SW ENGINEERING CSC 648/848
SECTION 02 FALL 2016

RentSFSU

Group 11: Joseph Costa (kamuela94@gmail.com), Ed Young, Martha Gomez, Yuning Hong, Josh Rubin, Ryan Tang

Milestone 2:
Detailed Requirements, Specs, and Architecture

10/25/2016

History Table:
10/10/2016: Document created

Content and structure for Milestone 2 document for review

1. Use Cases

Potential renter browsing

Crystal is a college student looking for a one bedroom **rental** while attending SFSU for the Spring 2017 semester. She is a heavy internet user and comes across our application during her search. Crystal does not want to drive to school but does not mind walking half a mile. She has a puppy and will need her place to allow dogs. She selects her preferences such as, bedrooms, distance from school, price range, pets, and duration of her stay to filter out the **listings** she will not consider.

Renter booking

Alexander is a freshman at SFSU and has found a listing to book on our application. He wants to book a one room **rental** for the 2016-2017 academic year. He creates a **renter** account proving he is a student at San Francisco State University. He contacts the **landlord** of the listing with the potential move in and move out dates, and requests a viewing of the place. Alexander then awaits for an email from the **landlord** to discuss the viewing.

Landlord

Martin is retired and has decided to put his house for rent on our application since he will out of the country for a year. He loves pets and does not mind if the renter has any. He uses his desktop moderately and is able to sign up for a **landlord** account and create his **listing**. Martin specifies the date range that the **listing** will be available for, the address, and how many rooms are in the house, and specifies that dogs and cats are allowed. After creating the **listing**, he adds images of the house to the **listing**. Martin checks his email regularly and loves that potential **renters** have their inquiries sent to his email through our application. Once Martin's house has been rented he logs into his **landlord** account to manage his **listing** and disables the house he rented.

Administrator (Admin)

Mia is in charge of monitoring our website's content. She is a proficient internet user and sees that there is a **listing** that has inappropriate images. She logs in to her **admin** account to manage the **listings** and removes the **listing** containing the inapropriate images. She then notifies the **landlord** that the **listing** was taken down. The same **landlord** continues to post irrelevant images on their listings so Mia deactivates their account. The **landlord** is notified about the deactivation.

2. Data Definition

User
A user of the site. Can be a renter, landlord, admin, or some combination of the three (based on context).

Renter
A user of the site renting an apartment from a landlord. Is not mutually exclusive from landlord or admin.

Landlord
A user of the site that lists a rental via a listing for renters. Is not mutually exclusive from renter or admin.

Administrator (Admin)
A user of the site that has the abilities to remove others' listings, ban users, and other administrative actions on the site. Is not mutually exclusive from renter or landlord.

Rental
An apartment, house, or other type of rental unit that is up for rent. Includes units up for lease.

Listing
A rental listing that includes the rental terms, such as monthly rent, inclusion of utilities, maintenance, and so on.

3. Functional Requirements

Priority: 1	Description
User	
1	User shall be able to create an account.
1	Users shall be able to create a listing for an apartment with the criteria: Address Number of bedrooms & bathrooms Smoking preferences Furnished/Unfurnished Utilities Square footage Monthly rent Deposits Pet policy Start date of rental agreement Rental agreement period
1	User shall be able to search the listings without login/registering.

1	Users shall be able to filter listings by number of rooms, bathrooms, distance from SFSU, smoking/nonsmoking, utilities, price range, and whether pets are allowed.
1	Users shall be able to see listings based on their criteria on the map after searching.
Landlord	
1	Landlords shall be able to upload images of their rental to their listings.
1	Listing form shall auto-populate if an existing rental with a matching address is found.
1	Landlords shall be able to edit the listing form if if has been auto-populated.
1	Landlords shall be able to list multiple rentals on a single listing.
1	Landlords shall be able to accept/deny requests for tours from potential renters.
1	Landlords shall be able to communicate with the prospective renters through registered accounts via an internal messaging system.
Renter	
1	Renters shall be able to contact Landlords via an internal messaging system.
Administrator (Admin)	
1	Administrative users shall be able to disable inappropriate listings.
1	Administrative users shall be able to disable accounts for inappropriate use and/or harassment.
Application	
1	Application shall offer a page with all available listings.
1	Application shall display each listing on the list with a summary of each Rental as well a a short description.
1	Application shall display a map of the general area of the listing.
1	Application shall obscure the physical location of a rental by placing a radius approximating the rental location for Landlord’s security
Priority: 2	
User	
2	Users shall be able to move through the map to see locations of the listings.
2	Users shall be able to bookmark listings for later use
2	Users shall be able to search based on distance from SFSU.

Landlord	
2	Landlords shall be able to manage their listings after they have been posted, including the following: Add/remove photos Deleting or disabling their listing Changing the description of the rental in their listing Changing the details of their rental for their listing (see list in Users section)
Application	
2	Application shall display a short description of listing on the map.
Priority: 3	
User	
3	Users shall be able to search based on distance from transportation.
Application	
3	Application shall calculate average transit time to SFSU by car and public transportation.

4. Non-Functional Requirements

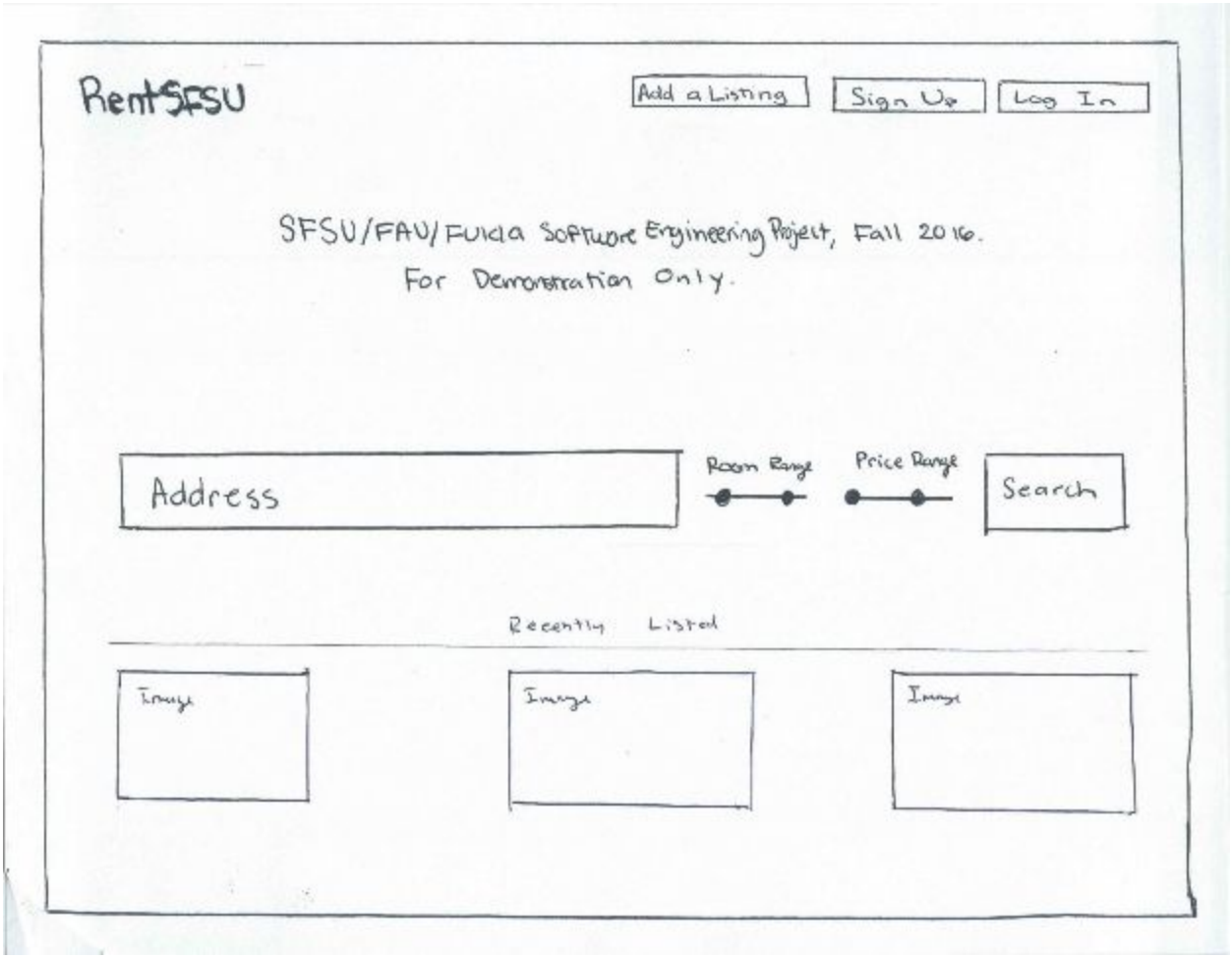
1. Application shall be developed using the provided LAMP stack.
2. Application shall be developed using the pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks shall be explicitly approved by Marc Sosnick on a case by case basis.
3. Application shall be hosted and deployed on Amazon Web Services.
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, and Chrome. It shall degrade gracefully for different sized windows using the 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. Site security: basic best practices shall be applied (as covered in the class)

14. 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
15. 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).

