Department of Electrical and Computer Engineering

Shadkan

Your Private Match-Making App

Unlike the Other Dating Apps:

Team Members:

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yehuda Cohen</td>
<td><a href="mailto:yehudac@gmail.com">yehudac@gmail.com</a></td>
</tr>
<tr>
<td>Amizan Jaleel</td>
<td><a href="mailto:amizan.jaleel@rutgers.edu">amizan.jaleel@rutgers.edu</a></td>
</tr>
<tr>
<td>Rohit Lakshmanatirthakatte</td>
<td><a href="mailto:rohit.lakshmana@rutgers.edu">rohit.lakshmana@rutgers.edu</a></td>
</tr>
</tbody>
</table>

Date: May 6th, 2014
Semester: Spring 2014
Advisor: Professor Shantanu Jha
Supervisor: Professor Manish Parashar
Area: Software Development
Course: 14:332:438:01
Table of Contents:

1. Abstract........................................................................................................................................3
2. Introduction/Overview......................................................................................................................3  
   a. Barriers Identified....................................................................................................................3  
   b. Solution....................................................................................................................................3  
3. Approach/Methods/Results............................................................................................................4  
   a. System Architecture................................................................................................................4  
   b. Implementation/Experimental Results.....................................................................................6  
4. Cost/Sustainability Analysis...........................................................................................................9  
5. Conclusion/Summary.......................................................................................................................10  
6. Individual Contributions................................................................................................................11  
7. Bibliography..................................................................................................................................13
1. Abstract

Shadkan, is a platform designed to encourage friends to connect romantically. We identified several barriers that prevent friends from connecting with each other romantically in the real world and existing dating platforms, and worked on eliminating these barriers through technology. In our design, we believe we successfully eliminated the most significant barriers.

2. Introduction/Overview

Shadkan was built for the purpose of solving the following barriers:

Barriers Identified

1. People will typically not approach their friends to express romantic interest for one of two reasons:
   a. They are shy.
   b. They fear that they will ruin their friendships if their friends do not reciprocate their feelings.
2. People are frequently attracted to multiple friends simultaneously - with varying degrees of interest. They do not, however, wish to start relationships with multiple people at once.
3. People are (rightly) skeptical of algorithms that match them romantically based on mutual interests and other attributes. After all, people are far more than the sum of their interests. They know their Facebook friends, and can identify their interests for themselves.
4. People do not want to be matched with someone when they have just begun a new relationship. This would be especially uncomfortable if they were matched with their friends.

Solution

Shadkan is built as a double blind system, where friends can privately express romantic interest in each other, without revealing their interest to each other. This is achieved through a central server that collates each user’s interest in their friends in the form of ratings. That is, users can rate their friends without expressly telling them how they feel.

This double blind system allows the central server to detect whenever two people are interested in each other. This solves the first problem. Friends no longer need to be shy, or fear that they will ruin their friendships by expressing romantic interest.

In order to capture the granularity of a person’s interest in a friend, we provide them with a granular scale to rate their interest in their friends. We also allow them to set their availability to prevent them from being matched up inconveniently. Additionally, we are able to calculate a measured compatibility for each user, and determine a user’s best match. Every time we make
a match, we set their status to unavailable until they make themselves available actively once more. This effectively deals with the second and fifth barriers entirely.

We employ an algorithm that is entirely dependent on mutual ratings between friends. This avoids the pitfalls of matching people up based on user attributes entirely, and leverages the users own personal interest to full effect. To fully understand how our algorithm achieves this effective focus on the user’s interest, consult the implementation portion of this documentation.

3. Approach/Methods/Results

System Architecture:

Shadkan is built upon cutting-edge technologies, frameworks, and design paradigms. Components were selectively chosen and configured to provide the ideal experience for the user. It has been packaged into 3 major components: a frontend for user interaction, a backend for executing algorithms, and the database for storing user data.

Frontend:
The frontend packages an AngularJS/HTML5 mobile app using Cordova and SteroidsJS libraries to access native phone APIs. The HTML5 nature of the app allows for easy transportability between different mobile platforms. The app could even be run from a web browser. AngularJS allows the developer to structure the app in the well-known M-V-C style framework (Models, Views, Controllers). SteroidsJS further structures the app by separating code files into different folders, such that code files written in different languages are not mixed in together.

This frontend also incorporates a small, lightweight framework called IonicJS, which is written completely in HTML5 and CSS. This powerful framework provides a selection of beautiful user interface components for the user and easily captures user input. It successfully ensures that developers rarely need to worry about writing custom CSS code.

The final and most important component of the frontend is Facebook’s Graph API. This is where Shadkan retrieves all of the user’s profile information, including his or her friends list, and even their information.

Backend:
The backend is built on a NodeJS core with a PostgreSQL persistence layer buried beneath our Object Relationship Mapping tool, which serves as a middle layer to allow for the database management system to be swapped out easily, if necessary.

