university housing. Hopefully with LiveGator, we can reduce that number, so students can focus entirely on their education.

LiveGator will be a platform that connects students to landlords. Landlords will be able to post photos, descriptions, and a price for their residences. What makes us different from other rental sites is that we automatically include the best ways to get to SFSU on each listing. We will have directions for taking public transit, walking, and driving to campus. Students will be able to browse through the site to find a place to rent. Once they've decided, they will have to make an account and message the landlord. When the connection is made, it is up to the landlord and the tenant to decide how to proceed.

The LiveGator team is made of seven SFSU computer science students, who are frustrated with how difficult it is to rent in San Francisco. We hope to be able to launch this site so we can focus on making software instead of worrying about shelter.

## 2. Personae and Main Use Cases

| Sam | About Sam: |
|---|---|
| Sam | About Sam:<br>• Entering Junior year at SFSU<br>• Cannot stay in dormitory for Junior year<br>• Likes to browse websites without signing up<br>• English major<br>• Poetry club |
| Behaviors:<br>• Sometimes wakes up late for class<br>• Sometimes indecisive<br>• Willing to move off campus for the right apartment<br>• Reads a lot | Needs and wants:<br>• Wants to find an apartment near campus<br>• Needs pictures to help make final decision<br>• Needs to compare prices |

| Loren | About Loren |
|---|---|
| | • Senior at SFSU<br>• Is a music major<br>• Plays the piano and drums<br>• Loves music<br>• Have used LiveGator before |
| Behaviors:<br>• Pracaticies music throughout the day<br>• Plays drums and instruments loudly<br>• Practicing disrupts people studying<br>• Does not like disrupting people studying | Needs and wants:<br>• Needs to practice playing his instruments<br>• Wants to practice music with other people<br>• Wants to meet other musicians |

## A. Unregistered Users:

Sam is entering her Junior year at San Francisco State University. No longer able to stay in the dorms, she has to find a place to live before the semester starts. She is unsure of whether she wants to rent an on-campus apartment or find one off-campus. She loves the idea of being close to school, so she browses LiveGator for apartments to rent and uses the pictures, price compare, and distance filter features to find the right apartment for her near campus. She finds her perfect home and is prompted to register for LiveGator in order to contact the poster.

## B. Registered Users:

Loren is a Senior and a music major at San Francisco State University. He has received a few complaints from his current housemates about his instrument practicing being too disruptive throughout the semester. Having used LiveGator before, he remembers there is a feature to filter rentals by majors. He uses LiveGator to find a new house full of other music majors, hoping not to disrupt other people studying because everyone else would be practicing their instruments as well. As he browses through the listings on LiveGator, he keeps an eye on the "about housemate" section where there are icons that indicate the major's of the other students living there (e.g. a book for english majors, a flask for science majors, a music note for music majors, ect).

## C. Landlord:

Shannon has a decided to rent out her second home to students. She has heard about the website LiveGator from her niece and has decided to use it instead of hiring a real estate agent. Although Shannon does not generally use computers, the easy to follow instructions prompts her to input all relevant information (e.g. pricing, photos, contact info, etc). She especially likes the "more info" section where she is able to post an introduction about herself and a set of house rules for prospective renters. The UI informs Shannon that her post must be approved, then she fills out all the information, and she must agrees before submitting that her post must be approved first before going live. She agrees and receives an email letting her know her post has been received and is being reviewed.

## D. Administrator:

Kerry is a LiveGator team member. She was hired because she has a background in working with spreadsheets. Because of this experience, she has some decent technical skills. When she was hired, she was given training on what is acceptable to post on the website. Kerry would log in like a normal user, except she would have an admin account. Then she will see a dashboard with pending posts and approved posts. She would then look through the pending posts and approve the ones that meet the website's guidelines. If the post do not meet guidelines, she rejects the post and ask the submitter to make the necessary changes accordingly.

## 3. List of Main Data Items and Entities

### A. Unregistered Users:

Users who have not signed up or logged in. They can use the website to browse the listings and house details. They cannot send renting requests or contact landlords.

### B. Registered User:

Registered users can be both customers and landlords at the same time, but they shall have different dashboard to view specific activities. They do not have to be SFSU students. They can browse the website and listing details. Also, they can contact landlords, send renting requests, and post their properties for renting.

### C. Administrator:

Users who have access to the database and perform administrative tasks.

### D. Listing

Listed properties on the website. Listings can be houses, apartments, condos or rooms.

### E. User record

User record includes user messages and user orders. Registered users can view their customer dashboard or landlord dashboard to view their completed orders and messages.

# 4. Initial List of Functional Requirements

## A. Unregistered User

1. Shall be able to register an account or log in
2. Shall be able to browse the available houses
3. Shall be able to view listing details
4. Shall be able to filter listings by price, size and distance range
5. Shall be able to sort listings by price, size and distance

## B. Registered User

Shall be able to do what unregistered users can do plus:

6. Shall be able to contact landlords
7. Shall be able to make a request to rent properties
8. Shall be able to contact registered customer who has sent renting request
9. Shall be able to make add, delete, and edit renting properties

## C. Administrators:

Shall be able to do what regular registered users can do plus:

10. Shall be able to access the database
11. Shall be able to delete listings
12. Shall be able to block users
13. Shall be able to approve listings for posting

## 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 <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).

## 6. Competitive Analysis

Currently in the market there are three primary ways students find housing around our campus. All three of them are free to use and easy to access: Craigslist, Zillow and Facebook.

On Craigslist and Zillow- master tenants, realtors or landlords can post listings of housing situations. After listings have been made- students can browse and search to find an appropriate fit. They can sort and filter by price and location. After they click on a listing there are photos and there is text information displayed. Despite having good browsing and search, Zillow and Craigslist both lack any form of messaging.

On Facebook, students can post in the SFSU housing group any vacancies they are trying to fill. This is different than the anonymous nature of Craigslist and Zillow as people making the listings can see the students responding and vice versa. However, Facebook is weak in it's browsing capabilities and search functionality. The search functionality is not detailed and it consistently shows listings that are outdated.

Coupled with knowing that all the users on LiveGator will be SFSU students- messaging will help renters build trust with landlords. In addition to all of the features these 3 competitors have, we are including a "Sort by distance to SFSU" feature to sort listings according to how far they are from the SFSU campus.

### Features:

| | User-Registration | Messaging | Browse | Search | Distance to SFSU |
|---|---|---|---|---|---|
| LiveGator | x | x | x | x | x |
| Craigslist | x | | x | x | |
| Facebook | x | x | x | | |
| Zillow | x | | x | x | |

As you can see, LiveGator is due to have all of the same features as current market leaders in addition to our "Distance to SFSU" feature. Each of the competition has their own weak functionalities that we hope to create better versions of.

## 7. High-level System Arch & Technologies Used

### A. Platform

1. Ubuntu 16.04 - Linux operating system
2. Amazon Web Service - Cloud service platform
3. Apache 2.4 - Web server

### B. Server-Side Language

1. Python 3.5 - high-level programming language

### C. Frameworks

1. Bootstrap v4.3.1 - CSS framework for responsive mobile-first websites
2. MySQL 8.0 - Database management system
3. Flask 1.0 - Web framework

### D. IDE

1. PyCharm IntelliJ - Integrated development environment specifically for the language Python

### E. Tools

1. GitHub - Git repository hosting service
2. Git - Distributed version control tool
3. MySql Workbench - Visual database design tool

### F. API

1. Google Analytics - Web service that tracks and reports website traffic
2. Google Maps - Web-based service that provides detailed geographical information around the world

### G. Supported Browsers

1. Chrome - version: 72.0.3626
2. Firefox - version: 65.0.1 & 60.5.1
3. Safari - version: 10.14 Mojave & 10.13 High Sierra

## 8. Team

| | |
|---|---|
| Amari Bolmer | Team Lead |
| Brian Ho | Front End Lead |
| Kim Wang | Back End Lead |
| Sushil Kumar Plassar | Front End/Github Master |
| Simon Tan | Front End |
| Kurtis Hoang | Back End |
| Adeel Bhatti | Back End/Document Master |

## 9. Checklist

1. Team found a time slot to meet outside of class -
DONE

2. Github master chosen -
DONE

3. Team decided and agreed together on using the listed SW tools and deployment server -
DONE

4. 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

5. Team lead ensured that all team members read the final M1 and agree/understand it before submission -
DONE

6. Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.) -
DONE

# Milestone 2

## March 19, 2019

History Table:
Date Submitted: March 22, 2019

## Table of Contents

**Data Definitions**
**3**

**Functional Requirements**
**3**

**UI Mockups and Storyboards (high level only)**
**3**

**High level Architecture, Database Organization**
**3**

**High Level UML Diagrams**
**3**

**Identify actual key risks**
**3**

**Project management**
**3**

# 1. Data Definitions

## A. Unregistered Users:

Users who have not signed up or logged in. They can use the website to browse the listings and house details. They cannot send renting requests, post a listing, nor contact landlords.

## B. Registered User:

Registered users can be both customers and landlords at the same time, but they shall have different dashboard to view specific activities. They can browse the website and listing details. Also, they can contact landlords, send renting requests, and post their properties for renting.

## C. Administrator:

Users who have access to the database and perform administrative tasks including approving postings before they go live and blocking users.

## D. Listing

Listed properties on the website. Listing types include:

- Houses
- Apartments
- Condos
- Townhomes
- Single rooms

## E. Landlord Dashboard

A page a user can see related information as a landlord. Information includes posted listings, orders, and messages to current or prospective customers. All users shall have both landlord dashboard and customer dashboard.

## F. Customer Dashboard

A page a user can see related information as a customer. Information includes orders and messages to landlords. All users shall have both landlord dashboard and customer dashboard.

## G. Order

Orders are defined as completed renting transactions. Since a user can both rent a property from our website and post a property on our web for others to rent, there can be two types of orders for a user - orders from a customer perspective, and orders from a landlord perspective. Therefore, we separate these two types of orders and showed them on both customer dashboard and landlord dashboard; on customer dashboard, a user can only see their orders from a customer perspective, while on landlord dashboard, a user can only see their order from a landlord perspective.

## H. Message

Similar as “Order” data,  message can also have two types: message from customer to landlord, and message from landlord back to customer. We display message the same way as we do for orders; on customer dashboard, a user can only see their message conversations with other landlord users, while on landlord dashboard, a user can only see their message conversations with other customer users.

# 2. Functional Requirements

## Priority 1:

### A. Unregistered User

1. Shall be able to register an account
2. Shall be able to browse the available houses
3. Shall be able to view listing details
4. Shall be able to filter listings by price, size and distance range
5. Shall be able to sort listings by price, size and distance

### B. Registered User

Shall be able to do what unregistered users can do plus:

6. Shall be able to login
7. Shall be able to contact landlords
8. Shall be able to make a request to rent properties
9. Shall be able to send email to the landlord
10. Shall be able to send email to the person looking to rent
11. Shall be able to post listing
12. Shall be able to make add, delete, and edit renting properties

### C. Administrators:

Shall be able to do what regular registered users can do plus:

13. Shall be able to access the database
14. Shall be able to delete listings
15. Shall be able to approve listings for posting

## Priority 2:

### A. Unregistered User

1. Shall be able to sign up using their google and facebook account

### B. Registered User

Shall be able to do what Unregistered users can do plus:

2. Shall be able to upload the profile picture
3. Shall be able to Edit the profile or change the notification preferences

## Priority 3:

### A. Unregistered User

1. Shall be able to see a promoted items on the front page

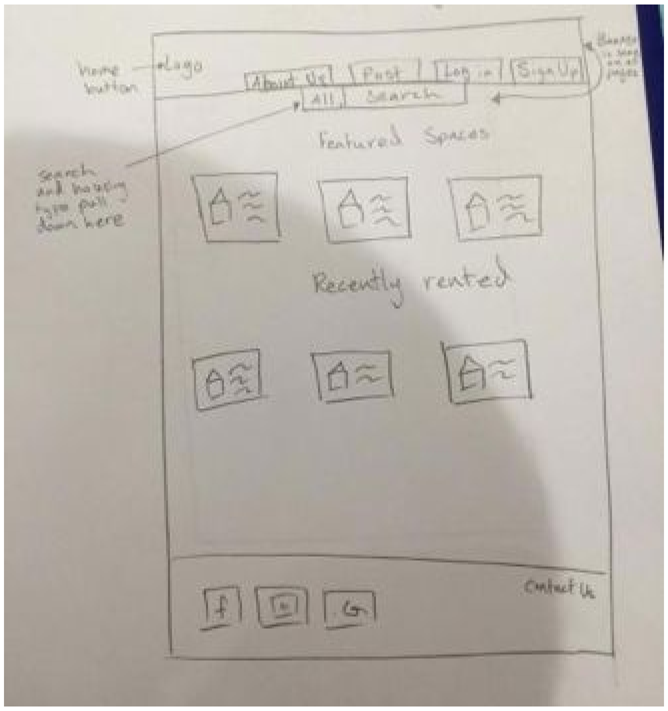
### B. Registered User

Shall be able to do what Unregistered users can do plus

2. Shall be able to send message to other registered users
3. Shall be able to message landlord through inbuilt web chat app
4. Shall be able to contact person looking to rent through inbuilt web chat
5. Shall be able to use reset/forgot password functionality
6. Shall be able to refer a friend
7. Shall be able to see travel times to school from location

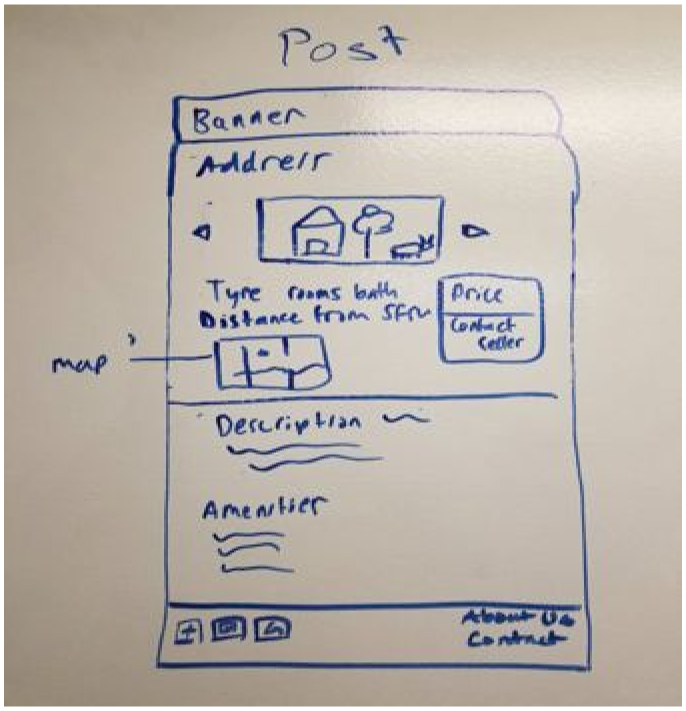
## 3. UI Mockups and Storyboards (high level only)

### a. Home page



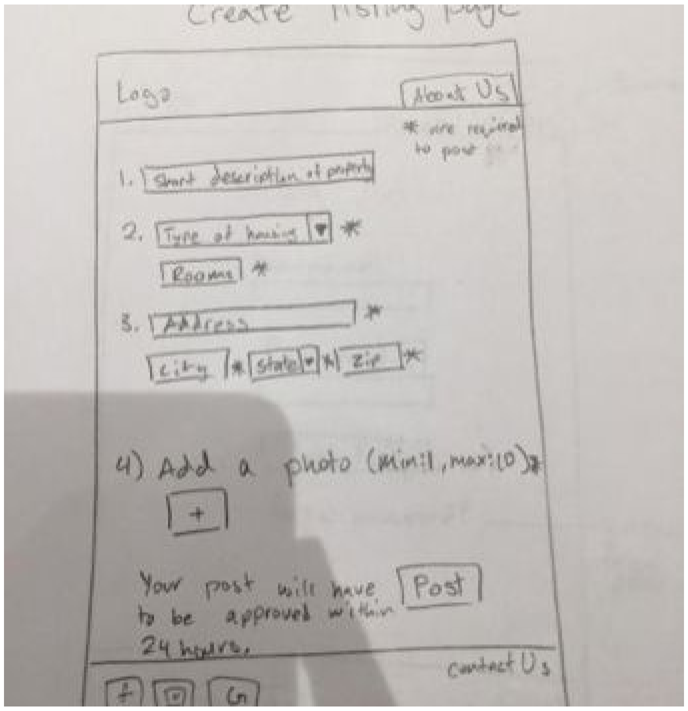
When any user enters our site, they see the home page. There is a banner at the top that stays the same throughout all pages. Search is also in the same place in each page. When a user clicks a house icon, they will be taken to a post.

