

**Computer Science and Engineering**

**DotaMatch**

**System Analysis Specification**

**Version 1.1**

Document Number: SAS-001

Project Team Number A11

Project Team Members:

Corey Chong cc3899

Albert Su as7353

Evans Yeung eby210

**REVIEWS AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function (Author, Reviewer, Approval)** | **Date** | **Signature** |
| Evans Yeung | Author | 4/14/2016 | Evans Yeung |
| Corey Chong | Reviewer | 4/17/2016 | Corey Chong |
| Albert Su | Reviewer | 4/17/2016 | Albert Su |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| April 14, 2016 | Version 1.0 | Initial Release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[1. INTRODUCTION 7](#_Toc448689774)

[1.1 Purpose 7](#_Toc448689775)

[2. SCOPE 7](#_Toc448689776)

[2.1 Identification 7](#_Toc448689777)

[2.2 Bounds 7](#_Toc448689778)

[2.3 Objectives 7](#_Toc448689779)

[2.4 System Overview 8](#_Toc448689780)

[2.5 Document Overview 8](#_Toc448689781)

[3. REFERENCE DOCUMENTS 9](#_Toc448689782)

[4. BUSINESS REQUIREMENTS 9](#_Toc448689783)

[4.1 Technology 9](#_Toc448689784)

[4.2 Economics 10](#_Toc448689785)

[4.3 Regulatory and Legal 10](#_Toc448689786)

[4.4 Market Considerations 10](#_Toc448689787)

[4.5 Risks and Alternatives 10](#_Toc448689788)

[4.6 Human Resources and Training 11](#_Toc448689789)

[5. LOGICAL ARCHITECTURAL SPECIFICATION 12](#_Toc448689790)

[5.1 Context Diagram 12](#_Toc448689791)

[5.2 System Capability Requirements 12](#_Toc448689792)

[*5.2.1 Capabilities* 12](#_Toc448689793)

[*5.2.2 Use Case Diagrams* 13](#_Toc448689794)

[*5.2.3 Use Case Descriptions* 14](#_Toc448689795)

[5.3 User Interface Requirements 15](#_Toc448689796)

[5.4 Component (Component/Package/Subsystem) Architecture 15](#_Toc448689797)

[*5.4.1 Component Descriptions* 15](#_Toc448689798)

[*5.4.2 Component Architecture Diagram* 16](#_Toc448689799)

[5.5 Class Diagrams 17](#_Toc448689800)

[5.6 Class Relationship/Interaction Diagrams 18](#_Toc448689801)

[5.7 Events 19](#_Toc448689802)

[*5.7.1 Motives* 20](#_Toc448689803)

[*5.7.2 Event Diagrams* 20](#_Toc448689804)

[5.8 Activity/State (Scenario) Section 20](#_Toc448689805)

[5.9 State Logic 20](#_Toc448689806)

[5.10 Behavior 20](#_Toc448689807)

[*5.10.1 Sequence Diagrams* 20](#_Toc448689808)

[*5.10.2 Collaboration Diagrams* 20](#_Toc448689809)

[5.11 Dictionaries 20](#_Toc448689810)

[6. NON-FUNCTIONAL/OPERATIONAL SPECIFICATIONS 21](#_Toc448689811)

[6.1 System External Interface Requirements 21](#_Toc448689812)

[6.2 Safety Requirements 21](#_Toc448689813)

[6.3 Security and Privacy Requirements 21](#_Toc448689814)

[6.4 System Environment Requirements 22](#_Toc448689815)

[6.5 Computer Resource Requirements 22](#_Toc448689816)

[*6.5.1 Computer Hardware Requirements* 22](#_Toc448689817)

[*6.5.2 Computer Hardware Resource Requirements* 22](#_Toc448689818)

[*6.5.3 Computer Software Requirements* 22](#_Toc448689819)

[*6.5.4 Computer Communications Requirements* 22](#_Toc448689820)

[6.6 System Quality Factors 22](#_Toc448689821)

[6.7 Design and Construction Constraints 23](#_Toc448689822)

[6.8 Personnel-Related Requirements 23](#_Toc448689823)

