SwoleMate

Team 7

Design Document

**Contributors**: Jaden Zerbe, Kevin Murcia, Ryan Sand, Samuel Ingram, Steven Dellamore

**1. Purpose**

Finding a workout partner can be difficult with how much skill and ability can vary between people. With SwoleMate you can quickly browse profiles of other exercise enthusiasts in your area to find a workout partner similar to yourself. Working out, whether it be running or going to the gym, can be a solitary activity, but the addition of a companion can add some much needed motivation, camaraderie, and morale to your workout. SwoleMate provides a platform to build your own personal fitness profile and schedule and connect with other active people around you. While the profile is useful for personal metrics such as schedule, type of workout, length of workout, and the finer details of what you accomplished, it is also able to be viewed by others to gauge your workout compatibility.

Functional Requirements

* Profile Management

As a user,

* + I would like to be able to create a SwoleMate account.
  + I would like to create and edit my profile.
  + I would like to login through Facebook or Google.
  + I would like to be able to reset my password if I have forgotten it.
  + I would like to create a bio that describes me and includes workout information and statistics.
  + I would like to upload a photo for my profile or open my camera to take a new photo for my profile.
  + I would like to hide some information on my profile from being seen by other users.
  + I would like to change the distance that I search for other users in.
  + I would like to easily login and logout of my profile.
  + I would like to share my location to view nearby profiles.
  + I would like to filter what profiles are shown to me.
  + I would like to delete my profile if I so desire to.
  + I would like to be able to hide my profile from other users and not show up in their results.
* Usability

As a user,

* + I would like a simple tutorial the first time I use the app.
  + I would like to easily view and scroll through other user’s profiles.
  + I would like to intuitively accept or reject other user’s profiles through simple gestures.
  + I would like to report malicious, spam, or fake accounts.
  + I would like to tap a profile to view more information on that profile.
  + I would like to be able to switch between imperial and metric units.
  + I would like my data to be securely stored.
  + I would like a help/FAQ section on the app.
  + I would like to access the different sections of the app through a side menu accessible with the press of a button.
  + I would like to share another user’s profile with others who might be interested.
* Exercise

As a user,

* + I would like to be able to keep track of my exercise stats.
  + I would like to add future workouts to a calendar.
  + I would like to set goals that are recorded on my profile.
  + I would like my previous workouts to be added to a workout history on my calendar.
  + I would like to display sports or activities I am interested in on my profile.
  + I would like to display or share any milestones I reach in my workouts.
  + I would like to set a favorite gym or workout location.
  + I would like to choose and display my fitness experience level.
* Matches

As a user,

* + I would like to receive notifications when I have a new match or message.
  + I would like to easily view all my matches and direct messages.
  + I would like to direct message other profiles who I have matched with.
  + I would like to unmatch with profiles I have matched with.
  + I would like to share my workout calendar with users I match with.

Non-functional Requirements

* Performance

As a developer,

* + I would like the different screens to change smoothly.
  + I would like the app to launch quickly.
  + I would like the app to not crash.
  + I would like the application to support thousands of users.
* Security

As a developer,

* + I would like to store my users’ passwords securely in the database.
  + I would like to limit users to one account.
* Usability

As a developer,

* + I would like the app to be intuitive.
  + I would like to provide the user with helpful information if they need it.
  + I would like the server to run continually.
* Appearance

As a developer,

* + I would like the app to clearly outline buttons and actions.
  + I would like the app to be visually appealing.

**2. Design Outline**

This project helps others connect to one another on the same platform for the purposes of exercising. Taking into consideration the fact that we will have multiple users and one platform, we will be using a Client-Server Model. We will be using one server that will handle multiple connections with clients simultaneously. The server will also communicate with a database which we will use to store user profiles, user data, and user locations.



Client

The client will offer a user interface for processing user input. The client will be responsible for taking user input and sending it to the server for processing and potentially displaying information in the UI for the user.

Server

The server will provide most of the logical functionality of the application. The server will process any data sent to it by the client. The main way the server will process data is storing and retrieving data from a database and returning a response to be used by the client.

Database

The database will store user data typically in the form of a profile. The server will frequently connect to the database and issue queries for information to send back to the client.

Overview

The client and server will send requests and responses using HTTPS protocol. The client will either send a POST request in which it provides data (usually user profile data or a message) to the server for storage. The client can also request data including profiles to swipe through or messages between users. The server will be hosted on Firebase and decide what to do with the received request. In general, the server will either store or retrieve needed data from the database. The database will be a NoSQL database also hosted on Firebase. If any data is transferred between the server and database, it will be in the form of a JSON.

