Milestone 2

SW Engineering CSC648/848 Spring 2019

Team 101

Project Zephyr

Revisions
3/1/19 Milestone1
3/25/19 Milestone2

Date: Mar 25, 2019

Team Members: Jiannan Li, Tigist Ambaw, Bijan Mahdavi, Canbin Mei, Sydney Lou Morano, Brendan Ng, William Xie, Samuel Zaffanella

[**Email:** daiweirealrealreal@gmail.com]

1. Data Definitions V2:

Key Entities:

Guest - Guest can search for an apartment based on selected desired criteria. Certain content will be hidden for normal guest, such as the reply button, the exact address of the ***listing***, and potentially pictures will be blurred.

Student - ***Student*** has association relationship with User entity. It will have access to the same filter functionality as normal guest but will also be able to reply to ***listing***s and will not have address blocked or pictures blurred.

Landlord - The ***Landlord*** has association relationship with User entity. It will have access to the same unrestricted search as the ***normal guest*** but can also request to post a new ***listing*** that must be approved by the ***Admin***.

Admin - The ***Admin*** has the same search and post ability as the ***Landlord***. Admin has association relationship with landlord entity. When the ***Landlord*** requests to post a new ***listing***, the ***Admin*** can approve or deny it. The ***Admin*** can also delete ***listing***s.

Listing - Created by a ***Landlord***. The listing object contains the basic housing information. It contains pictures, a zip code, a price, an address, a description, house size, number of bedrooms, number of bathrooms, pets allow, parking, home type, and share with roommate. For

initialization data field: picture path, zip code, an address, description section, house size, number of bedrooms and bathrooms, pets allow, parking space, home type, and share with roommate.

Share with roommate allows landlord to rent out the share with roommate type house to the public. It differs from the home type. Home type will have the following types: house, apartment, condo.

Favorite- This data entity created by user entity. User can add their favorite house into a list.

Data Items:

For data items, our group are using Javascript convention naming.

User:

User data entity will have the following data items: email, password, firstname, lastname, authentication. And all data types should be string.

Student:

Student data entity will have the following data items: studentId and studentPhone. All data types should be string.

Admin:

Admin data entity will have the following data item: adminId, which is a string data type, as a primary id to serve the purpose as Admin user.

Landlord:

Landlord data entity will have the following data items: landlordId, landlordAddress, landlordPhone. All data types should be in string. And landlordAddress should store the address of landlord instead of the house that the landlord is listing.

Listing:

Listing data entity will have the following data items: zipCode, price, address, description, bathroomCount, bedroomCount, petsAllow, parking, houseSize, homeType, shareWithRoommate.

bathroomCount, bedroomCount, homeType, and zipCode shall be integer data type. And for homeType, each number means different type of homes.

price, houseSize shall be double data type.

petsAllow, shareWithRoommate, parking shall be Boolean data type.
address, description shall be string data type.
And Listing will have share option can be assign from landlord.

Favorite:
Favorite data entity will have the following data items: favoriteId. And favoriteId should be Integer datatype.

Filter:
Filter is not a data entity, it shall not appear on the xml. Filter behavior like search functionality for the project.
Landlord need to modify shareWithRoommate while listing the house. If they click shareWithRoommate in the option. It means the house will be share with roommate/roommates. It has the following data items that allow user to use the filter to search the target value in the database. The filter will include price range, share/entire house. If user click shareWithRoommate, then the filter result will return all shareWithRoommate type house from listing data entity. Otherwise the filter will show the entire house type from Listing object.

2. Functional Requirements V2:

Functional requirement is being implemented according to the xml diagram.

Priority 1:

Homepage:

1. Homepage will allow user to register/login and browse apartment information.
2. Homepage shall allow user to use search functionality.

Guest:

1. Guests shall be able to view everything on the homepage.
2. Guests shall be able to sign up with an email and password. And the email should be verified as SFSU student.

3. Guests shall be able to search for apartment information.
4. Guests shall be able to check the apartment information as well as housing description. (search bar or filter).
5. Guests shall be prompted to sign up in order to contact with landlord.

Registered User:

User:

1. Registered users shall have the same functionality as registered users.
2. Registered users shall be able to login with their email and password.
    1. Student users will be verified with valid SFSU email address.
    2. And system will send confirmation email to finish the registration.

3. Registered users shall be able to contact landlord or acquire landlord information.

V2. User will have the following function to be implemented during the project

q gerUsername()/setUsername() which get/set the username for the user.

q getPassword()/setPassword() which get/set the password for the user.

q getFirstName()/setFirstName() which get/set the firstname and lastname for the user.

q getEmail()/setEmail() which get/set the email address for the user. And the email should have authentication system to verify the student's email belongs to @mail.sfsu.edu or @sfsu.edu.

Student: Student extends from User and have the similar functionality like user.

q getStudentID()/setStudentID() which get/set the student id for the student user. Student user can view the landlord email and send email to the landlord.