[6.9 Training-Related Requirements 23](#_Toc448689824)

[6.10 Logistics-Related Requirements 24](#_Toc448689825)

[6.11 Packaging Requirements 24](#_Toc448689826)

[6.12 Precedence and Criticality Requirements 24](#_Toc448689827)

[6.13 Other Requirements 24](#_Toc448689828)

[7. SYSTEM TEST PLAN REQUIREMENTS 24](#_Toc448689829)

[8. QUALIFICATION PROVISIONS 25](#_Toc448689830)

[8.1 Self Check 25](#_Toc448689831)

[8.2 Walkthrough 25](#_Toc448689832)

[8.3 Inspection 26](#_Toc448689833)

[9. REQUIREMENTS TRACEABILITY 26](#_Toc448689834)

[10. RATIONALE 26](#_Toc448689835)

[11. NOTES 26](#_Toc448689836)

[12. APPENDICES 27](#_Toc448689837)

[12.1 Dictionaries 27](#_Toc448689838)

[12.2 UML Diagrams 27](#_Toc448689839)

[12.3 Schedule Tracking 27](#_Toc448689840)

[12.4 Defect Tracking 28](#_Toc448689841)

[12.5 Gantt Chart 30](#_Toc448689842)

# 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

Systems Analysis Specification, SAS-001, Version 1.0, April 14, 2016

## 2.2 Bounds

* One basic type of user.
* Users can create a profile, send private messages, and start a matchmaking queue.
* Login and password is used for the identification of users.

## 2.3 Objectives

* Priority will be given to developing an interface the users can use to coordinate teams and review various statistics.
* Iterative and Incremental lifecycle
* Software Requirement (SRS) Version 1.0 3/7/2016
* Software Requirement (SRS) Version 2.0 3/21/2016
* Software Project Management Plan (SPMP) 4/6/2016
* Software Analysis Specification (SAS) 4/18/2016
* Software Design Document (SDD) 4/25/2016
* 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.
* Section 5 contains the context, use case, component, class, and event diagrams and other information on how the system works.
* Section 6 contains the various requirements for the product.
* Section 7 contains the testing plan requirement.
* Section 8 contains the types of reviews.
* Section 9 contains the traceability of documents backwards and forward.
* Section 10 contains additional rationale for implementation.
* Section 11 contains additional notes for the product.
* Section 12 contains the appendices, including dictionaries, UML diagrams, schedule tracking, and defect tracking.

# 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 will be submitted to NYU Classes on April 18, 2016.Once submitted it can be referenced through NYU Classes. Software Project Management Plan (SPMP) Version 1.0 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

