

**Computer Science and Engineering**

**DotaMatch**

**Software Engineering Standards**

**System Requirements Specification**

**Version 2.0**

Document Number SRS-001

Project Team Number A11

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 | 3/5/2016 | Evans Yeung |
| Evans Yeung | Reviewer | 3/5/2016 | Evans Yeung |
| Corey Chong | Reviewer | 3/6/2016 | Corey Chong |
| Albert Su | Reviewer | 3/7/2016 | Albert Su |
| Albert Su | Approval | 3/7/2016 | Albert Su |
| Evans Yeung | Author | 3/18/2016 | Evans Yeung |
| Evans Yeung | Reviewer | 3/22/2016 | Evans Yeung |
| Corey Chong | Reviewer | 3/22/2016 | Corey Chong |
| Corey Chong | Approval | 3/22/2016 | Corey Chong |
| Albert Su | Reviewer | 3/22/2016 | Albert Su |
| Albert Su | Approval | 3/22/2016 | Albert Su |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| March 5, 2016 | Version 1.0 | Initial Release |
| March 18, 2016 | Version 2.0 | Modification/Correction |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[**1. INTRODUCTION** 7](#_Toc447646789)

[**1.1 Purpose** 7](#_Toc447646790)

[**2. SCOPE** 7](#_Toc447646791)

[**2.1 Identification** 7](#_Toc447646792)

[**2.2 Bounds** 8](#_Toc447646793)

[**2.3 Objectives** 8](#_Toc447646794)

[**3. OVERALL SYSTEM OVERVIEW** 8](#_Toc447646795)

[**3.1 Context Diagram** 8](#_Toc447646796)

[**3.2 Additional Descriptive Items** 9](#_Toc447646797)

[***3.2.1 Product Functions*** 9](#_Toc447646798)

[***3.2.2 User characteristics*** 9](#_Toc447646799)

[**4. DOCUMENT OVERVIEW** 9](#_Toc447646800)

[**4.1 Description of Document** 9](#_Toc447646801)

[**4.2 Organization** 10](#_Toc447646802)

[**5. REFERENCE DOCUMENTS** 11](#_Toc447646803)

[**6. BUSINESS REQUIREMENTS** 11](#_Toc447646804)

[**6.1 Technology** 11](#_Toc447646805)

[**6.2 Economics** 11](#_Toc447646806)

[**6.3 Regulatory and Legal** 11](#_Toc447646807)

[**6.4 Market Considerations** 11](#_Toc447646808)

[**6.5 Risks and Alternatives** 11](#_Toc447646809)

[**6.6 Human Resources and Training** 12](#_Toc447646810)

[**7. SPECIFIC REQUIREMENTS (DESCRIPTIVE FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS)** 12](#_Toc447646811)

[**7.1 Functional Descriptive Detailed Requirements** 12](#_Toc447646812)

[***7.1.1 Account Registration*** 12](#_Toc447646813)

[***7.1.2 Matchmaking Service*** 13](#_Toc447646814)

[***7.1.3 Rating System*** 14](#_Toc447646815)

[***7.1.4 Add Partner*** 14](#_Toc447646816)

[***7.1.5 Public Posting*** 15](#_Toc447646817)

[**7.2 Requirement Use Cases** 16](#_Toc447646818)

[***7.2.1 Use Case Diagram*** 16](#_Toc447646819)

[***7.2.2 Use Case Descriptions*** 17](#_Toc447646820)

[**7.3 Non-Functional Descriptive Detailed Requirements** 17](#_Toc447646821)

[***7.3.1 System Capabilities, conditions, and constraints*** 17](#_Toc447646822)

[***7.3.2 Physical Resource Requirements*** 17](#_Toc447646823)

[***7.3.2.2 Computer Hardware Resources Requirements*** 18](#_Toc447646824)

[***7.3.2.3 Computer Software Requirements*** 18](#_Toc447646825)

