Software Engineering

CSC 648/848 Fall 2021

Project: RoomMe

Section 02

Team 06

Raul-Izaiah Rodriguez - Team lead - rrodriguez2@mail.sfsu.edu

Jay Jaber - Backend engineer/Deployment

Kaung Htun - Backend engineer

Vy Ngo - Frontend engineer

Nael Yun - Frontend engineer

Tanishq Pradhan - Full-stack engineer/Github master

Vandit Malik - Database manager

Milestone 2: Documentation

Date: Oct 5th, 2021

Date Submitted	**Oct 5th, 2021**
Date Revised	

1: Functional Requirements (prioritized)

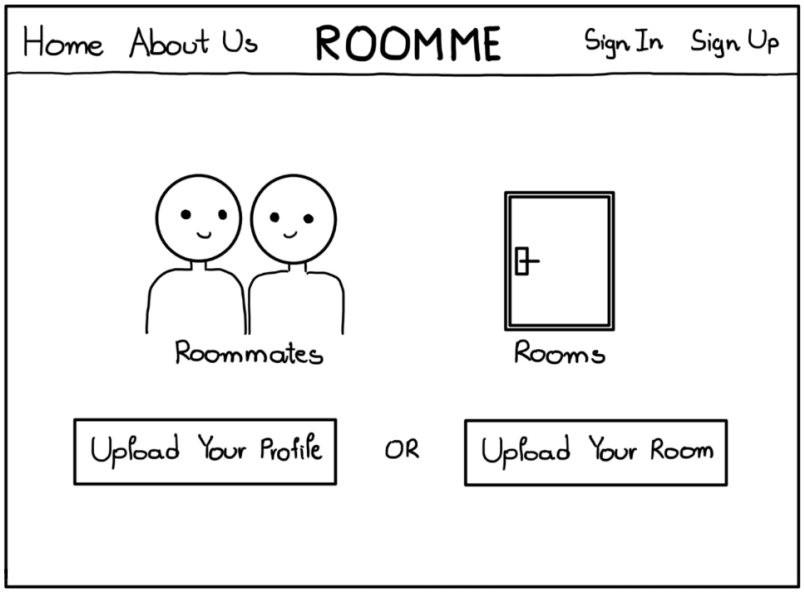
- **In terms of presentation, for easier review, please group all requirements first by priority i.e. list Priority e.g. Priority 1 requirements first, then Priority 2 etc. and within each priority section you should group them by actors (users, admin)**

1. Users shall be able to register for a personal account. **(1 – must have)**
2. Users shall be able to log into their account. **(1 – must have)**
14. Users shall be able to contact each other for inquiries. **(1 – must have)**
3. Users shall be able to list one or more units/rooms to be rented. **(1 – must have)**
6. Users shall be able edit posted listings. **(1 – must have)**
7. Users shall be able to delete listings. **(1 – must have)**
8. Users shall be able to search for rooms to rent. **(1 – must have)**
9. Users shall be able to rent a room alone or with roommates. **(1 – must have)**
10. Users shall be able to find new roommates. Users shall be able to post and search roommate listings to find others with similar interests. **(1 – must have)**
15. Users shall be able to search for rooms based on location. **(1 – must have)**
4. Users shall be able to limit their rooms maximum occupancy. **(2 – desired)**
5. Users shall be able to set rooms as vacant or occupied. **(2 – desired)**
18. Users shall be able to set the duration for their units to be rented. **(2 – desired)**
11. Users shall be able to filter for roommates by preference. These include pets, smoking, gender, age. **(2 – desired)**
12. Users shall be able to search for rooms based on amenities offered. These include laundry machines, kitchen, pool, gym, parking, furniture. **(2 – desired)**
13. Users shall be able to find rooms based on the number of tenants. **(2 – desired)**
16. Users shall be able to view rooms for rent on a map. **(2 – desired)**
17. Users shall be able to filter rooms by price. **(2 – desired)**
19. Users shall be able to filter rooms based on duration of stay. **(2 – desired)**
20. Users shall be able to bookmark rooms/ roommates they are interested in. **(3 – opportunistic)**

2: UI Mockups/Storyboards

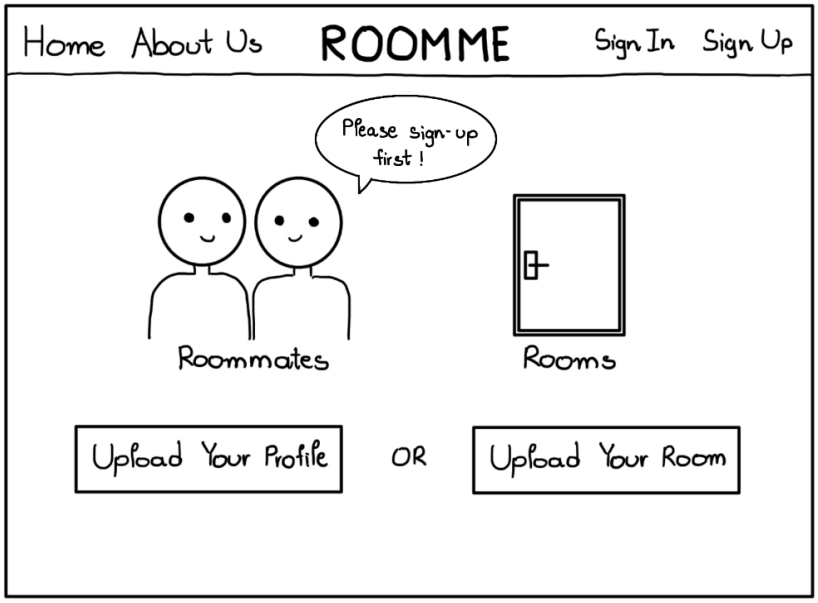
The user wants to post the availability of a room on our site. The user is prompted to login/signup on our website before posting. The user, after creating an account or logging in to the account, creates a listing of their room along with parameters attached to it.

Home Page

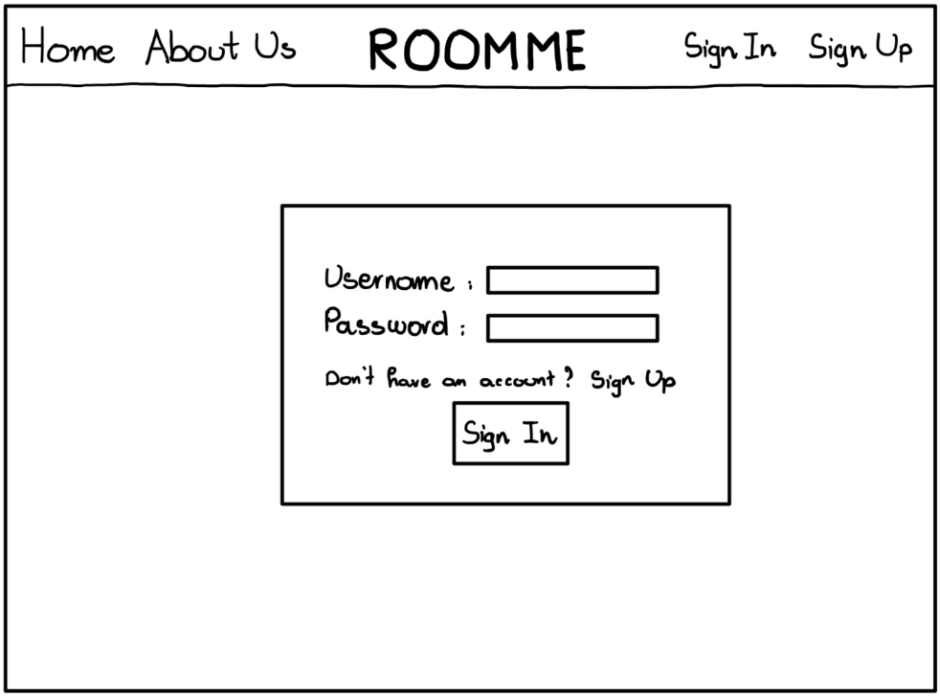


If the user tries to post content, a pop-up will appear and ask to sign up.

