

# MatchMaker



## Project Summary

MatchMaker is a dating tool that relies on people, not algorithms to suggest matches. A user is presented with two lists of other users and suggests matches between users in the lists. If enough people suggest the same pair, both users in the pair can see the match. If both users accept the match, they are given the opportunity to chat one on one.

To motivate users to match other players, users will earn credits for each successful match. They can use these credits to boost their own profile for other matches or request more users to match.

MatchMaker has two user categories: people who are looking to find a match, and people who are only matching other users. All users can suggest matches, but only those who elect to make themselves available are able to be matched.

## Proposed Value

Many existing dating apps use machine learning to attempt to gauge the interests and preferences of their users. This information is then used to serve each user more people that potentially align with their preference profile.

We feel that this can potentially damage the user's experience for two reasons:

- This can lead to an increasingly homogeneous experience. The user's curated experience within the application can make it harder to find *that someone* who might typically be outside of their preferred dating pool.
- We don't always know what's best for ourselves. MatchMaker relies on the 'Wisdom of the Crowd' to potentially match you with your partner.

Outside of using MatchMaker as a dating app, it's also a game that allows you to play matchmaker for others. A user doesn't have to have an active profile to enjoy our service!

## Technical Project Requirements and Specifications

MatchMaker is a web based application. For each user, the application needs to support a profile page, matchmaking page, recent suggested matches, chat functionality, and personal stats page / leaderboard.

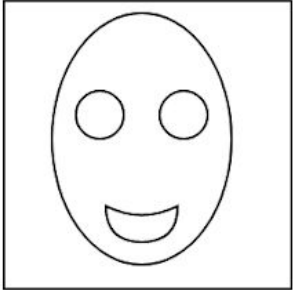
### Profile Page

This page will let the user view and update their description and settings. In addition to a profile picture, each user can write a short blurb about themselves and select their favorite song from Spotify. The user will also be able to toggle whether or not they want to be matched with others.

Profile

Matchmaking

Suggested Matches



Michael Smith

Profile information here.  
Freeform text or answers to questions. Lorem ipsum dolor sit est. Enor al iben epsio sin.

Settings for your account will be on this page

Matchmaking Stats


### Matchmaking Page

The matchmaking page is the center of the application. Here, users will see two lists of users. By rearranging the lists, users can align people they think will make good matches. A button will appear next to each pair to let the user submit the match. After a pair is submitted, two new people will be added to the lists. This process continues as long as the user continues to play.


Profile

Matchmaking


Suggested Matches




Name ♂ ♀ ♀




Name ♂ ♀ ♀




Name1 ♂ ♀ ♀





Name ♂ ♀ ♀



Name ♂ ♀ ♀




Name ♂ ♀ ♀


Name1 and Name2

details about each of them. Both bios are visible here.


Submit this match suggestion




Name ♂ ♀ ♀




Name ♂ ♀ ♀




Name2 ♂ ♀ ♀



Name ♂ ♀ ♀



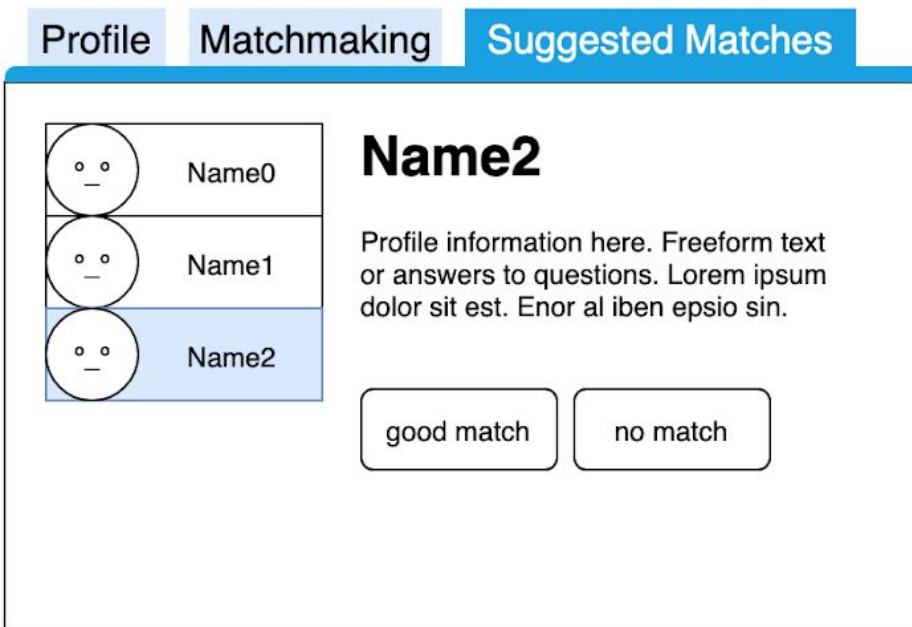
Name ♂ ♀ ♀



Name ♂ ♀ ♀

### Suggested Matches Page

On this page, users will see a list of recent suggested matches. They will be able to view the profiles of matches and accept or reject a match. After both users in a match accept the match, a chat button will appear for both users.



