SW Engineering CSC648/848 Section 2 Fall 2016

SFSU Rental Hub

Team Number 16

Members:
Darin Evanow (devanow@mail.sfsu.edu)
Abhilash Shrivastava
Felix Lee
Steven Lum

Milestone 2

October 27th, 2016

Revision Number	Revision Date
Initial Version	October 27th, 2016

I. Use Cases

A. Looking to Rent

John has recently graduated high school in LA. He is planning to attend SFSU in the Spring 2017 semester. All of the dormitories are already filled up, and he does not have a friend or relative in San Francisco to live with. He is wondering if there is anyone living in the area with a spare room that he can rent. John recalls the name of this website SFSU Rental Hub he saw on Facebook the other day, and checks it out. He finds he is able to view many rental listings all at once, and that the website is specifically catered to SFSU students in need of a place.

B. Looking to Post

Mike and Mary live in the Sunset District of San Francisco. They have a daughter, age 18, who is about to move to the East Coast tomorrow for college. The parents will have an empty room for the next 6 months. They could use a bit of extra money and know that SFSU is an impacted school, so they go on Google to search for "San Francisco room rentals" to try and rent out their daughter's room. They scroll past Craigslist after having previous bad experiences with it, and decide to try SFSU Rental Hub. Within the hour, Mike and Mary have posted their renting listing with relevant information about the room.

C. Administrator

Carl has joined SFSU Rental Hub as an administrator, to help moderate the listings that are being posted. One day when browsing he notices that someone has put some obscene material in one of the photo sections for a listing. Since Carl is logged into his administrator account, he quickly deletes the listing, simultaneously removing the content so that no one else can see, and sending an email to the rental owner to inform them of the action taken.

II. Data Definitions

1. Rental Space: The type of unit that will be rented out (apartment, in-law, room).
2. Rental Listing: Listing created by a registered user with all rental unit information (price, dimensions, address, pictures, etc.).
3. Registered Student: A user who has registered themselves on the website, so that they can rent rental listings.
4. Registered Landlord: A user who has registered themselves on the website, so they they can post rental listings.
5. Guest User: A user who has not registered themselves on the website. They are still allowed to browse rental listings, but cannot post or rent rental listings.
6. Administrator: A power user who has the ability to remove rental listings that go violate the Terms of Service, as well as see additional information about registered users.

III.Initial List of Functional Specs

User Functional Specs

Spec	Priority
A. All users shall be able to view rental listings.	1
B. All users shall be able to use filters when browsing rental listings.	1
C. Guest users shall be able to register an account.	1
D. Guest users shall not be able to post rental listings.	1
E. Guest users shall not be able to contact landlords.	1
F. Registered landlords shall be able to post rental listings.	1
G. Registered students shall be able contact registered landlords.	2
H. Registered users shall be able to change their account information.	2
I. Registered users shall be able to delete their account.	2

Administrator Functional Specs

Spec	Priority
J. Administrators shall be able to remove rental listings.	1
K. Administrators shall be able to view basic account information of registered users, solely for legal purposes.	1

