Simplease

Team 36 - Design Document

Rohan Rao, Matthew Fouts, Nikitha Jagadish, Rizwanulla Mohammed, Jordan Stoddard, Abhi Gupta

Table of Contents

Purpose **3**
Functional Requirements 3
Non-Functional Requirements: 5
Design Outline **7**
Design Issues **8**
Non-functional Issues 8
Functional Issues 9
Database Schema **1**1
Design Details **1**2
Explanation of the Data Classes **1**3
States of the UI **1**5
Sequence of Events When a User Creates a Maintenance Request 16
Example: When a User Adds a New Building and then an Apartment to that Building 17
State Diagram from Tenant’s Perspective 18
Landlord Client Sequence/Flow Diagram 19
User Interface Mockups
Listing Apartments - Image View **20**
Listing Apartments - Map View 21
Notification Screen Mockup 22
Maintenance Screen Mockup 23

Purpose

Potential residents of apartments want accurate information and current residents want easy communication with apartment administration. Apartment administration wants an easy way to market their apartments to potential future residents as well as a way to monitor the satisfaction of current residents.

Simplease creates an easy line of communication between residents, future residents, and apartment administration. It allows for residents to efficiently lodge complaints like leaking pipes, broken locks, water issues, and so much more. At the same time it connects current residents with potential future residents so that future residents can get an accurate representation of the apartment. Future residents will also have consolidated information of all apartments in the area that are available for leasing.

Simplease will take data provided by residents and landlords and incorporate them into a simple and user-friendly mobile application. We provide a more general solution that can be utilized in any city with apartments for lease to create a friendlier experience without having to develop a unique application at each location.

Functional Requirements

Users can control their notifications:

As a user,

1. I would like to be able to control when and how I get my notifications.

As a tenant,

2. I would like to be notified when my bill has been posted.

As a landlord,

3. I would like to send notifications to my tenants (rent is due, late bills, emergencies, etc.)
4. I would like to notify tenants of packages that have been dropped off at the office (if applicable).

As a service provider,

5. I would like to be notified on my assignments and their locations. [If time permits]

All information should be secure:

As a user,

6. I would like my personal information to be secured.

As a tenant,

7. I would like to be able to store important documents safely within the app,

As a landlord,

8. I would like all my customers' sensitive information to be stored safely

<u>Account information:</u>

As a user,

9. I would like to be able create a Simplease account.
10. I would like to be able sign into my account.
11. I would like to be able to make changes to my account information.

As a tenant,

12. I would like to be able to view my documents whenever I want.

As a landlord,

13. I would to be able to update a potential tenant to a tenant account permissions and vice versa.

<u>Displaying, updating and accessing apartment listings and apartment availability:</u>

As a guest,

1. I want to view apartment listings based on location

As a potential tenant,

14. I would like to flag apartments that I am interested in renting so I can view them again later.
15. I would like to find apartments based on a set of variables (# of rooms, cost, location, etc.)
16. I would like to browse apartment listings.
17. I would like to read other tenants reviews to decide if I want to rent

As a tenant,

18. I would like to be able to have easy way of subleasing my apartment
19. I would like to rate my apartment when I move out, making it easier for potential tenants to browse listings.
20. As a tenant, I would like to upload pictures of my apartment.

As a landlord,

21. I would like to easily list available apartments/houses for potential tenants to see.
22. I would like to update listings to reflect apartments that are either available or not.

<u>Communication between users:</u>

As a potential tenant,

23. I would like to schedule apartment tours easily

As a tenant,

24. I would like to communicate with my landlord.
25. As a tenant, I would like to be able to communicate with other tenants in the same apartment structure, using an in-app messaging service.
26. As a tenant, I would like to be able to opt into or out of communicating with potential tenants about my experience.
27. I would like to request maintenance (General Issues).
28. I would like to request emergency services that would have a fast response time.
29. I would like to be able to document any damage or issues concerning my apartment to solve any disputes immediately.

As a landlord,

30. I would like to easily communicate with my tenants.
31. I would like to access contractors from the app, in order to take care of maintenance requests. [If time permits]
32. I would to be able to contact my maintenance workers and update them on their current assignment and give new assignments. [If time permits]

As a service provider,

33. I would like to exchange messages with my employer (landowner). [If time permits]

<u>**Customer services:**</u>

As a potential tenant,

34. I would like to be able to find potential roommates.
35. I would like to view up to date pictures of the apartment

As a tenant,

36. I would like to be able to pay my rent.

As a landlord,

37. I would like to have a checklist for tenants to know what they need to do before they move out.

Non-Functional Requirements:

As a client,

- The app should be uncluttered and easy to navigate
- Due to personal information stored within the accounts, we must ensure the security of user information
- This app must be compatible with Android and iOS mobile operating systems so that any user can access the application.

- The interface must be very user-friendly and fast, in keeping with the goal of streamlining the process as much as possible.
- Need to have separate backend (probably hosted on AWS), to host all the documents/messages
- Since each type of user has different privileges, we will ensure that they only get access to the features accessible to that user
- The app must be able to handle a user account for each person involved in the building for each building
- The app must be able to have every apartment listing without having downtime
- The app must be able to handle 100 concurrent users