### b. Listing page



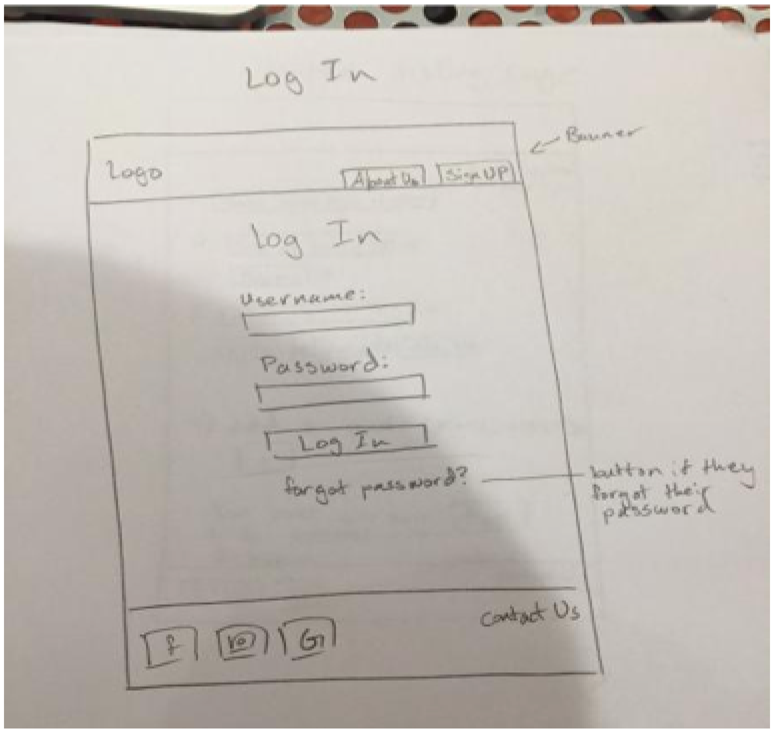
The post shows relevant information at the top. Then the user will see any other information that the landlords want them to see.

## c. Create listing page



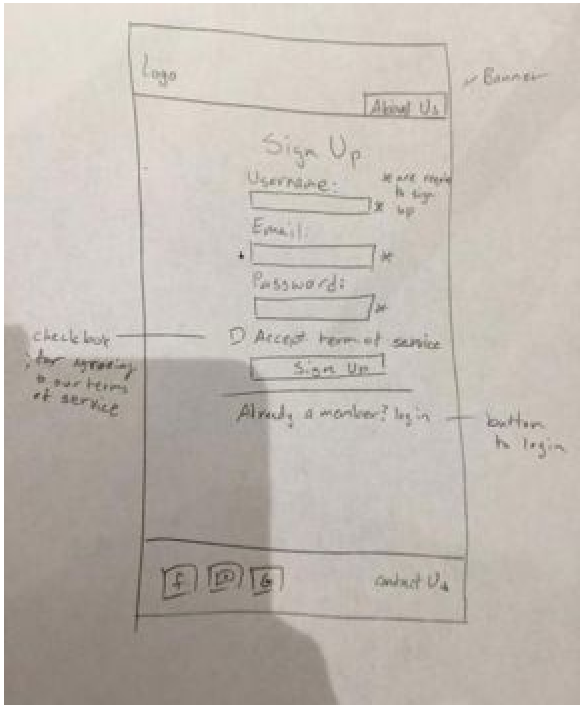
When a registered user presses the post button, they get taken to the create listing area. This form will have uniform fields and fonts to make the user experience better.

## d. Login page



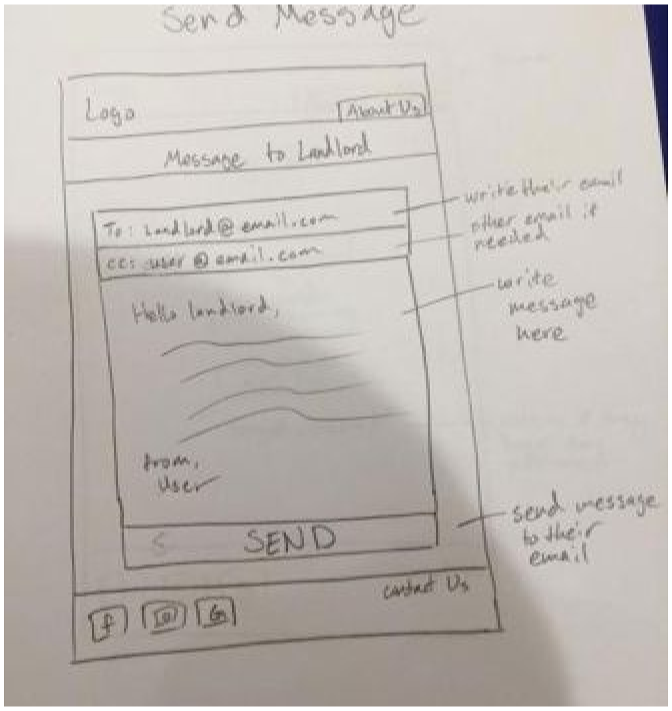
The login page will be pretty simple. There are just 2 fields for username and password. If a user is forgetful like most normal user, they can choose to reset their password.

## e. Signup page



If a user does not have an account and tries to post something, they will be taken to this sign up page. This is because the user has already invested time into creating a post and they would not want to waste their efforts. There will also be a button for registered users to log in.

## f. Send message page



**This is a prototype for messaging screen. We will try to make it look like an instant messenger. If we can’t figure it out we will make it look like an email client.**

## Admin Storyboard

The admin storyboard will consist of:
i) Admin Home Page
ii) Admin Approval Page
iii) View History Page

## Admin Home Page

The below screenshot shows the landing page for the admin. It will have the links for him/her to view all the listings, approve/reject the new postings under ‘Pending Requests’ and see the User details under User

## Pending Request page

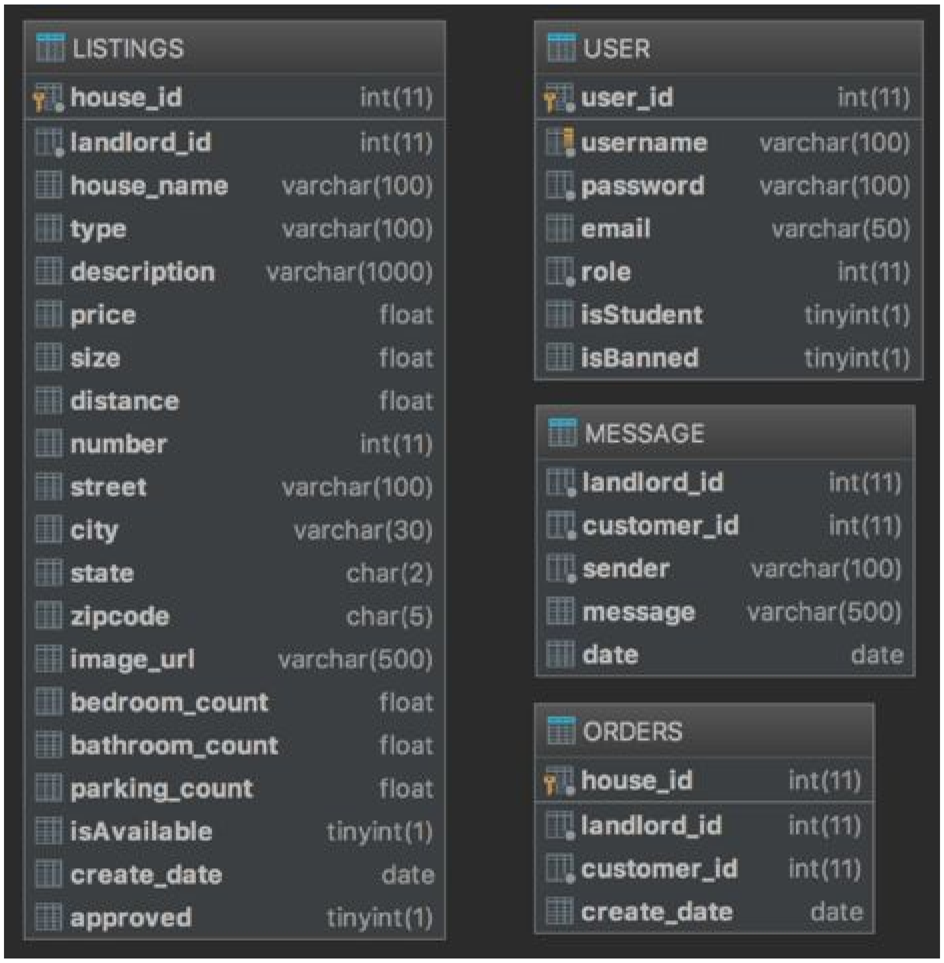
The admin can view the new posts for approval or rejection under this page. The below screenshot shows the mock page layout:

## User Detail Page

The admin can view user details along with the posted ads from the user on this page. Below screenshot shows the mock dashboard for the same

# 4. High level Architecture, Database Organization

## Database Organization



LISTINGS

| | |
|---|---|
| house_id | int(11) |
| landlord_id | int(11) |
| house_name | varchar(100) |
| type | varchar(100) |
| description | varchar(1000) |
| price | float |
| size | float |
| distance | float |
| number | int(11) |
| street | varchar(100) |
| city | varchar(30) |
| state | char(2) |
| zipcode | char(5) |
| image_url | varchar(500) |
| bedroom_count | float |
| bathroom_count | float |
| parking_count | float |
| isAvailable | tinyint(1) |
| create_date | date |
| approved | tinyint(1) |

USER

| | |
|---|---|
| user_id | int(11) |
| username | varchar(100) |
| password | varchar(100) |
| email | varchar(50) |
| role | int(11) |
| isStudent | tinyint(1) |
| isBanned | tinyint(1) |

MESSAGE

| | |
|---|---|
| landlord_id | int(11) |
| customer_id | int(11) |
| sender | varchar(100) |
| message | varchar(500) |
| date | date |

ORDERS

| | |
|---|---|
| house_id | int(11) |
| landlord_id | int(11) |
| customer_id | int(11) |
| create_date | date |

## Media storage

Media Storage: BLOBs (Binary Large Objects)
Images: Images shall be in the format .JPG or .PNG
Audio: All audio files shall be .MP3
Video: All videos shall be .MP4
GPS: GPS and maps shall be implemented using Google Maps

### Search/filter architecture and implementation

Filters:

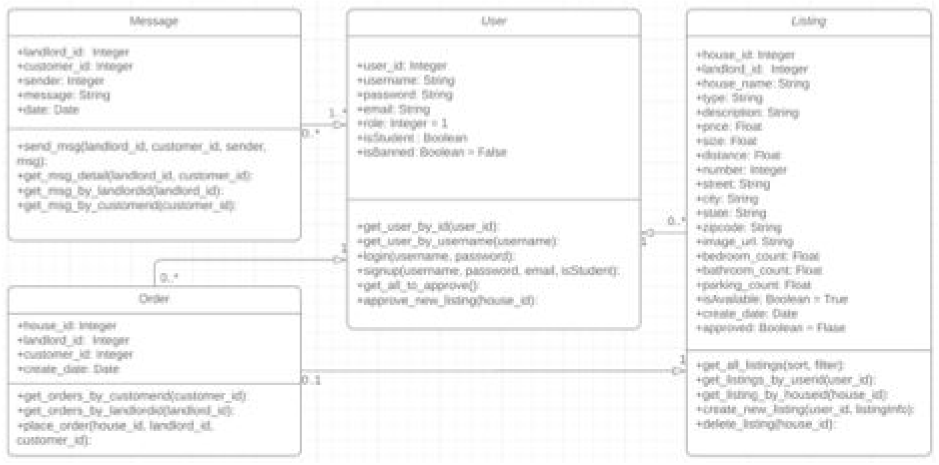
- **Distance**: Users shall be able to search for listings based on a distance to a certain location using google maps geocode/geolocation to find the distance. The location of the listing will be computed once and stored in the listing database.
- **Major**: Users shall be able to search for listings based on housemates with similar majors.
- **Listing types**: Users shall be able to search by types of listings (i.e apartment, house, single bed rooms, shared bedroom)

Search:

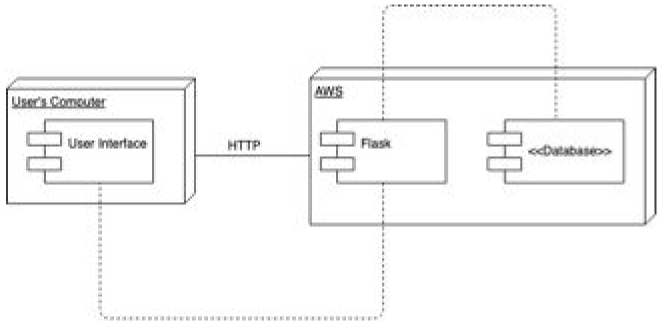
- **%Like:** MySQL built-in search architecture,%like, shall be used to search the database based on the filters.

## 5. High Level UML Diagrams

### Class diagram


Message
+landlord_id: Integer
+customer_id: Integer
+sender: Integer
+message: String
+date: Date
+send_msg(landlord_id, customer_id, sender, msg):
+get_msg_detail(landlord_id, customer_id):
+get_msg_by_landlordid(landlord_id):
+get_msg_by_customerid(customer_id):
User
+user_id: Integer
+username: String
+password: String
+email: String
+role: Integer = 1
+isStudent : Boolean
+isBanned: Boolean = False
+get_user_by_id(user_id):
+get_user_by_username(username):
+login(username, password):
+signup(username, password, email, isStudent):
+get_all_to_approve():
+approve_new_listing(house_id):
Listing
+house_id: Integer
+landlord_id: Integer
+house_name: String
+type: String
+description: String
+price: Float
+size: Float
+distance: Float
+number: Integer
+street: String
+city: String
+state: String
+zipcode: String
+image_url: String
+bedroom_count: Float
+bathroom_count: Float
+parking_count: Float
+isAvailable: Boolean = True
+create_date: Date
+approved: Boolean = Flase
+get_all_listings(sort, filter):
+get_listings_by_userid(user_id):
+get_listing_by_houseid(house_id):
+create_new_listing(user_id, listingInfo):
+delete_listing(house_id):
Order
+house_id: Integer
+landlord_id: Integer
+customer_id: Integer
+create_date: Date
+get_orders_by_customerid(customer_id):
+get_orders_by_landlordid(landlord_id):
+place_order(house_id, landlord_id, customer_id):
1..*
0..*
1
0..*
0..*
1
1
0..1

**Deployment diagram**



# 6. Identify actual key risks

## - Skill Risk & Scheduling Risk

- Skill risk associated within our team project relies on the inability at current of members being unskilled in various languages.  As we begin, the complexity of the project depends upon members ability to quickly learn the skills necessary for their role. This is negated by giving certain coding practices to study to various members.

