**NYU Polytechnic School of Engineering**

**Computer Science and Engineering**

**eHarbinger**

**Requirements/Analysis Documentation**

**Version 2.0**

**Document Number: RAS-002**

Project Team B3

Project Team Members:

Meghan Clark mc4677

Brian Marks bm1549

Priyam Nidhi pn613

Benson Tsai bt773

**REVIEWS AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function (Author, Reviewer, Approval)** | **Date** | **Signature** |
| Brian Marks | Author | October 10, 2015 | Signature on File |
| Meghan Clark | Author | October 10, 2015 | Signature on File |
| Priyam Nidhi | Author | October 10, 2015 | Signature on File |
| Benson Tsai | Author | October 10, 2015 | Signature on File |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision Level**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| September 29, 2015 | Version 1.0 | Initial Release |
| October 10, 2015 | Version 2.0 | Use cases, context diagrams, and class diagrams updated. Graphics were reworked to work with MSWord |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[1. Introduction 7](#_Toc432356432)

[1.1 Purpose 7](#_Toc432356433)

[2. Scope 7](#_Toc432356434)

[2.1 Identification 7](#_Toc432356435)

[2.2 Bounds 7](#_Toc432356436)

[2.3 Objectives 7](#_Toc432356437)

[2.4 System Overview 8](#_Toc432356438)

[2.5 Document Overview 9](#_Toc432356439)

[3. Reference Documents 10](#_Toc432356440)

[4. Business Requirements 11](#_Toc432356441)

[4.1 Technology 11](#_Toc432356442)

[4.2 Economics 11](#_Toc432356443)

[4.3 Regulatory and Legal 11](#_Toc432356444)

[4.4 Market Considerations 11](#_Toc432356445)

[4.5 Risks and Alternatives 12](#_Toc432356446)

[4.6 Human Resources and Training 12](#_Toc432356447)

[5.Context Diagram 12](#_Toc432356448)

[5.1 High level (level -0) 12](#_Toc432356449)

[6 Functional Requirements Analysis Specification 13](#_Toc432356450)

[6.1 Functional Descriptive Detailed Requirements 13](#_Toc432356451)

[6.2 System Capability Requirements 14](#_Toc432356452)

[6.2.1 Capabilities 14](#_Toc432356453)

[6.3 User Interface Requirements 17](#_Toc432356454)

[6.4 Component Architecture 18](#_Toc432356455)

[6.4.1 Component Descriptions 18](#_Toc432356456)

[6.4.2 Component Architecture Diagram 19](#_Toc432356457)

[6.5 Class Diagrams 19](#_Toc432356458)

[6.6 Class Relationship/Interaction Diagrams 22](#_Toc432356459)

[6.7 Event Section 22](#_Toc432356460)

[6.7.1 Event Dictionary 22](#_Toc432356461)

[6.7.2 Event Diagrams 24](#_Toc432356462)

[6.8 Activity/State (Scenario) Section (To be completed in design) 27](#_Toc432356463)

[6.8.1 Activity (Scenario) Diagrams 27](#_Toc432356464)

[6.8.2 Activity (Scenario) Specification 28](#_Toc432356465)

[6.9 Sequence Diagrams 28](#_Toc432356466)

[6.10 Collaboration Diagrams 29](#_Toc432356467)

[6.11 Dictionaries 29](#_Toc432356468)

[7 Non-Functional/Operational Requirements 29](#_Toc432356469)

[7.1 System External Interface Requirements 29](#_Toc432356470)

[7.2 Safety Requirements 30](#_Toc432356471)

[7.3 Security and Privacy Requirements 30](#_Toc432356472)

[7.4 System Environment Requirements 30](#_Toc432356473)

[7.5 Computer Resource Requirements 30](#_Toc432356474)

[7.5.1 Computer Hardware Requirements 30](#_Toc432356475)

[7.5.2 Computer Hardware Resource Requirements 30](#_Toc432356476)

[7.5.3 Computer Software Requirements 30](#_Toc432356477)

[7.5.4 Computer Communications Requirements 30](#_Toc432356478)

[7.6 System Quality Factors 30](#_Toc432356479)

[7.7 Design and Construction Constraints 31](#_Toc432356480)

[7.8 Personnel-Related Requirements 31](#_Toc432356481)

[7.9 Training-Related Requirements 31](#_Toc432356482)

[7.10 Logistics-Related Requirements 31](#_Toc432356483)

[7.11 Packaging Requirements 31](#_Toc432356484)

[7.12 Precedence and Criticality Requirements 31](#_Toc432356485)

[7.13 Other Requirements 31](#_Toc432356486)

[8. System Test Plan Requirements 31](#_Toc432356487)

[9. Qualification Provisions 32](#_Toc432356488)

[10. Requirements Traceability 33](#_Toc432356489)

[11. Rationale 33](#_Toc432356490)

[12. Notes 34](#_Toc432356491)

[13. Appendices 35](#_Toc432356492)

[13.1 Dictionaries 35](#_Toc432356493)

[13.2 UML Diagrams, if not included in the body of the document 40](#_Toc432356494)

[13.3 Schedule Tracking 40](#_Toc432356495)

[13.4 Defect Tracking 42](#_Toc432356496)

# 1. Introduction

## 1.1 Purpose

The purpose of this RAS is to explain in detail the features, interfaces, and constraints of the eHarbinger system. The intended audience of this document is developers and investors of the product where the product’s layout and boundaries are discussed.