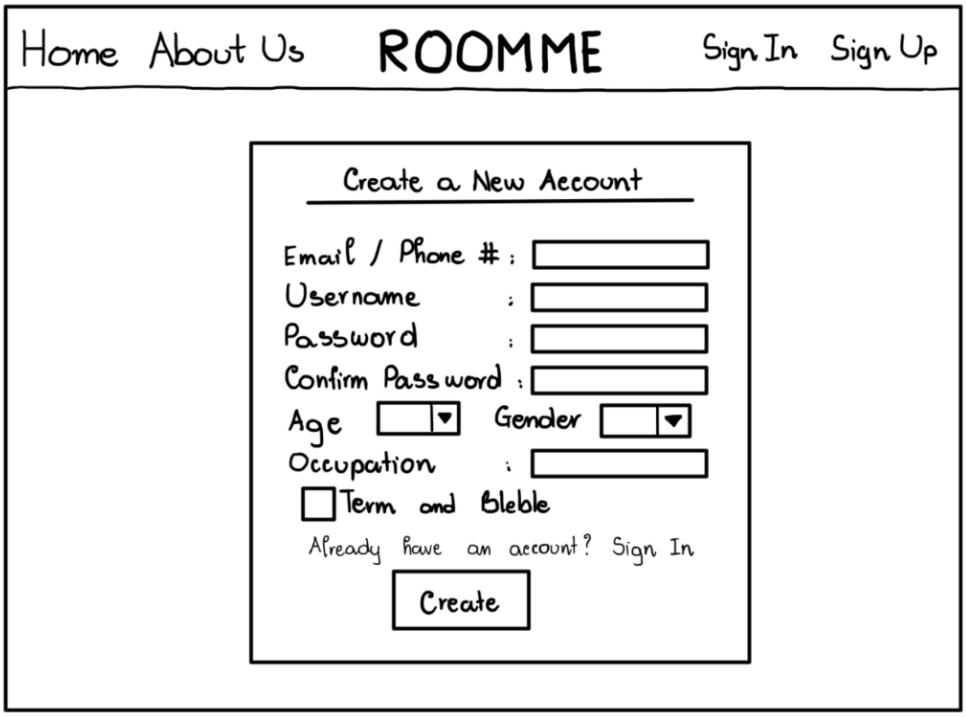
Home Page



Sign In Page

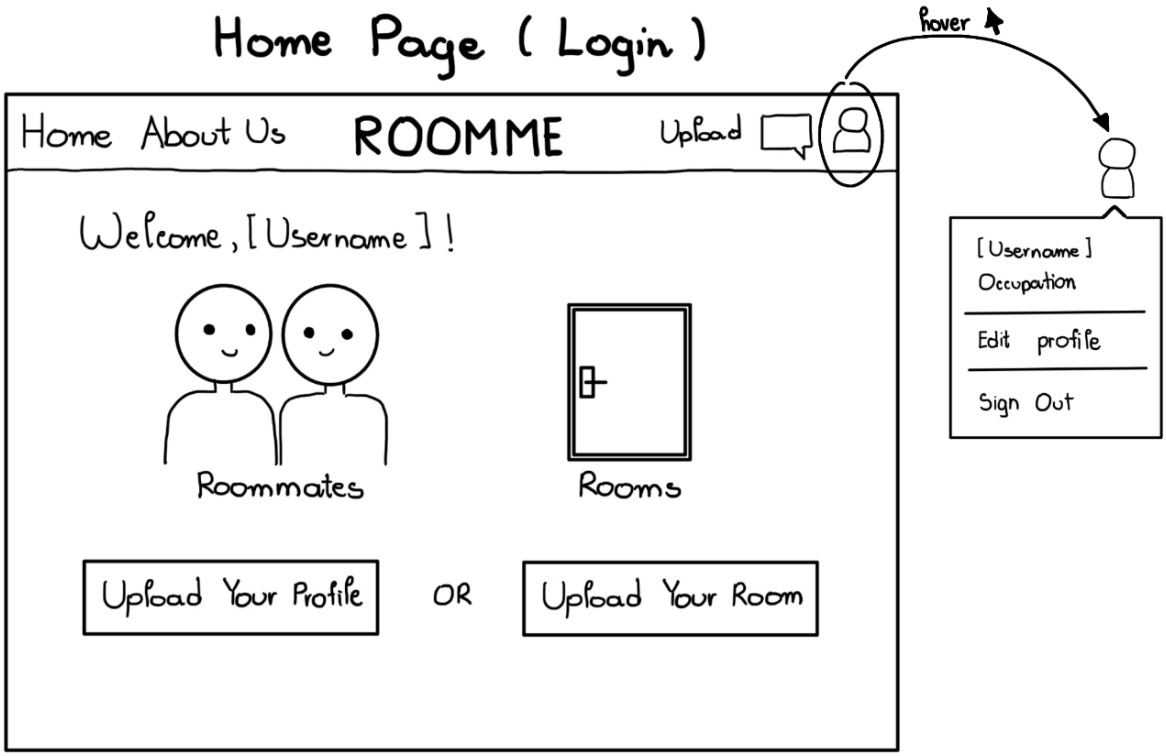


Sign Up Page



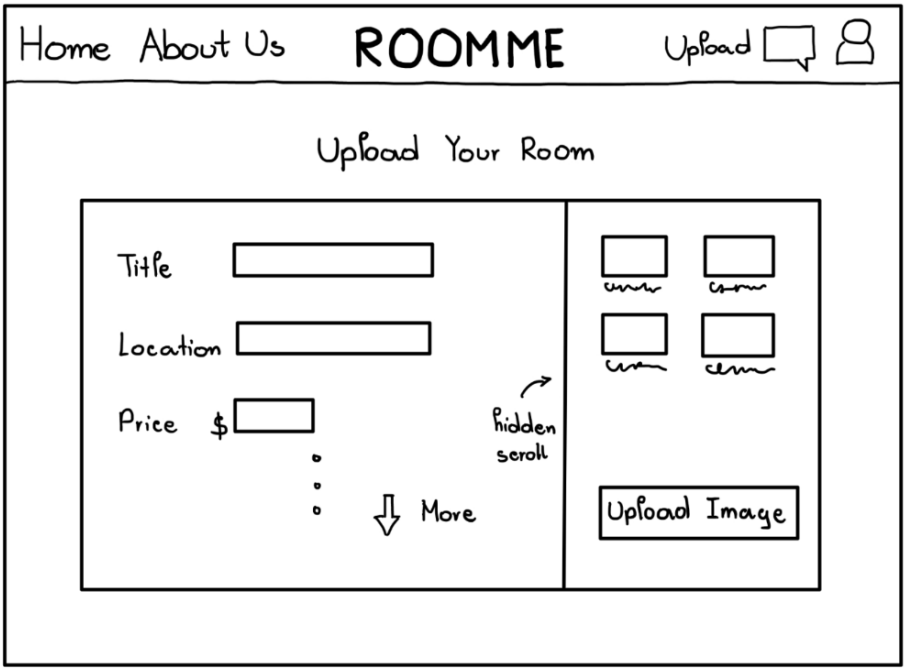
User needs to login or register first before posting.

