BUNKBUD - BUFF ROOMMATE MATCHER

Amatullah Sethjiwala
amse5199@colorado.edu

Keerthika Rajvel
kera5806@colorado.edu

Swarnalatha Natarajan
swna2675@colorado.edu

1 PROJECT SUMMARY

This responsive web application solves one of the most common problems faced by college students everywhere. The central idea is to help students look for a place to live. The application will match the student with an appropriate roommate according to their preferences.
The web application requires an account creation. The application consists of two kinds of users, students looking for roommates as well as students who need a place to stay. Students who have a house and are looking for a roommate can post vacancy information along with floor plan, amenities and rental details. On the other hand, students who need a place to stay would be able to search for vacancies. Preferences need to be filled out in both cases in order to enable the process of matching with a roommate.

2 FINAL STATE OF THE SYSTEM

2.1 System Architecture

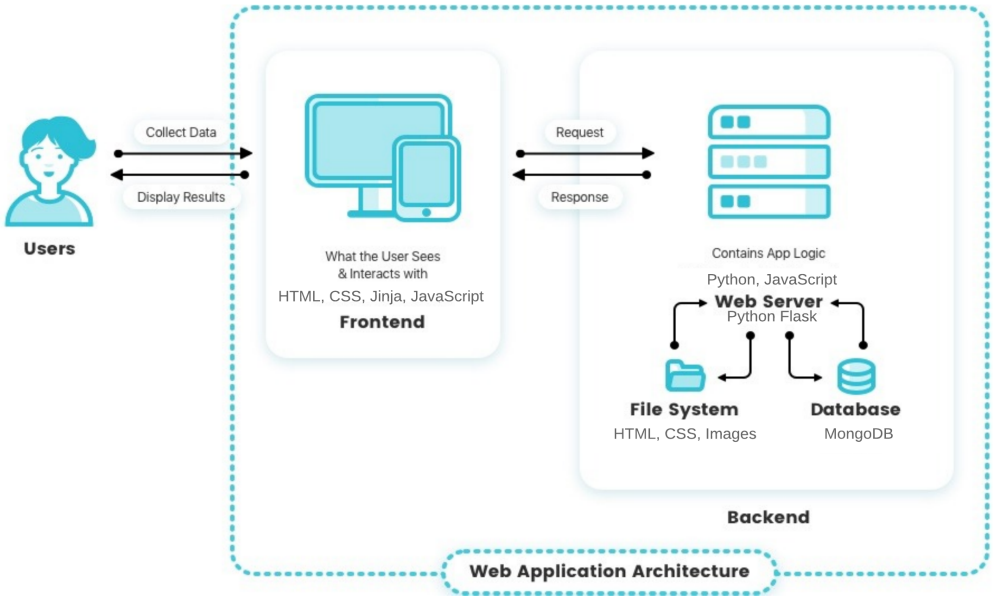

Figure 1: System Architecture

The final architecture of the system is as seen in Figure 1.The user interacts with the system via the front-end. The front-end of the system is developed using HTML, CSS, Jinja and JavaScript. The web service requests are sent to the back-end according to the user interaction. The back-end primarily consists of a web server (Flask). According to the request, the web server interacts with

the MongoDB database and the file system to send the appropriate response to the front-end module which in turn displays the response to the user.
This was the initially proposed system architecture and we were able to implement it without any compromise.

2.2 System Features

The final system comprises of the following features.

1. Login feature for an already existing user.
2. Sign-Up feature for a new user.
3. Separate questionnaires for users looking to find a room and for users looking to fill a room. This helps in understanding the user's roommate preferences.
4. A matching algorithm that matches the user to other users based on their preferences.
5. A dashboard displaying the user's matching results along with how strong of a match is to the user.
6. Option to directly email the user's matches to discuss further.

These were the initially proposed system features and we could implement all the features effectively without having to drop any of them.

2.3 Working of the system

The following series of figures depict the working of the system and its flow of control.