Landlord:

1. Landlord users shall be able to post/remove/modify their posts.
2. Apartment information can contain photos and description.
3. Landlord users shall be able to receive message/email from students.

q getLandlordID()/setLandlordID() which get/set the landlord id for the landlord user. This is the identifier to distinguish the landlord user from normal student user.

q SubmitPost() allow landlord user to post the housing information.

q ModifyPost() allow landlord user to modify the post.

q DeletePost() allow landlord user to delete the post which he has posted.

Admin:

1. Admin user shall have the same functionality as registered users.
2. Admin user shall review the content posted by landlord and approve/disapprove the post.
3. Admin user shall be able to remove the post.
4. Admin user shall notify landlords the post has been approved/disapproved.

q GetAdminID()/setAdminID is the identifier that distinguish Admin user from normal user.

q DeleteListing() allow Admin user to delete the listing that posted from landlord user.

q ApproveLising allow Admin user to approve the listing from landlord user.

q DenyListing() allow Admin user to deny the listing from the landlord user

Listing:

q getPicture()/setPicture() will allow landlord user to attach the picture when listing the house or apartment. This should get the picture path. And the picture should be stored as path within the database.

q getPrice()/setPrice() will allow landlord user to set the price when listing the house.

q setDescription()/getDescription() will allow landlord user to descipe the house or apartment they are listing for.

q setRoomCount()/getRoomCount() will allow landlord to setup the available room for the entire house or apartment.

q getListingID()/setListingID() is the identifier that keep in track of the listing id number.

Filter functionality:

1. Filter functionality shall allow user to pick price range. Such as minimum and maximum.
2. Filter functionality shall allow user to pick the size of the house, number of bathroom and bedroom, home type as well as allowing pets, parking, home type.
3. Filter shall allow user to pick share with roommate or individual house from the listing.

Priority 2:

1. Registered users shall have their profile.
2. Registered users can star the apartment information.

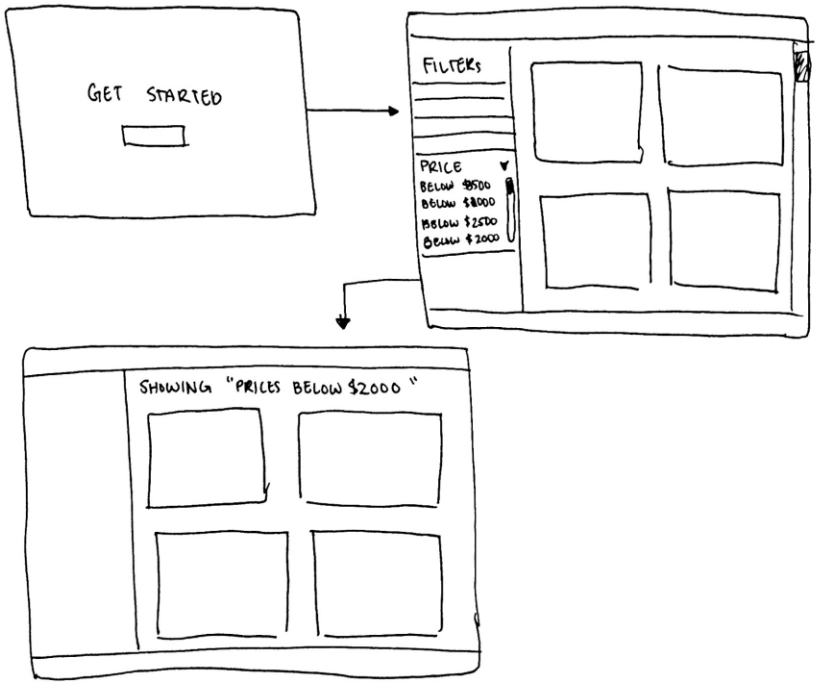
3. Admin user can have a pipeline to approve/disapprove the post.
4. Live chat system between landlords and students.

Priority 3:

1. Recommendation system based on students' profile.
2. More specific options under search functionality.
3. Registered users can review apartments.