[***7.3.2.4 Computer Communications Requirements*** 18](#_Toc447646826)

[***7.3.3 Environmental Conditions*** 18](#_Toc447646827)

[***7.3.4 System Performance Characteristics*** 18](#_Toc447646828)

[***7.3.5 Safety Requirements*** 19](#_Toc447646829)

[***7.3.6 Security and Privacy Requirements*** 19](#_Toc447646830)

[***7.3.7 System Human Interfaces*** 19](#_Toc447646831)

[***7.3.8 System Maintainability*** 19](#_Toc447646832)

[***7.3.9 System Quality Factors*** 20](#_Toc447646833)

[***7.3.10 Design and Construction Constraints*** 20](#_Toc447646834)

[***7.3.10.1 Life Cycle Model*** 20](#_Toc447646835)

[***7.3.10.2 Policies and standards - Methods, tools, and techniques*** 20](#_Toc447646836)

[***7.3.11 Personnel-Related Requirements*** 21](#_Toc447646837)

[***7.3.12 Training-Related Requirements*** 21](#_Toc447646838)

[***7.3.13 Logistics-Related Requirements*** 21](#_Toc447646839)

[***7.3.14 Packaging Requirements*** 22](#_Toc447646840)

[***7.3.15 Precedence and Criticality Requirements*** 22](#_Toc447646841)

[***7.3.16 Other non-functional Requirements*** 22](#_Toc447646842)

[**8. SYSTEM TEST PLAN REQUIREMENTS** 22](#_Toc447646843)

[**9. QUALIFICATION PROVISIONS** 22](#_Toc447646844)

[**9.1 Review Process** 22](#_Toc447646845)

[***9.1.1 Self-Check*** 22](#_Toc447646846)

[***9.1.2 Walkthrough*** 23](#_Toc447646847)

[***9.1.3 Inspection*** 23](#_Toc447646848)

[**10. REQUIREMENTS TRACEABILITY** 24](#_Toc447646849)

[**10.1 Traceability** 24](#_Toc447646850)

[**11. EVOLUTION OF THE SRS** 24](#_Toc447646851)

[**12. RATIONALE** 24](#_Toc447646852)

[**13. NOTES** 24](#_Toc447646853)

[**14. APPENDICES** 25](#_Toc447646854)

[**14.1 Schedule Tracking** 25](#_Toc447646855)

[**14.2 Defect Tracking** 26](#_Toc447646856)

# **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 explain the purpose and features of the system: what the system will do, what its system interfaces are, etc. This document is intended for the developers so they will understand how to design the application.

# **2. SCOPE**

This application will be a MOBA matchmaking system for Dota players. Users of this application will create accounts that are linked to their Dota profiles. They are able to set desired preferences of the type of players they expect 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. The user is then able to determine whether or not they would like to play with this matched player. 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.

## **2.1 Identification**

Systems Requirements Specification, SRS-001, Version 2.0

## **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 2.0 3/21/2016
* Software Project Management Plan (SPMP) 3/30/2016
* Software Analysis Specification (SAS) 4/11/2016
* Software Design Document (SDD) 4/25/2016
* Project Completion and Finished Product Fall 2016

# **3. OVERALL SYSTEM OVERVIEW**

## **3.1 Context Diagram**

Credential   
Management

Direct

Messaging

Add Friends

User

Join Party

Rate

Teammates

Public Text

Post

## **3.2 Additional Descriptive Items**

### ***3.2.1 Product Functions***

* Matchmaking service to match players with others of their preference.
* Player rating system
* Partner system for players who would like to form teams with their matches.
* Public posting ability
* Profile management

### ***3.2.2 User characteristics***

* Create a user-level account.
* Users can initiate a matchmaking queue with up to four members.
* Users are able to communicate with other users.
* Users are able to create public posts.
* Allow the user to decide on whether to play with a matched player.
* Ability to rate a player after playing a match with them.
* Ability to create partners with other users.
* Users can initiate a team matchmaking queue.

