Team 18, Magnet Matching App: Yuanqi Cao, Natalie Evoniuk, Pratim Moulik, Esteban Richey

Team 18

Project: **Magnet**

CS 30700: Design Document Grading Rubric

1 Purpose (0.5 points)

(a) Briefly explain the system you are designing and its purpose.

The application will include two main lines of communication: communication between the user profile and a Central Matching System, and then communication between the Central Matching System and a secure online database. The streamlined system will be as simplified as possible for easy maintenance and will also include multiple external APIs for additional features.

We came up with this idea originally as a way to find people nearby to play specific sports with. This idea then evolved into an application that helps you connect with others nearby based on any interest or hobby. This project will be beneficial for people looking to make new friends over a specific hobby or activity. Our system will use a simple algorithm to connect you with people that have similar or exact same interests as you. Our project will also allow the creation of groups around similar interests, which can greatly enhance the experience of many activities.

Many apps have a similar match-based interface, for example, Tinder and Bumble use location and age-based matching. However, these apps have a limited scope with which they match you with people: the only qualifier is gender and age. Our app has more qualifiers, such as hobbies and interests. Another limitation of those apps is that they're used mainly for casual dating, which not everyone in the population is looking for. Our app addresses those limitations by being focused on making friends over a shared connection, which is a need possessed by much more of the population. With our app, we hope to bring people together.

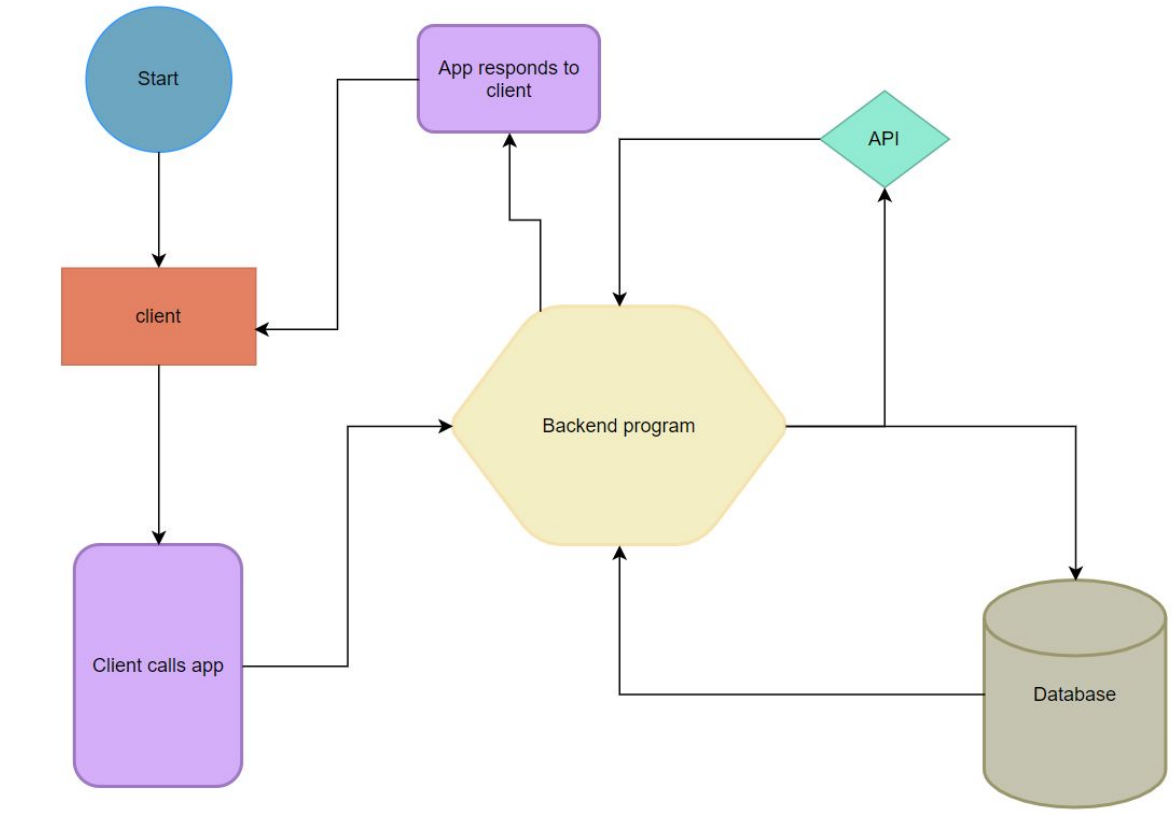
2 Design Outline (3.0 points)