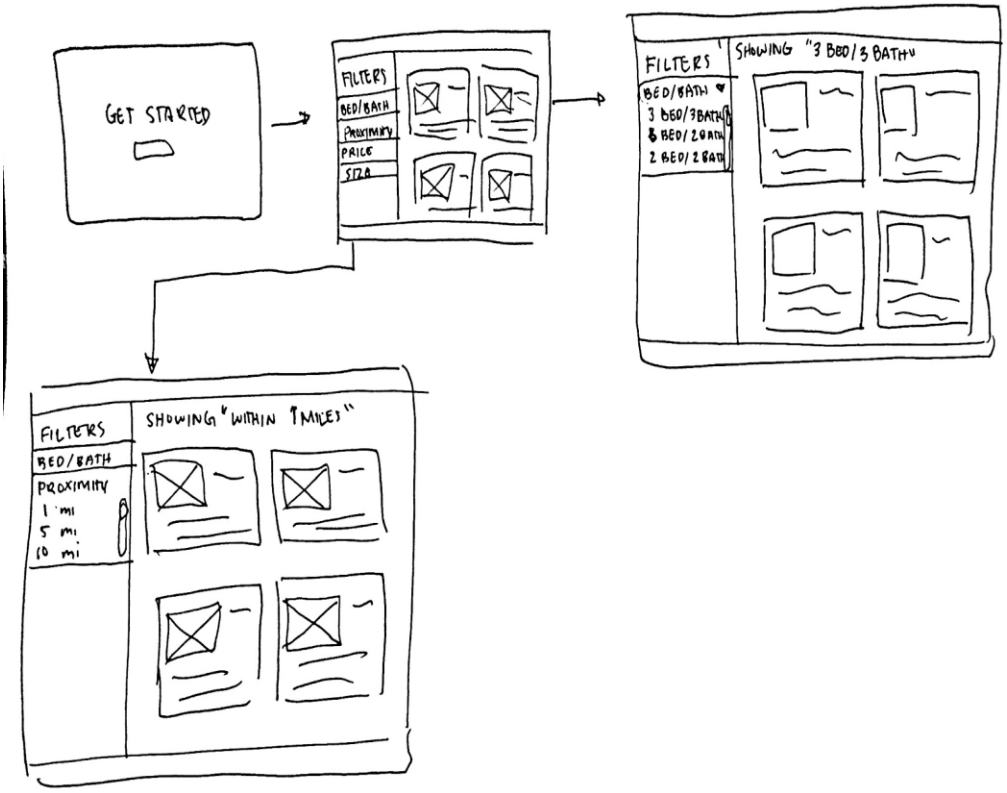
3. UI Mockups and Storyboards (high level only):

Storyboards:



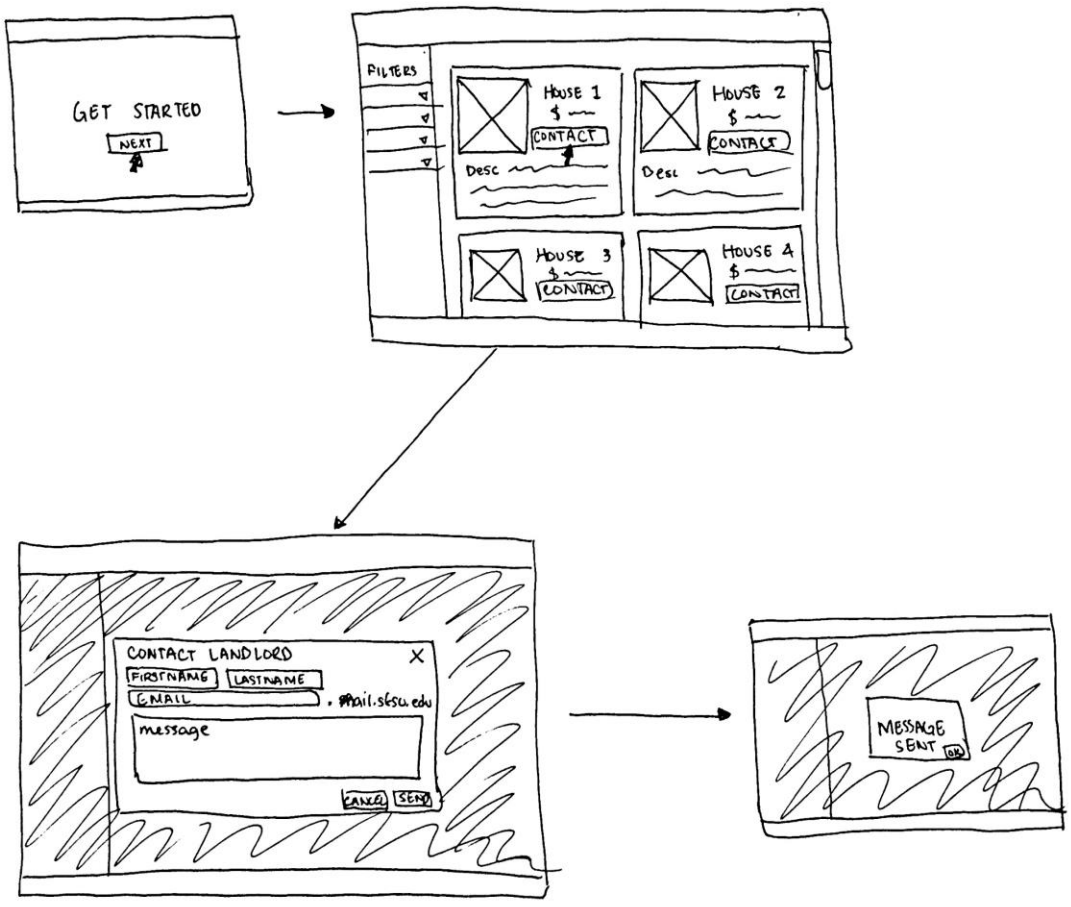
USER STORY: FERRARI

User story: Ferrari shows usage of price search within website.