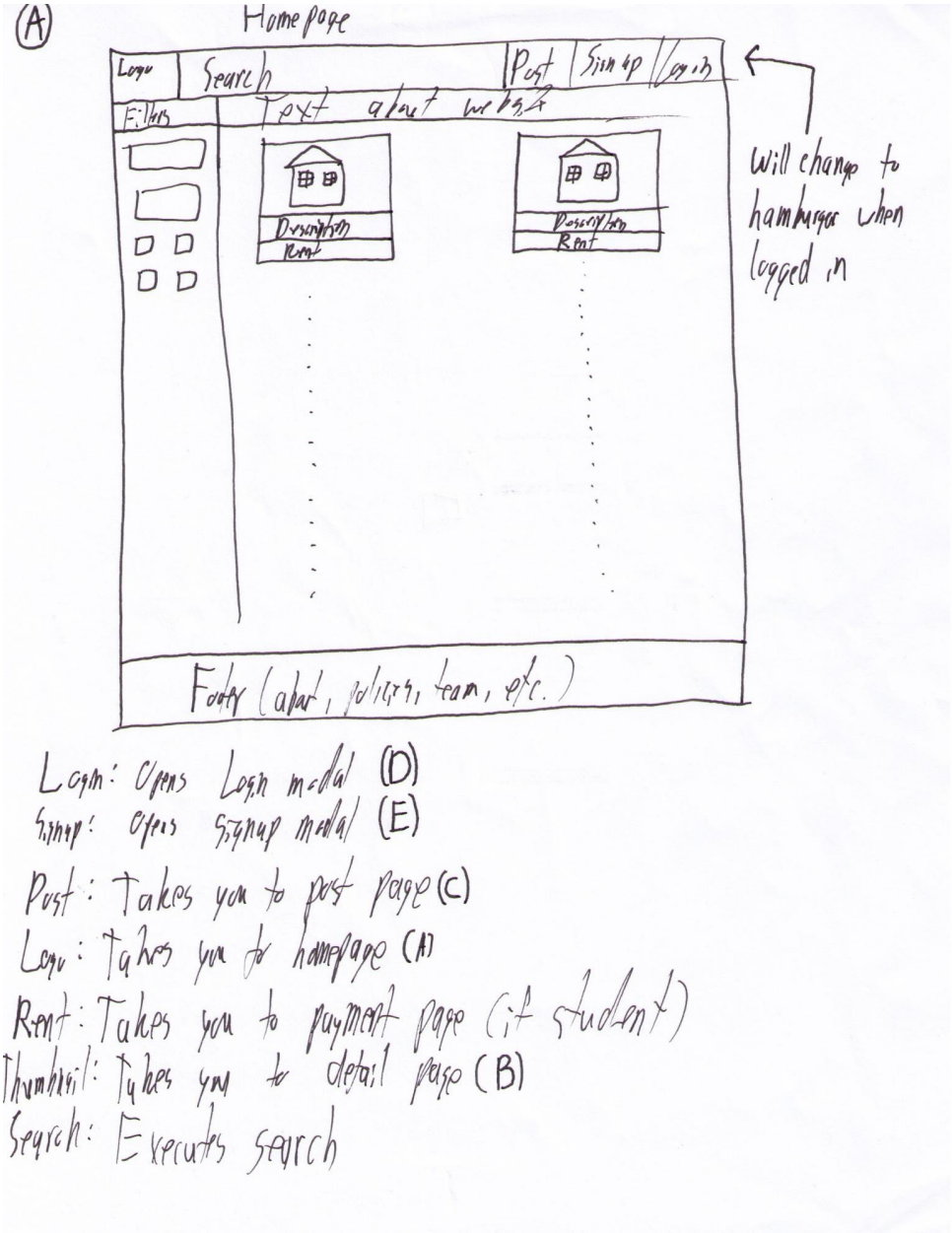
Features Functional Specs

Spec	Priority
L. The relative location of a rental listing shall be shown via Google Maps.	2

V. List of Non-Functional Specs

1. Application shall be developed using class provided LAMP stack.
2. Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks shall be explicitly approved by Marc Sosnick on a case by case basis.
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class.
4. Application shall be optimized for standard desktop/laptop browsers, and shall render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. It shall degrade nicely for different sized windows using class approved programming technology and frameworks so it can be adequately rendered on mobile devices.
5. Data shall be stored in the MySQL database on the class server in the team's account.
6. Application shall be served from the team's account.
7. No more than 50 concurrent users shall be accessing the application at any time.
8. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
9. The language used shall be English.
10. Application shall be very easy to use and intuitive. No prior training shall be required to use the website.
11. Google analytics shall be added for major site functions.
12. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services.
13. Pay functionality (how to pay for goods and services) shall be simulated with proper UI, no backend.
14. Site security: basic best practices shall be applied (as covered in the class).
15. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development, and only the tools and practices approved by instructors.
16. The website shall prominently display the following text on all pages "SFSU/FAU/Fulda Software Engineering Project, Fall 2016. For Demonstration Only". (Important so as to not confuse this with a real application).

