

**Computer Science and Engineering**

**DotaMatch**

**Requirements/Analysis Specification**

**Version 1.0**

Document Number: RAS-001

Project Team Number B11

Project Team Members:

Corey Chong cc3899

Albert Su as7353

Evans Yeung eby210

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function (Author, Reviewer, Approval)** | **Date** | **Signature** |
| Evans Yeung | Author | 10/6/2016 | Evans Yeung |
| Corey Chong | Author | 10/6/2016 | Corey Chong |
| Albert Su | Author | 10/6/2016 | Albert Su |
| Evans Yeung | Reviewer | 10/6/2016 | Evans Yeung |
| Corey Chong | Reviewer | 10/6/2016 | Corey Chong |
| Albert Su | Reviewer | 10/6/2016 | Albert Su |
|  |  |  |  |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| October 6, 2016 | Version 1.0 | Initial Release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[1. INTRODUCTION 7](#_Toc463601520)

[1.1 Purpose 7](#_Toc463601521)

[2. SCOPE 7](#_Toc463601522)

[2.1 Identification 7](#_Toc463601523)

[2.2 Bounds 7](#_Toc463601524)

[2.3 Objects 7](#_Toc463601525)

[2.4 System Overview 8](#_Toc463601526)

[2.5 Document Overview 8](#_Toc463601527)

[3. REFERENCE DOCUMENTS 8](#_Toc463601528)

[4. BUSINESS REQUIREMENTS 9](#_Toc463601529)

[4.1 Technology 9](#_Toc463601530)

[4.2 Economics 9](#_Toc463601531)

[4.3 Regulatory and Legal 9](#_Toc463601532)

[4.4 Market Considerations 9](#_Toc463601533)

[4.5 Risks and Alternatives 10](#_Toc463601534)

[4.6 Human Resources and Training 10](#_Toc463601535)

[5. CONTEXT DIAGRAM 11](#_Toc463601536)

[5.1 High Level (level – 0) 11](#_Toc463601537)

[6. FUNCTIONAL REQUIREMENTS ANALYSIS SPECIFICATION 11](#_Toc463601538)

[6.1 Functional Descriptive Detailed Requirements 11](#_Toc463601539)

[6.1.1 Account Registration 11](#_Toc463601540)

[6.1.2 Matchmaking Service 12](#_Toc463601541)

[6.1.3 Rating System 13](#_Toc463601542)

[6.1.4 Add Partner 14](#_Toc463601543)

[6.1.5 Public Posting 15](#_Toc463601544)

[6.2 System Capability Requirements 16](#_Toc463601545)

[6.2.1 Capabilities 16](#_Toc463601546)

[6.3 User Interface Requirements 16](#_Toc463601547)

[6.4 Component Architecture 17](#_Toc463601548)

[6.4.1 Component Descriptions 17](#_Toc463601549)

[6.4.2 Component Architecture Diagram 18](#_Toc463601550)

[6.5 Class Diagrams 19](#_Toc463601551)

[6.6 Class Relationship/Interaction Diagrams 20](#_Toc463601552)

[6.7 Event Section 20](#_Toc463601553)

[6.7.1 Event Dictionary 20](#_Toc463601554)

[6.7.2 Event Diagrams 20](#_Toc463601555)

[6.8 Activity/State (Scenario) Section 22](#_Toc463601556)

[6.8.1 Activity (Scenario) Diagrams 22](#_Toc463601557)

[6.8.2 Activity (Scenario) Specification 23](#_Toc463601558)

[6.9 Sequence Diagrams 25](#_Toc463601559)

[6.10 Collaboration Diagrams 25](#_Toc463601560)

[6.11 Dictionaries 25](#_Toc463601561)

[7. NON-FUNCTIONAL/OPERATION REQUIREMENTS 26](#_Toc463601562)

[7.1 System External Interface Requirements 26](#_Toc463601563)

[7.2 Safety Requirements 26](#_Toc463601564)

[7.3 Security and Privacy Requirements 26](#_Toc463601565)

[7.4 System Environment Requirements 27](#_Toc463601566)

[7.5 Computer Resource Requirements 27](#_Toc463601567)

[7.5.1 Computer Hardware Requirements 27](#_Toc463601568)

[7.5.2 Computer Hardware Resource Requirements 27](#_Toc463601569)

[7.5.3 Computer Software Requirements 27](#_Toc463601570)

[7.5.4 Computer Communications Requirements 27](#_Toc463601571)

[7.6 System Quality Factors 27](#_Toc463601572)

[7.7 Design and Construction Constraints 28](#_Toc463601573)

[7.8 Personnel-Related Requirements 28](#_Toc463601574)

[7.9 Training-Related Requirements 28](#_Toc463601575)

[7.10 Logistics-Related Requirements 28](#_Toc463601576)

[7.11 Packaging Requirements 28](#_Toc463601577)

[7.12 Precedence and Critically Requirements 29](#_Toc463601578)

[7.13 Other Requirements 29](#_Toc463601579)

[8. SYSTEM TEST PLAN REQUIREMENTS 29](#_Toc463601580)

[9. QUALIFICATION PROVISIONS 29](#_Toc463601581)

[9.1 Review Process 29](#_Toc463601582)

[9.1.1 Self-Check 29](#_Toc463601583)

[9.1.2 Walkthrough 30](#_Toc463601584)

[9.1.3 Inspection 30](#_Toc463601585)

[10. REQUIREMENTS TRACEABILITY 30](#_Toc463601586)

[11. RATIONALE 31](#_Toc463601587)

[12. NOTES 31](#_Toc463601588)

[13. APPENDICES 31](#_Toc463601589)

[13.1 Dictionaries 31](#_Toc463601590)

[13.2 UML Diagrams 31](#_Toc463601591)

[13.3 Schedule Tracking 31](#_Toc463601592)

[13.4 Defect Tracking 33](#_Toc463601593)

[13.5 Gantt Chart 35](#_Toc463601594)

# 1. INTRODUCTION

## 1.1 Purpose

DotaMatch is a matchmaking application that will make the process of finding fellow players a more fun and enjoyable experience by introducing other factors besides skill into the equation. Users will be able to set their own preferences for the type of players they would like to be matched with.

The purpose of this document is to design the features of the system. This includes how the system and its features work. This document is intended for the developers so they will understand how to design the application.