Home Page (Login)



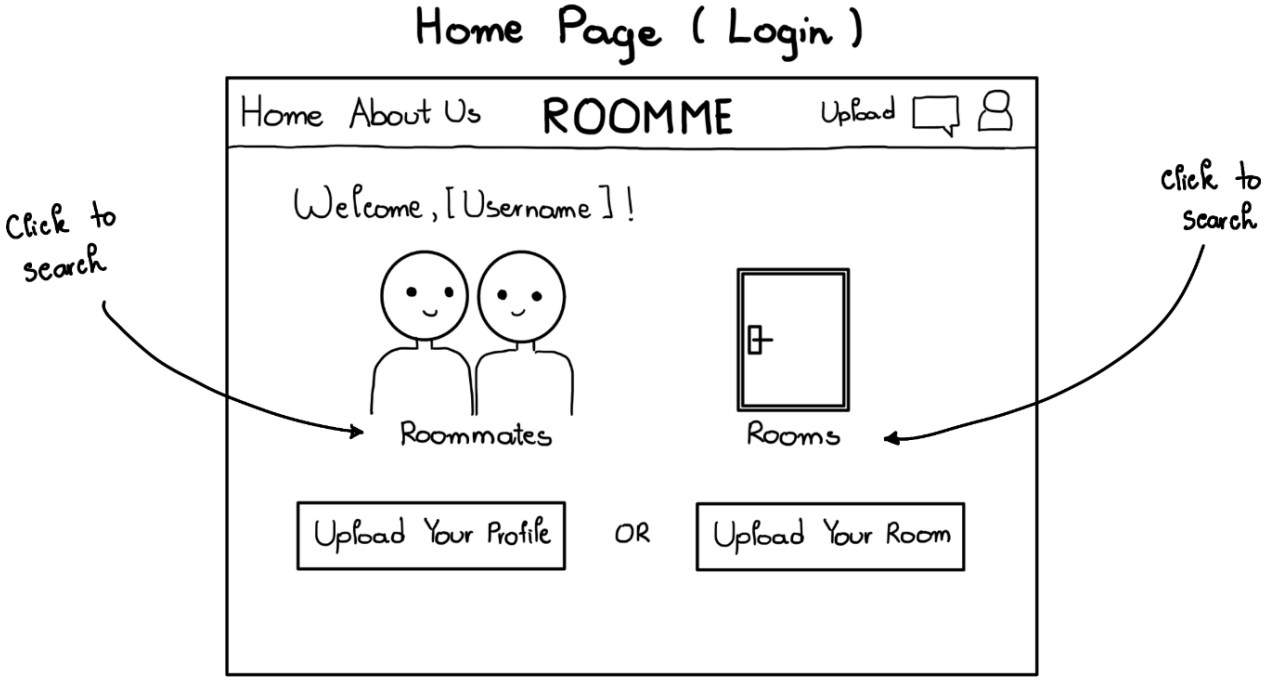
After logging in, a welcome message appears.

Upload Room Page

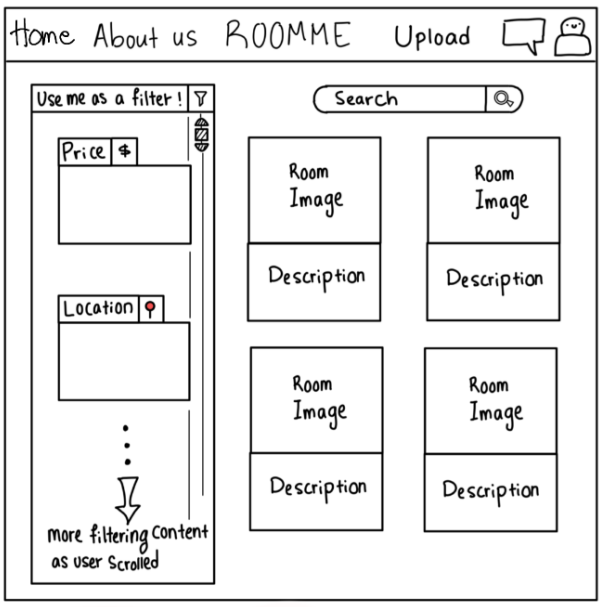


The uploading process for room content.

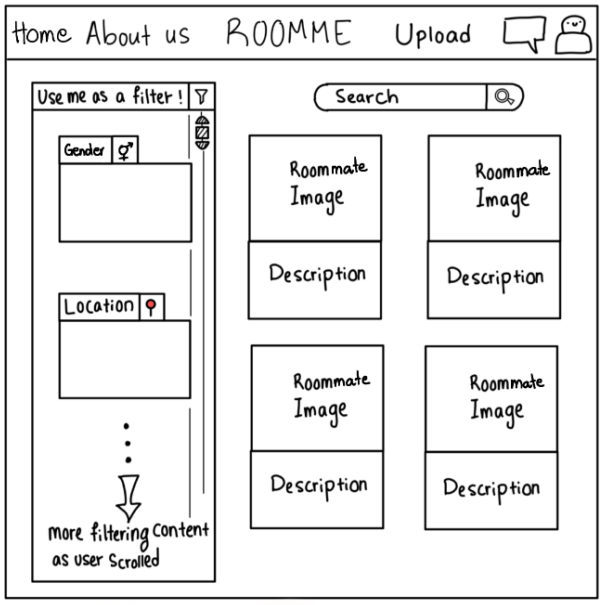
The user wants to search for a room. The user uses the filter feature to apply a filter to the searching parameters such as and not limited to location of the room, budget of the user, gender of roommate, amenities available, major of roommate, pet policy, etc. The user then reviews the list of search results.



The user can click one of the icons to search for roommates or room.

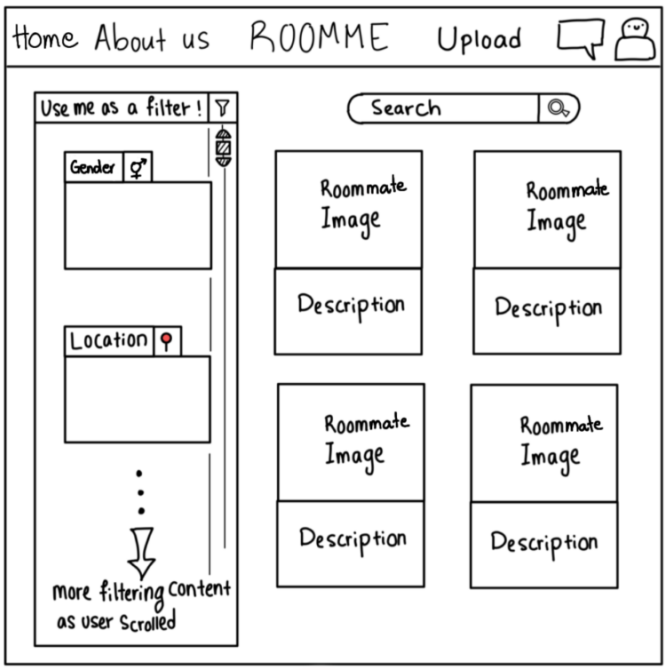


Room content will display if the user clicks "room"



