

SW Engineering CSC648/848 Spring 2019

SFSURent

Team 11

Cory Lewis(Team Lead, clewis9@mail.sfsu.edu)

Xinyu Zou(GitHub Master)

Soheil Ansari(Back End Lead)

Junwei Liang

Poorva Rathi

David Dropping(Front End Lead)

Chintan Sanjay Puri

Milestone 2

03/21/19

History Table:

1. Data Definitions

1.1. User

- 1.1.1. *Unregistered User* (can browse the website but won't be able to contact the Landlord and post listings)
- 1.1.2. *Registered User* (can perform all actions of an Unregistered User and contact the Landlord and post listings)
 - 1.1.2.1. *Tenant* (can view the public images but need to register to see private images and to communicate with landlord)
 - 1.1.2.2. *Landlord* (can post the images and details of the house; login/register required)
 - 1.1.2.3. *Admin* (can access, modify, edit and delete all the necessary data; can block the fraud user; login required)
 - 1.1.2.4. *Blocked user* (details of user's blocked by the admin)

1.2. Listing

- 1.2.1. *Subject/Title* (listing header text)
- 1.2.2. *Address and rent details of the house*
- 1.2.3. *Landlord ID* (to track the ownership of the listing)
- 1.2.4. *Listing Images* (posted by Landlord and approved by Admin)
 - 1.2.4.1. *Image formats: jpg, png*
- 1.2.5. *Listing Features* (features like furnished/not furnished, has pool, has air conditioner, pet allowed, carpeted or not, number of bathrooms, another roommate's information and many more)
- 1.2.6. *Type of Listing and size* (type like 2BHK/3BHK, apartment/ bungalow, floor)
- 1.2.7. *Distance from university* (for each commuting mode).
- 1.2.8. *Time from the university* (for each commuting mode).
- 1.2.9. *Listing status (approved / disapproved / expired / rented)*
- 1.2.10. *Posted Timestamp*
- 1.2.11. *Rating* (a rating for the listing, calculated based on size, features, distance from university, etc.)

1.3. Message

- 1.3.1. *Sender ID* (to determine who has sent the message)
- 1.3.2. *Listing ID* (to track the related listing)
- 1.3.3. *Message Body*
- 1.3.4. *Timestamp*

1.4. *Tenant Search History* (history of the search by the tenant to be able to find the previous results)

1.5. *Favorite listings* (tenant's favorite listings which will help the tenant to find the same listing again)

2. Functional Requirements

Priority 1

2.1 Unregistered User

- 2.1.1 Shall browse listings by simply visiting the website.
- 2.1.3 Shall be able to view the listings on a map.
- 2.1.5 Shall be able to read the description of postings.
- 2.1.7 Shall view images of the listing to give the user an idea of the living circumstances.
- 2.1.8 Shall register for an account by completing an online form.

2.2 Registered User (Can perform all tasks of Unregistered User)

- 2.2.1 Shall contact landlord through the platform.
- 2.2.3 Shall logout of session by clicking logout.
- 2.2.4 Shall be able to edit their profile.
- 2.2.5 Shall change password for login by viewing profile and clicking “change password”.
- 2.2.6 Shall post listings.

2.3 Landlord

- 2.3.2 Shall view postings.
- 2.3.3 Shall be able to view and response messages from potential tenants.
- 2.3.4 Shall delete own postings.
- 2.3.5 Shall edit own postings.
- 2.3.6 Shall read through registered users' profiles.

2.4 Administrator

- 2.4.1 Shall delete any posting.
- 2.4.2 Shall block accounts.
- 2.4.3 Shall approve postings of Landlords
- 2.4.4 Shall disapprove postings of Landlords

Priority 2

2.1 Unregistered User

- 2.1.2 Shall be able to sort the listings.
- 2.1.4 Shall sort/filter listing by location, distance to SFSU, cost, utilities included.
- 2.1.8 Shall be able to search postings by keywords.

2.2 Registered User

- 2.2.6 Shall use forgot password to recover login.