# 2. SCOPE

## 2.1 Identification

Requirements/Analysis Specification, RAS-001, Version 1.0, October 6, 2016

## 2.2 Bounds

* One basic type of user.
* Users can create a profile, send private message, and start a matchmaking queue.
* Log-in and password is used for the identification of user.

## 2.3 Objects

* Priority will be given to developing an interface users can use to coordinate teams and review various statistics.
* Iterative and Incremental lifecycle.
* Software Requirements (SRS) Version 1.0 3/7/2016
* Software Requirement (SRS) Version 2.0 3/21/2016
* Software Project Management Plan (SPMP) Version 1.0 4/6/2016
* Software Analysis Specification (SAS) Version 1.0 4/18/2016
* Software Project Management Plan (SPMP) Version 2.0 9/27/2016
* Requirements/Analysis Specification (RAS) Version 1.0 10/6/2016
* Project Completion and Finished Product Fall 2016

## 2.4 System Overview

DotaMatch is a matchmaking application that will make the process of finding fellow players a more fun and enjoyable experience by introducing other factors besides skill into the equation. Users will be able to set their own preferences for the type of players they would like to be matched with. Our algorithm takes into account these preferences and other factors to decisively match players in the database. Once matchmaking is complete, the user will be presented with a matched player’s profile which includes: in-game statistics, preferred role, favorite characters, etc. A player rating system is implemented so that the user will be able to rate the players they have played with. Overall this application is designed with the user’s freedom of choice in mind in order to create a positive experience for all of our users.

## 2.5 Document Overview

* Section 1 contains the purpose of the project and this document.
* Section 2 contains the scope of the product’s lifecycle including the document milestones.
* Section 3 contains the reference documents.
* Section 4 contains business requirements, including market considerations, risks, and resources.
* ...

# 3. REFERENCE DOCUMENTS

All documents are electronically submitted to NYU Classes and can be referenced through NYU Classes.

* Project Proposal was electronically submitted to NYU Classes on February 17, 2016.
* Software Requirements Specifications (SRS) Version 1.0 was electronically submitted to NYU Classes on March 7, 2016.
* Software Requirements Specifications (SRS) Version 2.0 was electronically submitted to NYU Classes on March 21, 2016.
* Software Project Management Plan (SPMP) Version 1.0 was electronically submitted to NYU Classes on April 6, 2016.
* Software Analysis Specification (SAS) Version 1.0 was electronically submitted to NYU Classes on April 18, 2016.
* Software Project Management Plan (SPMP) Version 2.0 was electronically submitted to NYU Classes on September 27, 2016.
* Requirements/Analysis Specification (RAS) Version 1.0 will be submitted to NYU Classes on October 6, 2016. Once submitted it can be referenced through NYU Classes. It can currently be referenced through Google Doc.

# 4. BUSINESS REQUIREMENTS

## 4.1 Technology

The purpose of DotaMatch is the allow Dota players to be matched with players of their choosing instead of arbitrary players that are matched through Dota’s in-game matchmaking. Allowing the user to choose freely on who they want to play with will make the game more enjoyable and overall a positive experience.

This technology will impact the Dota community by lower the negative experiences Dota players have by being matched with random players. When you get matched with a random player, you never know what kind of player you’ll get matched with.

## 4.2 Economics

If DotaMatch receives positive reviews, then more Dota players will enjoy the game more. This will ultimately increase the Dota user-base. In return, this will increase DotaMatch’s user base. DotaMatch will initially be released free of charge to all Android users. However, transitioning towards a “pay for extra or upgraded features” type of application is not out of scope.

## 4.3 Regulatory and Legal

Not required.

## 4.4 Market Considerations

* Competition from other applications that perform similar matchmaking functionalities as DotaMatch.
* Possible portability for iOS in future iterations. Need to consider iOS users.
* Cater towards Dota players. Need to consider the needs of the Dota user base.

## 4.5 Risks and Alternatives

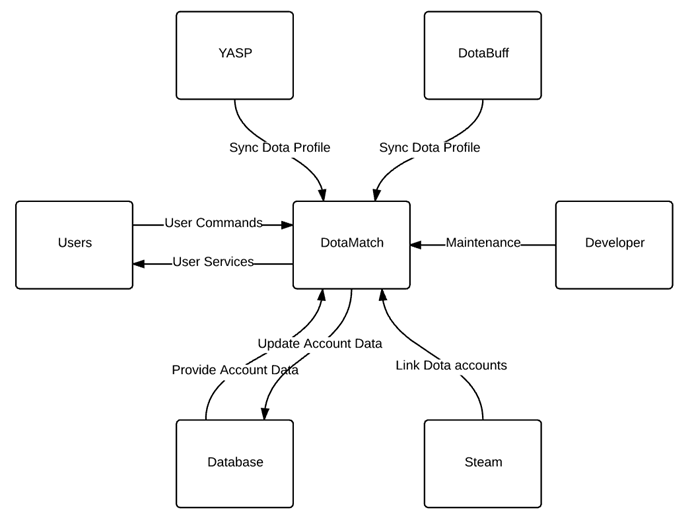
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Probability | Detection | Person Responsible For Addressing Risk | Current Status |
| Lateness | 25% | Scheduling Chart  Gantt Chart | Members of Team B11 | On time |
| Over-budget | 0% | Project does not require a budget | Members of Team B11 | Project does not require a budget |
| Residual Fault | 50% | Self-check  Walkthrough  Inspection  Testing by SQA | Members of Team B11 | Project currently follow requirements set in SRS |
| Member May Leave | 25% | Will be informed by leaving member | Members of Team B11  Professor Strauss | All members are still on the team |

## 4.6 Human Resources and Training

* Project requires a team of at least 3 members.
* Team members should be trained in either C++/C or Java.
* Team members should be trained in PostgreSQL.
* Team members understand risk and risk mitigation.
* The staff will be trained in-house by other teammates, in addition to any additional training that may be provided online. Staff will also learn required skills individually during their own time.

# 5. CONTEXT DIAGRAM

## 5.1 High Level (level – 0)



# 6. FUNCTIONAL REQUIREMENTS ANALYSIS SPECIFICATION

## 6.1 Functional Descriptive Detailed Requirements

### 6.1.1 Account Registration

**Function Specification:**

**1.** Users can create a personal account and link their Dota account if they wish to. Once linked accounts will be updated with in-game statistics from the profile.