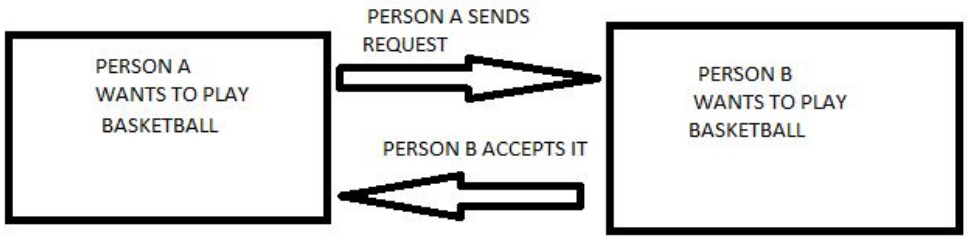
(a) Outline your design decisions (for example client-server model), identify the components of your system, and describe the purpose of each component.

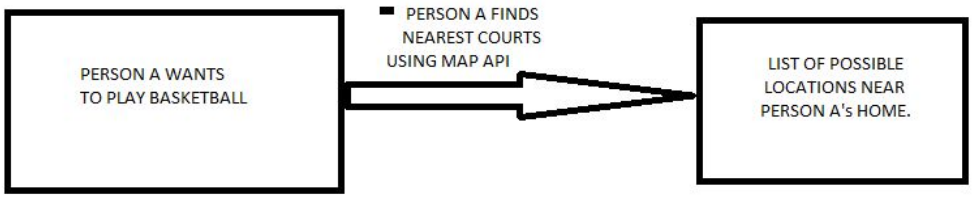
The system will be a basic client server model that will start from the user interface and link up with multiple APIs. It will start from the user interface which will be made from XCode and Swift. This friendly user interface will connect to multiple services, which require Map and Search APIs and other messaging features. The APIs will then connect to a central matching system. This system will process all the users information and send it back to the client while storing it in the backend database.The purpose of this central matching system is to make matching more efficient. Rather than having the matching system implemented externally, keeping it on the device itself will allow our matching algorithm to run only when the user runs the app. We will either implement this through a normal algorithm or AI. Lastly, the administrative panel will be used as official customer support and the backend will compose of a secure database which will store all the information.

(b) Describe the interactions between individual system components.
After the client inputs all the information needed for the matching system, the matching system will communicate with the online database in order to find potential matches. Once complete, a large list of matches for the user to go through will appear in the UI. Once the user has gone through the potential matches(whether that be in the same sessions or after multiple sessions in the app), the matching system will communicate with the database to once again make another list of potential matches for the user. The Central Matching System will also communicate with our messaging API by allowing users the chance to communicate with each other. Additionally, the Central Matching System will communicate with our Administrative System with regards to matches that raise client concerns. The Client Services will mainly interact with the Central Matching System and the Database to provide information about the users that will be integral in matching them.

(c) Include at least one UML diagram that clearly shows high-level structure of your system.







3 Design Issues (2.0 points)

(a) Ensure you spend enough time thinking about the design issues. Only a few design issues will not be sufficient to get full credit

Functional Issues:

- **Consistency of Data between Central Matching system and Secure Database** It may be difficult implementing the central matching system to communicate with the database, especially if multiple users are communicating it at once.
    - Solution 1: could potentially relocate the central matching system to a cloud-based solution.
    - Solution 2: Structure our algorithm and our database in a way that data being updated in the database will not .
- **Testability** The dependency within the individual parts may make them difficult to test. Multiple components of the program could be causing unintended consequences that will be hard to track while they communicate with each other
    - Solution: delegate testing of each component among group members
    - Solution 2: Enable a debug mode that prints out a log that we could analyze in order to track down a specific bug
- **Client-Server Data Syncing** The secure database might have problems synchronizing with clients and services, leading to issues such as duplicate matches
    - Solution 1: focus on strengthening database sync and data verification
    - Solution 2: focus on functionality and simplification of entire streamline system processing
- **Matching Algorithm Heavy CPU Power Requirement** The matching algorithm, because it is being run locally on the device, might require a lot of CPU power if it is not implemented efficiently. Computing power is limited on mobile devices, so this could drain battery if the user goes through a lot of matches
    - Solution: Run the algorithm long enough to find a lot of matches to present, that way the user does not have to run the algorithm multiple times in a session unless they go through many matches.
    - Solution: Move some parts of the matching algorithm using cloud servers in order to relieve some pressure from local CPU
- **UI Usability:**
    - If app layout is too complex and not intuitive then users will have a difficult time using every aspect of the app.
    - Solution 1: keep the UI simple, remove unnecessary pages, maintain a personability within the app so it feels modernized.

- **Troubleshooting**
    - One component could be causing major performance issues
    - Solution 1: effective monitoring to keep track of anomalies
    - Solution 2: Relocate some issues towards cloud servers and databases

Non-Functional Issues:

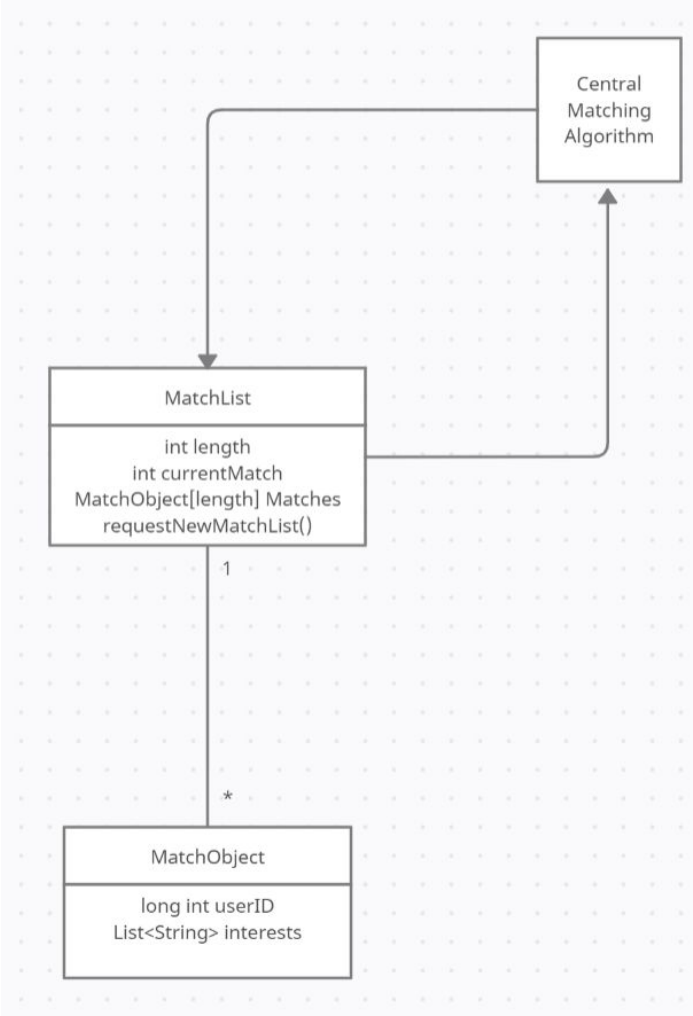
- **Overwhelming Front-End Features:**
    - Design could have additional features presented in the Client Services that may confuse the user and make it difficult to find and use key features of the app.
    - Solution 1): Make the front end design and APIs to an appropriate amount to the system and the user does not get overwhelmed.
    - Solution 2): Make sure the user can freely change most parts of the front end designs in order to make them feel more personalized
- **Clients with unpopular areas**
    - Some clients may feel frustrated because there are little to no people for a large radius that use the app or share their interests.
    - Solution 1): Maybe this could be addressed in the customer support/help page and give recommendations on other areas to visit
    - Solution 2): Make sure the matching algorithm and central matching system matches accurately with expanded radius and also make sure the maps API function correct

