San Francisco State University
Computer Science Department
SW Engineering CSC648/848 Fall 2021 Section 02
Group 4
Milestone 2
10/05/2021

Team lead, GitHub lead/master, Document lead: Georgina Shirazi (email: gshirazi@mail.sfsu.edu)

Gabriel Pena, Database lead
Eanguy Eng, Database

Swetha C, Backend lead
Zhiling Huang, Backend

Sanket Naik, Frontend lead
William Zhong, Frontend

GatorRoomer

Version	**Date submitted**	**Action**
1.0	10/05/2021	Submitted
2.0	10/27/2021	Revised & Submitted

1) Functional Requirements - prioritized

Priority 1:

1) Users shall be able to filter through room listings using price, location, tags.
- Be able to filter out room listings that don't meet the users price or location requirements.
- User shall be able to set up their roommate preference (male only or female only, undergrad, grad, professor
- Also tags from the lister can be used to filter.

For room
- Users shall be able to filter for:
    - pets allowed
    - appliances
    - negotiable
    - college-oriented
    - smoking
    - parking availability
    - guest allowed overnight
    - private restroom
    - disability friendly
    - room size
    - number of bedrooms
    - number of bathrooms

For roommates:
- Users shall be able to filter for roommates by:
    - major
    - grad level
    - age
    - gender
    - user score

2) Users shall be able to use bookmark/ “like” room listings.
- Have the ability to save room listings for later.
- (with the option of making this available to landlords--who may offer a discount); or make it private

3) User shall be able to have separate sessions for each user.

Have separate sessions for each user. Need to make sure each user has their own info and not other user's info

4) User shall have new post notifications -
- Send a notification to users when a new listing is added by any user.

5) User shall be able to ask questions to landlord or roommate (webmail) -
- Send messages to the OP about any questions or doubts using the in-built messaging feature.

6) User shall receive email verification -

- Send an email to the user's email ID with a link that they can click to verify their emails.

7) User shall able to upload a picture(s) of the room
- No one wants to rent a room they can't see. Give the lister the ability to add 1 or many pictures of the room.

Priority 2:

1) User shall be able to report the spam
- Add the option to report a post as spam which then gets highlighted on the admin panel so that the admin can remove/take down the post.

2) App shall be able to display a map with pins showing the available apartments (Similar to PadMapper)

Priority 3:

1) User shall be able to provide reviews and upvote locations

How good is this location, roommate, landlord? Leave a rating and an optional review.

2) User shall be able to personalize account (person description, and social media links)
- Have a description so users can get to know you. An alternative could be to link your social media.

3) User shall be able to see landlord rent guarantee

make it mandatory for landlords to stick to their rental agreement right after paying for

renting, so it cannot be “sold” to another user. Could be something listed on a “terms of agreement” document.

4) Users shall be able to check distance to restaurants, stores, and gas stations nearby SFState.
How close is the nearest restaurant (cuisine type, has a drive-through); a drugstore, pharmacy, grocery store, department store; park; gas station; hospital; trail; the beach?

5) User shall be able to rate other users
- Like Uber, be able to rate your roommate based on how satisfied you were with the experience. Put less emphasis on listings from users with bad scores.

6) User shall be able to have a “saved post” draft system for landlords who want to save listings before publishing them
If you are a landlord and are working on a room-to-rent post, but it’s WIP… a location landlords can save and access drafts.

7) A calculator to do the math on how much it'll cost for a certain amount of time
- Calculate the total cost of a room for a certain amount of time (Months) so you know what you are getting into.

8) Landlord or roommate “vacation” or “away for a certain period of time” banner
Have a banner on the user’s profile (with notifications, so said user won’t forget they enabled such banner) which notifies other users that this user is away, and provides an optional rough estimate on when they will get back

Removed:

room listing durability
- Each room can be only listed during a certain period. Listing extension or renewal may be required.
Old posts do not need to be removed. Our website would not have enough traffic (at this point) to warrant cleaning up posts.

User shall be able to check the roommate's quietness level
How quiet is your roommate? Do they use headphones or use speakers? Potential field on roommate’s profile and search function feature when seeking out roommates.
This is subjective. This can also be written inside a user’s bio description by the user.

landlord ability/ how to contact a landlord/ landlord ratings/ landlord profile
If there is a problem with the sink, how easy is it to contact the landlord? (rating)

Contact information for landlord, and a personal profile similar to roommates (name, description, social media, et cetera).

This can be included in the Landlord's profile. It could be in the description.

an "if I brought a guest to the room, what would I do" section on potential roommate's profiles

If a roommate brought a guest, what would they do? Where will the guest park? A little optional field roommates can put on their profiles.

This can be a hardcoded text section above the user's description. Could be shown when the user is updating their profile's description. Doesn't have to be a separate field.

a "make an offer" option when renting out rooms.

Avoid the $1 listing price with a "make an offer" description. Avoid having such post listed with a price--instead have the post put in a section where the user must contact the landlord for the price.

Negotiable tag in room replaces this. To avoid $1 listings, this could be a rule the user agrees to when they are publishing a room to be rented out. Can be hardcoded in plaintext.