**Function Definition:**

**1.1** Users register by clicking on the register button.

**1.2** This will bring the user to the register page where they will fill out a form.

**1.3** They must input their email, first and last name, and password.

**1.3.1** Their email will act as their username.

**1.3.2** Passwords must be between 8 and 16 characters including both letters and numbers.

**1.4** Upon submitting the form a validation email will be sent to the email specified in the form.

**1.5** User go to the email they’ve provided as click on the validation link to activate their account.

**1.5.1** Accounts are not usable until validated.

**1.6** Once activated, the user has to link their Dota accounts by providing their in-game profile.

**1.7** After linking their Dota profile, their DotaMatch account will be updated with their Dota profile statistics.

**1.8** The displayed name on each user’s DotaMatch account will be the same as the name on their Dota profile.

**1.9** Users are given the option to set preferences for potential players they would like to be matched with.

**1.10** Users will be provided with limited ability to adjust visibility of content in their profile.

### 6.1.2 Matchmaking Service

**Function Specification:**

**2.** Users can join a matchmaking queue to look for possible party members. The maximum number of members in a party is 5. Queue creators can start the matchmaking queue with up to a total of 4 users in the party queue (including the creator). The desired number of members in a party can be adjusted by queue creator. The remaining positions in the party will be filled through our matchmaking service until desired number of members is meet.

**Function Definition:**

**2.1** A user can initiate a matchmaking queue by clicking the start queue button.

**2.2** The queue creator has the option to invite his/her friends or partners to fill spots on the party before the queue begins.

**2.3** Queue creator can set additional characteristics to look for in potential players.

**2.4** Once the queue creator is satisfied with the settings of the queue, the creator can start matchmaking by pressing start.

**2.5** Once matchmaking has begun, users within this queue cannot leave unless one chooses to exit the queue.

**2.6** The matchmaking algorithm will take the characteristics inputted by the creator as additional parameters.

**2.7** Matched users’ statistics will be displayed on the queue creator’s app.

**2.8** The queue creator can then swipe right if they wish to play with the matched user or swipe left to move onto the next match.

**2.9** The matched user will be notified that another user wishes to play with them.

**2.10** Users who reach an agreement must join the queue creator’s in-game party.

### 6.1.3 Rating System

**Function Specification:**

**3.** After completing a gaming session with users meet through the matchmaking service, each user has the ability to rate one another on their performance.

**Function Definition:**

**3.1** Once the Dota game has been completed, users have the option to proceed to the rating page on our app.

**3.2** Matched players will be rated based on their skill level, attitude, and preferred role.

**3.2.1** These skill level and attitude will be rated on a scale of 0 to 10.

**3.2.2** Skill level indicates the degree of mechanical ability

**3.2.3** Attitude corresponds to their behavior and how they acted during situations in-game.

**3.2.4** Preferred role refers to lane positions (carry or support).

**3.3** Ratings will be used internally to help improve matchmaking and will not be seen by users.

**3.3.1** Ratings should not be overly influential since players can have bad games.

**3.3.2** However if a user has an excessive amount of negative rating, a warning will be issued to the user.

### 6.1.4 Add Partner

**Function Specification:**

**4.** Users will be provided with the ability to add matched users they’ve had a positive experience with as partners. If the matched user accepts the partner request then each user will be able to see whether they are online. If both are online they will be able to message each other through the chat system. Messages can be sent while a partner is offline as well. When the partner signs online, they will be notified that a partner has sent them a message. Queues invites can be sent to partners who are online.

**Function Definition:**

**4.1** Users have the option to search for other users by inputting their Dota in-game names into the search bar.

**4.1.1** If the in-game name is found in our database, then return the profile to the user to the searcher.

**4.1.2** Else if the in-game name is not found, then return “X was not found, verify name and try again.” Where X is the input in-game name.

**4.1.3** If user is found, the searcher can send a partner request to the user.

**4.2** Users can also send a partner request after a Dota match has been completed with users found through matchmaking.

**4.2.1** Matched users will appear on the user’s match history.

**4.2.2** Users can now directly request the other user to be a partner.

**4.3** Accepted partners will appear on the user’s Partner tab.

**4.4** Partner tab will display whether partners are online or offline.

**4.5** Users are able to message partners through our chat system.

**4.5.1** Partners who are offline will be able to receive messages once they log on.

**4.6** Queue creators can send invites to partners who are online.

### 6.1.5 Public Posting

**Function Specification:**

**5.** Users can post on the posting tab. Posts made on this tab should be only for matchmaking purposes. All posts in this tab are readable and writable by all users of DotaMatch.

**Function Definition:**

**5.1** To create a public post users make go to the posting tab and select “Create New Post”.

**5.2** Posts should contains specific details on what the user is looking for and their preferences.

**5.3** Users reading the tab are allowed to responds to the posting by commenting on the post.

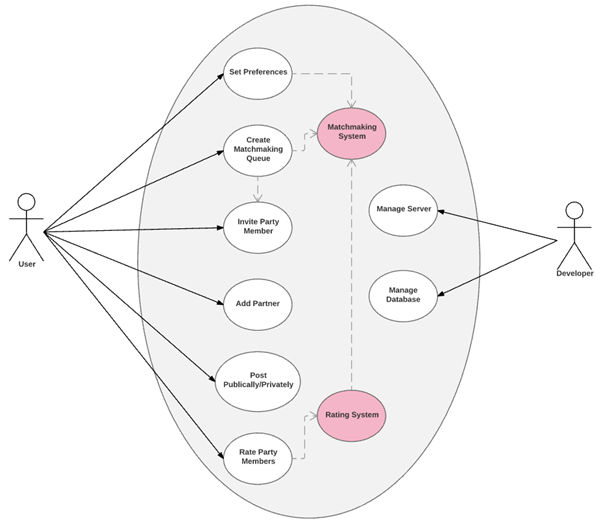
**5.4** Post creators will be notified when other users comment on their post.

**5.5** Posts can be deleted by the creator. Creator must select the delete option on the post.

**5.6** Posts can also be modified by selecting “Edit” on the post.

## 6.2 System Capability Requirements

### 6.2.1 Capabilities



## 6.3 User Interface Requirements

User interface is required to be built with Android’s View and ViewGroup objects. ViewGroup holds other objects including View and ViewGroup. It is used to define the layout of the interface.