5. UI Mockups and Storyboards

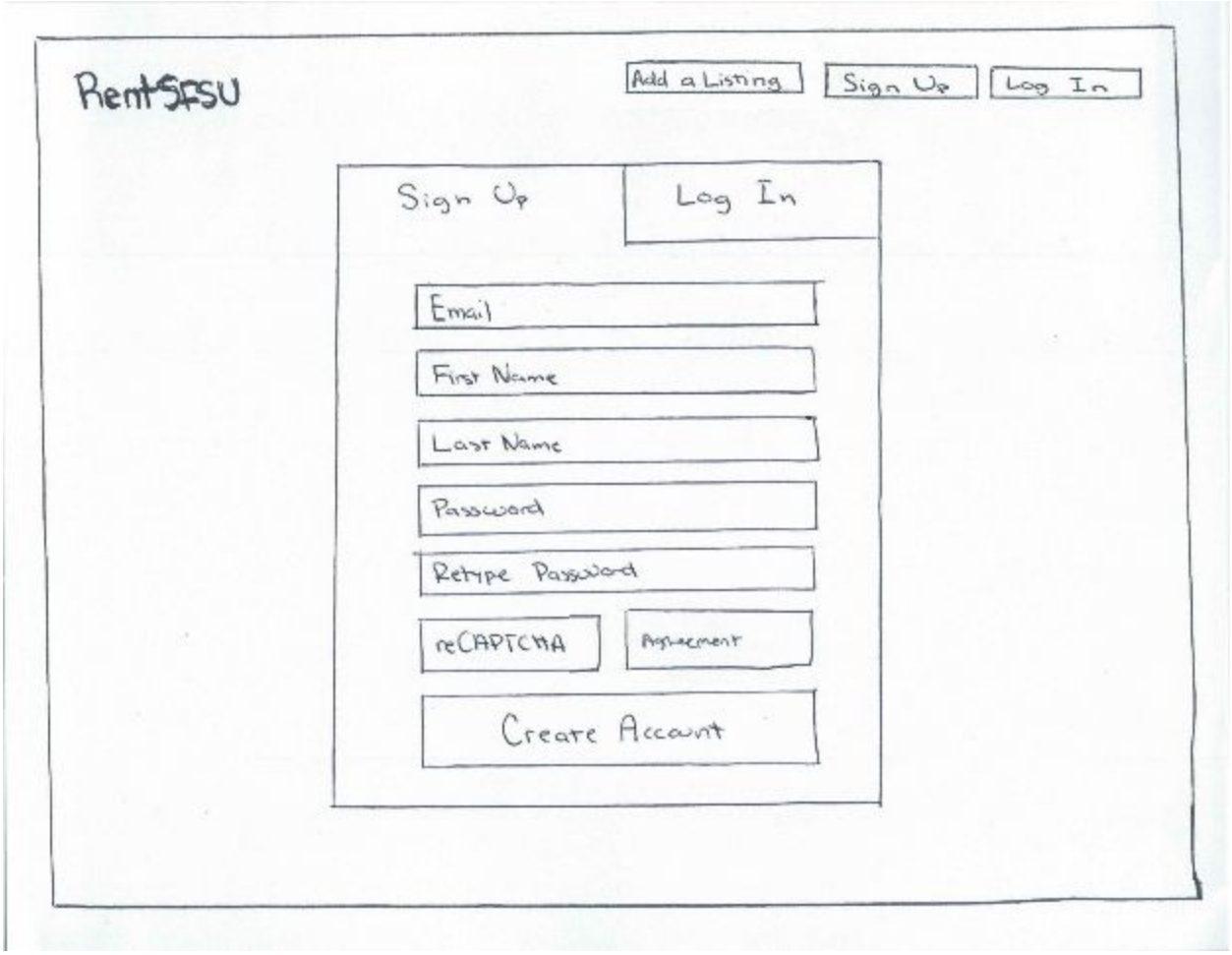
Home Page

User can navigate to signup/login, add listing, and listing page through the home page. The RentSFSU button at the top left corner brings User back to the home page. User can navigate to the listing page in two ways. First is by clicking on the search button. Second is by clicking on a recently listed(bottom of the page). Before clicking on the search button, user can filter the search by using the address search bar, room range slider, and/or price range slider.



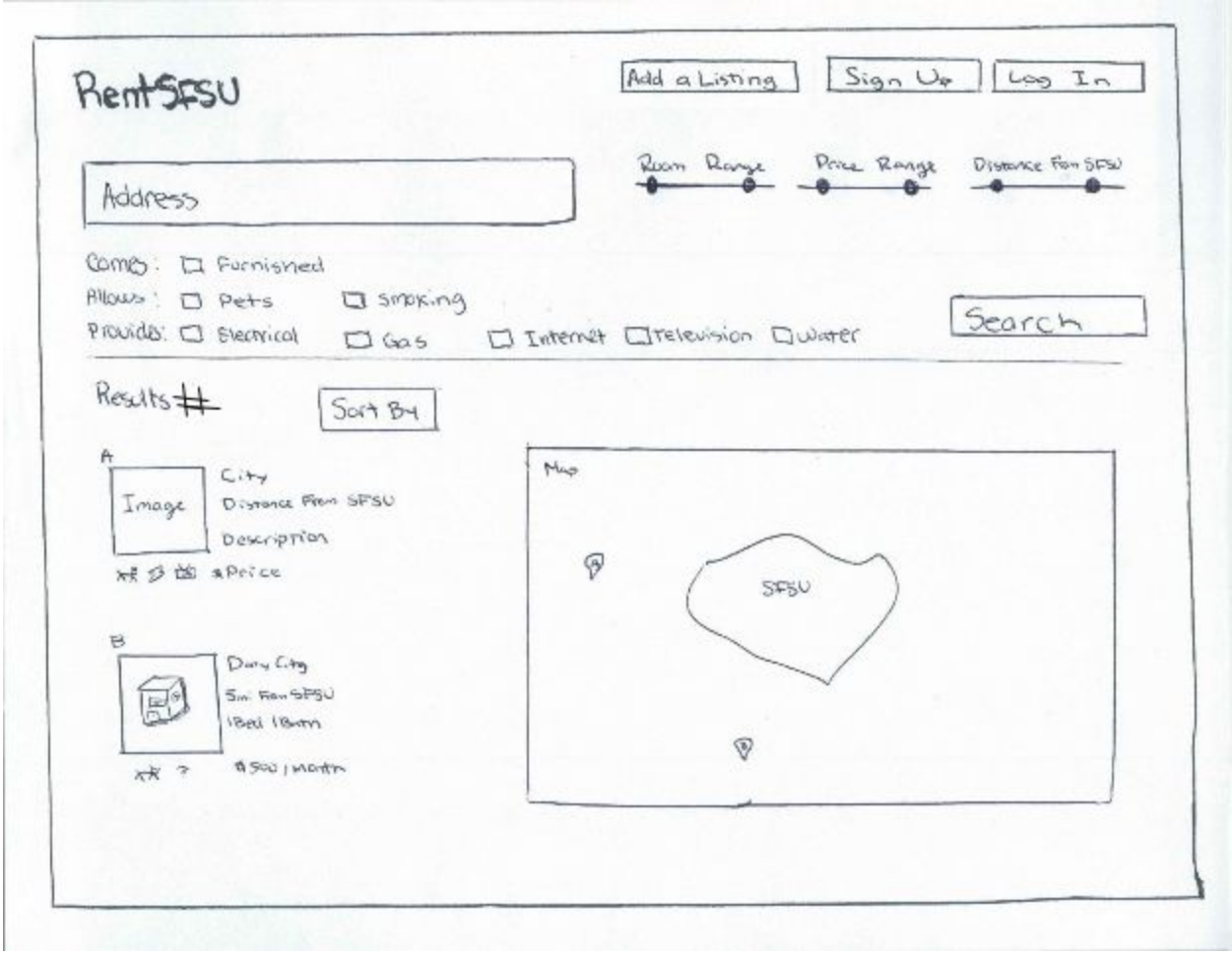
Sign Up/Log In

This page will be a modal window. User can either sign up or sign in by clicking the the Sign Up/Log In tabs. User will not be able to create an account unless they fill out all the fields.