- Only one member of our team is fluent in python, our backend lead, yet our backend lead is not very well skilled in AWS.  Tutorials will be used to help with understanding.

- Scheduling Risk, and may pose an issue when troubleshooting the server if the backend lead is unavailable to help and we are debugging an issue within a language we are not familiar with.

- Skill Resolution, this issue will be handled by having multiple members working on the backend, as the backend lead is gifted in python, this will allow other members to focus on understanding Amazon Web Server.

- Scheduling Risk, the time slot we originally planned is unable to be met by all members.

- Scheduling conflicts will be resolved by members in front-end and backend teams making appropriate separate times to meet in seperate groups to allow deliberation on their respective task if they cannot make the appropriate group discussions.

# 7. Project management

## - Management Details

Current Tools:

- Trello (Task Assigning and Progress Checker)
- Discord App ( Voice & Messaging)
- Discord App (Screen Sharing Feature)

Future tools will remain from the current tools being implemented. Project deadlines are important so pushing the team to complete assignments early allows for revision time. If any comments or changes would like to be addressed, there is time. Trello is easy for assigning task and monitoring but will stress the importance of checking Trello at least twice a week. Moving forward, with discord we have the ability to @everybody which sends a push notification to all members of the team and can remind members to complete task. Members may forget deadlines, but using direct communication for reminders, team members are quick to remain on task. Tutorials and videos are being used to ensure members understand their assignments.

# Milestone 3

**Section: 02 Team: 13 Date: 05/03/2019**

## Instructor to Check and comment below:

· Git/Github organization (e.g. organization of branches)

Gh organization looks good, usage of benches are ok

· Git/Github usage: Comments on positing; Number of posting to github; Appr. even distribution of submissions among team members (check github post stats for all members)

Git messages are fine

```
    48  Yangyifan Wang
    20  stan625
     9  AmariRules
     9  sushilkplassar
     8  pancreaspinch
     7  AmariBolmer
     5  Kurtis Hoang
     3  Adeel
     3  csc648848Instr
     2  Anthony
     2  adeel182
     1  Anthony J Souza
     1  Simon
     1  kurtis hoang
```

- Code documented (header, in code) with good coding style

Coding style is good, little to no comments, missing comment headers.

- MVC/OO patterns followed up

Followed through framework

- Frameworks (back end front end) deployed correctly

Flask deployed

- Database organization (tables, naming…)

Database looks fine,

- Blobs being used? If so, is it working?

Image paths seemed to be used

- Adherence to best practices of security (PW encrypted, search inputs verified etc.)

Passwords don’t seem to be encrypted

- Efficiency (proper use of image thumbnails, efficient search eytc.)

No thumbnails present, search is fine

- Other

# M3 Feedback and Plan

Team 13

Coding style and Consistent naming appropriate
Basic Header comments will be added

GitHub commits – More descriptive

Master Branch update clean code

Change “Find your rental” from homepage
Search bar should remain Static

Persistent search will be added
Search bar must be visible at all times

Use only the most useful filters,
Sort by:
Type, Price, and distance

Add :
Clear Button
Apply button

Change “Featured homes” to:
Recently Posted

Add:
User login,
New user please register:

Change “listing info” to:
“Post listing” – Action oriented

Next to post add, “May need 24 hours for approval”

I agree to terms, Opt-in box
Enforce field check box

Username and email same info.

Messaging -
Back and forth messaging to be removed
1 message to landlord to be added

# Milestone 4
# May 8, 2019

History Table:
Submission: May 8th, 2019
Revised: May 12, 2019

## <u>Product summary</u>

LiveGator

LiveGator is a rental website targeted towards helping San Francisco State University students find somewhere to live. SFSU students will easily be able to find a place specifically to their needs, whether its is based on price, size or distance from SFSU.

Committed Functions:

- Users shall be able to sign up, log into, and log out of the website.
- Users shall be able to use the search bar to find all available listings.
- Users shall be able to filter listings by price and size.
- Users shall be able to sort listings by price and size.
- Sellers shall be able to post and delete listings.

LiveGator is uniquely focused on students having the ability to search for rooms, housing, or apartments with the option to sort by distance to campus of San Francisco State University. The function to sort by distance shall give students peace of mind to make the decision of where to buy or rent easy as a click of a button. We shall employ an admin function that will ensure quality control over post to ensure content that is posted is not malicious in content.

Website url: http://18.223.29.166/

## Usability test plan

Test Objective:

We will be testing the post a listing function. Listings are integral to the success of our product. If landlords find it difficult to post their property, the purpose of our product is defeated. We want the landlords to be able to easily create a listing because more properties listed on the website will bring more potential renters to our site. We also think that the landlords will have less experience with software because they tend to be older. With this in consideration, we want to make the posting process as easy as possible.

Test Description:

The user will attempt to post a listing for a property they would like to lease. The user will attempt to use a firefox web browser to begin the process. The main purpose of this test is to measure the satisfaction of our landlords when they make a post. Users will start as a registered user on the homepage of our site. The job of the user is to post a property they want to rent.

We will collect data with a questionnaire after they have completed the task. The user will be tested on Chrome browser on a laptop. One of our team members will be observing the user while they post the listing. Feedback will be collected in a questionnaire and we will be evaluating the user satisfaction after they have completed the task. The user will be a volunteer. The final report will contain the level that the user agrees with the below statements.

Usability Task Description:

- Locate posting page
- Access posting page
- Recognize required fields
- Create listing and post it to the server

Questionnaire:

| | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
|---|---|---|---|---|---|
| The posting page was easily found. | | | | | |
| The required inputs were easily indicated. | | | | | |
| The inputs were relevant to my objective. | | | | | |
| Creating a listing was easy and intuitive. | | | | | |

## QA test plan

**Feature to be Tested:**
Post a listing feature (Only accessed by landlord)

**Test Objective:**

We are testing the “post a listing” feature for the landlords using our app. We are making sure this feature allows a user to create a database instance of a listing properly.

We are going to be testing three different types of inputs. One will be a normal address input, and the other two will be invalid input and an empty input. It would be optimal that when there is an invalid or empty input the user listing is notified immediately so that they can modify it and adhere to a greater overall user experience where addresses are accurate. Also checking the database after a post is created to ensure proper posting has been achieved, then searching for the listing after post is approved.

**Hardware/SW:**

We are using a Macbook Pro with two different browsers (Chrome and Firefox) to test the feature. It is currently being tested hosted from local machine, with database being pulled from AWS server. The feature will be tested from the “add a new listing” page which can be found on our local machine here:

The URL of the page is: http://127.0.0.1:5000/renter_dashboard/add_a_new_listing

**Browser used for testing: Firefox 66.0.5 (Official Build) (64-bit)**

| **ID** | **Test Description** | **Test Input** | **Expected Output** | **Actual Results** | **Pass/Fail** |
|---|---|---|---|---|---|
| T_01 | User enters a valid input while submitting a post request on listing page | 1600 Holloway Ave. San Francisco, CA | Listing gets posted successfully | Listing gets posted successfully | Pass |
| T_02 | Enter an "Invalid/alphanumeric Character/Input" in the input fields on listing page | "Jkjd9$!" in Street Number and ZipCode | A prompt with "Invalid Input" should be displayed to user | Listing gets posted successfully | Fail |
| T_03 | To test if the User is forced to populate the minimum input fields before submitting the post requests | All Fields Left Empty | Fields Required* Prompt should be displayed to the user | Listing gets posted successfully | Fail |

**Browser used for testing: Google Chrome 73.0.3683.103 (Official Build) (64-bit)**

| **ID** | **Test Description** | **Test Input** | **Expected Output** | **Actual Results** | **Pass/Fail** |
|---|---|---|---|---|---|
| T_01 | User enters a valid input while submitting a post request on listing page | 1600 Holloway Ave. San Francisco, CA | Listing gets posted successfully | Listing gets posted successfully | Pass |

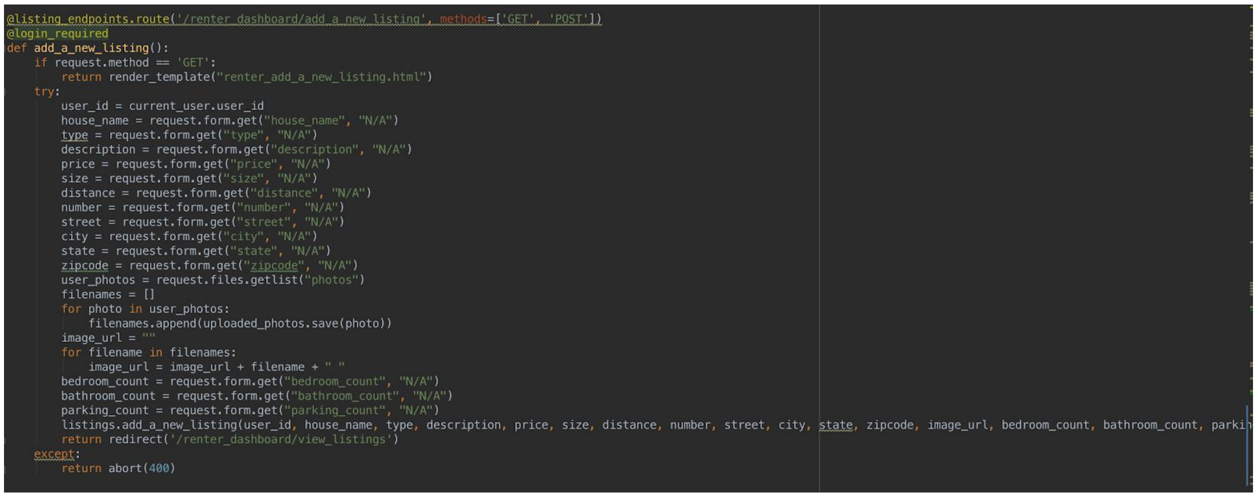
|  |  |  |  |  |  |
|---|---|---|---|---|---|
| T_02 | Enter an “Invalid/alphanumeric Character/Input” in the input fields on listing page | “Jkjd9$!” in Street Number and ZipCode | A prompt with “Invalid Input” should be displayed to user | Listing gets posted successfully | Fail |
| T_03 | To test if the User is forced to populate the minimum input fields before submitting the post requests | All Fields Left Empty | Fields Required* Prompt should be displayed to the user | Listing gets posted successfully | Fail |

## Code Review:

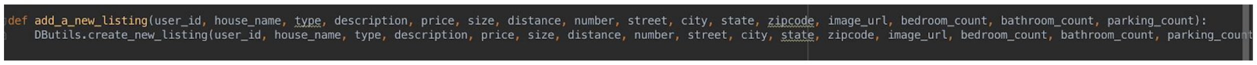
a) The coding style we have chosen is PEP 8

b) Example of the code under review (from post a listing feature):

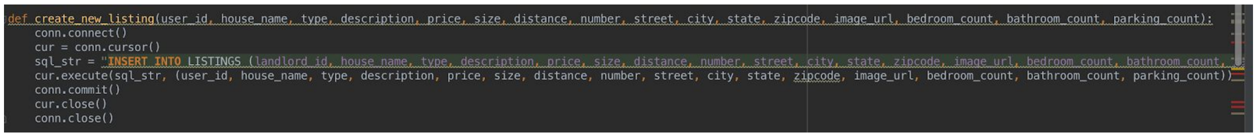
**Original code submitted by Adeel:**



```python
@listing_endpoints.route('/renter_dashboard/add_a_new_listing', methods=['GET', 'POST'])
@login_required
def add_a_new_listing():
    if request.method == 'GET':
        return render_template("renter_add_a_new_listing.html")
    try:
        user_id = current_user.user_id
        house_name = request.form.get("house_name", "N/A")
        type = request.form.get("type", "N/A")
        description = request.form.get("description", "N/A")
        price = request.form.get("price", "N/A")
        size = request.form.get("size", "N/A")
        distance = request.form.get("distance", "N/A")
        number = request.form.get("number", "N/A")
        street = request.form.get("street", "N/A")
        city = request.form.get("city", "N/A")
        state = request.form.get("state", "N/A")
        zipcode = request.form.get("zipcode", "N/A")
        user_photos = request.files.getlist("photos")
        filenames = []
        for photo in user_photos:
            filenames.append(uploaded_photos.save(photo))
        image_url = ""
        for filename in filenames:
            image_url = image_url + filename + " "
        bedroom_count = request.form.get("bedroom_count", "N/A")
        bathroom_count = request.form.get("bathroom_count", "N/A")
        parking_count = request.form.get("parking_count", "N/A")
        listings.add_a_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city, state, zipcode, image_url, bedroom_count, bathroom_count, parki
        return redirect('/renter_dashboard/view_listings')
    except:
        return abort(400)
```

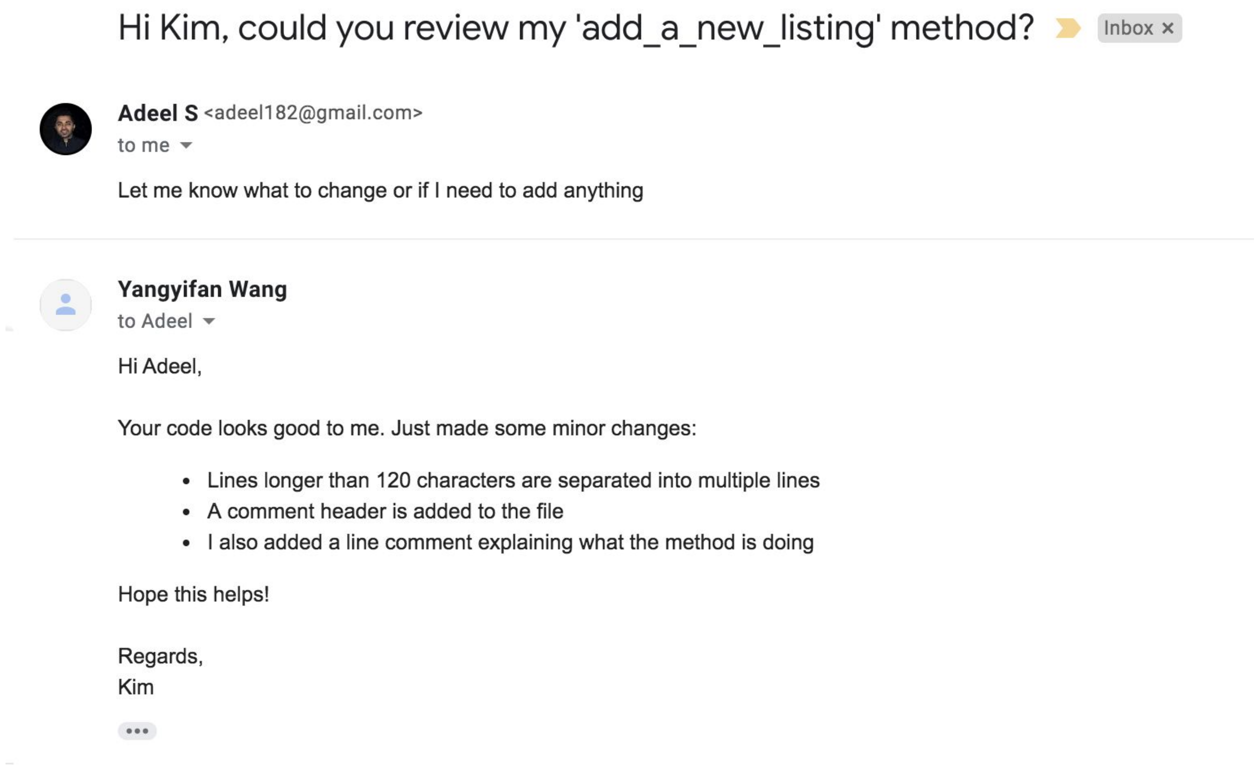


```python
def add_a_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city, state, zipcode, image_url, bedroom_count, bathroom_count, parking_count):
    DButils.create_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city, state, zipcode, image_url, bedroom_count, bathroom_count, parking_coun
```



```python
def create_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city, state, zipcode, image_url, bedroom_count, bathroom_count, parking_count):
    conn.connect()
    cur = conn.cursor()
    sql_str = "INSERT INTO LISTINGS (landlord_id, house_name, type, description, price, size, distance, number, street, city, state, zipcode, image_url, bedroom_count, bathroom_count,
    cur.execute(sql_str, (user_id, house_name, type, description, price, size, distance, number, street, city, state, zipcode, image_url, bedroom_count, bathroom_count, parking_count))
    conn.commit()
    cur.close()
    conn.close()
```

**Email communication:**


Hi Kim, could you review my 'add_a_new_listing' method?