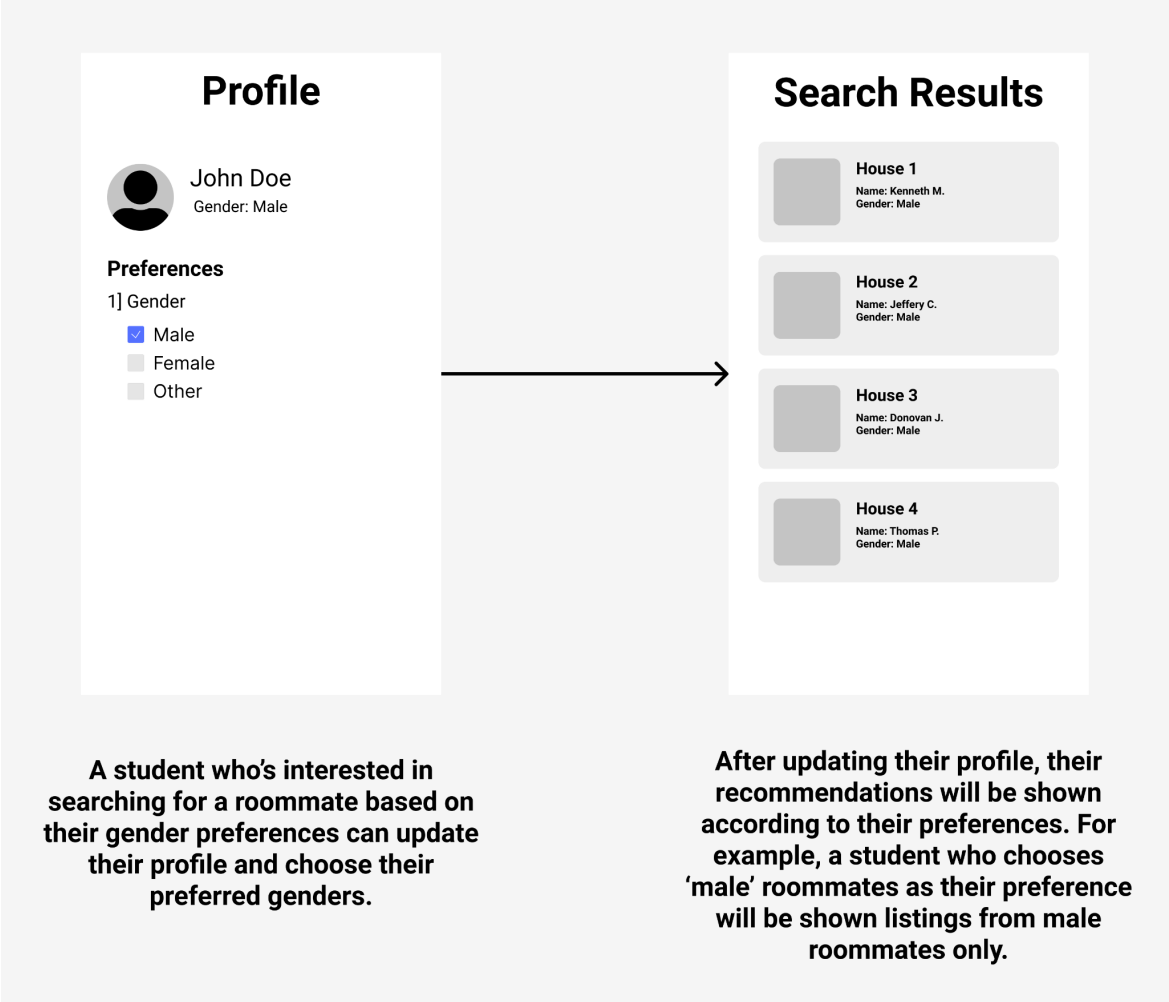
roommate chore-ability(?)

Optional. A roommate's ability to do chores, like wash their own dishes, et cetera.

This can be a hardcoded text section above the user's description. Could be shown when the user is updating their profile's description. Doesn't have to be a separate field.