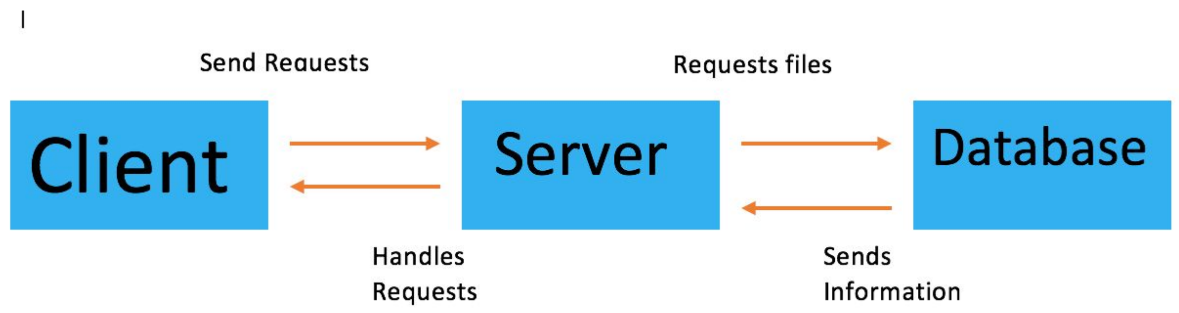
Design Outline

This project will be an application that allows tenants and landlords to allow easy communication, handle bill payments, and streamline move out processes. This application will also create an easy method for landlords to advertise open properties and for potential renters to find and view properties they are interested in. To accomplish this we will be using the Client-Server Model with one backend server hosting all the different user interaction. The server will implement the ModelViewController(MVC) pattern. The server will access a database that which host apartments listings, user information, and allows for varying access to this information depending upon the user's account privileges.

1. Client
    a. The client will be where user interaction takes place
    b. The client will show features of the user, which features will depend on the type of user that is using client
2. Server
    a. The server will handle all communication between the client application and the database
    b.
3. Database
    a. Database will store apartment listings and user information regarding their current rentals or rental inquires
    b. The majority of data will be info about apartments and pictures of each one

High Level Overview of the System

This project will have many clients connected to the server at the same time. The clients will send requests through the server when they login, search apartments, or communicated with other users. The server will send that data request to the database which will retrieve the data and return it to the data. The data will be displayed out to the user.



Design Issues

Non-functional Issues

1. How are we going to implement the front end services?
    a. **Solution 1:** React Native
        i. Explanation: React Native allows us to port the app to different softwares (Android, iOS, Web page) without completely changing the code each time.
    b. Solution 2: Programming for Android, then porting to iOS
        i. Members of our group are more familiar with Java than with Swift, so we would write the app in Android first, then convert it to be iOS compatible.
2. How are we going to implement the back end services?
    a. **Solution 1**: AWS

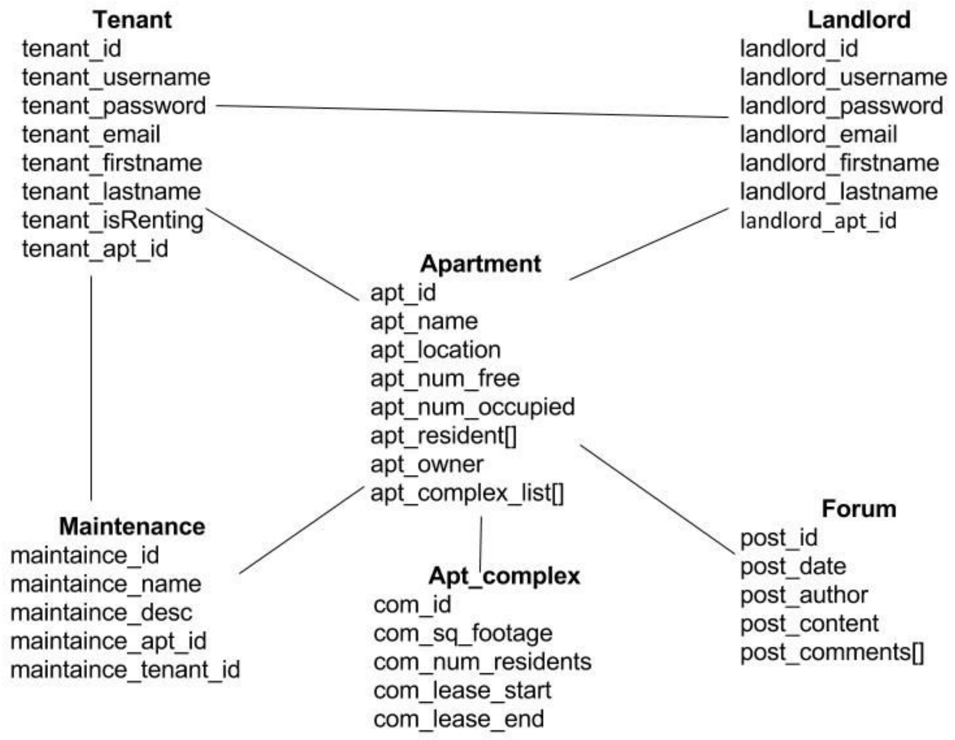
   b. Solution 2: Azure
   c. Solution 3: Private cloud providers
      i. We will use AWS, because we're more familiar with it than with Azure. It provides us with a versatile, cheap way to host the backend server for our app.
3. How can we keep the information current?
   a. **Solution 1:** Refreshing the information at a set interval
   b. **Solution 2:** Refresh the data when important changes are made
   c. Solution 3: Refresh the data when the user specifies
      i. For the most part, we will refresh the data when important changes are made. For example, apartment information only needs to be refreshed when the landlord or current tenant makes some change to its information (availability, up-to-date photos). However, for some parts of the apps, like the message forums, we will refresh the information at a set interval. This will keep from overloading the server with requests; handling that is something that could be done, but might increase the amount of work we have to do by a large amount and might not be feasible.