# 2. Scope

## 2.1 Identification

This document is the Requirements/Analysis Specification(RAS-001, Initial Release) with no revisions at this stage.

## 2.2 Bounds

The project is aimed at communities of gamers who want to play with other like-minded gamers. Since the proposed target group is vast, we must be sure that the software we create is easy to use and applicable to many groups of people. The system should be designed to get publicity from this group. Since eHarbinger does not already exist in the market, a wide initiative needs to be taken to promote it in the targeted client group. Furthermore, minor adjustments can be made to make the eHarbinger platform available to other extended areas. The project uses a database of user information and their responses to questions; the user’s public profile will be visible to all other users, while the responses will be used internally to assist in the matchmaking process. To bring out staged delivery of the product, it will involve careful planning at both the management and technical level.

## 2.3 Objectives

The eHarbinger system aims to bring users together who have similar interests in gaming together so that the gaming experience can be more enjoyable. This product will be aimed at users who play video games but are not a part of a gaming community. The need for the product arises because there are a number of people who enjoy playing video games, but do not have other people to play with. eHarbinger attempts to match these people together to build strong communities.

The project priority is high. The project would use the agile life cycle model which will allow us to meet many changing user requirements. Since the user base for this project will be very high, we need to have a flexible life cycle model to keep up with user demands. The product would move with a user-driven functionality. The initial deliverables are divided into five parts, as listed below:

|  |  |
| --- | --- |
| Project proposal | 09/16/2015 |
| Software Project Management Plan (SPMP) | 09/23/2015 |
| Requirements/Analysis Documentation (RAS) | 09/30/2015 |
| Design Description (SDD) - Initial | 10/21/2015 |
| Design Document Final (w/Code) | 11/18/2015 |

## 2.4 System Overview

The product function is to match users with other users based on their compatibility in various video games. The product will contain a user profile with publicly available information about the users, for users that have been matched together, a chat will be available for the users to communicate, and there will be a section private to each user where he/she may answer questions. The responses to these questions will determine how the users are matched together.

1. User Characteristic: The user would be a person who enjoys playing video games and wants to play with other like-minded players. The users answer a questionnaire upon joining and information obtained from the questionnaire is used to match them with others. Users can opt to answer additional questions in order to get more reliable results.
2. Constraints: The primary constraint that the system would face arises from the human component of the product. Many people may answer questions in a way that is not entirely truthful or join the site just to cause harm to other users. We will take preventive measures against such users by implementing a rating system where users can rate other users based on their experiences. This information will help us identify the problem users and rectify the situation.
3. Assumptions and Dependencies: The majority of the product will be coded modularly. By doing so, changes can easily be made without having a negative shift in performance in other areas. Dependencies would include linking profiles with external sources, such as Facebook and Steam. If the API changes for these sites, a fix would need to be applied. Furthermore, we need to keep our list of games up to date so that users can find other users for new games.
4. Requirements Subsets: There are 2 types of requirements that need to be considered in this project:

* Core Requirements: In order for the product to meet its Acceptance requirements, all core requirements must be met and the deliverables should be executed on time. The design must follow these specific requirements.
* Non-core requirements: These are additional requirements that can be added to the system in the future revisions. Changes in the user interface based on consumer feedback would be implemented in the revisions made to the product.

## 2.5 Document Overview

The subsequent portions of this document describe the project break down:

Section 3 provides references to Proposal, SPMP, RAS, and SDD.

Section 4 discusses the business constraints, risks and opportunities of this project.

Section 6 lays out the Logical Architectural Specification with the context diagrams.

Section 7 contains the system capability requirements, component architecture, class diagrams and relationships along with event diagrams and dictionaries.

Section 7 covers the non functional and operational specifications with the various system requirements.

Section 8 has system plan requirements with scenario testing and required simulators.

Section 9 covers qualification provisions which includes review and quality processes.

Section 10 describes how every requirement is accounted for in the documentation and

system implementation.

Section 11 covers the rationale behind the project.

Section 12 provides additional notes.

Section 13 is the appendix with schedule and defect tracking.

