MAGIC

Final Report
May 10, 2016

Table of Contents

Our Team:

Full Stack Development:

Erik Ronning

Frontend Development:

Florina Heredea
Alejandro Molina

Backend Development:

Aravind Elangovan
Quinn Herrera

Visual Design:

Ingrid Zippe

Abstract:

MAGIC is a unique dating web application that allows users to customize the algorithm displaying potential matches. Based on characteristics listed in their Facebook profile, such as likes, music or location check-ins, our users are shown their best options first. After users go through all profiles that fit their preferences, we display profiles through our default queue algorithm, based on location. MAGIC allows users to find and communicate with matches by simply accessing our website and connecting with a Facebook profile, without requiring any application download or filling out forms. Additionally, MAGIC targets users with ages between 18 and 36, which distinguishes it from the competitor applications with older demographics.

User Research:

Mentor Feedback:

In the early stages of our development process, we discussed our application features with Angela. She advised us to conduct research on potential target users about how they would like a dating web application to work. She agreed with us that simple profiles and navigation would ensure a smooth user experience, and she supported the idea of matching users based on their preferences. Since MAGIC allows users to interact with strangers, Angela stressed the importance of deciding how to match users and what information to display before and after a match. An initial idea was providing a link to the match's Facebook profile after a match was made, which we decided provided too much information too soon. Sharing contact details should happen only upon a user's decision, through private messages.

The initial technologies we thought of using were CSS, HTML, Bootstrap, Angular, React and Javascript for the front end. Node and Express were the technologies we considered for implementing our back end.

User Survey:

We conducted our research through interviewing users in the initial stage and then collecting feedback on our design. Each of our team members interviewed 4 potential users. In total, we interviewed 24 undergraduate students from Brown University and RISD.

In our user interviews, most individuals mentioned specific features in dating and applications they were familiar with that they found useful. The most popular were Tinder and Match.com. 6 out of 10 potential users mentioned the importance of simplicity in signing up for the app and communicating with their matches. 6 out of 10 potential users felt that the matching process did not give them enough options that shared the same interests. The users supported the idea of choosing what matters most to them when looking for a potential match. One user mentioned that the distance feature was critical so that a user can match with others who are nearby, so we decided to write our default algorithm based on the user's current location.

During our meetings with potential users, we observed them using competitors' applications and websites. This way, we could see what works well with applications like Tinder, Bumble, the League, and Tribe and dating websites like Match.com, eHarmony.com, BlackPeopleMeet.com, and LatinoPeopleMeet.com. Users liked features like Tinder's Facebook login and the match

messaging option, but disliked the fact that there was very little information about why they were shown their potential matches. Users often preferred to make their swipe decision based only on the visual appearance of the potential match, age, and/or location because going through the potential matches' profiles to find out whether they have common interests required too much time and effort.

Based on this research, we decided which features we would include in our application. MAGIC aims at giving the user more control over their experience. The users can select which features on their Facebook profile are prioritized during the matching process. Logging in with Facebook makes it easy for the users to start using the app. Furthermore, the user is able to select the information we have access to when they first log in. MAGIC does not post to Facebook, so the users' activity on the app will not be shared with their friends.

Development Teams:

Full-Stack Development Team:

In charge of handling user authentication via Facebook, managing and creating a cloud hosted MongoDB database as well as pulling user information from the Facebook Graph API.

Generated a universal CSS stylesheet to be used for each page on the website. Performed linking of data requested from the backend to the frontend. Compiled basic HTML pages to dynamically load in information transmitted from the backend.

Backend Development Team:

Responsible for establishing a matching algorithm for users based on preferences of their choice. Handled the matching of users and storing of matching information. Setup realtime websocket messaging between users and stored message information in the database.

Frontend Development Team:

Worked on creating HTML and CSS for individual pages. Integrated website pages HTML with AngularJs formatting.

Graphic Design Team:

Performed case studies and researched optimal website UI design. Created mockups of potential design directions with which the website could move forward.

Frontend Implementation:

Login:

Users login in and signup for the MAGIC dating website by authenticating via Facebook. This allows users to quickly login and starting matching to find some magic (Figure 5).

Personal Profile and Settings:

The user is able to navigate to their personal profile page and and view all of their profile information. This includes lists of the characteristics they chose to share when signing up, such as likes, music, or check-ins. Users are able of picking how they want to be matched. This allows for each user to customize their own matching algorithm (Figure 6).

Potential Match Profile:

The potential match profile displays the potential match's first name, age, hometown and matching score. A list of all the mutual friends and common interests is also displayed (Figure 7).

Match List and Messaging:

The matching page contains a list of profiles that the user has matched with. The user is able of navigating to a private chatroom with their match. All updates for matching and messaging are performed in real time. (Figure 8)

Frameworks:

The project is utilizing Angular 2 as a modular framework to handle our application architecture. Angular 2 allowed us to build a more flexible and seamless app, as its single directional data flow model allows for more explicit data flows, with no circular dependencies between bindings. It is also using Bootstrap's CSS for the CSS framework and components along with ng2-bootstrap for interactive Angular components.

