UMass Roommate Finder Application Design Document

Authors: Anurati Bhosekar, Noah Dixon, Sijia Hao, Dishank Deven Jhaveri

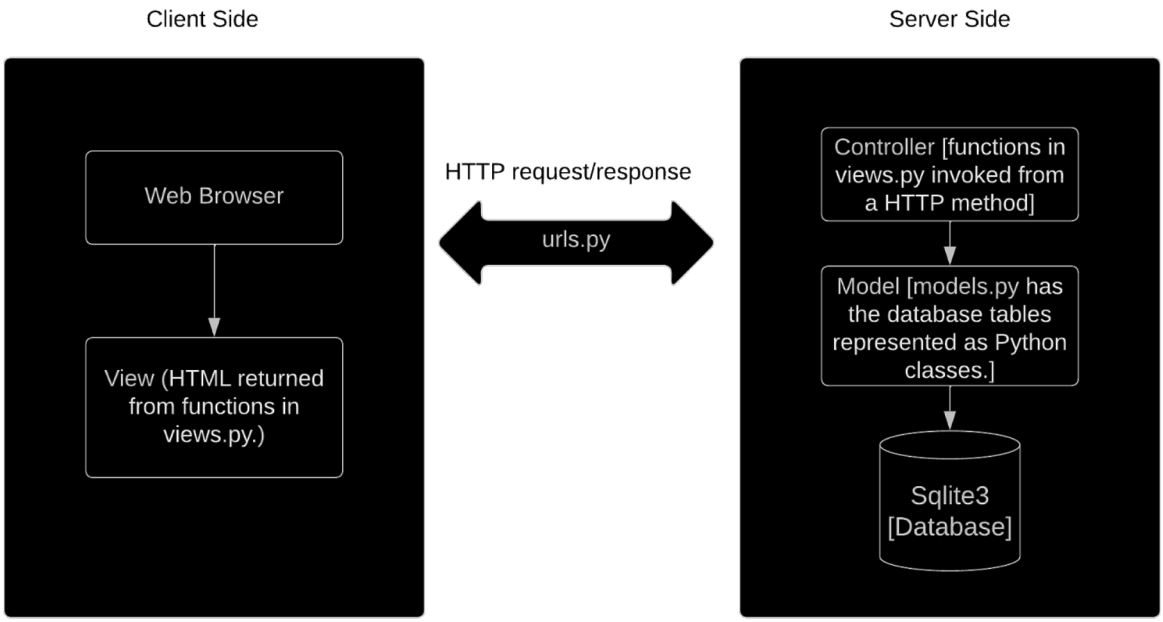
Architecture

The architecture of the University roommate finder web application is a combination of the client-server model and MVC architecture model. On the client-server level, The client-side handles the presentation of search filters, user data, and profile interaction, while the server-side manages data processing for new users, creating contact information requests, etc, and data storage in the form of Django Models. The client and server communicate via HTTP request and post methods. On the MVC architecture level, the model is represented by the database [Django's models.py file will be used as the metadata to hold the database structure] and the backend code that directly interacts with the database. This code contains Model objects that are capable of running SQL queries and can wrap the data retrieved from the database. The view is represented by the html pages stored in the templates folder [HTML and CSS files which contain the physical components of the UI, as well as the Javascript files used to directly manipulate the UI]. Finally, the controller consists of the backend code that manipulates both the model and the view [represented by views.py in Django]. It performs tasks such as filtering roommate search results or sending a contact request from one user to another.

The application is divided into three main components:

1. Front-end: The user interface that users interact with. It displays the information and features of the application to the user, allowing them to navigate the site and search for potential roommates. The front-end is built using HTML, CSS, and Javascript.
2. Back-end: The server-side code that handles user requests and processes data. It handles user authentication, filtering/searching for compatible roommates, and delivering notifications. The backend is built using Python and the Django Framework.
3. Database: The container that stores the data necessary for the application to function such as user information and roommate matches. The database is built using SQLite3.