Inbox ×

**Adeel S** <adeel182@gmail.com>
to me

Let me know what to change or if I need to add anything

**Yangyifan Wang**
to Adeel

Hi Adeel,

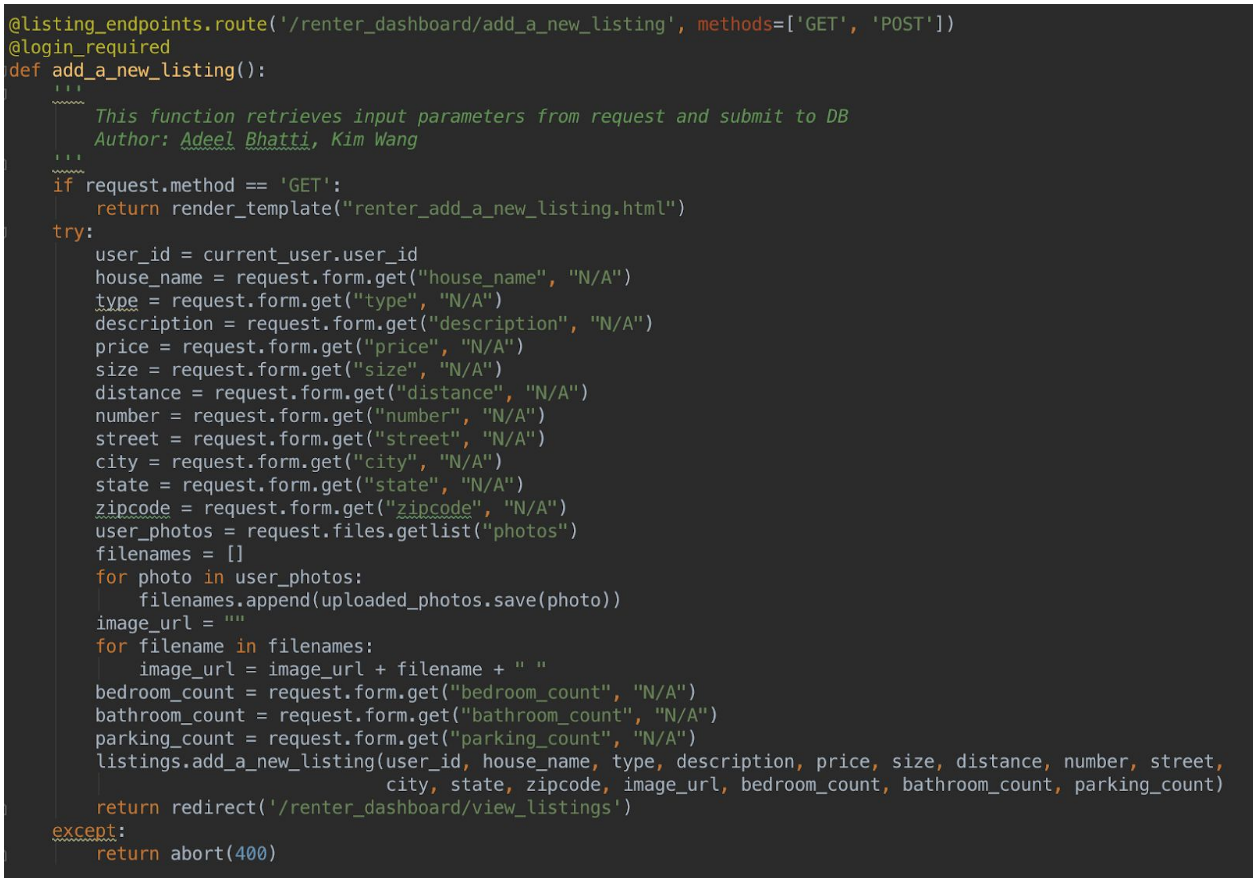
Your code looks good to me. Just made some minor changes:

- Lines longer than 120 characters are separated into multiple lines
- A comment header is added to the file
- I also added a line comment explaining what the method is doing

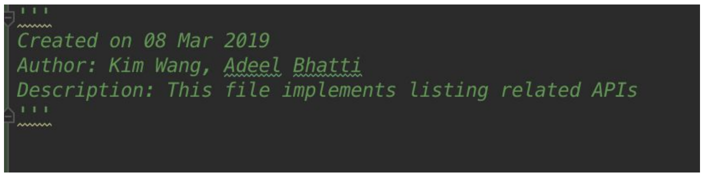
Hope this helps!

Regards,
Kim

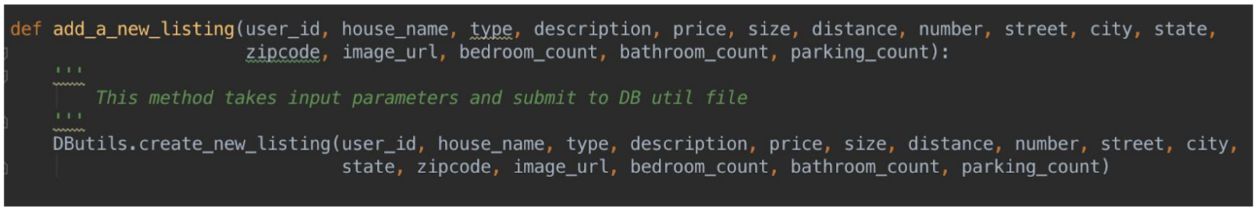
**Updated code after code review from Kim:**



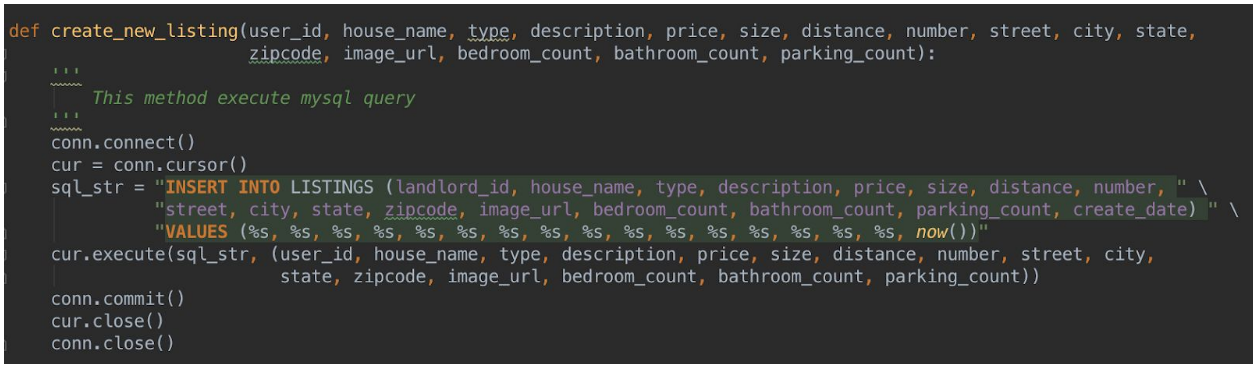
```python
@listing_endpoints.route('/renter_dashboard/add_a_new_listing', methods=['GET', 'POST'])
@login_required
def add_a_new_listing():
    '''
        This function retrieves input parameters from request and submit to DB
        Author: Adeel Bhatti, Kim Wang
    '''
    if request.method == 'GET':
        return render_template("renter_add_a_new_listing.html")
    try:
        user_id = current_user.user_id
        house_name = request.form.get("house_name", "N/A")
        type = request.form.get("type", "N/A")
        description = request.form.get("description", "N/A")
        price = request.form.get("price", "N/A")
        size = request.form.get("size", "N/A")
        distance = request.form.get("distance", "N/A")
        number = request.form.get("number", "N/A")
        street = request.form.get("street", "N/A")
        city = request.form.get("city", "N/A")
        state = request.form.get("state", "N/A")
        zipcode = request.form.get("zipcode", "N/A")
        user_photos = request.files.getlist("photos")
        filenames = []
        for photo in user_photos:
            filenames.append(uploaded_photos.save(photo))
        image_url = ""
        for filename in filenames:
            image_url = image_url + filename + " "
        bedroom_count = request.form.get("bedroom_count", "N/A")
        bathroom_count = request.form.get("bathroom_count", "N/A")
        parking_count = request.form.get("parking_count", "N/A")
        listings.add_a_new_listing(user_id, house_name, type, description, price, size, distance, number, street,
                                   city, state, zipcode, image_url, bedroom_count, bathroom_count, parking_count)
        return redirect('/renter_dashboard/view_listings')
    except:
        return abort(400)
```



```python
'''
Created on 08 Mar 2019
Author: Kim Wang, Adeel Bhatti
Description: This file implements listing related APIs
'''
```



```python
def add_a_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city, state,
                      zipcode, image_url, bedroom_count, bathroom_count, parking_count):
    '''
        This method takes input parameters and submit to DB util file
    '''
    DButils.create_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city,
                               state, zipcode, image_url, bedroom_count, bathroom_count, parking_count)
```



```python
def create_new_listing(user_id, house_name, type, description, price, size, distance, number, street, city, state,
                       zipcode, image_url, bedroom_count, bathroom_count, parking_count):
    '''
        This method execute mysql query
    '''
    conn.connect()
    cur = conn.cursor()
    sql_str = "INSERT INTO LISTINGS (landlord_id, house_name, type, description, price, size, distance, number, " \
              "street, city, state, zipcode, image_url, bedroom_count, bathroom_count, parking_count, create_date) " \
              "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, now())"
    cur.execute(sql_str, (user_id, house_name, type, description, price, size, distance, number, street, city,
                          state, zipcode, image_url, bedroom_count, bathroom_count, parking_count))
    conn.commit()
    cur.close()
    conn.close()
```

Code review communication email attached:

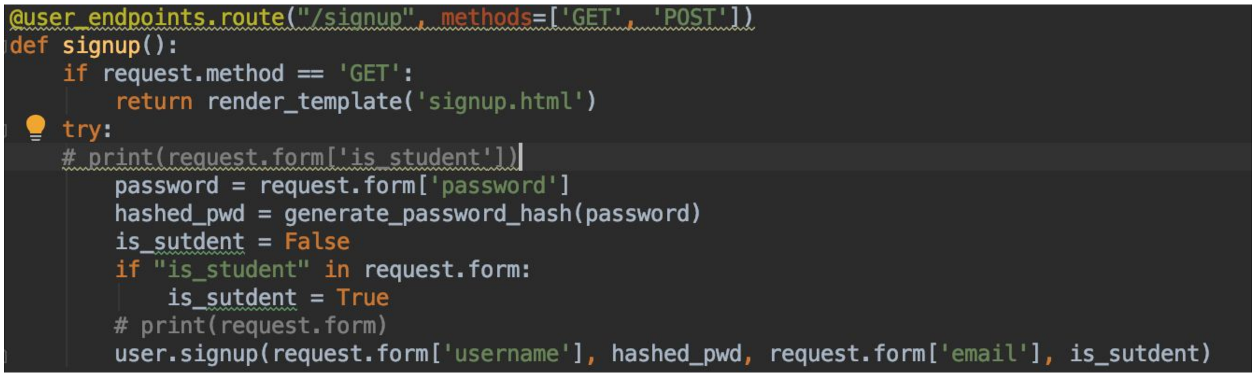
message:%3CCAC1ipTG5mssopz4NprCNAGmx72kt6SeVyRdrzOZey8OrgVykkg@mail.gmail.com%3E

## Self-Check on best practices for security

Major assets we are protecting:

1. User information such as username, password, email
2. Listing information such as description, location, and images

To encrypt the password, we are hashing the password, when the user signs up. This protects the user's personal information, if there was a breach in our database.



```
@user_endpoints.route("/signup", methods=['GET', 'POST'])
def signup():
    if request.method == 'GET':
        return render_template('signup.html')
    try:
    # print(request.form['is_student'])
        password = request.form['password']
        hashed_pwd = generate_password_hash(password)
        is_sutdent = False
        if "is_student" in request.form:
            is_sutdent = True
        # print(request.form)
        user.signup(request.form['username'], hashed_pwd, request.form['email'], is_sutdent)
```

To confirm input data validation, we are limiting the user input to a maximum of 40 characters in the username, password, and email fields. We are also encrypting the user's information and validating the login. We do validation for the search bar based on price, size, and distance.

## Self-Check: Adherence to original Non-functional specs

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

- 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.