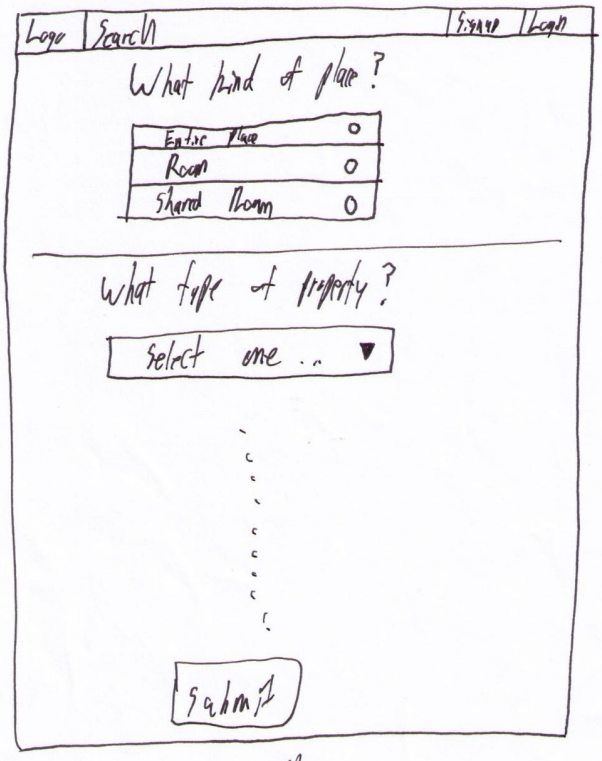
V. Mockups and Storyboards



Ⓑ Details Page

1. Rent: Takes user to rental page (if logged in)
2. Email: Gives user the email of host to ask questions (if logged in)
3. Carousel: Cycles through different images the host has uploaded