- **Clients with unpopular interests:**
    - The app might not connect people with extremely obscure hobbies/interests. This is an important issue to take into consideration because we want our users to actually benefit from our app.
    - Solution 1: Have the option to expand the search area if the user has gone through every potential match. This solution might make it harder to do things in person but it will increase the likelihood of a match occuring
- **Personalization of Recommendations:**
    - If the user feels like the app is not catered to their interests they may lose interests
    - Solution: Interface could present data in the most user-oriented way. For example, having a "recommended for *username*" before presenting the matches.
- **Privacy protection**

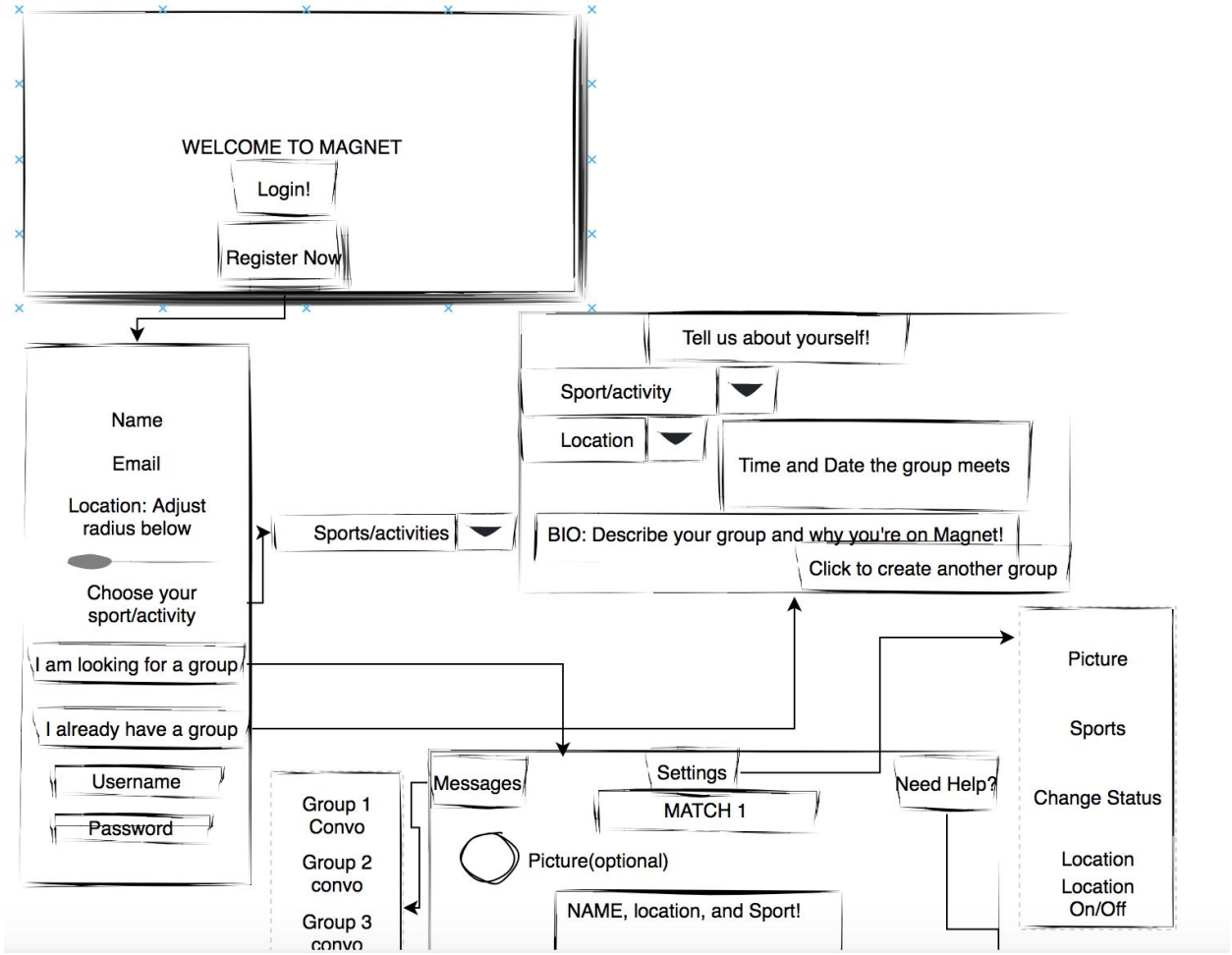
- Matching apps like this will require a lot of user data and information, plenty, very sensitive. Although most of the user data will be secure, it is necessary to make sure the app focuses on user privacy the most.
- Solution 1: develop a secure database that will match the code's algorithm such as mySQL or Mongo and make sure the system only transfers consensual data to the internet.
- Solution 2: Add user agreements and consensual to inform the users where their data went and why the app is using their data for different systems in the app