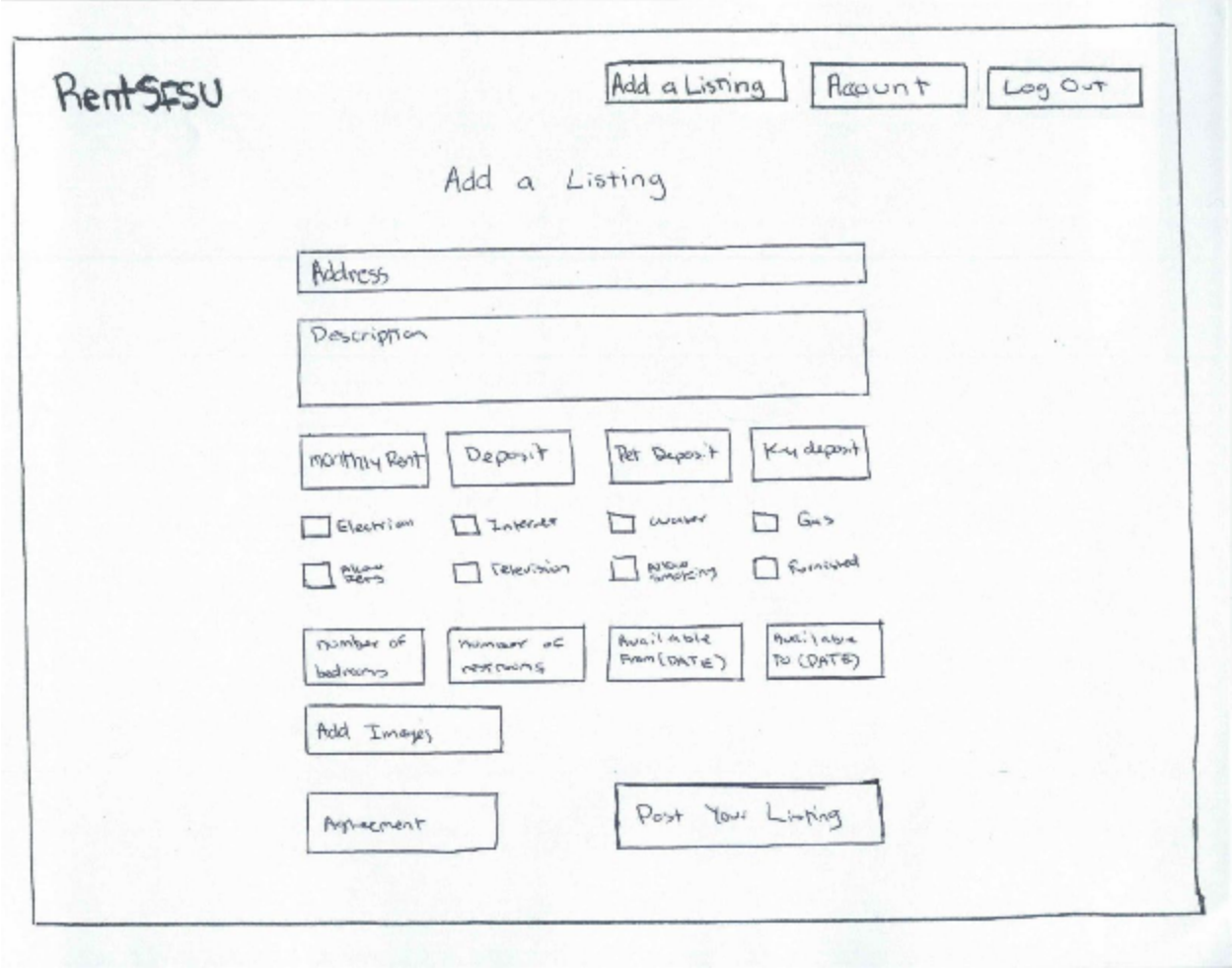
Listing Page

In this page, user can check out listings by landlords. User can filter out the listings by using the search bar, sliders, and checkboxes. The sort by dropdown button is used to sort listings. A map will be shown on the right side of the page. Clicking on the listing images navigates the user to a detailed listing page.



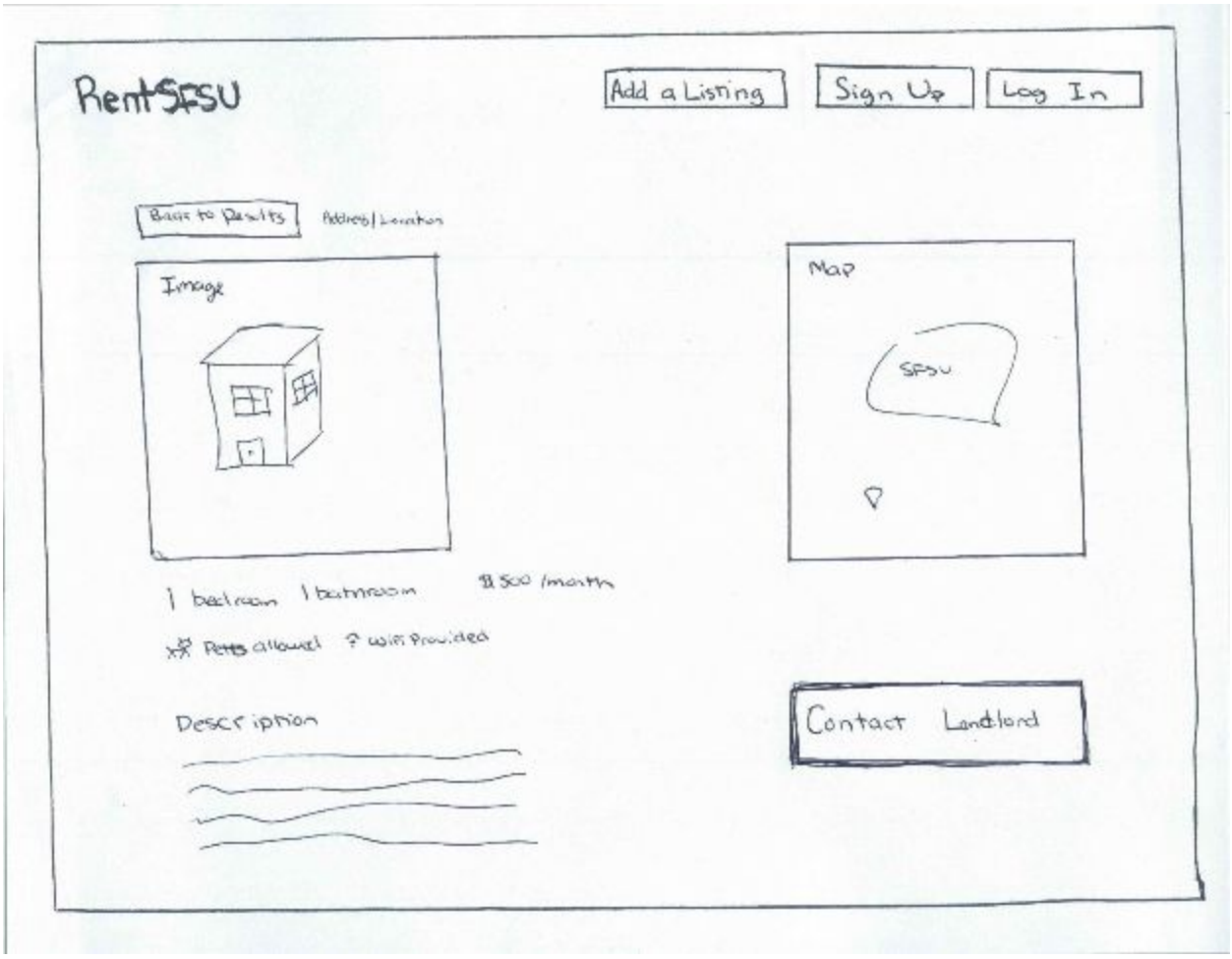
Add Listing Page

This page can only be accessed by a landlord. This page is accessed through the add a listing button on the top right corner of any page. On the page, landlord adds a listing by filling out a form. The form consist of the following input fields: address, description, monthly rent, deposit, pet deposit, key deposit, #bedrooms, #restrooms, available from/to date. The form also consist of a list of utility checkboxes: electrical, internet, water, gas, allow pets, television, allow smoking, furnished