# 3. Reference Documents

|  |  |  |  |
| --- | --- | --- | --- |
| Title | Author | Version | Date |
| Project proposal | Project Team | v1.1 | 09/16/2015 |
| Software Project Management Plan (SPMP) | Project Team | v1.1 | 09/23/2015 |
| Requirements/Analysis Documentation (RAS) | Project Team | v1.0 | 09/30/2015 |
| Design Description (SDD) - Initial | Project Team | v1.1 | 10/21/2015 |
| Design Document Final (with Code) | Project Team | v1.2 | 11/18/2015 |

# 4. Business Requirements

## 4.1 Technology

The technology in the project is related to the database of information for the user profiles and how it is presented on the front-end; this is seen as the critical value driver and we would place its benefit to operational business activities at the center of our design work and implementation. The IT applications would cover execution and effectiveness of the product.

For this stage, the system is restricted to only gamers and we will be enabling the matching capabilities only in the environment of games. For our system environment, the user profiles and the chat are the two variables that the platform would focus on.

## 4.2 Economics

A stable business climate is one of the key economic drivers. The project focuses on gamers who are in need of a group to further enhance the experience. We do not have an exact number of the market size that we are entering but we do know that our product would be able to vastly improve the bottom-line of the gamers as they will have access to resources. Once the project satisfies the material needs of gamers, we can even expand to new areas such as extending the product to gardening matches, tutoring matches and so on.

## 4.3 Regulatory and Legal

The team would follow copyright guidelines and there design would be unique from other front end web layouts. Except that, there are no other legal requirements at this stage.

## 4.4 Market Considerations

The project aims to target gamers who are looking to find a group with specified skills and to enjoy a new, well-defined and scalable model that helps them do so without hassle. The market that we are aiming at is teenagers or college students who enjoy playing games. This is an enormous market which gives the product the required exposure and popularity after its launch.

## 4.5 Risks and Alternatives

The system needs to reduce the vulnerabilities that it is exposed to. A few risks could be:

* Remote access to servers is not monitored. This would lead to unauthorized access to system data.
* Key Person Dependency: Loss of this key person could result in a system downtime.
* Loss of critical documentation or software would result in data corruption or loss.

We can deal with this by having a democratic team where each team member is aware of the software. Although there would be a key person responsible for server, the team member leading Database Development can step in case of any server-related issues or if the key person is unavailable. The servers need to be optimally monitored for requests; adding a firewall can help deal with this. User profiles and the forum need to be monitored by the UX team member so that she can upgrade and facilitate any new improvements.

## 4.6 Human Resources and Training

The resources that the project needs is a team that can help execute the 3-tier architecture for the platform. It can be broken down as follows (the skillset required is mentioned in parentheses):

* Team for Database Development and Management (Knowledge of PostgreSQL)
* Team for Server and User-Interface Development (Knowledge of C++, Python, PHP, Javascript)
* Marketing and Business Operations (Skills in marketing and sales operations)

# 5.Context Diagram

## 5.1 High level (level -0)

eHarbinger

Gaming companies

Creates Popularity for web app

matches them with the right team/opponent

Can post ads on the forum in later stages

Can connect with other gamers on the forum or private chat.

Customer

(Gamers)

Another

Customer

# 6 Functional Requirements Analysis Specification

## 6.1 Functional Descriptive Detailed Requirements

**1.0** A website to hold all components of project together.

**1.1** The website will be able to create a profile for each user based on platforms they play on, types of games they are interested in, and how advanced a player they are in certain games.

**1.2** The website must be able to connect users with similar preferences together through an algorithm based on similar interests and skill level.

**1.3** The website must be able to allow users to create posts on a forum when looking for other users to play with.

**1.4** The website will allow for users to give feedback to other users on how they play through a thumbs up and thumbs down system.

**2.0** Database to hold all users’ information.

**2.1** There will be a database of the users’ names, usernames for devices to play on, emails, preferred gaming systems, games interested in, level of gaming ability, and any feedback given to the user.

**2.2** The database will also have the users’ usernames and password specifically made to the website, but will not allow for the passwords to be made public.

**3.0** A server written in C or C++.

**3.1** The server will allow the website to remain online and connect the website to the databases of information when needed.

## 6.2 System Capability Requirements

### 6.2.1 Capabilities

**Case Diagram**

Look for Partners to Play With

Give Feedback to User

Create Profile

Maintain Website

User 1

User 2

Maintainer

**Use Case Descriptions**

Use Case 1.0

|  |
| --- |
| The user wishes to create a profile on the website. |

|  |  |
| --- | --- |
| Description | The user will go to the website and create a profile. |
| Preconditions | The user has internet connection. |
| Flows: Basic or Normal Flows | The user will go to our website’s URL and be prompted to create a profile. They will fill out all necessary information which will be saved in the database before giving the user access to the rest of the website. |
| Flows: Alternate Flows | None |
| Postconditions | User will have a profile. |
| Special Requirements | User will need to answer various questions based on information on gaming preferences and contact information. |
| Extension Points | None |