**3. Design Issues**

Functional Issues

**Design Issue #1** - How should a user sign up with a new account?

* Option 1: Email + username + password
* Option 2: Sign in through Facebook or Google
* Option 3: Both 1 and 2

Choice: Both 1 and 2

Discussion: Giving users more options will help tailor to what they want thus not having our sign up itself as a barrier to entry. Users should be able to easily sign in using their method of choice. It may be easiest to implement only one method from a technical standpoint, but if we have enough time, we will want to implement both options 1 and 2.

**Design Issue #2** - Which way do users need to swipe to accept/ignore profiles?

* Option 1: Right for accept, Left for ignore
* Option 2: Up for accept, Down for ignore

Choice: Right for accept, Left for ignore

Discussion: Most other apps that have a similar function use the convention described in option 1. For the sake of consistency and keeping everything intuitive, we will follow the same convention so that we don’t confuse new users of our app.

**Design Issue #3** - At what point should a user’s location be updated?

* Option 1: On reopening the app
* Option 2: On some predefined interval
* Option 3: On account creation
* Option 4: On a user’s discretion

Choice: On reopening the app

Discussion: Many of other options were either not frequent enough or just way too frequent (i.e. updating every 5 minutes or something). Updating the location upon opening the app provides a happy medium between the frequencies, and it ensures that the location is updated when it matters most.

**Design Issue #4** - How important should profile pictures be in an user’s profile?

* Option 1: Users should be allowed multiple pictures on their profile and these pictures should should be highly prevalent on a user’s profile
* Option 2: Users should only be allowed one picture and these pictures should be less significant on a user’s profile

Choice: Only allow one picture which should play a small role on user profiles

Discussion: While the applications we’re modeling ours off of encourage the use of multiple pictures and make them the main focus of a user’s profiles, we feel as this is unnecessary and takes away from our goal, to match users based on their workout needs and requirements rather than on appearance alone.

Non-Functional Issues

**Design Issue #1** - Where should messages between users be stored?

* Option 1: In the database
* Option 2: In the client

Choice: Compromise - Store messages in database, cache messages in client

Discussion: We talked about the most efficient manner to store messages and decided that storing them client-side would take up too much storage space on a user’s device as well as possibly cause concurrency issues. Storing them server-side, however, could mean slowing down the viewing of previous messages. We therefore settled on a compromise to do both. Save all messages on the server and cache recent messages in the client.

**Design Issue #2** - What should be used for the server side implementation?

* Option 1: Create our own server implementation
* Option 2: Node.js
* Option 3: Apache Tomcat

Choice: Node.js

Discussion: We decided that creating our own server implementation would not be as efficient nor as secure as Node.js. Also, creating our own server implementation would take a lot of time away from the actual development of app functionality. Also, Node.js provides many easy to use methods to set up a server quickly and efficiently as compared to other options.

**Design Issue #3** - What platform/framework should be used for the client side?

* Option 1: React Native
* Option 2: Android Studio using Java and API’s
* Option 3: Xamarin or other Framework

Choice: React Native

Discussion: Most of the members of our team had little to no experience developing a mobile application. One member had experience with React Native and recommended it. The other two options there was little to no experience within our group. After a brief description and a little research on React Native, we decided it would not be too difficult to pick up and seemed like a relatively easy framework.

**Design Issue #4** - What should be used for the database?

* Option 1: MySQL
* Option 2: NoSQL

Choice: NoSQL

Discussion: While MySQL is a proven and optimized way of storing data, NoSQL provides some nice abstractions on top of the usual procedures involved in storing data in an SQL server. These abstractions can help us as developers implement a solid database without having to become experts in writing SQL queries. Furthermore, in line with the decision made in Issue #5, NoSQL integrates nicely with the Firebase platform that we intend to utilize for our app.

**Design Issue #5** - What platform should our server and database live on?

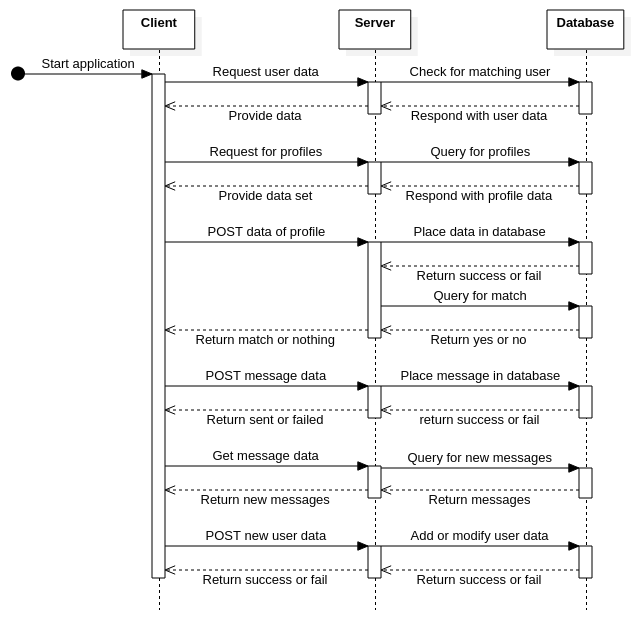
* Option 1: AWS
* Option 2: Heroku
* Option 3: Firebase
* Option 4: Azure