4 Design Details (7.5 points)

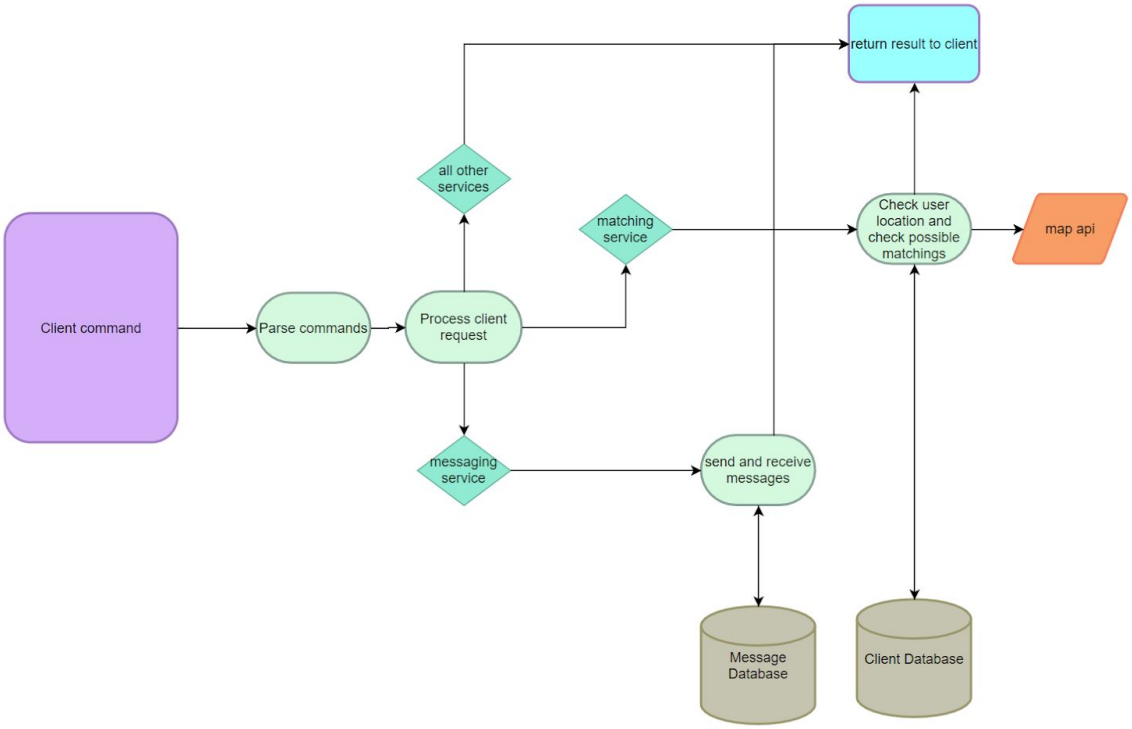
Central Matching System Class Level Design



WireFrame for User Interface



Database activity Diagram:



5 Overall Organization (1.0 points)

(a) Styling, clarity, right information in the right section, etc.

(b) Please make sure to include your project name, team number and the names of all your team members.

Total: 14 points (14% of your project grade)