A RESTful API called Restify provides services that allow for the frontend to talk to the backend, efficiently and asynchronously. These RESTful services are also used to make frequent communication with Facebook’s Graph API to acquire the user’s information.
All data passed between the frontend, backend, and the database is structured using JSON blocks. Since Shadkan is written entirely in Javascript, JSON was the obvious choice to package data and for easy parsing.

Database:
Shadkan uses PostgreSQL to manage large quantities of data. Here, Shadkan separates data into 4 tables: User table, Preference table, Rater table, and the Match table.

The User table stores all Shadkan users including their information and even their Facebook friends.

Each Shadkan user is mapped to a set of preferences, represented by a row in the Preference table. This table simply stores the user’s match and alert preferences and their availability status.

The Rater table records each user’s rating value when he or she actually rates a friend as dateable.

Two entries in the Rater table, representing two users who have rated each other, results in a match entry in the Match table. The Match table records the actual compatibility index between the two raters computed by the backend.
Implementation/Experimental Results:

Shadkan was implemented entirely in Javascript, both frontend and backend. Our algorithm is dependent on users’ ratings of their Facebook friends and finding mutual interests through such ratings. As such, one of the first user stories we focused on was implementing a Facebook login that would allow users to log into Shadkan via their facebook username and password. This login system would also give our app access to their Facebook information and friends lists. We use Facebook’s Graph API to obtain a user's Facebook access token, and we pass this access token in requests between frontend and backend to ensure secure communication for each user.
Once a user has logged in, they are free to rate their friends in order to determine a match. When a user goes to Shadkan’s Rate My Friends feature, we use their access token in a call to Facebook’s Graph API in order to retrieve all of the user’s friends and their respective profile pictures. We then filter this list of friends according to the user’s match preferences. For instance, if a particular user is only interested in women aged somewhere from 20-25, we’ll only display his friends in that category. The user is asked “Would you ever date ####?” for which they rate on a discrete scale if they press “Yes”, or press “No” if they are not at all interested (which we store as a rating of 0). In addition, every time a friend is shown on the screen, Shadkan checks to see if that friend is stored in the database (user table). If they are not in the database, then Shadkan stores their information and adds a timestamp (Added Date). The friend becomes a user, if he/she later joins Shadkan, for which we store their Facebook access token and add a “Join Date”.
Once two raters have rated each other, we store that as a potential match in our database. A match is defined by 2 users and their Compatibility Index, which we calculate as the product of the two ratings of each other. For instance, if John rates Amy a 7, and Amy rates John a 5, then the John-Amy compatibility is 35. This would also mean that if one person is not interested in the other person, Shadkan will always output a compatibility of 0 between the two, no matter how much the other side rates the former.
In addition to this, each user has their own “Desperation Threshold” value that we use when determining matches. Also, each user has a “Currently Available” checkbox (stored as a Boolean value), that they can disable when they don’t want to get matched with anyone at the moment.

We run the match algorithm when the user selects “Match Me Up”. It begins by fetching all match pairs in the database that this user is a part of. We discard from this array any matches with a compatibility below this user’s threshold. We then sort the array by compatibility, and then select the match with the highest compatibility, as long as it is also above the threshold of the other person, and that person is available. We then notify the users of the match, and make the requesting user unavailable (by unchecking the “Currently Available” status). We make only the user who requested a match unavailable because we do not know whether the other user will accept the resulting match.

Below is a workflow diagram of the user interacting with Shadkan.

![Workflow Diagram]

Hence, Shadkan enables the two friends to embark on their romantic journey together.

4. Cost/Sustainability Analysis:

Our goal was to complete this project with no material cost other than the cost of the webserver running Shadkan. Currently, Shadkan sits on a webserver that cost $35 as a one-time fee, and no other costs were incurred.
In hindsight, it might have been valuable to buy certain components which took a long time to manually write, such as facebook connector components supported by the vendor of the open source platform SteroidsJS that we used to package our application. Nonetheless, in building Shadkan, we demonstrated that it is possible to build a viable product with extremely little capital.

We have received great interest investment in our product, and if we had the time, we would spin off a startup based on this project. For now, though, we will support Shadkan as a side hobby.

5. Conclusion/Summary:

We have faced several challenges along the way of our implementation, perhaps the greatest of which was integrating all the different frameworks and technologies we used, illustrated in the architecture diagram above. On the one hand, this gave us great portability. Our NodeJS server could be run on both Windows and Linux, our frontend app runs on both iOS and Android. However, this came at the cost of us needing to figure out how to get these frameworks to communicate with each other.

In hindsight, perhaps it would have been easier and faster to use a predefined framework that preconfigured even some non-ideal components instead. The configuration time in this project was extremely steep, and saving this time would be tremendous.

On top of that, some of these technologies are relatively new and not very well-documented. As such, we spent hours reading through incomplete API documentation to find the functionality that we needed. However, using cutting-edge technologies and APIs came at a disadvantage of smaller communities, and pervasive bugs that need to be repaired. Hindsight would have us re-examining the choices we made.