Routing:

The server consolidated routing to one file (routes.js) which handles all GET and POST requests made to the server. Each of these functions implement a middleware function which ensures that a user is logged in and is attempting to access the correct page.

Future Additions:

There will be notifications for when users have a new match and when they receive new messages.

Users will also have the option to remove a profile from their match list.

Users will be able to view the profile pages of their matches.

Backend Implementation

Server:

Node.js using Express.js web app framework for routing. The server is currently being persistently run on heroku (cs132-magic.herokuapp.com). The program flow can be viewed in figure 1.

Build System:

Gulp, a build system for Node, is used to automate the processing (minifying, bundling, transpiling, sourcemapping, etc) of source files into production assets served by the Node server. The build process varies depending on the environment. In production, it would simply run the build process once and exit. In development, it continues to watch the source files and reprocesses any changed files to aid development time.

Authentication:

Passport is used to authenticate with Facebook. Passport gains permissions to user information through a specified scope (user_friends, user_likes, etc). Passport is middleware used to keep track of the user's authentication token throughout their session.

Information Gathering:

Following authentication into the website, information is gathered about the individual users. A vast range of information such as users likes to user location is gathered from the Facebook Graph API. The fbgraph node module is used as a middleware to make requests to the Graph API. The module will pull all of the information about a user each time a user signs in to ensure that the program has updated information regarding the user.

Currently, all of a user's friends that also use the application are gathered using the module. The use of this module will be extended to include all of the information about the user that is deemed necessary in order to be able to create a complete profile for the user.

Database:

A cloud hosted MongoDB is used for the database. Using the cloud hosted MongoDB allowed for easy integration with Heroku. Mongoose ODM is module used by the server for reading/writing the database. Our user table was extended to include all the Facebook characteristics of the user (Figures 2,3). The database also contains a messages table that stores all messages sent between users (Figure 4).

Matching Algorithm:

The matching algorithm assigns a score to potential matches based on the user's preferences. The potential match profiles with the highest score are displayed first.

The matching algorithm iterates through the lists of all preferences selected by the user. The score is increased by 1 for each characteristic the user has in common with the potential match. This ensures that the first potential matches shown to the users are always those who they have the most in common with.

Future Additions:

There will be an option to match by selecting an age range as opposed to be matched with all users in the same location.

A more extensive case study would allow us to find the best way to display the match score for users, and how to normalize the score according to the user's settings. Our ideas for display included a star rating out of 5 stars and a percentage match.

Visual Design

The design of our application has been kept minimal for ease of user testing and early-stage development. Priority has been placed upon coding inherent functionality into the web application, and therefore UI and stylistic choices have been kept minimal for ease of development (see Figure 5-9). In future iterations, the design elements constructed in Photoshop and Illustrator, which we have coded into basic HTML and CSS, will be linked to the back-end.

In the first visual prototype (see Figure 10), the application aimed to combine elements borrowed from social applications that are already successful, such as Facebook and Tinder. Buttons that are easily recognizable to users were borrowed from Tinder, while the initial inclination to create a robust social environment complete with messaging and notifications about new matches comes from our observations of the Facebook user interface.

In the second prototype (see Figure 11), our team decided we wanted the application to appear more game-like, that way a user could easily resize the browser to a smaller dimension and click to find new matches alongside any other web application or open browser window. We eliminated all unnecessary elements.

After we settled on an intuitive visual system for application, the primary focus was on graphic stylization of the interface, which can be seen in the third and fourth prototypes (see Figure 12 and Figure 13). User interviews were conducted, and the primary comments received were that the interface felt intuitive to use, but that the background was too dark to be comfortable for long periods of time and that the logo should appear in a single color. Adjustments were made accordingly.

Technological Challenges:

Framework challenges:

The main issue we encountered was integrating Angular into our front end framework, and connecting the front end with the back end components of the application. We were initially using EJS to inject data into templates and made this change because we wanted an MVC pattern for our application.

We encountered minor difficulties in using the Bootstrap grid system for making our layouts responsive. We eventually decided against the grids.

Bug Documentation and Testing:

A large part of our early testing process was included in our submission for the Testing lab. This includes compatibility, accessibility, security, usability, front end and print testing. We repeated these tests on the final version of our application to ensure proper functionality.

We created Test User profile on Facebook to allow us to test logging in as a new user, and seeing if edge cases such as not having any potential matches were handled correctly.

We found issues such as using tagged places as part of our preference settings, since the objects had no name property. In addition, we decided against using location as a setting since our default algorithm is based on the current location listed on the user's Facebook profile. Our initial matching preferences have friends and likes enabled by default as characteristics to be used in the matching process..