2) UI Mockups and Storyboards

Students by Gender - Gender Preferences

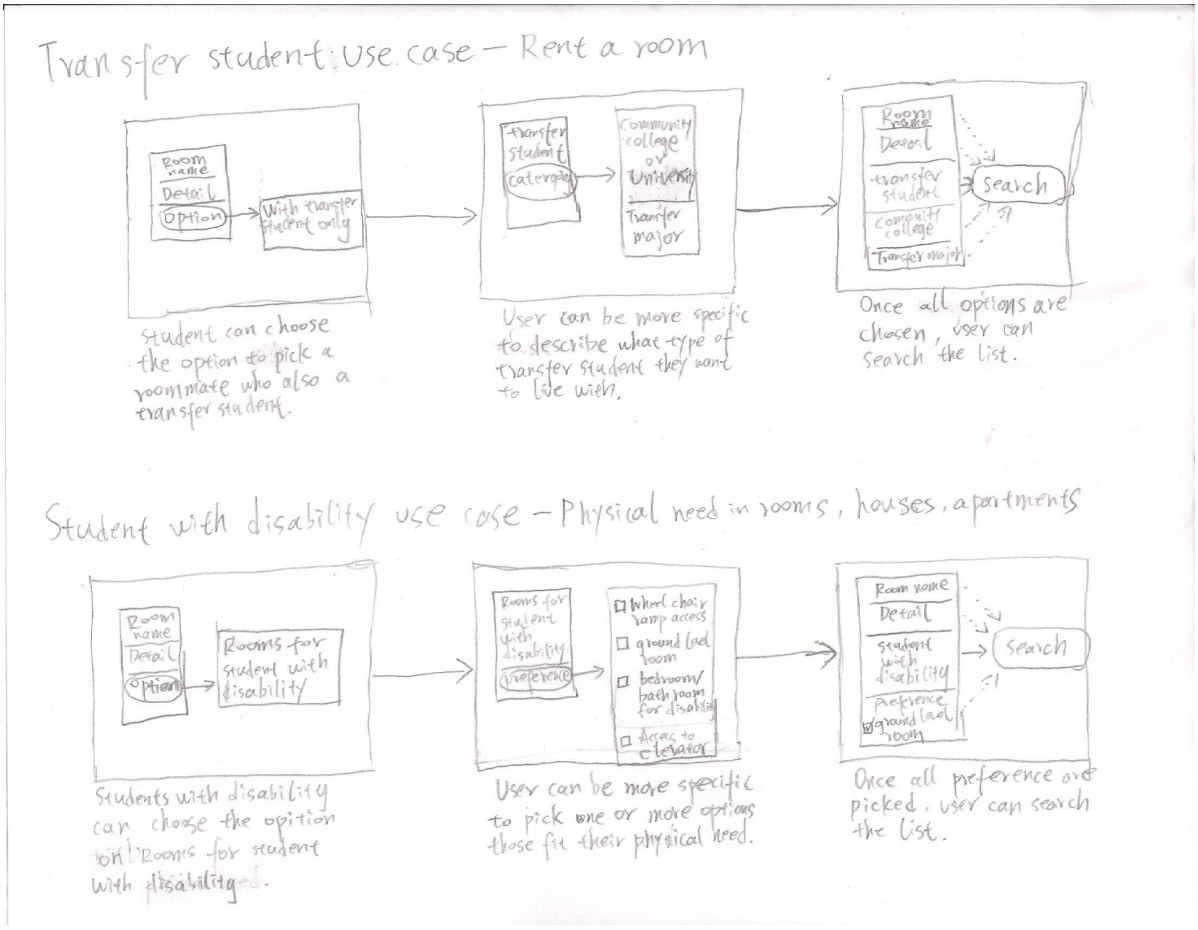


A student who's interested in searching for a roommate based on their gender preferences can update their profile and choose their preferred genders.

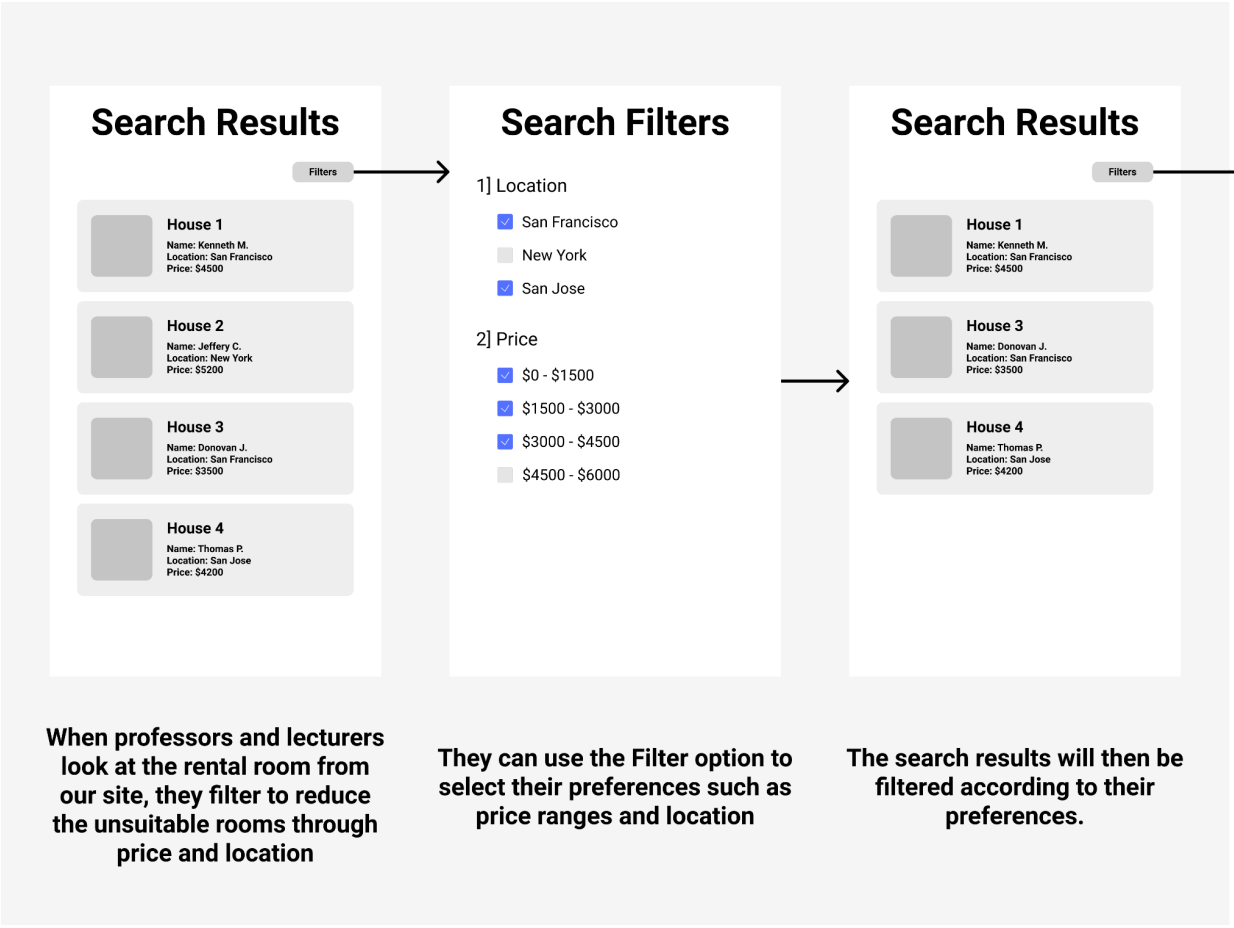
After updating their profile, their recommendations will be shown according to their preferences. For example, a student who chooses 'male' roommates as their preference will be shown listings from male roommates only.