Architecture Diagram:



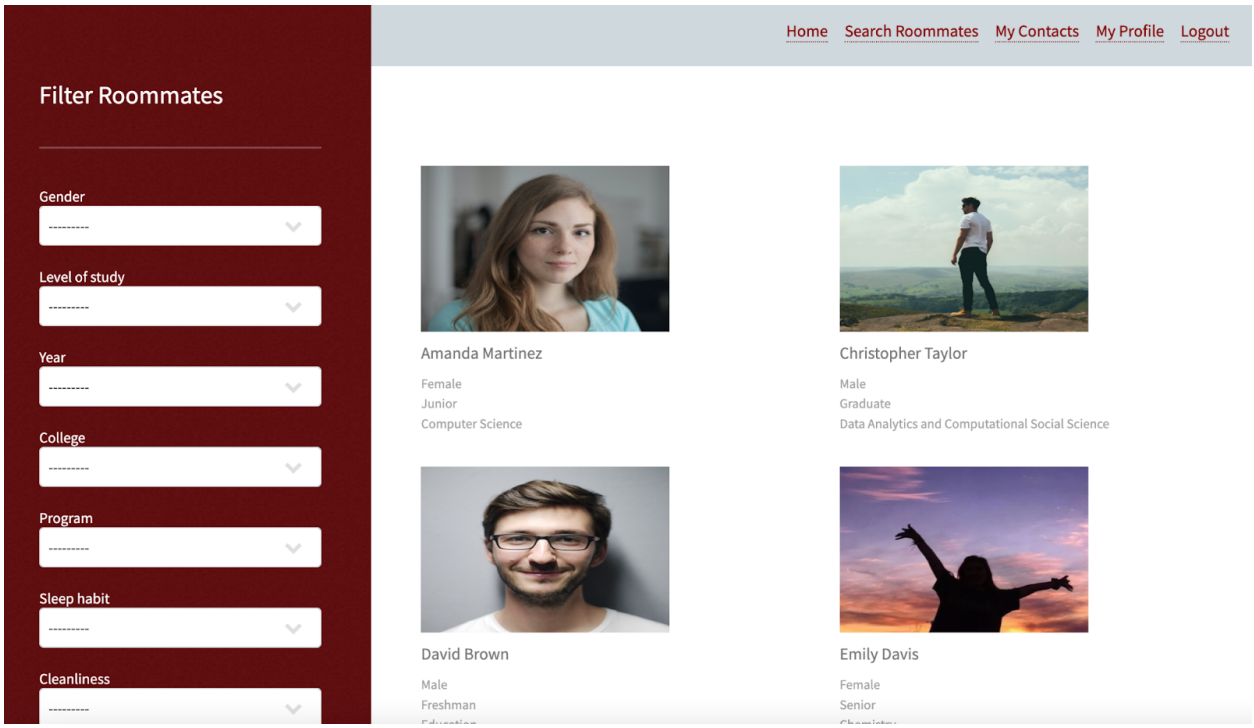
Tech Stack

In selecting the tech stack for this project we used our non-functional requirements to drive our decision making. We selected the following technologies to construct our web application:

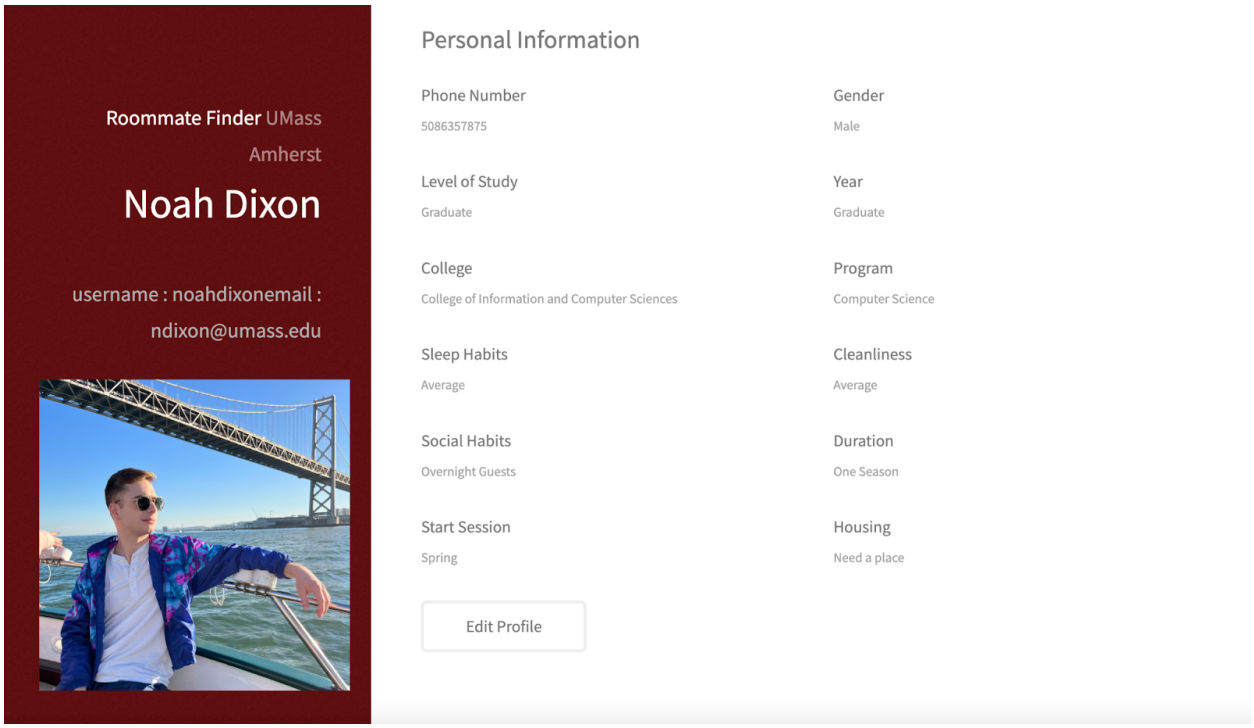
1. Django/Python: We chose to use the Python Django framework because it allows for rapid development, has a built-in admin interface, and robust security features. This will allow us to develop a highly maintainable, modular, and secure web application.
2. SQLite3: We chose to use SQLite3 for the database because it is a lightweight, file-based database system that is ideal for small to medium-sized applications. We do not anticipate the number of users will exceed 20,000 for our application, so this database should provide enough scalability. The use of a database in general will also aid in security and performance.
3. HTML / CSS / Javascript: Selecting HTML, CSS, and Javascript to construct the front end of our application was an easy choice as these are the standard industry tools used for creating web pages. They will make our application extremely compatible as any device with a web browser will be able to load and interact with the front end.

UI Design

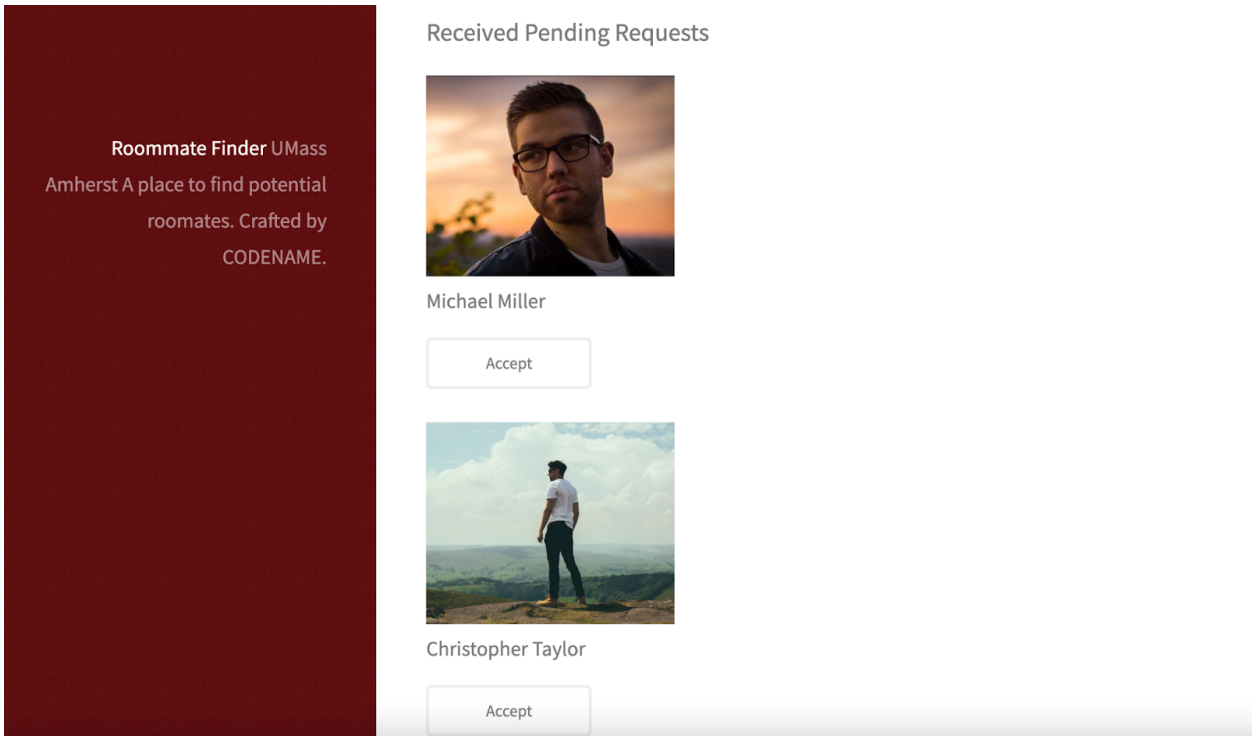
Search Roommates Page:



Edit Profile Page:

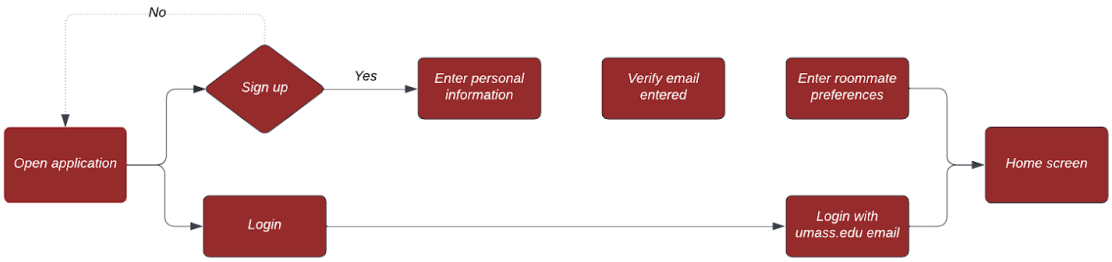


My Connections Page:

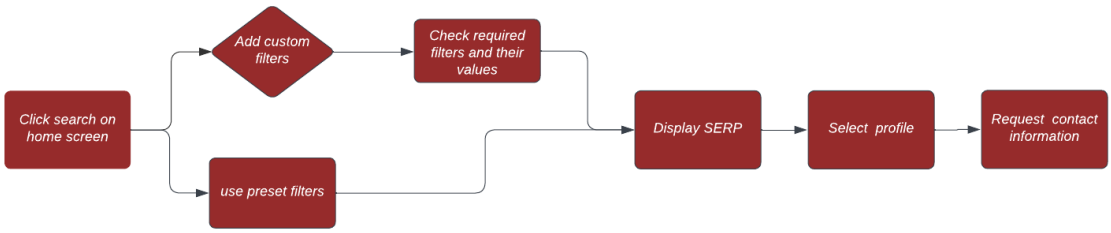


User Flow for Roommate application:

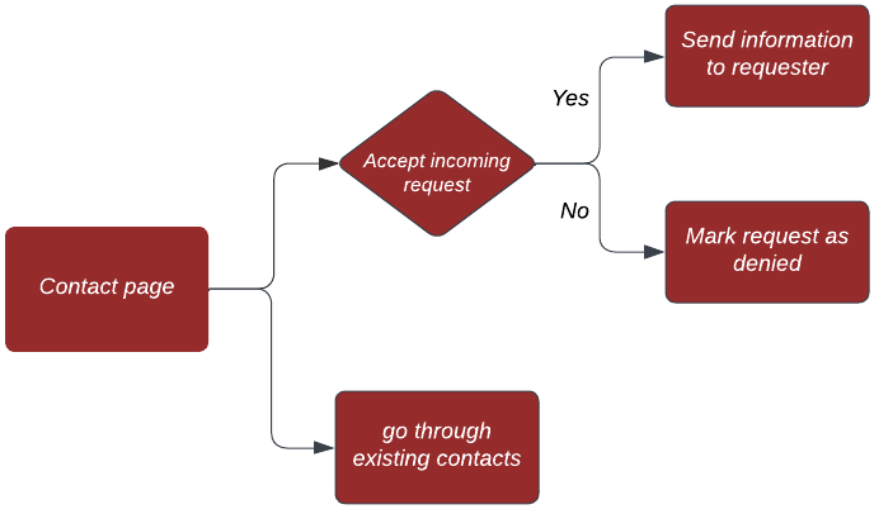
Signup/Login Flow :