Ultimately, however, we judge our project successful, and we have achieved the portability and functionality we sought out to achieve.
6. Individual Contributions

Yehuda Cohen:

- Initial product idea and naive high level design
- Designed metrics used to determine compatibility and heat index
- Configured all different components to work together, and provided naive examples for team demonstrating usage:
  - Frontend:
    - Configured Angular.JS app structure within steroids framework.
    - Integrated Ionic UI framework into frontend.
  - Backend:
    - Configured Restify server to connect with database via Node-Orm2
    - Separated structure into model-controller framework using Node.js Require() module to connect separate pieces.
    - Configured necessary HTTP headers to allow frontend to talk to backend.
- Built out facebook connect API:
  - Allows a user to hit an endpoint on our server with facebook id and access token and sign up to Shadkan.
- Built out frontend security suite (Currently disabled due to vendor Steroids.JS bug on Android platform that causes app to crash).
  - Integrated facebook-phonegap API to allow our mobile app to communicate with native facebook app.
  - Interrupts a user as soon as it doesn’t recognize him / her, and requests facebook connect.
  - Facebook invite module allows users to add facebook friends via facebook dialog after login.
  - Intuitive API that can be added to any angular module very easily.

Rohit Lakshmanatirthakatte:

- Implemented the entire Rate My Friends feature (both Frontend and Backend).
  - Also acquired initial user ratings for each friend so user can see what they previously rated them as.
  - Implemented Heat Index so that each friend shows up onto the screen according to global popularity.
- Provided all frontend code (using IonicJS) for Welcome screen, Dashboard, Match Preferences, Alert Preferences, Desperation Threshold, Rate My Friends, and some portion of Match Me Up.
  - Acquired default values for the user’s preferences from the backend so user can know what they had before and change them accordingly.
- Added a new javascript file to define server ip/port, user’s facebook id and access token, which are global to all files.
- Wrote all necessary error handling code backend and frontend.
● Implemented the email alerts for both users after a user requests for a match.
● Fixed a small bug that initially gave blank alert messages in Match Me Up.
● Commented (including the headers on the top of the file) all code backend and frontend except for frontend facebook connect.
● Implemented the following major modules in backend:
  ○ userController.getName
  ○ userController.getShadkanId
  ○ userController.getFriends
  ○ raterController.sendRating
  ○ raterController.postRating
  ○ userController.storeFriend

● For deliverables:
  ○ Drew up the initial design (see Initial Design.pdf) as other team members suggested ideas.
  ○ Wrote the Rate My Friends interaction diagram
  ○ Provided visuals for the video using Movie Maker.
  ○ Wrote the Architecture & Design section for the Final Report.

Amizan Jaleel:
● In design phase:
  ○ Had a hand in creating Data Model and determining table relationships
  ○ Contributed to designing workflow from User account inception to Match
● Implemented Preferences table on the back-end
  ○ Created default preferences for User whenever User is added to DB
  ○ Filtered User’s RateMyFriends results based on Preferences
  ○ Created routes to allow Front-End to send JSON data to update Preferences back-end
  ○ Implemented one-to-one relationship between User and Preference table via Node-ORM2
● Implemented Match table on the back-end
  ○ Created match whenever two users rated showed mutual interest
  ○ Implemented the MatchMeUp! interface that returns the User’s highest match
  ○ Implemented final Match result screen on front-end
● Created the whole Poster for poster day
● Wrote implementation section in report
7. Bibliography

We have learned from and used a great deal of documentation from the following frameworks, such that it would be impractical to track down and record every URL of every page we visited. Hence, we summarize this bibliography to the home pages of each documentation.

1. Google AngularJS: https://angularjs.org/
3. IonicJS CSS Components: http://ionicframework.com/docs/components/
4. Some IonicJS examples were written in CodePen especially the side menu and range slider example: http://codepen.io/calendee/pen/njmbC
5. Apache Cordova: https://cordova.apache.org/
7. Facebook Developer Documentation, including Graph API: https://developers.facebook.com/docs/, https://developers.facebook.com/docs/graph-api
10. Restify: http://mcavage.me/node-restify/
12. Some NodeJS add-on libraries we used made by other developers:
   1. Request Library: https://github.com/mikeal/request
   2. Node-Postgres: https://github.com/brianc/node-postgres
13. CORS on Restify: http://stackoverflow.com/questions/14338683/how-can-i-support-cors-when-using-restify
15. Code sharing:
   1. Git and configuration tutorials:
      http://git-scm.com/
      http://gitimmersion.com/
   2. Bitbucket: https://bitbucket.org/
15. Our official website: http://shadkan.bitbucket.org/
16. Capstone Video: http://www.youtube.com/watch?v=6AwPQbKbL-c
17. Fiddler on the Roof “Matchmaker” used in the Capstone Video: http://www.youtube.com/watch?v=59Hj7bp38f8