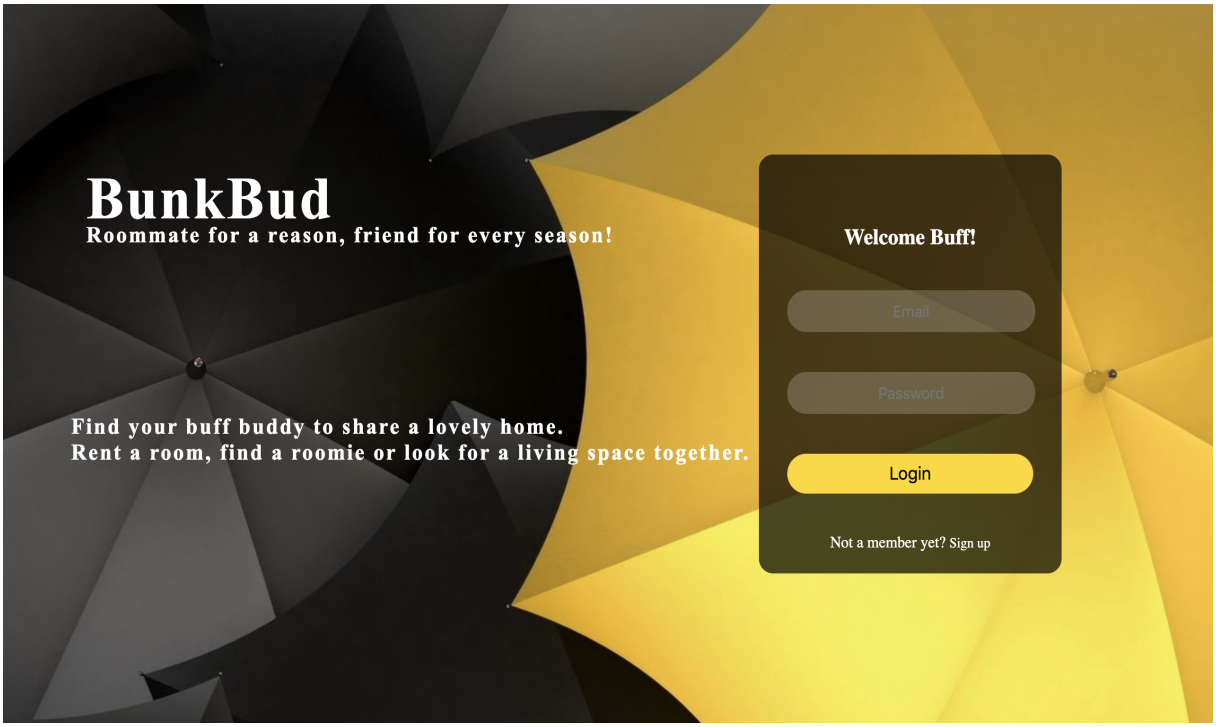

Figure 2: Login Page

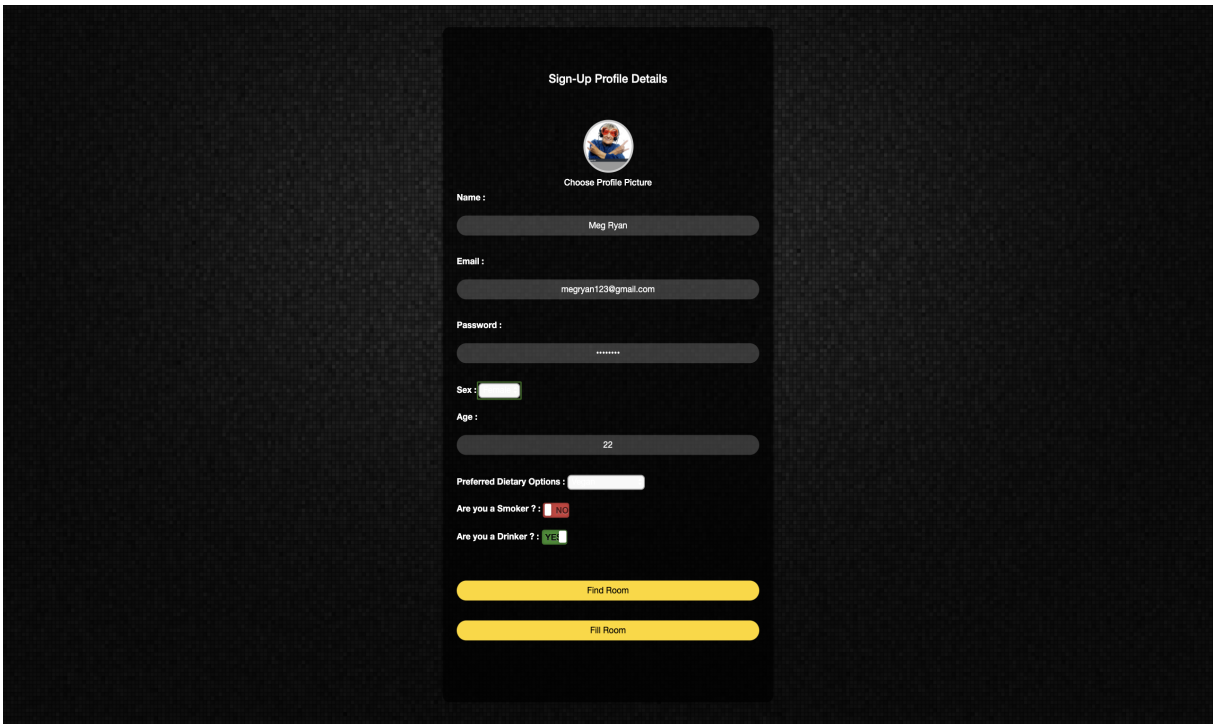

Figure 3: SignUp Page

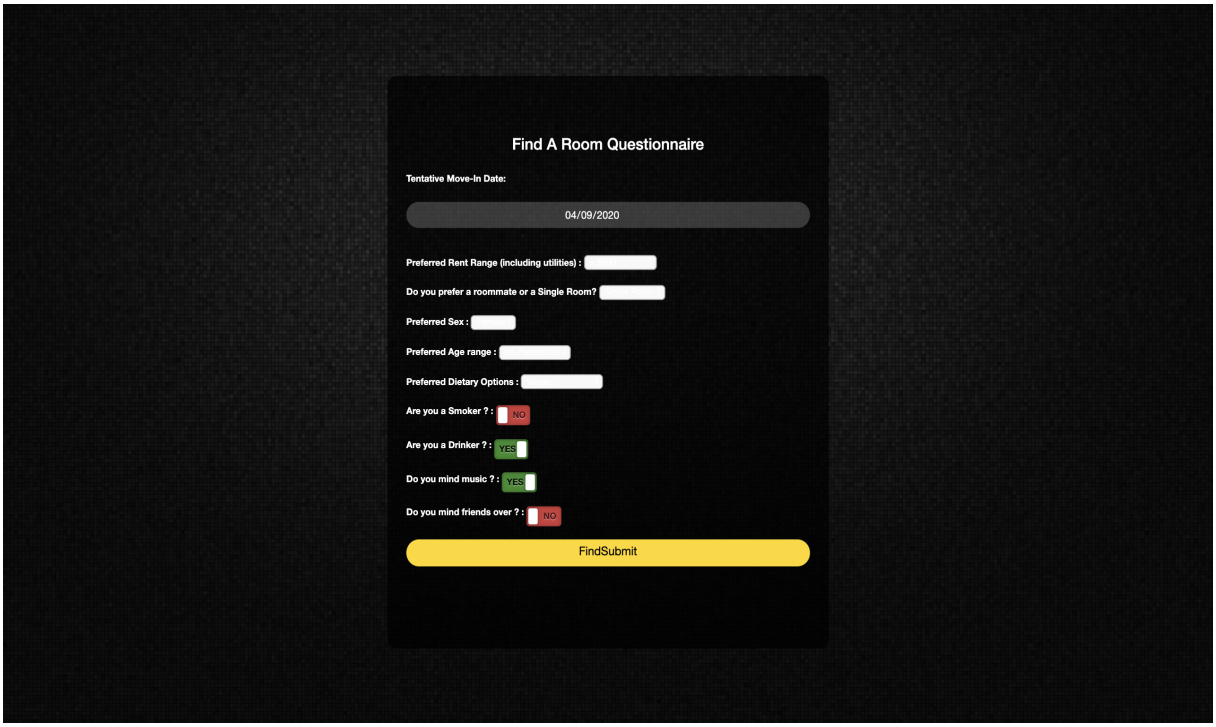

Figure 4: Find Room Questionnaire

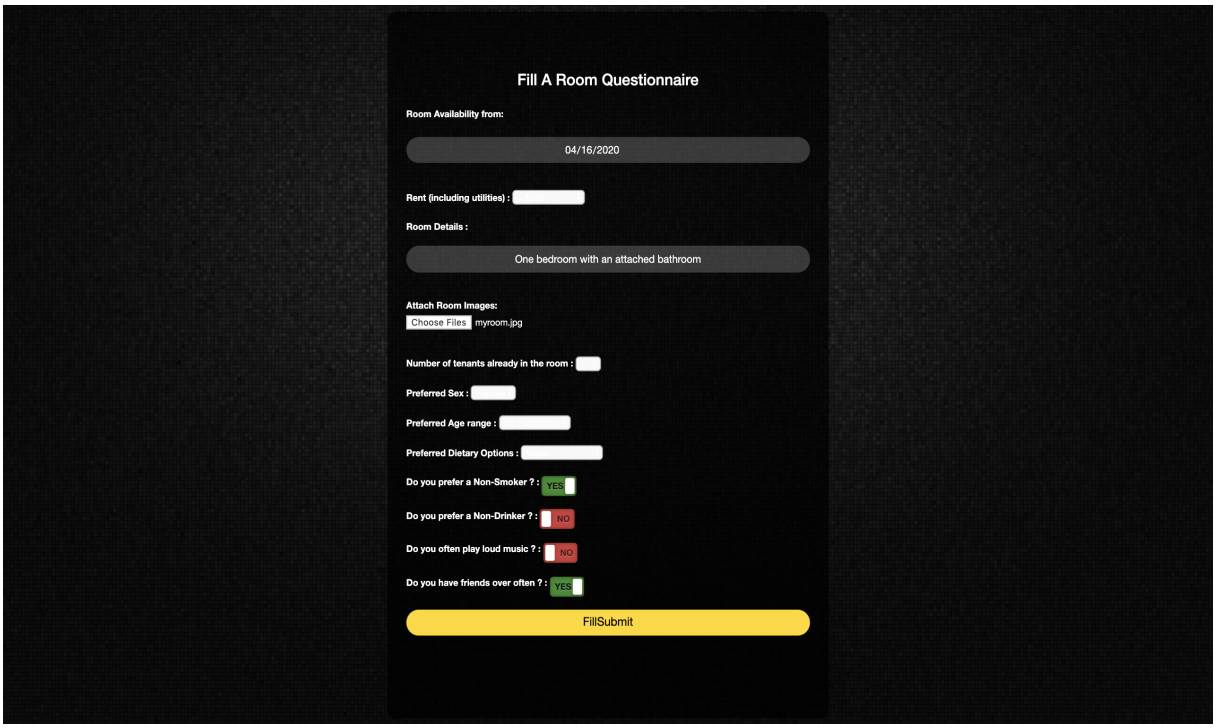

Figure 5: Fill a Room Questionnaire

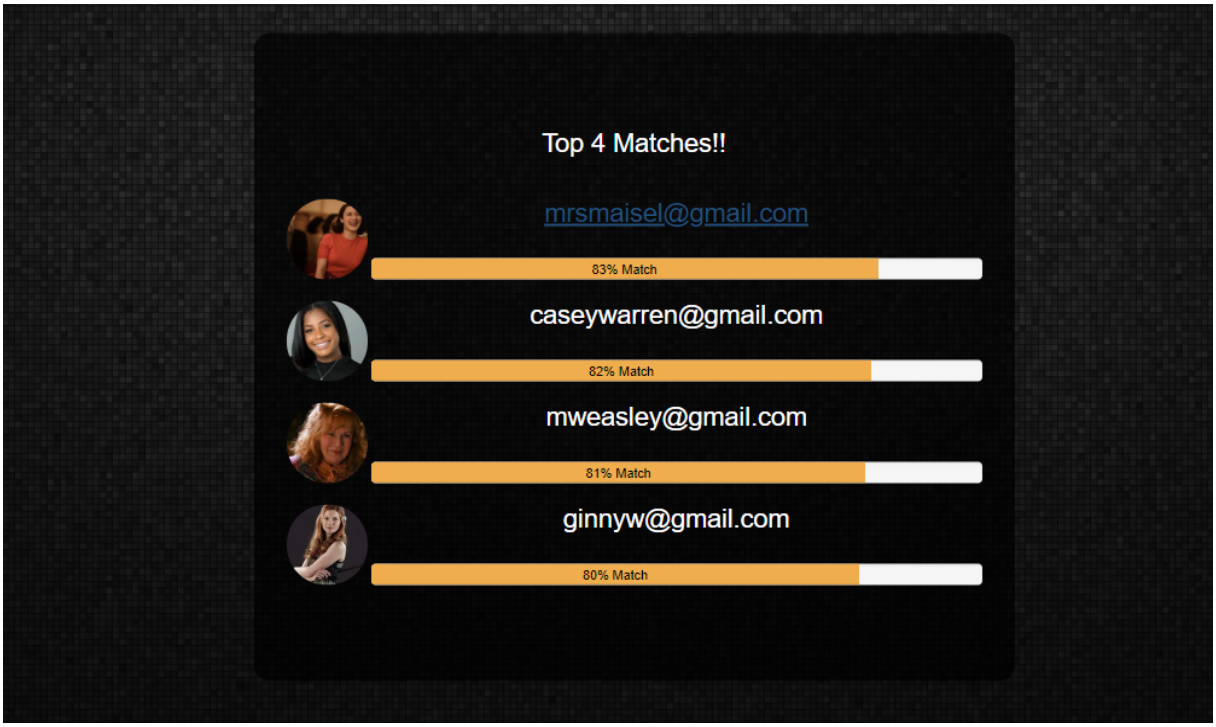

Figure 6: Dashboard Displaying Matches

3 FINAL CLASS DIAGRAM AND COMPARISON STATEMENT

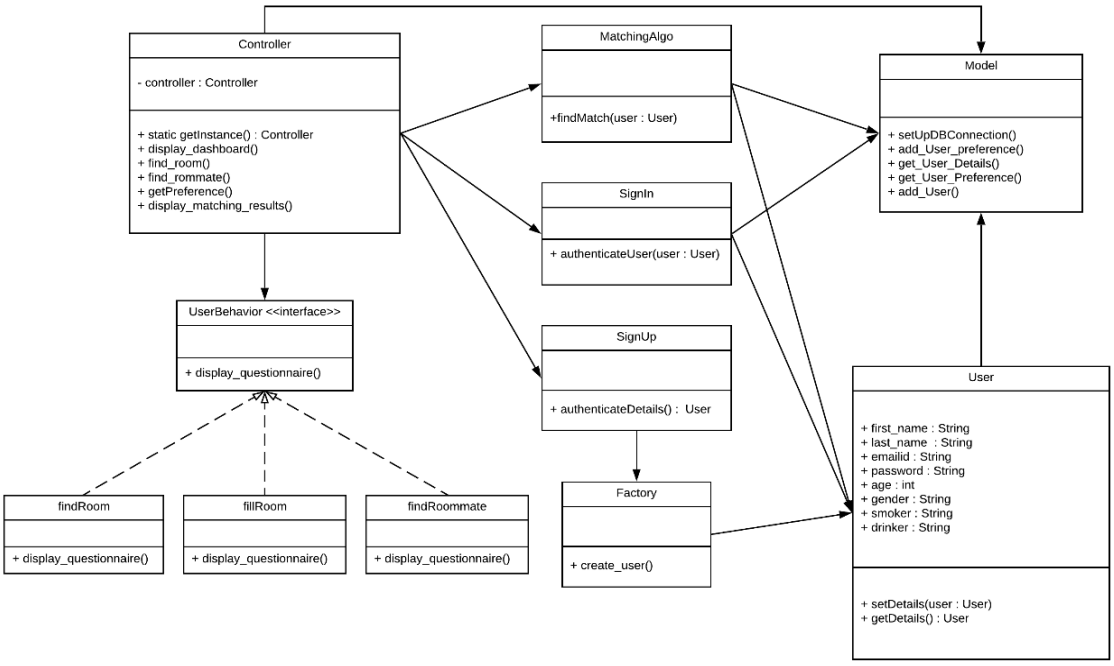

Figure 7: Class UML Diagram from Project 4

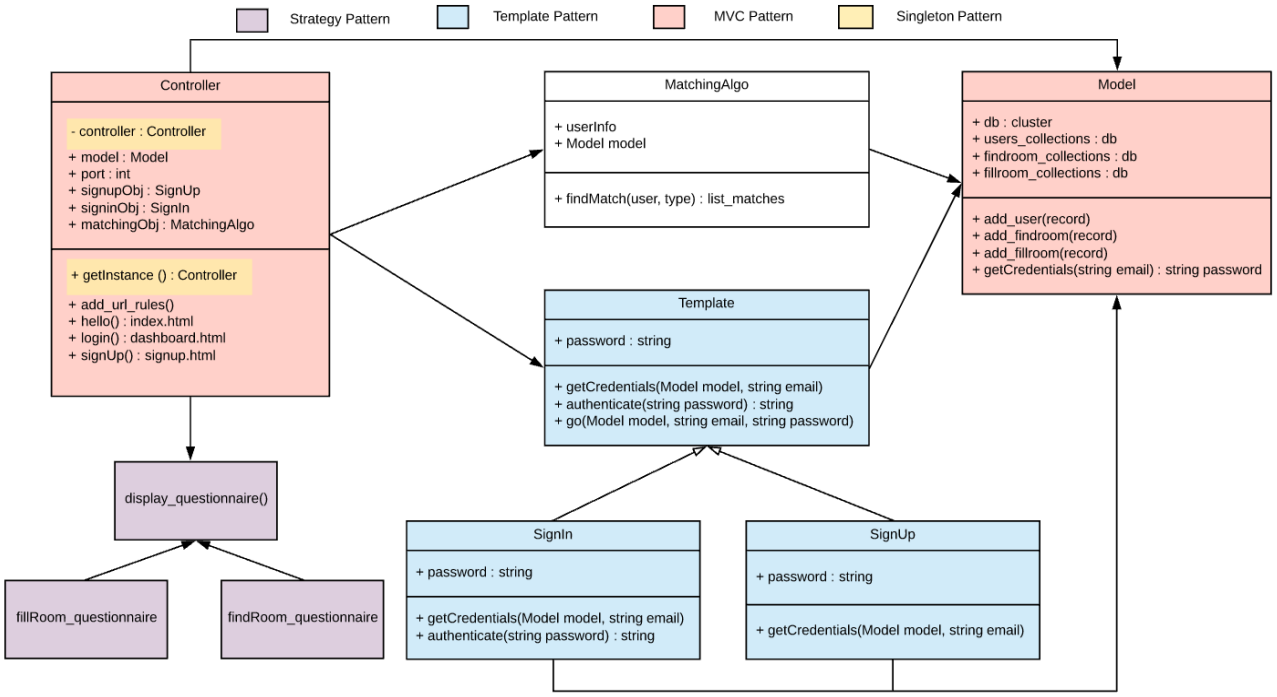

Figure 8: Final Class UML Diagram of the Overall System

As you can see from Figure 8, for our final system we have implemented the Singleton Pattern, Strategy Pattern, MVC Pattern and Template Pattern. Implementing the above patterns in our design