Project Idea: RateMyProfessor/Tinder for Roommates

This web app aims to help Duke students find the best-fit roommate. It asks each user a series of questions and try to give a list of most recommended potential roommates based on the answers provided by each user. The database needs to store the following information

- A user has a unique netid, a name and a profile picture. (Possibly with username and password if we are not using shib).
- A rating is a general rating about a specific user. A rating has a netid, an overall score, a score for cleanliness, for friendliness, for conscientiousness, for self-report accuracy, and number of times he/she has been reported.
- A review has a reviewer and a reviewed user. It also has text, overall rating, a rating for cleanliness, for friendliness and for conscientiousness.
- A report consists of a reporter, a reported user, and a reason for the report.
- A recommend consists of three users. A recommender (A), a user being recommended (B), and a user to whom A is recommending B (C). In short, A recommends B to C. It also has a reason for the recommendation.
- A question has a unique question ID and text content.
- An answer text associated with a question has an answer ID and its text. (E.g. Question: Which CS class is the best? 1. CS310 2. CS316. Here CS310 and CS316 are both answer texts and their associated answer ID are 1 and 2).
- An answer to a question by a user has 2 additional attributes. The user needs to provide his/her answer, and a weight indicating the importance of this question to him/her. Highly positive weight indicates wanting someone similar to you to be the roommate; highly negative weight indicates wanting someone different from you to be the roommate; 0 means don't care.

Relations (note: we will inevitably change these as necessary)

- users(netid, name, profpic) (username and password if we aren't using shib)
- ratings(netid, overall_score, cleanliness, friendliness, conscientiousness, self_report_accuracy, number_of_reports)
- review(reviewer_netid, reviewed_netid, text, overall_rating, cleanliness, friendliness, conscientiousness)
- report(reporter_netid, reported_netid, reason)
- recommend(recommender_netid, recommendee_netid, recommended__netid, reason)
- questions(qid, question_content)
- answer_text(qid, answer_id, text)
- answer(netid, qid, answer_id, weight)
    - Weight is 0 if you don't care, highly positive if you really want someone to have the same as you in something, and highly negative if you want someone with the opposite (an introvert who really wants an outgoing friend, z.B.)
    - Weight can be any integer from -5 to 5 (inclusive).

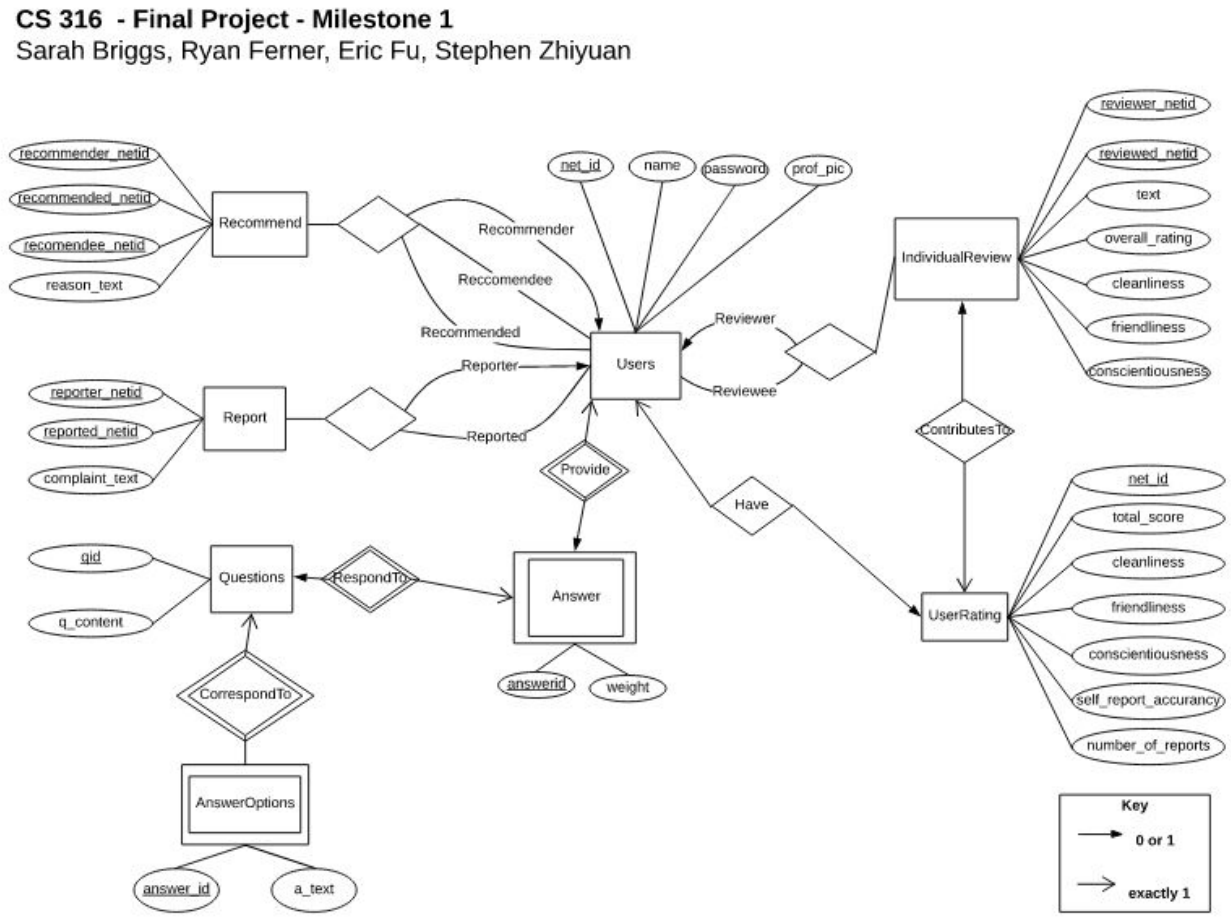
Data generation

We will randomly generate the data using some common names, with their cleanliness, friendliness, and conscientiousness, and question answers randomly generated. An early prototype of this process can be found in the “Data” file, with datageneration.py being the script used to populate the various xml files with fake data.

Assumptions

We will probably assume that these different traits that we will be analyzing are independently distributed during our random generation of people, even though even things like first and last name are hardly independent.

ER Diagram



Description of App

When the user opens the page, they are greeted by an authentication screen. We may choose to use Shibboleth authentication, since this is a service primarily for Duke students, or we may do authentication ourselves. A user can either log in or create an account, which will register them as a new user. They can also add their preferred name, a profile picture, and a short bio.

The user is directed to a questionnaire, where they fill out several different questions about themselves, including whether they would like to be matched to someone who answered the same or the opposite for each question, and how important that aspect is to them.

Once they fill out the questionnaire, they are displayed a list of people, in order, who most closely match their requirements. This will be calculated by summing their similarity on each question, weighted by each user's importance level, and filtering out the ones that are not applicable (a guy who wants to room with another guy will not be matched with a girl). If they want, they can filter by several different criteria, or they can reach out to another user to become friends. If the other user accepts, they can share some contact info and talk on their own.

If a user is being abusive, they can get reported. If they get reported, they can be banned, in which case they are no longer able to log in and use the service.