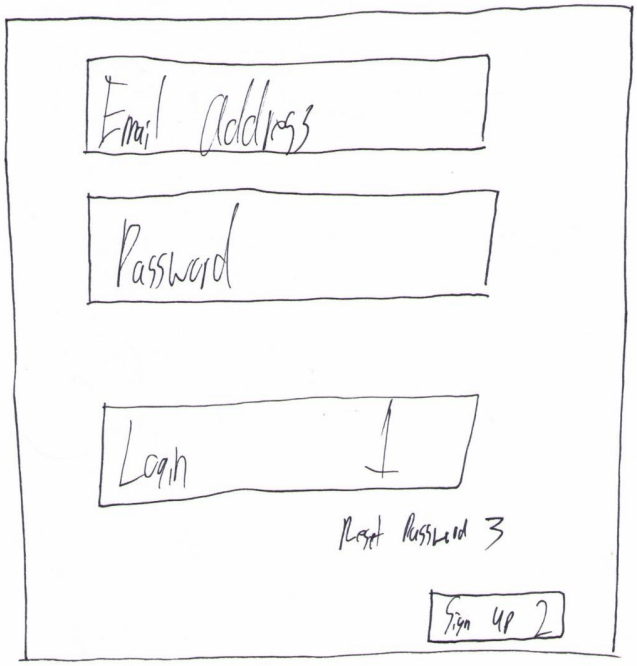
(c) Posting Process



Hosting process is all one page

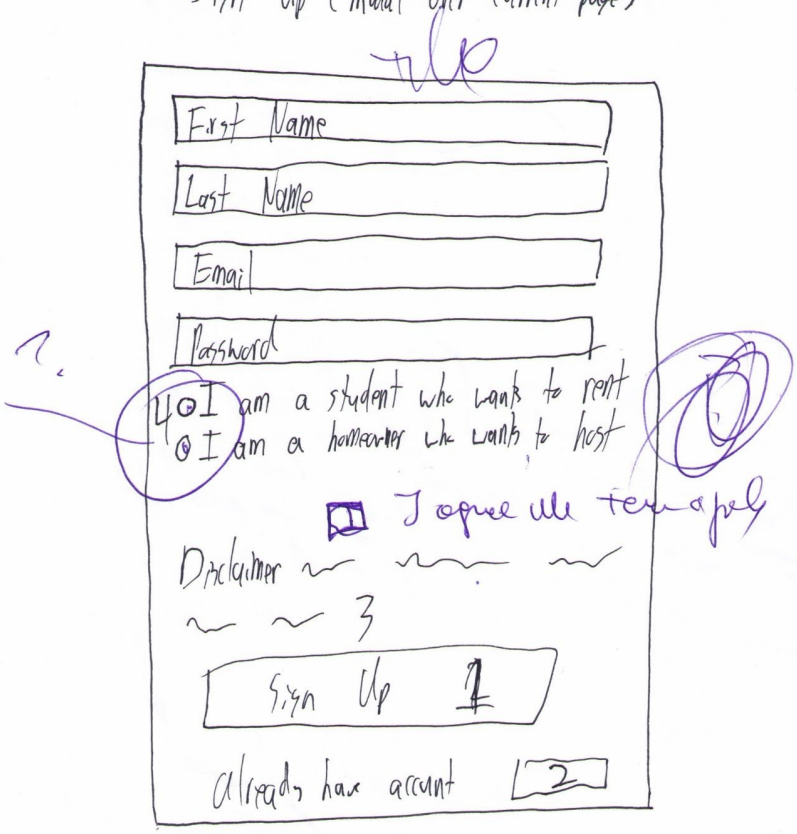
If user is not logged in, submit button will prompt them for login

Ⓓ Sign In (modal over current page)



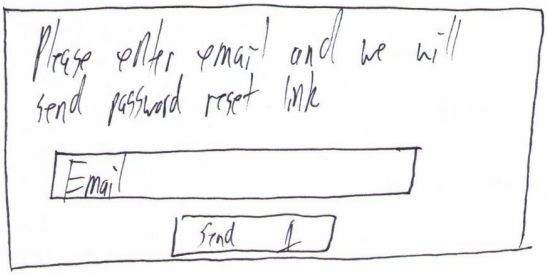
1. Login: Logs in user if info is correct. Flashes error otherwise
2. Sign up: Changes to the sign-up modal
3. Reset password: Shows reset password modal

(E) Sign up (modal over current page)



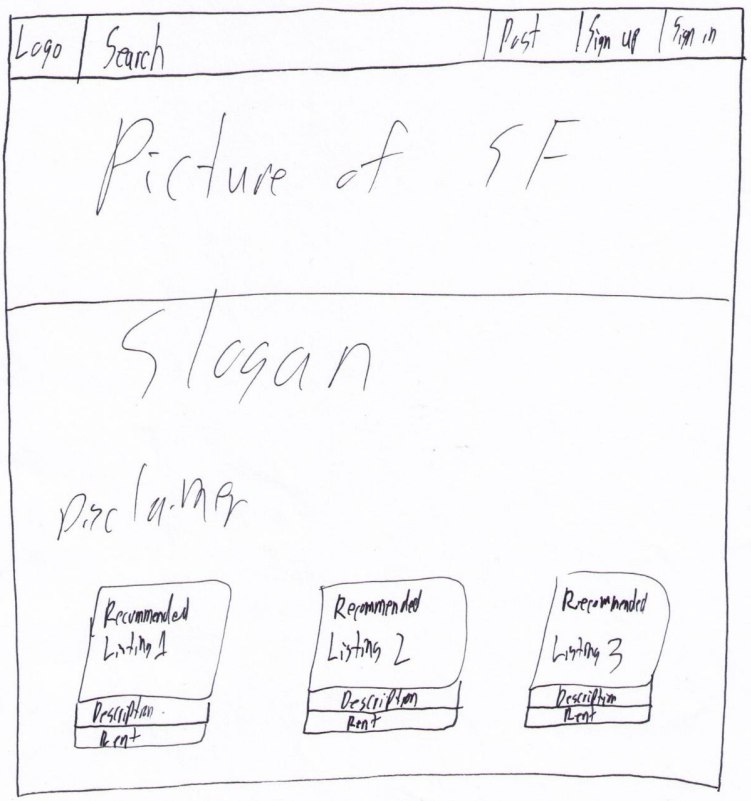
1. Sign up: completes registration
2. Sign In: Changes to the sign in modal
3. Disclaimer: Takes user to Terms of service page
4. Radio buttons: select the type of user, since only students with valid sfsu emails can rent.