Transfer Students - Rent a Room

Students with disabilities - physical need in rooms, houses, apartments.



Professors - Filter by Location and Prices

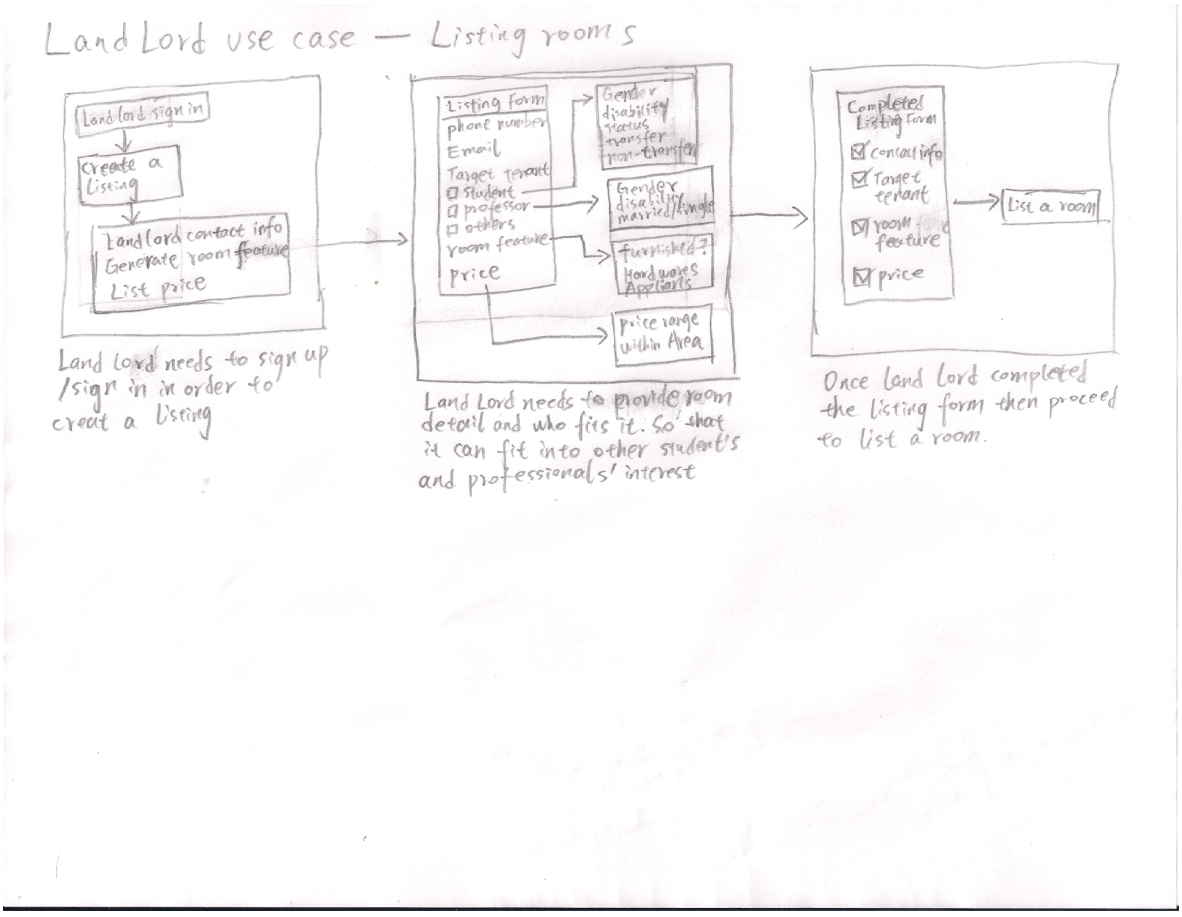


When professors and lecturers look at the rental room from our site, they filter to reduce the unsuitable rooms through price and location

They can use the Filter option to select their preferences such as price ranges and location

The search results will then be filtered according to their preferences.