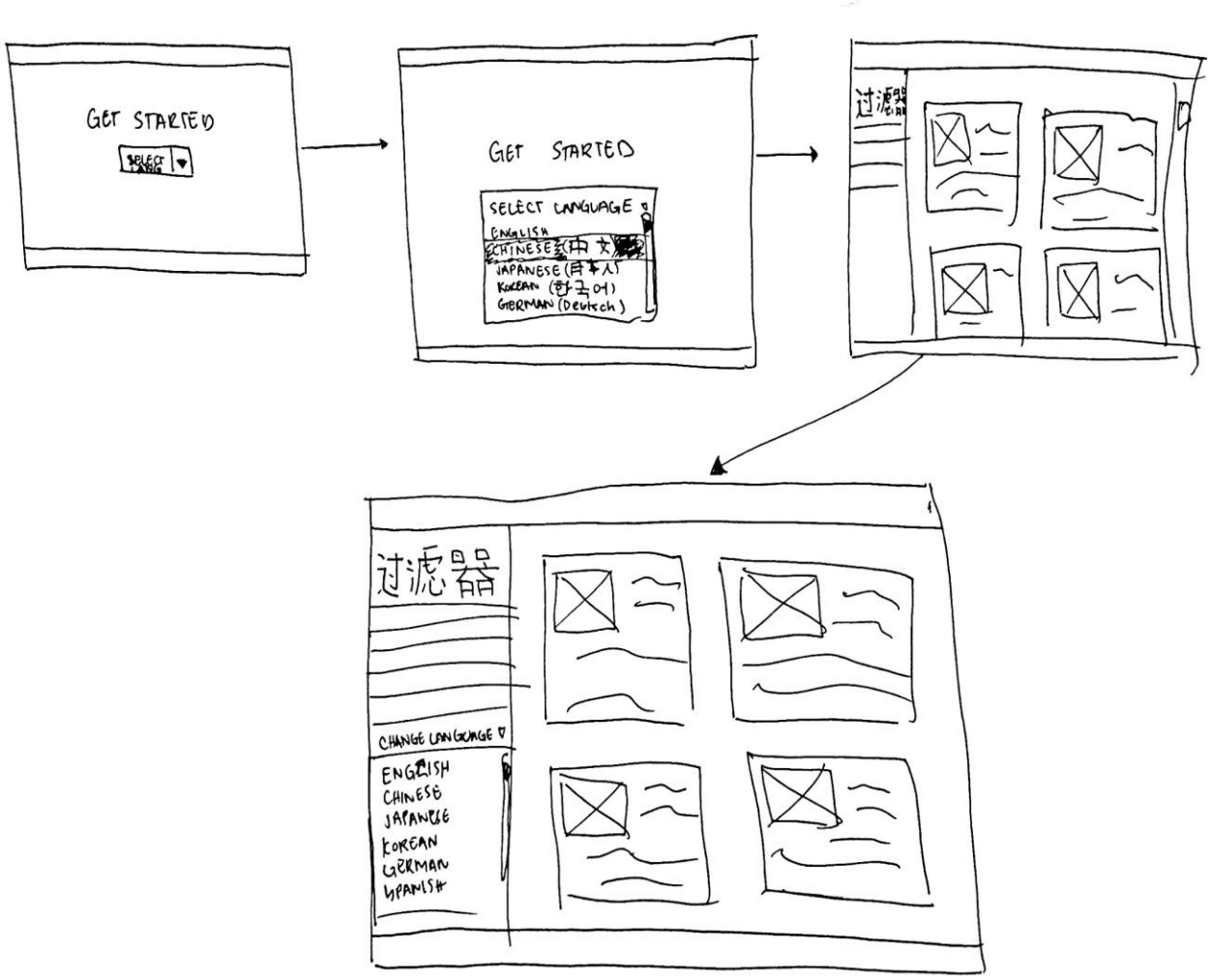
USER STORY : JENNIFER LI & EMILY

User story: Jennifer Li and Emily show usage of bed/bath and proximity features.