Priority 3

2.1 Unregistered User

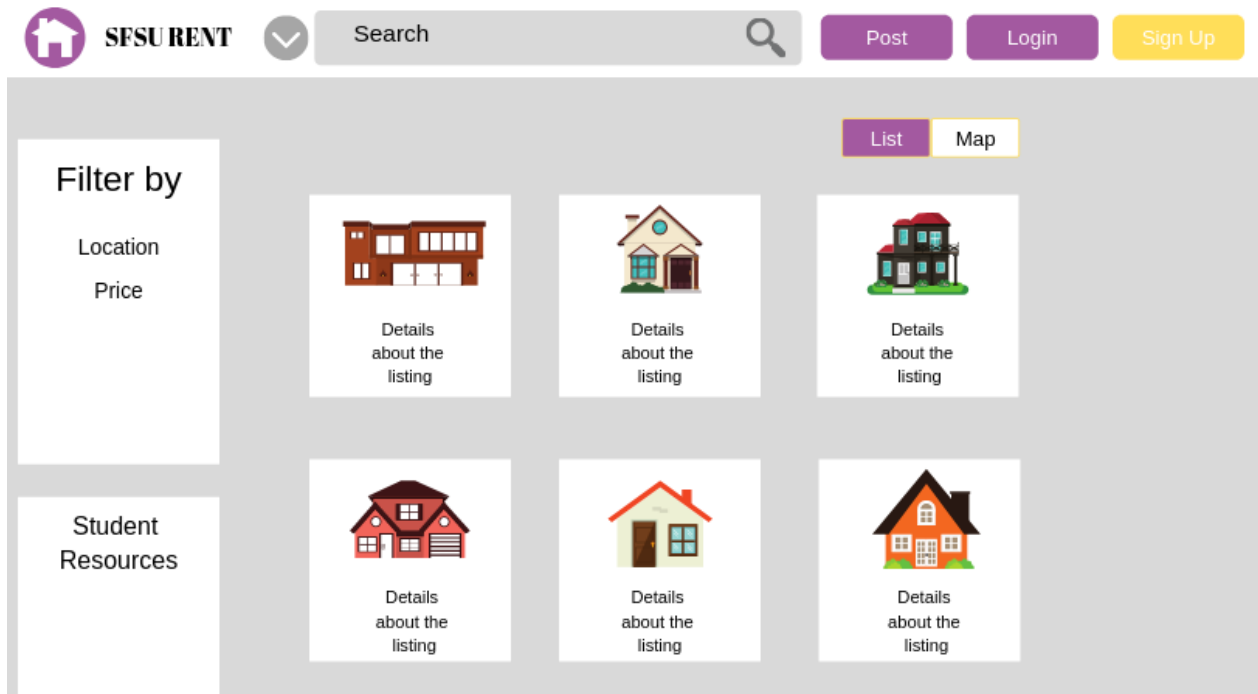
- 2.1.6 Shall check commute time of posting to SFSU via Google Maps (walk time, bike time, bus time, Bart time, driving time).

2.2 Registered User



- 2.2.2 Shall favorite postings so they can easily be found in the future.
- 2.2.7 Shall share the SFSURent website with friends via Social Media through referral links (not the actual listings).