Landlords - Listing Rooms



3) High level Architectures, Database Organization

RoomListing

Field name	Data type
Room ID	Integer
Lister ID	Integer
Location	VarChar
Zip Code	Integer
Size of room	Integer
Type	VarChar (example: (Room, House, share room, et cetera)
Description	VarChar
Price	Decimal
ListTime	Date
Size	Decimal
Number of bathrooms	Integer
Number of bedrooms	Integer
Tags	Varchar
Available	Int

Base User

UserID	Integer
Email	Varchar
Password	Varchar
Name	Varchar
Age	Integer
Social Security (Integer)	Integer
User Score	Integer

Gender	Varchar
Type	Varchar
Hidden	Int

Student

SchoolID	Integer
Major	Varchar
Grad_level	Varchar
StudentID	Integer

Professional_User

Prof_ID	Integer
JobLocation	Varchar
Zipcode	Integer

Messages

Message ID	Integer
Sender	Integer
Sendee	Integer
Contents	LongText

ProfilePictures

UserID	Integer
UserPic	varchar

Tags (This table can be used for our reference so that we can know what are the search tags)

TagID	Integer
TagDes	varchar

ProfilePictures

UserID	Integer

UserPic	varchar

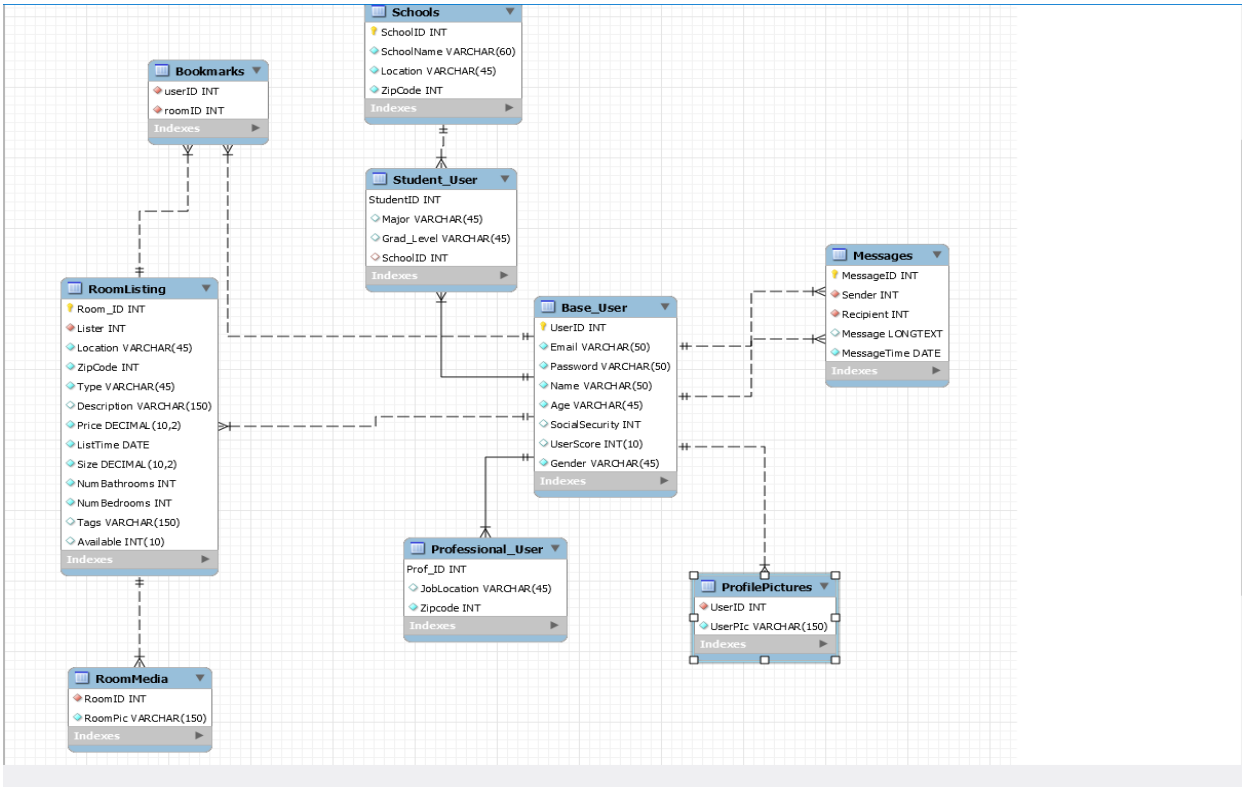
RoomMedia

RoomID	Integer
RoomPic	varchar

Bookmarks

userID	Integer
toomId	Integer

Database Schema:



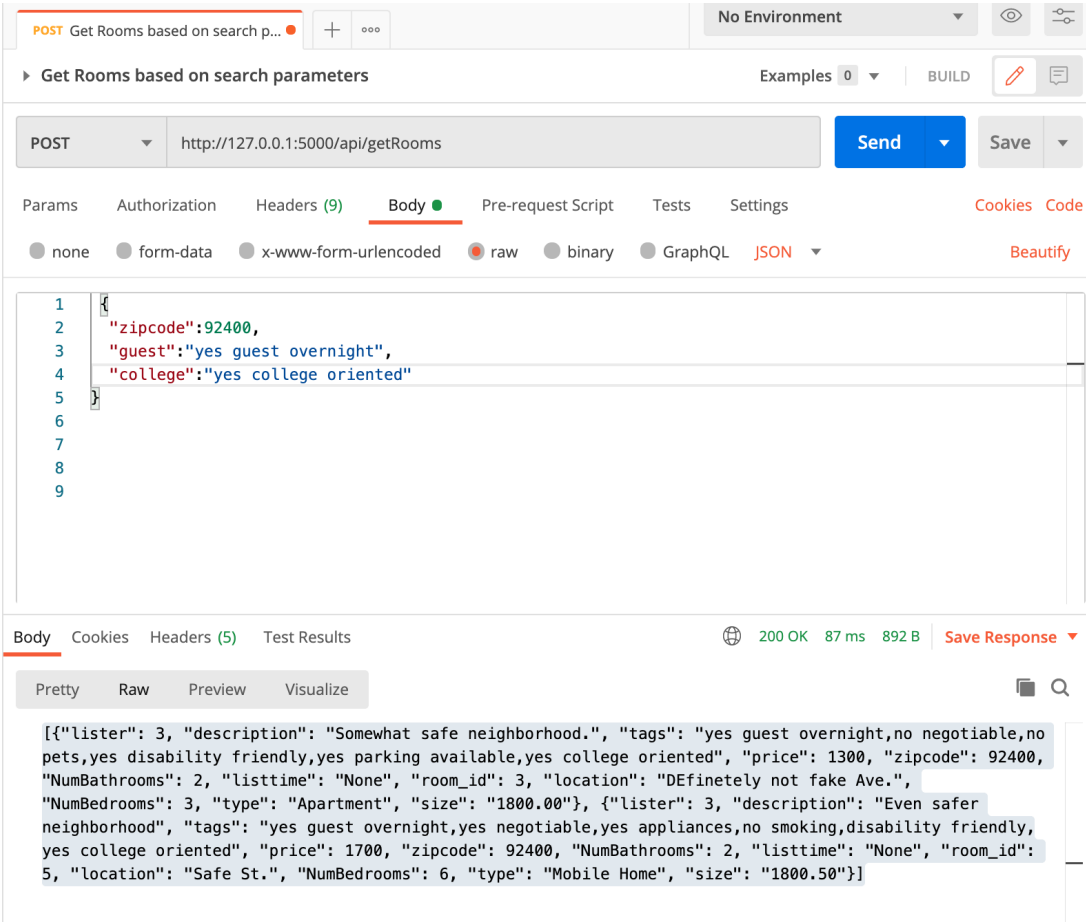
Media Storage: We shall be using Amazon S3 to store the images because it is easier to retrieve and store any amount of data in its buckets. The URLs for each item will be stored in the Database.

Search/filter architecture and implementation: see the API implementation below.

a) Get rooms for a given zip code/location with search tags.

NOTE: Zip Code or location is mandatory. The search is a strict AND search
Method name: getAllRooms
Method type: POST
Return data type: List
End point: /api/getRooms

examples: request by Zipcode and request by Location



Request JSON:

```
{
 "zipcode":92400,
```

"guest":"yes guest overnight",
"college":"yes college oriented"
}

Response JSON: [
{
"lister":3,
"description":"Somewhat safe neighborhood.",
"tags":"yes guest overnight,no negotiable,no pets,yes disability friendly,yes parking available,yes college oriented",
"price":1300,
"zipcode":92400,
"NumBathrooms":2,
"listtime":"None",
"room_id":3,
"location":"DEfinetely not fake Ave.",
"NumBedrooms":3,
"type":"Apartment",
"size":"1800.00"
},
{
"lister":3,
"description":"Even safer neighborhood",
"tags":"yes guest overnight,yes negotiable,yes appliances,no smoking,disability friendly,yes college oriented",
"price":1700,
"zipcode":92400,
"NumBathrooms":2,
"listtime":"None",
"room_id":5,
"location":"Safe St.",
"NumBedrooms":6,
"type":"Mobile Home",
"size":"1800.50"
}
]

CURL: curl --location --request POST 'http://127.0.0.1:5000/api/getRooms' \
--header 'Content-Type: application/json' \
--data-raw '{