Use Case 2.0

|  |
| --- |
| Users wish to search for another user to play with. |

|  |  |
| --- | --- |
| Description | The user will go to the website and search for another person to play with. |
| Preconditions | The user has internet connection. |
| Flows: Basic or Normal Flows | The user logs into their profile. They will go on the explore page and see any other potential players to play with based on their preferences. Then the user chooses who they wish to play with a sends them a message to chat. |
| Flows: Alternate Flows | None |
| Postconditions | User 1 and user 2 are able to talk to each other and start a game with each other. |
| Special Requirements | None |
| Extension Points | None |

Use Case 3.0

|  |
| --- |
| User gives feedback about other user on their profile. |

|  |  |
| --- | --- |
| Description | The user will go to the other user’s profile and give the feedback they want through a thumbs up or thumbs down. |
| Preconditions | The user has internet connection. |
| Flows: Basic or Normal Flows | The user will be logged into their page and then search for the player they played with by the other player’s username. Then they will go to that profile to click a thumbs up for positive feedback or thumbs down for negative feedback. That feedback will be saved in the database and the player’s profile will be updated with the feedback. |
| Flows: Alternate Flows | None |
| Postconditions | The second user is given positive or negative feedback based on how they played together. |
| Special Requirements | None |
| Extension Points | None |

## 6.3 User Interface Requirements

The life cycle being used is the agile. This means that after development of the project there will be constant planning, designing, implementing, and then testing of new users’ demands from the product. The bulk of testing will be done through the browser of HTML, CSS, Javascript, and PHP, to ensure that the website will be easily used by the public as well as able to create any action needed by the user within the database.

## 6.4 Component Architecture

### 6.4.1 Component Descriptions

There are multiple components that are found in the system of the product to make it function properly. The three main components needed to create the product are, the user interface, an application server, and databases.

The user interface will be coded in a mix of HTML, CSS, and Javascript. The user interface’s main purpose is to be simple and easy for the user to be able to look through the webpage and find what they need. The user interface must be able to get information from the databases about any information the user allows others to see as well as information made private to the user. The user interface must get this information through the databases but cannot directly do so.

To solve this issue, the application server will have system buses that go to the  
user interface and databases. The databases will contain the information of both hardware components and vendors and have its own system bus to communication information to the application server. Inside the application server, there will be classes for each person or object that will be trying to interact with the user interface, or the product in general, as well as methods for each action needed. The methods and classes that are needed in the application server for the system to work can be found in the diagram in section 6.4.2 of this document.

### 6.4.2 Component Architecture Diagram

User Interface

Databases

Application Server

Directory

Create Profile

Update Profile

Delete Profile

Customer

View Profiles

Chat with Users

Give Feedback

User

## 6.5 Class Diagrams

|  |  |
| --- | --- |
| Name: | Directory |
| Attributes: | 1. userLoc - Pointer Variable to User information in Database |
| Methods: | 1. getUserName - gets the user’s real name and username 2. getUserGames - gets user’s game preferences 3. getUserSkill - gets the user’s skill level in gaming 4. chatWithUser- alerts user that another user wishes to chat 5. updateUser - updates any information needed in the database from the user class 6. deleteUser - deletes user profile 7. createUser - creates user profile |

|  |  |
| --- | --- |
| Name: | User |
| Attributes: | 1. fullName - user’s first and last name 2. userName- user’s public username for contacting 3. password- User’s password for profile 4. gamingPrefrence- platforms/games user prefers to play 5. gamingSkills - skill level of user 6. feedbackLevel - number of thumbs up and thumbs down user has received |
| Methods: | 1. getFullName - gets user’s full name 2. getUserName - gets user’s username 3. getUserGames - gets user’s game preferences 4. getUserSkill - gets the user’s skill level in gaming 5. updateFullName - changes full name of user 6. updateUserName - changes username 7. updatePassword - changes password 8. updatePreferences - change game preferences 9. updateSkills - change skill level 10. updateFeedback - updates feedback of user 11. setFullName - sets full name of user 12. setEmail - sets email of user 13. setUserName - sets username 14. setPassword - sets password 15. setPreferences - sets preferences of user 16. setSkills - sets skill level of user |

|  |  |
| --- | --- |
| Name: | Customer |
| Attributes: | 1. userLoc - Pointer Variable to user profile |
| Methods: | 1. createProfile - allows customer to create Profile 2. viewProfile - allows customer to view other user’s profiles 3. giveFeedback - allows customer to give feedback to other users 4. chatUser - allows customer to chat with other user 5. updateProfile - updates profile information 6. deleteProfile - deletes user’s profile |

## 6.6 Class Relationship/Interaction Diagrams

Directory

User

Customers

Keep list of users to navigate through

Customer is able to view and chat with other users

Customer is linked through other users from directory.

1

m

m

m

m

1

## 6.7 Event Section

### 6.7.1 Event Dictionary

1. **A new customer goes onto the website.**

Once a customer enters the website they will be prompt to create a profile to continue. Then the customer must enter their full name, username they choose, a strong password, platforms and games they prefer, and a general skill level they have in games. Once this is completed their profile will be added to the database as well as their feedback level set to zero.