3. UI Mockups and Storyboard

3.1 Home Page (logged out)



3.2 Login Page

 **SFSU RENT** 


Search 


Post Login **Sign Up**

Filter by


Location
Price


Student
Resources


Details about the listing


Details about the listing



Login



Details about the listing


Details about the listing

Post Map

3.3 Sign Up Page

 **SFSU RENT** 


Search 


Post Login **Sign Up**

Filter by

Location
Price


Student
Resources



Details about the listing


Details about the listing

name
email
password
re-enter password

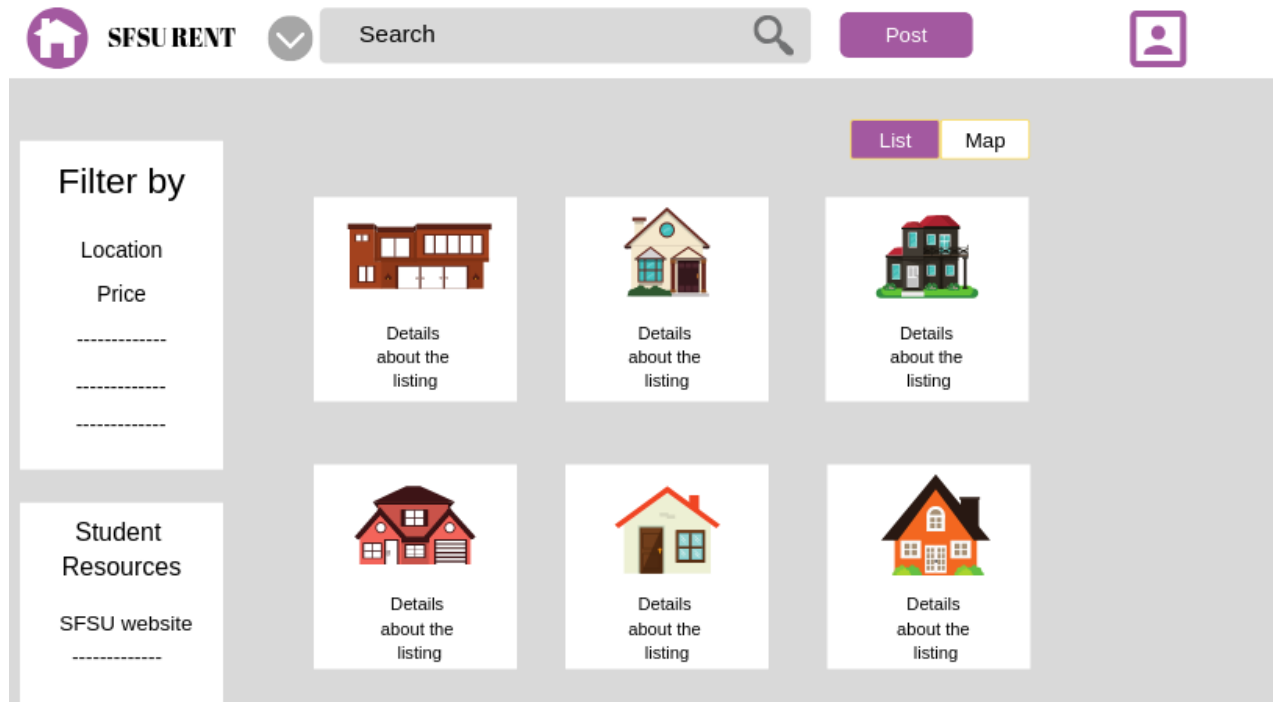
Sign Up


Details about the listing

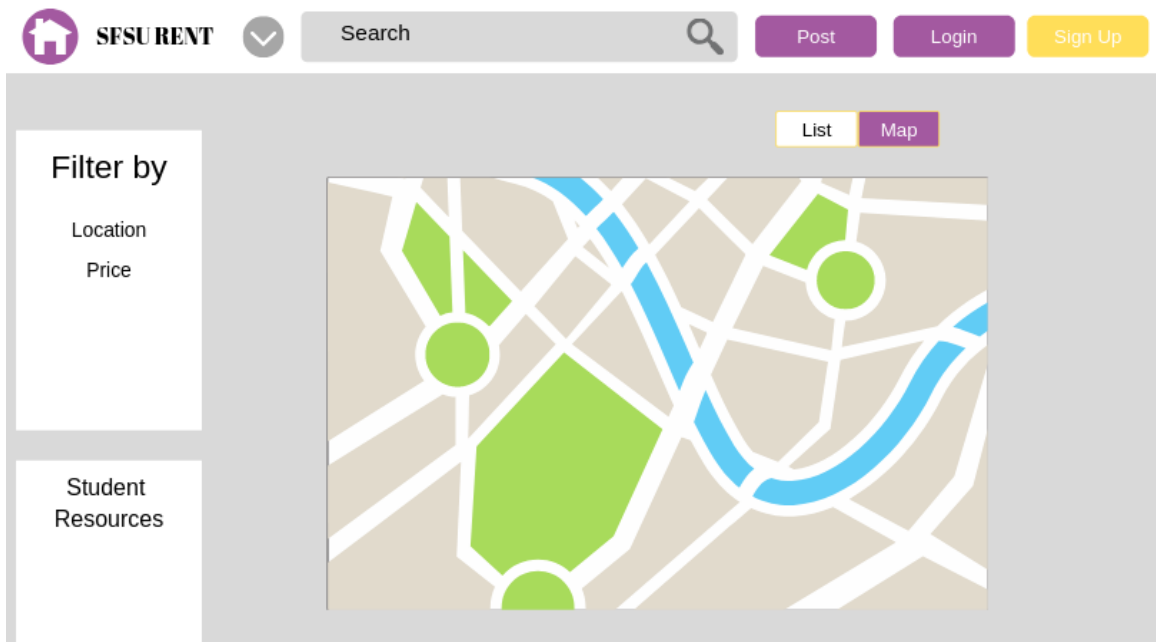

Details about the listing

Post Map





3.4 Home Page (logged in)



3.5 Map Page



3.6 Post Page

 **SFSURENT**   [Post](#) 

Create a Listing

Basic Information

Title





Address

Rent

Details


Size

Number of Beds

 **SFSURENT**   [Post](#) 





Number of Baths

Images




[Post](#)