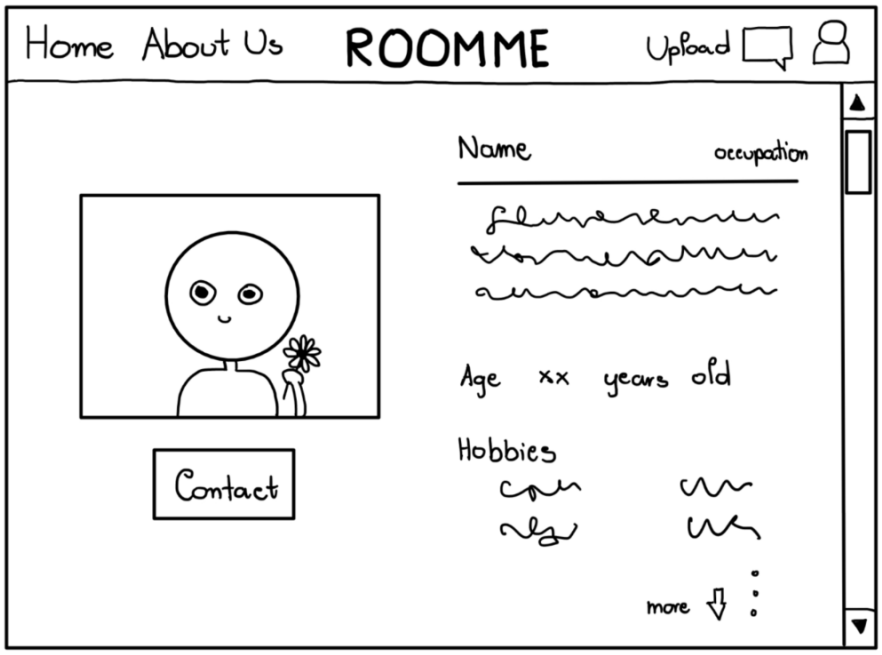
Roommate content will display if the user clicks "roommate"

The user wants to search for a roommate. The user can review the profiles of various users looking for a room on our website. The user after reviewing the profiles selects potential roommates and contacts them.



If the user clicks on any roommate profile, a detailed profile post will show.

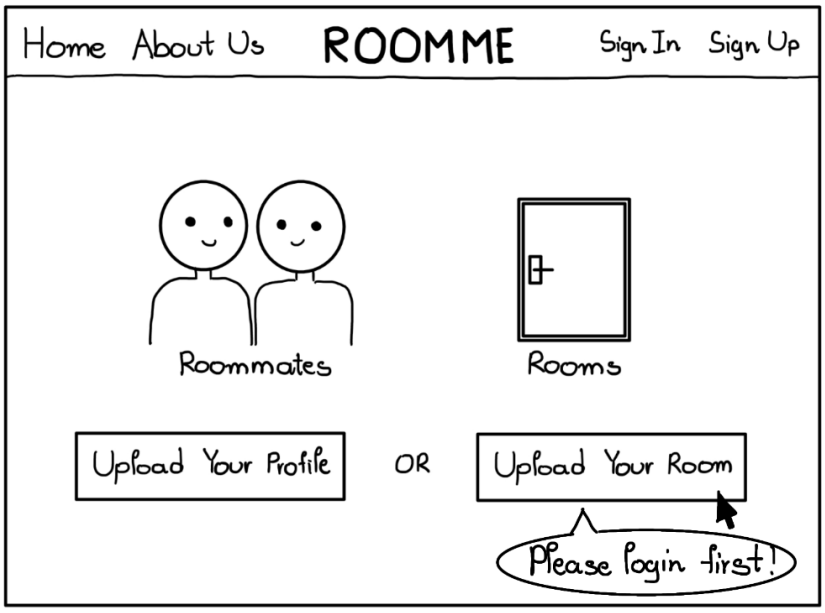
View Profile Page



If the user finds a potential roommate, then the user can contact them.

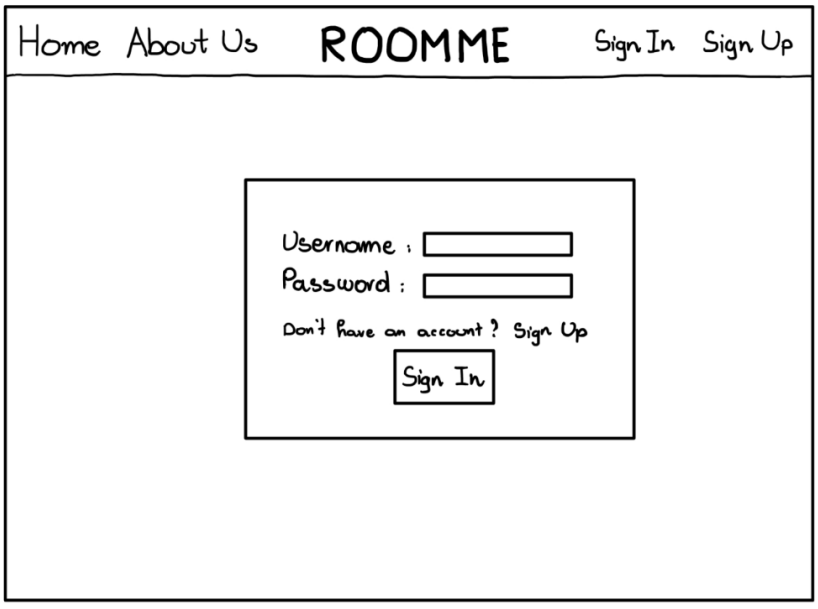
The user wants to create their own profile information page on our website. The user is prompted to login/signup on our website before doing so. Once they are logged in the user can type some information and upload an image of themselves. This profile is available for anyone to see.

Home Page



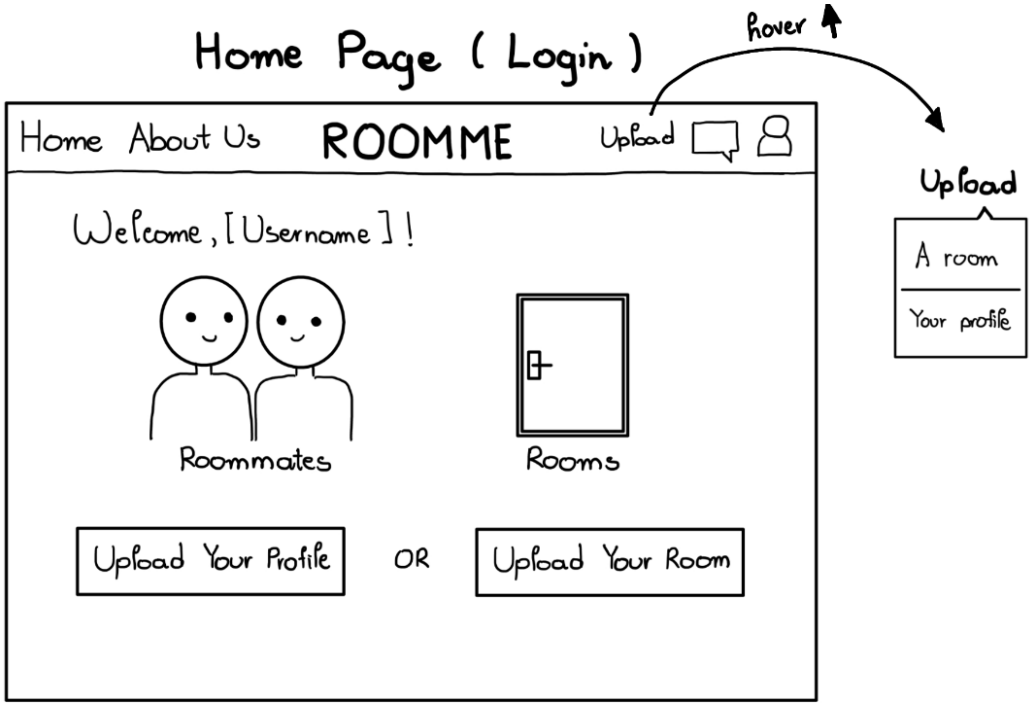
If the user tries to upload a room with logging in, a pop-up message will appear.