Functional Issues

1. How do we determine what permissions different users get?
   a. **Solution 1**: Implement different user classes
   b. Solution 2: Separate the permissions from the user classes entirely (make them their own class)
      i. We will use solution 1; since there aren't that many users, it'll be relatively simple to assign their permissions based on who they are. If we had a case where we had more types of users, then it would make more sense to create a permissions system.
2. How are we going to display apartment listings?
   a. **Solution 1**: list them individually according to distance
   b. **Solution 2**: show the locations of the apartments on a map, centered on the user's location
      i. We actually hope to implement both of these features in the app, but listing them as individual elements is our first priority. The map would be harder to implement, making it a secondary priority.
3. How do we display the status of an apartment?
   a. Solution 1: Clicking on the apartment in the list should bring up a screen displaying basic information about the apartment, including its availability

b. **Solution 2**: Display the availability of the apartment on the listing screen
    i. We will display the availability of the apartment on the listing screen, because it will make it easier for users to browse listings. Including it only on the info screen for one specific apartment makes it unnecessarily cumbersome for the user.

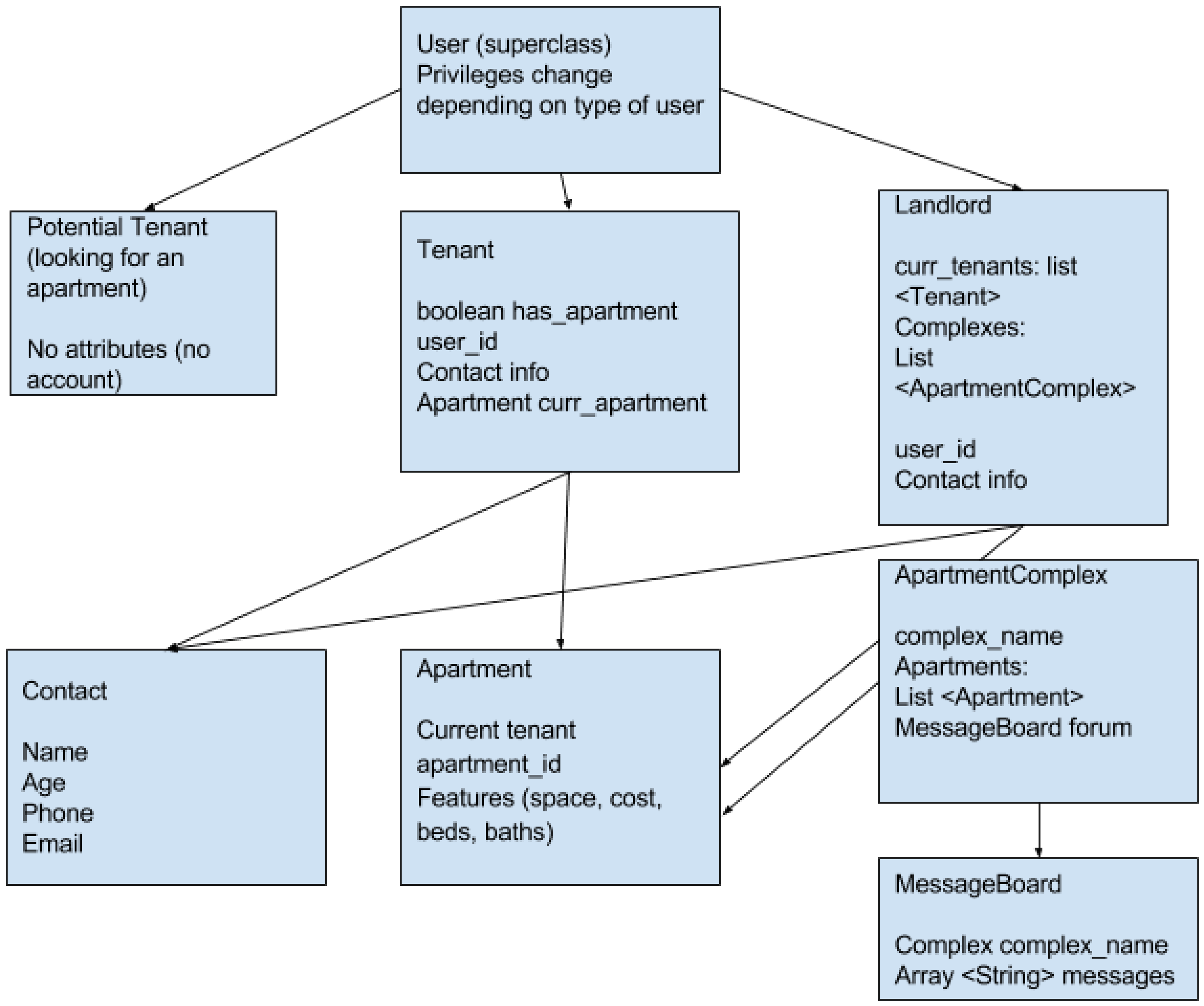
Database Schema



This diagram shows the type of information being stored in the database and the relationships that information has to other schemas. The two types of users, whose data will be saved, will each connect to an apartment object. The apartment object will have information about the landlord who owns that object as well as all the tenants that will be associated to that apartment. The apartment object will be connected to an Apt_complex object which is the individual units inside of an apartment. For instance, if there is a building with 5 different units available for rent, the building itself will be an Apartment object will the individual units will be an Apt_complex object. A Maintenance object will be connected to the tenant filing the complaint as well as the apartment associated with the complain. Finally the Forum will be connected to the apartment object that the forum is related to as well as the tenant that is posting onto the forum.

Design Details

The entirety of our app will be intertwined. In the app, we will have three types of users: tenant, landlord, and guest. The landlord will be the owner of the building(s), and the tenants are the people who live in that building. A guest is a user, who is not required to create an account, but is still able to view listings based on location. The tenant-landlord relationship will be simplified by our app, as we allow easier methods handling the logistics such as paying bills, requesting maintenance, and receiving reminders.