Choice: Firebase

Discussion: Firebase provides a nice abstraction on top of many of the needed functions with our app. While other platforms may be better for pure server hosting, Firebase provides many different features and tools to help make the lives of us developers much nicer.

**4. Design Details**

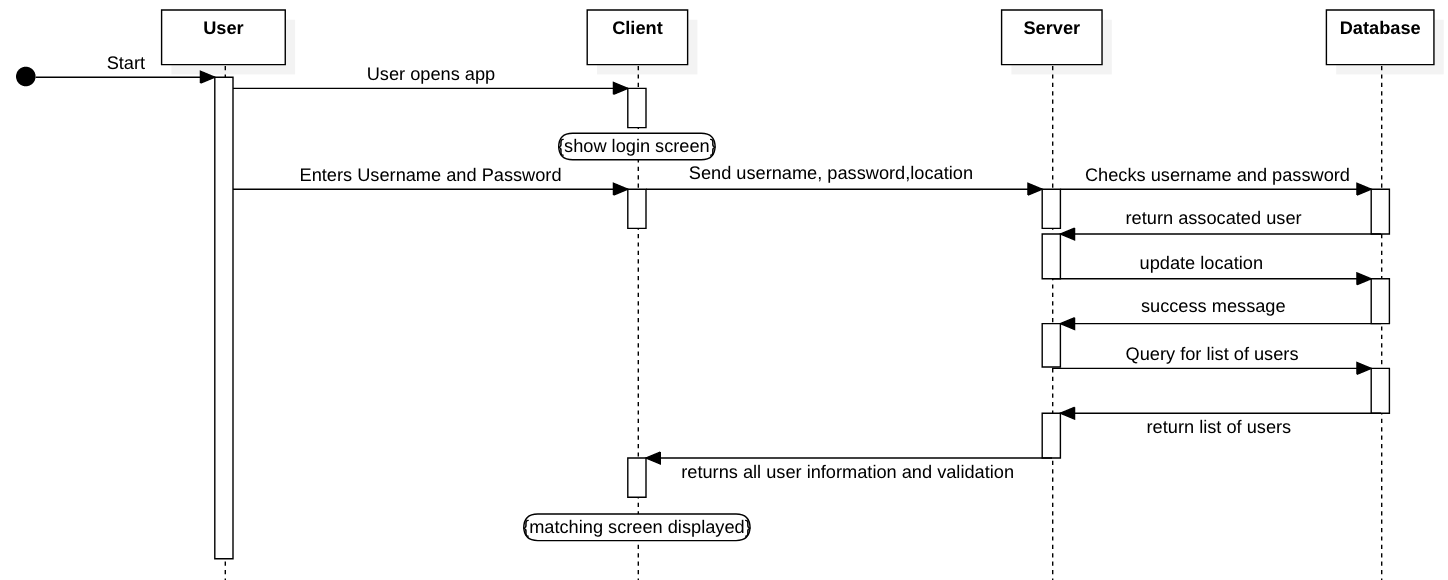
General Client/Server/Database Sequence



**Broad Overview of Sequence of Events**

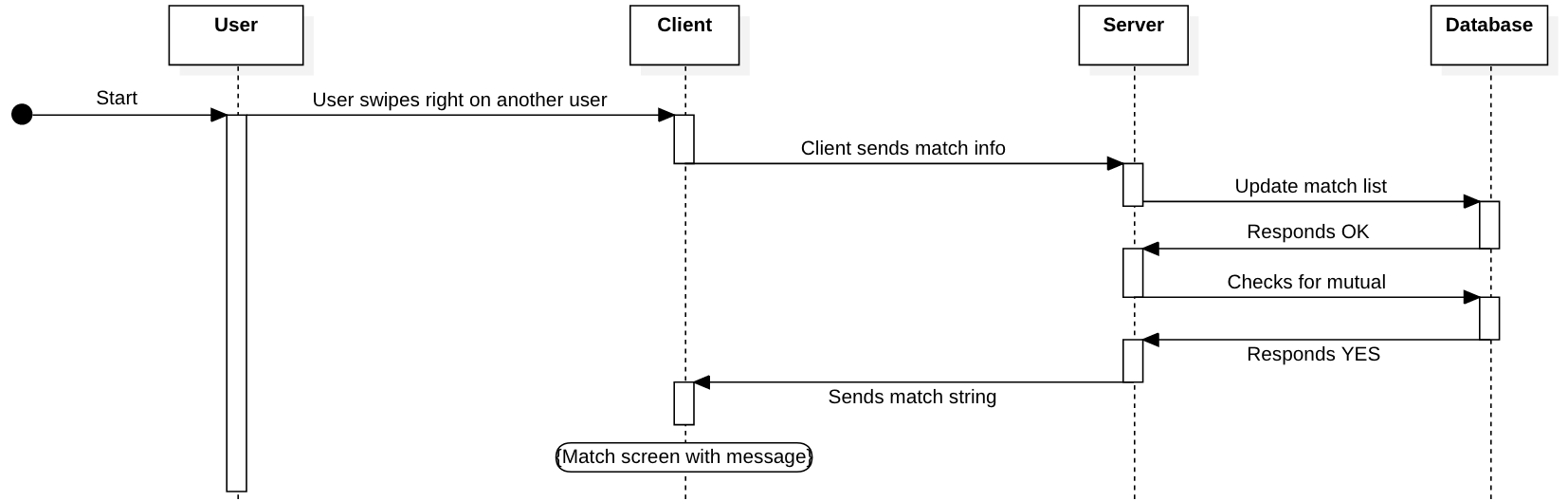
The diagram above represents a typical sequence of events when a user uses our application. The client will send requests to the server