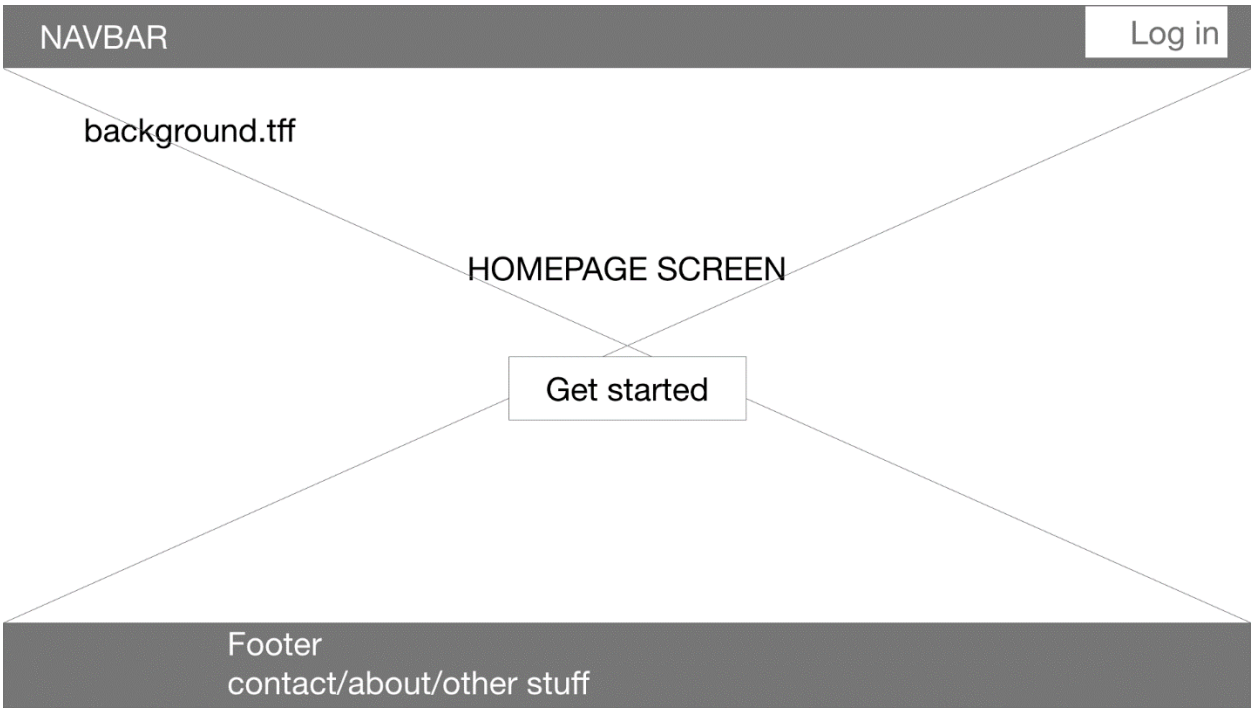
USER STORY : JENNIFER LI

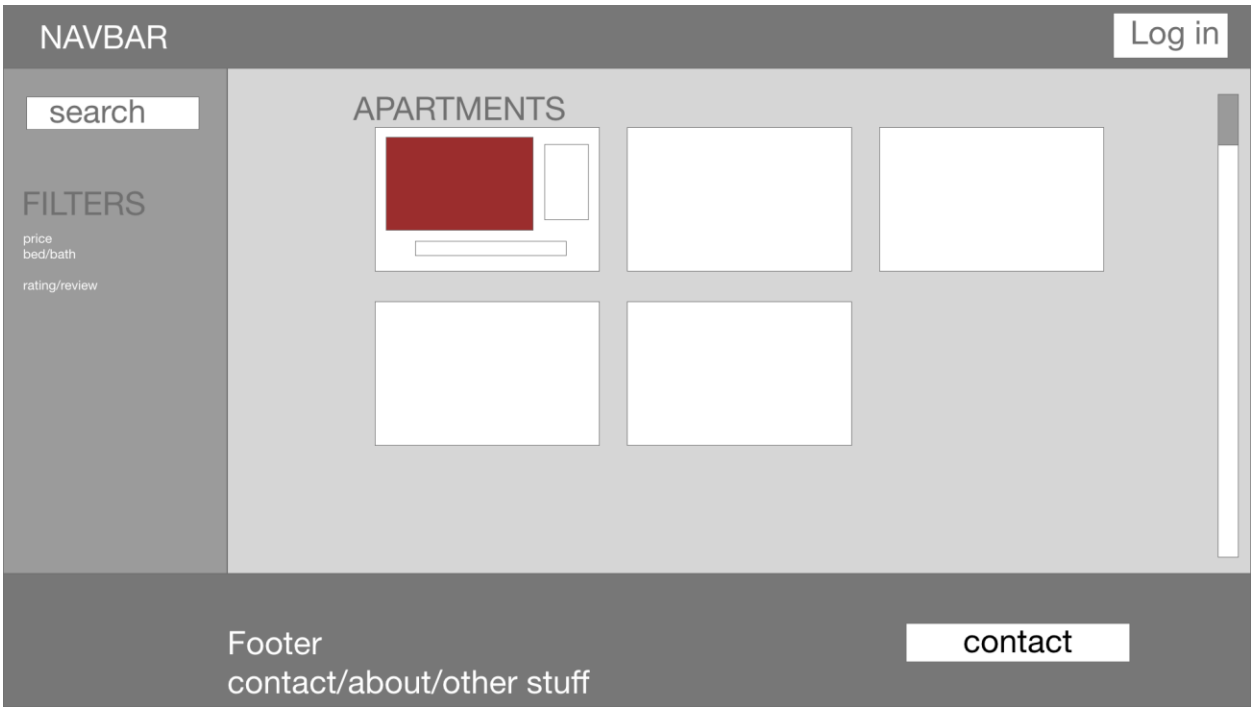
User Story: Jennifer Li shows “Contact Landlord” Feature.