## 6.4 Component Architecture

### 6.4.1 Component Descriptions

User Interface (UI):

* UI connects to the server.
* UI is seen by the user and developers.
* User inputs are sent to the server.
* UI receives server responds and is displayed on screen for the user.

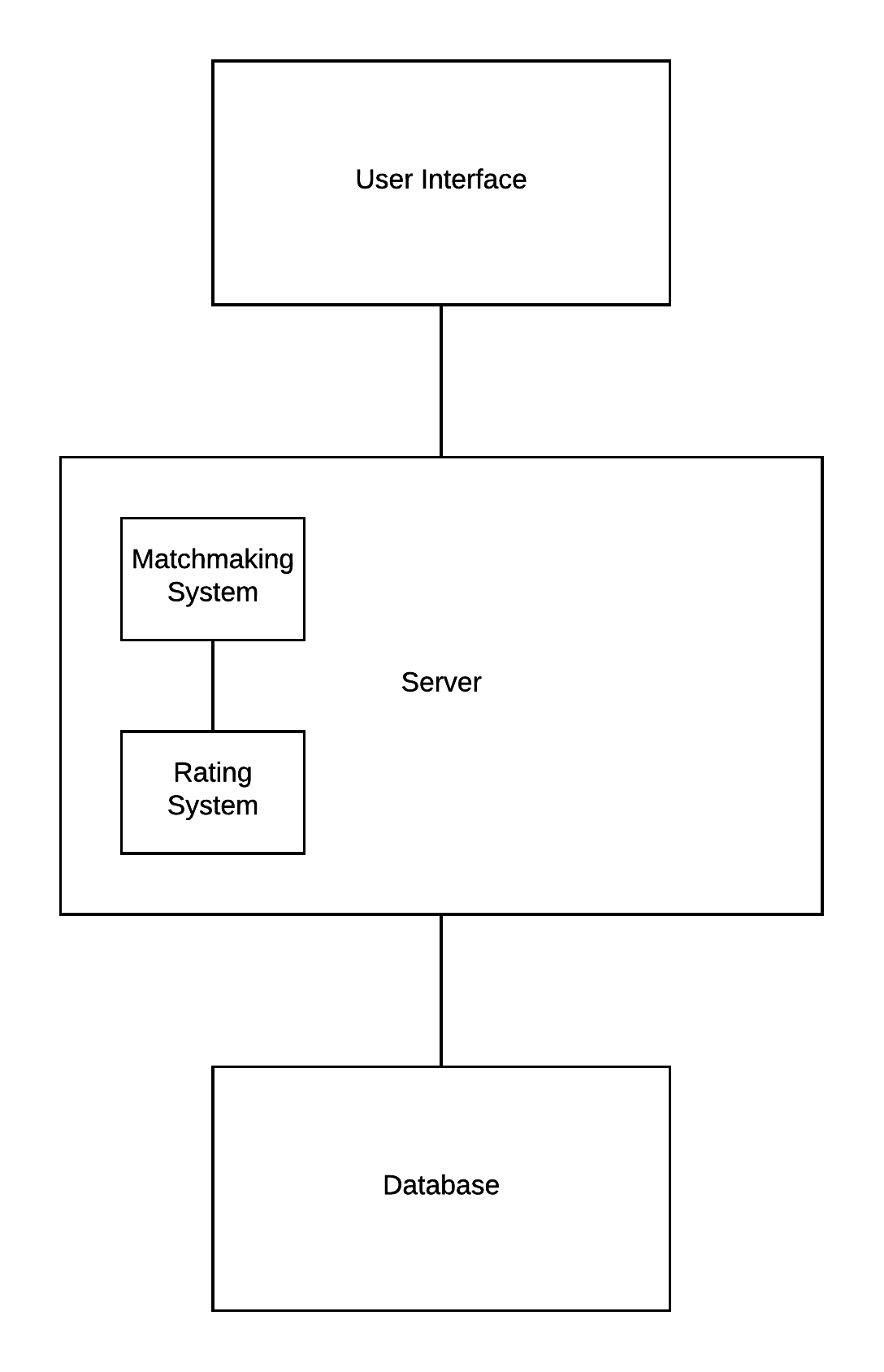
Server:

* Contains methods and functions and interaction dependencies.
* Functions performed are determined by account type.
* Developers have the ability to manage the database and server.
* Users have the ability to create parties, invite members, add partners, start queues, and rate other players.
* Server receives user input from the user interface.
* Server responds to what the user asked based on account permission type.
* Server accesses the database.

