* The problem description, motivation, and survey of related work as in the project proposal, but more detailed and refined.

How many would-be relationships never happened because both parties were too shy to start the conversation? Overcoming the activation energy associated with making the first move can be quite a tall task, and countless opportunities may have slipped away, especially in today’s fast-paced society. Our application seeks to address this issue and facilitate matchmaking among shy individuals who already know each other by providing a way to anonymously voice feelings. A user can send a “crush” message to another user provided both are connected as friends. The recipient will be notified of and have access to the anonymous message, but will only be able to see the sender’s identity if a “crush” is reciprocated. Anonymous messages thus serve as incentives for recipients to express their own feelings. More importantly, the anonymity of the messages will encourage shy individuals to express their emotions, thereby catalyzing the formation of meaningful relationships.

Most modern dating applications focus on bringing strangers together, whether firsthand or via a third party.1 Tinder and OkCupid all allow users to locate other nearby users, explore their profiles and express their preliminary affections in the hopes of finding a match. PlentyOfFish achieves the same goal but spans over 70 million users across the entire English-speaking world. Bumble is similar to Tinder except that only women can send the first message, and Grindr is a dating application geared towards gay men. Hitch allows the user to provide suggestions to set other users up with each other. Although these dating applications primarily focus on connecting people for romantic purposes, relatively few applications currently give users a chance to connect with people they already know. Crush strikes an interesting balance by giving users autonomy over their own interests without stepping directly into the online dating sphere. Rather than promoting more random and potentially short-term relationships or hookups like most other dating applications, Crush aims to facilitate personal matchmaking among users who are already friends. It also supports dating for people of all sexualities. It provides an online and easily navigable interface where users can find friends and express differing levels of interest. Not only does Crush provide a comfortable non-confrontational avenue for connecting friends in a more intimate way, but it also eliminates the modern stigma associated with online dating.

* An in-depth discussion of your system, including the design choices you made.

The overarching structure of our system is: Angular 🡨🡪 Node (Restful) 🡨🡪 PostgreSQL.

We chose PostgreSQL for our relational database management system because it is convenient to use and offers more features than other common relational DBMS such as MySQL. The details of our database implementation can be found in the FinalProjDatabase4.sql file. The database consists of five main tables: UserInf(uid, name, password, gender, email, birthday, phone, city, joindate, commitLevel, interestedIn, profpic), UserInterests(uiid, interests), Notifications(nid, nFrom, nTo, ts, text), Friend(fid1, fid2), and Relationships(user1, user2, isReciprocated). The table NotifState(sid, seen, seenTS) has not actually been currently implemented but may be used in the future to give the sender information about when their notification was viewed by the recipient. The UserInf table currently has constraints on what attributes can be entered into gender, commitLevel and interestedIn. These constraints keep responses uniform and facilitate matching for suggestions. We initially had the attributes “interest1,” “interest2” and “interest3” in the UserInf table, but we decided to create a separate UserInterests table for easier access and query manipulation, primarily for matching interests for suggestions. The Notifications table holds information about sender, receiver, time and the message itself, which can only be up to 400 characters. The friend table is pretty self-explanatory and includes pairs of friends who are connected to each other. The Relationship table stores information about users who have previously sent messages to other users as well as the reciprocation status. Triggers are in place to address the following cases: 1) if a user is sending a message to another user, and neither user has sent messages to each other in the past, then a new tuple consisting of the two users will be created where isReciprocated is set to “false,” and 2) if user A is sending a message to user B who has previously sent a message to user A, then no new tuple is added, and the isReciprocated attribute in the corresponding tuple is updated to “true.” Once this happens, a notification, which manifests as the users appearing in each other’s “Successful Crushes” section on their homepages, is then sent to both users alerting them of a match. These five main relations currently capture all the relevant information needed for the application to run properly.

We initially planned to write the program in Swift but later switched to Angular to accommodate time constraints and feasibility. We decided that a mobile version of the application would not be able to be completed within the time frame because of lack of familiarity with the Swift language for our members. Therefore, a web platform was much more feasible and also within the goals of an easily accessible platform.

Front-end separated CSS, JS, HTML files as well as Node.js and Angular parts. This was to ensure improved readability and reduce repeated code in case multiple files wanted to utilize the same CSS or JS. Because Angular provides two-way data binding, one design choice we made was to reduce and minimize the amount of hard-coded text into our files. Rather, we utilized scope to implement easily changeable components. This meant that we didn’t have to create a new profile.html page for every single user but rather, the data was just swapped out depending on whose profile we were looking at. Moreover, we used ng-show to make a profile page different if it was the active user or if it was a random profile the user was looking at. This meant that we could use the same HTML file for both purposes.

