

Design Document

Team Name: KirbyDownB

Team Members and Emails: Aditya Acharya, aacha002@ucr.edu;

John Shin, jshin029@ucr.edu;

Eric Ong, eong001@ucr.edu

Francisco Lagos Vilaboa, flago001@ucr.edu;

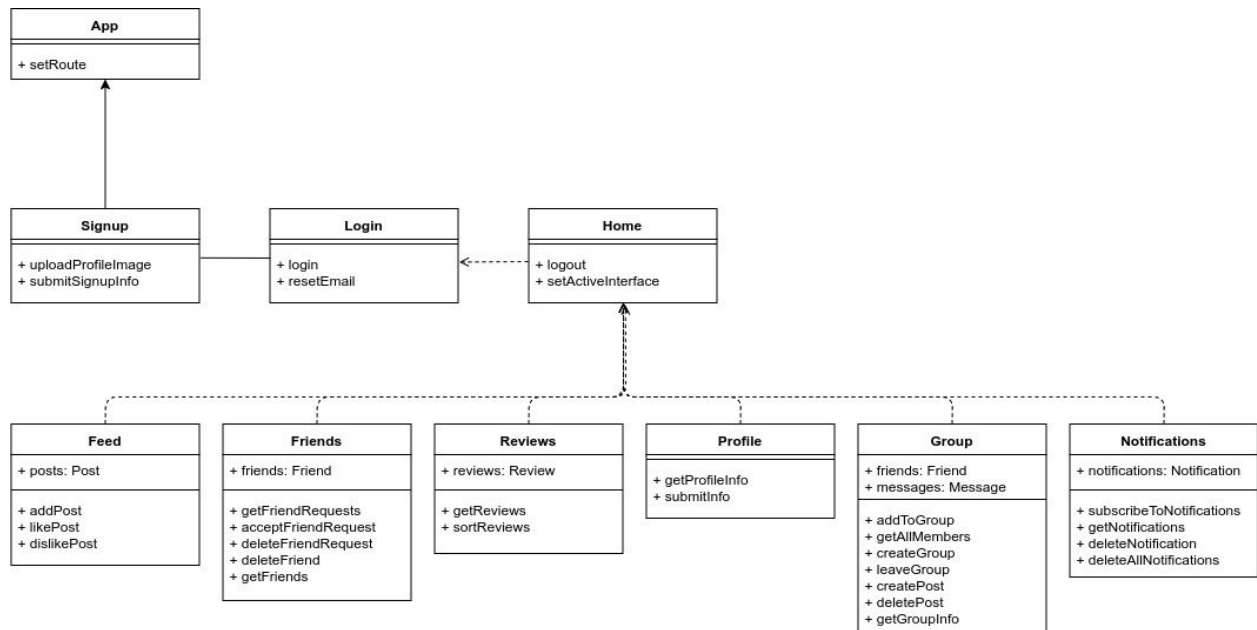
Project: Roomster

Communication Channel: Text Messages and Facebook Messenger

Architecture

Roomster will be comprised of 3 core components that help serve the app. The React frontend will communicate with the Flask-RESTPlus backend, which will handle various endpoint requests via HTTPS. The back end will store the data in a PostgreSQL database.

- Frontend: React
 - Redux will be used as a more global state management system and to handle the JWT
- Back End: Flask-RESTPlus
- Database: Postgres