1. **A returning customer goes onto the website.**

A returning customer who is not logged in will be prompted to enter their username and password. If they were already logged in and haven’t logged out, they will be brought to their profile.

1. **A customer clicks on another user’s profile.**

As the customer sees other users, they will most likely find a user they might want to play with. Once the customer clicks on another user’s profile, that user’s information will be displayed for the customer to look at.

1. **A customer clicks to chat with another user.**

If the customer decides that they would want to play a game with the user whose profile they were looking at, they will have the option to chat with them. Once this happens the user will be given a notification that the customer wishes to chat. As soon as the user agrees to chat, a window will pop up for messages between the customer and user to appear.

1. **A customer clicks to give feedback to another user.**

Once a customer connects with another user, they have to option to give a thumbs up or thumbs down to the other user. This is done by clicking on the feedback level on the user’s profile by the customer to give the feedback they wish. Then this is updated on that user’s profile as well as that user’s feedback level in the directory.

1. **A customer clicks to change something on their profile.**

If a customer ever needs to change information on their profile, they will click on the edit button on their own profile. After doing this a form will appear to allow the customer to change what they need to. Once the customer is finished they will click to save changes and these changes will be updates in the directory and on the user’s profile.

1. **A customer clicks to delete their profile.**

If the customer feels that they no longer feel that they need to use the website, they will have the option to delete their profile. Once they choose to delete this profile, they will be given an alert to ensure that the customer wants to delete their profile. Once the customer confirms this, the user will be given a “hope to see you again soon” message, and the profile will be deleted from the directory.

### 6.7.2 Event Diagrams

1. **A new customer visits the website.**

Visits website

Stores data

Posts information

Customer

User Interface

Customer

Creates profile

Directory

New Profile Added

Database

1. **A returning customer goes onto the website.**

Retrieves profile

Visits website

Enters credentials

Queries data

Returns data

Retrieves profile

Customer

User Interface

Customer

Display Profile

Directory

Get Profile Information

Database

1. **A customer clicks on another user’s profile.**

Retrieves profile

Returns data

Queries data

Retrieves profile

Click on name

Customer

User Interface

Customer

Views User Profile

Directory

Get Profile Information

Database

1. **A customer clicks to chat with another user.**

Retrieves user

Stores data

Posts message

Enters message

Customer

User Interface

Customer

Chats

Directory

Alerts User of Chat

Database

User

1. **A customer clicks to give feedback to another user.**

Retrieves user

Stores data

Rates user

Posts information

Customer

User Interface

Customer

Give User Feedback

Directory

Changes User Feedback

Database

User

1. **A customer clicks to change something on their profile.**

Posts information

Stores data

Retrieves profile

Submits new information

Customer

User Interface

Customer

Change Info on Profile

Directory

Changes Profile Info

Database

1. **A customer clicks to delete their profile.**

Updates file

Retrieves user

Posts request

Requests deletion

Customer

User Interface

Customer

Delete Profile

Directory

Delete all information of customer profile

Database

## 6.8 Activity/State (Scenario) Section (To be completed in design)

### 6.8.1 Activity (Scenario) Diagrams

Will be completed in Design.

### 6.8.2 Activity (Scenario) Specification

Will be completed in Design.

## 6.9 Sequence Diagrams

Customer

User

Directory

Database

Creates Profile

Saved to Database

Display Profile

View other profile

Display Other profile

Get Profile

Chat with other user

Get Profile

Send notification of chat

Connect with customer

Give Feedback

Update User Profile

Saved to Database

Display Feedback

Change Profile Information

Saved to Database

Display Changes

Delete Profile

Delete from Database

Update Directory

Display Goodbye

Update Directory

Update Directory

## 6.10 Collaboration Diagrams

Customer

User

Directory

getUserName(); getUserGames(); getUserSkill(); updateUser(); deleteUser(); createUser()

chatWithUser();

getUserName(); getUserGames(); getUserSkill(); updateUser(); deleteUser(); createUser()

createProfile();

viewProfile();

updateProfile();

deleteProfile();

giveFeedback()

chatUser();

giverFeedback();

chatUser();

getFullName();

getUserName();

getUserSkill();

updateFullName();

updateUserName();

updatePassword();

updatePrederences()

updateSkills();

updateFeedback();

setEmail();

setFullName();

setUserName();

setPassword();

setPreferences();

setSkills();

## 6.11 Dictionaries

The dictionaries include all classes, methods, attributes, messages, events and their association.

Section 13.1 contains a detailed breakdown of the classes and attributes required at this stage of eHarbinger analysis.