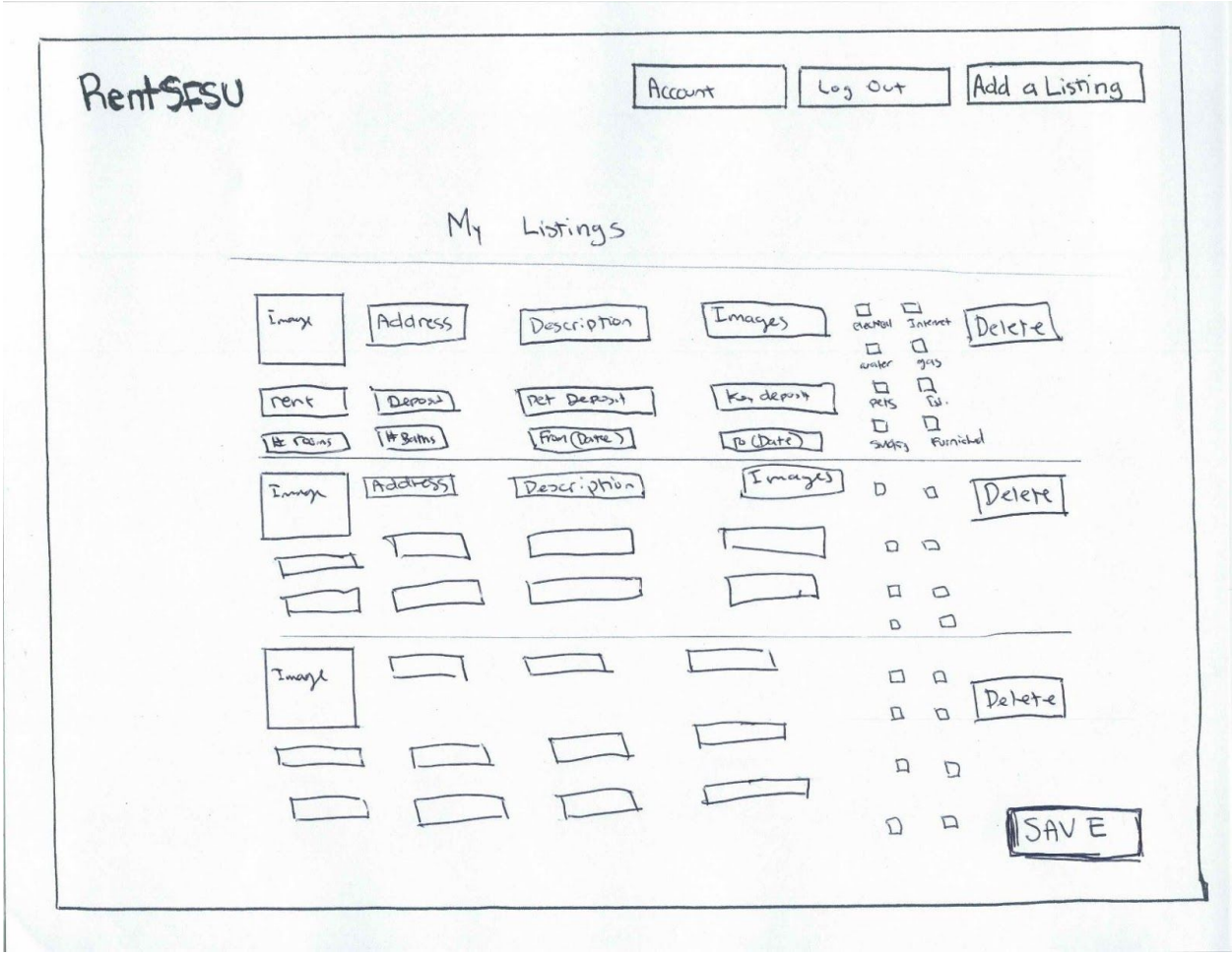
Detailed Listing Page

This page contains all the details about the listings. We provide a button for users to return back to their search results page. Images that the landlord uploads are displayed along with a map of the area for the rental. If logged in, you can contact the landlord by clicking the contact landlord button. If users are not logged in and click the contact landlord button you will be prompted to log in or sign up.



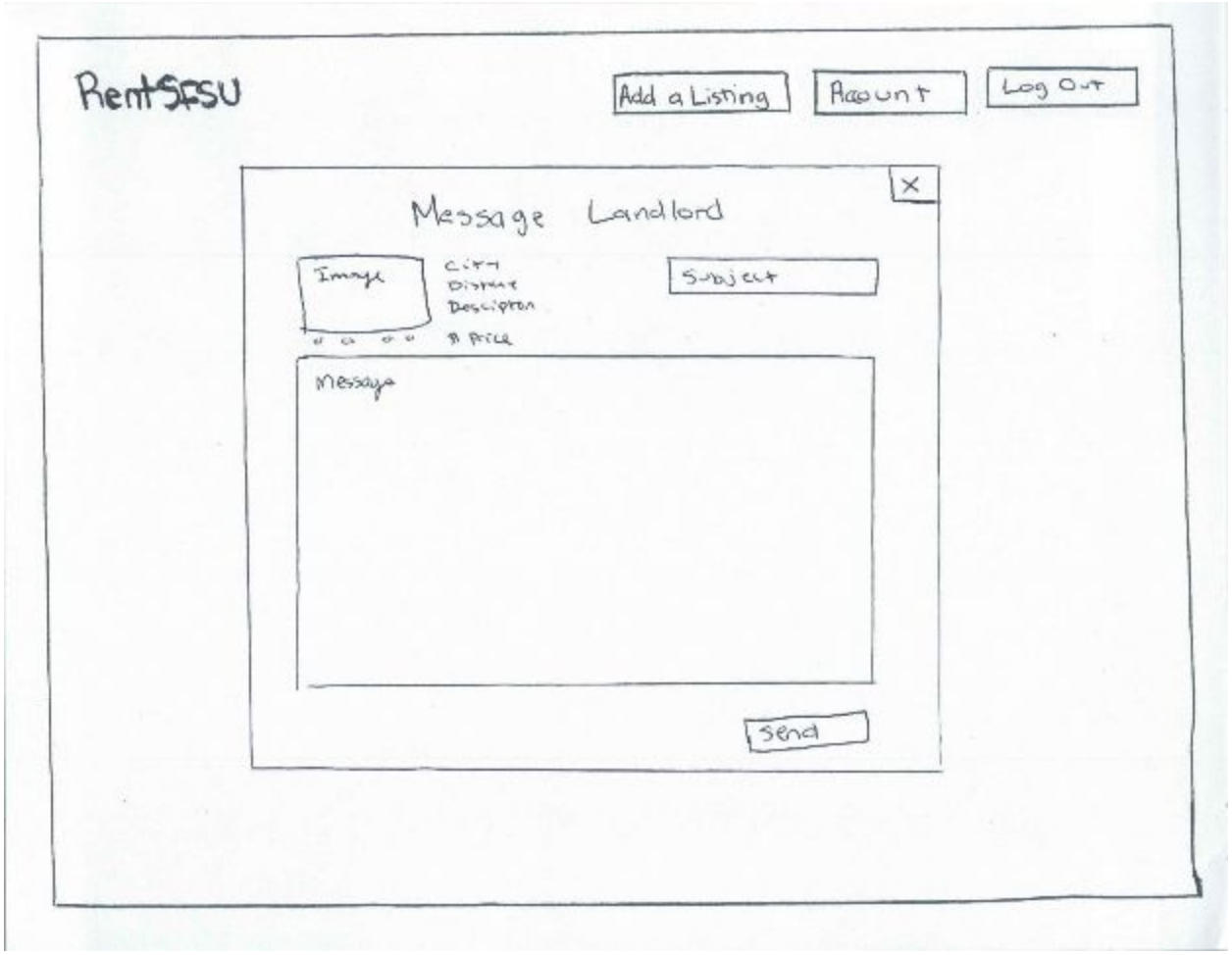
Landlord Account Page

Landlords will be able to edit any part of their listing or delete it through the account button, which appears once landlords are logged in.



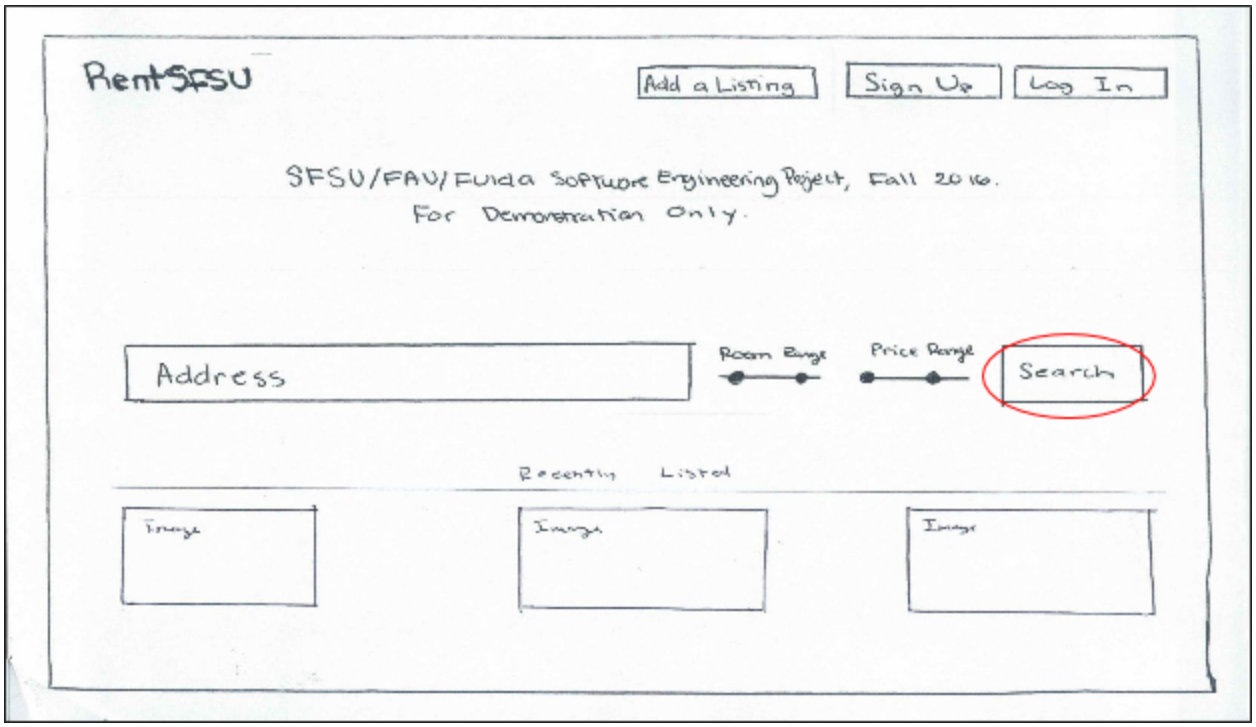
Contact Landlord Page

Contacting landlord page will be a modal window. This modal can only be accessed by a renter through the detailed listing page by clicking on the contact landlord button. This is where renter contacts the landlord if they want to rent out the place. Renter must fill out the subject and message input fields before sending.