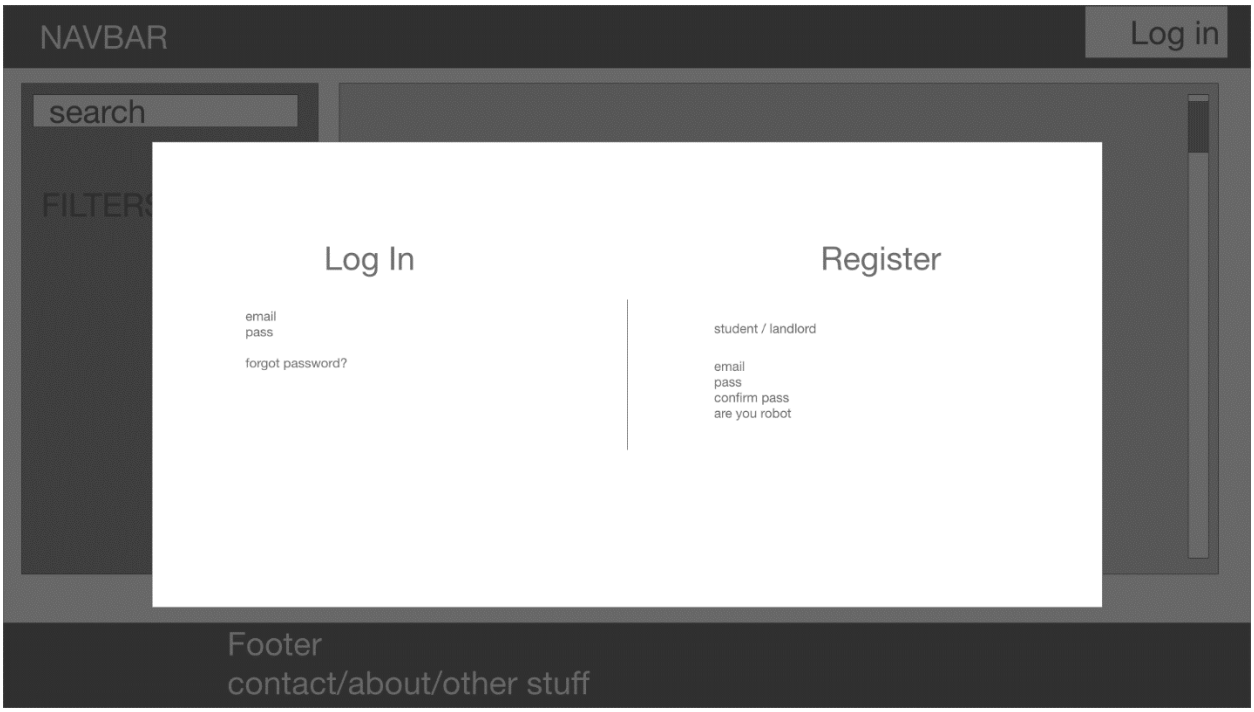


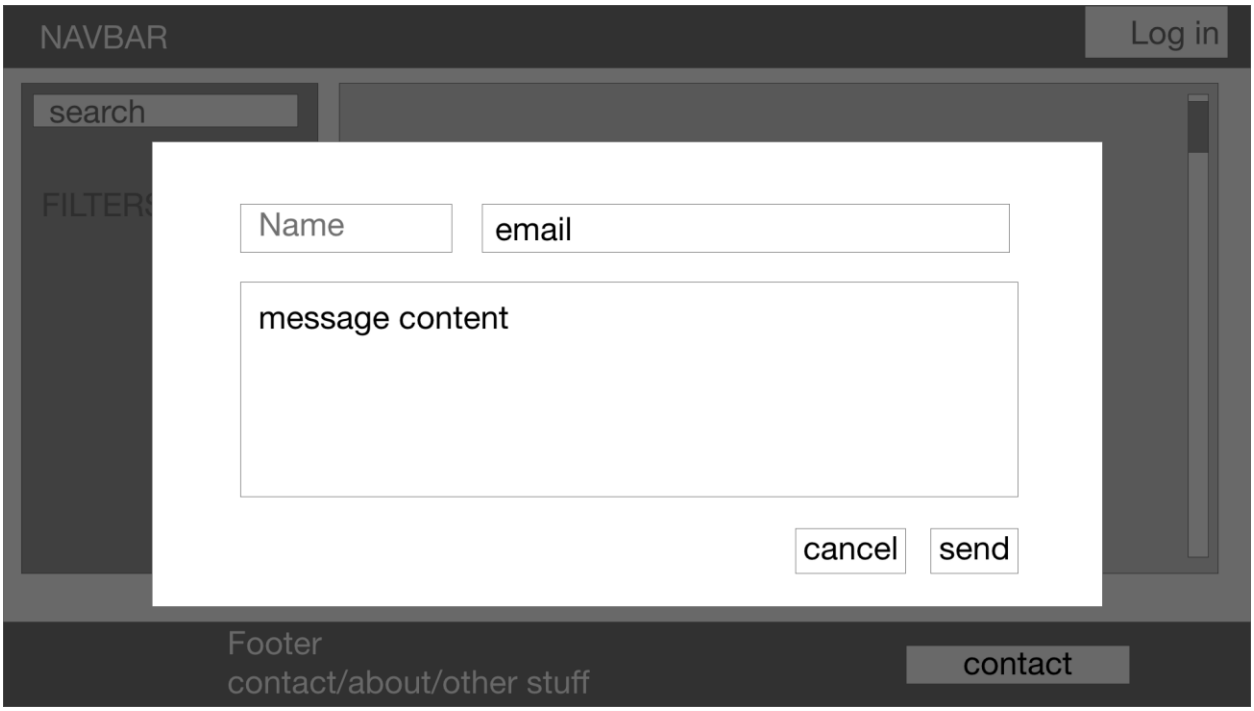
User Story: Jason shows usage of the Language Translation tool implemented throughout the website.