# 7 Non-Functional/Operational Requirements

## 7.1 System External Interface Requirements

The system should allow users to navigate throughout the website without difficulty to utilize the main functions of the product: user profile, questionnaire, forum and chat. There should be a standard theme that is applied to all interfaces and pages that appear in the product, details of which will be determined by the UX design member. At this stage of the project, communication should all be made through web browser.

## 7.2 Safety Requirements

Database containing username and password should be encrypted to provide protection against identity theft. A firewall should be employed on the server to protect against external attack.

## 7.3 Security and Privacy Requirements

Content of the system should only be visible to registered members, similar to Facebook. There should be a fine grained visibility settings where all user data should be accessible to approved audience only.

## 7.4 System Environment Requirements

None.

## 7.5 Computer Resource Requirements

### 7.5.1 Computer Hardware Requirements

For a client-side web application, eHarbinger does not require a tremendous amount of resource to run; therefore, any computer that is capable of handling standard web browsing should be able to provide sufficient computing power. It is recommended, however, that the user computer meet the minimum requirements for the games they intend to play.

### 7.5.2 Computer Hardware Resource Requirements

For client, any hardware that is capable of browsing the internet is sufficient. For server-side, the hardware resources should allow capacity typical of a small gaming platform, with flexibility for expansion if required.

### 7.5.3 Computer Software Requirements

Software environment should allow compatibility with programming and database platforms used in the project.

### 7.5.4 Computer Communications Requirements

Internet is the primary method of communication and must be available for use of the product.

## 7.6 System Quality Factors

Quality of the system should be measured in speed, space and user friendliness.

## 7.7 Design and Construction Constraints

None.

## 7.8 Personnel-Related Requirements

Each team member should be updated on what everyone on the team is doing for the project. Each member must also be able to review the other member’s work and have basic knowledge of how each team member is doing their work.

## 7.9 Training-Related Requirements

No training is required; however, the team should familiarize themselves with all parts of the program to allow flexibility in work distribution in the development of the project.

## 7.10 Logistics-Related Requirements

None.

## 7.11 Packaging Requirements

None.

## 7.12 Precedence and Criticality Requirements

None.

## 7.13 Other Requirements

None.

# 8. System Test Plan Requirements

Compatibility information of hardware is crucial to the functionality and dependability of this system, and therefore accuracy of this information must be rigorously tested. In the initial stage, such information will be manually checked for accuracy. Detailed plan will be laid out pending further development of the specification of system.

Furthermore, there will be several walkthroughs done as well as team meetings in order to find as many errors as possible. Initial testing will be done using a compiler and fixing any syntax errors that occur. Secondary testing will be done by using many different inputs to insure that valid inputs are allowed and produce the correct results and invalid inputs are dealt with accordingly. After the testing, the following items need to be cross checked:

* System test report
* System test output data
* Problem reports

After designing the system tests, it needs to be ensured that all environmental needs are satisfied to perform these. After this, the problem report needs to be generated with the Pass/Fail reports of the system processes. Once the environmental needs and input specifications are met, the test cases can be successfully executed.

# 9. Qualification Provisions

A software quality assurance plan has been made to monitor the quality of work processes and their products. There would be inspections initiated by the project manager in between the milestones to discuss the progress of the project and to detect faults in the work done so far. Inspecting the product's code would be a major way of detecting faults. This subsection describes the Quality Assurance process for work products to ensure that the quality is consistent for the entire project.

In order to ensure consistency, there will be testing, demonstration, analysis, and inspection. Testing will be done constantly throughout the software project development process and demonstration will be done on an as-needed basis. Analysis and inspection will be left to the end of each milestone. There will also be periodic reviews to make sure everything is on track to complete. At the end of each milestone, there will also be an audit an assessment done to verify the quality of the work of the software team.

The quality review sessions will be done in meetings where all of the software development team are in attendance. Reviews will be done to make sure documentation is correct and the code closely follows the documentation. The audit sessions will be appended at the end of the review sessions, so that all team members are present.

There would also be informal walkthroughs initiated by a team member in every phase to resolve any queries that might arise as the project progresses. In conducting these quality-adherence tests, the team would check that:

* Artifacts meet the set quality criteria
* Requirements should be traceable and verifiable
* Any defects in the earlier stages of the project should be immediately identified and removed.

# 10. Requirements Traceability

This section is to provide information about the reasoning for certain requirements in relation to the artifacts the project contains. Included is a description of the requirements and related dependencies to be able to track changes to the requirements.

1. Since this project will be focused around an online market, all operating systems will be supported as long as they have a current web browser.
2. For the system to provide up to date information on the newest games and user matches, there must be a server and database with information being maintained on a daily basis. To accomplish this task, there must be clear defined interfaces between the server and database.