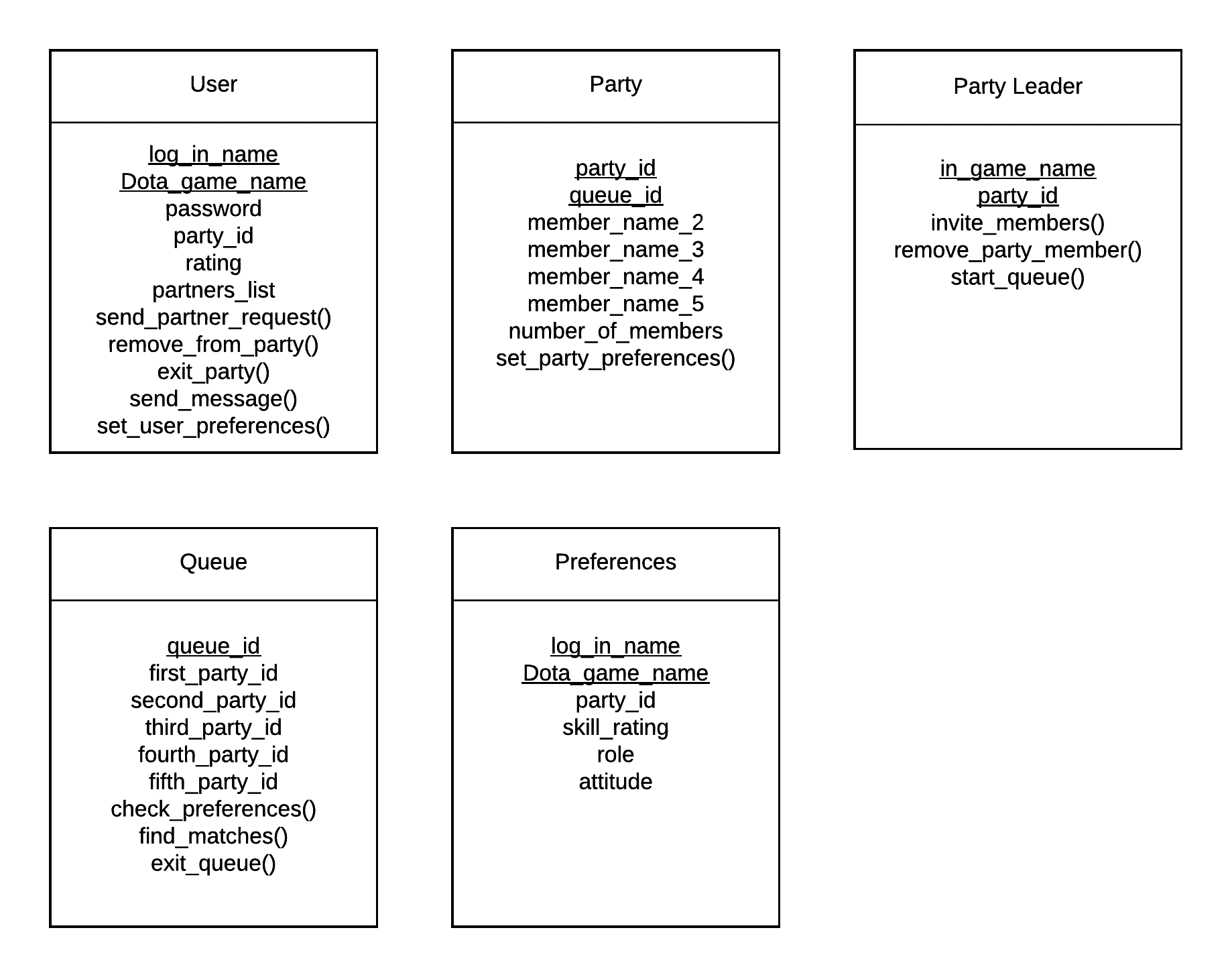
Database:

* Account type can either be user or developer.
* Developers can manage the database.
* Contains individual account information.

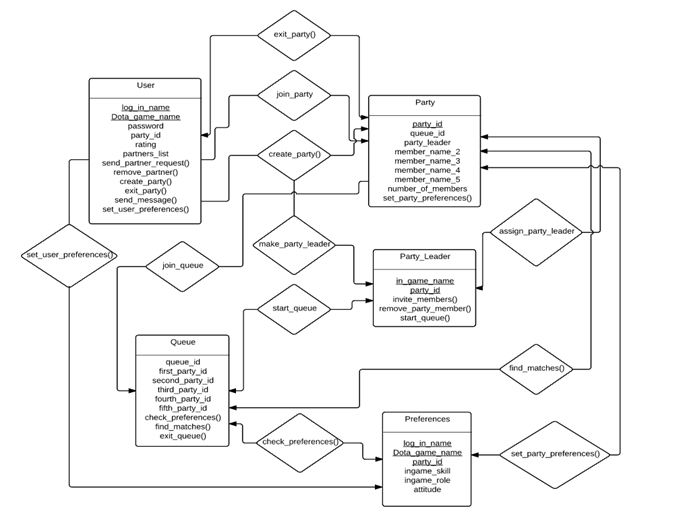
### 6.4.2 Component Architecture Diagram



## 6.5 Class Diagrams



## 6.6 Class Relationship/Interaction Diagrams



## 6.7 Event Section

### 6.7.1 Event Dictionary

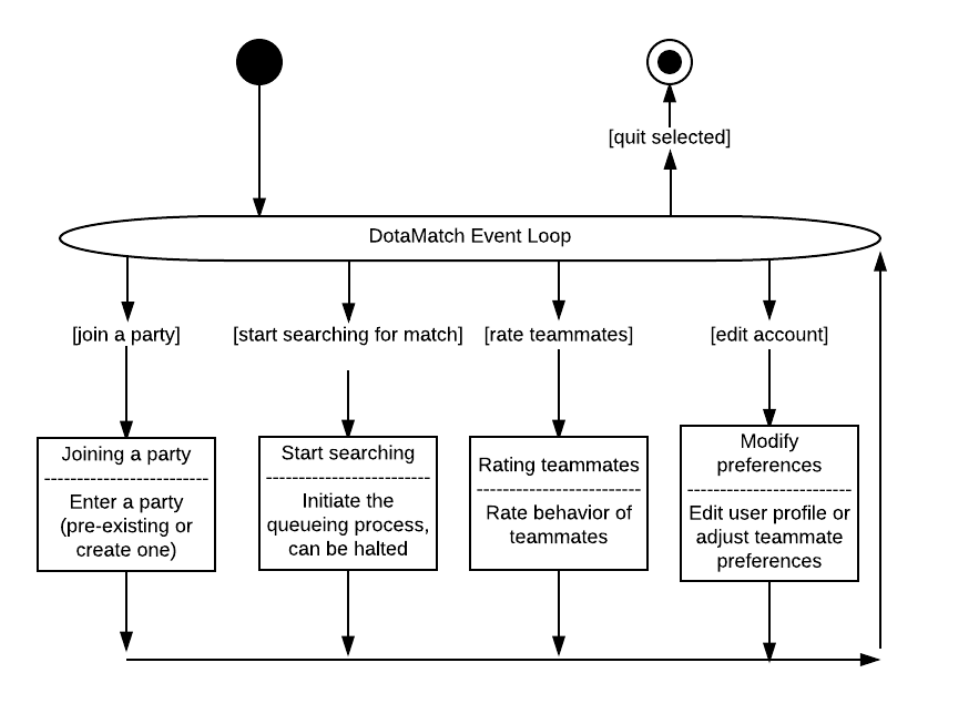
Entering a party -> Be invited to party -> Join or create a party

Party has been created -> All members ready to start -> Initiate queuing process

Match has ended-> Review performance of teammates -> Rate team skill and behavior