GUI Design (Using Adobe XD):




NAVBAR
Log in
search
APARTMENTS
FILTERS
price
bed/bath
rating/review
Footer
contact/about/other stuff
contact


NAVBAR
Log in
search
FILTERS
Log In
email
pass
forgot password?
Register
student / landlord
email
pass
confirm pass
are you robot
Footer
contact/about/other stuff



4. High level Architecture, Database Organization:

Server Host: AWS Server t2.micro 1

Operating System: 64Bit Amazon Linux/4.8.0

Database: MySQL 5.6.41

Web Server: Nginx 1.14.2

Server-Side Language: Node.js 10.15.1

Additional Technologies: Frontend Framework: React 16.8.0

Backend Framework: Express 4.16.4

IDE: Visual Studio Code/ WebStorm

Web Analytics: Google Analytics

Database organization:

1. MVC modeling control. DB should be model part. Team are using Sequelize API to connect NodeJS with MySQL database.
2. Schema
3. Database migration

Sequelize allows you to map your relational database to objects. Those objects have methods and properties that enable you to avoid writing SQL statements.

Schema:

Database: CSC648

Table: User

Columns: username (String), password(String),firstName(String), lastName(String),email(String)

Table: Student

Column: studentId (Int)

Table: Admin

Column: adminId(String)

Table: Listings

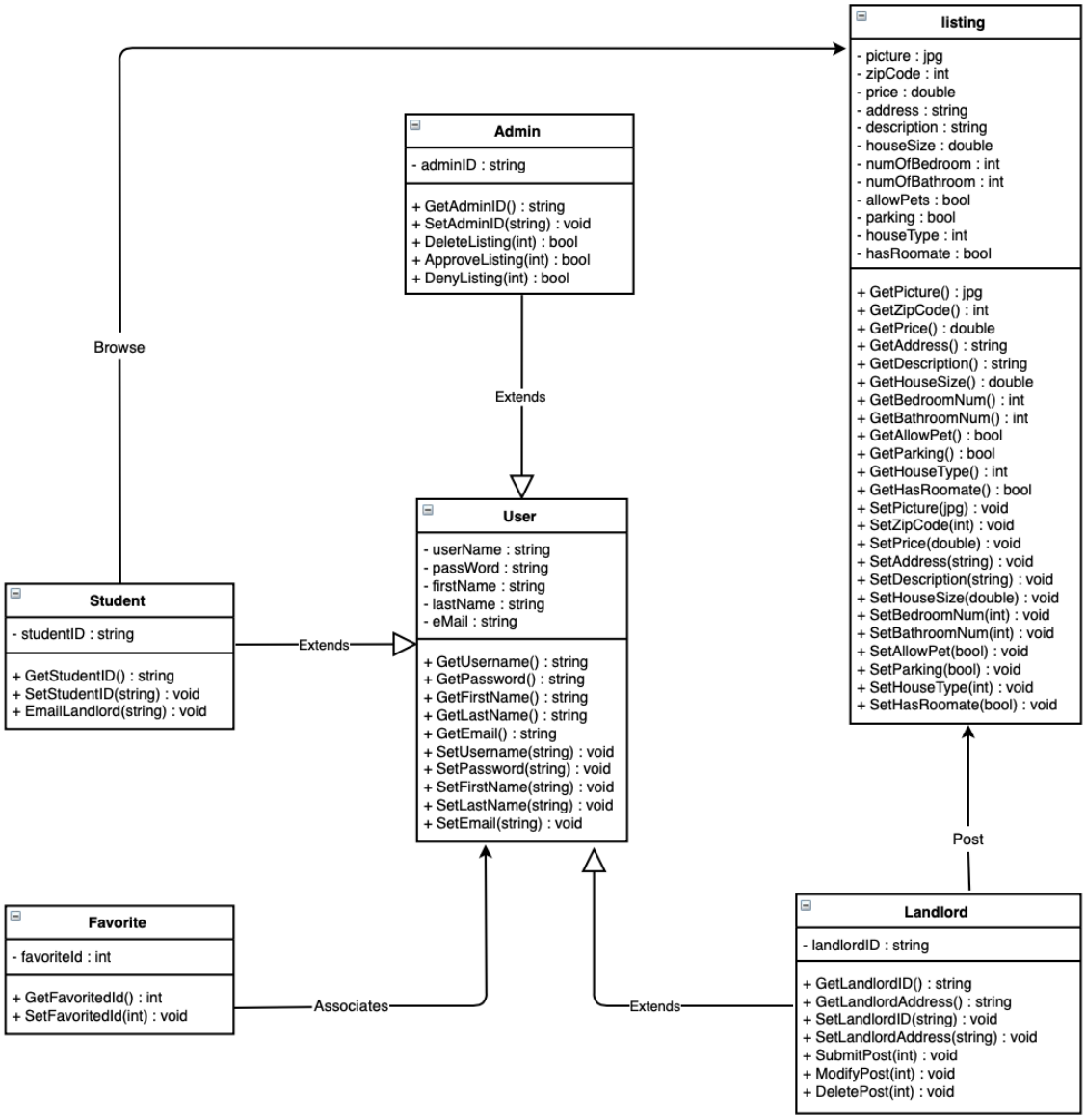
Columns: zipCode(Int), price(double),address(String),description(String),houseSize(Double),