Explanation of the Data Classes

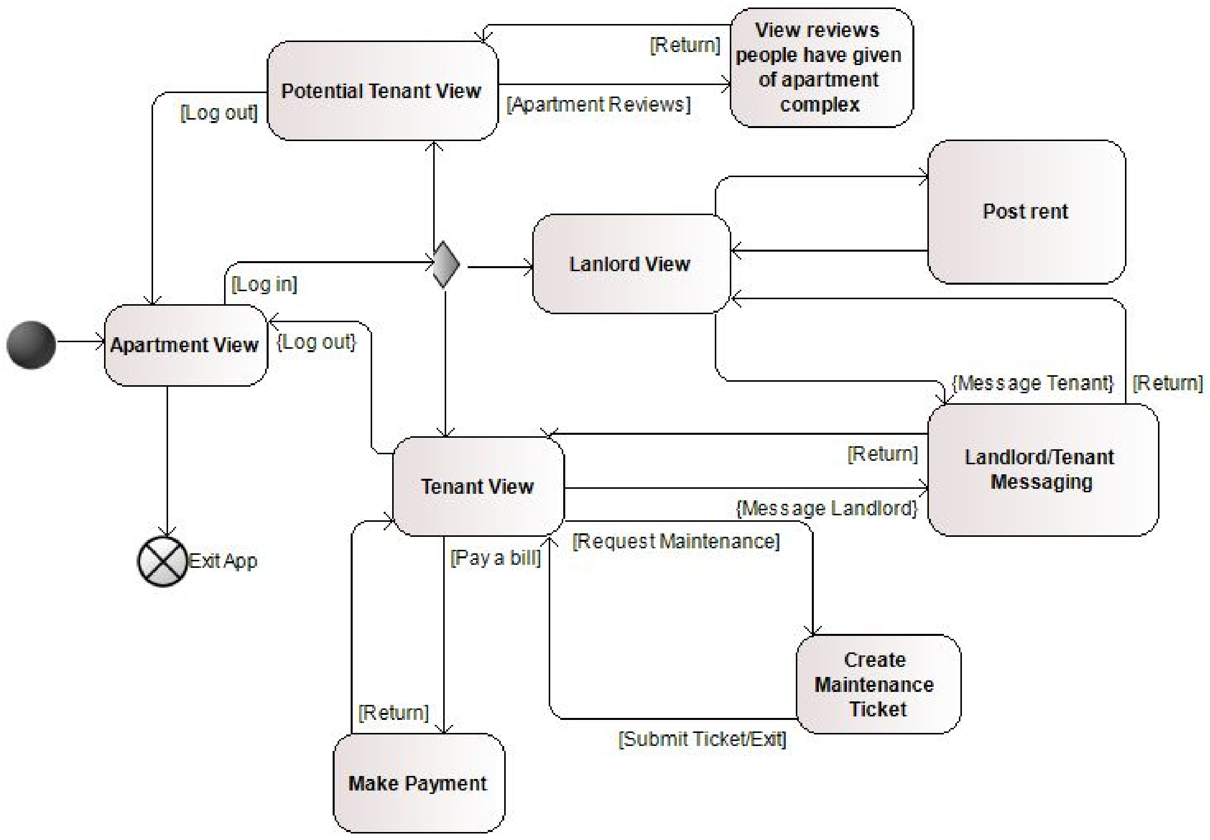
The class design above contains all the different classes needed to properly implement the objects required for this kind of app. The frontend of our app will use the classes as above, and the information for each object will be described in JSON format if necessary for our backend.

- Contact
  - Contains a set of strings describing basic contact information for a user
    - Name
    - Age
    - Phone Number
    - Email
- Potential Tenant
  - Used for a guest user
  - Only privilege are browsing apartment listings nearby
  - To interact with tenants/landlords, the app user must make an account, thereby upgrading to Tenant privileges
- Tenant
  - Describing a user that has an account in the app
  - Has a Contact variable describing contact information for the user, that other users can access if necessary
  - Boolean variable saying whether they have an apartment at the moment
  - User ID used in app specific operations (adding/removing tenants, notifications)
- Landlord
  - Someone that owns apartments
  - Has a list of available apartments
  - Has a list of tenants that are currently leasing an apartment they own
  - User ID used in app specific operations (adding/removing tenants, notifications)
  - Has a Contact variable describing contact information for the user, that other users can access if necessary
- Apartment Complex
  - Contains a list of apartments contained in the complex
  - Contains a MessageBoard variable that describes the current discussions for the complex-specific forum
- MessageBoard
  - Contains an array of the messages that tenants of its complex have posted on the board