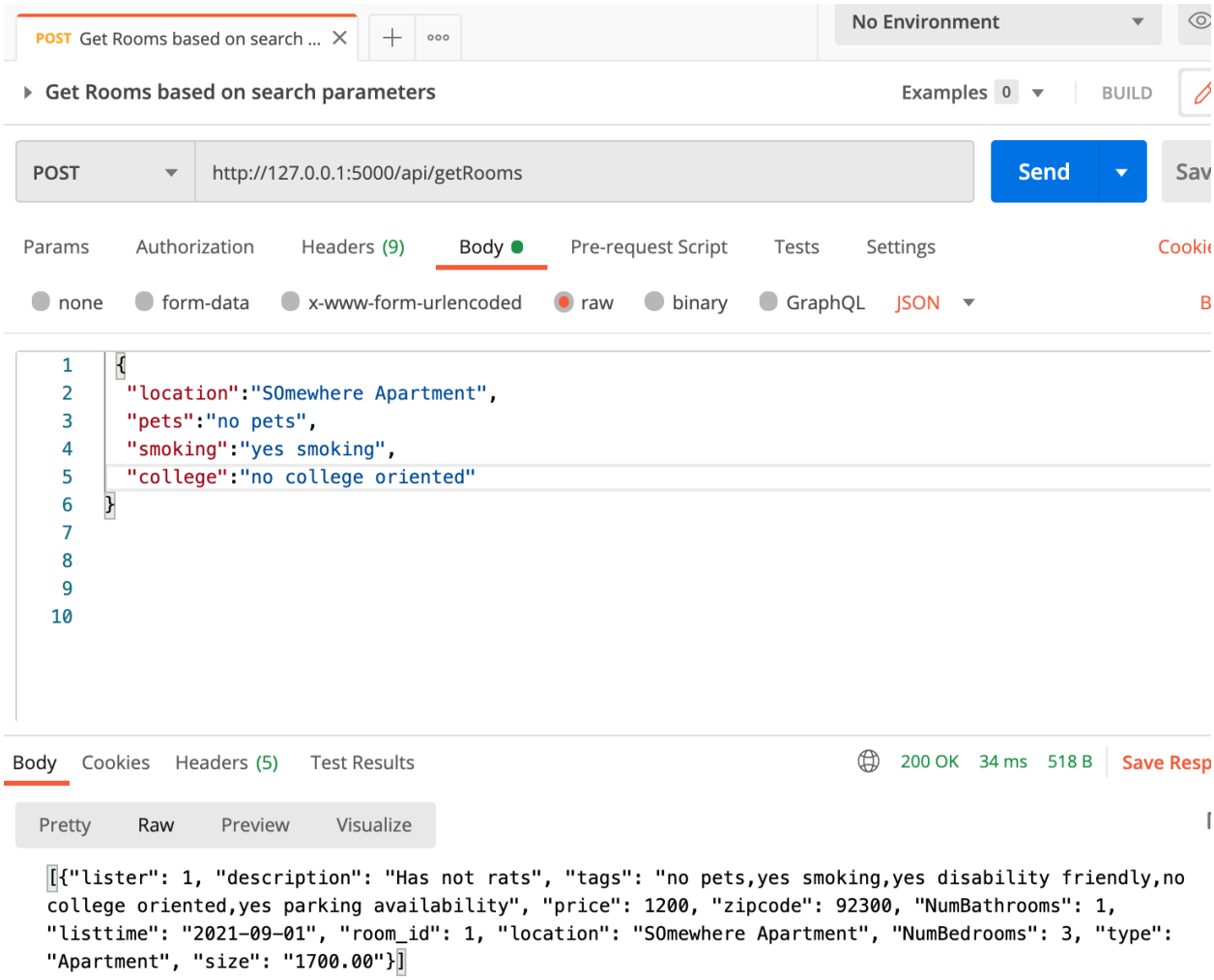
"zipcode":92400,

"guest":"yes guest overnight",

"college":"yes college oriented"

}'

a) Based on location & search tags



Request JSON:

{

"location":"SOmewhere Apartment",

"pets":"no pets",

"smoking":"yes smoking",
"college":"no college oriented"
}

Response JSON:

[
  {
        "lister":1,
        "description":"Has not rats",
        "tags":"no pets,yes smoking,yes disability friendly,no college oriented,yes parking availability",
        "price":1200,
        "zipcode":92300,
        "NumBathrooms":1,
        "listtime":"2021-09-01",
        "room_id":1,
        "location":"SOmewhere Apartment",
        "NumBedrooms":3,
        "type":"Apartment",
        "size":"1700.00"
  }
]

CURL: curl --location --request POST 'http://127.0.0.1:5000/api/getRooms' \
--header 'Content-Type: application/json' \
--data-raw ' {

"location":"SOmewhere Apartment",
"pets":"no pets",
"smoking":"yes smoking",
"college":"no college oriented"
}'

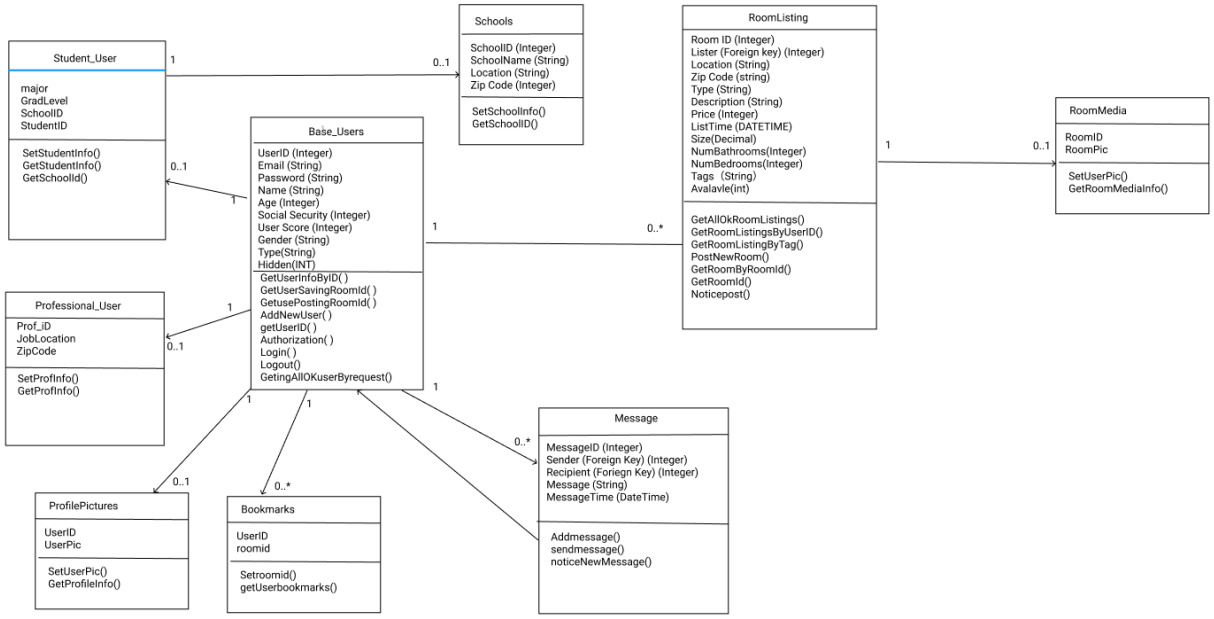
A similar process will be applied to the Roommate finder.