has made our code more clean, maintainable and organized. The patterns are also helping us create more robust and uniform code. The incorporation of these patterns in the project design helps in the following ways:

1. **Singleton Pattern :** This helps us create only one instance of the Controller class. Since the controller is in charge of running the Flask server in a particular port, this pattern helps in preventing port conflicts.
2. **Strategy Pattern :** Since we have two kinds of users that have two kinds of matching questionnaires, we have encapsulated the different user behaviours using the strategy pattern. In future, if we decide to extend the system to another kind of user, we believe that this pattern would come in handy.
3. **MVC Pattern :** Our Controller is the web server, our View is the front end and our Model is the class that helps us connect to the MongoDB database. This greatly helps in separation of concerns.
4. **Template Pattern :** The login and sign-up features share the common functionalities of checking if the given username exits in the database and alerting the user accordingly. In this case, code duplication was significantly reduced using the template pattern.

From Figure 7 and Figure 8 you can see we have made a few changes. We removed the Factory pattern and User Class mentioned in Project 4. We thought this would be redundant as the information stored and retrieved from our database (Model Class) already comes in the form of a dictionary with attributes and values. We added the Template Pattern to our design as the SignUp and Login functions share common features and this helped reduce data redundancy.

4 OOAD PROCESS FOR OVERALL PROJECT

For this project idea we decided to go with creating a website for ease of applying OO Patterns and Designs. The following are challenges and things we learned and faced along the way :

1. **Design Phase :** Even though it was easy to come up with use cases from a website point of view, we faced difficulties in trying to translate the use cases so that it complies with the WAVE rule. Creating activity diagrams and sequence diagrams for use cases was difficult. However the diagrams helped us create a better design for the system.
2. **Incorporation of Patterns in our Design :** We faced several road blocks when it came to identifying and implementing