When testing, we found minor issues such as the match list not updating in real time and gender preference set to "Both" instead of the opposite gender.

Security

A fair amount of security testing needs to be done. At the moment very little security testing has been done. Most of the time spent on the project so far has been dedicated to getting the core functionality of the application completed. At the moment the application has secured passwords by making sure that they are hashed and salted. Our application should be secure against JavaScript injection and XSS attacks (this is handled by the Angularjs framework). The backend implements a middleware which validates users when requesting routes for the backend which will protect against Direct Object Reference attacks. The website is vulnerable to both DoS and CSRF attacks. The server could be setup to implement throttling and debouncing which will reduce a user's ability to be able to implement a denial of service attack. Considering that we are using Nodejs we are going to use a module which would protect the application against Cross Site Request Forgery. Before the system was released for public use, the program would undergo several rounds of penetration testing to locate and patch vulnerabilities in the system.

Appendix I

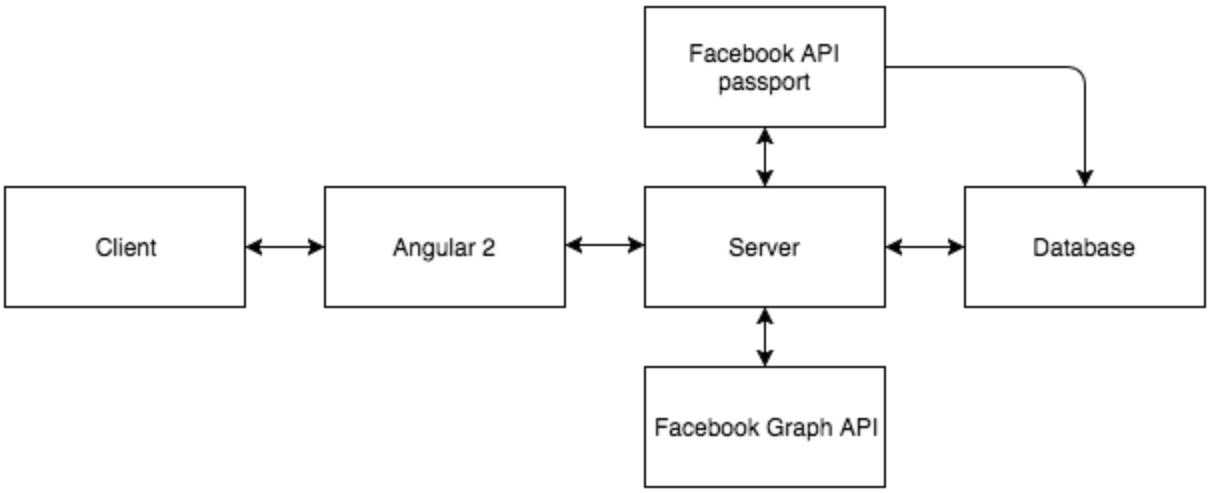

Figure 1

```
User        : {
        id              : String,
        token           : String,
        email           : String,
        name            : String,
        photo           : String
      }
```

Figure 2

```
User                :{
    authenticate    : {
        id          : String,
        token       : String,
        email       : String,
        name        : String,
        photo       : String
```

```
},
first_name      : String,
last_name       : String,
gender          : String,
birthday        : String,
email           : String,
hometown        : {
   id           : String,
   name         : String
},
location        : {
   id           : String,
   name         : String
},
likes           : [{
   name         : String,
   id           : String,
   created_time : String
}],
photos          : [{
   created_time : String,
   name         : String,
   id           : String
}],
friends         : [{
   name         : String,
   id           : String
}],
tagged_places   : [{
   id           : String,
   created_time : String,
   place        : {
      id        : String,
      location  : Object,
      name      : String
   }
}],
events          : [{
   description  : String,
```

```
        name        : String,
        place       : {
            id      : String,
            location : Object,
            name    : String
        },
        start_time  : String,
        id          : String,
        rsvp_status  : String
    }],
    music           : [{
        name        : String,
        id          : String,
        created_time : String
    }],
    books           : [{
        name        : String,
        id          : String,
        created_time : String
    }],
    settings        : Object,
    matches         : Object
}
```

Figure 3

```
Message       : {
    fromId      : String,
    toId        : String,
    message     : String,
    timestamp : String,
}
```