Sign In Page



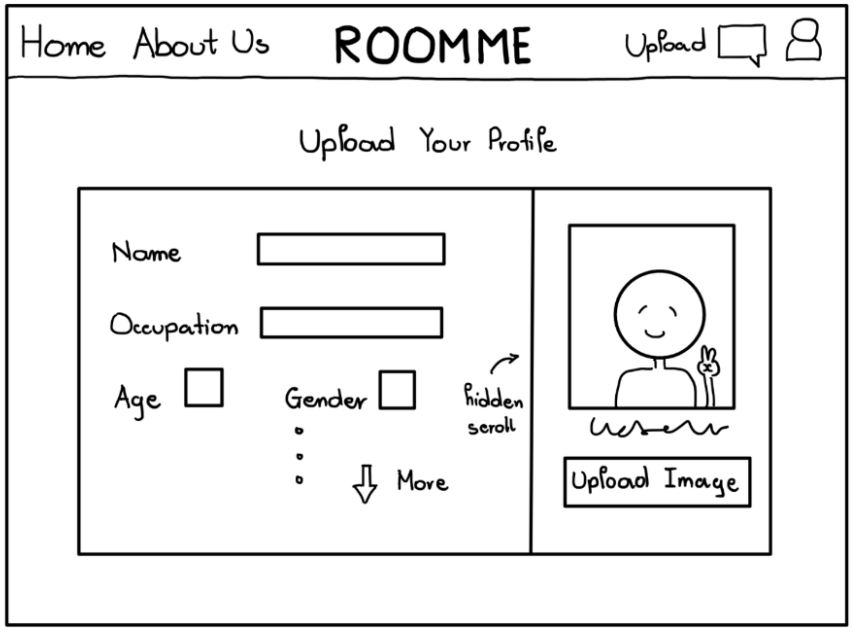
The user has to login first before they want to upload anything.

Home Page (Login)



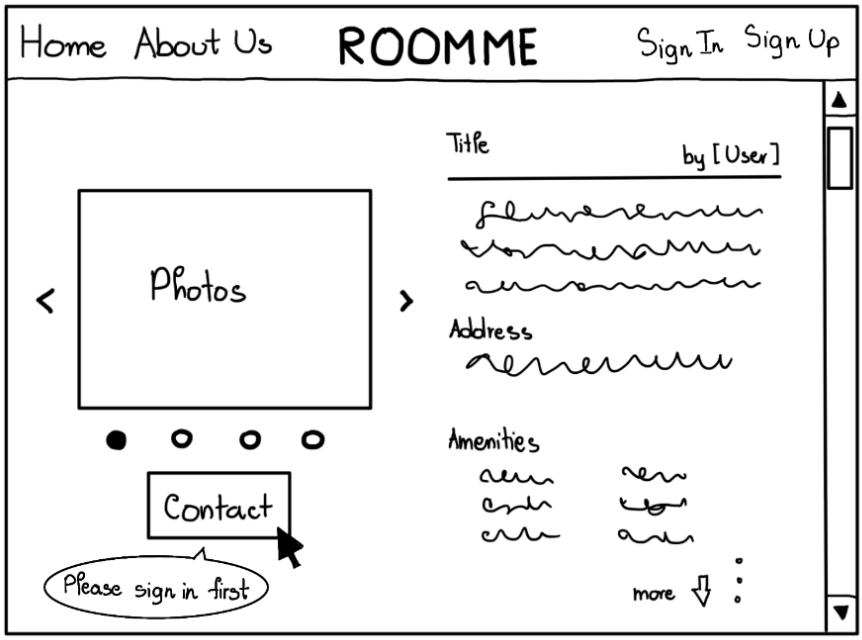
There are 2 ways for user to upload their roommate profile.

Upload Profile Page



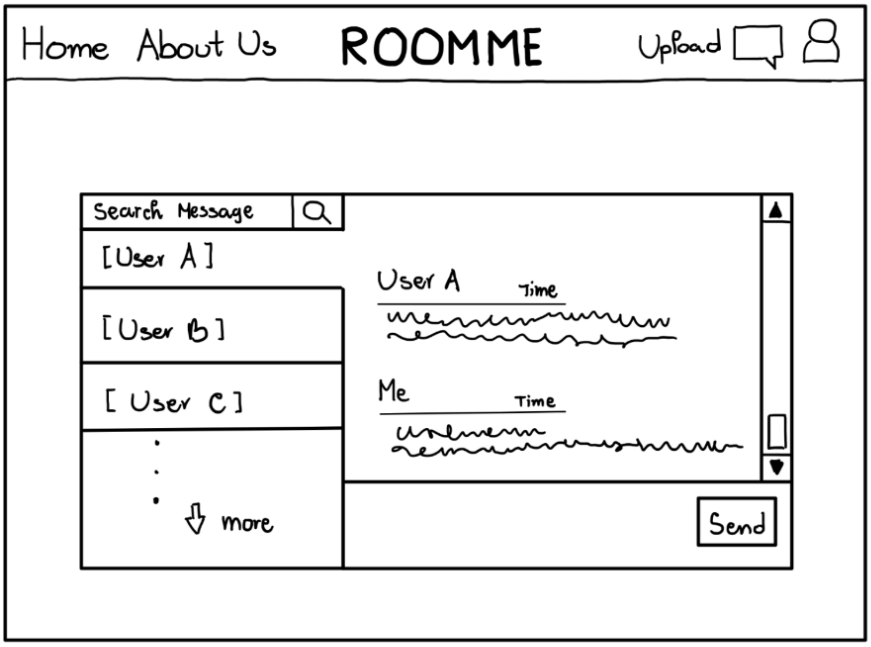
The user is looking for a room on our website. After viewing the different listings available the user selects one of them. The user is prompted to login/signup on our webpage before they can contact the tenant who listed the room. Once logged in the user can contact the tenant and can ask further questions to finalize the room.

View Room Page



A pop-up message will appear if the user tries to contact without logging in.