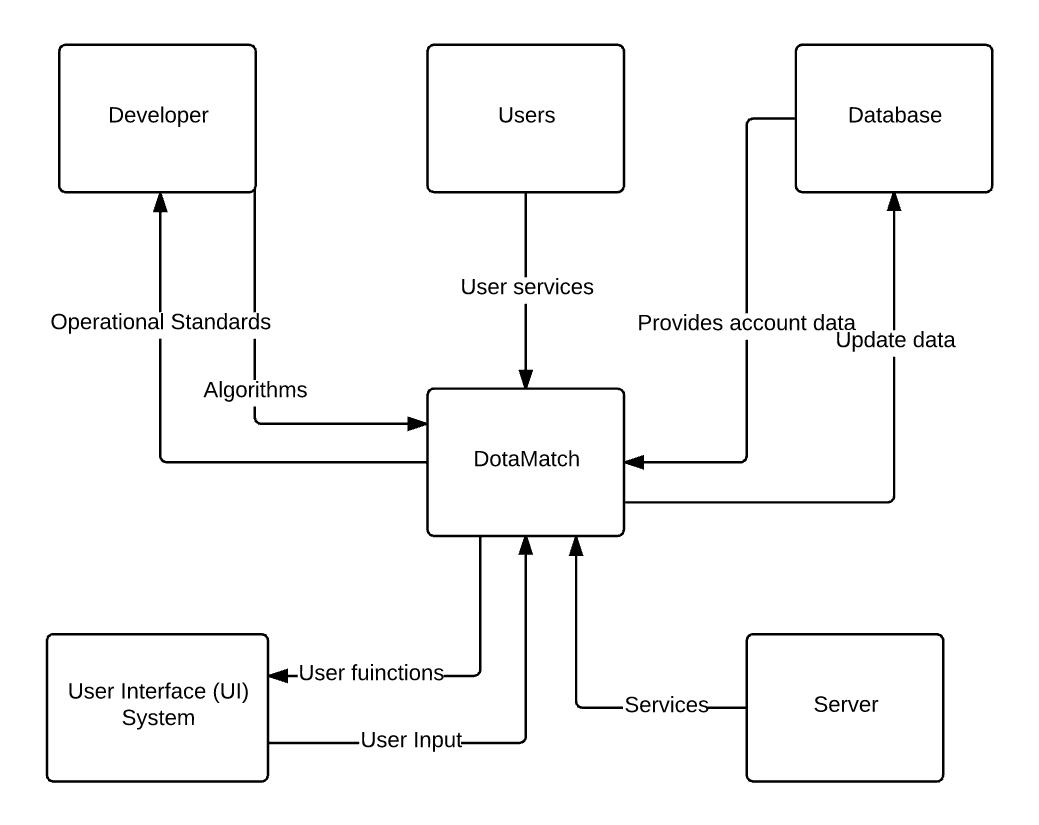
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Probability | Detection | Person Responsible For Addressing Risk | Current Status |
| Lateness | 25% | Scheduling Chart  Gantt Chart | Members of Team A11 | On time |
| Over-budget | 0% | Project does not require a budget | Members of Team A11 | Project does not require a budget |
| Residual Fault | 50% | Self-check  Walkthrough  Inspection  Testing by SQA | Members of Team A11 | Project currently follow requirements set in SRS |
| Member May Leave | 25% | Will be informed by leaving member | Members of Team A11  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. LOGICAL ARCHITECTURAL SPECIFICATION

## 5.1 Context Diagram



## 5.2 System Capability Requirements

### *5.2.1 Capabilities*

Reliability:

* System must provide the right matches more than 99% of the time.

Availability:

* System must be available to users more than 99% of the time.

Security:

* System must be secure. Users must log in with their login and passwords.

Maintainability:

* Corrective Maintenance - removal of residual faults while leaving specifications unchanged.
* Perfective maintenance - application can be extended if the client believes it will improve the effectiveness of the product. New implementation of functions should not create regression faults.
* Adaptive maintenance - changes made in response to changes in environment in which the application operates.

### 

### *5.2.2 Use Case Diagrams*

### 

Manage Database



Manage Server

Developer

Create Matchmaking Queue

Login

Matchmaking System

User



Add partner

Post publically

Rating System

Rate other users

### *5.2.3 Use Case Descriptions*

|  |  |
| --- | --- |
| System Use Case | |
| Description | Users and developers have to log into the server using log-in credentials. Developers have the ability to manage the database and server. Users have the ability to create a matchmaking queue, add partners, post publically, and rate other users. |
| Flows | 1. User or developer logs into the server using log-in credentials. 2. Server will check the input log-in credentials against the credentials in the database. 3. If approved, user or developer will be brought to the user interface. If not approved, server will ask the user to try again. 4. User can create a party. User has the options to invite partners before starting queue. 5. Once matchmaking queue has started, it will find potential party members. 6. Party leader will have the option to add matched players to the party. 7. Once game has finished, users have the option to rate other users in the party. |
| Post Conditions | User is able to get matched with and play with other players. |
| Special Requirements | Must have already created a DotaMatch account in order for log-in to be successful. |

## 5.3 User Interface Requirements

Prototyping is not required in the Iterative and Incremental life cycle model.

## 5.4 Component (Component/Package/Subsystem) Architecture

### *5.4.1 Component Descriptions*

User Interface (UI):

* UI connects to the server.
* UI is seen by the user and developers.