- Done

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

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

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

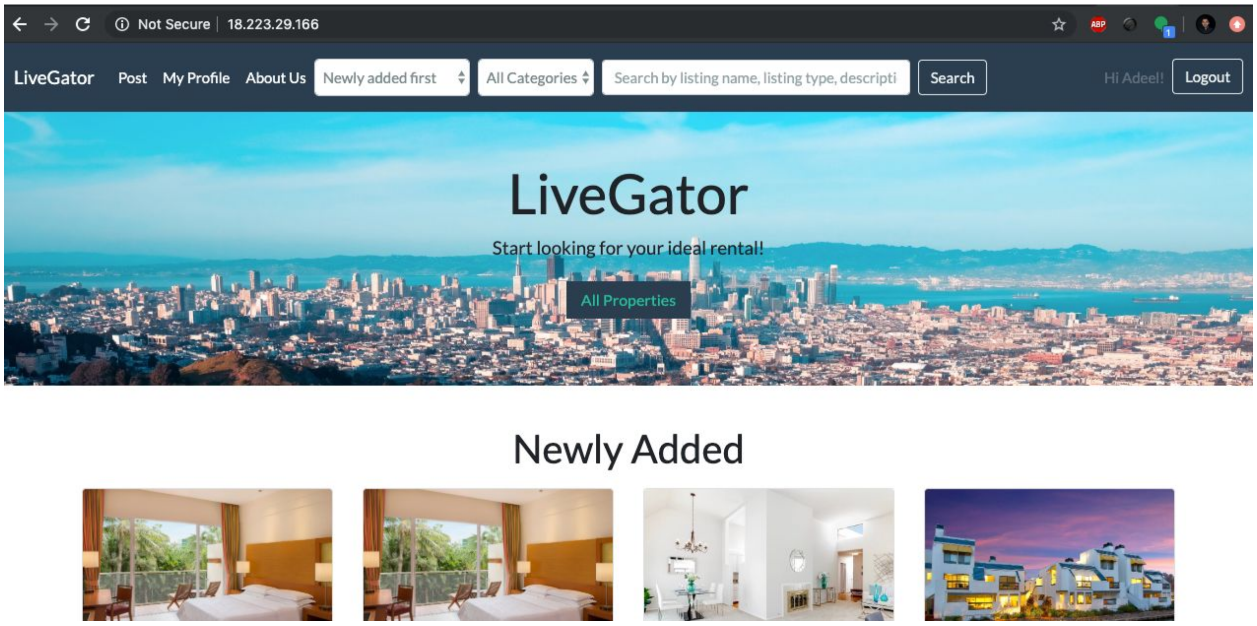
14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
    - Done / Continuous development

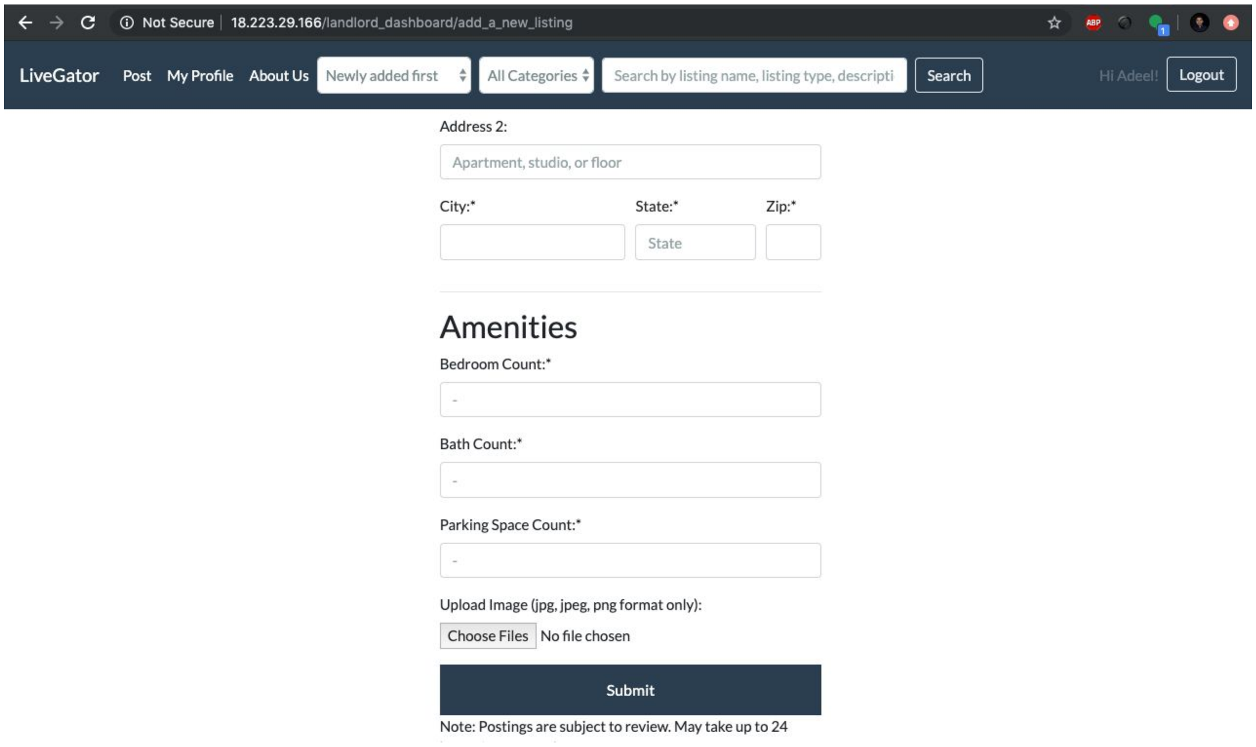
15. The website shall prominently display the following exact text on all pages "SFSU

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).

- On Track

**4) Screenshots of actual final product as shown in the demo, one per page**





Not Secure | 18.223.29.166/landlord_dashboard/add_a_new_listing

Address 2:

Apartment, studio, or floor

City:* State:* Zip:*

State

## Amenities

Bedroom Count:*

-

Bath Count:*

-

Parking Space Count:*

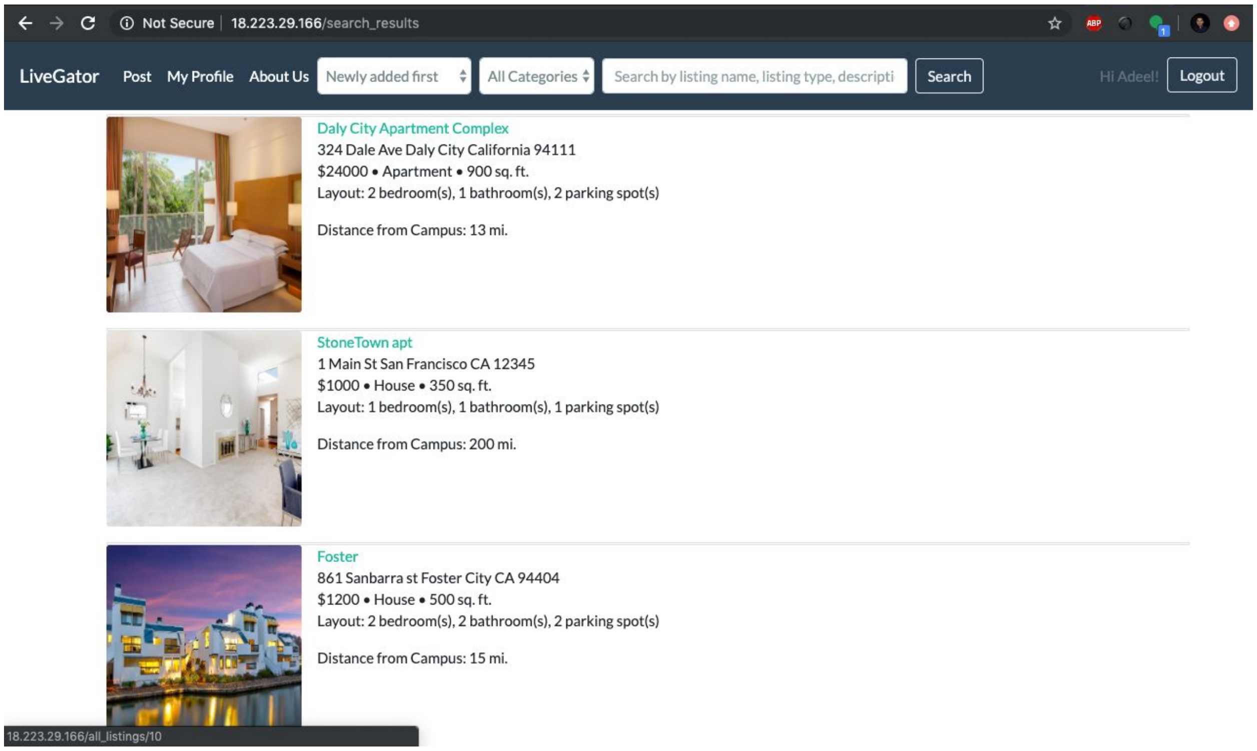
-

Upload Image (jpg, jpeg, png format only):

Choose Files No file chosen

Submit

Note: Postings are subject to review. May take up to 24


Not Secure | 18.223.29.166/search_results
LiveGator
Post
My Profile
About Us
Newly added first
All Categories
Search by listing name, listing type, descripti
Search
Hi Adeel!
Logout

Daly City Apartment Complex
324 Dale Ave Daly City California 94111
$24000 • Apartment • 900 sq. ft.
Layout: 2 bedroom(s), 1 bathroom(s), 2 parking spot(s)

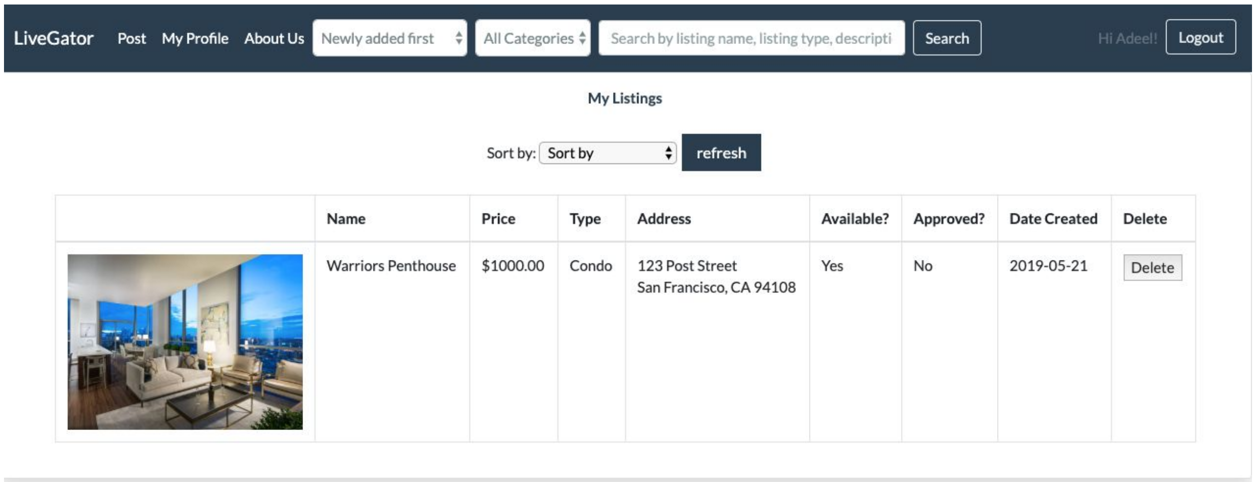
Distance from Campus: 13 mi.

StoneTown apt
1 Main St San Francisco CA 12345
$1000 • House • 350 sq. ft.
Layout: 1 bedroom(s), 1 bathroom(s), 1 parking spot(s)

Distance from Campus: 200 mi.

Foster
861 Sanbarra st Foster City CA 94404
$1200 • House • 500 sq. ft.
Layout: 2 bedroom(s), 2 bathroom(s), 2 parking spot(s)

Distance from Campus: 15 mi.

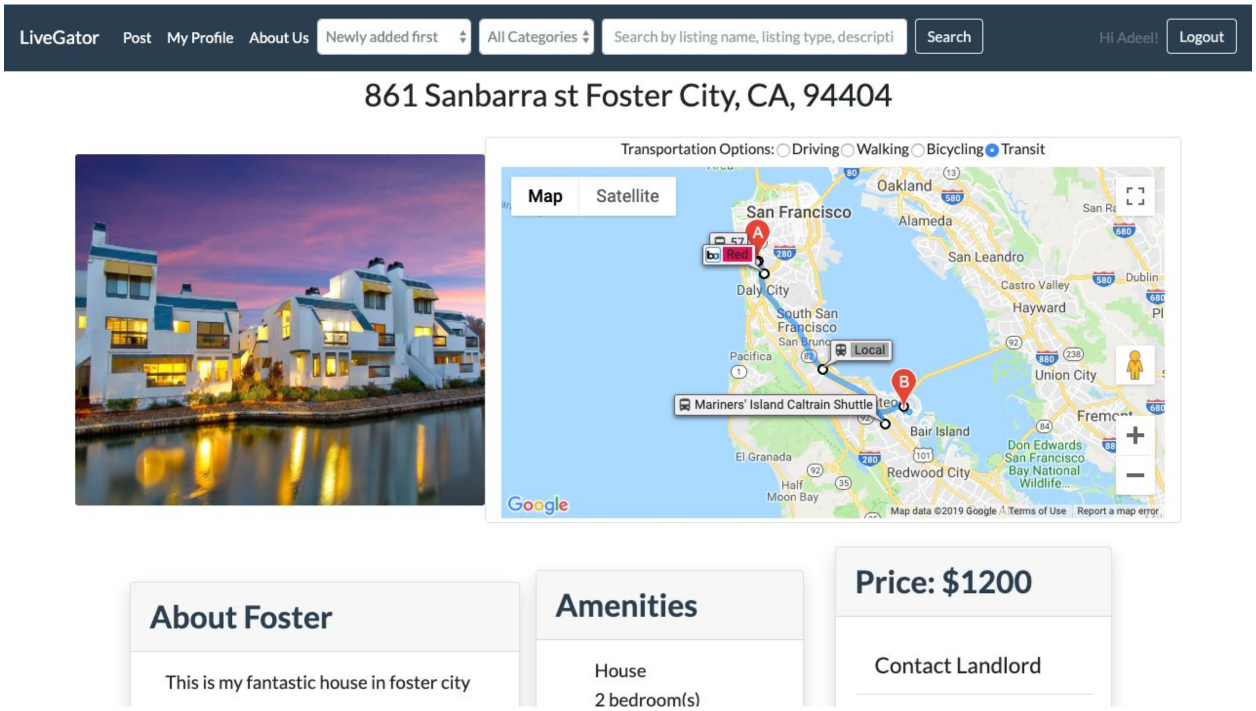


LiveGator Post My Profile About Us Newly added first All Categories Search by listing name, listing type, descripti Search Hi Adeel! Logout

My Listings

Sort by: Sort by refresh

| | Name | Price | Type | Address | Available? | Approved? | Date Created | Delete |
|---|---|---|---|---|---|---|---|---|
| | Warriors Penthouse | $1000.00 | Condo | 123 Post Street<br>San Francisco, CA 94108 | Yes | No | 2019-05-21 | Delete |



LiveGator Post My Profile About Us Newly added first All Categories Search by listing name, listing type, descripti Search Hi Adeel! Logout

861 Sanbarra st Foster City, CA, 94404

Transportation Options: Driving Walking Bicycling [✓ Transit]

**About Foster**

This is my fantastic house in foster city

**Amenities**

House
2 bedroom(s)

**Price: $1200**

Contact Landlord

## 5) Screen shots of key DB tables (1-2 pages)

Show a snapshots of all important DB tables such as user table, item table, category table. Make it easy to read and review