Changed software tools or frameworks:

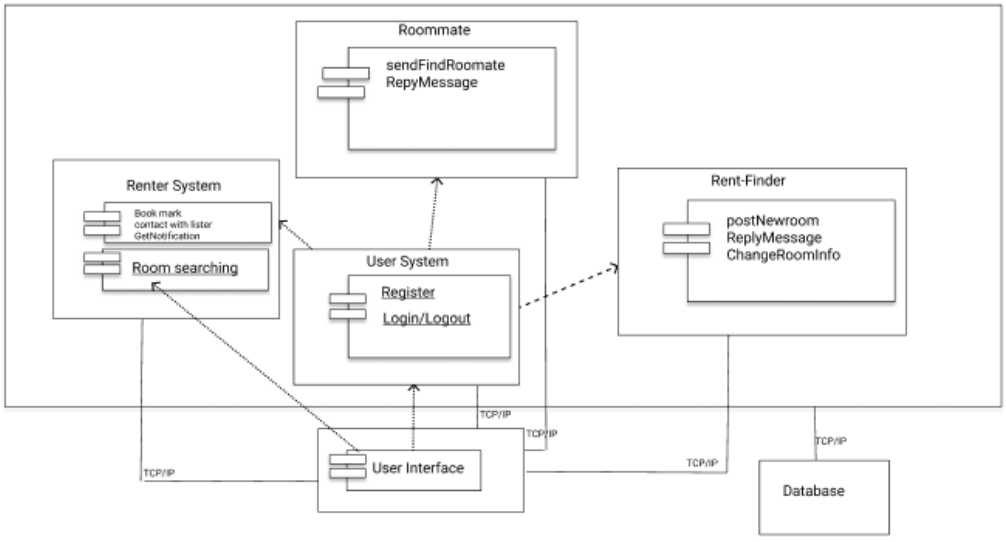
We added Amazon S3 as a software tool to store images and videos. It is a cloud service which provides users the ability to upload objects to a “bucket” and access them via a URL.

4) High level UML Diagrams

UML Class Diagram



UML Component and Deployment Diagram



5) Identify actual key risks for your project at this time

skills risks (do you have the right skills)

We do not know everything. A solution is to search the internet for tutorials and functions and methods from APIs. Googling helps, youtube tutorials, reading the amazon web service documentation, flask documentation, bulma documentation.

schedule risks (can you make it given what you committed and the resources)

We have regular meetings -- 2 to 3 times per week (we are making it to 3 times per week). We ask team members which times work for them, and have a google spreadsheet for our members to indicate which days and times they are available so we can meet at a team.

technical risks (any technical unknowns to solve)

We are using ec2 as our server. We shut it off to avoid aws going over its free tier hours, and started the instance again to use it and ssh into it.

Every time the instance is shut off and started again the public IP changes.

For unknown reasons (perhaps the boot loader), the connection timed out and even on amazon's side, it could not connect to the instance. We followed the documentation to try to resolve this issue, but ended up creating a new instance and reinstalling the stack on it.

We tested this new instance three times:
stopping the instance, starting the instance, sshing into it
installing part of the software on the stack, stopping the instance, starting the instance and sshing into it
installing the rest of the software on the stack, stopping the instance, starting the instance and sshing into it

All three were successful, so we started to use this instance instead of the faulty one.

teamwork risks (any issues related to teamwork)

Backend has a lot of work to implement. Solution: bring people from other teams to help the backend implement the APIs.

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

We have not had any issues with this so far. A lot of the images supplied to the S3 have been from Unsplash photos. The photo credits are in the image name and we are planning to have an image credits page.

6) Project Management

M2 was managed and planned out via consulting with the team, using Zoom scheduled meetings, Zoom break-out rooms, Google Drive and Google Documents, Google Calendar, and our Discord Server. We shall use Trello in the future to distribute tasks.

In the beginning, the first thing we did as a team was go over M1 and implement missing elements into M2. All the tasks were presented as a todo list where team members freely decided which task to complete.

Then, we broke M2 into tasks per team:
Frontend: storyboards
Backend: develop and use search queries for the database; create UML diagrams;
Database: design and implement a database schema
The Vertical Prototype

We utilized the Vertical Prototype to assist in planning and further developing our database schema and our Backend's own apis. The frontend started working on the Vertical Prototype. They created a UI with a free text box and drop downs. Our backend started working on search queries, and we decided to use a strict search from there. The team collaborated on planning the database schema from there, we added new tags, renamed existing tags, and deleted some tags. The tags are for a tagging system, for room listings.

We used Google Drive to store our M1 work and Google Documents to collaborate on the M2 document and its sections. As a team, we worked on parts 1 and 5 together (Functional Requirements and Risks). Zoom break-out rooms and Discord were used for teams to directly communicate with each other, either with DMs or on their own channels (Frontend, Backend, and Database each have their own channel).

Team members were instrumental in combining all the documents into M2.

For M2, all the different teams (Frontend, Backend, and Database) were more autonomous and communicated with the different teams.