Figure 4

Appendix II

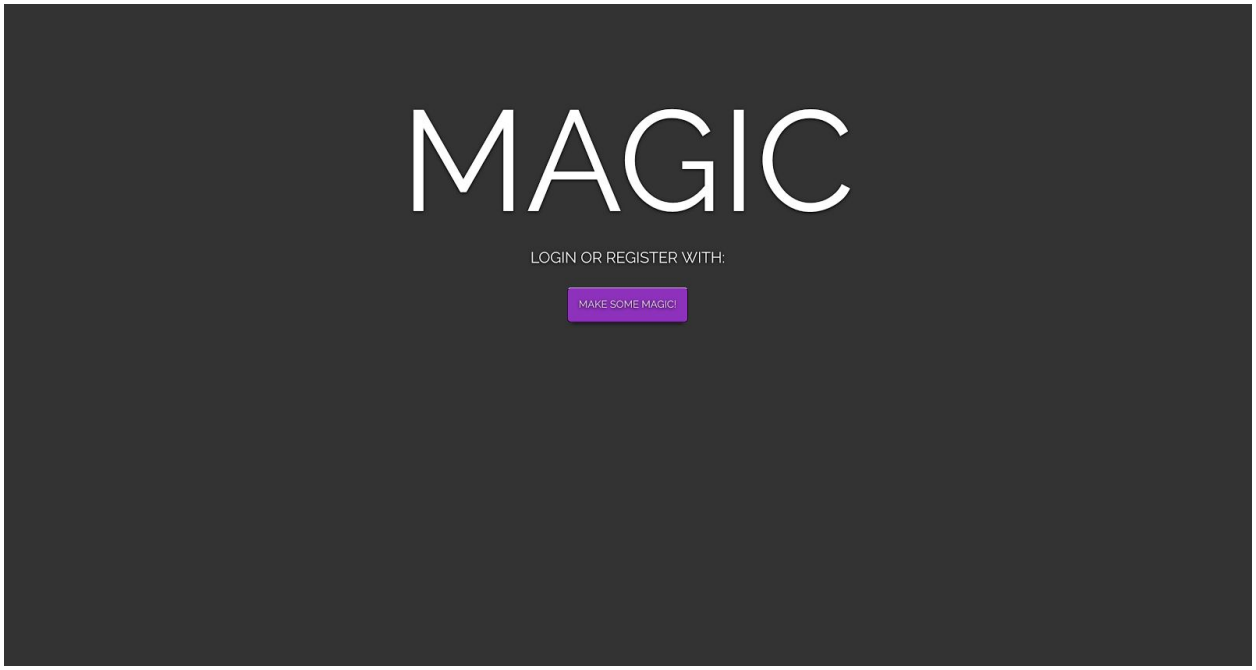

Figure 5

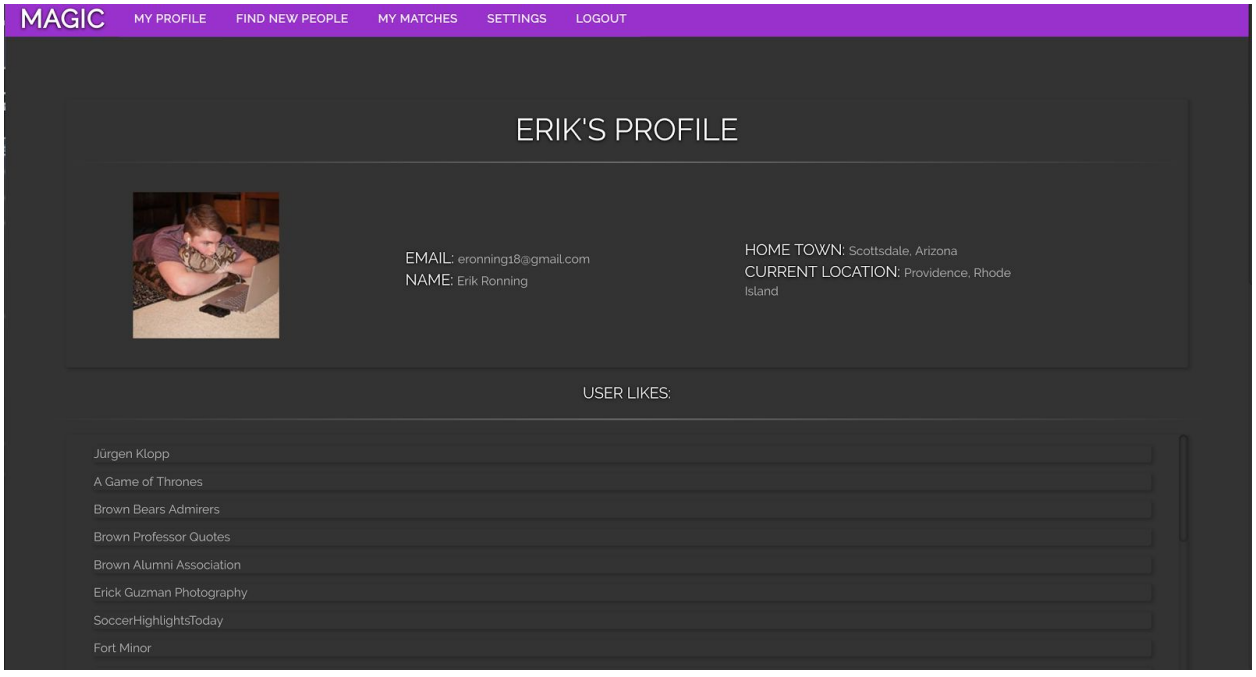

Figure 6