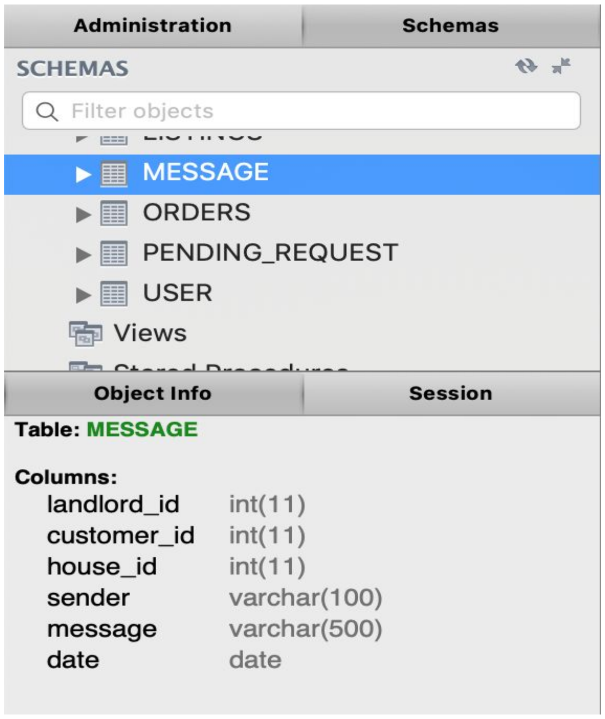
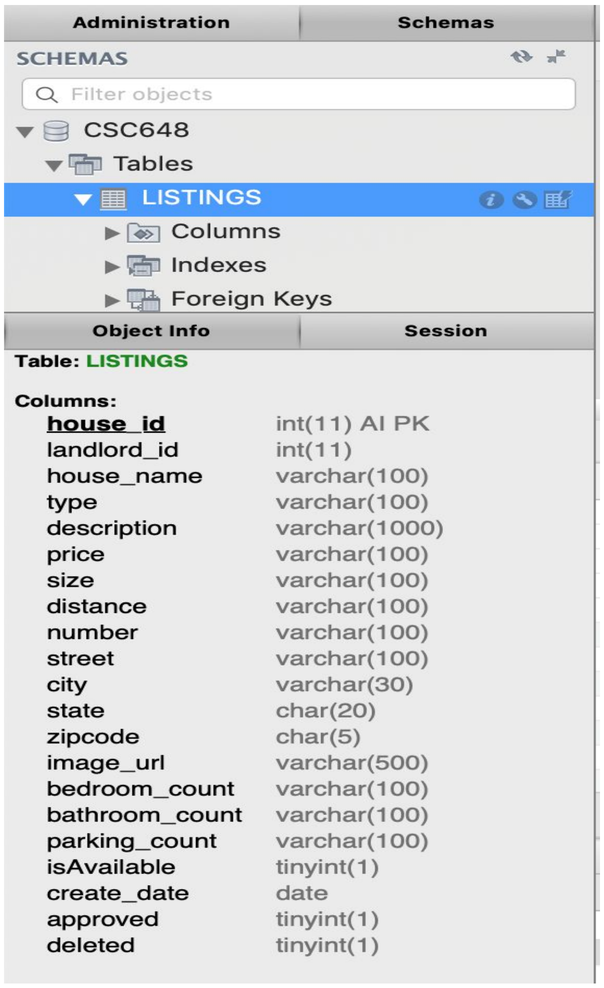
| Table Message | Table Listings |
|---|---|
|  |  |

**Table: MESSAGE**

**Columns:**

| | |
|---|---|
| landlord_id | int(11) |
| customer_id | int(11) |
| house_id | int(11) |
| sender | varchar(100) |
| message | varchar(500) |
| date | date |

**Table: LISTINGS**

**Columns:**

| | |
|---|---|
| **house_id** | int(11) AI PK |
| landlord_id | int(11) |
| house_name | varchar(100) |
| type | varchar(100) |
| description | varchar(1000) |
| price | varchar(100) |
| size | varchar(100) |
| distance | varchar(100) |
| number | varchar(100) |
| street | varchar(100) |
| city | varchar(30) |
| state | char(20) |
| zipcode | char(5) |
| image_url | varchar(500) |
| bedroom_count | varchar(100) |
| bathroom_count | varchar(100) |
| parking_count | varchar(100) |
| isAvailable | tinyint(1) |
| create_date | date |
| approved | tinyint(1) |
| deleted | tinyint(1) |

| Table User |
|---|

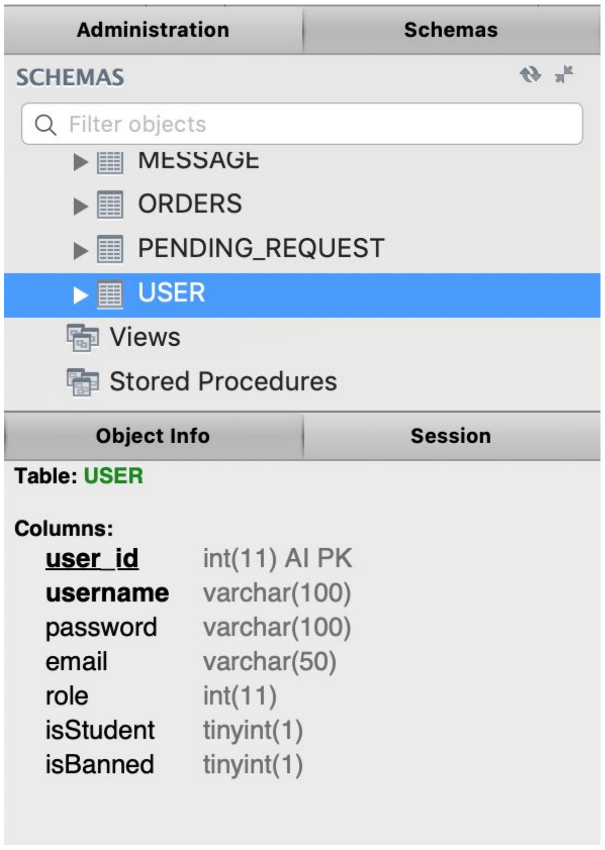
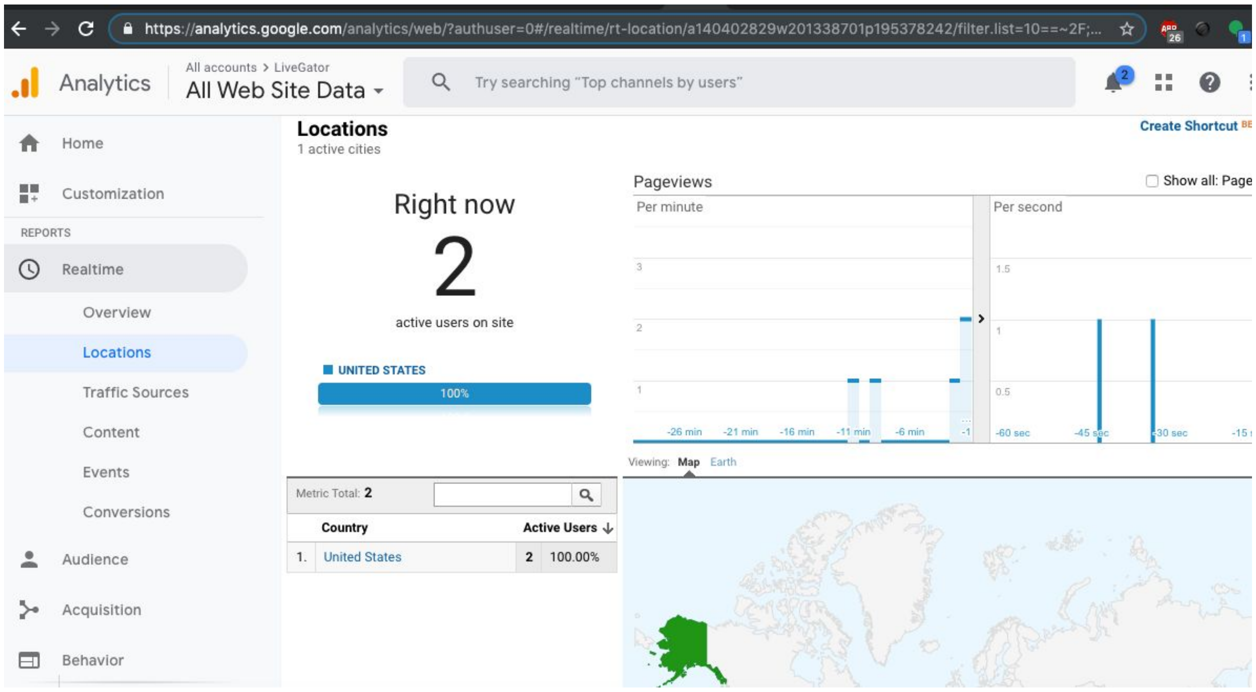


Table: USER

Columns:

```
user_id     int(11) AI PK
username    varchar(100)
password    varchar(100)
email       varchar(50)
role        int(11)
isStudent   tinyint(1)
isBanned    tinyint(1)
```

**6) Google analytics plot for your WWW site (1 page)**



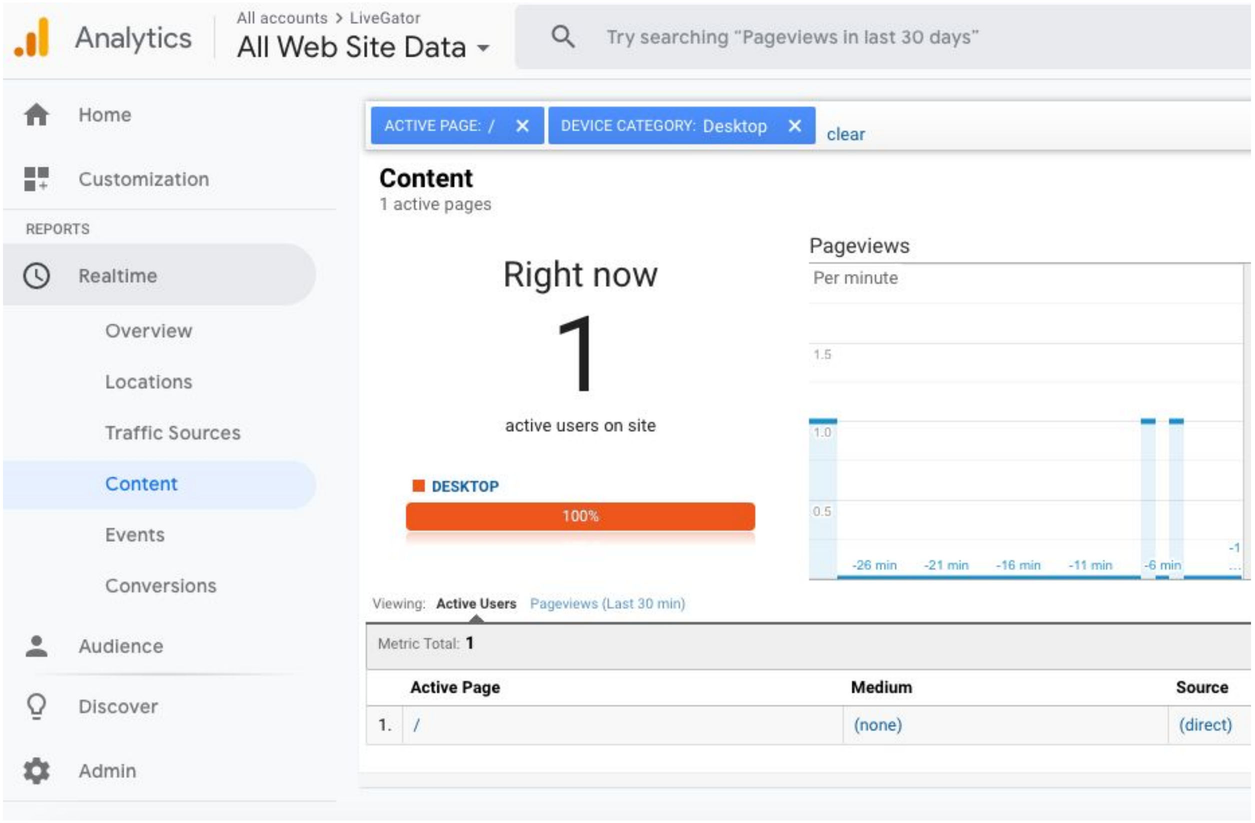
Locations
1 active cities

Right now
2
active users on site

UNITED STATES 100%

Metric Total: 2

| | Country | Active Users |
|---|---|---|
| 1. | United States | 2 100.00% |



ACTIVE PAGE: / DEVICE CATEGORY: Desktop clear

Content
1 active pages

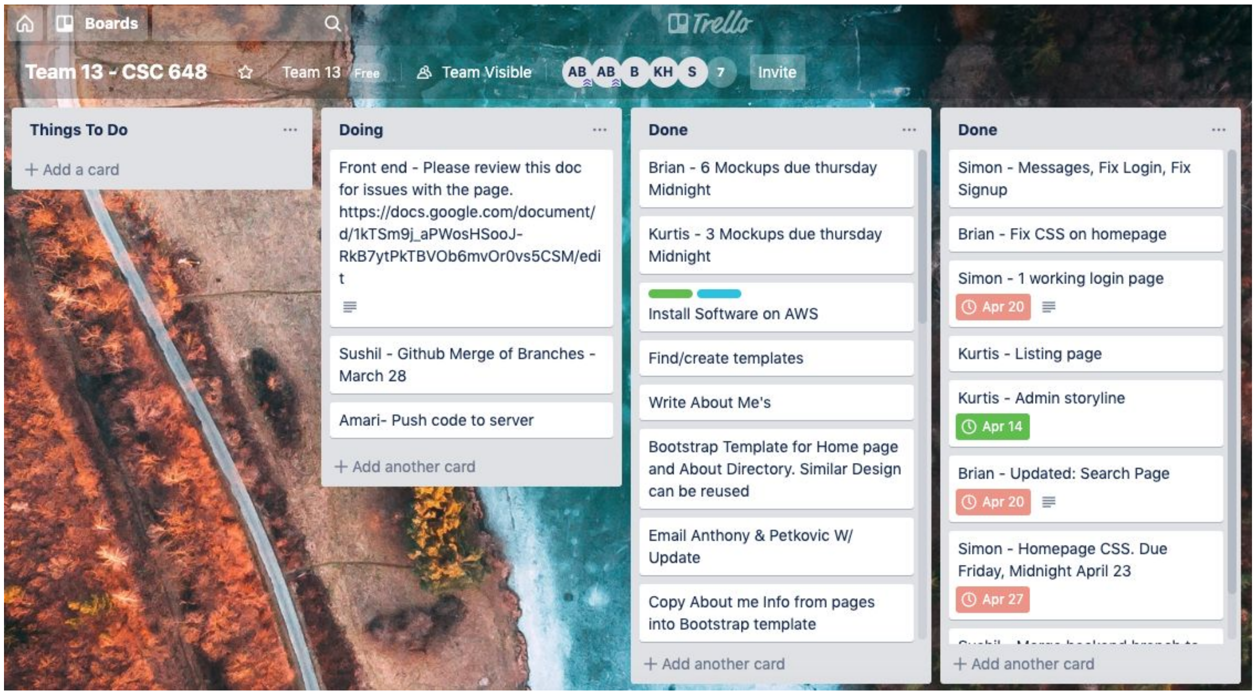
Right now
1
active users on site

DESKTOP 100%

Metric Total: 1

| | Active Page | Medium | Source |
|---|---|---|---|
| 1. | / | (none) | (direct) |

**7) Screen shot(s) of your task management system (like Trello) showing a snapshot of your project management**



**Things To Do**

+ Add a card

**Doing**

- Front end - Please review this doc for issues with the page. https://docs.google.com/document/d/1kTSm9j_aPWosHSooJ-RkB7ytPkTBVOb6mvOr0vs5CSM/edit
- Sushil - Github Merge of Branches - March 28
- Amari- Push code to server

+ Add another card

**Done**

- Brian - 6 Mockups due thursday Midnight
- Kurtis - 3 Mockups due thursday Midnight
- Install Software on AWS
- Find/create templates
- Write About Me's
- Bootstrap Template for Home page and About Directory. Similar Design can be reused
- Email Anthony & Petkovic W/ Update
- Copy About me Info from pages into Bootstrap template