3.7 Profile Page

 **SFSURENT**   

Your Profile

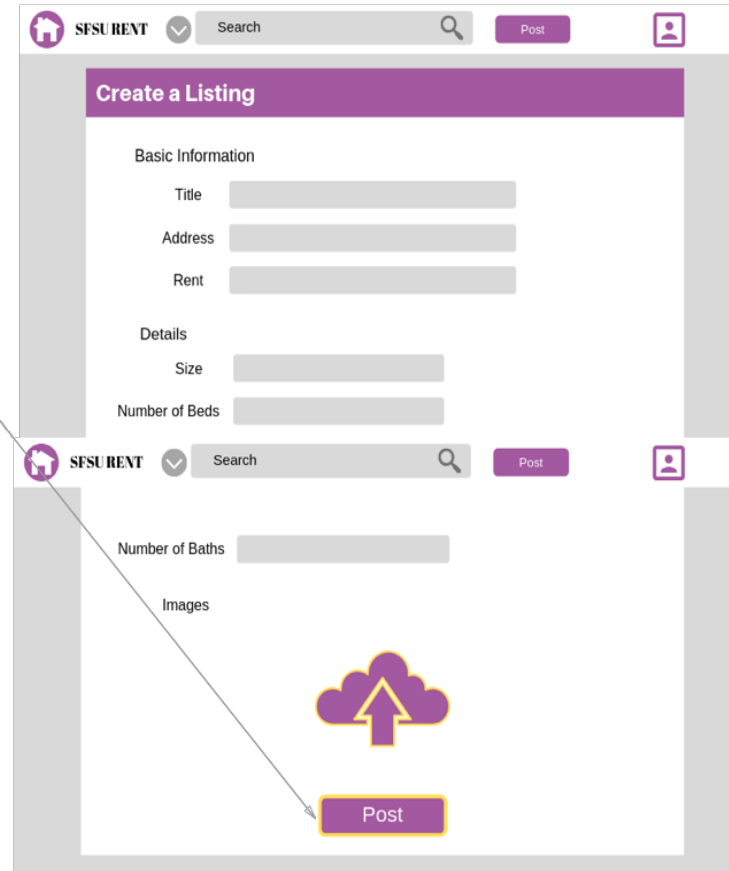
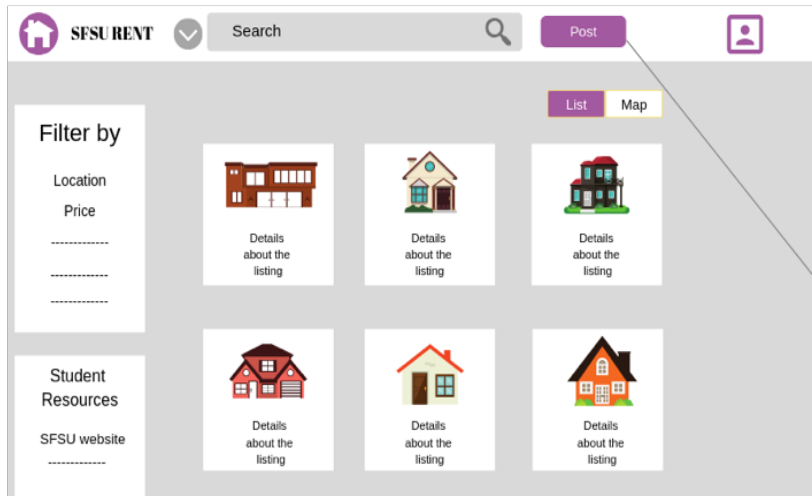