patterns for our system especially since this was a website. Even though we have had experience in web development from past projects, it proved to be a challenge to incorporate patterns in our design. The patterns that we felt would work in theory failed when we went into the implementation phase.
3. **Pattern Implementation using Python :** We found that implementation of patterns in Python is not as seamless as it is in Java. We found very limited Python Flask documentation for pattern implementation. This resulted in a lot of trial and error with implementing patterns in our system.
4. **Experience in using OO Principles :** While designing and developing the project we have gained a massive amount of knowledge and experience in how to design and code a project using OO Principles and Pattern. This experience in the future will help us create more clean, robust, error-free and efficient code.

5 THIRD PARTY CODE vs. ORIGINAL CODE

The below mentioned links were only used as **references** for our system. **The code written for our project is all original and not copied from anywhere.**. The following are a few references that helped us during code development :

1. https://github.com/giri68/roommate-finder-client
2. https://www.tutorialspoint.com/python_design_patterns/python_design_patterns_singleton.htm
3. https://www.tutorialspoint.com/python_design_patterns/python_design_patterns_strategy.htm
4. https://codepen.io/dsalvagni/pen/BLapab
5. https://codepen.io/estrepitos/pen/qGmHc
6. https://reinvently.com/blog/fundamentals-web-application-architecture/