# **4. DOCUMENT OVERVIEW**

## **4.1 Description of Document**

The rest of the Software Requirements Specification (SRS) includes:

* Section 5 contains references to the Software Project Management Plan (SPMP).
* Section 6 contains Business requirements. This section is subdivided into 6 subsections for Technology, Economics, Regulatory and Legal, Market Considerations, Risks and Alternatives, and Human Resource and Training.
* Section 7 contains function specifications and definition, use case model, and other specific requirements.
* Section 8 contains the test plans with scenario testing and required simulators.
* Section 9 contains the review process for quality.
* Section 10 contains the traceability process.
* Section 11 contains the evolution of the Software Requirements Specification (SRS) throughout the life cycle.
* Section 12 contains explanations for certain implementations.
* Section 13 contains additional notes explaining certain decisions.
* Section 14 contains tables for tracking schedule and defects.

## **4.2 Organization**

The Software Requirements Specification (SRS) version 2.0 is organized into 14 sections with each section having its own subsections. Section 1, Introduction, explains the purpose the Software Requirements Specification (SRS). Section 2, Scope, which contains the identification of the document, system description, bounds, and objectives. Section 3, Overall System Overview, describes general factors that affect the product and its requirement. Section 4, Document Overview, describes the organization of the document. Section 5, Reference Document, contains reference documents to the Software Project Management Plan (SPMP). Section 6 contains Business Requirements.

Section 7, Specification Requirements, contains the function specifications and definition, use case models, etc. Section 8, Test Plan Requirements, contains future testing of implemented functions. Section 9, Qualification Provisions, describes how the SRS will be reviewed. Section 10, Requirements Traceability, describes how each requirement is traceable throughout workflows. Section 11, Evolution of the SRS, describes possible future revisions of Version 2.0. Section 12, Rationale, contains explanations for each implementation. Section 13, Notes, contains additional information and explanation for the requirements. Section 14, Appendices, contains tracking tables for schedule and defects.

# **5. REFERENCE DOCUMENTS**

* Project Proposal was submitted on February 17, 2016. The Project Proposal can be referenced through NYU Classes.
* Software Project Management Plan (SPMP) will be completed and submitted by March 30, 2016.

# **6. BUSINESS REQUIREMENTS**

## **6.1 Technology**

Compatible for Android mobile devices only. Android devices must at the minimum run Lollipop Version 5.0 or higher.

## **6.2 Economics**

Product currently has no intentions to produce profits. Product will be initially be release free of charge to all Android users. May consider selling product in the future is demand is high.

## **6.3 Regulatory and Legal**

Not required.

## **6.4 Market Considerations**

* Competition from other applications that perform similar functionalities as DotaMatch.
* Possible portability for iOS in future iterations.
* Need to cater towards Dota players.

## **6.5 Risks and Alternatives**

* Risks:
  + Lateness
  + Over-budget
  + Residual faults - not meeting user expectations.
* Alternatives:
  + Detect faults early
  + Have high quality and proper specifications and requirements.
  + Test throughout each workflow of the life cycle.
  + Documentation is performed in parallel with development and maintenance.
  + Make sure specifications and requirements meet client’s expectations.

## **6.6 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.

# **7. SPECIFIC REQUIREMENTS (DESCRIPTIVE FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS)**

## **7.1 Functional Descriptive Detailed Requirements**

### ***7.1.1 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 log in.

**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.

### ***7.1.2 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.

### ***7.1.3 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 whether their

**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.

### ***7.1.4 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.

### ***7.1.5 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.

## **7.2 Requirement Use Cases**

### ***7.2.1 Use Case Diagram***

Manage Database



Manage Server

Developer

Create Matchmaking Queue

Login

Matchmaking System

User



Add partner

Post publically

Rating System

Rate other users

### ***7.2.2 Use Case Descriptions***

## 

## **7.3 Non-Functional Descriptive Detailed Requirements**

DotaMatch will be on a NYU server with Internet capability. The speed of matchmaking will depend on NYU’s server speed and the speed of the user’s device. DotaMatch uses NYU’s database to store user information.