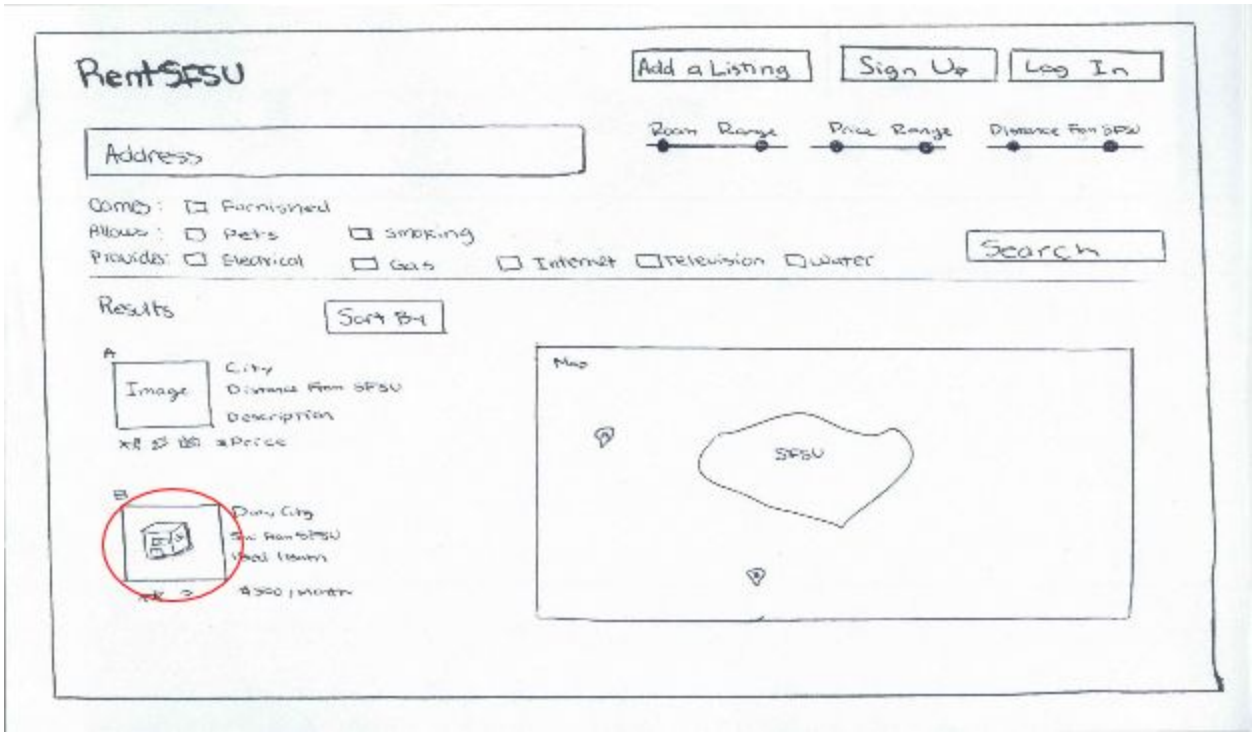


Renter Example

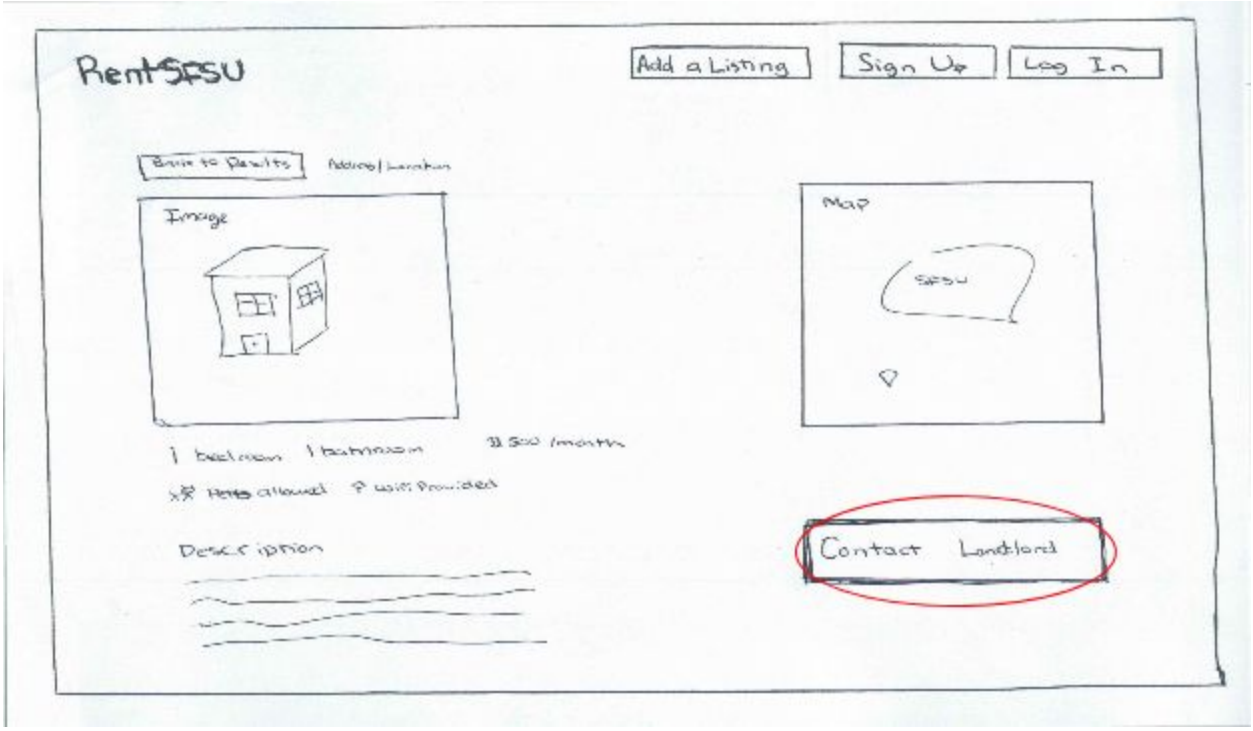
User starts off at the home page.



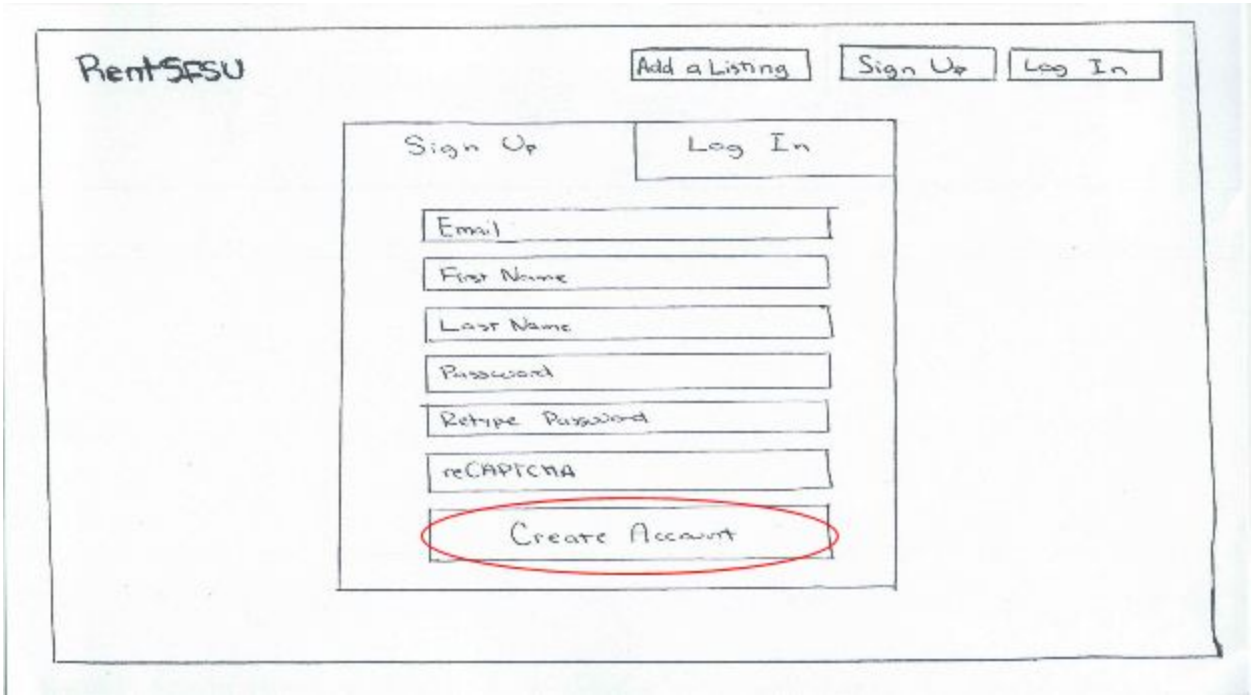
User then hits search and navigates to the listing page.



