

SE 3XA3: Software Requirements Specification

Frets On Fire

Team #16, Rhythm Master

Almen Ng, nga18

David Yao, yaod9

Veerash Palanichamy, palanicv

April 12, 2021

Contents

1	Project Drivers	1
1.1	The Purpose of the Project	1
1.2	The Stakeholders	1
1.2.1	The Client	1
1.2.2	The Customers	1
1.2.3	Other Stakeholders	1
2	Project Constraints	2
2.1	Mandated Constraints	2
2.1.1	Solution Design Constraints	2
2.1.2	Implementation Environment of the Current System	2
2.1.3	Partner or Collaborative Applications	2
2.1.4	Off-the-Shelf Software	2
2.1.5	Anticipated Workplace Environment	2
2.1.6	Schedule Constraints	2
2.1.7	Budget Constraints	2
2.1.8	Enterprise Constraints	2
2.2	Naming Conventions and Terminology	3
2.3	Relevant Facts and Assumptions	3
2.3.1	Facts	3
2.3.2	Assumptions	3
3	Functional Requirements	5
3.1	The Scope of the Work and the Product	5
3.1.1	The Context of the Work	5
3.1.2	Work Partitioning	7
3.1.3	Individual Product Use Cases	8
3.2	Functional Requirements	8
4	Non-functional Requirements	10
4.1	Look and Feel Requirements	10
4.1.1	Appearance Requirements	10
4.1.2	Style Requirements	10
4.2	Usability and Humanity Requirements	10
4.2.1	Ease of Use Requirements	10
4.2.2	Personalization and Internationalization Requirements	10
4.2.3	Learning Requirements	10
4.2.4	Understandability and Politeness Requirements	11
4.2.5	Accessibility Requirements	11
4.3	Performance Requirements	11
4.3.1	Speed and Latency Requirements	11
4.3.2	Safety-Critical Requirements	11
4.3.3	Precision or Accuracy Requirements	11

4.3.4	Reliability and Availability Requirements	11
4.3.5	Robustness or Fault-Tolerance Requirements	11
4.3.6	Capacity Requirements	12
4.3.7	Scalability or Extensibility Requirements	12
4.3.8	Longevity Requirements	12
4.4	Operational and Environmental Requirements	12
4.4.1	Expected Physical Environment	12
4.5	Requirements for Interfacing with Adjacent Systems	12
4.5.1	Productization Requirements	12
4.6	Release Requirements	12
4.7	Maintainability and Support Requirements	13
4.7.1	Maintenance Requirements	13
4.7.2	Supportability Requirements	13
4.7.3	Adaptability Requirements	13
4.8	Security Requirements	13
4.8.1	Access Requirements	13
4.8.2	Integrity Requirements	13
4.8.3	Privacy Requirements	13
4.8.4	Audit Requirements	14
4.8.5	Immunity Requirements	14
4.9	Cultural and Political Requirements	14
4.9.1	Cultural Requirements	14
4.9.2	Political Requirements	14
4.10	Legal Requirements	14
4.10.1	Compliance Requirements	14
4.10.2	Standards Requirements	14
4.11	Health and Safety Requirements	14
5	Project Issues	14
5.1	Open Issues	14
5.2	Off-the-Shelf Solutions	15
5.3	New Problems	15
5.4	Tasks	15
5.5	Migration to the New Product	15
5.6	Risks	15
5.7	Costs	15
5.8	User Documentation and Training	15
5.8.1	Documentation	15
5.8.2	Training	16
5.9	Waiting Room	16
5.10	Ideas for Solutions	16
6	Appendix	18
6.1	Symbolic Parameters	18

List of Tables

1	Revision History	iv
2	Table of Naming Conventions and Terminology	3
3	Work Partitioning Events	7
4	Work Partitioning Summaries	7

List of Figures

1	OLD Context diagram for Rhythm Master	5
2	NEW Context diagram for Rhythm Master	6
3	Use case diagram that displays the main functionalities of the application. .	8

Table 1: Revision History

Date	Version		Notes
February 2, 2021	0.1		Initial Document
February 8, 2021	0.2		Finished Project Drivers
February 10, 2021	0.3		Finished Non-Functional Requirements Section
February 12, 2021	0.5		Added Functional Requirements Section and respective diagrams
April 11, 2021	1.0		Fixed spelling errors and grammar mistakes as well as added fit criterion to Non-Functional Requirements.

This document describes the requirements for the **Frets on Fire** project created by Unreal Voodoo. The template for the **Software Requirements Specification (SRS)** is a subset of the Volere[1].