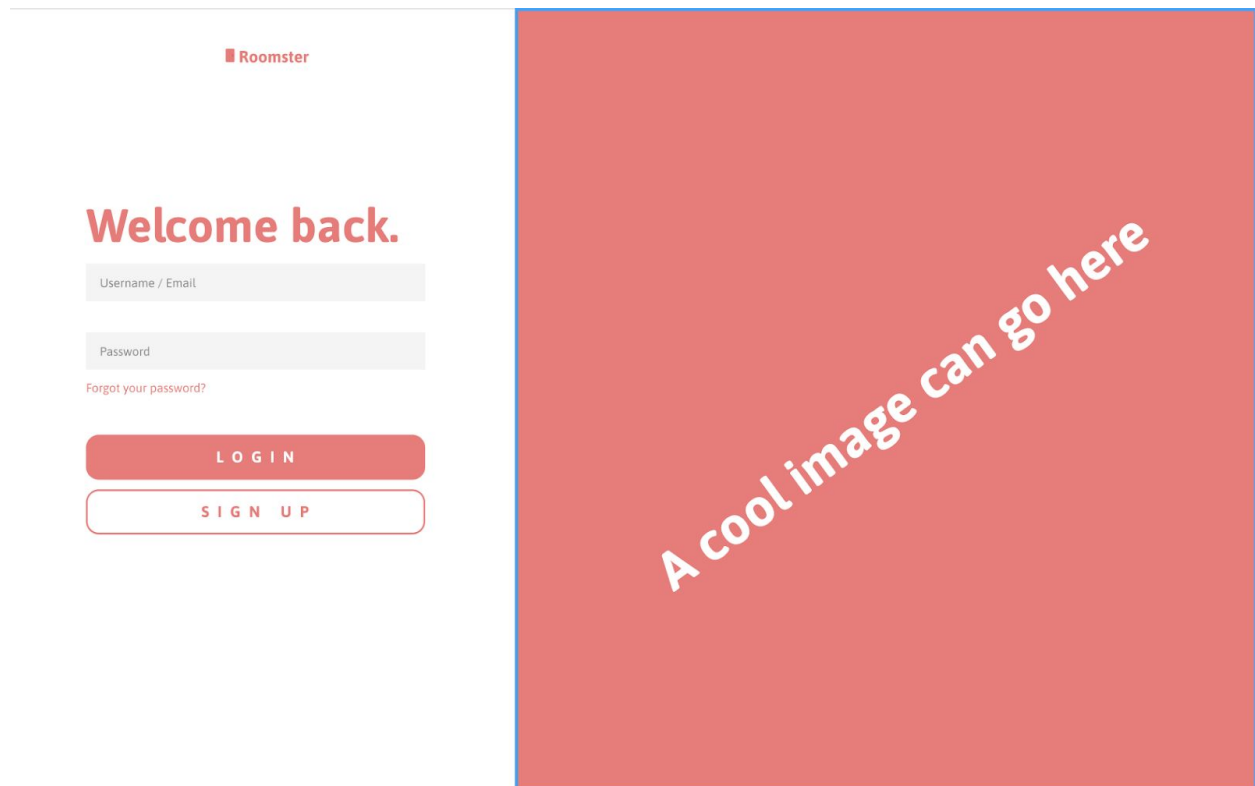
Sprint 1


The goal of this sprint will be to have some of the core functionalities of the app. More specifically, there will be a component that conditionally renders a Login or Signup component, which is rendered based on what the user clicks. Both of these components will interact with the Flask-RESTPlus backend. Next, the user should be able to see a basic rendering of all the listings from other users looking for roommates.

Upon logging in, a JWT will be returned to verify that the user will stay logged in and set. This will also ensure that when the user refreshes the page, he/she will still be logged in. The user profile information will also be set in redux store for easy access to user information when logged in.

Upon signing up, the user will receive an email confirmation via the SendGrid API to confirm that he/she has logged in. In addition, when a user logs in, the JWT will be set in local storage and the user will be routed to the dashboard/ home page.

After logging in, the user will be able to see a grid of listings, showing potential roommate options. When the user clicks on any one of these listings, he/she will be routed to a separate component, which will show basic information about the potential roommate.



Roomster

Personal Information

Room Details

Submit

Rooming made easy for you.

Email *

Username / Email

Password *

Password

Confirm Password *

Confirm Password

First Name *

First Name

Last Name *

Last Name

Address

Address

Previous

Next

Sprint 2

For sprint 2, we will implement several features and re-do some others. In terms of what we will redo, we will switch from an SQL database to a NoSQL database. In sprint 1 our database was SQLite with SQLAlchemy as our ORM. For this sprint, and for the rest of the project, we will switch to MongoDB with MongoEngine as our ORM. This is because we had several issues with migrating changes to the models in sprint one, especially in Flask. Flask does not have a built in migration tool, unlike some other frameworks. The only option for migrations is Flask-Migrate, which we could not get to work. Because of the fast paced nature of our project, we felt that we should change the models to be more flexible and easy to change at a moment's notice. As a result, we will change to MongoDB, which gives us the freedom to store the documents as we please.

The models will now look like this

```
class User(db.DynamicDocument):

    email = db.EmailField(unique = True, required = True)
    password_hash = db.StringField(required=True)
    first_name = db.StringField()
    last_name = db.StringField()
    # address = db.StringField()
    phone_number = db.StringField()
    date_of_birth = db.StringField()
    gender = db.StringField()
    range = db.StringField()
    location_of_interest = db.StringField()
    age = db.StringField()
    pf_pic = db.URLField()
    bio = db.StringField()
    number_of_roommates = db.StringField()
    ethnicity = db.StringField()
    price_range_min = db.StringField()
    price_range_max = db.StringField()
    duration = db.StringField()
    friend_requests = db.ListField()
    friends = db.ListField()
    occupation = db.StringField()
    likedPosts = db.ListField()
    dislikedPosts = db.ListField()

class Posting(db.DynamicDocument):

    name = db.StringField()
    poster_email = db.EmailField()
    date = db.DateTimeField(default=datetime.now())
    content = db.StringField()
    tags = db.ListField()
    likedEmails = db.ListField()
    dislikedEmails = db.ListField()
    images = db.ListField()
```

In this sprint we will also implement the Postings, where users who have signed up for our service can post about themselves and signal to other users that they want to find roommates. This will be stored in a separate set of documents

We will also allow users to upload images, both for their profile pictures and for their posts. This will be done using the `FileStorage` object, `Firebase`, and `MongoDB`. The `FileStorage` object will allow our endpoints to accept multiple files from the client side (React). These files will then be uploaded to `firebase`, where a link/reference to that image will be generated based on a unique naming schema. Then, this link/reference will then be stored in the posting or users' `MongoDB` document object. It can then be used on render to display the image.

Users will now be able to send friend requests , add friends, and delete friends. When a user sends a friend request, the person will see that person in their friend requests list. Then will then have the option of accepting it or ignoring it. Once the friend has been accepted, the user has the option to delete them.

Sprint 3

For Sprint 3, our goal is to add features such as ratings and reviews of other potential roommates. This functionality will include being able to read experience of people having others as roommates. The user should be able to go to a potential roommate profile, and see their information as well as all the reviews that have been written on the profile of the potential roommate. In a person's profile you will also be able to write a review of a person by clicking to write a review button

We will also be working on being able to form groups in which people will be able to interact with each other. You will be able to have different groups with different people. People will be able to interact with each other. When you get a message in a group you will receive a notification through `twilio`. This is an important functionality of the webpage because people need to be able to communicate to decide if they want to live with a certain person.

Our third and final user story for this sprint will be the `zillow` api. We will use it for showing listings for houses. As a user you will be able to look up for houses, apartments and studios with different price ranges. When you find these houses you will be able to send it to a group so that other people within you group will be able to see it.

This is our final sprint so it is important that we also focus on doing minor fixes for current and past sprint tasks. In this sprint we will focus on perfecting the website and making sure that everything works as expected.