Initial Design

Requestr

Overview

The purpose of Requestr is to provide a unified platform for MIT students to request help from other students. The reason behind the app stems from the fact that there is no known viable forum for MIT students to ask for help. There's piazza for academic help and facebook groups, but they are very inefficient (due to the fact that not many people use them) or people just go to office hours. But what about everything else besides academics? We feel that this app can facilitate stronger interpersonal relationships between students and ultimately build a stronger community through a shared helping app.

Design Essence

Key Concepts

Request

A request is the most significant concept of Requestr. It is motivated by the need to provide a consistent format for users to ask for help on the site. Users can submit a request with the necessary details needed for some helper to complete the request. Helpers can then view requests and decide whether they are able or willing to perform the request. For example, a user who wants help moving luggage into their dorm might submit a request detailing the location, timeframe, and maybe a reward for helping move his luggage in his request. Then someone can view the request, accept it and then go to help.

Tag

A tag is a way to filter requests. This aids helpers in looking through requests for ones they want to do. For example, a request for moving luggage could be tagged with manual labor which some helpers might not want to do and can filter out.

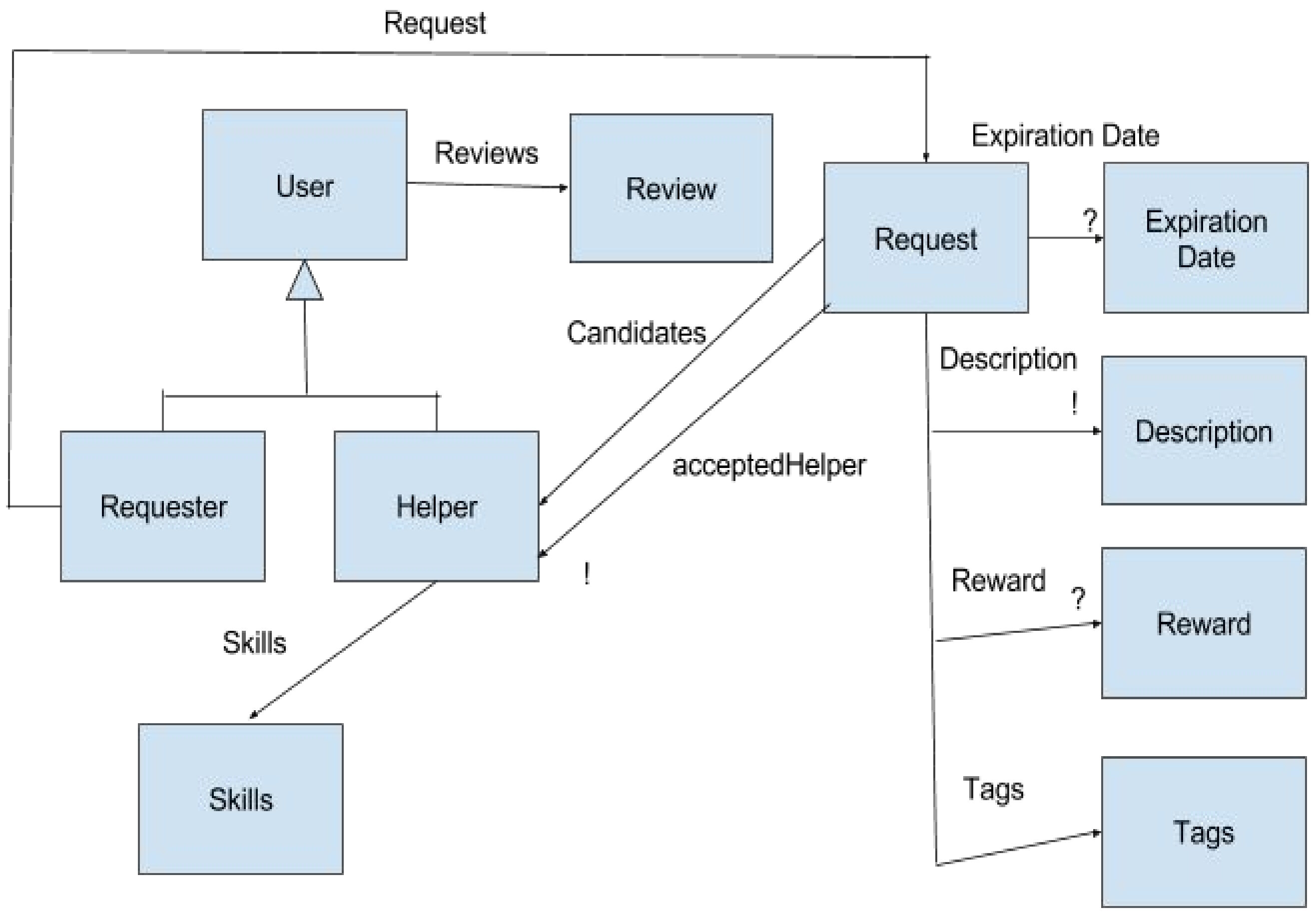
Review

A review for both users making requests and users taking requests to check the reliability of the opposite party. Whenever a request is completed, the user who made the request can write a review for the user who completed his request and vice versa. Then these reviews can be looked up when those users are involved in future requests to check how reliable they are.