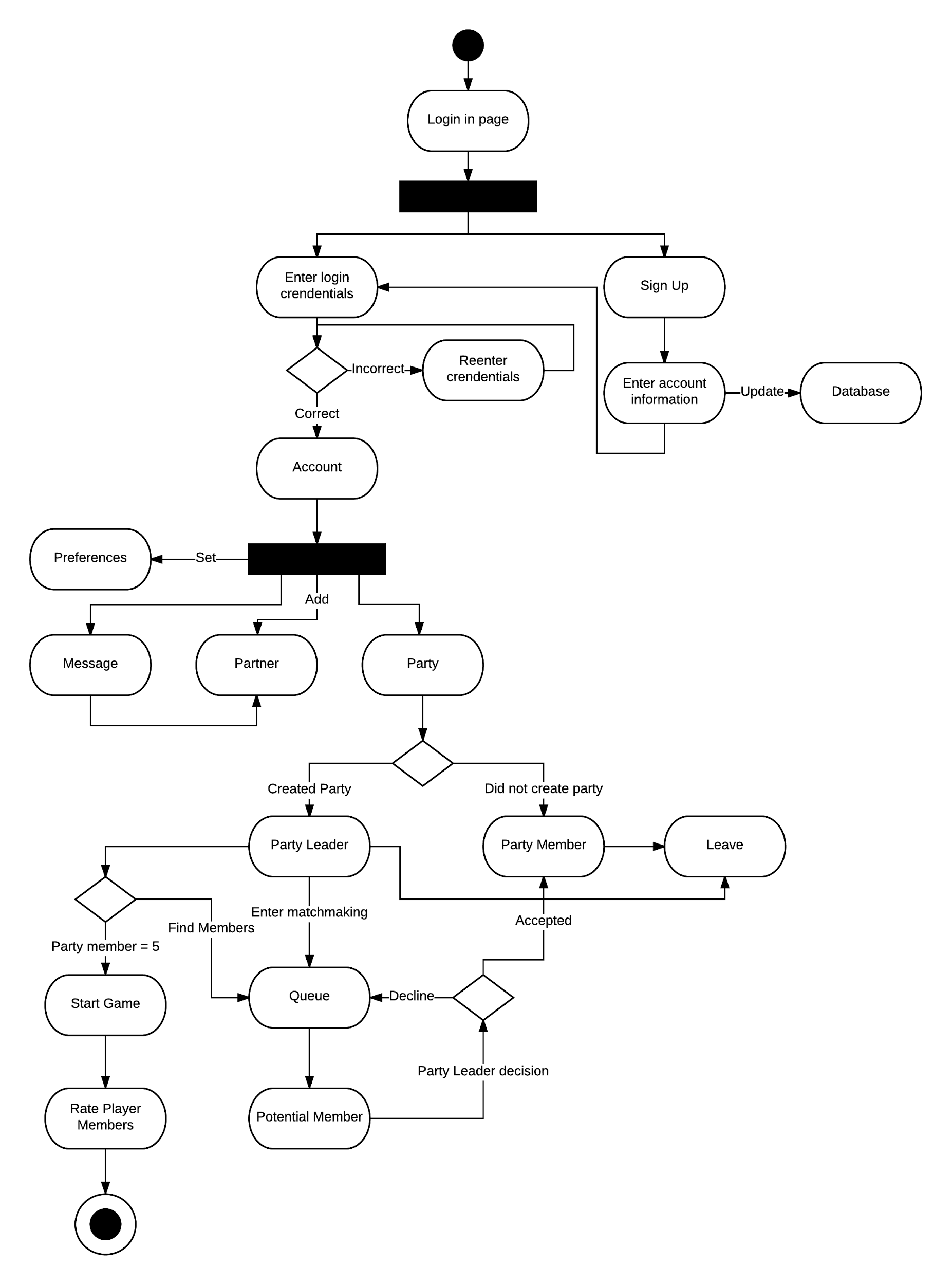
Modifying teammate preferences -> Edit profile -> Change preferred type of teammate

### 6.7.2 Event Diagrams



### 6.8 Activity/State (Scenario) Section

### 6.8.1 Activity (Scenario) Diagrams



### 6.8.2 Activity (Scenario) Specification

**Login Page**

if (User already has an account)

Enter credentials

Check credentials

if (Credentials are correct) Move to account page

else Re-enter credentials

else

Go to sign up page

Enter account information

Return to login page and enter credentials

**Partner**

Send invite to potential partner

Potential partner receives invite

if (Partner agrees)

Add partner to both player’s partner list

else

Do nothing

**Message**

Obtain user’s partner list from database

Choose partner from partner list to send message

Send message to partner

**Preferences**

Change variables on a scale of 0 to 10 based on the value the user set

**Party**

If (You created the party)

Become party leader

Else

Become party member

**Party Leader**

Case1: Enter matchmaking queue by setting pointer to queue object

Case2: Remove chosen party member’s pointer from party object

Case3: Disband party will result in removing all party member pointers from party object

Case4: Have party leader look at potential member’s account information

if (Accept)

Have potential member’s pointer point to party

else (Decline)

Have potential member’s pointer point to queue

Case5: If (Party Members = 5)

Remove party pointer from queue object

Start Game

Else (Part Member != 5)