Name

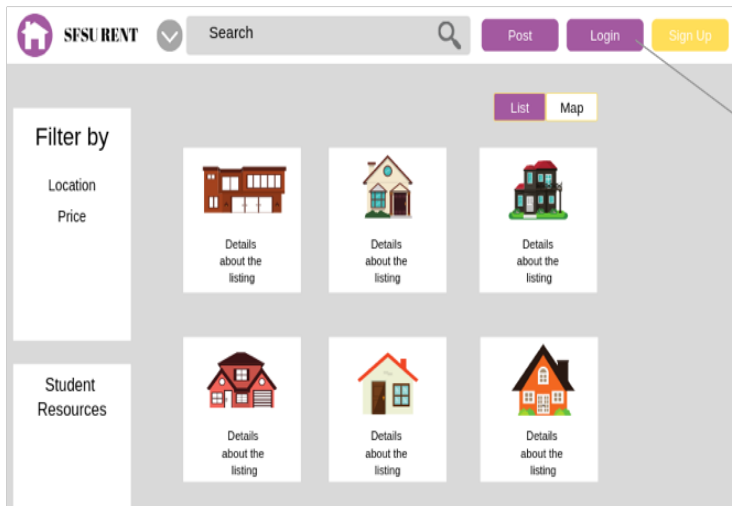
Role

4. Storyboards

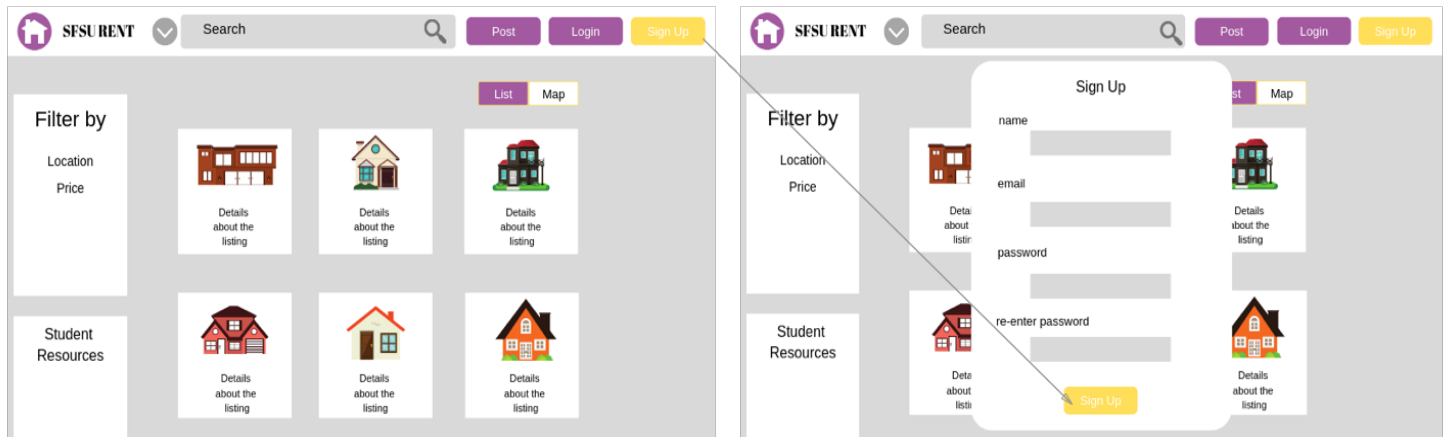
4.1 Posting



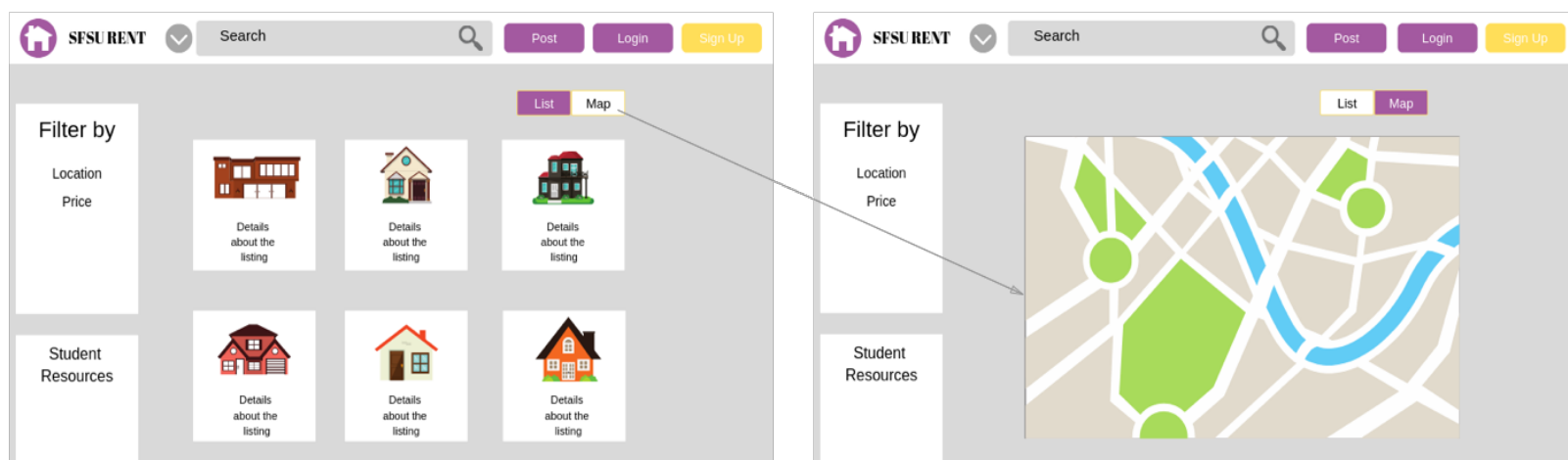
4.2 Logging



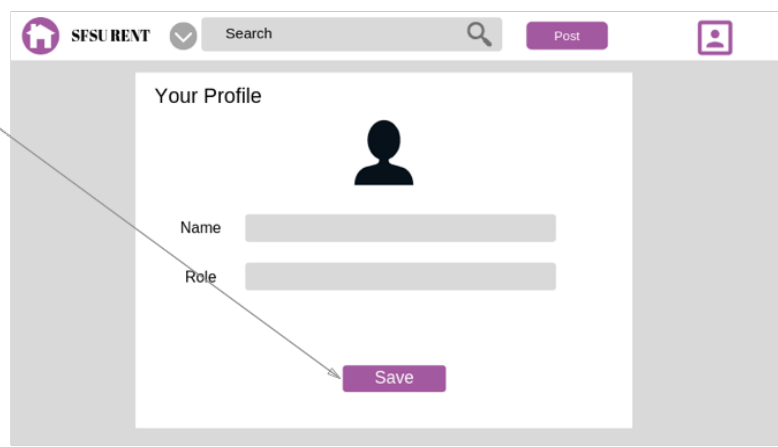
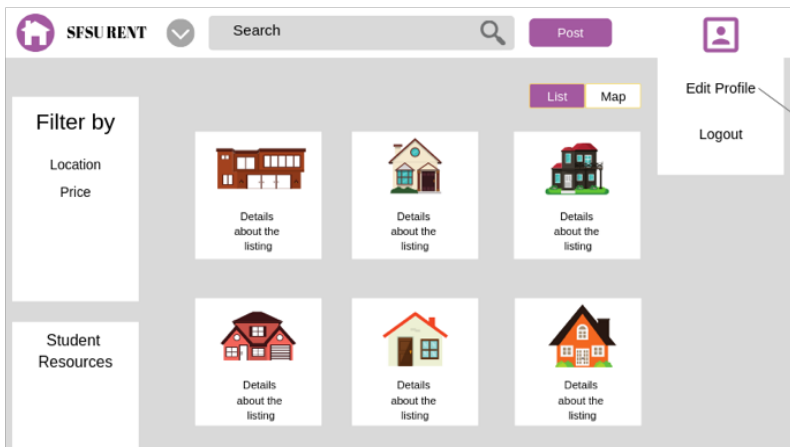
4.3 Signing Up



4.4 Opening the map that displays all the listings.



4.5 Editing Profile



4.High Level architecture, Database Organization

Table	Purpose	Entities
User	All registered user details according to their roles.	Id, name, email, password, phone, role, profile_image, thumb, last_login, created_at, updated_at
Password_Token	Registered users can reset their password.	Id, email, token, created_at
Message	Tenant and landlord can communicate with each other through messages.	Id, user_id, listing_id, body, created_at
Favorite	Tenants can store their favourites in a list.	Id, user_id, listing_id, created_at
Search_History	To store the search history of the user.	Id, user_id, params, created_at
Listing	All the listing's details posted by the authorized user.	Id,user_id, price, title, desc, num_bed, num_bath, size, address, score, lat, lng, zip-code, thumb, status, type_id, created_at, updated_at
Listing_Features	List of the features in the house.	listing_id, feature_id
Features	Details of features in the house.	Id, name
Listing_Type	Type of listing, example- apartment/ bungalow, room etc	Id, name
Listing_Image	Images of the listing i.e pictures of the house.	Id, listing_id, image, title, order
Listing_Commute	Details of the commuting modes to the university.	Id, listing_id, unit, type, value, created_at