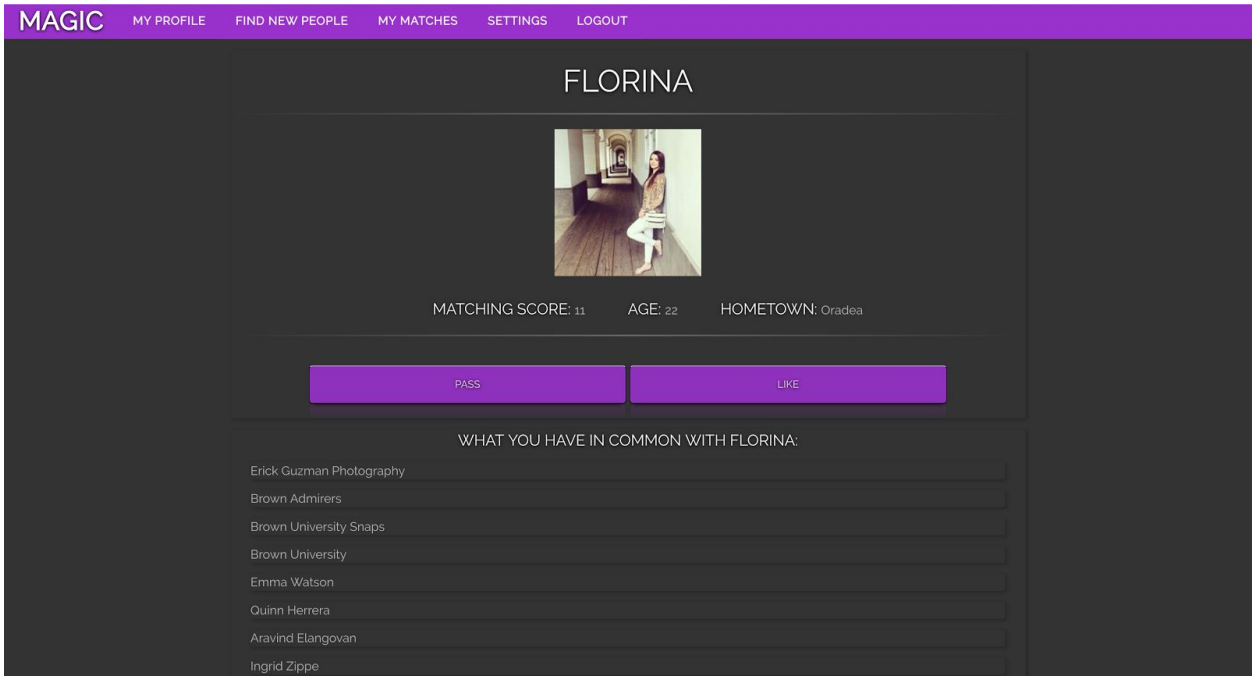

Figure 7

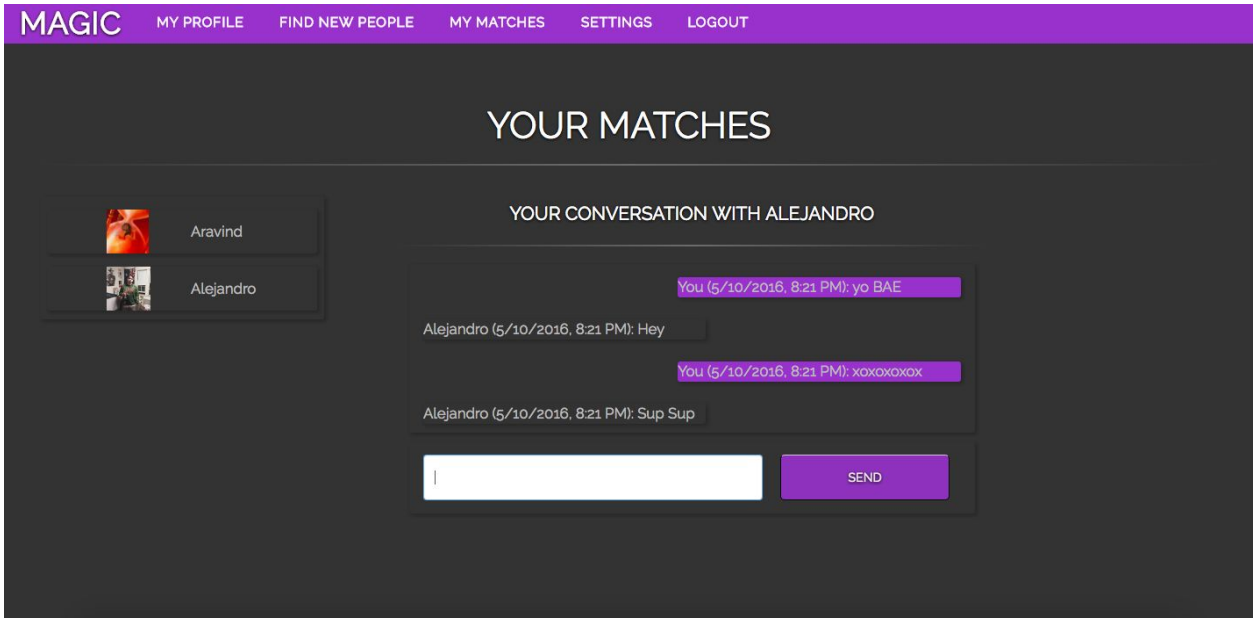

Figure 8

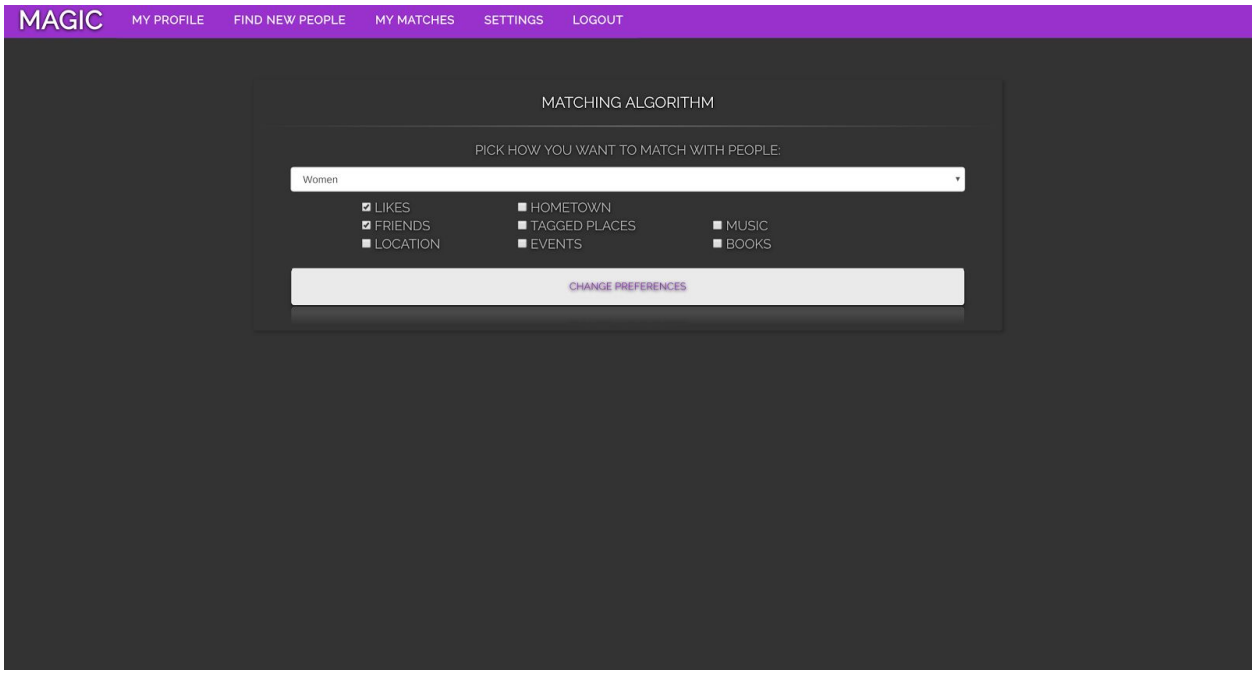