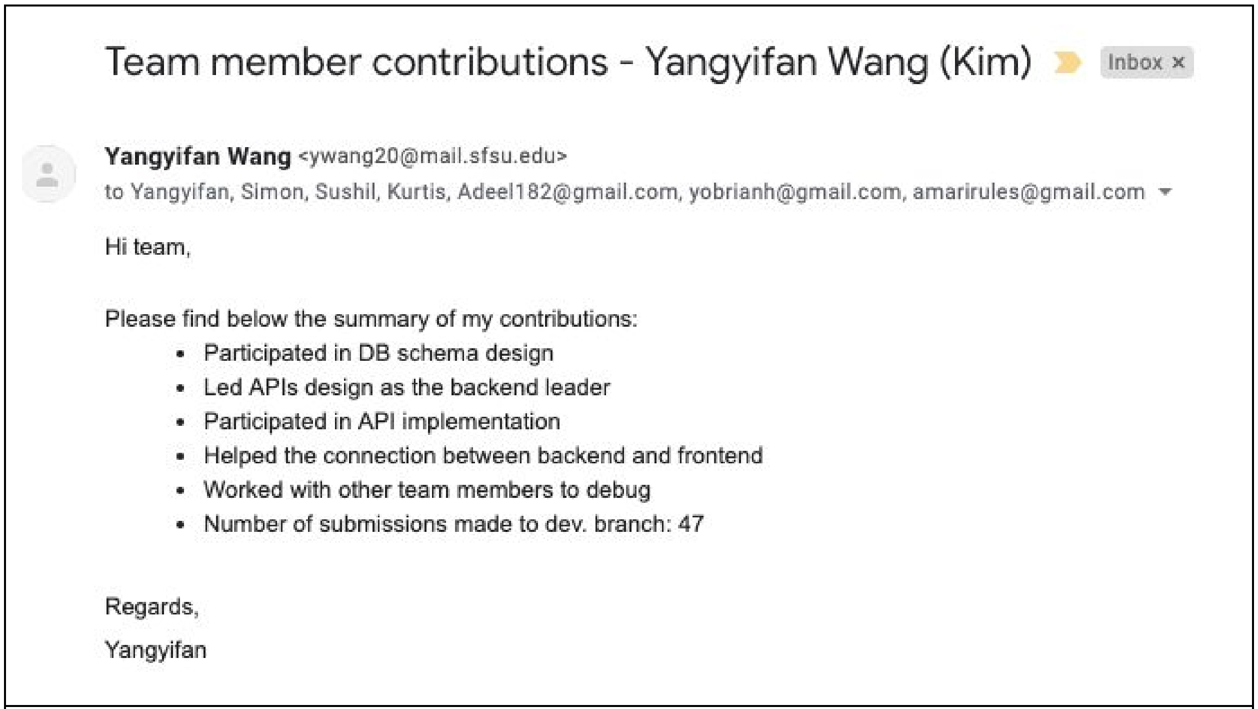
+ Add another card

**Done**

- Simon - Messages, Fix Login, Fix Signup
- Brian - Fix CSS on homepage
- Simon - 1 working login page — Apr 20
- Kurtis - Listing page
- Kurtis - Admin storyline — Apr 14
- Brian - Updated: Search Page — Apr 20
- Simon - Homepage CSS. Due Friday, Midnight April 23 — Apr 27

+ Add another card

## 8) Team member contributions



Team member contributions - Yangyifan Wang (Kim) Inbox ×

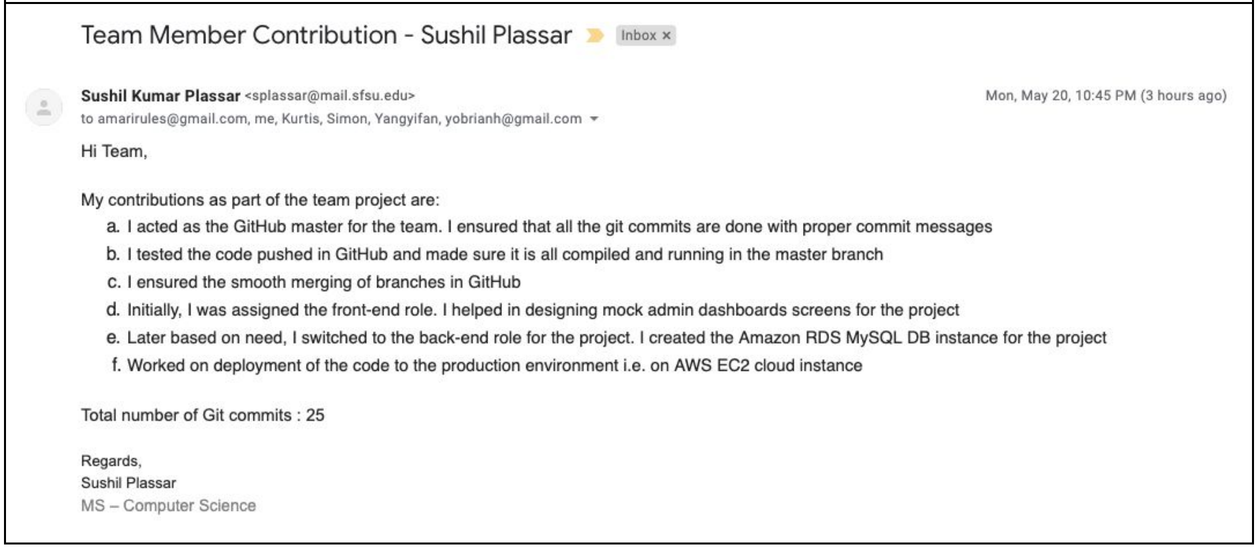
**Yangyifan Wang** <ywang20@mail.sfsu.edu>
to Yangyifan, Simon, Sushil, Kurtis, Adeel182@gmail.com, yobrianh@gmail.com, amarirules@gmail.com

Hi team,

Please find below the summary of my contributions:

- Participated in DB schema design
- Led APIs design as the backend leader
- Participated in API implementation
- Helped the connection between backend and frontend
- Worked with other team members to debug
- Number of submissions made to dev. branch: 47

Regards,
Yangyifan



Team Member Contribution - Sushil Plassar Inbox ×

**Sushil Kumar Plassar** <splassar@mail.sfsu.edu> Mon, May 20, 10:45 PM (3 hours ago)
to amarirules@gmail.com, me, Kurtis, Simon, Yangyifan, yobrianh@gmail.com

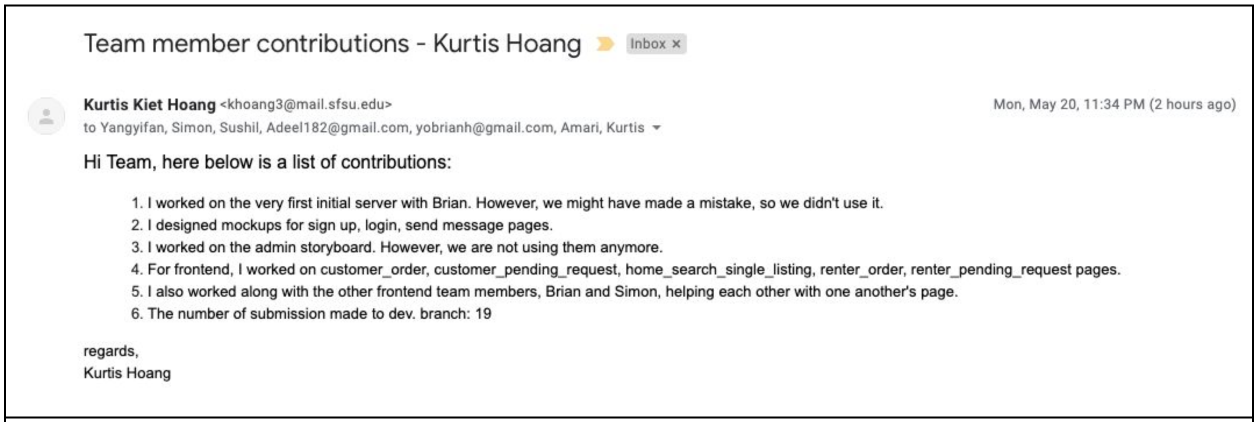
Hi Team,

My contributions as part of the team project are:

a. I acted as the GitHub master for the team. I ensured that all the git commits are done with proper commit messages
b. I tested the code pushed in GitHub and made sure it is all compiled and running in the master branch
c. I ensured the smooth merging of branches in GitHub
d. Initially, I was assigned the front-end role. I helped in designing mock admin dashboards screens for the project
e. Later based on need, I switched to the back-end role for the project. I created the Amazon RDS MySQL DB instance for the project
f. Worked on deployment of the code to the production environment i.e. on AWS EC2 cloud instance

Total number of Git commits : 25

Regards,
Sushil Plassar
MS – Computer Science



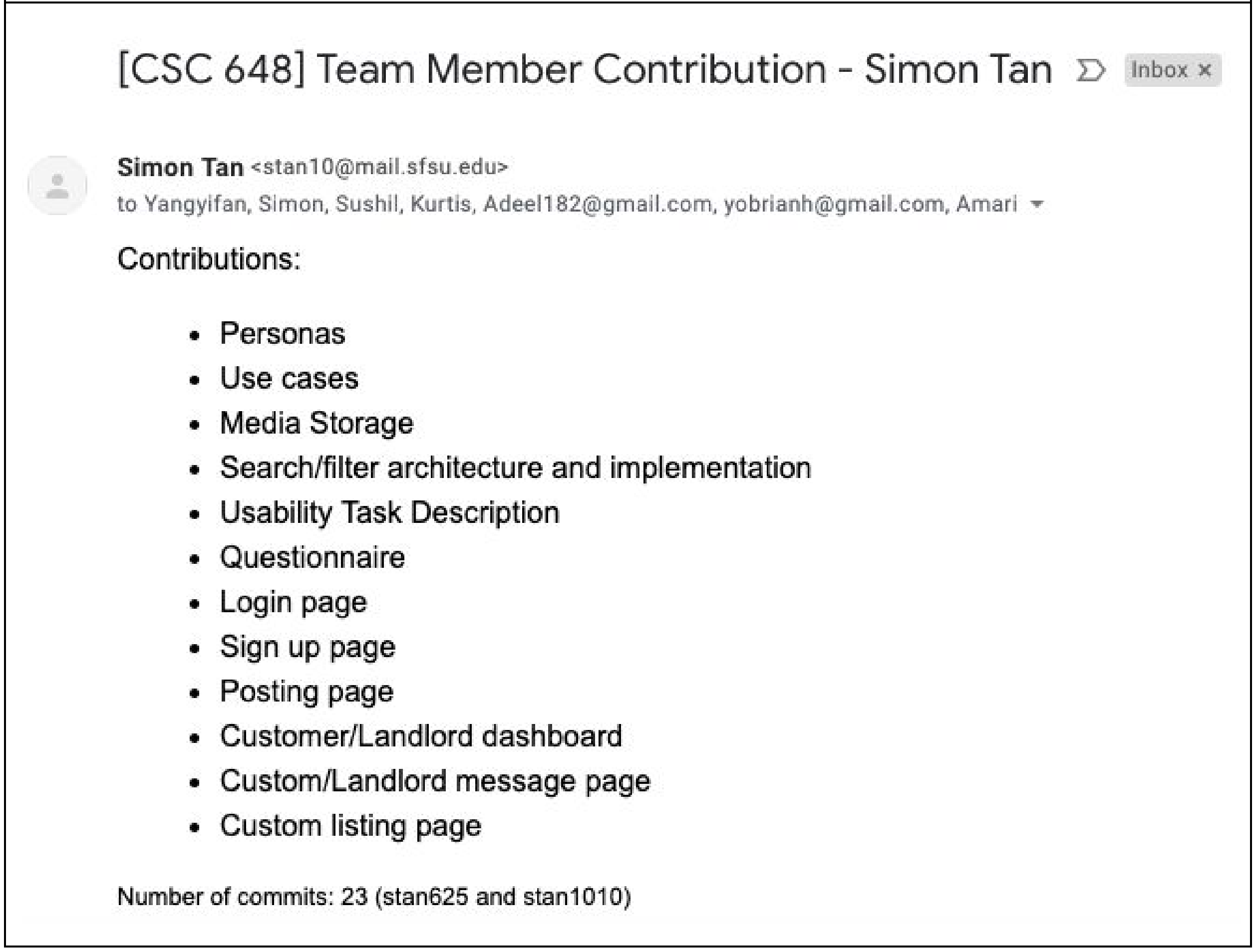
Team member contributions - Kurtis Hoang Inbox ×

**Kurtis Kiet Hoang** <khoang3@mail.sfsu.edu> Mon, May 20, 11:34 PM (2 hours ago)
to Yangyifan, Simon, Sushil, Adeel182@gmail.com, yobrianh@gmail.com, Amari, Kurtis

Hi Team, here below is a list of contributions:

1. I worked on the very first initial server with Brian. However, we might have made a mistake, so we didn't use it.
2. I designed mockups for sign up, login, send message pages.
3. I worked on the admin storyboard. However, we are not using them anymore.
4. For frontend, I worked on customer_order, customer_pending_request, home_search_single_listing, renter_order, renter_pending_request pages.
5. I also worked along with the other frontend team members, Brian and Simon, helping each other with one another's page.
6. The number of submission made to dev. branch: 19

regards,
Kurtis Hoang



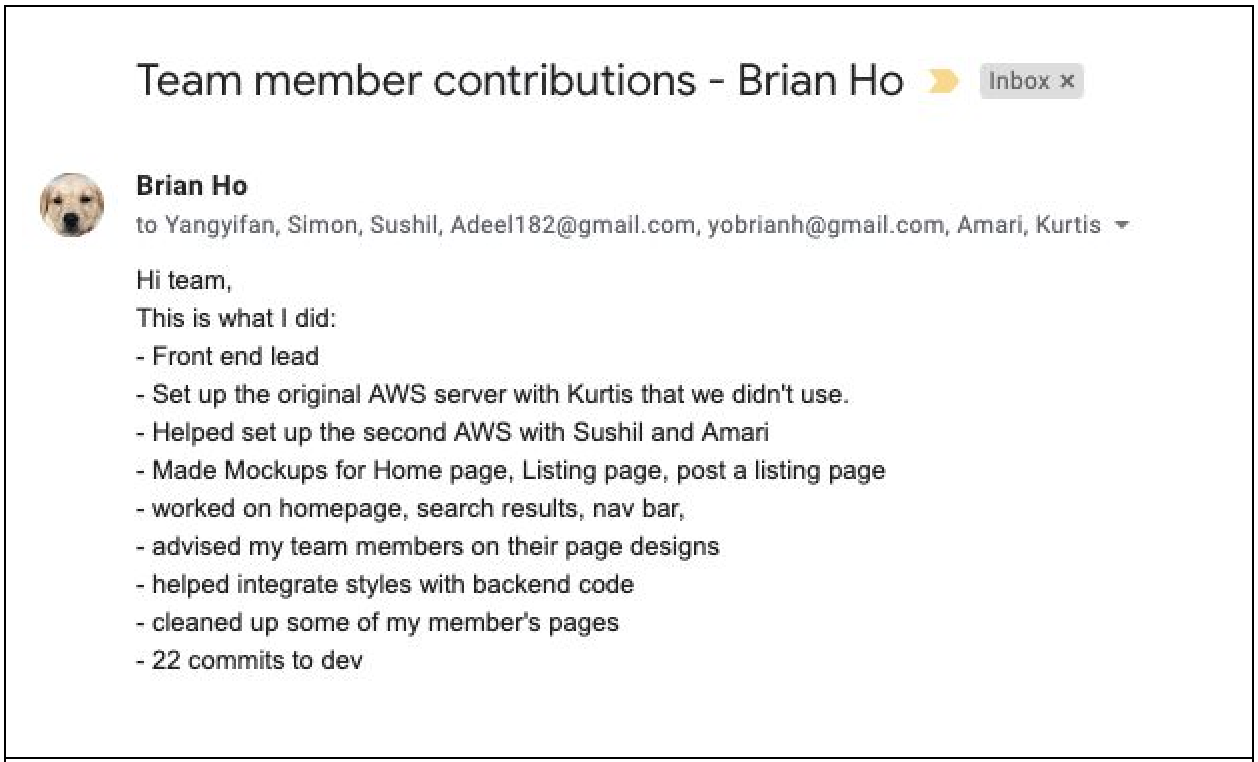
[CSC 648] Team Member Contribution - Simon Tan Inbox ×