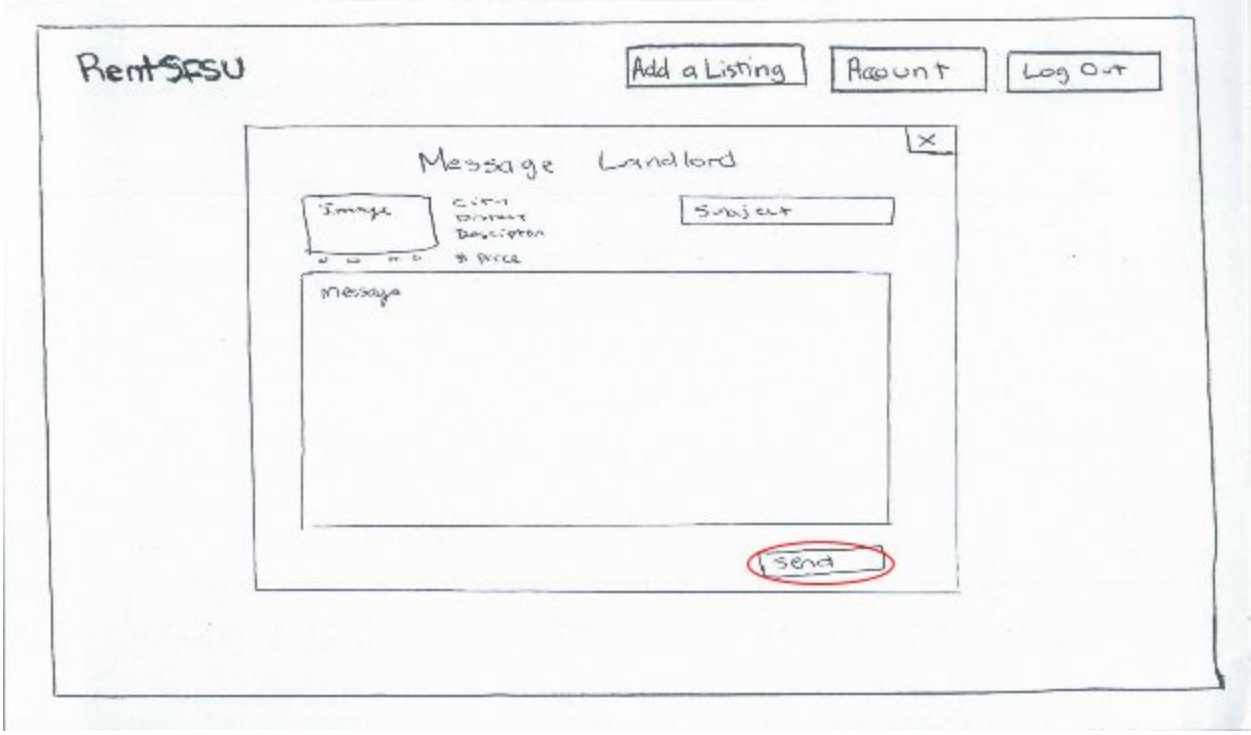
User finds a listing he/she is interested in, and navigates to the detailed listing page by hitting an image from the listing page.



User decides this is the place he/she wants and decides to contact the landlord. After hitting the contact landlord button from the detailed listing page, the user is prompted by a signup/login modal.

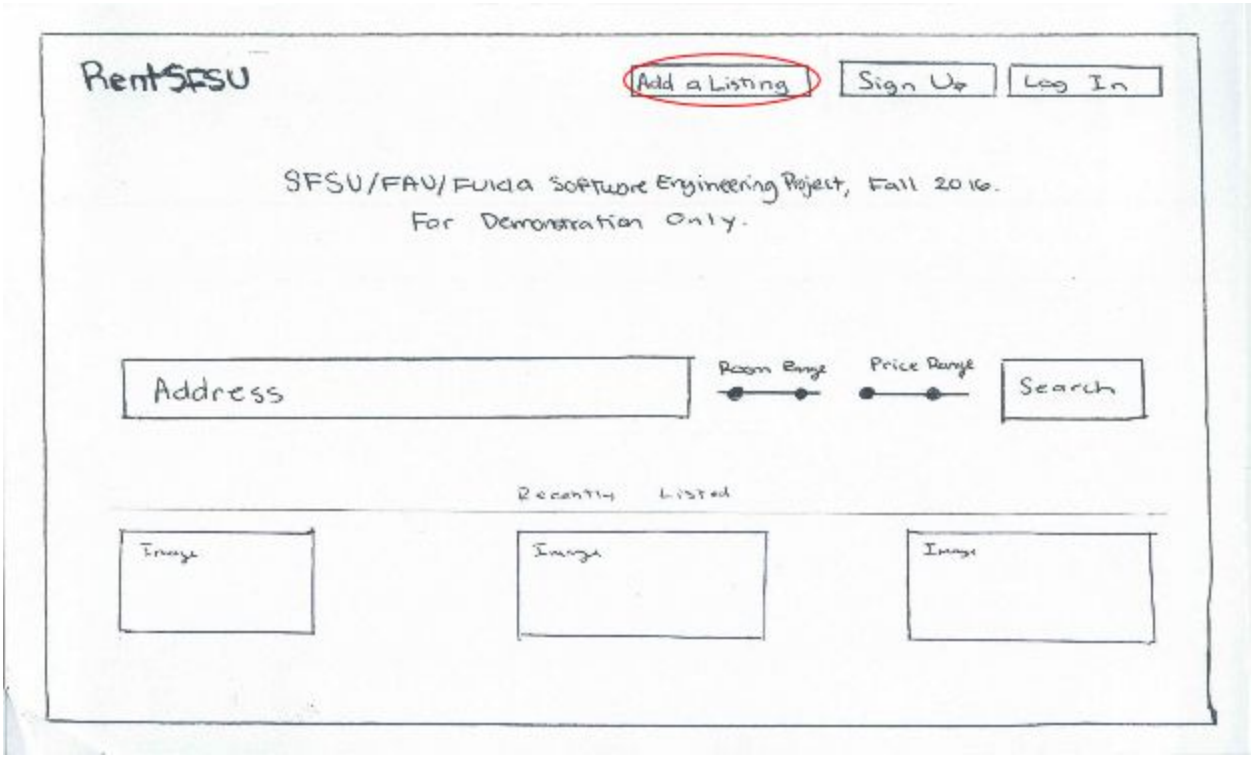


After signing up, the user contacts the landlord through a contact landlord modal. The user requests a viewing of the page and hits the send button.