Figure 9

Appendix III

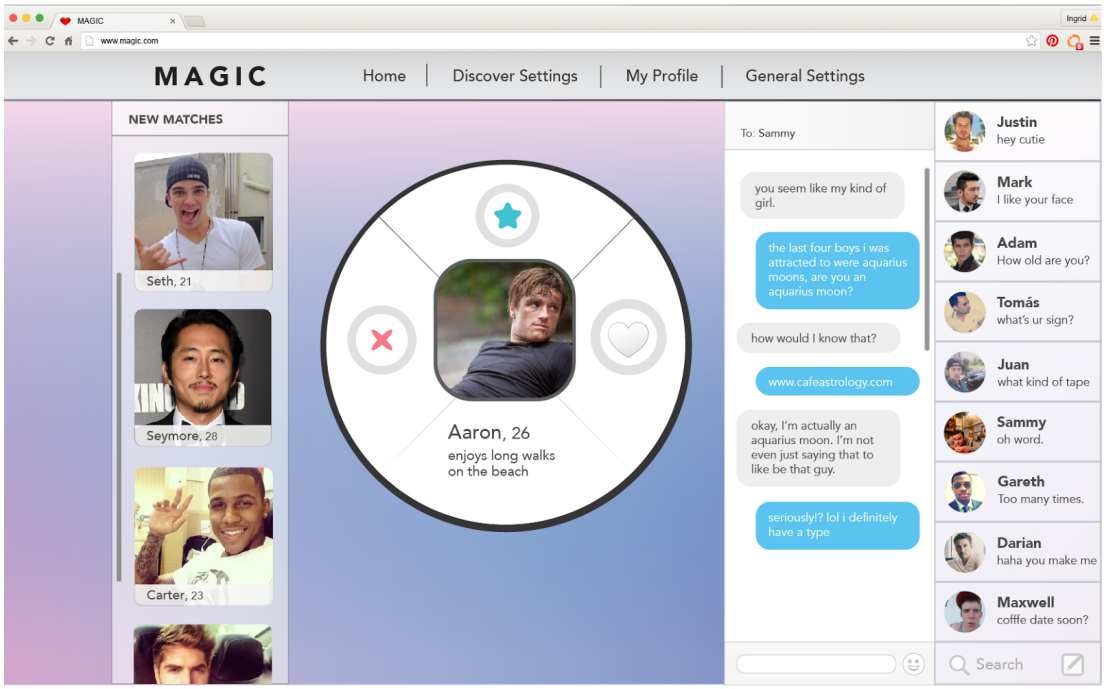

Figure 10

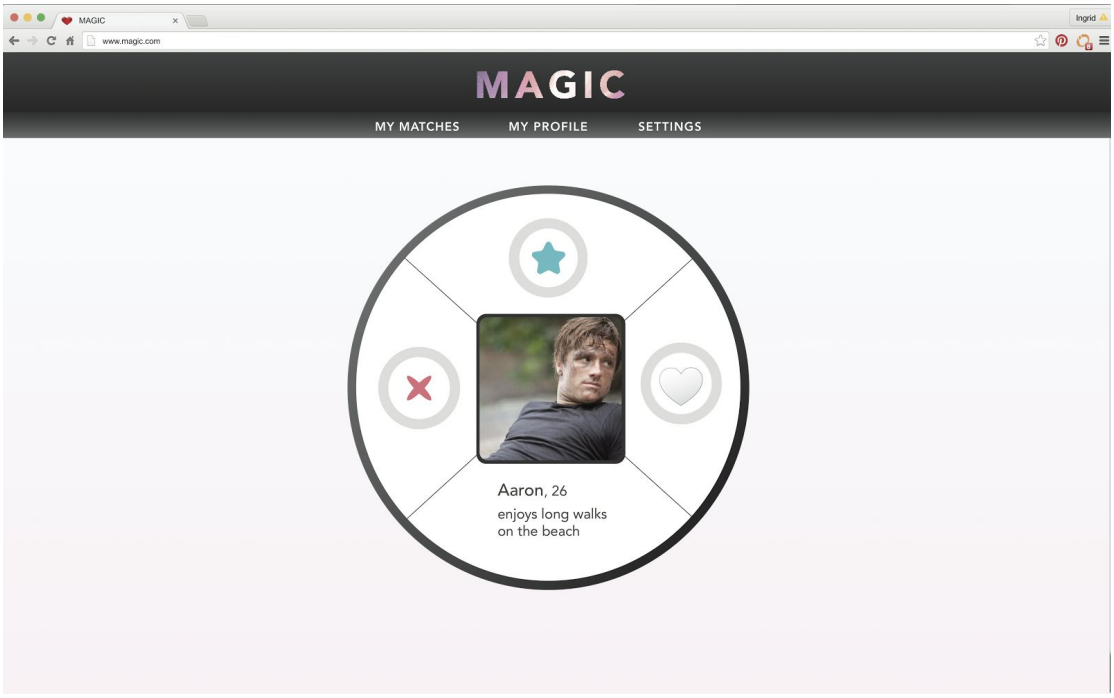