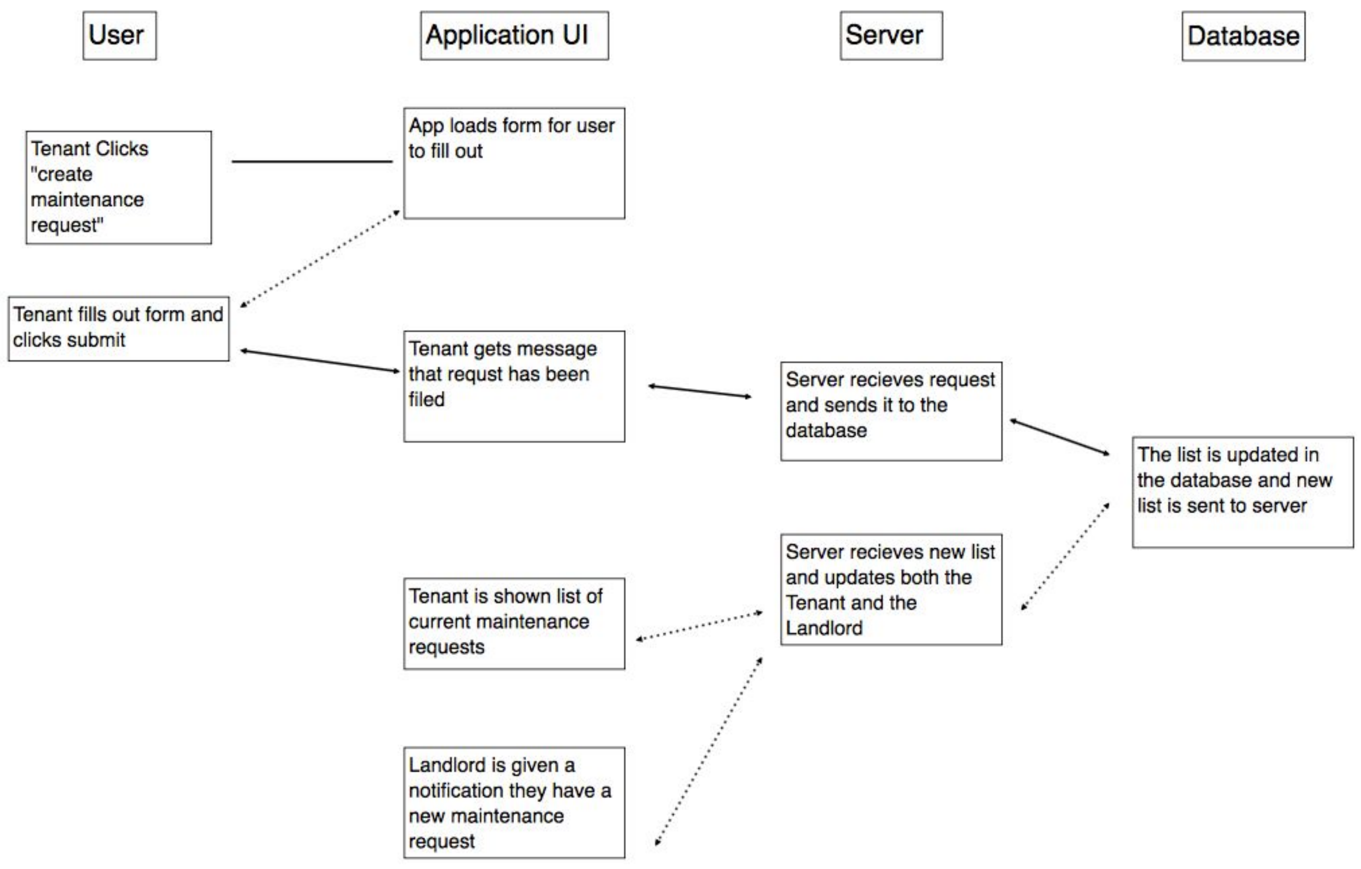
- Apartment
  - Class containing information about a specific apartment
  - Variable containing the user id of the current tenant
    - NULL if there is no tenant
  - Int to describe the apartment id, used to assign a user to the apartment
  - Features (separate variables describing things like cost, space, beds, baths)

States of the UI

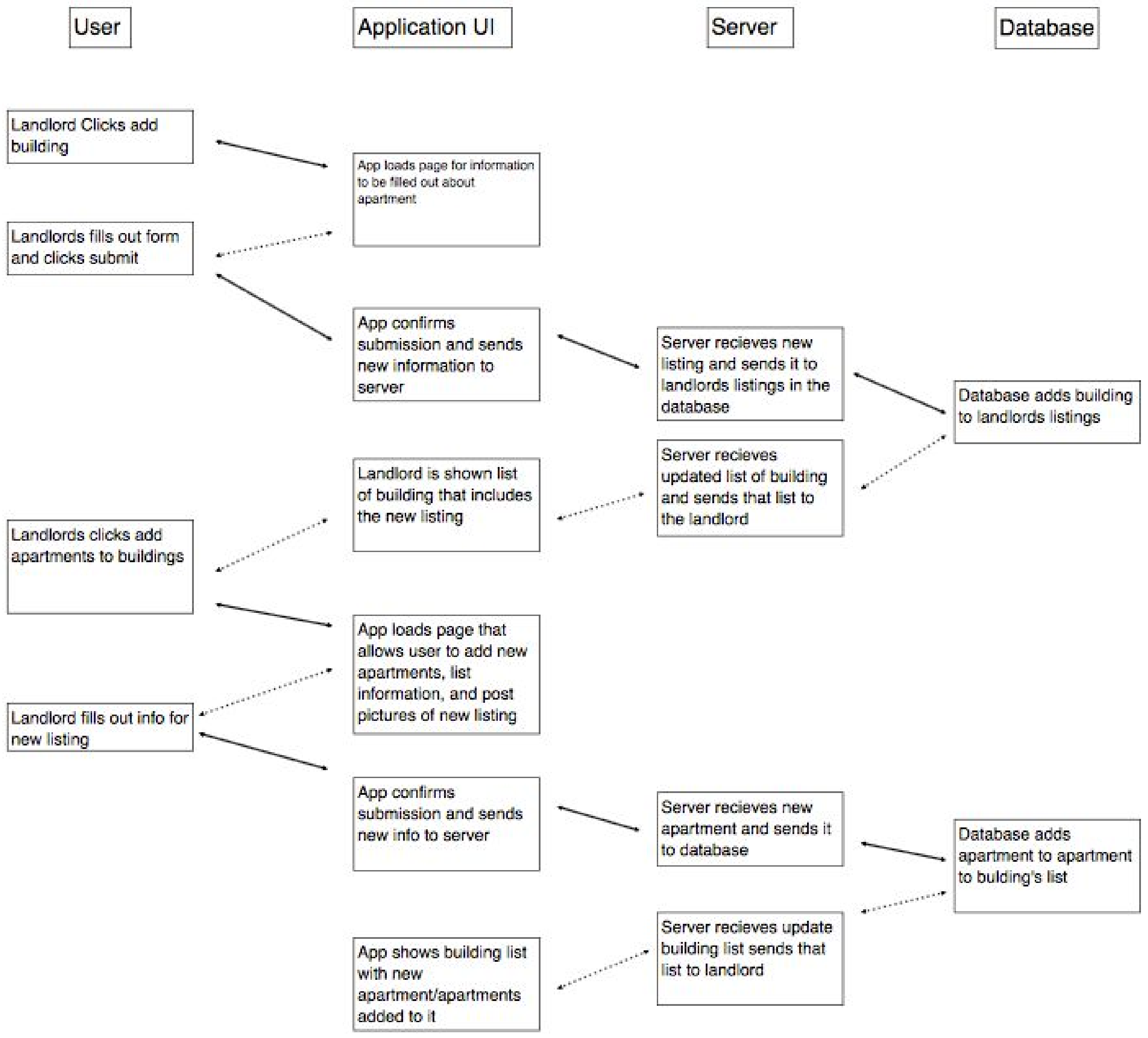
The application will start with a basic view of the available apartments that can be browsed without logging in. Upon logging in, the user is upgraded based on what type of user they are, being Potential Tenant, Tenant or Landlord, and can perform various actions based on what kind of user they are with some examples shown:



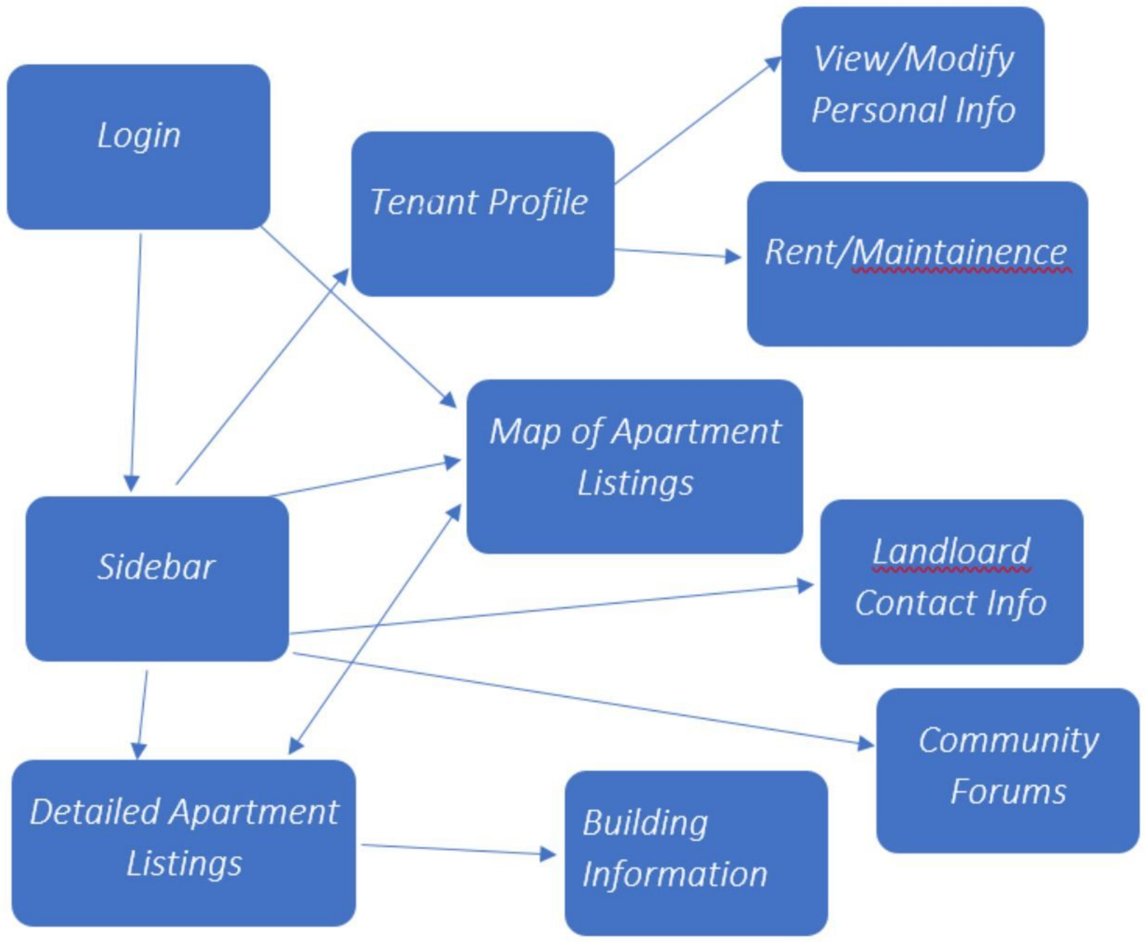
Sequence of Events When a User Creates a Maintenance Request


User
Application UI
Server
Database
Tenant Clicks "create maintenance request"
App loads form for user to fill out
Tenant fills out form and clicks submit
Tenant gets message that requst has been filed
Server recieves request and sends it to the database
The list is updated in the database and new list is sent to server
Server recieves new list and updates both the Tenant and the Landlord
Tenant is shown list of current maintenance requests
Landlord is given a notification they have a new maintenance request