Ⓕ Reset password (modal over current page)



1. Send Link: will send a password reset link to the email entered.

(G)


Logo
Search
Post
Sign up
Sign in
Picture of SF
Slogan
Disclaimer
Recommended Listing 1
Description
Rent
Recommended Listing 2
Description
Rent
Recommended Listing 3
Description
Rent

VI. High Level Architecture, Database Organization

Front End Technologies
- JavaScript
- HTML5
- CSS3
- jQuery & jQueryUI
- jQuery Plugins
- Twitter Bootstrap 3
- LESS

Back End Technologies (LAMP stack)
- Linux (Amazon EC2 Instance)
- Apache HTTP Web Server
- MySQL Database
- PHP Programming Language

IDE
- NetBeans

APIs
- Google Analytics Google Maps

Software Framework
- Model-View-Controller(MVC) MINI PHP Framework

Client Side Debugging Tool
- Google Chrome Developer Tool

Browser Support
- Google Chrome, Mozilla Firefox, Safari (latest and second-to-latest versions)

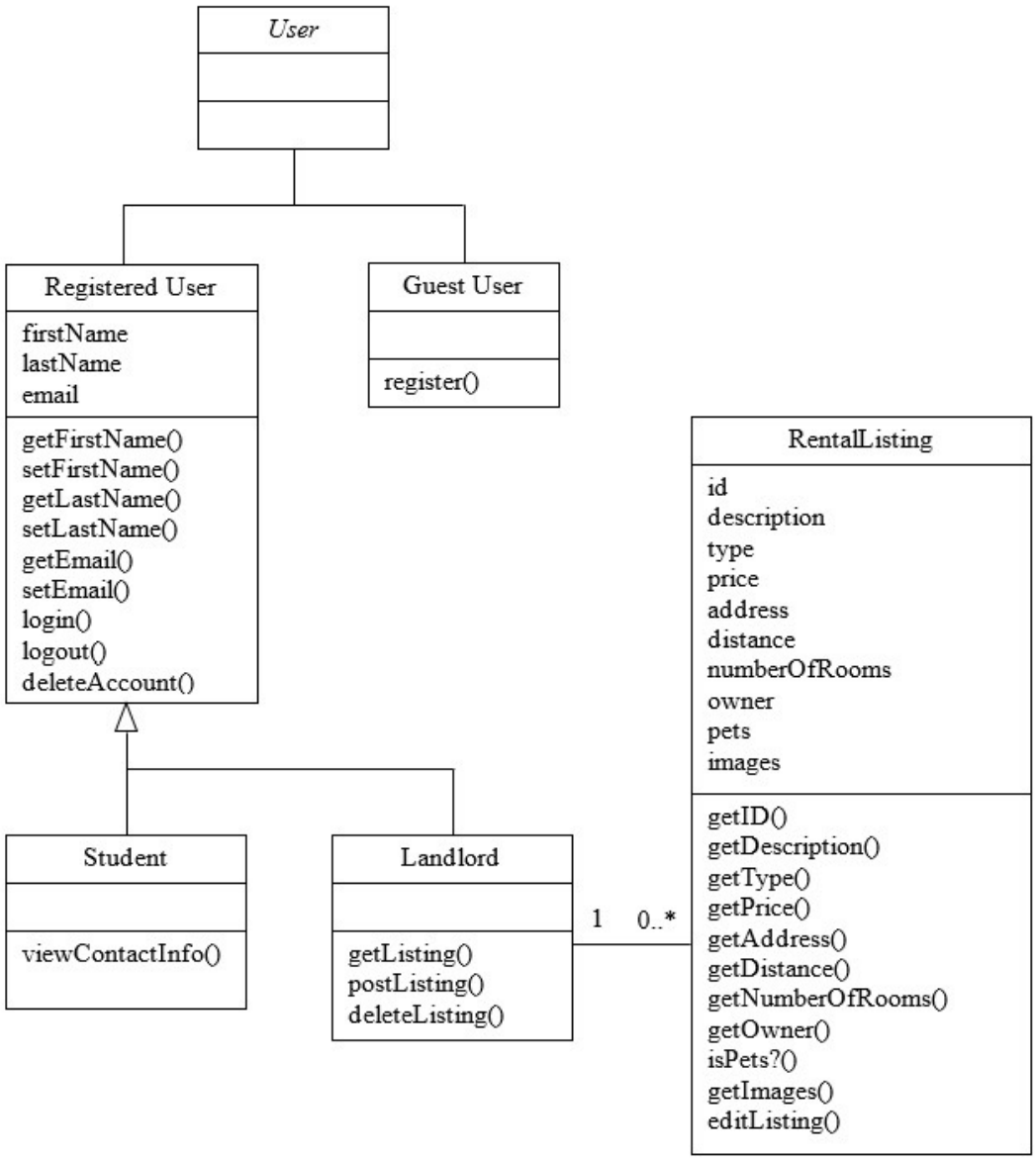
Database Schema: The database is broken into three tables
- Landlords: This table will contain all of the information of the landlords, such contact information, IDs of their listings
- Students: This table will contain all of the information of the students, such as contact information.
- Listings: This table will contain all of the information of the listings, such as owner, address, number of rooms, and type of listing.

