TEAM NAME: Edgy Fireflies
ROSTER: Grace Mao (PM), Tammy Chen, Joseph Lee, Jun Tao Lei
TITLE: Read and Chill

FRONTEND FRAMEWORK: Bootstrap

Bootstrap and Foundation both offer similar resources and abilities in terms of page organization, so because our group members are most familiar with Bootstrap, it was clear that we would lose nothing by choosing the one that we are most comfortable with.

DESIGNATED ROLES:

Project Manager: Grace Mao

Frontend: Tammy Chen, Grace Mao

I. Template creation and management, with interaction provided (TC)
   A. Login mechanism
   B. Connecting to and retrieving from database
   C. Display of each page with framework (with GM)
II. Addition of Javascript where applicable (GM)
   A. In later stages

Backend: Joseph Lee, Jun Tao Lei

I. Database management with sturdy relations and key pairs
   A. Connecting users based on common traits and books
   B. Chat history retrieved easily
   C. Book information retrieved from API and stored
II. Connection to the frontend through SQLAlchemy
III. Algorithms to match users and recommend books

DESCRIPTION

The objective of this project is to create a Tinder-like web application for book lovers to communicate. The application will allow users to meet new people with similar genre/book interests and bring people together (at a reasonably safe distance). Users can swipe left/right on other users just like the Tinder app & message people in real-time, with the use of Web Sockets. Users can also browse through a variety of book selections with the help of the Google Books API.

After signing up for our app, users will be able to customize their experience through their profile, creating a book list for themselves, setting preferences in match, and such. In particular, users will want to enter their preference of genres, as well as who they're looking for on the site. They can also upload a picture of themselves, and fill in a

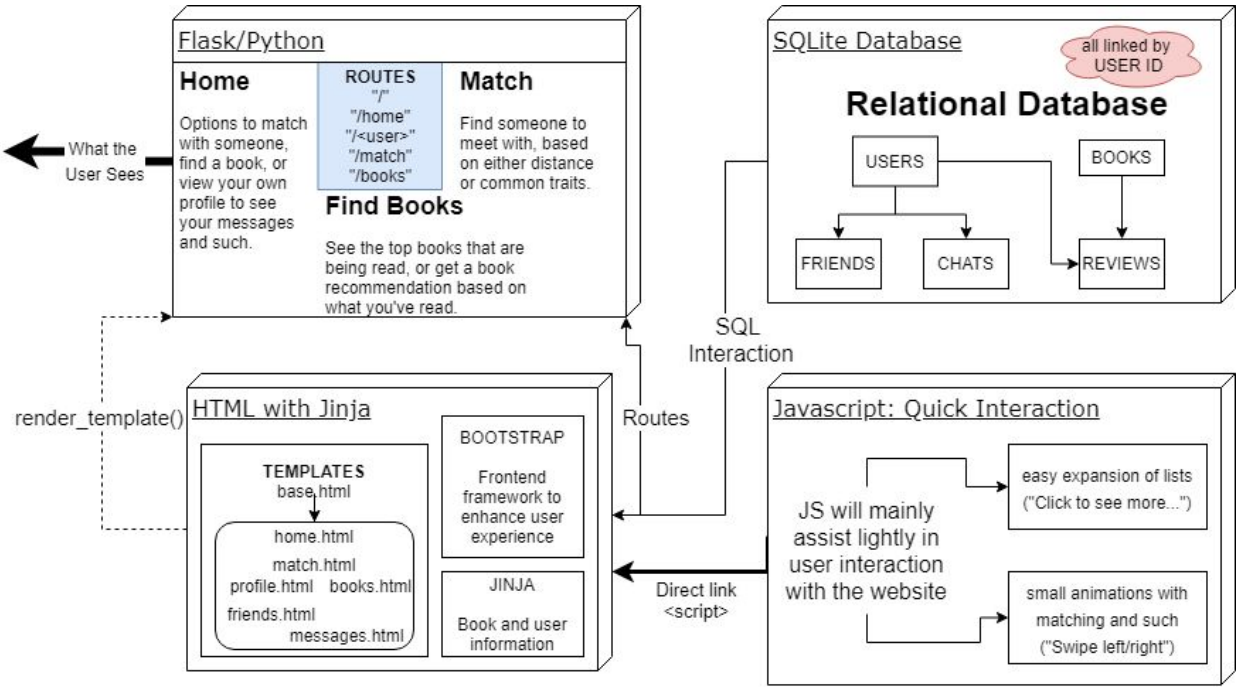
reading list with what they've read recently and what they're currently reading. They'll be able to see who else is reading that book.