Message Page

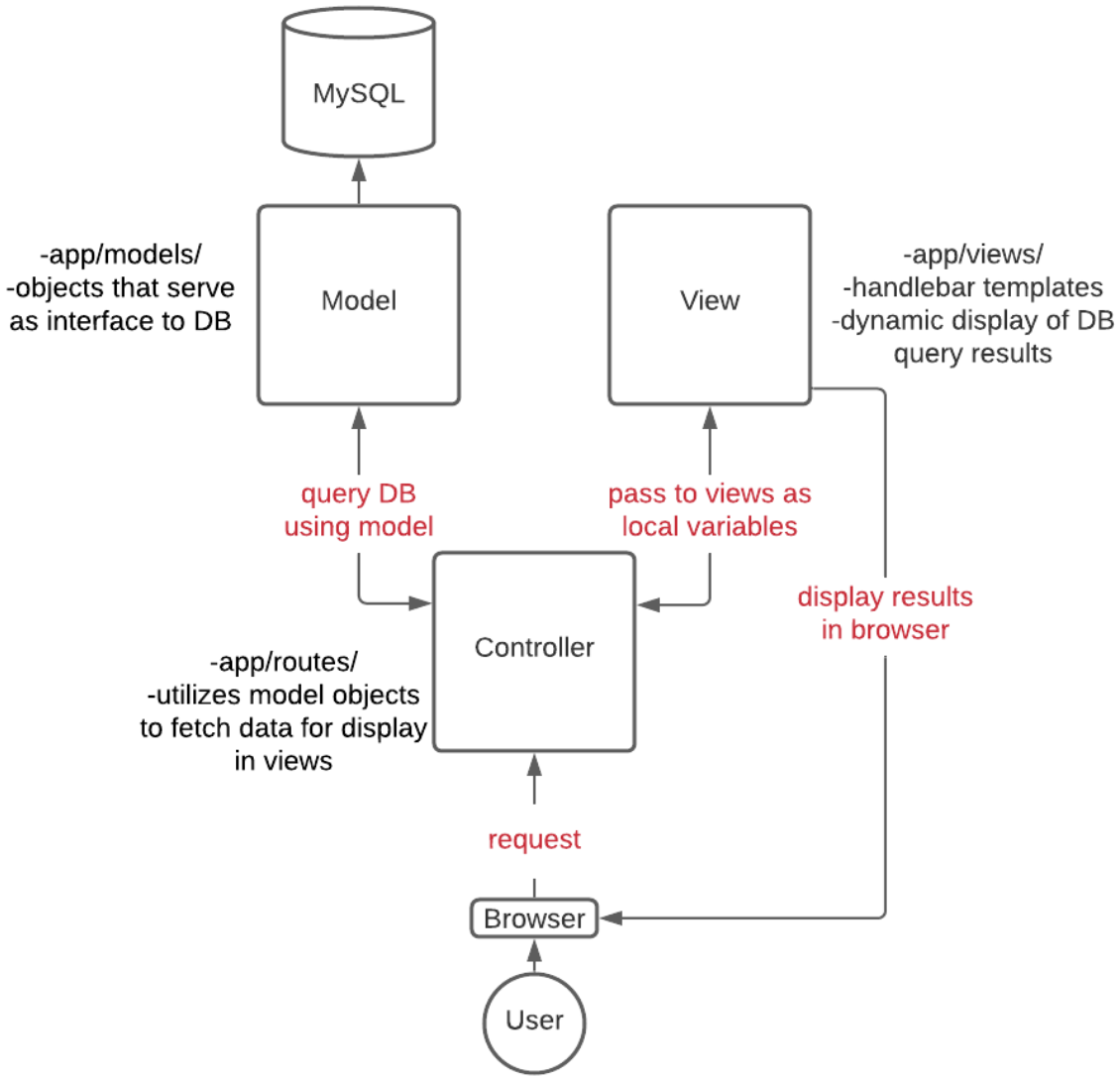


After logging in, the user will be redirect to their inbox

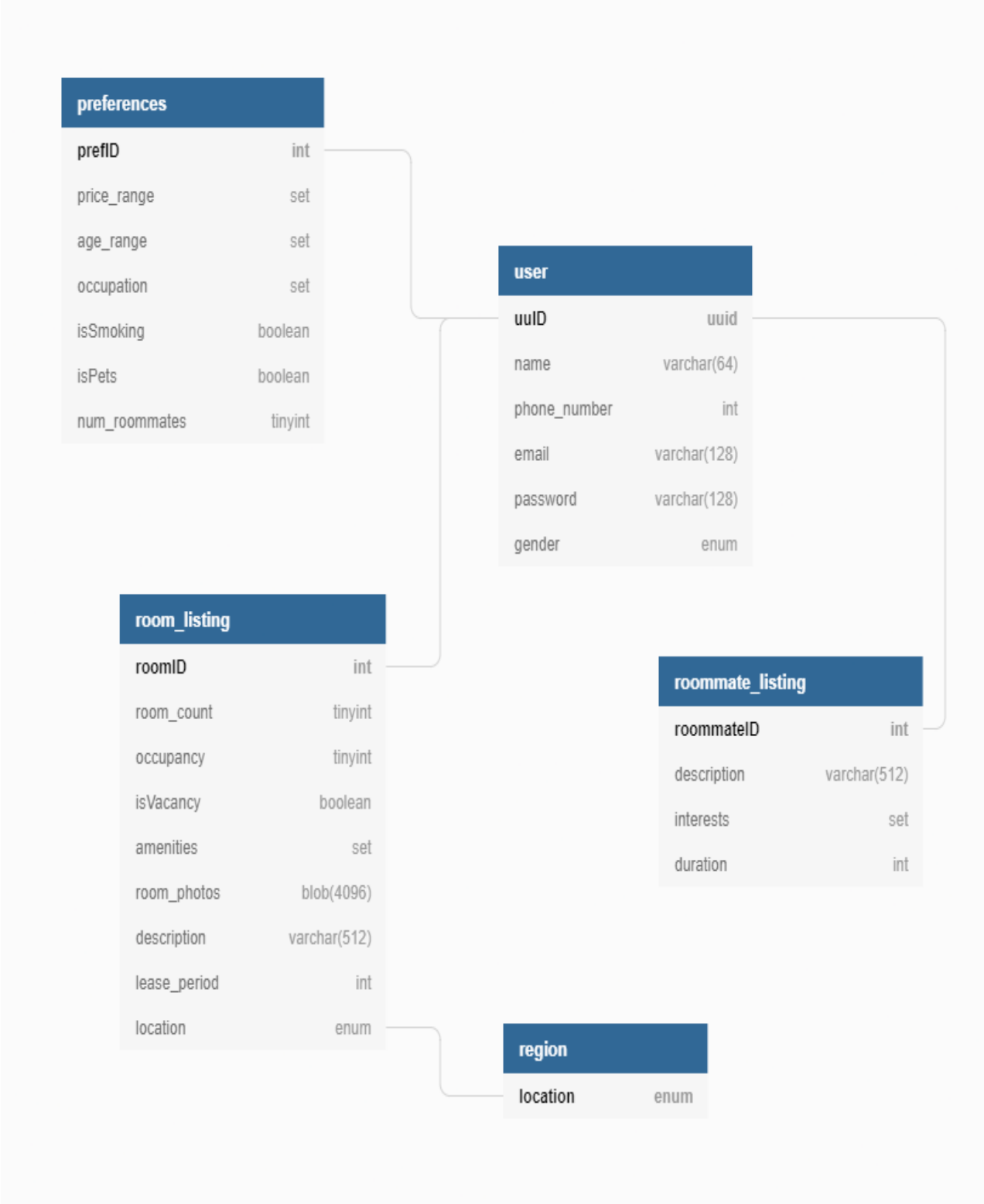
3: High Level Architecture, DB Organization

Architecture

- Application will employ the MVC design pattern.



Database Tables and Organization



Media Storage:

- Our group has elected to use Database BLOB's to store our images. We figured that we'd eliminate the need for node modules that handle file system photo storage, while keeping DB data all in one place.

Search/filter Implementation and Architecture

- Our platform will allow users to search in two ways, free text and parameterized/filter search. Most of the heavy lifting is done by SQL operators, in our case, we will utilize SQL LIKE and table joins to implement our functionality.

Free text search:

- Search process begins when the user enters their query in the free-text search bar. By default, results will include rooms and roommates with data members matching the query. Data is sent to the server via POST request from the browser.

- Once body data is received, the server will construct an appropriate query using SQL LIKE operator. Depending on the desired results, the query will be run against either:

    - Room Listing: Search against amenities, description, location
    - Roommate Listing: Search for matching keyword in description, interests

Parameter search:

- Search process begins from the user feed, where a section will be dedicated to search filters. These can be adjusted to manipulate the results shown to the user. These filters will include:

    - Dropdown filters - amenities(multiple select),
    - Location - Text input
    - Lease period- number input (1 month increments)
    - Max Price - number input
    - Min Price- number input

- Filters for search will be sent to the server in POST request, to be used in an SQL table join.