**Simon Tan** <stan10@mail.sfsu.edu>
to Yangyifan, Simon, Sushil, Kurtis, Adeel182@gmail.com, yobrianh@gmail.com, Amari

Contributions:

- Personas
- Use cases
- Media Storage
- Search/filter architecture and implementation
- Usability Task Description
- Questionnaire
- Login page
- Sign up page
- Posting page
- Customer/Landlord dashboard
- Custom/Landlord message page
- Custom listing page

Number of commits: 23 (stan625 and stan1010)



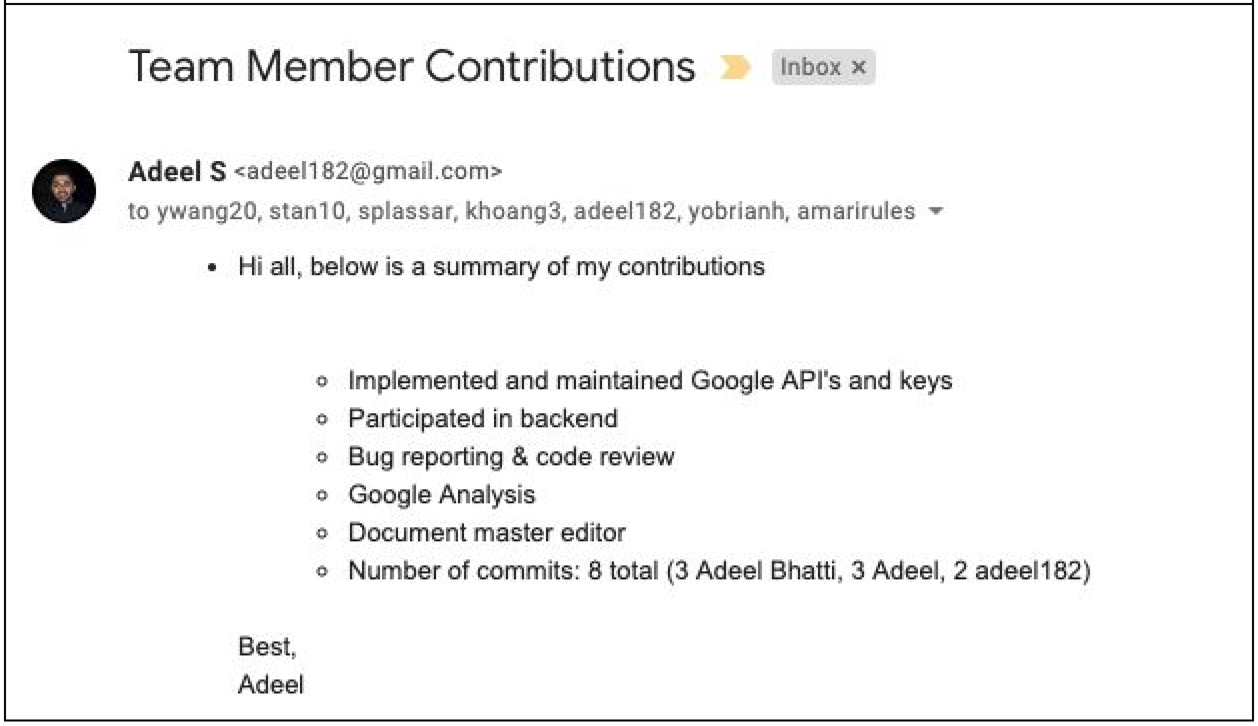
Team member contributions - Brian Ho

Inbox ×

**Brian Ho**

to Yangyifan, Simon, Sushil, Adeel182@gmail.com, yobrianh@gmail.com, Amari, Kurtis

Hi team,
This is what I did:
- Front end lead
- Set up the original AWS server with Kurtis that we didn't use.
- Helped set up the second AWS with Sushil and Amari
- Made Mockups for Home page, Listing page, post a listing page
- worked on homepage, search results, nav bar,
- advised my team members on their page designs
- helped integrate styles with backend code
- cleaned up some of my member's pages
- 22 commits to dev



Team Member Contributions

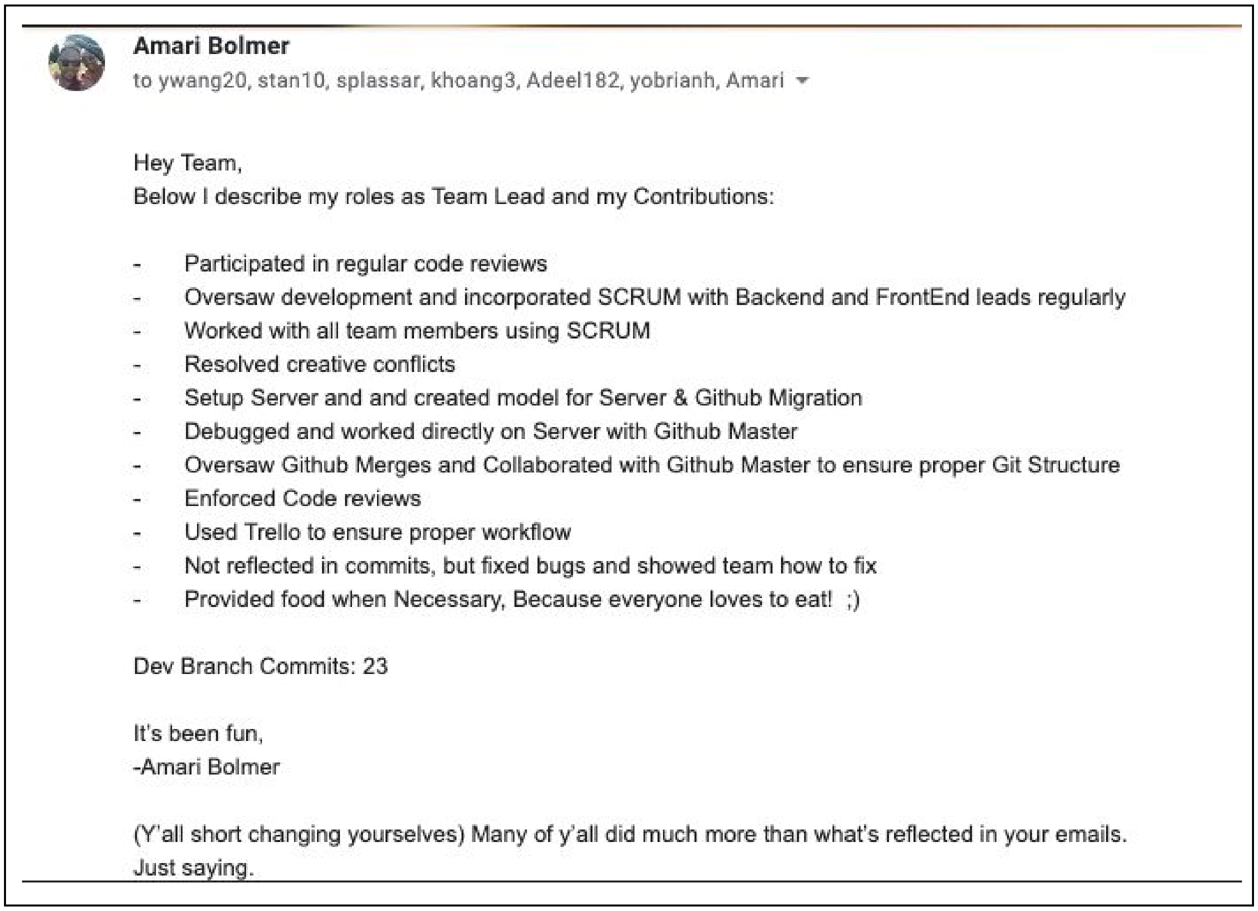
Inbox ×

**Adeel S** <adeel182@gmail.com>

to ywang20, stan10, splassar, khoang3, adeel182, yobrianh, amarirules

- Hi all, below is a summary of my contributions
    - Implemented and maintained Google API's and keys
    - Participated in backend
    - Bug reporting & code review
    - Google Analysis
    - Document master editor
    - Number of commits: 8 total (3 Adeel Bhatti, 3 Adeel, 2 adeel182)

Best,
Adeel



Amari Bolmer
to ywang20, stan10, splassar, khoang3, Adeel182, yobrianh, Amari

Hey Team,
Below I describe my roles as Team Lead and my Contributions:

- Participated in regular code reviews
- Oversaw development and incorporated SCRUM with Backend and FrontEnd leads regularly
- Worked with all team members using SCRUM
- Resolved creative conflicts
- Setup Server and and created model for Server & Github Migration
- Debugged and worked directly on Server with Github Master
- Oversaw Github Merges and Collaborated with Github Master to ensure proper Git Structure
- Enforced Code reviews
- Used Trello to ensure proper workflow
- Not reflected in commits, but fixed bugs and showed team how to fix
- Provided food when Necessary, Because everyone loves to eat! ;)

Dev Branch Commits: 23

It's been fun,
-Amari Bolmer

(Y'all short changing yourselves) Many of y'all did much more than what's reflected in your emails.
Just saying.

## 9) Post analysis – lessons learned

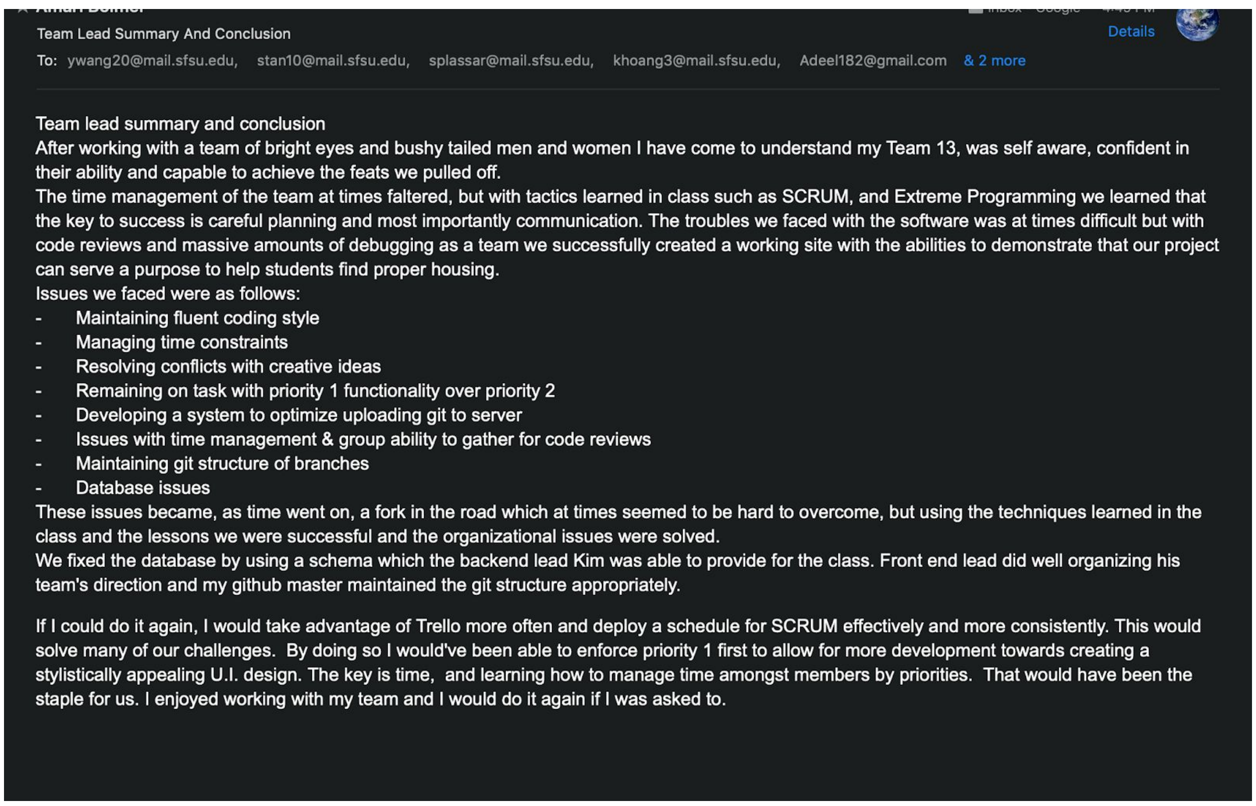
Team lead summary and conclusion
After working with a team of bright eyes and bushy tailed men and women I have come to understand my Team 13, was self aware, confident in their ability and capable to achieve the feats we pulled off.
The time management of the team at times faltered, but with tactics learned in class such as SCRUM, and Extreme Programming we learned that the key to success is careful planning and most importantly communication. The troubles we faced with the software was at times difficult but with code reviews and massive amounts of debugging as a team we successfully created a working site with the abilities to demonstrate that our project can serve a purpose to help students find proper housing.
Issues we faced were as follows:

- Maintaining fluent coding style
- Managing time constraints
- Resolving conflicts with creative ideas

- Remaining on task with priority 1 functionality over priority 2
- Developing a system to optimize uploading git to server
- Issues with time management & group ability to gather for code reviews
- Maintaining git structure of branches
- Database issues

These issues became, as time went on, a fork in the road which at times seemed to be hard to overcome, but using the techniques learned in the class and the lessons we were successful and the organizational issues were solved.
We fixed the database by using a schema which the backend lead Kim was able to provide for the class. Front end lead did well organizing his team's direction and my github master maintained the git structure appropriately.

If I could do it again, I would take advantage of Trello more often and deploy a schedule for SCRUM effectively and more consistently. This would solve many of our challenges. By doing so I would've been able to enforce priority 1 first to allow for more development towards creating a stylistically appealing U.I. design. The key is time, and learning how to manage time amongst members by priorities. That would have been the staple for us. I enjoyed working with my team and I would do it again if I was asked to.



Team Lead Summary And Conclusion

To: ywang20@mail.sfsu.edu, stan10@mail.sfsu.edu, splassar@mail.sfsu.edu, khoang3@mail.sfsu.edu, Adeel182@gmail.com & 2 more

Team lead summary and conclusion
After working with a team of bright eyes and bushy tailed men and women I have come to understand my Team 13, was self aware, confident in their ability and capable to achieve the feats we pulled off.
The time management of the team at times faltered, but with tactics learned in class such as SCRUM, and Extreme Programming we learned that the key to success is careful planning and most importantly communication. The troubles we faced with the software was at times difficult but with code reviews and massive amounts of debugging as a team we successfully created a working site with the abilities to demonstrate that our project can serve a purpose to help students find proper housing.
Issues we faced were as follows:
- Maintaining fluent coding style
- Managing time constraints
- Resolving conflicts with creative ideas

- Remaining on task with priority 1 functionality over priority 2
- Developing a system to optimize uploading git to server
- Issues with time management & group ability to gather for code reviews
- Maintaining git structure of branches
- Database issues

These issues became, as time went on, a fork in the road which at times seemed to be hard to overcome, but using the techniques learned in the class and the lessons we were successful and the organizational issues were solved.
We fixed the database by using a schema which the backend lead Kim was able to provide for the class. Front end lead did well organizing his team's direction and my github master maintained the git structure appropriately.

If I could do it again, I would take advantage of Trello more often and deploy a schedule for SCRUM effectively and more consistently. This would solve many of our challenges. By doing so I would've been able to enforce priority 1 first to allow for more development towards creating a stylistically appealing U.I. design. The key is time, and learning how to manage time amongst members by priorities. That would have been the staple for us. I enjoyed working with my team and I would do it again if I was asked to.