- BLOBS will be used for storing images, and videos in this project.

Search

SFSURent will use a simple search algorithm to retrieve listings according to the user criteria. Technically, the search algorithm can be divided into two main types. The first type is the keyword search which is performed using the % operating in the MySQL database. For example, the user can input "sunny room" and the script will search for all the fields (that user can type in) and returns all the results that have the given keyword. The following query demonstrates this type of search:

```
SELECT *  
FROM listings  
WHERE title LIKE '%sunny%bedroom%'  
      OR desc LIKE '%sunny%bedroom%'
```

The second type of search is used for specific parameters such as the number of bedrooms and listing type. For the second type of the listing, the SQL query will be dynamically generated based on the user input. The following diagram shows a sample SQL query used for listing searching:

```
SELECT *  
FROM listings  
WHERE price < 1500  
      AND num_bed = 2  
      AND ( num_bath = 1  
            OR num_bath = 2 )  
      AND zipcode = 94112
```

To search in the pivot tables (such as the features), first the tables are joined and then the features are concatenated to allow for simple search using % operator.

```
SELECT listing.*, Group_concat(feature.name)  
FROM listings  
JOIN listing_feature  
ON listing_feature.listing_id = listing.id  
JOIN feature  
ON feature.id = listing_feature.feature_id  
GROUP BY listing.id  
HAVING feature.NAME LIKE '%furnished%'  
OR      feature.NAME LIKE '%bathtub%'
```