Now they're ready to match with someone! Our matching algorithm will display a profile of someone else based on either distance or common traits (the user gets to pick). In terms of common traits, users will be presented with people who have similar interests with them, such as genres of books or reading history. Bonus if they are currently reading the same book! The user then gets to choose if they'd like to match or not by swiping left or right. If not, a new match will be made and someone new will be displayed.

After matching with someone, they will automatically be added to your friends list, and vice versa. You'll then be brought to your chat history, in which you can now send them a message. The user will be able to access all past chats and matches from their profile, which they can also edit as they go on their journey with our website.

In our book section, users can see the most popular books being read or whatever we recommend for them past on their reading list. Books are all fetched from the Google Books API, and displayed for easy readability. Our relational database links books and users by ID and foreign keys.

COMPONENT MAP:



DEVELOPMENT STAGES:

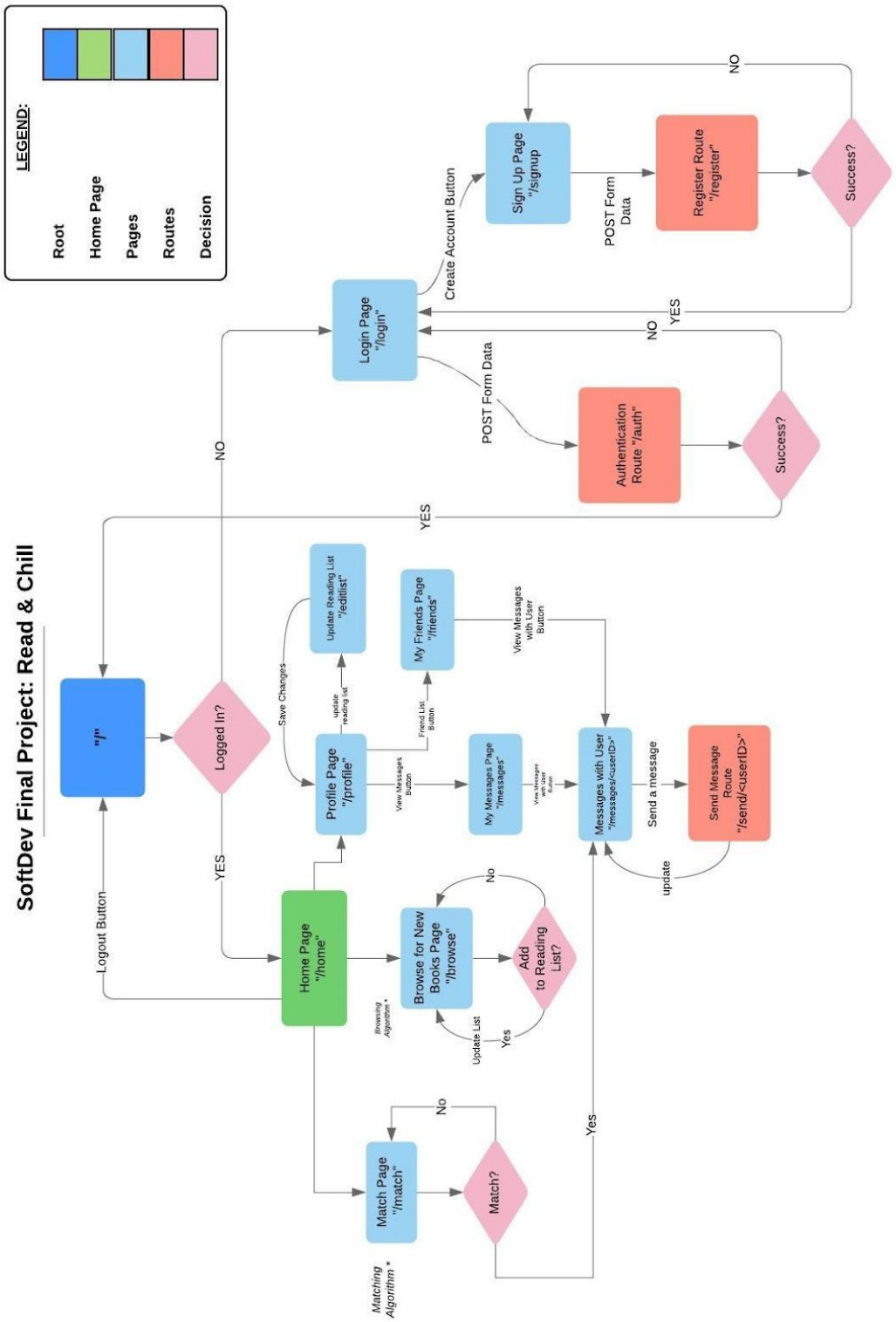
- Minimum Viable Product:
    - Backend:
        - Build the database schema & set up the relational mappings
        - Create a way to store messages in either plaintext or database effectively
        - Matching algorithm as well as book recommendation algorithm created for easy usage (nothing to be hardcoded)
    - Frontend:
        - The user is able to create an account & log in
        - Flask routing is completed & templates for each page are made
        - Search functional & connected with Google Books API
        - Users able to match as well as edit their profile and see the most popular books
- Expected Product:
    - Users are able to communicate with one another via messages (mostly similar to emails, less instant due to functionality restrictions)
    - Users are able to see add/delete friends (functioning match algorithm)
    - Javascript somewhat implemented to enhance user interaction
- Extra Features:
    - Users are able to add reviews for books on their reading list.
    - Check out what books people reading the same book as you and form a book club