# 11. Rationale

The analysis of this document was done using case diagrams and the constraints that the project would face at this point. As laid out in the project proposal, many gamers wish to play with other like-minded gamers and our product fills that particular niche. A user can answer questions such as how competitive they are, how competitive they want their teammates to be, as well as their skill level and the skill level they expect of their teammates. These questions, along with other similar questions, helps us pair users up with other users that will get along together. The product will be created through object-oriented Programming. In order to have all the workflows for the product, each object will need to be put in a class. For example, to allow a gamer to look through our database of user profiles, we must have an object in the code that is able to perform his actions. These methods will be based on the context diagram and will change if either the requirements are changed from the client or the team discovers an addition is required.

Many times, people deciding to play together randomly doesn’t go very well. There is a lot of trash talking and there aren’t many expectations of each other. eHarbinger gives the users a framework to communicate and collaborate and be able to meet not only gaming buddies, but possibly life-long friends based on their compatibility.

We will update our list of games daily in order to not exclude any new games and we will have users continue to answer more and more questions to keep their matches as relevant as possible. As mentioned above, the more questions a user asks, the easier it is for us to match users together. We will implement an easy to use interface that everyone can figure out upon a glance. This is important since our product is a user-centered product.

After thorough analysis, as well as feedback from potential customers, we believe there exists great demand for such a system, and the need has not yet been fulfilled. We are confident this project is feasible and will be well-received by the market when it comes to fruition.

# 12. Notes

None in this release.

# 13. Appendices

## 13.1 Dictionaries

Dictionaries for each class, method, and attribute from Section 6.11 are a part of this document.

**Classes**:

|  |  |  |  |
| --- | --- | --- | --- |
| Directory | A collection of user profiles to access by other users. | userLoc; | getUserName();  getUserGames();  getUserSkill();  chatWithUser();  updateUser();  deleteUser();  createUser(); |

|  |  |  |  |
| --- | --- | --- | --- |
| User | Controls actions of users who aren’t the current user on the device. | fullName;  userName;  password;  gamingPreferences;  gamingSkills;  feedbackLevel; | getFullName();  getUserName();  getUserSkill();  updateFullName();  updateUserName();  updatePassword();  updatePreferences();  updateSkills();  updateFeedback();  setFullName();  setEmail();  setUserName();  setPassword();  setPreferences();  setSkills(); |

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | The holding place for the customer’s information while they navigate through the site. | userLoc; | createProfile();  viewProfile();  giveFeedback();  chatUser();  updateProfile();  deleteProfile(); |

**Methods**:

|  |  |  |  |
| --- | --- | --- | --- |
| getUserName | Gets the username the user has created for account. | **email**(needs email to connect to username) | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| getUserGames | Gets the user’s game preferences and platforms | **userName**(user that needs game preferences to display) | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| getUserSkill | Gets the user’s skill level | **userName**(user that needs game skill displayed) | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| chatWithUser | Alerts user a customer wishes to chat with them | **user.userName**(username of who to alert)  **customer.userName**(username of customer who wishes to chat | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| updateUser | Updates information customer wishes to update | **userName**(user that wants to update information) | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| deleteUser | Deletes user from database | **userName**(user that wishes to delete profile) | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| createUser | Creates new profile of user | **email,userName, password, fullName, gamePreferences, skills** | Directory |

|  |  |  |  |
| --- | --- | --- | --- |
| getFullName | Displays full name. | **userName**(user who needs full name displayed) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| getUserName | Displays username. | **email**(user who needs username displayed( | User |

|  |  |  |  |
| --- | --- | --- | --- |
| getUserSkill | Displays skill level | **userName**(user who needs skill level displayed) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| updateFullName | Changes first and last name | **userName**(user who needs to change name) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| updateUserName | Changes username | **email**(user who needs to change username) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| updatePassword | Changes password | **userName**(user who needs to change password) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| updatePreferences | Changes game and platform preferences | **userName**(user who needs to change preferences) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| updateSkills | Changes skill level | **userName**(user who needs to change skill level) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| updateFeedback | Changes feedback level | **userName**(user who needs to change feedback) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| setFullName | Sets first and last name | **email**(user who needs to set name) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| setUserName | Sets username of user | **email**(user who needs to set username) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| setPassword | Sets password | **email**(user who needs to set password) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| setPreferences | Sets game and platform preferences | **email**(user who needs to set preferences) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| setSkills | Sets skill level | **email**(user who needs to set skill level) | User |

|  |  |  |  |
| --- | --- | --- | --- |
| createProfile | Creates profile through User class methods | **email**(user who needs to create profile) | Customer |

|  |  |  |  |
| --- | --- | --- | --- |
| viewProfile | Displays profile information through user and directory class | **userName**(user who needs to display information) | Customer |

|  |  |  |  |
| --- | --- | --- | --- |
| giveFeedback | Changes feedback level of another user | **userName**(user whose feedback will be changed)  **feedbackLevelChange**(whether the feedback will go up or down) | Customer |

|  |  |  |  |
| --- | --- | --- | --- |
| chatUser | Creates chat with another user | **userName**(wanted user to chat with) | Customer |

|  |  |  |  |
| --- | --- | --- | --- |
| updateProfile | Updates profile through User class methods | **userName**(user who needs to change profile) | Customer |

|  |  |  |  |
| --- | --- | --- | --- |
| deleteProfile | Deletes profile | **userName**(user who needs to delete profile) | Customer |