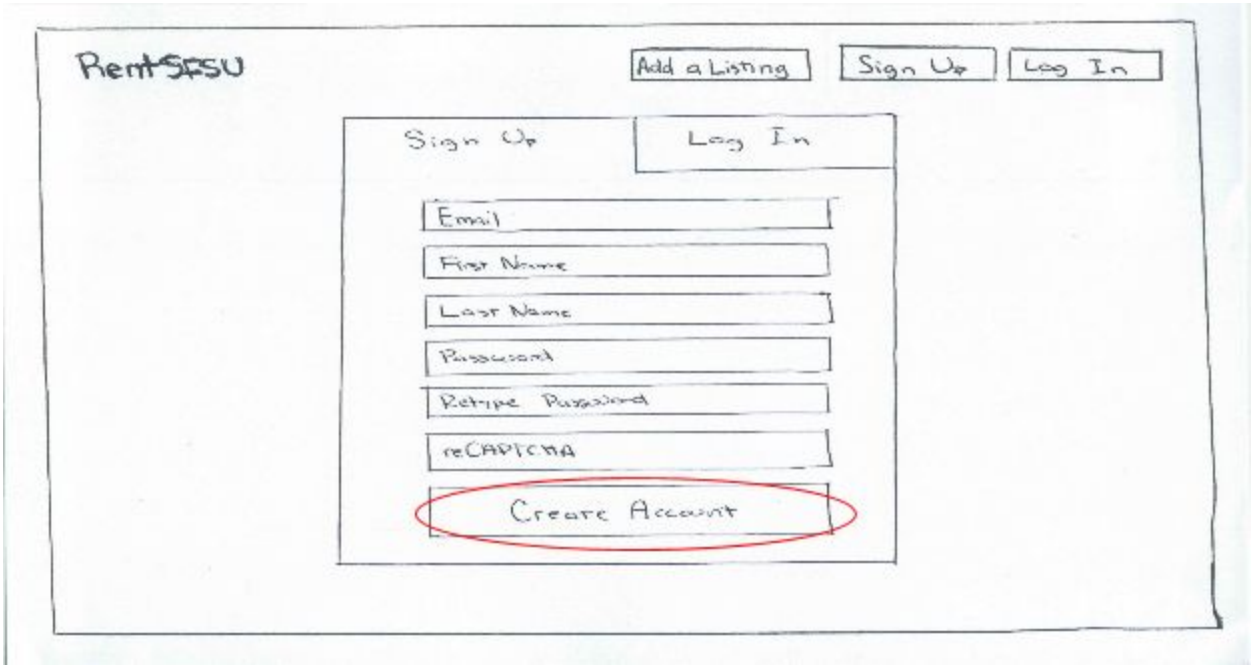


Landlord Example

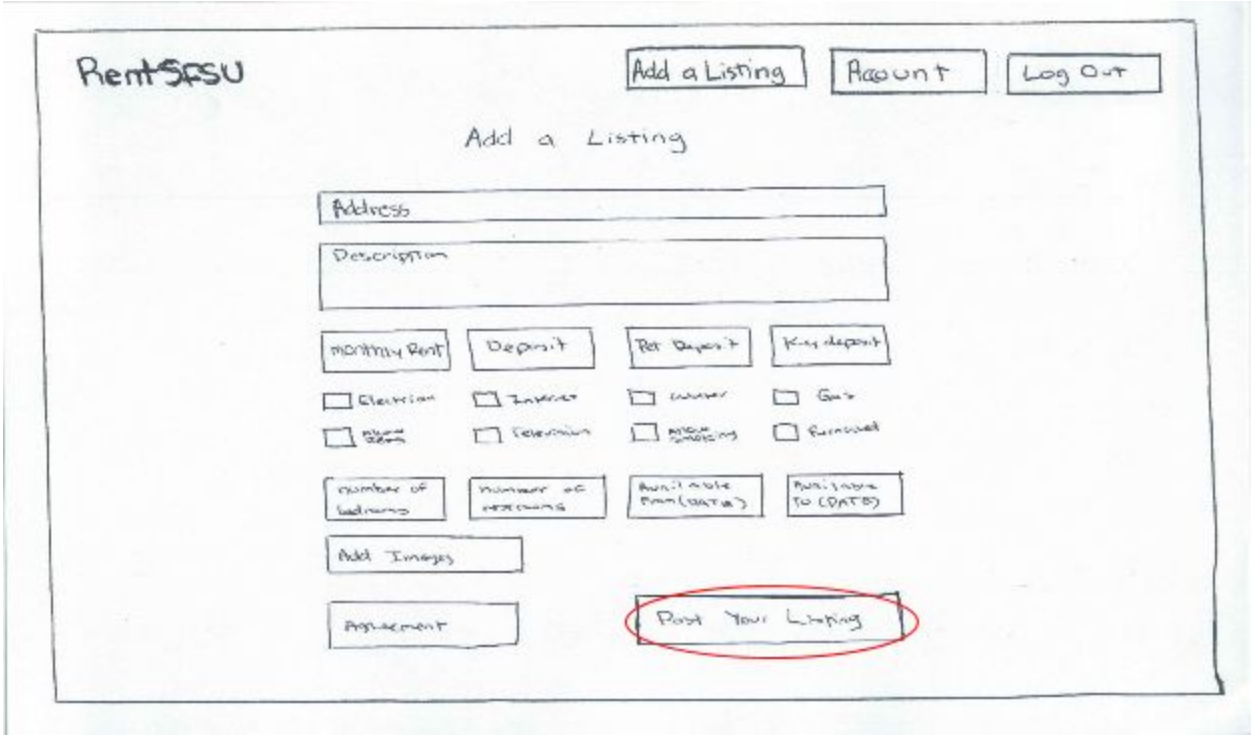
User starts off at the home page.



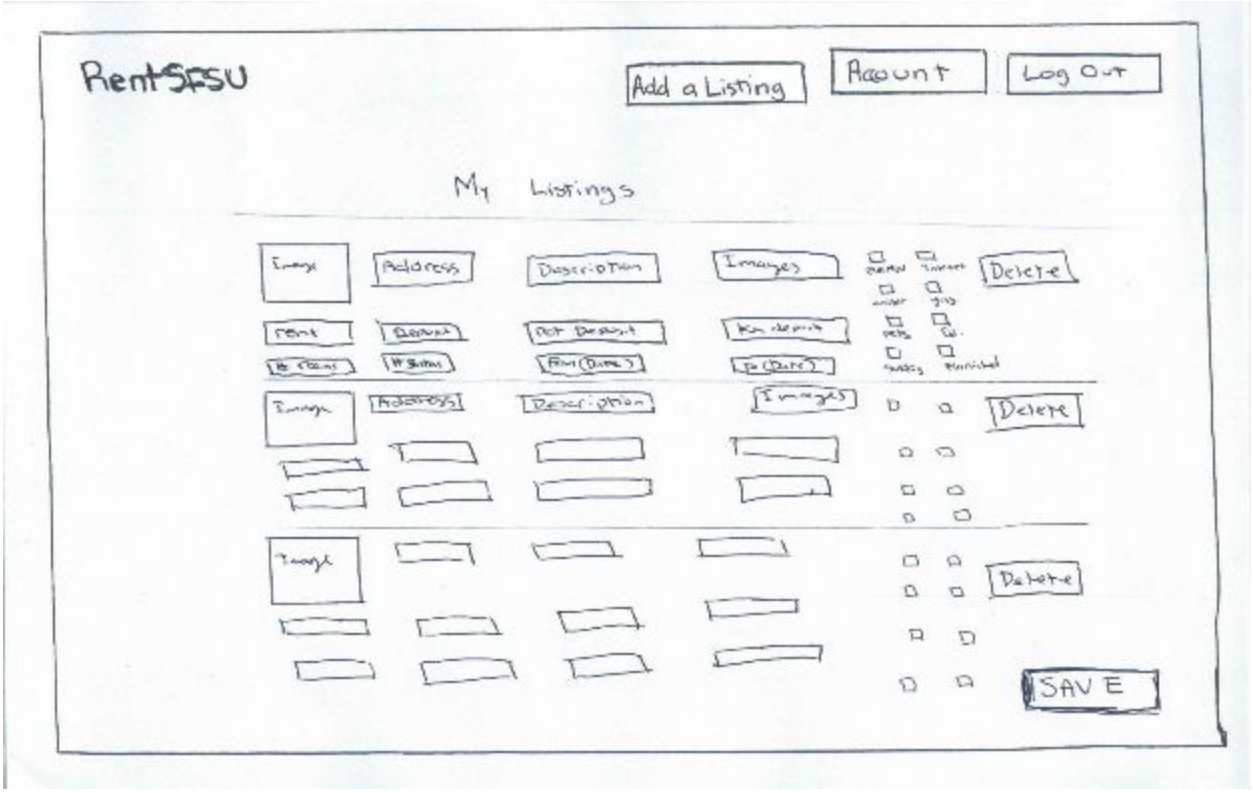
User is prompted with a sign up/sign in modal after clicking on add listing button from the home page.



After signing up as a landlord, the user resumes to add a listing by filling out a form on the add listing page.



After filling out the listing form, the user is directed to his/her landlord account page.



6. High-Level System Architecture and Database Organization

High-level System Architecture

- LAMP stack hosted on Amazon Web Services
    - Ubuntu 16.04 LTS (Xenial Xerus)
    - Apache 2
    - MySQL 5.7.12
    - PHP 7.0.8
- MINI PHP Framework: a simplified Model View Controller (MVC) framework
- Netbeans IDE
- CSS
    - Bootstrap: framework for responsive design, scales for Desktop and Mobile
    - SCSS: Extension to CSS which adds variables & mixins
    - Compass: SCSS mixin library that extends base SCSS for portability across browsers
- JavaScript
    - jQuery: DOM manipulation
- Git repo management - GitLab
- Google Analytics for major site functions
- Supports any desktop/laptop compatible browser
    - Mozilla, Safari, Chrome (current and most recent, previous version)
    - No mobile development, but application shall resize to fit mobile screens
- Search Architecture
    - Search by distance from SFSU
    - Search by address

Database Organization

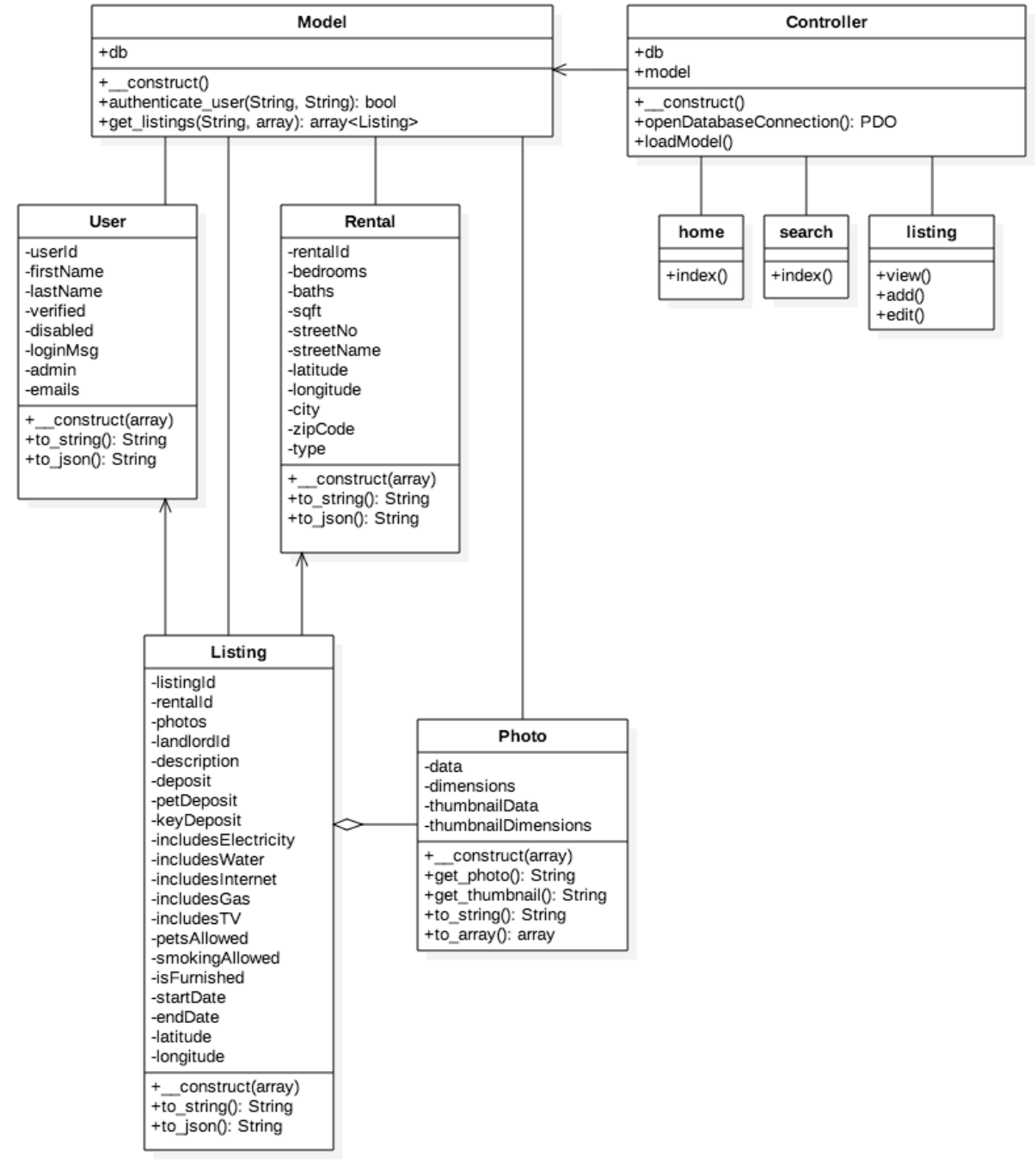
- Users
    - Id: Integer not null auto increment
    - FirstName: Nvarchar(20) not null
    - LastName: Nvarchar(40) not null
    - Password: Char(60) Binary not null
    - Verified: Boolean
    - Disabled: Boolean
    - LoginMsg: Nvarchar(200)
    - Admin: Boolean