1 Project Drivers

1.1 The Purpose of the Project

The purpose of this project is to recreate the main functionality of the **video game, Frets on Fire**, by following the software development process midst providing proper detailed documentation. Additionally, we will be programming the **game** in a modern language, C#, improving on the outdated graphics, as well as providing meaningful test cases using Unity Test Framework.

1.2 The Stakeholders

1.2.1 The Client

The clients of this **project** is the instructor of SFWRENG 3XA3, Dr. Ashgar Bokhari, and the teaching assistants (TAs) of the course, Mohammed Mirajkar and ~~Maryam~~ Maryam Hosseinkord. As the clients, they will provide instructions on what deliverables need to be completed, offer assistance wherever possible, and evaluate the degree to which the **project** meets the requirements outlined in the **SRS**.

1.2.2 The Customers

The customers of this **project** are any individuals who are interested in playing **Frets on Fire** or an open source game similar to **Guitar Hero** on **PC**. The **project** does not target a specific demographic and is available for the general public who have the hardware and software requirements to download and play.

1.2.3 Other Stakeholders

Group 16 members, are considered stakeholders of the **project** as their skills are necessary in the development of the **project**, including responsibilities such as implementing, testing and documenting the **project**, and they are interested in the success of the **project**. In addition, The developers of the **Frets on Fire**, Unreal Voodoo, and Github Users who have forked the **Frets on Fire** repository are also stakeholders as they wish to seek improvements of the **game** and, like Group 16 members, are interest in the success of it.

2 Project Constraints

2.1 Mandated Constraints

2.1.1 Solution Design Constraints

Description: The **game** must operate on any machine running on Windows 7 or newer, macOS Sierra 10.12 or newer, or Linux Ubuntu 16.04 or newer.

Rationale: The potential users of the **game** will need to have the listed operating systems in order to run the project smoothly without any additional installations/configurations.

Fit Criterion: The **game** will be made to run on Windows 7 or newer, macOS Sierra 10.12 or newer, or Linux Ubuntu 16.04 or newer.

2.1.2 Implementation Environment of the Current System

N/A

2.1.3 Partner or Collaborative Applications

N/A

2.1.4 Off-the-Shelf Software

N/A

2.1.5 Anticipated Workplace Environment

N/A

2.1.6 Schedule Constraints

Description: The **project** must follow the project schedule shown in the **Tasks section**

Rationale: The **project** needs to follow a predefined plan in order to ensure the completion of the deliverables by their respective due dates and the **project** by the end of the course.

Fit Criterion: The **project** will be completed with all deliverable submitted on time by April 16, 2021.

2.1.7 Budget Constraints

N/A

2.1.8 Enterprise Constraints

N/A

2.2 Naming Conventions and Terminology

Table 2: Table of Naming Conventions and Terminology

Term	Definition
C Sharp(C#)	The programming language used in this project.
Clone Hero	A Guitar Hero clone made for personal computers, rather than game consoles.
Fret Board	A vertical musical staff upon which notes will be displayed.
Frets on Fire	An open source Guitar Hero clone.
Game/ Project/ Rhythm Master	The game that will be made by Group 16.
Guitar Hero	A rhythm game where users simulate playing a guitar to a music track of their choice.
Note	An indicator for a button for the player to press.
Player/ User	The individual playing the game .
PC	A personal computer.
Python	The programming language used in Frets on Fire .
Score	A numerical value quantifying the player's performance in their last game.
SRS	Acronym for Software Requirements Specification; A document that describes what the system will do and the expected performance
System	The software of the game
Game track	The game track is where the gameplay happens. It consists of music track where the user interacts to score points.
Pause menu	The menu the user can open during a game track

2.3 Relevant Facts and Assumptions

2.3.1 Facts

- The original repository contains approximately 8000 lines of Python 2 code.

2.3.2 Assumptions

- **Users** have the necessary peripherals, a mouse and keyboard, to play the **game**.

- **Users** have an elementary proficiency in English.
- **Users** know how to operate a **PC** .
- **Users** ~~the~~ **game** know of the game, **Guitar Hero**.
- **Users** have the visual and physical capabilities to play the **game**.