## 

### ***7.3.1 System Capabilities, conditions, and constraints***

* 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.
* Portability:
  + Application should be portable to iOS in future iterations.
* Constraints:
  + 2GB of Database storage
  + Internet connection speed
  + Server responsiveness

## 

### ***7.3.2 Physical Resource Requirements***

NYU Server

* NYU Database
* Development Machine

## 

### ***7.3.2.2 Computer Hardware Resources Requirements***

* + Standard consumer hardware.
  + Android mobile device

## 

### ***7.3.2.3 Computer Software Requirements***

* + Android SDK

## 

### ***7.3.2.4 Computer Communications Requirements***

* + Server needs to be able communicate with the database.
* User Interface needs to be able to communicate with the server.

### ***7.3.3 Environmental Conditions***

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.

### ***7.3.4 System Performance Characteristics***

System should be reliable. Matchmaking should provide matches that are more than 99% correct.

### ***7.3.5 Safety Requirements***

None

### ***7.3.6 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.

## 

### ***7.3.7 System Human Interfaces***

Interface will be primarily touch-based for smartphones.

* + Touch-based navigation
  + Touch based keyboard

## 

### ***7.3.8 System Maintainability***

* + Server must be up and running more than 99% of the time.
* Server must be able to connect to the database at all times.
* Faults founded must be fixed without changing specifications.
* Introducing new functions should not cause regression faults.

### ***7.3.9 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.

### ***7.3.10 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.

### ***7.3.10.1 Life Cycle Model***

Iterative and Incremental Life Cycle Mode

* Artifact is constructed piece by piece (incrementation).
* Each increment goes through multiple versions (iteration).
* Project as a whole is divided into smaller mini projects. Every mini project extends the workflows.
* There are 5 different workflows throughout the whole life cycle. The workflows are Requirements, Analysis, Design, Implementation, and Test. Testing occurs at the end of each workflow. Planning and documentation activities are performed throughout the life cycle.

### ***7.3.10.2 Policies and standards - Methods, tools, and techniques***

* Iterative and Incremental Life Cycle Model
* CASE Tools
* Use case model
* UML diagram

### ***7.3.11 Personnel-Related Requirements***

* Must be a 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.

### ***7.3.12 Training-Related Requirements***

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

### ***7.3.13 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.

### ***7.3.14 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.

### ***7.3.15 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.

### ***7.3.16 Other non-functional Requirements***

No other additional non-functional requirements.

# **8. 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.

## **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**

## **10.1 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. EVOLUTION OF THE SRS**

The Software Requirements Specification (SRS) version numbers will be incremented as needed. Version numbers will be incremented by a whole number if major changes occurs. While decimal incremental will occur if minor changes occur. Tracking tables in Section 14 will be updated throughout the lifecycle. Section 2 deadlines maybe be postponed if project is not on time. Overall system functions in Section 3 may change as some functions are no longer necessary or additional functions are required. Section 12 will have additional rational if explanation is required. Section 13 will contain additional notes if required.

Overall the Software Requirements Specification (SRS) will undergo changes in the future the as faults, deficiencies, shortcomings, inaccuracies, or changes in system environment arises.

# **12. RATIONALE**

No additional rationales.

# **13. NOTES**

No additional notes.

# **14. APPENDICES**

## **14.1 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 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Corey Chong | 11 | 11 | 0 |
| Albert Su | 11 | 12 | 1 |
| Evans Yeung | 12 | 15 | 3 |

## **14.2 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 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Corey Chong | 30 | 42 | 12 |
| Albert Su | 34 | 41 | 7 |
| Evans Yeung | 32 | 40 | 8 |