Example: When a User Adds a New Building and then an Apartment to that Building

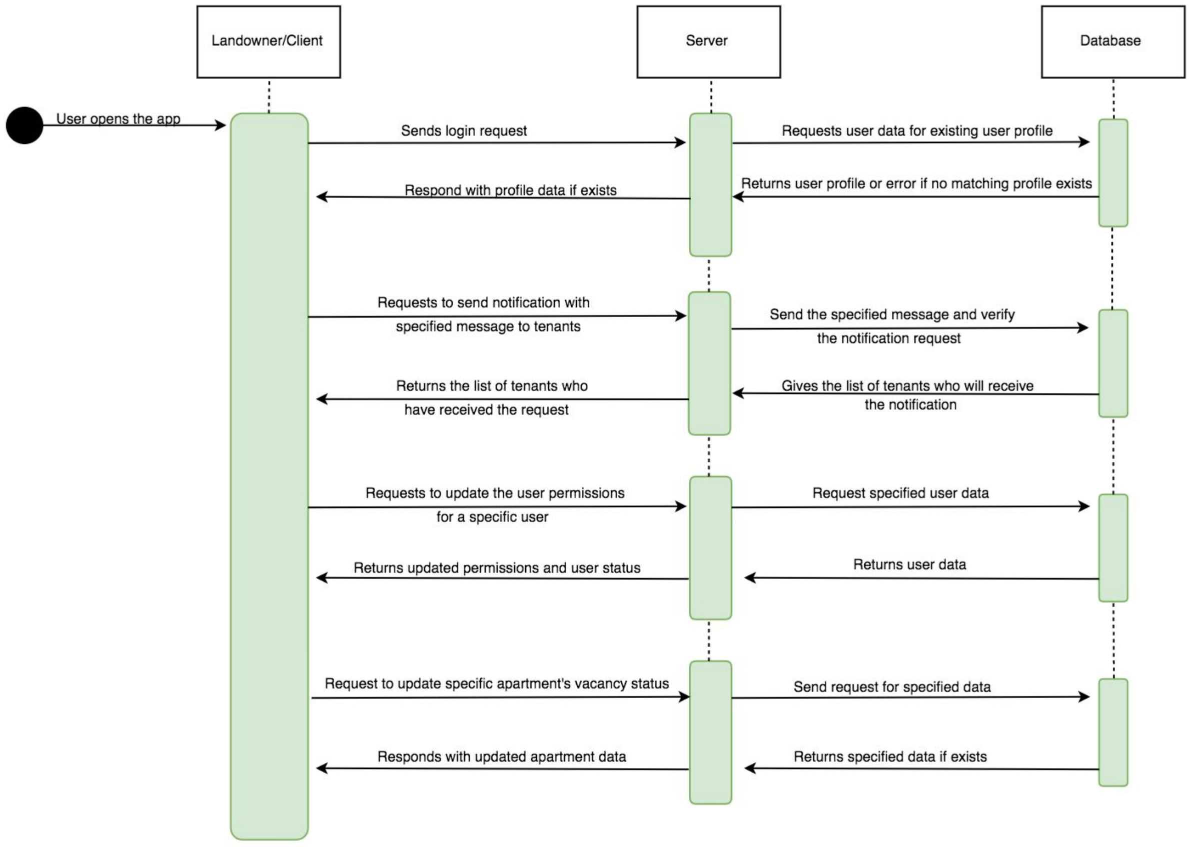

User
Application UI
Server
Database
Landlord Clicks add building
App loads page for information to be filled out about apartment
Landlords fills out form and clicks submit
App confirms submission and sends new information to server
Server recieves new listing and sends it to landlords listings in the database
Database adds building to landlords listings
Server recieves updated list of building and sends that list to the landlord
Landlord is shown list of building that includes the new listing
Landlords clicks add apartments to buildings
App loads page that allows user to add new apartments, list information, and post pictures of new listing
Landlord fills out info for new listing
App confirms submission and sends new info to server
Server recieves new apartment and sends it to database
Database adds apartment to apartment to bulding's list
Server recieves update building list sends that list to landlord
App shows building list with new apartment/apartments added to it

State Diagram from Tenant's Perspective



This diagram shows how the app will flow from a tenant's perspective. Initially, they have the option to login. After that, they will be shown a map of the area that they are currently in. They can navigate through the rest of the functional features of the app with the sidebar.

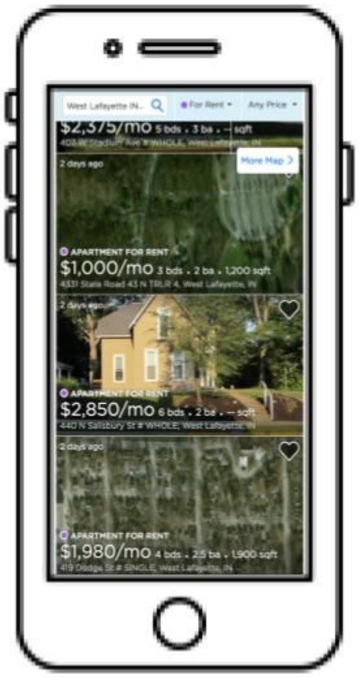
Landlord Client Sequence/Flow Diagram


Landowner/Client
Server
Database
User opens the app
Sends login request
Requests user data for existing user profile
Returns user profile or error if no matching profile exists
Respond with profile data if exists
Requests to send notification with specified message to tenants
Send the specified message and verify the notification request
Gives the list of tenants who will receive the notification
Returns the list of tenants who have received the request
Requests to update the user permissions for a specific user
Request specified user data
Returns user data
Returns updated permissions and user status
Request to update specific apartment's vacancy status
Send request for specified data
Returns specified data if exists
Responds with updated apartment data

This depicts the flow of the app from the perspective of a user with Landlord privileges. After logging in to the app, the landlord has several actions they can perform. Sending a notification, updating the status of tenants, and updating the status of apartments are some of the more important interactions that the landlord can perform.