Search: Search will be implemented by using the LIKE statement combined with the ‘%’ wild-card. We will concatenate relevant information, such as zip code, street, etc., and then use this statement to calculate how similar the search terms provided by the user are to these concatenated fields.

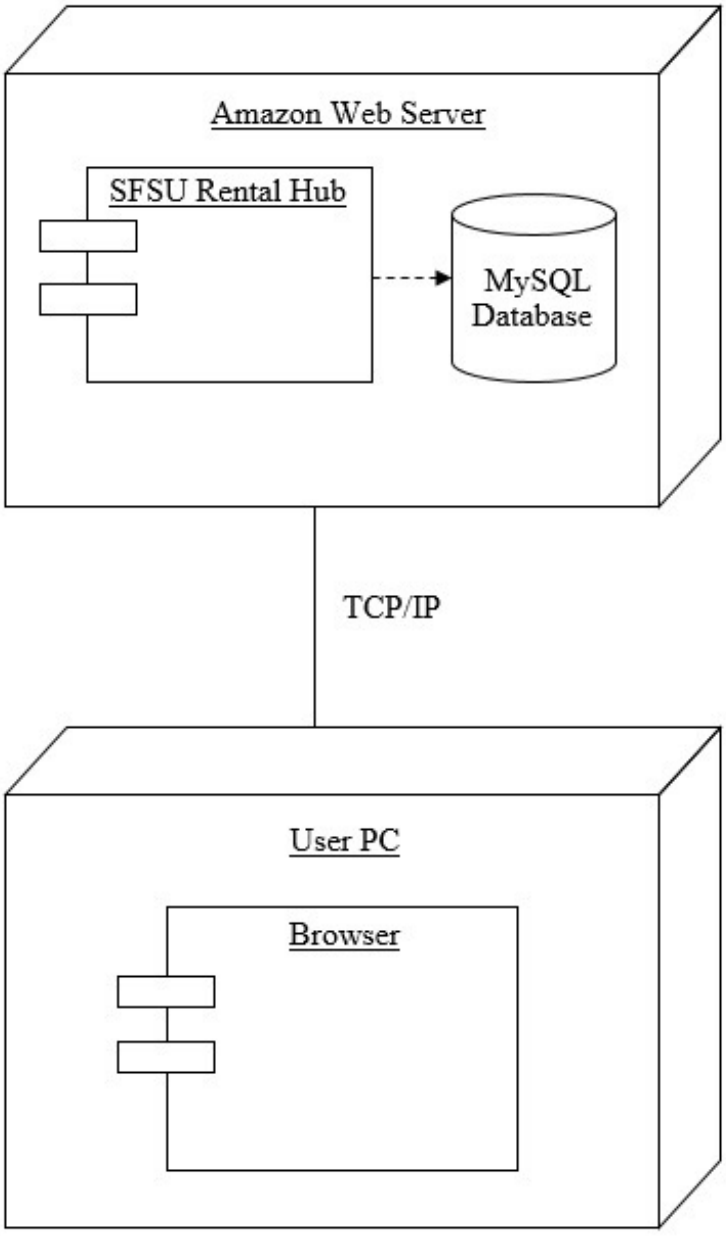
Images: We have decided to store our images in a file structure format. This decision was made after a short investigation of the benefits of BLOBs over file structure, and after deliberation the latter was chosen.