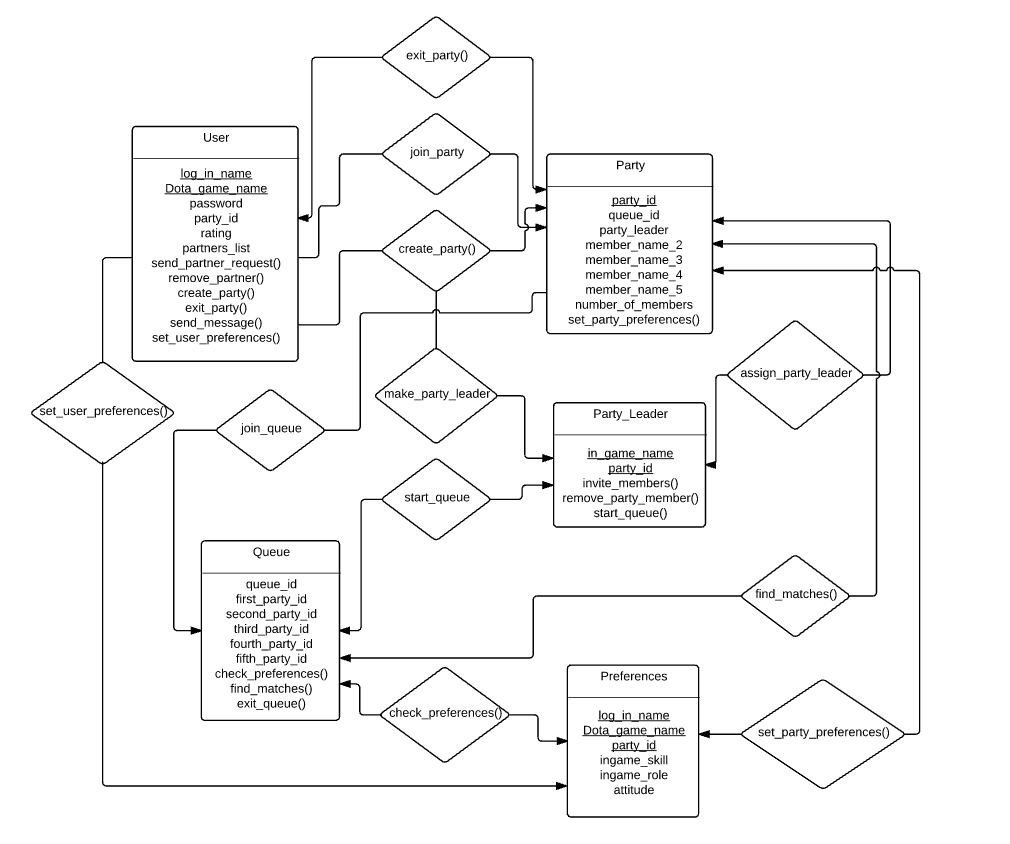
Server:

* Contains functions and how they interact.
* Functions 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 connects with the database.

Database:

* Contains individual account information.
* An account type can either be user or developer.

### *5.4.2 Component Architecture Diagram*



Server

UI

Database

## 5.5 Class Diagrams

User

Administrator

Party

party\_id

queue\_id

party\_leader

member\_name\_2

member\_name\_3

member\_name\_4

member\_name\_5

number\_of\_members

set\_party\_preferences()

log\_in

Dota\_game\_name

password

party\_id

role\_in\_party

rating

partners\_list

send\_partner\_request()

remove\_partner()

create\_party()

exit\_party()

send\_message()

set\_user\_preferences()

Queue

queue\_id

first\_party\_id

second\_party\_id

third\_party\_id

fourth\_party\_id

check\_preferences()

find\_matches()

exit\_queue()

name

first\_name

last\_name

level\_of\_clearance

remove\_from\_ database()

add\_to\_database()

Party\_Leader

party\_id

in\_game\_name

invite\_members()

remove\_party\_member()

start\_queue()