**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| userLoc | Pointer Variable to current user’s information in directory | Simple | Directory  Customer |

|  |  |  |  |
| --- | --- | --- | --- |
| fullName | Pointer Variable to current user’s full name | Simple | User |

|  |  |  |  |
| --- | --- | --- | --- |
| userName | Pointer Variable to current user’s username | Simple | User |

|  |  |  |  |
| --- | --- | --- | --- |
| password | Pointer Variable to current user’s password | Simple | User |

|  |  |  |  |
| --- | --- | --- | --- |
| gamingPreferences | Array of games and platforms current user prefers | Complex | User |

|  |  |  |  |
| --- | --- | --- | --- |
| gamingSkills | Array of Skills in different games | Complex | User |

|  |  |  |  |
| --- | --- | --- | --- |
| feedbackLevel | Level of positive or negative feedback | Simple | User |

## 13.2 UML Diagrams, if not included in the body of the document

UML diagrams are created through use cases in Section 6.2.1 of this document.

## 13.3 Schedule Tracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated  (Hours) | Actual | Difference |
| SPMP | Brian | 5 | 4 | 1 |
| SPMP | Priyam | 2 | 4 | 2 |
| SPMP | Meghan | 3 | 2 | 1 |
| SPMP | Benson | 3 | 4 | 1 |
| SPMP | Team | 13 | 14 | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated  (Hours) | Actual | Difference |
| RAS | Brian | 3 | 3 | 0 |
| RAS | Priyam | 3 | 3 | 0 |
| RAS | Meghan | 5 | 5 | 0 |
| RAS | Benson | 3 | 3 | 0 |
| RAS | Team | 14 | 14 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated  (Hours) | Actual | Difference |
| SDD (INITIAL) | Brian | 3 | N/A | N/A |
| SDD (INITIAL) | Priyam | 3 | N/A | N/A |
| SDD (INITIAL) | Meghan | 3 | N/A | N/A |
| SDD (INITIAL) | Benson | 3 | N/A | N/A |
| SDD (INITIAL) | Team | 12 | N/A | N/A |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated  (Hours) | Actual | Difference |
| SDD (FINAL) | Brian | 2 | N/A | N/A |
| SDD (FINAL) | Priyam | 2 | N/A | N/A |
| SDD (FINAL) | Meghan | 2 | N/A | N/A |
| SDD (FINAL) | Benson | 2 | N/A | N/A |
| SDD (FINAL) | Team | 2 | N/A | N/A |

**Cumulative\***

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated  (Hours) | Actual | Difference |
| Brian | 8 | 7 | 1 |
| Priyam | 5 | 7 | 2 |
| Meghan | 8 | 7 | 1 |
| Benson | 6 | 7 | 1 |
| Team | 26 | 27 | 1 |

**\***Excludes SDD

## 13.4 Defect Tracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated | Actual | Difference |
| SPMP | Brian | 3 | 3 | 0 |
| SPMP | Priyam | 4 | 4 | 0 |
| SPMP | Meghan | 7 | 7 | 0 |
| SPMP | Benson | 5 | 5 | 0 |
| SPMP | Team | 19 | 19 | 0 |

## 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated | Actual | Difference |
| RAS | Brian | 2 | 2 | 0 |
| RAS | Priyam | 4 | 4 | 0 |
| RAS | Meghan | 6 | 6 | 0 |
| RAS | Benson | 4 | 4 | 0 |
| RAS | Team | 16 | 16 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who (individual  or Team) | Estimated | Actual | Difference |
| SDD (INITIAL) | Brian | 3 | N/A | N/A |
| SDD (INITIAL) | Priyam | 2 | N/A | N/A |
| SDD (INITIAL) | Meghan | 5 | N/A | N/A |
| SDD (INITIAL) | Benson | 3 | N/A | N/A |
| SDD (INITIAL) | Team | 13 | N/A | N/A |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact 1or  Deliverable | Who (individual  or Team) | Estimated | Actual | Difference |
| SDD (FINAL) | Brian | 5 | N/A | N/A |
| SDD (FINAL) | Priyam | 3 | N/A | N/A |
| SDD (FINAL) | Meghan | 5 | N/A | N/A |
| SDD (FINAL) | Benson | 2 | N/A | N/A |
| SDD (FINAL) | Team | 15 | N/A | N/A |

**Cumulative\***

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual or Team) | Estimated | Actual | Difference |
| Brian | 5 | 5 | 0 |
| Priyam | 8 | 8 | 0 |
| Meghan | 13 | 13 | 0 |
| Benson | 19 | 19 | 0 |

**\***Excludes SDD