Messages

A message provides a way for requesters to contact candidates to complete their request without giving away any personal information. When candidates accept a request, the requester can send them messages with more details if necessary for further clarification or vice versa.

Data Model



Textual Constraints:

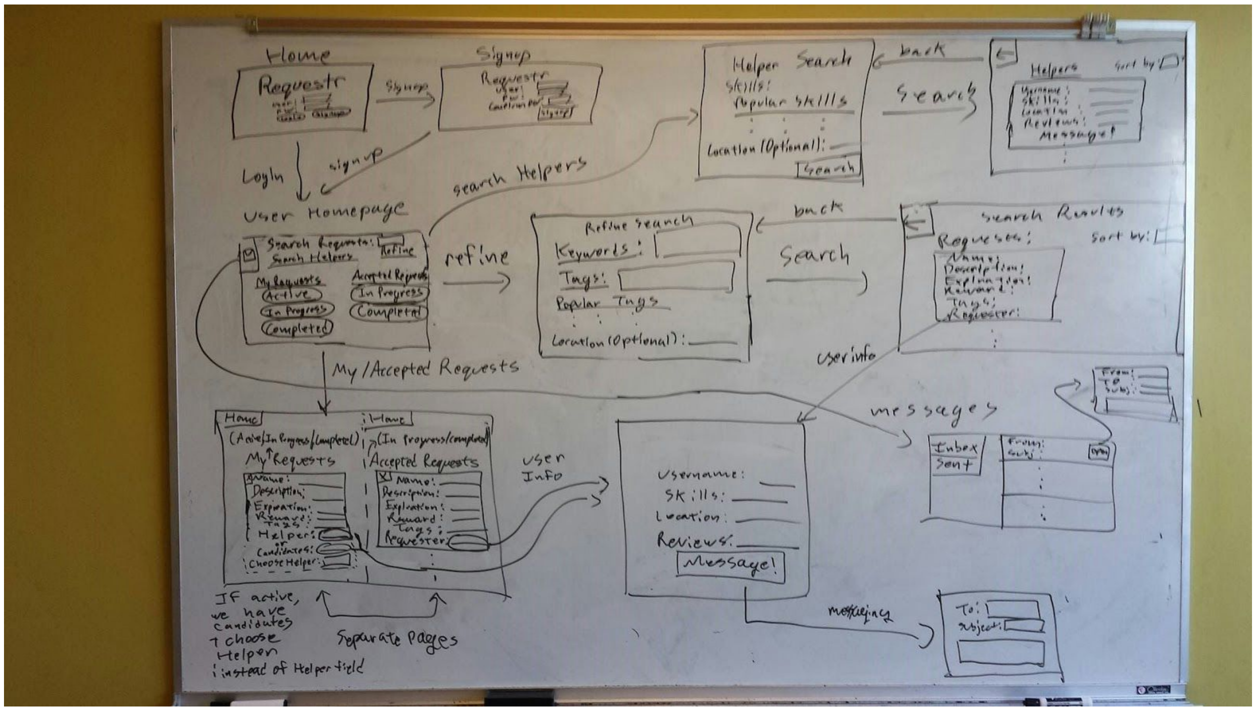
- acceptedHelper for a request must be one of its Candidates
- A helper for a request cannot be the user that made the request

Security Concerns

Obviously, first and foremost, we will ensure our forms are sanitized to prevent script injection into any input fields. That will stop most web attacks such as cross-site-scripting and sql injections. Secondly, we enforce that users who register must provide an email for verification in order to stop spam attacks.

Our web app doesn't operate on any sensitive data transmission, so the highest level of concern a user will have is that someone else will be impersonating their login and/or making/satisfying requests without their permission. Therefore, whatever security holes we forget to address (hopefully none) will not give attackers any leverage on their victims (besides poor reviews/ratings).

User Interface



Design Challenges

Tags

Should there be predefined tags to be used, or should users be able to make tags for their requests? What about tags that mean the same thing but use different words or are spelled differently? We're deciding to make tags user-defined, as a list of predefined tags runs the risk of being completely inadequate.

User Interaction

How do we model the interaction between requesters and the users who accept their request? There were many options for this. For example, we could have it so helpers put up their availability and what kind of tasks they are willing to do and then when someone makes a request they can just choose from any of the helpers that meet the requirements. This would force helpers to set aside blocks of time where they should be ready to be called on at any time similar to Uber drivers.

Alternatively, we model it with requesters posting their request so that other users can look through requests and easily choose the ones they want to do that are more convenient for them. This allows for more flexibility on both sides, but also makes it so it might be less likely for your request to be completed as there are no set of "helpers" that are guaranteed to be around to help.

We ended up deciding on the second idea since we do not want the site to become a job where helpers just sign up and wait to get money for menial tasks.

Rewards

How do we enforce rewards? One option is to restrict rewards to only being monetary. We didn't like this option much because we want the app to be more focused on students in the MIT community helping each other out rather than monetary gain. Instead we are considering using a review system for requesters as well to rate whether they consistently give the rewards the claim.