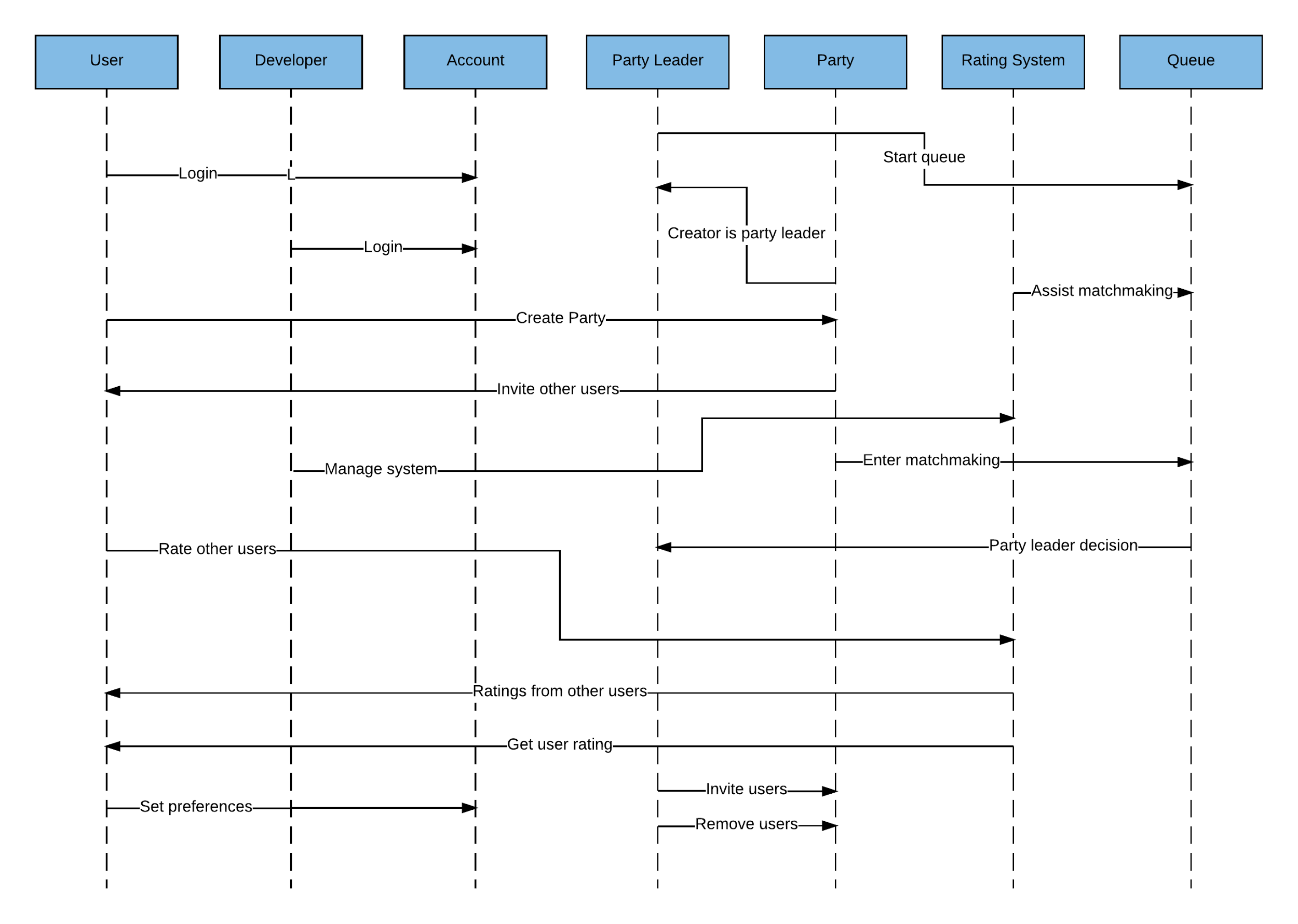
Party pointer still points to queue object

**Rate Player**

Each user will rate their party member on a scale of 0 to 10

Enter user’s values for each individual user attribute

## 6.9 Sequence Diagrams



## 6.10 Collaboration Diagrams

See sequence diagrams.

## 6.11 Dictionaries

Classes

* User - either general user of developer level user.
  + set\_user-preference() - allows user to set preferences to for future party members.
  + send\_partner\_request() - sends a request to another user asking to add them to friends list.
  + send\_message() - user can send messages to other users
  + remove\_from\_party() - if the user is the party leader, they can remove party members from the party.
  + exit\_party() - if the user is the party leader, they can disband the party.
* Party - contains a maximum of 5 users.
  + set\_party\_preferences() - variables conditions are set to look for users who meet the criteria.
* Party Leader - user who created and has control over party functions
  + invite\_member() - send invites to partners to join the party.
  + remove\_party\_member() - removes a user from the party.
  + start\_queue() - place party into matchmaking queue
* Queue - matchmaking queue to find party members.
  + check\_preferences() - use variable conditions to look for party members
  + find\_matches() - find party members.
  + exit\_queues() - take party out of queue.
* Preferences - variables that are preset by users for future party members.

# 7. NON-FUNCTIONAL/OPERATION REQUIREMENTS

## 7.1 System External Interface Requirements

* The product must be able to parse dotabuff.com and yasp.co profile data and migrate them to their respective DotaMatch profiles.
* Linking accounts should require additional authentication through the Steam OpenID service.

## 7.2 Safety Requirements

* None

## 7.3 Security and Privacy Requirements

* Accounts must be validated through email before use.
* Logins and passwords of users must be hashed and be kept securely within the database.
* Users must use registered email as their log in.
* Passwords must be 8-16 characters in letter and includes both letters and numbers.
* User information will not be used in any sort without the user consent.
* Database will have its own security to prevent unauthorized *write*/*delete*. Read access will only be given to personnel with clearance.
* Servers will have its own security to prevent unauthorized access. Only approved personnel may access servers.

## 7.4 System Environment Requirements

The environment in which the product operates in will constantly change through the life cycle. Appropriate changes must be made to in response to these environmental changes. Changes must not alter specifications.

## 7.5 Computer Resource Requirements

### 7.5.1 Computer Hardware Requirements

* Standard consumer hardware.
* Android mobile device.

### 7.5.2 Computer Hardware Resource Requirements

* Android device must have Lollipop Version 5.0 or higher.

### 7.5.3 Computer Software Requirements

* Android SDK

### 7.5.4 Computer Communications Requirements

* Server needs to be able communicate with the database.
* Server need to be able to communicate with dotabuff.com and yasp.co.
* User Interface needs to be able to communicate with the server.

## 7.6 System Quality Factors

* Runtime of matchmaking algorithm.
* If algorithm takes too long to find matches, then users will be discouraged and will not use DotaMatch.
* Number of users.
* If there are not enough users using DotaMatch than matchmaking times will increase.
* Servers must have high speed Internet capability to decrease lag and response time.
* Aggregate reviews on the Google Play store platform.

## 7.7 Design and Construction Constraints

* Must be implemented with an object-oriented programming language.
* Interface must be responsive (adjust for both landscape and portrait orientation.)
* Interface is only available in English.
* Only registered users who have linked their Dota profiles may use the product.
* Only implemented for Android devices

## 7.8 Personnel-Related Requirements

* Must be a NYU Tandon School of Engineering student.
* Must be taking CS 4523 Design Project during the Fall 2016 semester. Team members are allowed to leave the project at any given time.
* Must be part of Team B11.
* Must be able to work with others in a team environment.
* Must present work on time.