Preferences

log\_in\_name

Dota\_game\_name

party\_id

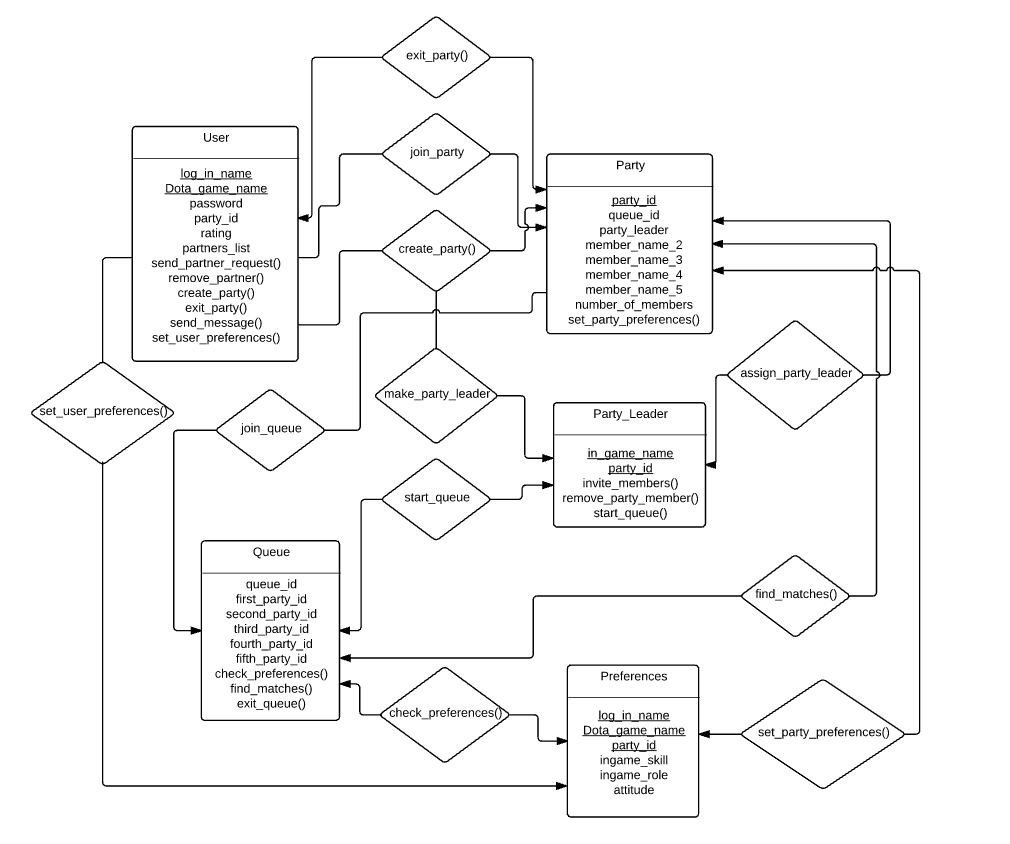
ingame\_skill

ingame\_role

attitude

## 

## 5.6 Class Relationship/Interaction Diagrams



## 

## 5.7 Events

* Account Registration
* Joining Party
* Public Text Post
* Rating Players
* Concluding Matches
* Edit Credentials

### *5.7.1 Motives*

The application shall improve matchmaking quality according to the user’s preferences.

### *5.7.2 Event Diagrams*

Credential   
Management

Direct

Messaging

Add Friends

User

Join Party

Rate

Teammates

Public Text

Post

## 5.8 Activity/State (Scenario) Section

To be completed in CS-4523 Senior Design.

## 5.9 State Logic

To be completed in CS-4523 Senior Design.

## 5.10 Behavior

* The application should respond only to user input and instructions set forth by the database.

### *5.10.1 Sequence Diagrams*

Queue

Party Member

Party

Leader

User

Manage

Database

Manage

Server

Administrator

­­­

### 

### *5.10.2 Collaboration Diagrams*

To be completed in CS-4523 Senior Design.

## 5.11 Dictionaries

* log\_in\_name
* Dota\_Game\_Name
* exit\_party()
* join\_party()
* create\_party()
* make\_party\_leader()
* join\_queue
* start\_queue
* set\_user\_preferences()
* assign\_party\_leader
* Party\_Leader
* find\_matches()
* set\_party\_preferences()
* check\_preferences()
* party\_id

# 6. NON-FUNCTIONAL/OPERATIONAL SPECIFICATIONS

## 6.1 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.

## 6.2 Safety Requirements

All documents must be updated incrementally. Each member must self-check. The product must go through testing. Following these procedures will mitigate risk. Risk include lateness, over-budget, and residual faults. These risk must be calculated through various charts found in the Software Requirement (SRS), Software Project Management Plan (SPMP), and Software Analysis Specification (SAS).