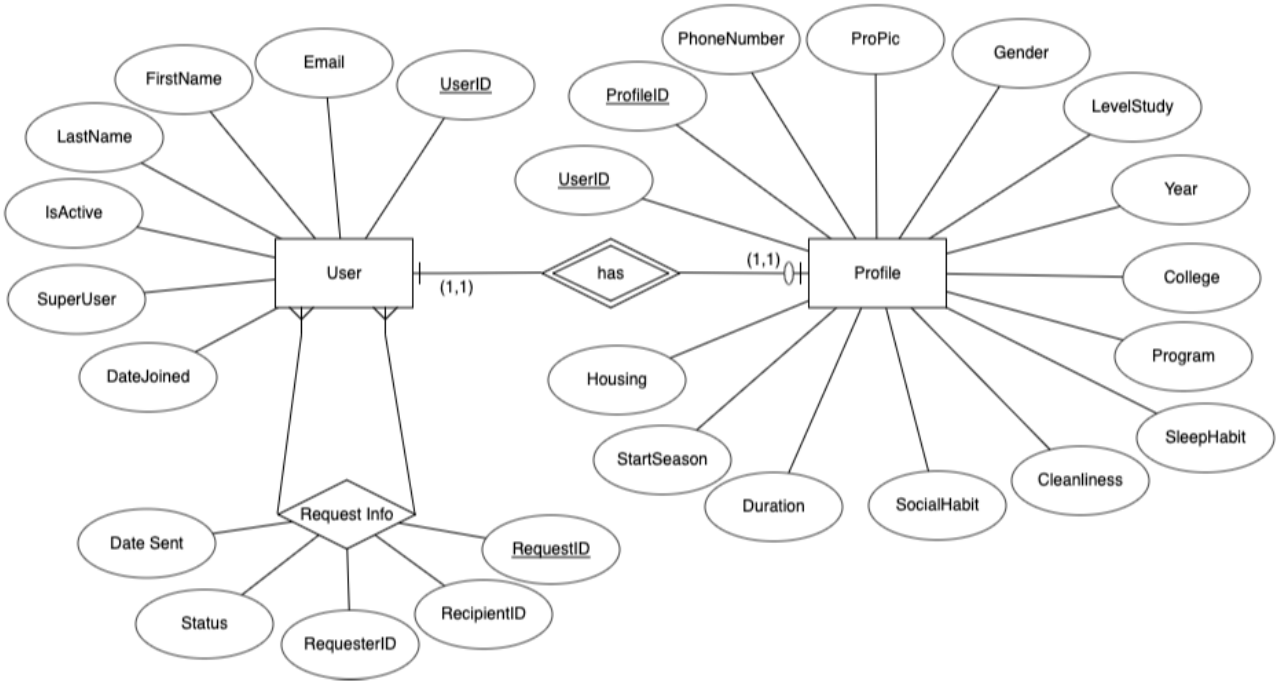
Search Roommates Flow :



Contact Information Flow :



Data Model


FirstName
Email
UserID
LastName
IsActive
SuperUser
DateJoined
User
(1,1)
has
(1,1)
Profile
PhoneNumber
ProPic
Gender
ProfileID
LevelStudy
UserID
Year
College
Program
SleepHabit
Cleanliness
SocialHabit
Duration
StartSeason
Housing
Request Info
Date Sent
RequestID
Status
RequesterID
RecipientID