- Emails
    - EmailId: Integer not null auto increment
    - Address: Nvarchar(80) not null
    - UserId: Integer not null (Foreign Key)
    - IsPrimary: Boolean
- Rentals
    - RentalId: Integer not null auto increment
    - StreetNo: Integer
    - StreetName: String not null
    - City: Nvarchar(20) not null
    - ZIP: Char(5) not null
    - RentalType: Integer not null (Foreign Key)
    - Bedrooms: Integer
    - Baths: Integer
    - SqFt: Integer
    - Latitude: Double
    - Longitude: Double
- RentalType
    - RentalTypeId: Integer not null auto increment
    - Description: String not null
- Listings
    - ListingId: Integer not null auto increment
    - RentalId: Integer not null (Foreign Key)
    - LandlordId: Integer not null (Foreign Key)
    - MonthlyRent: Integer not null
    - Description: Nvarchar(2000)
    - Deposit: Integer
    - PetDeposit: Integer
    - KeyDeposit: Integer
    - Electricity: Boolean
    - Internet: Boolean
    - Water: Boolean
    - Gas: Boolean
    - Television: Boolean
    - Pets: Boolean
    - Smoking: Boolean
    - Furnished: Boolean
    - StartDate: Date not null
    - EndDate: Date

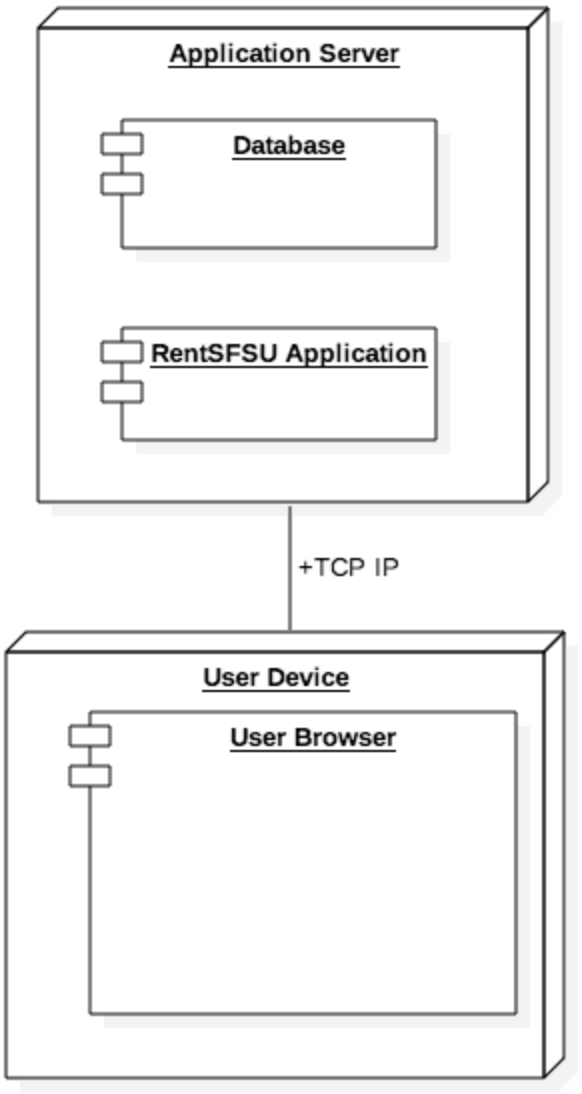
    - Latitude: Double
    - Longitude: Double
- Photos
    - PhotoId: Integer not null auto increment
    - ListingId: Integer not null (Foreign Key)
    - PrimaryPhoto: Boolean not null
    - Order: Integer not null
    - Height: Integer not null
    - Width: Integer not null
    - Data: BLOB not null
    - Format: String not null
- Thumbnails
    - ThumbnailId: Integer not null auto increment
    - PhotoId: Integer not null (Foreign Key)
    - Height: Integer not null
    - Width: Integer not null
    - Data: BLOB not null
    - Format: String not null
- Messages
    - MessageId: Integer not null auto increment
    - SenderId: Integer not null (Foreign Key)
    - RecipientId: Integer not null (Foreign Key)
    - ListingId: Integer not null (Foreign Key)
    - Title: Char(140)
    - Body: Char(1000) not null

7. High-level UML Diagrams

Class Diagram



Component Diagram



8. High-level APIs

Google Maps API

Create map with nodes of listings
Directions to campus
Obfuscation of address

Models

This object allows you to search through the database to find specific listings.

fetch_listings(String);

Listings

This object controls information inside a Listing

__construct(array);
insert_photos(BLOB, int);
get_photo(int); Returns BLOB
remove_photos(int);
make_primary(int);

set_description(nvarchar(2000));
get_description(); Returns array(2000)

set_deposit(int);
get_deposit(); Returns int

set_pet_deposit(int);
get_det_deposit(); Returns int

set_key_deposit(int);
get_key_deposit(); Returns int

set_electricity(boolean);
get_electricity(); Returns boolean

set_water(boolean);
get_water(); Returns boolean

set_gas(boolean);
get_gas(); Returns boolean

set_television(boolean);
get_television(); Returns boolean

set_pets(boolean);
get_pets(); Returns boolean

set_smoking(boolean);
get_smoking(); Returns boolean

set_furnished(boolean);
get_furnished(); Returns boolean

set_start_date(Date);
get_start_date(); Returns boolean

set_end_date(Date);
get_end_date(); Returns string

set_longitude(Double);
set_latitude(Double);

Users

This class holds the Users information

__construct(array);
get_user_id(); Returns int

set_first_name(String);
get_first_name(); Returns String

set_last_name(String);
get_last_name(); Returns String

set_password(String);
verify_password(String); Returns boolean

is_disabled(boolean);
check_disabled(); Returns boolean

is_verified(boolean);
check_verified(); Returns boolean

is_admin(boolean);
check_admin(); Returns boolean

set_login_msg(array[200]);
get_login_msg(); Returns array

Emails

This object handles a Users email information

__construct(array);
add_address(String);
remove_address(int); Returns String
get_address(); Returns String

set_user(int);

set_primary(boolean);
is_primary(); Returns boolean

Rentals

This object handles the rental information in Listings

__construct(array);

set_street_no(int);
get_street_no(); Returns int

set_street_address(String);
get_street_address(); Returns String

set_city(String);
get_street_address(); Returns String

set_zip(String);
get_zip(); Returns String

set_rental_type(int);
get_rental_type(); Returns int;

set_bedrooms(int);
get_bedrooms(); Returns int

set_baths(int);
get_baths(); returns int

set_sqft(int);
get_sqft(); returns int

set_longitude(double);
set_latitude(double);

RentalType

Holds the information of what kind of Rental it is.

__construct(array);

set_description(String);
get_description(); returns String

Photos

Holds and changes the Photo information.

set_listing_id(int);

set_primary_photo(int);
get_primary_photo(); returns
is_primary_photo(); Returns boolean

set_order(int);
set_height(int);
set_width(int);
insert_data(BLOB);
set_format(String);

Thumbnails

set_photo_id(int);
set_height(int);
set_width(int);
set_format(String);

Messages

Creates an instance in which Users can talk to one another. Also provides the Listing they are talking about.

set_sender_id(int);
set_lister_id(int);

set_listing_id(int);
set_title(String);
set_body(String);

9. Key Risks

Skills risks: Poor knowledge of tech stack. Lack of knowledge in Javascript, complete lack of knowledge in SQL. Acceptable knowledge of PHP. Extensive training in tech stack needed. Angular, though approve, to be dropped from the tech stack as a result.

Scheduling risks: Current team scrum time may be too close to class scrum, impromptu meetings may be necessary.

Technical risks: No known technical risks at this time.

Teamwork risks: Perceived lack of motivation from team to learn things of their own volition. Will continue to monitor the situation, but in the meantime, proactively handing team members ownership over portions of the application, such as user authentication and the add listing form, so they have some direction to begin learning the tech stack.

Legal/SW risks: No known Legal/SW risks at this time

10. Team Organization

Joseph Costa: Team lead and Backend developer

Ed Young: CTO and full stack developer

Martha Gomez: Frontend developer

Yuning Hong: Frontend developer

Josh Rubin: Backend developer

Ryan Tang: PHP developer