User Interface Mockups

Listing Apartments - Image View



West Lafayette IN...
For Rent
Any Price

$2,375/mo 5 bds . 3 ba . -- sqft
403 W Stadium Ave # WHOLE, West Lafayette, IN

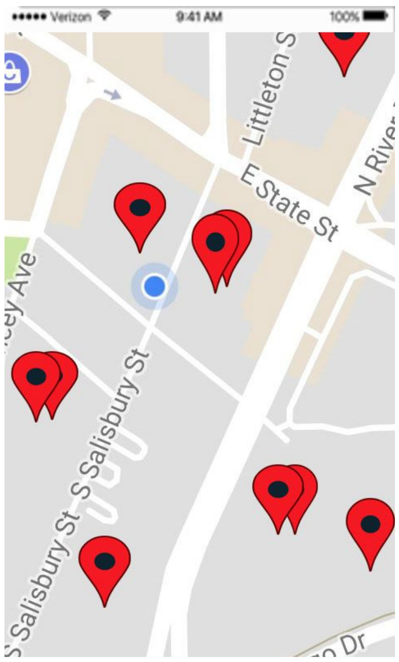
2 days ago
More Map
APARTMENT FOR RENT
$1,000/mo 3 bds . 2 ba . 1,200 sqft
4331 State Road 43 N TRLR 4, West Lafayette, IN

2 days ago
APARTMENT FOR RENT
$2,850/mo 6 bds . 2 ba . -- sqft
440 N Salisbury St # WHOLE, West Lafayette, IN

2 days ago
APARTMENT FOR RENT
$1,980/mo 4 bds . 2.5 ba . 1,900 sqft
419 Dodge St # SINGLE, West Lafayette, IN

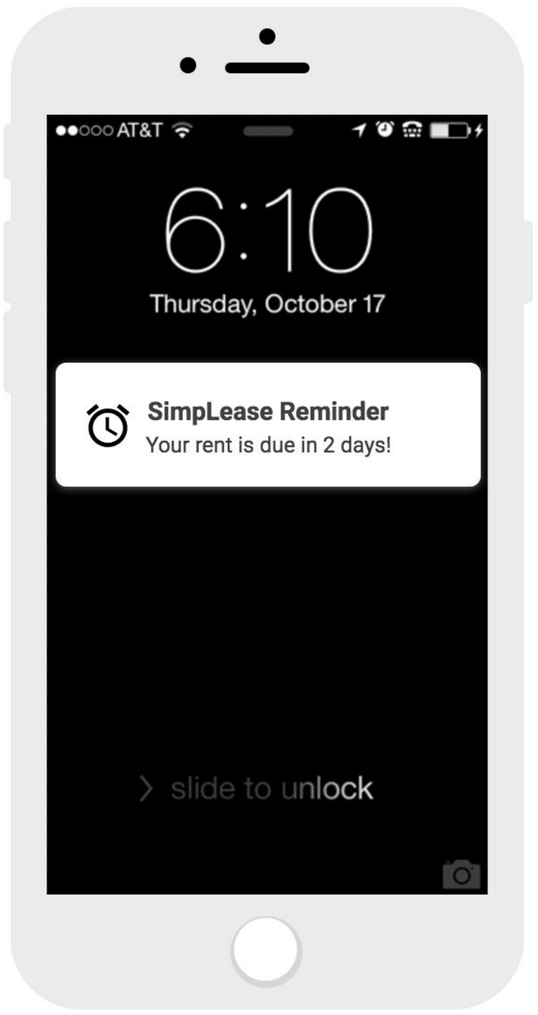
This is an example of what nearby listing will look like. The images of the apartments will be accompanied with information about price, size, address, and more. The user will be able to scroll through this image and look at all listings.

Listing Apartments - Map View



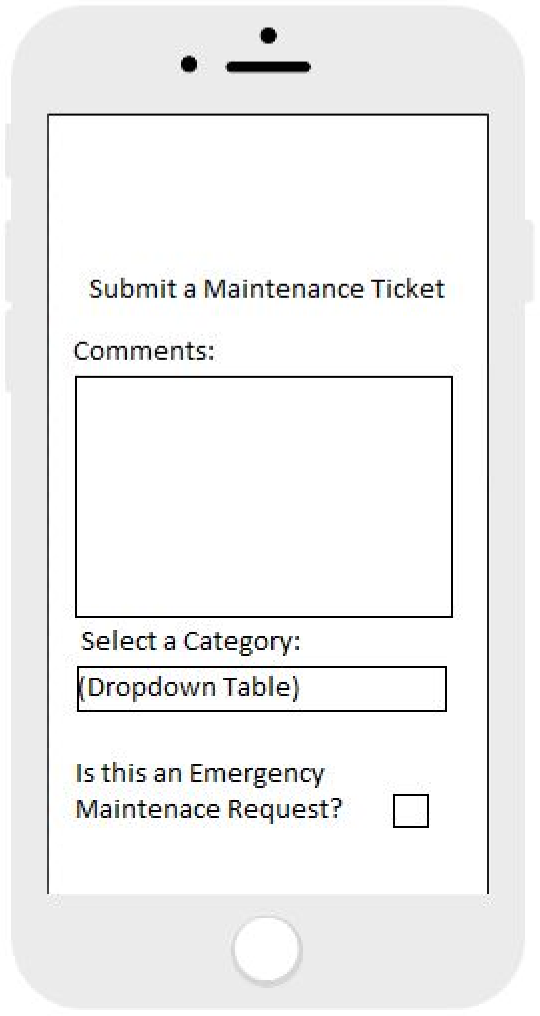
This is an example of what the map of nearby apartments would look like. We would implement the Google Maps api to list apartments based on the location of the user.

Notification Screen Mockup



This is an example of what it would look like for a tenant, when their landlord sends them a reminder.

Maintenance Screen Mockup



This is a design idea for submitting a maintenance request, from the perspective of a current tenant. The user can choose from the dropdown categories to specify what kind of issue they're having, and include additional comments to help the landlord in responding to the request. Checking the box for Emergency Request would send an emergency notification to the landlord, prompting them to respond as soon as possible.