for data or to post data. The server will then send these requests or posts to the database. Changes to the database will return with a success/fail message or with data which the server will modify or pass on the data to the client. All requests that can be completed by the client are handled by the client.

Login Sequence Diagram



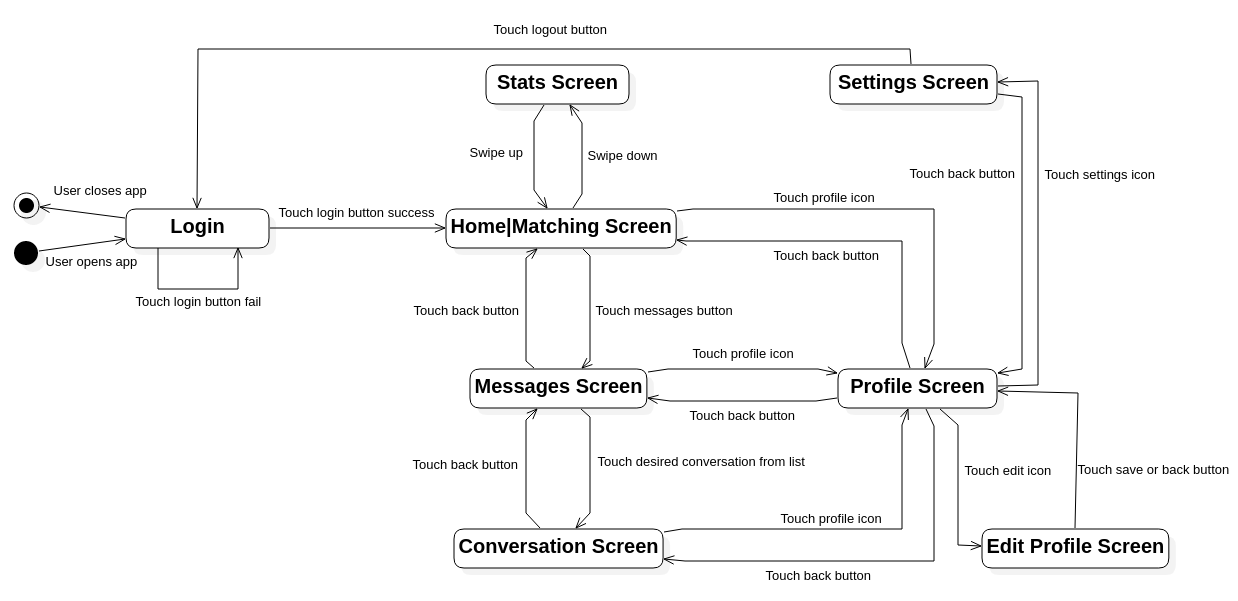
This sequence diagram describes a case in which a user performs a successful login. After properly validating the login info given, the user's location is updated, and the query of users to use for the matching screen is returned.