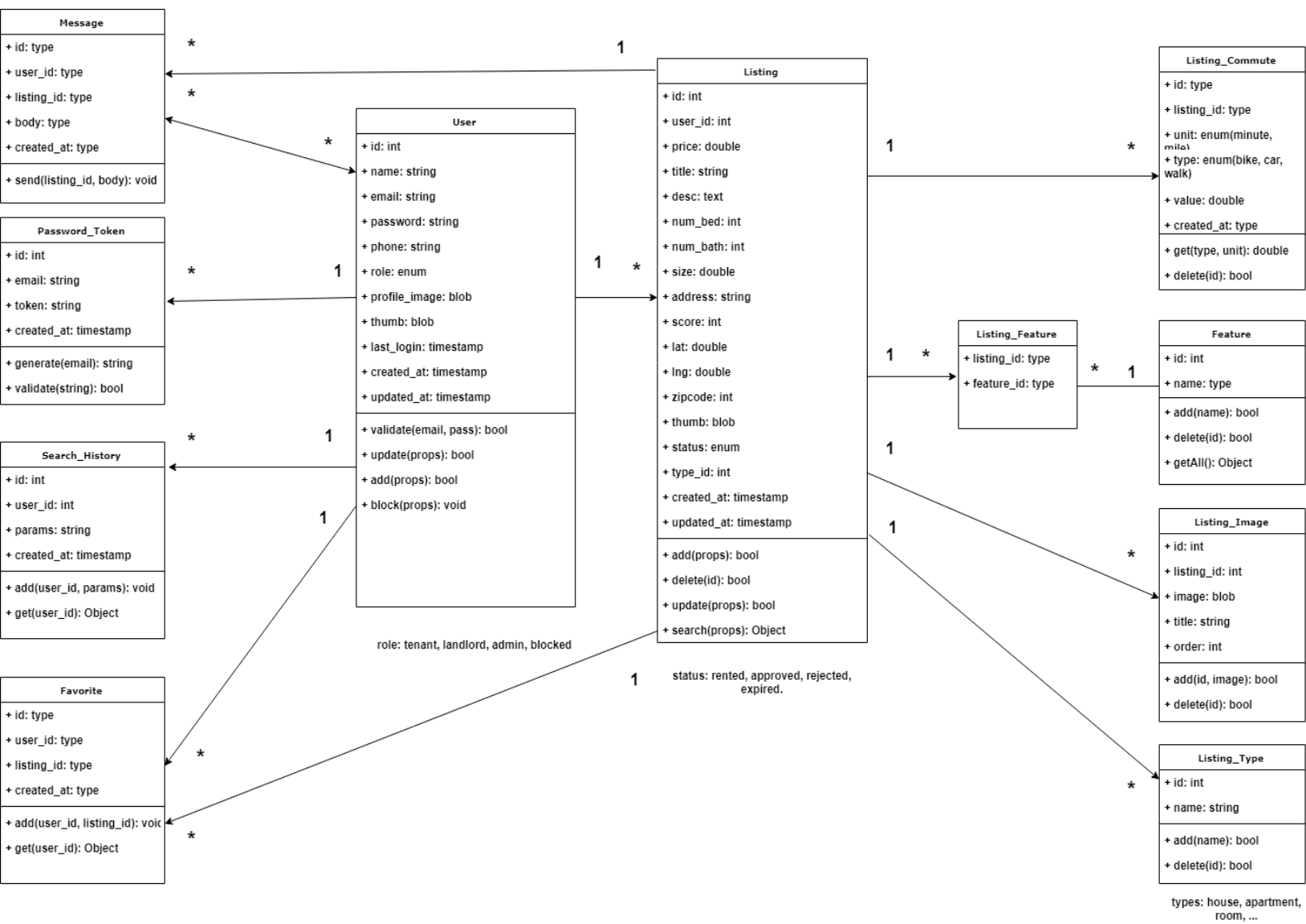
Moreover, it is necessary to note that all the above searching strategies will be combined to allow for more flexibility.

SFSURent has a ranking algorithm that will give determine which listing is the best for SFSU students by factoring in the: price, distance to school, laundry & dryer accessibility, Utilities included in cost, and shopping nearby. This ranking system will be a score of 1 to 5 so the users have a visual representation of which listings might be better options for them. Using USDA average food cost for college students, we incorporated this into our algorithm to cater to students working part time (20hrs per week) making minimum wage (\$15/hr). The average food cost for a college student is \$164.40 per month and the monthly income for a minimum wage student is \$995 before taxes. We come to the conclusion that \$830.60 is available for students to pay for housing without cutting into savings.