Figure 11

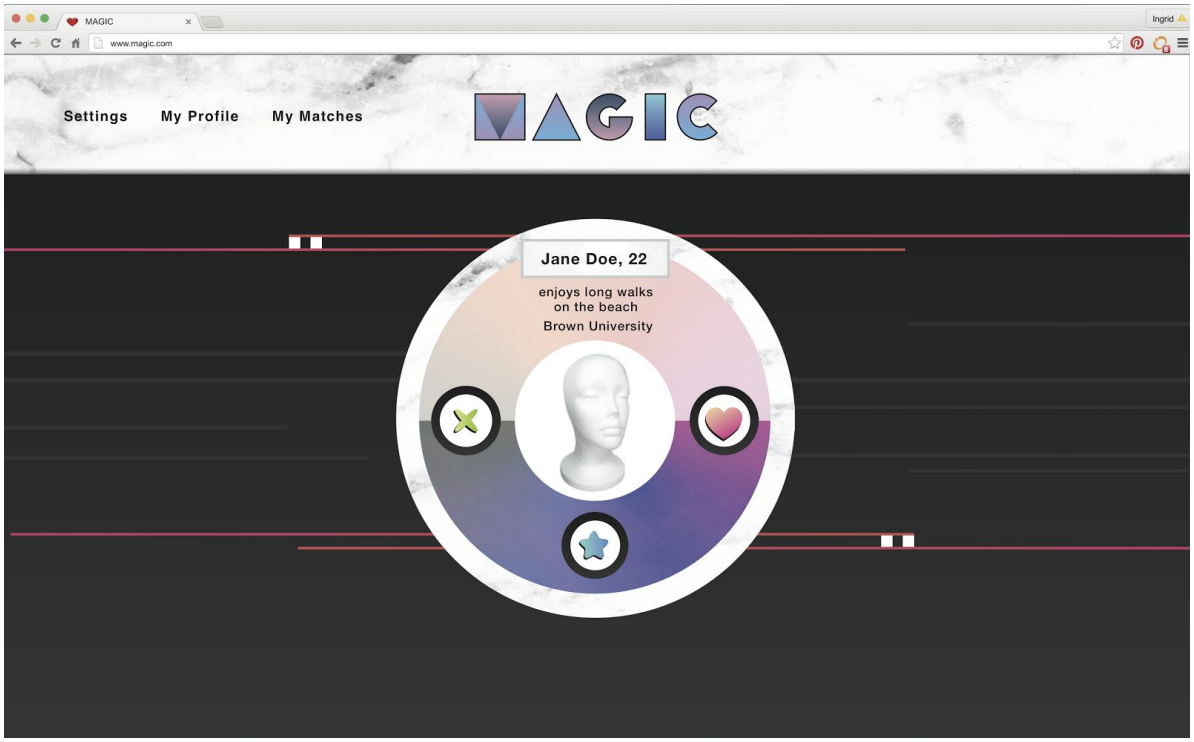

Figure 12

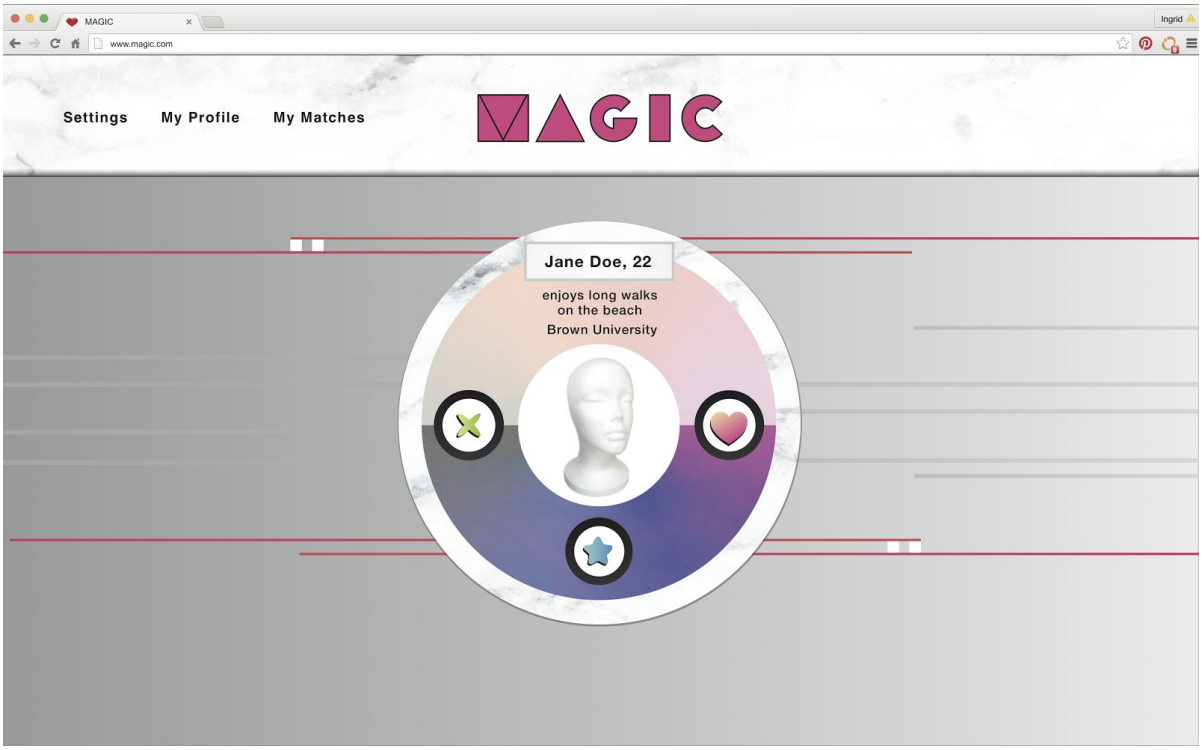
MAGIC
Ingrid
www.magic.com
Settings
My Profile
My Matches
MAGIC
Jane Doe, 22
enjoys long walks
on the beach
Brown University

Figure 13