Match Sequence Diagram



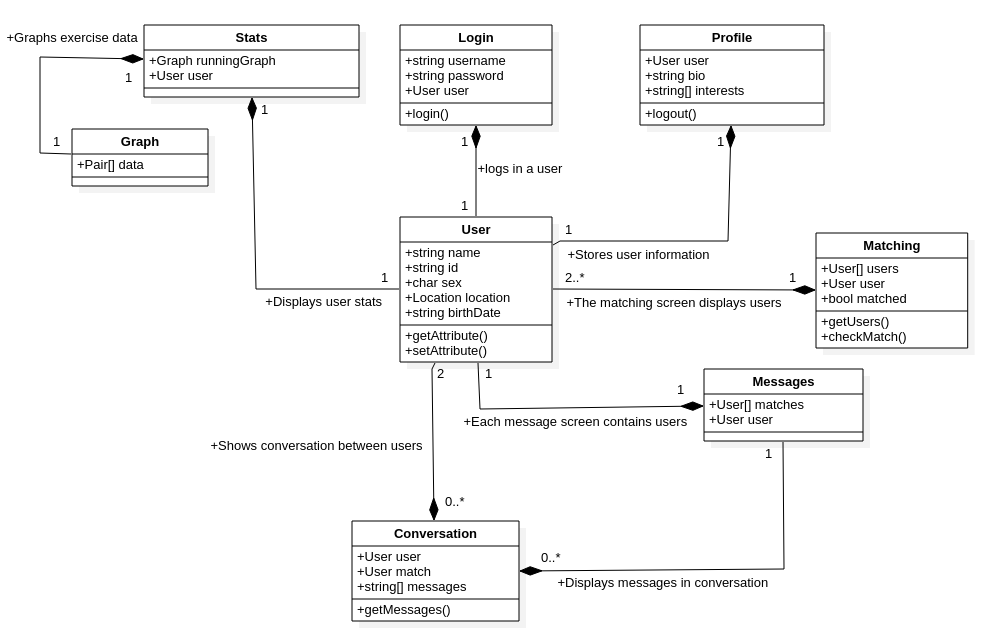
This sequence diagram describes a case in which a user wishes to match with another user (assuming the other user has also chosen to match with them). The user will swipe on their desired match which will send a message to the server to validate the existence of a mutual match. Once all updates and queries to the database have been performed, the server will respond with a message for the client to display a match message to the user.

State Diagram



This state diagram describes the possible connection of different parts of our application. The user will navigate by performing some action (i.e. Swiping or button press). The user will sign in before entering the app and sign out before exiting. Practically, the user would be able to logout anywhere on the app but for the sake of cleanliness it is only shown in the settings screen.

Class Diagram



**Descriptions of Classes**

* **User**
  + User object will be created once a user logs in
  + Users “name” will be filled out with the users full name
  + Users ID will be a unique ID for that user. The ID will be used to query the database on the user.
  + Sex will be filled out with the specified sex of the user.
  + Location will hold the physical location of the user.
  + The users birthdate will be stored in the birthDate string.
  + The getAttribute() and setAttribute() will be called if the user is adding or removing an Attribute from their profile page.
* **Login**
  + Login screen that is the initial screen shown once user opens app
  + User will input username and password which will be sent to server for authentication through the login() function
  + If authentication is successful, then the server will respond with the data of the user which will be stored in the User variable
* **Matching**
  + The “home” screen of the application that is displayed after login
  + Displays the profile of another user from an array of all users in an area to be judged and swiped on
  + If the user swipes right on a profile and that other user also swiped right on the first’s profile, then their matched status will be set to true from the default false
  + The class gets the user array from the database and checks the database for a matched status using getUsers() and getMatch() respectivley
* **Messages**
  + The messages screen will contain a list of matches that the user can choose to message
  + A selection on one of the matches will open up a conversation screen between the two users
* **Conversation**
  + The conversation screen will display all the messages between one user and another
  + Each conversation is created from a mutual match
* **Profile**
  + The profile screen will display the information on a given user
  + In many cases, this will show the user themselves, but it will be abstracted to support displaying of other users in the event that one wants to pull up more details on a specific user
* **Stats**
  + The stats page will have user stats to be displayed
  + The stats page will also offer a graph object for a more visual display of different statistics
* **Graph**
  + The graph object will be an object used in the Stats screen
  + The graph object will be in charge of storing data points in the form of Pairs array
  + The graph will be able to display said points and be overlayed onto the stats page

UI Mockups 