## 7.9 Training-Related Requirements

* Software engineering knowledge
* Life Cycle and workflow knowledge
* C++/C programming language
* Java programming language
* PostgreSQL
* Server management
* Database management

## 7.10 Logistics-Related Requirements

Amount of human hours worked must be tracked and updated in Section 14. Faults detected by each individual must also be tracked and updated in Section 14.

## 7.11 Packaging Requirements

Packaging is not required. DotaMatch is a software application which is downloaded through the Google Play Store and installed on mobile devices. Compatible devices include android phones.

## 7.12 Precedence and Critically Requirements

Development of product should follow specifications and requirements. Product should be tested during each workflow to ensure faults are found early. Work must be presented on time. These requirements are the most critical, however, all other requirements should also be followed with equal precedence in mind.

## 7.13 Other Requirements

No other requirements.

# 8. SYSTEM TEST PLAN REQUIREMENTS

* System test plans will be developed and conducted by the Software Quality Group (SQA).
* SQA will also run the product test.
* Product will be tested at the end of each workflow
* Product will be tested with a small sample of users to determine that the implemented functions work as intended.
* After implementation testing is successful, a larger user base will be introduced to test the endurance of the system.
* Acceptance test will be conducted by the client. The client can ask the SQA to assist them but it is the responsibility of the client to approve the product.

# 9. QUALIFICATION PROVISIONS

## 9.1 Review Process

### 9.1.1 Self-Check

Each member of the project is required to self-check the Software Requirements Specification (SRS). During the self-check, each reviewer will be thoroughly looking for correctness, ambiguity, completeness, consistency, stability, verifiability, modifiability (malleable) and traceability. Undiscovered faults can lead to future faults. After completing the self-check, each member must sign and date.

### 9.1.2 Walkthrough

A walkthrough team consist of four to six individuals. These members should be experienced senior technical staff members because they tend to find faults that would have a major negative impact on the project. There will be at least one representative for each workflow from the project group. There is also a client representative and a SQA representative. Material for the walkthrough needs to be distributed to the participants in advance. Each reviewer should study the material and develop two lists. One list should contain items the reviewer does not understand. The other list should contain items the reviewer believes to be incorrect. The person leading the walkthrough will be the SQA representative. There are two ways to conduct the walkthrough. The first is where the participant presents their list of unclear items and items they think are correct. The second is document driven. The person responsible for each document should walk the participant through the document. As a result the second type is interactive between the presenter and the participants. Faults found throughout the walkthrough are to be recorded and corrected later on.

### 9.1.3 Inspection

Inspections have five formal steps and should go beyond a walkthrough. The first step is an overview of the document to be inspected is provided to the participants. In the preparation step, the participants try to understand the document in detail. Next the participant walks through the document with the inspection team. Fault finding commences afterward. Within one day, the leader of the inspection team (the moderator) must produce a written report of the inspection. The next step is rework, where those responsible for the documents correct faults that were noted on the written report. The last step is follow-up, where the moderator must ensure that every issue raised has been resolved satisfactorily, by either fixing the document or clarifying items incorrectly flagged as faults. If more than 5 percent of the material inspected has been reworked, then the team must reconvene for a 100 percent re-inspection.

# 10. REQUIREMENTS TRACEABILITY

Requirements should have forward (to all artifacts spawned by this document) and backward (to previous stages of development) traceability. Forward traceability will allow tracing in a forward direction tracing from requirements to postdelivery workflows. While backward traceability allows tracing in the direction of postdelivery to requirements workflows.

# 11. RATIONALE

No additional rationales.

# 12. NOTES

No additional notes.

# 13. APPENDICES

## 13.1 Dictionaries

Can be found in Section 6.11.

## 13.2 UML Diagrams

Included in body of document.

## 13.3 Schedule Tracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (Individual or Team) | Estimated | Actual | Difference |
| Software Requirements Specification (SRS) Version 1.0 | Corey Chong | 6 | 7 | 1 |
| Software Requirements Specification (SRS) Version 1.0 | Albert Su | 6 | 7 | 1 |
| Software Requirements Specification (SRS) Version 1.0 | Evans Yeung | 6 | 7 | 1 |
| Software Requirements Specification (SRS) Version 2.0 | Corey Chong | 5 | 4 | 1 |
| Software Requirements Specification (SRS) Version 2.0 | Albert Su | 5 | 5 | 0 |
| Software Requirements Specification (SRS) Version 2.0 | Evans Yeung | 6 | 8 | 2 |
| Requirements and Analysis Specification (RAS) Version 1.0 | Corey Chong | 2 | 2 | 0 |
| Requirements and Analysis Specification (RAS) Version 1.0 | Evans Yeung | 4 | 3 | 1 |
| Requirements and Analysis Specification (RAS) Version 1.0 | Albert | 1 | 2 | 1 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Corey Chong | 37 | 34 | 3 |
| Albert Su | 33 | 31 | 2 |
| Evans Yeung | 44 | 40 | 4 |

## 13.4 Defect Tracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (Individual or Team) | Estimated | Actual | Difference |
| Software Requirements Specification (SRS) Version 1.0 | Corey Chong | 15 | 20 | 5 |
| Software Requirements Specification (SRS) Version 1.0 | Albert Su | 14 | 17 | 3 |
| Software Requirements Specification (SRS) Version 1.0 | Evans Yeung | 18 | 25 | 7 |
| Software Requirements Specification (SRS) Version 2.0 | Corey Chong | 15 | 22 | 6 |
| Software Requirements Specification (SRS) Version 2.0 | Albert Su | 20 | 24 | 4 |
| Software Requirements Specification (SRS) Version 2.0 | Evans Yeung | 14 | 15 | 1 |
| Requirements and Analysis Specification (RAS) Version 1.0 | Corey Chong | 10 | 8 | 2 |
| Requirements and Analysis Specification (RAS) Version 1.0 | Albert Su | 7 | 8 | 1 |
| Requirements and Analysis Specification (RAS) Version 1.0 | Evans Yeung | 15 | 6 | 9 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Corey Chong | 111 | 123 | 8 |
| Albert Su | 118 | 116 | 2 |
| Evans Yeung | 120 | 127 | 7 |

## 13.5 Gantt Chart