## 6.3 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.

## 6.4 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.

## 6.5 Computer Resource Requirements

### *6.5.1 Computer Hardware Requirements*

* Standard consumer hardware.
* Android mobile device
* Development device

### *6.5.2 Computer Hardware Resource Requirements*

* Android device must have Lollipop Version 5.0 or higher.

### *6.5.3 Computer Software Requirements*

* Android SDK

### *6.5.4 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.

## 6.6 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.
* Amount 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.

## 6.7 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.

## 6.8 Personnel-Related Requirements

* Must be an NYU student.
* Must be taking CS 4513 Software Engineering.
* Should be taking CS 4523 Design Project in the following semester (not required). Team members are allowed to leave the project at any given time.
* Must be part of Team A11.
* Must be able to work with others in a team environment.
* Must present work on time.

## 6.9 Training-Related Requirements

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

## 6.10 Logistics-Related Requirements

Amount of human hours worked must be tracked and updated in Section 12. Faults detected by each individual must also be tracked and updated in Section 12.

## 6.11 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.

## 6.12 Precedence and Criticality 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.

## 6.13 Other Requirements

No other additional requirements.

# 7. SYSTEM TEST PLAN REQUIREMENTS

* 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.
* Software Quality Assurance (SQA) group will test the product against the requirements.

# 8. QUALIFICATION PROVISIONS

## 8.1 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.

## 8.2 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.

## 8.3 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.

# 9. REQUIREMENTS TRACEABILITY

All documents can be referenced through NYU Classes. All documents 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.

# 10. RATIONALE

No additional rationale.

# 11. NOTES

No additional notes.

# 12. APPENDICES

## 12.1 Dictionaries

Can be found in Section 5.11.

## 12.2 UML Diagrams

Can be found in Section 5.10.1.

## 12.3 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 |
| Software Project Management Plan (SPMP)  Version 1.0 | Corey Chong | 10 | 8 | 2 |
| Software Project Management Plan (SPMP)  Version 1.0 | Albert Su | 8 | 5 | 3 |
| Software Project Management Plan (SPMP)  Version 1.0 | Evans Yeung | 12 | 9 | 3 |
| System Analysis Specification (SAS) Version 1.0 | Corey Chong | 5 | 6 | 1 |
| System Analysis Specification (SAS) Version 1.0 | Albert Su | 5 | 6 | 1 |
| System Analysis Specification (SAS) Version 1.0 | Evans Yeung | 12 | 10 | 2 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Corey Chong | 26 | 25 | 1 |
| Albert Su | 24 | 23 | 1 |
| Evans Yeung | 36 | 34 | 2 |

## 12.4 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 | 7 |
| Software Requirements Specification (SRS) Version 2.0 | Albert Su | 20 | 24 | 4 |
| Software Requirements Specification (SRS) Version 2.0 | Evans Yeung | 14 | 15 | 1 |
| Software Project Management Plan (SPMP)  Version 1.0 | Corey Chong | 25 | 23 | 2 |
| Software Project Management Plan (SPMP)  Version 1.0 | Albert Su | 24 | 16 | 8 |
| Software Project Management Plan (SPMP)  Version 1.0 | Evans Yeung | 19 | 24 | 5 |
| System Analysis Specification (SAS) Version 1.0 | Corey Chong | 16 | 12 | 4 |
| System Analysis Specification (SAS) Version 1.0 | Albert Su | 15 | 10 | 5 |
| System Analysis Specification (SAS) Version 1.0 | Evans Yeung | 30 | 28 | 2 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Corey Chong | 71 | 77 | 6 |
| Albert Su | 73 | 67 | 6 |
| Evans Yeung | 81 | 92 | 11 |

## 12.5 Gantt Chart