- Another filter will allow users to see only rooms or roommates matching the filters specified. Depending on the desired results, SQL table join will be run on one or both of the following:
    - Room Listing
    - Roommate Listing
    - SQL join can pull data from Room Listing or Roommates Listing tables, depending on the parameters sent in from the frontend

e.g.
```
SELECT user.name …., room.description ….,
FROM users
INNER JOIN room_listings
ON user.id = room.userID
WHERE ….
```

RoomMe API's

- Our backend will include a general CRUD api for all tables in our database. Will correspond to routes of the following format: /api/*

- Add/Delete/Edit/Search for Users.
    - Get all: GET /api/users
    - Get by id: GET /api/user/(:id)
    - Update by id: PUT /api/user/(:id)
    - Delete: DELETE /api/user/(:id)

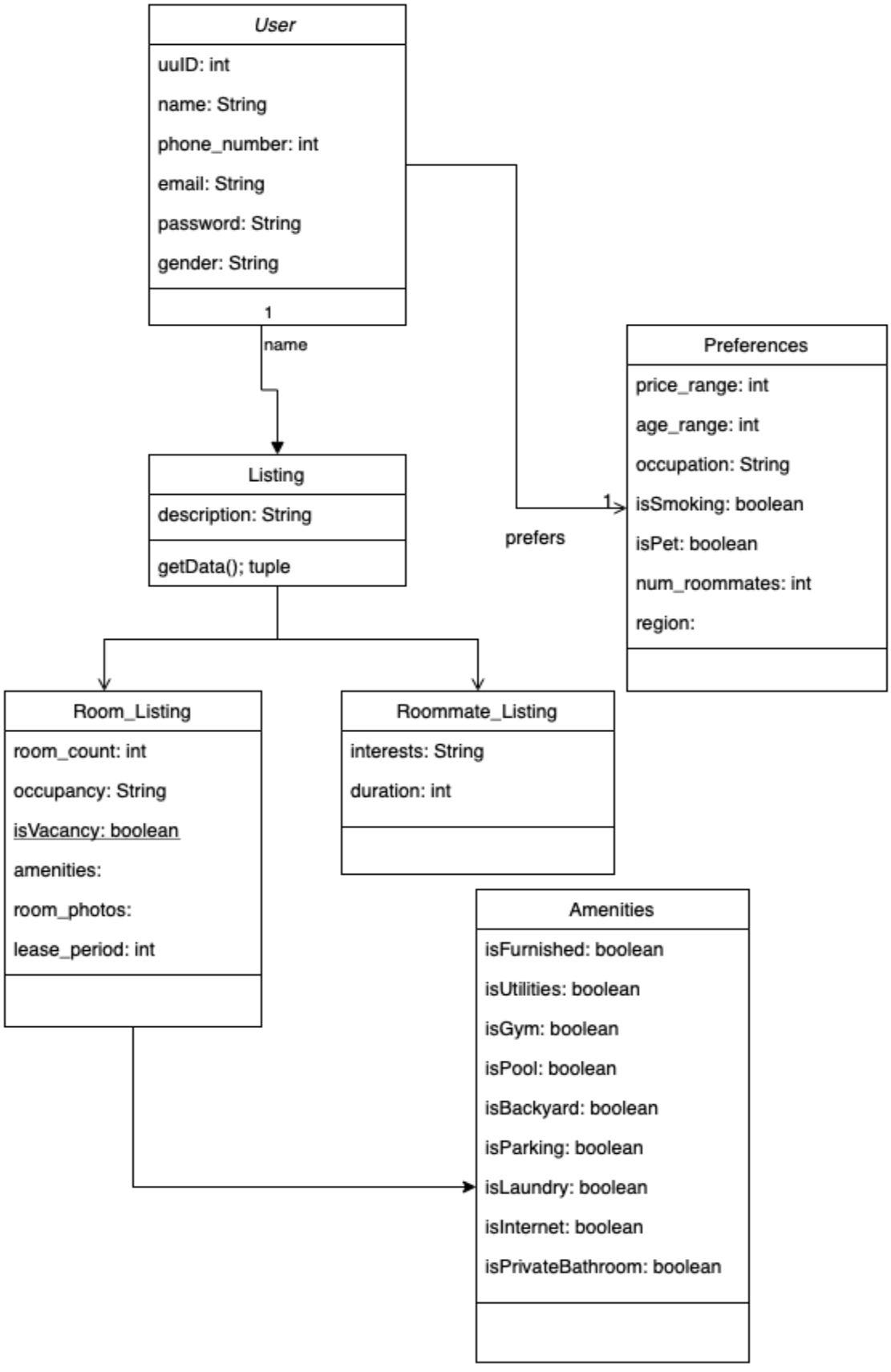
- Add/Delete/Edit/Search for Room Listings.
    - Get all: GET /api/listings
    - Get by id: GET /api/listing/(:id)
    - Update by id: PUT /api/listing/(:id)
    - Delete: DELETE /api/listing/(:id)

- Add/Delete/Edit/Search for Roommates Listings.
    - Get all: GET /api/roommate-listings
    - Get by id: GET /api/roommate-listing/(:id)
    - Update by id: PUT /api/roommate-listing/(:id)
    - Delete: DELETE /api/roommate-listing/(:id)

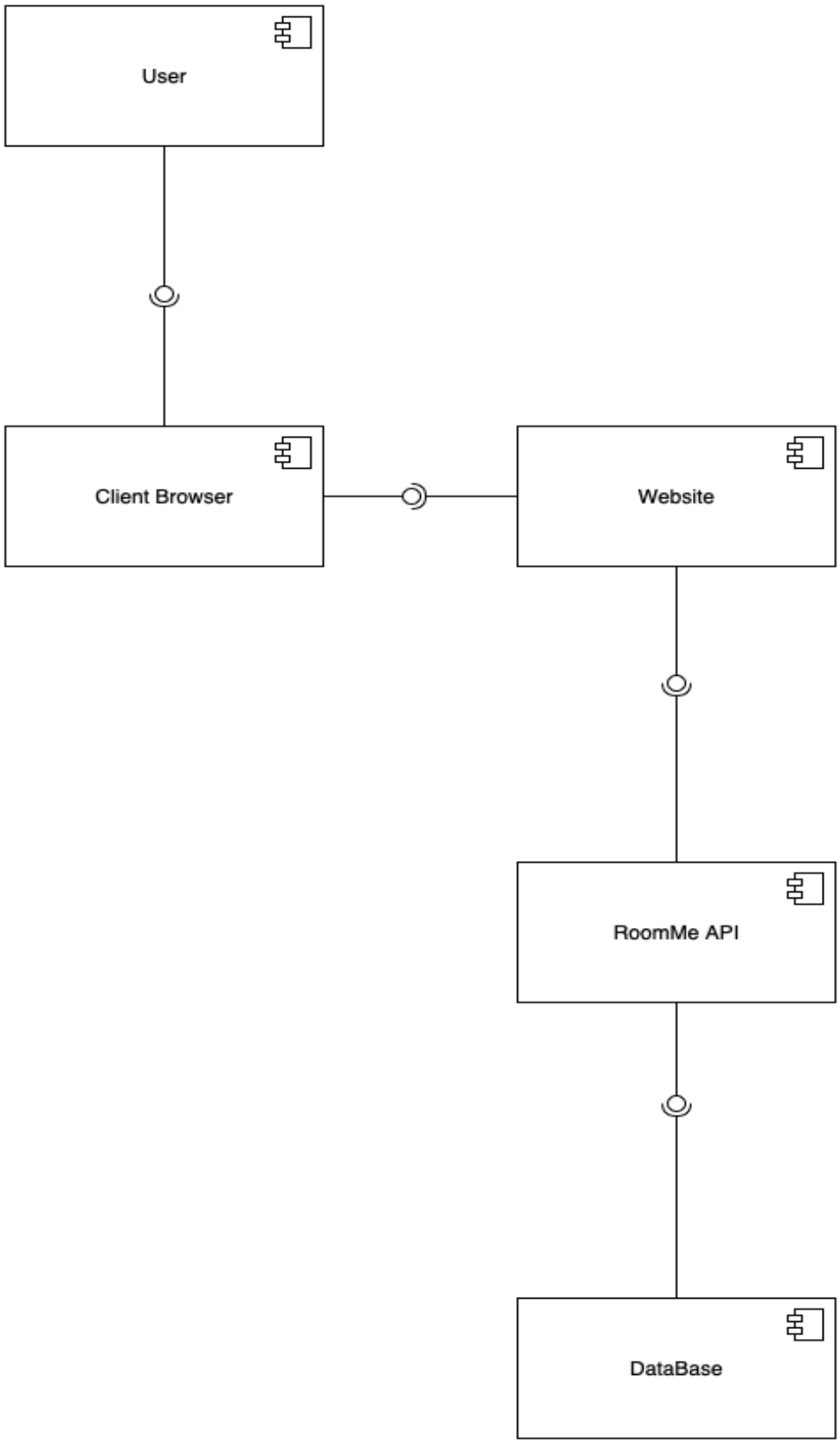
Additional non-trivial processes: Sorting

