Project Proposal

Coleman Gibson, Derek Kuhnert, Kieran Groble

March 31, 2017

1 Features

1.1 Searching Users

A main feature of our application will be to allow users to search for other accounts based on multiple properties, including name, interests (e.g. Vim or Emacs), and location. This will potentially result in a sort of user browser, which users can use to find potential matches.

1.2 Rank User Matches

Something which could be used in numerous parts of our application is the ability to rank user matches. This could be done off a number of factors, including user interests and their preferences in what they are looking for in other users, such as giving operating system preferences a high weight.

1.3 Basic Messaging

To make matching with other users useful, users of our app must be able to contact their matches in some way. We plan to do this with a basic messaging system.

1.4 User Caching

To make our site faster, we hope to implement a simple form of caching for user profiles. Users are likely to search for others far more than they will update their own profile, so we plan on storing a preconstructed representation of our users in a database with excellent read speeds. This representation will then be invalidated on any updates and reconstructed on request.

2 Tools

2.1 Front End

2.1.1 React

All three of us in the group are interested in learning React.js. We will use this with JavaScript, HTML, and CSS to make a basic user interface to our database project. It would be fairly unreasonable to make a command line dating application, no matter how nerdy our users are.

2.2 Back End

2.2.1 Python

Python is a language which we know all of our databases support well and is a language all of our group members are familiar with. This makes it a good choice for our primary back end language.

2.2.2 Flask

We believe that a larger framework like Django would be overkill for a three week project, but would not like to write our web server with no support. We think that Flask will give a good middle ground for what will be a relatively small project.

3 Databases

3.1 ArangoDB

This will be our main store for any data involving relationships. For example, we will use this to store things like user operating system preferences and matches between users. The graph structure this database gives us will make it extremely easy to do things like match user preferences and check the number of shared matches two users have.

3.2 Redis

We will use Redis as a basic store for noncritical data like user sessions in our site. We would also like to experiment with using Redis as a user cache so that we can serve documents constructed from ArangoDB and MongoDB without reconstructing them from the databases.

3.3 MongoDB

MongoDB will be used for storing user profiles (including name, age, description) as well as the chat data for our basic messenger. In short, it will be used to store data which does not have any relationships. This will give us some of the write and read speed benefits of Mongo where we can use them. We wont lose the ability to store relationships entirely as we will still be using ArangoDB for our relational data.

4 Visual Representation

4.1 ArangoDB

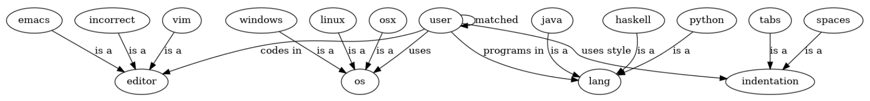

Figure 1: ArangoDB data model

4.2 Redis

```
<username>-session = user-session
<username>-cached-document = user-document
```

4.3 MongoDB

```
user {
    username,
    name,
    birthday,
    description
}

chat {
    user1,
    user2,
    text
}
```