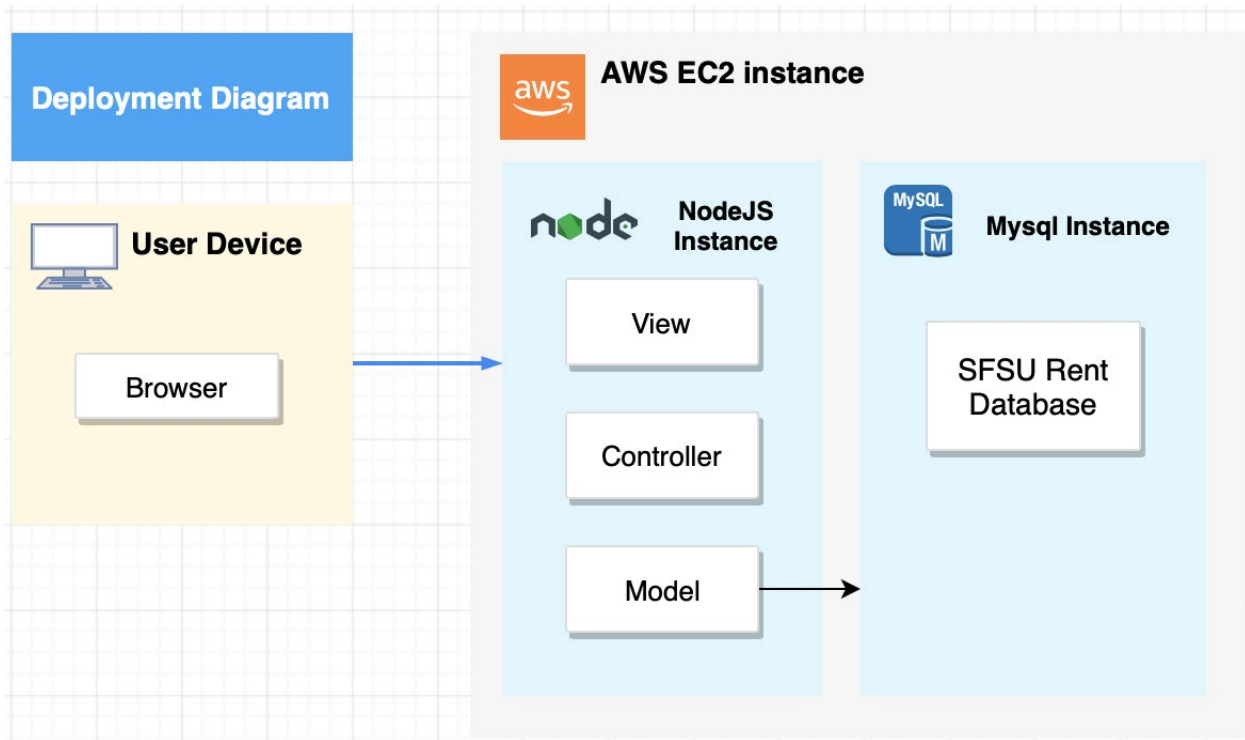
$$score = \frac{\$830.6}{Housing\ price} + if(SFSU\ distance < 5\ miles) + (laundry) + (Utilities\ incl) + (shopping\ close)$$

Each area in the algorithm caters to students. If the price for housing extends \$830.60 then housing will lose 1 point. If the distance to SFSU is greater than 5 miles the listing loses a star. Laundry, utilities, and shopping are boolean values that are worth 1 point each. So essentially the most ideal living situation for an SFSU student is a price less than \$830.6 is within 5 miles of school, has access to laundry, Utilities are included in price, and shopping is close by. Keeping in mind that the food cost is factored into the algorithm for best case scenario.

5.High Level UML Diagrams



The following diagram shows the deployment diagram for the project. We will be using an instance of AWS EC2 virtual private server to run the Node.js process and MySQL database. The node process can connect locally to the MySQL database since it is hosted on the same server. The user will navigate to the website on a supported browser and the node process will receive the requests.



6. Identify actual key risks for your project currently

- Skills risks
 - My team is evenly split with experienced web developers versus less experienced web developers and the members who have less experience with Node.js or any back end or front-end frameworks have been viewing tutorials to familiarize themselves with our stack of technologies
 - Working with GitHub in a large group has been a learning process for my team but we are all working through it together
- Schedule risks
 - Yes my team can complete the assignment given to us since we have a well distributed team that is communicating very well. The only downfall my team might have is the busy semesters each of us have.
- Technical risks
 - No technical unknowns currently
- Teamwork risks
 - No issues related to teamwork
- legal/content risks
 - No copyright risks that we are aware of

7. Project Management

My team is using Trello for task management. The cards/tasks are broken up between Front End, Back End, and Features. Each category has a: TODO, Doing, and Done area so the team is aware of progress. My entire team has joined this Trello board and is using it for our milestone 2 prototype.