- User feed displays listings from DB, by default is sorted by newest first
- Can sort by lowest price

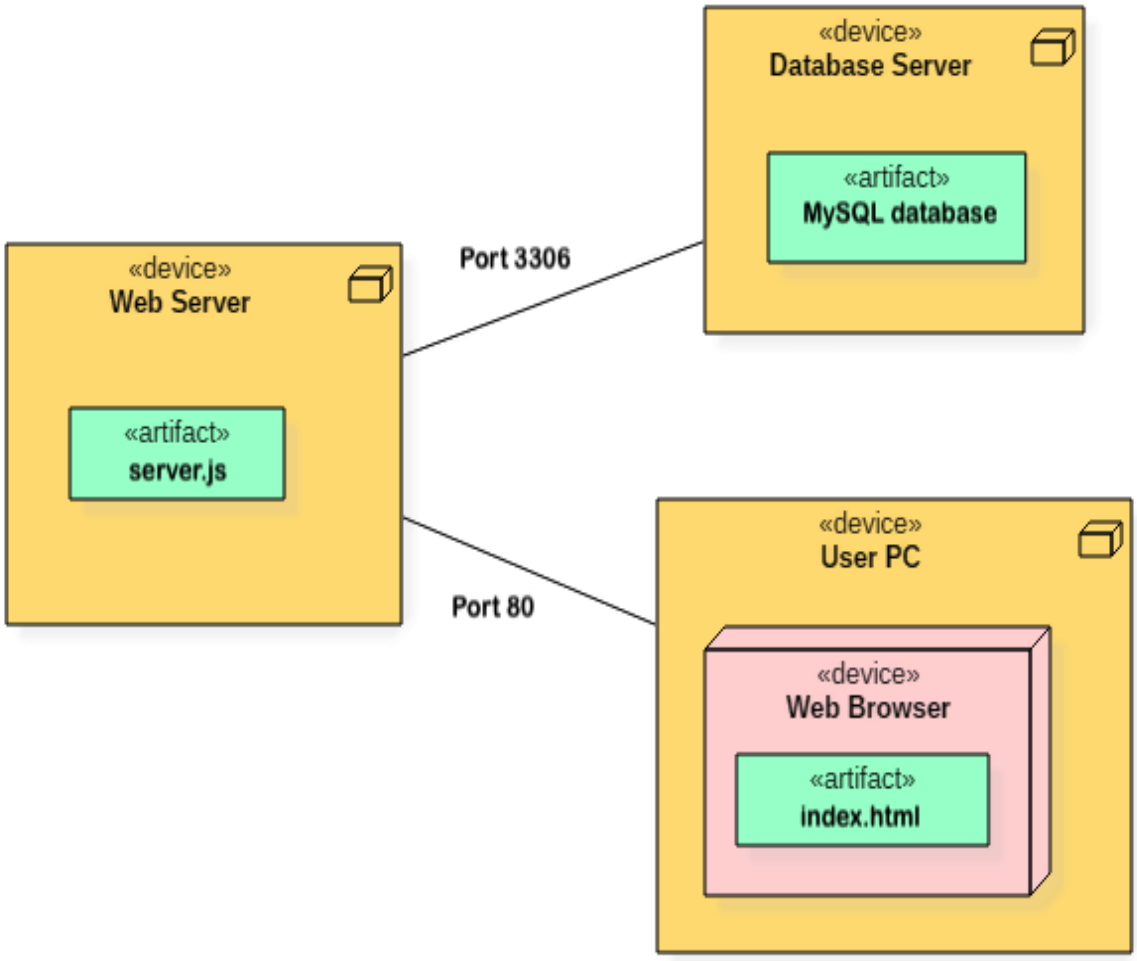
4: High Level UML Diagrams



Component diagram:



Deployment Diagram:



5: Key risks for the project

A. **Skills risks**

a. Not having enough knowledge and experience about the programming languages that are going to be used in developing the app would delay the development of the app. For most of the team members, this project is the largest we've worked with: this comes with a bit of a learning curve.

b. To remedy it, people working on the app need to practice more coding in those programming languages and try to code quickly.

B. **Scheduling risks**

a. Delay in developing the app because the computers used for its development are not powerful enough and takes a lot of time to process anything, making it harder to meet the committed deadline. Another possible issue is scheduling conflicts amongst teammates, since each of us have our own routines.

b. To avoid this issue, request permission from the university to use their resources which will help speed up the development and testing time. We can alleviate scheduling issues by ensuring that we all communicate our availability, and try to fit meetings where they accommodate everyone.

C. **Technical risks**

a. The app may not work as intended on some devices or even not work at all. That could happen due to differences in models of phones, their display aspect ratio, or the version of the operating system it may be running.

b. Use device emulators to emulate several kinds of devices and test the app on the emulated devices to verify that the app works on them as intended. This helps to ensure all screen sizes and operating systems are compatible with our app.

D. **Teamwork risks**

a. Disagreements among teammates over certain aspects of the app, its development, or a feature of it could increase the tension among the team members and hinder the development of the app. Additionally, unintentional miscommunication can lead to wasted efforts in the development process.

b. Try to solve the issue diplomatically by letting all team members listen to both sides with an open mind and collectively come to a conclusion. Miscommunication can be mitigated by discussing issues thoroughly, and ensuring that teammates are on the same page before they begin writing the code.

E. **Legal/content risks**

a. One risk is using certain content in the app that we may not have the proper license to use, or using parts of code that are not open-source could lead to intellectual property infringement.

b. Consult a lawyer regarding the contents and codes the team has used in the app that they have not created themselves and how to avoid intellectual property infringement.

6: Project Management

Milestone 2:

The second milestone of the project is the most dense we've received thus far. Just the same as any issue/task that seems overwhelming at first, organization, communication, and prioritization have helped the team to continue making progress. The team's process of turning milestones into tangible units of work is becoming more refined as we progress through the semester. This improvement has culminated in the adoption of task tracking software: the team uses a Trello workspace as a common area to visualize progress, and as a pool of 'to-do's' which they can select and begin working on. The workspace contains boards (and tasks) organized by the aspect of the project: backend, frontend, and documentation.

The ticket system will likely form the basis of the team's collaboration for future milestones, as it effectively serves our need for progress checking and task compartmentalization. Not only this, but the common workspace promotes collaborative efforts, allowing each member to contribute with ease. Receiving digestible units of work allows individual members to focus on the task at hand. My responsibility as team lead is to ensure an accurate and effective interface to the professor, the milestone requirements, and general progress of the project.