### Chat Page

The chat page is essential for users to get to know each other. Users can send private messages to each other and plan a date. We may outsource this section to a 3rd party messaging service.

### Stats + Leaderboard Page

As the user plays more, they will build up stats on the number of matches they were correct about (the users ended up accepting the match), the total number of matches they have made, and the number of times they have been matched.

### The Match Economy

For each proposed match, a matcher earns themselves one credit. If a proposed match is accepted by both parties, that matcher earns an additional 14 credits.

These credits can be spent in a number of ways:

- A user can use credits to 'boost' their profile for 24 hours, placing them at the top of every matcher's queue. (100 credits)
- A user can use credits to post a 'match bounty', which allows them to increase the reward for matchers who successfully match them.  
( [any amount] <= [user's credit balance]/100 )
- Credits can also be used to purchase more users to be matched (normally replenished every 24 hours). (50 credits)

These credits incentivize the process of matching, integral to the function of our application, and allow users to optimize the application for their needs.

#### User (Backend):

Each user will be granted a unique user-id. With the user-id we will store all user information, whether the user's profile is in circulation, and match-data, in a database.

This includes:

- User Profile (Picture, Favorite Song, Interests, isActive)
- User Credits
- Suggested Matches (and how many times the match has been suggested)
  - Each suggested match also includes a list of Users that have suggested this match (so they can be compensated).
- Accepted Matches

#### **Proposed Solutions**

- Backend
  - Ruby on Rails
- SQL server
  - PostgreSQL or SQLite
- Frontend
  - HTML5
  - CSS
  - Vanilla JavaScript
- Web Hosting
  - Heroku or AWS

#### **Team Background and Related Work**

##### Team Background

Michael has worked on a wide range of project prior to this class. He is most comfortable working on Web or iOS based applications. Some similar projects he has worked on include HikePool, a Python Flask based web app, and Tower Defence II, a vanilla HTML and javascript game. He prefers working on the backend and looks forward to working with Heroku.

Griffin has a demonstrated background in Python development, and Cloud infrastructure. He is most comfortable with backend development, and system architecture. Some similar projects he has worked on include EasyJoinAPI: a containerized REST API written in PHP/Python to automate joining Enterprise Windows machines to Active Directory, and DataFlow: a web-application written in HTML5/CSS/Javascript/Python as a solution to data entry for Progressive Insurance IT recruitment. Griffin also has experience developing firmware in C/C++.

Vincent has experience with web development in Ruby on Rails and Node.js with Express. He has made projects for school that range from a spotify-esque database system webapp to a Java-like language interpreter in Scheme. Vincent attends hackathons occasionally and has made personal projects with Javascript, Python, and Unity (C#). He has worked professionally with Ruby and Javascript.

### Related Work

Ship:

<https://apps.apple.com/us/app/ship-dating-made-fun-again/id1448104758>

Tawkify:

<https://tawkify.com>

The List:

<https://apps.apple.com/us/app/the-list-matchmaking-dating/id1440330373>

Each app we've listed above has implemented some of the concepts we'd like to deploy in our own application - the closest to our vision is 'Ship' (first link). However, our application takes a different approach to keeping users interested and engaged; our application will attempt to maintain the sensitive and personal nature of dating apps, but within a game-like environment.

With features like a Leaderboard/Stats page, a point reward system, and a point spending system that allows users to leverage their game play in order to get more matches, we hope to encourage an ongoing interest with our application that extends beyond that of the typical dating app.

### **Work to be Done**

- Specify design more
  - Database design, specification of required fields
- Implementation
  - Database
    - Create model
    - Generate test data
  - Backend
    - Profile
    - Matching
    - Post match
  - Frontend
    - Profile
    - Matching
    - Post match
- Testing and revision

### **Management Plan**

To ensure everyone on the team is kept up to date, we will use a number of tools to manage this project. To keep track of the documents we need to do and administrative work, we will use Trello. To host our repository and keep track of issues and bugs in our code, we will use Github. We will use Google Docs to write each document.

<b><u>Event</u></b>	<b><u>Date</u></b>
Establish Database design and front end functionality	September 27
Rails project set up with initial database	October 4
User profiles working and establish in-app currency values	October 11
<b>Progress Report 1 Due</b>	<b>October 15</b>
Simple match making operation working	October 18
Suggested Matches Page working	October 25
Progress Report 2 Complete	November 1
<b>Progress Report 2 Due</b>	<b>November 4</b>
Update match making to incorporate currency	November 8
Stats and Leaderboard page working	November 15
Test with users and balance currency	November 22
Update based on user feedback, stretch: Chat functionality	November 29
<b>Final Project Due</b>	<b>December 5</b>