3 Functional Requirements

3.1 The Scope of the Work and the Product

3.1.1 The Context of the Work

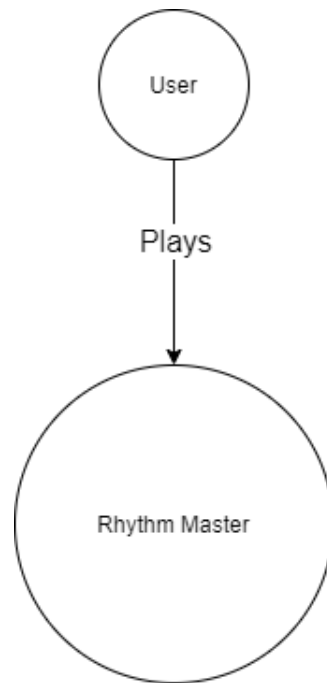


Figure 1: **OLD** Context diagram for Rhythm Master

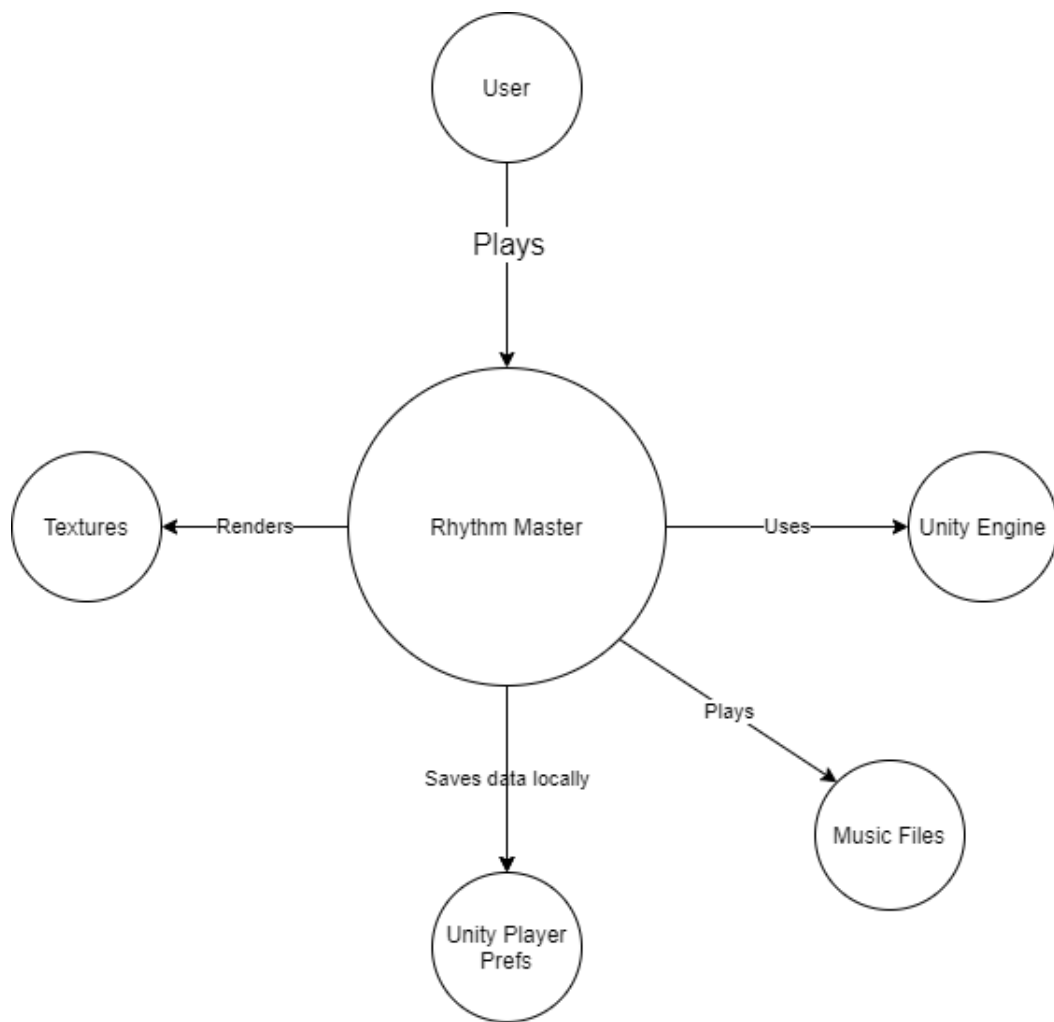


Figure 2: **NEW** Context diagram for Rhythm Master

3.1.2 Work Partitioning

Table 3: Work Partitioning Events

Event Number	Event Name	Input	Output
1	Starting a new game track	Keyboard/Mouse	Final Score
2	Reading the instructions	Keyboard/Mouse	Instructions
3	Opening the pause menu during game-play	Keyboard	Pause menu
4	Opening the settings menu	Mouse	Manipulated Hand
5	Viewing the leaderboard	Mouse	Leaderboard

Table 4: Work Partitioning Summaries

Event Number	Summary
1	The user, through the keyboard or mouse input, decides to start a new game track. At the end of the game track, the user will be shown their score.
2	The user, through keyboard or mouse input, chooses to read the instructions of playing Rhythm Master.
3	During a game track , the user can use keyboard input to open the pause menu.
4	During a game track, the user can use keyboard input to open the main menu.
5	The user, through the keyboard or mouse input, views the leaderboard for a specific game track .