numOfBedroom(Int),numOfBathroom(Int),allowPets(Boolean),parking(Boolean),houseType(Int), hasRoomate(Boolean)

Table: Landlord

Column: landlordId(Int)

5. High Level UML Diagrams:



The high-level UML diagram is still in prototype stage. Our team will keep adding entities into the diagram later for this project.

6. Identify actual key risks for your project at this time:

Skills risks:

The front-end team is still getting more familiar with the react javascript library. Early on prototypes of the website will be heavily using html and css and less so of javascript. A similar problem is applied to the back-end team with the framework/javascript using node.js. We also need to deal with database migration and how we organize our data.

Schedule risks:

We can without a doubt make a fully functioning apartment rental website. However, we need to make sure each person's code is compatible with each other's in the front end. Furthermore, each group member has vastly different schedules, so it is somewhat difficult to create a meetup time outside of class. Lastly, the front end needs to connect with the back-end more often. We are constantly making new design decisions so there will always be new resources.

Technical risks:

Currently there are some compatibility issues between javascript files created by various front-end members. This shouldn't be a long-term issue, rather just something that we will need to make tweaks to on one or more files.

Teamwork risks:

Each team member is effectively contributing their part by fulfilling their role in this group project. Unfortunately, as mentioned, there has been many scheduling conflicts so it's somewhat difficult for members to meet up and review each other's progress. In other words, each member

is working separately from one another and thus making progress on this project in their own ways.

Legal/content risks (can you obtain content/SW you need legally with proper licensing, copyright):

As of now, there are not any licensing or copyright issues with our project. There is not any software that requires proper legal licensing that we do not already have. This may come up if we decide to use specific creations extracted straight from the internet (e.g. Photos, videos, audio files, etc.).

7. Project management:

After finishing up milestone one we had formed two teams, the frontend and backend team, which aided in managing task delegation for milestone two. In milestone two there are seven main requirements we needed to fulfill, we went through each requirement to see whether it is more of a frontend/backend question or product owner question.

After splitting them up it was the responsibility of the team members to decide who is going to fulfill which requirement and using which tools. An example of this is Sydney, our frontend lead, is responsible for the UI mockups and storyboards. It wouldn't make sense to have someone in the backend group to make the UI mockups as it is a frontend question. Sydney was able to accomplish the UI mockups by using Adobe XD. Another tool we use for this project is Visual Studio Code; this editor makes editing code a lot simpler by displaying all the files in the project on the sidebar.

We are planning to stick with the system we have now for future milestones as it quickly sorts tasks to which team they belong to, however there are a few changes that we could implement moving forward. Once we split each part of milestone two, it made it simpler to decide who is responsible for each requirement. Each milestone is essentially a task to be accomplished with a sequential order that helps breakdown the larger project. But each milestone has within it, subtasks that need to be met. So, an improvement moving forward would be to

identify those subtasks and accomplish those in a timely manner. Another management change that would improve the overall efficiency and productivity of our group is periodically checking in with each other to guarantee that everyone is on the same page and it would allow opportunity to support one another if an issue occurs.