There weren’t too many advanced design choices we made in the front-end. Rather, we focused more on designing an efficient and organized system with our technology stack. All the technologies were new and so ensuring quality communication and interactions between Angular, Node, and PostgreSQL was just as important as any design choice within the code itself.

* Detailed description of any new approaches or algorithms that you are developing.

The primary novel approach developed in Crush involves the anonymous notification system. Although not incredibly sophisticated, the application must ensure that “crushes” sent from one user to another for the first time give anonymous notifications to the recipient, whereas “crushes” sent as reciprocations deliver notifications to both users that a match has been made. This is handled primarily in the SQL code by checking the isReciprocated attribute of the Relationships table. If this attribute is false, then an anonymous notification is sent; if the attribute is true, then both parties will be notified of the match.

The other important algorithm we implemented provides users with other suggested compatible users. For a given user, the UserInf database is first filtered for those users who mutually match in their sexuality interests as well as in their commitment levels. This list of users is then further filtered for those who have overlapping interests, with users having more overlapping interests suggested first. The algorithm is reflected in the Suggestions.sql document.

* Evaluation of your system, and if applicable, comparison with competing systems. Be clear about what your evaluation metric is. If you have experimental evaluation, describe the experimental setup in enough detail so that others can repeat your experiments.

Because our application currently does not support real users, it is difficult to offer an objective evaluation of our application success. At best, we can provide a subjective projected expectation for our application.

Once our application is refined (as described in the future directions work below) and successfully introduced to the public, we expect that our application will primarily target people in the age group between 15 and 35. Because of its unique implementation, we expect that our application will attract many users regardless of whether they have used or currently use any other dating applications.

Unlike the audiences targeted by most other dating applications such as Tinder, we expect our users to be primarily interested in more serious relationships rather than hookups. Because legitimate relationships tend to rise on most people’s priorities over time, we expect our application to be competitive on the dating application market. With very few other similar dating applications available, we expect Crush to become popular in mainstream society. With number of users as the evaluation metric, Crush should be relatively high tier.

* Any open issues or directions suitable for future work.

One major aspect of the application that needs to be addressed is designing and implementing a viable heuristic for preventing users from abusing over-messaging. Having users who send meaningless “crushes” to multiple people just to humor themselves or to find out who sent them messages will undermine the validity of our application. Our intention is to provide a platform for users who have true feelings for others to comfortably express them. Thus, eliminating the possibility of ill-intentioned users is of utmost importance. A possible heuristic for controlling against misuse is setting an upper limit on the number of messages that can be sent within a given period of time (i.e. only one message may be sent every two weeks). The heuristic should be ideally chosen to combat the problem while also not being so restrictive that users stray away from our application.

Other areas of improvement include adding more refined constraints for certain attributes in our database relations, such as more “gender” and “commitLevel” options in UserInf and a limited or predefined set of “interest” options in UserInterests to ensure matching. We could start with a preliminary set of interests that users could enter and later incorporate machine learning algorithms to add more interest possibilities for our drop-down menu. We could also group certain similar interests together to facilitate interest matching. For example, the first level of interest matching could be verbatim (two users both like chess), but the second level of interest matching could be categorical (user A likes chess and user B likes checkers, so both like board games).

The front end could be improved by separating our API built on Node.js into multiple controllers. This would help in terms of scalability if we were to ever continue building our application to its full potential. Separating out our complex code in app.js and routes.js, which hold our API and connections to our database system, would improve readability to aid in future development. Moreover, including error checking mechanisms and more streamlined UI/UX in our Angular/HTML/CSS code would drastically improve the overall front-end appearance and functionality. Error checking when updating settings or displaying suggestions with more than just names are both examples of how the front-end could be improved in the future.

Once the front end of our application is refined, our next major step would be to connect users via Facebook to pull their friendship information. This also explains why our current application does not support a friend-searching feature. Once we allow users to connect via Facebook, any new Facebook friendships will automatically be reflected in our database system. Not only does this take the hassle out of getting started with Crush, but it will also officially introduce the application to mainstream society. Users will feel more comfortable in using our application for advancing their relationships lives. Finally, Facebook streamline processes in our application such as changing the profile picture or importing more interests from Facebook user information as well.

Other future goals include developing a mobile iOS and/or Android version of our application so users have easier access to Crush.

References

1. <http://www.tomsguide.com/us/pictures-story/639-best-dating-apps.html>