VII. High Level UML Diagrams

UML Diagram of Database



Deployment Diagram



VIII. High Level APIs

Guest User:

- A Guest User is a non-registered user of our website, and it is used to differentiate between a user who is logged in and a user who is not logged in.
- Functions:
    - void register(): A Guest User can register to our website and create an account.

Registered User:

- A Registered User is a user who has created an account and has logged in to our website, and Registered Users are allowed to post and delete their rental listings.
- Functions:
    - String getFirstName(): Return the first name of the Registered User
    - void setFirstName(newFirstName): Set the value of firstName to newFirstName
    - String getLastName(): Return the last name of the Registered User
    - void setLastName(newLastName): Set the value of lastName to newLastName
    - String getEmail(): Return the email of the Registered User
    - void setEmail(newEmail): Set the value of email to newEmail
    - void deleteAccount(): Delete the Registered User's account

Student:

- A Student is a subclass of Registered User and has the ability to view the contact information of a RentalListing. This is to emphasize that our website aims to provide students housing, not just Registered Users.
- Functions:
    - void viewContactInfo(): Allows a Student to view the contact information of a RentalListing

Landlord:

- A Landlord is a subclass of Registered User and has the ability to post and delete Rental Listings.
- Functions:
    - RentalListing getListing(id): Return the RentalListing specified by id
    - void postListing(): Create an instance of RentalListing
    - void deleteListing(id): Delete the RentalListing specified by id

RentalListing:

- A RentalListing is listing created by a Registered User with all rental unit information.
- Functions:
    - int getID(): Return the unique id of the RentalListing
    - String getDescription(): Return the description
    - String getType(): Return the type (apartment, room, etc.)
    - String getPrice(): Return the price
    - String getAddress(): Return the address
    - double getDistance(): Return the distance from SFSU (in miles)
    - int getNumberOfRooms(): Return the number of rooms available
    - String getOwner(): Return the first and last name of the owner
    - Boolean isPets?(): Return true if pets are allowed and false if pets are not allowed
    - Image getImages(): Return the images
    - RentalListing editListing(id): Edit the data fields of a RentalListing specified by id and return a new RentalListing

IX. Key Risks

Skill Risks: As of now, all the members of our group have shown to be highly capable programmers. We do not believe there will be any skill risks in this project

Schedule Risks: Due to the fact that the Fulda team is joining later, there is a bit of a schedule risk since we have had less time to work as a full group, and will also need to onboard the Fulda team.

Technical Risks: Since the stack has been examined thoroughly by Marc Sosnick, and our group is not using any new or risky technologies, there should be no technical risks.

Teamwork Risks: Our group has demonstrated strong and consistent teamwork since the beginning of the project. However, since we have not met the Fulda team I cannot speak for our teamwork in that area, so there is a risk due to unknown variables.

Legal/Content Risks: For the content of the website, our group is going to be creating the majority of it through fake listings. The main risk involved is if we choose to use pictures of houses with proper permission, so our group must be diligent in ensuring we choose content which we are legally allowed to use.

X. Team Organization

Darin Evanow (Team Lead): Darin has been mainly focused on organizing the group, mapping out the high-level concepts of the application, and serving an intermediary between the professors and the group. He will continue along this path, as well as building out parts of the frontend later on in the project cycle.

Abhilash Shrivastava (Tech Lead): Abhilash has been making the majority of the technical investigations and decisions, as well as maintaining the codebase and deployment process. Throughout the project he will continue to be the main source of technical knowledge, while contributing to both the frontend and backend..

Felix Lee (Software Engineer): Felix has spent most of his energy in building out the database model, implementing the database model for the vertical prototype, and writing documentation to inform the rest of the team about the structure of our application. He will continue to focus his energy in building out backend functionality as the project progresses.

Steven Lum (Software Engineer): Steven has been a major player in building out the technical specifications of our project, including UML diagrams, deployment diagrams, and high-level APIs. As the project progresses he will apply his skills in both the frontend and backend where it is required.