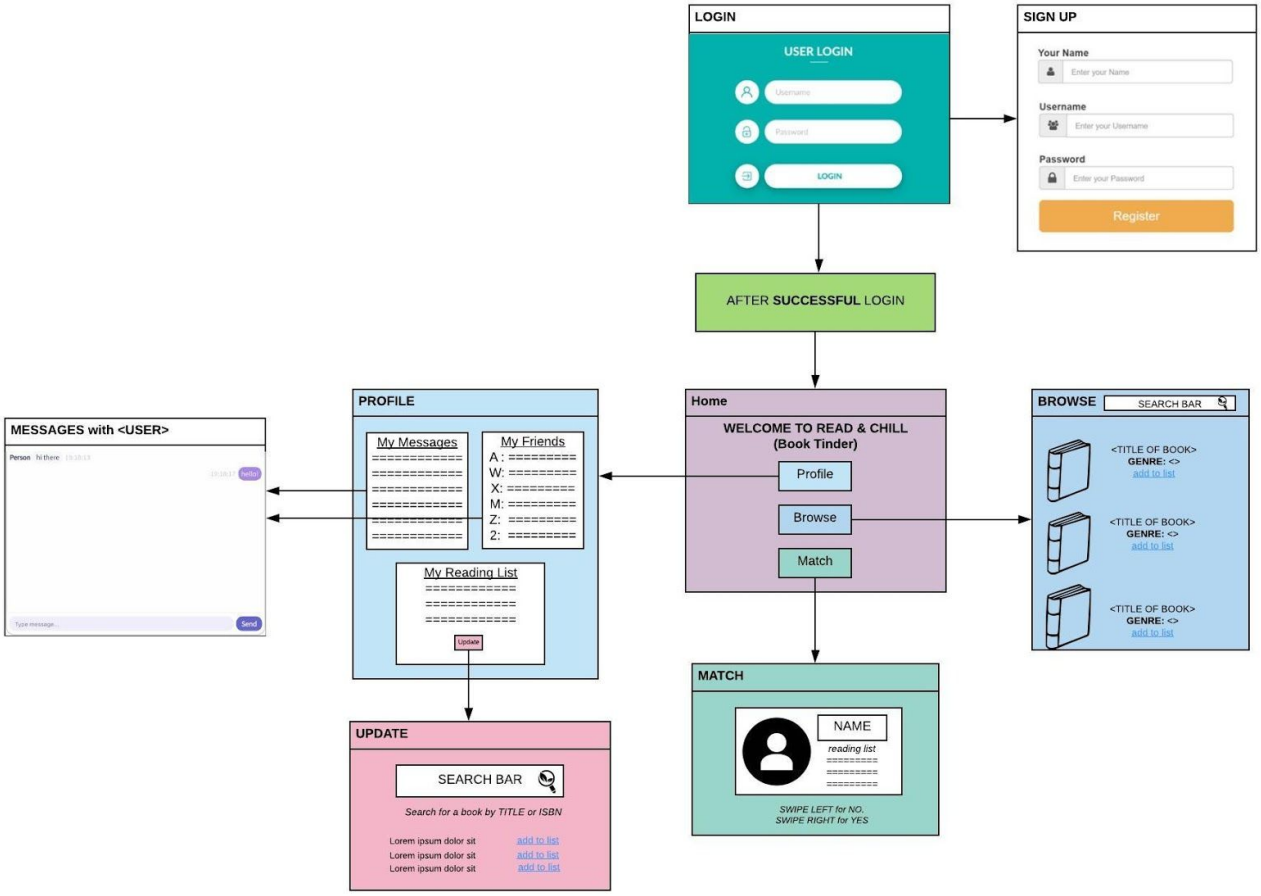
MATCHING ALGORITHM:

1. The user should have favorited books with ratings and written reviews.
2. Get all other users that have the same or a subset of the user's favorite books.
3. Get the user ratings and written reviews for all other users that meet the criteria from the previous step.
4. Perform semantic analysis on all written reviews.
5. Construct vectors that compare the original user to all other users for each book.
6. Calculate the angle or the distance difference between the vectors for a measure of "user closeness".
7. Sum the angle or the distance differences and multiply it by the number of shared favorite books since the other users could have fewer books than the original set.
8. Return the users in order of lowest score.

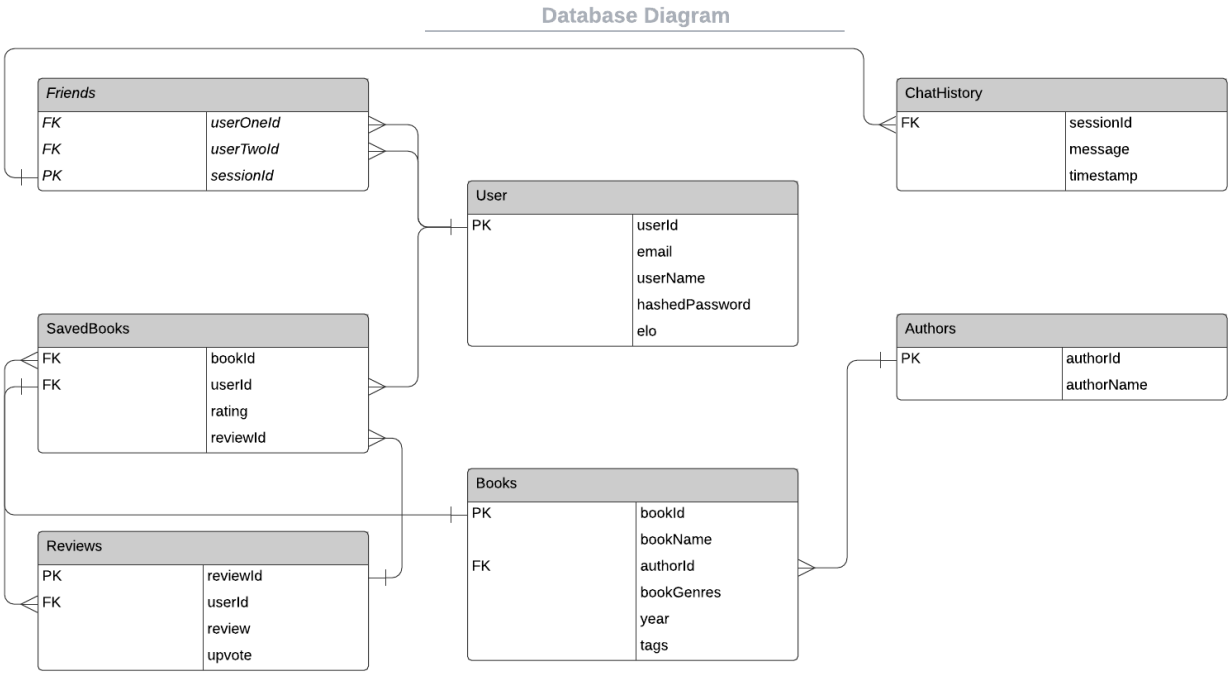
SITE MAP:

https://app.lucidchart.com/invitations/accept/6591e966-e091-4550-ba39-97c3866e36de





DATA ORGANIZATION:



CONNECTING THE BACKEND AND FRONTEND:
This application essentially follows the model-view-controller architecture. Each feature of the application will have view(s) that the users can see like the rendered HTML templates. The user can send a POST request to a route that will return fetched data or modify the database by invoking SQL statements using predefined and mapped models.

LIBRARIES AND APIS:

- flask-socketio
- flask-sqlalchemy
- textblob
- Google Books API:
    - This will allow users to look up books by searching titles, authors, and ISBN numbers.

KEY ISSUE: How to keep people from stealing books or logging bad reviews?

- Create a system where people that have a history of stealing books would be banned.
- Create a system where both lender and borrower have to check off before lending and returning a book. (There is no way to penalize someone unless you impose a fine, which is out of the scope of this project.)
- Use an ELO system to rank reviewers by reputation (Rank people by the number of helpful or unhelpful upvotes for the individual's written review).