3.1.3 Individual Product Use Cases



Figure 3: Use case diagram that displays the main functionalities of the application.

The use case diagram above shows the various way a user can interact with our application. As starting a game track involves the gameplay itself, it has lots of included use cases. The other uses cases such as view instructions, open settings menu, and view leaderboard are self-explanatory. The user can also access these 3 use cases using the pause menu, and that is shown using the extend relationship.

3.2 Functional Requirements

BE1. The **user** plays a new **track**.

FR1. The **system** must present the **user** with a blank **fret board** upon initializing the **track**.

FR2. The **system** must initialize the **score** to *INITIAL_SCORE* when initializing the **track**.

- FR3. The **system** must display **notes** for the **user** to play along with an indication of when to play them.
- FR4. The **system** must allow the **user** to play the **notes** through some interactive method.
- FR5. The **system** must award points every time the **user** plays a note accurately.
- FR6. The **system** must display the **user's** score during gameplay.
- FR7. The **system** must allow the **user** the option to save their score under a username when they are done with the **track**. The score should be saved locally using that username.
- FR8. The **system** must allow the **user** the option to redo the **track**.
- FR9. The **system** must allow the **user** the option to go back to the main menu.
- BE2. The user reads the instructions to playing Rhythm Master.
 - FR10. The **system** must provide instructions to the **user** on how to play the game.
 - FR11. The **system** must provide a way for the **user** to return to the main menu.
- BE3. The **user** opens the pause menu during a **track**.
 - FR12. The **system** must pause the game while the **pause menu** is open.
 - FR13. The **system** must allow the user to select one of the following options: opening the settings menu, going back to main menu, or restarting the **game track**.
 - FR14. The **system** must allow the user to close the **pause menu** and resume the game.
- BE4. The **user** opens the settings menu.
 - FR15. The **system** must allow the **user** to change the volume of the **game**.
 - FR16. The **system** must specify the version of the **game**.
 - FR17. The **system** must allow the **user** to rebind input.
 - FR18. The **system** must provide a way for the **user** to return to the main menu.
- BE5. The **user** opens the leaderboard.
 - FR19. The **system** must present the **user** with a list of **players** and their respective **scores**.
 - FR20. The **system** must allow the **user** to filter the **scores** based on when they were submitted
 - FR21. The **system** must provide a way for the **user** to return to the main menu.

4 Non-functional Requirements

4.1 Look and Feel Requirements

4.1.1 Appearance Requirements

LF1. The **user** interface must consist of only essential information relevant to the gameplay.

Fit Criterion: Two out of the three developers of **Rhythm Master** indicate that they agree with the above statement.

4.1.2 Style Requirements

LF2. The **user** must interpret the design to be heavily inspired by **Guitar Hero**.

Fit Criterion: 80% of all testers surveyed indicate that the game is heavily inspired by **Guitar Hero**.

LF3. The background of the **game track** must not use colours that will distract the **user** from the gameplay.

Fit Criterion: 90% of all testers surveyed indicate that the colours that were used did not distract them from the gameplay.

4.2 Usability and Humanity Requirements

4.2.1 Ease of Use Requirements

UH1. Default **game** controls must all be reachable at the same time using at most two hands.

Fit Criterion: 80% of all testers were able to reach the default **game** controls with at most two hands.

~~UH2. The **game** must only expect **user** inputs when they are first displayed on the screen.~~

UH2. The **game** must be easy for children age *MIN_AGE* to ~~use~~ play.

Fit Criterion: Tests will be ran with users of *MIN_AGE* with 80% stating that the **game** was easy to play.

4.2.2 Personalization and Internationalization Requirements

~~UH3. The **game** must provide themes for the **user** to choose from based on their preferences.~~

4.2.3 Learning Requirements

UH3. **Users** should be able to understand game mechanics within *MAX_PLAYTHROUGHS* play-throughs.

Fit Criterion: After *MAX_PLAYTHROUGHS* play-throughs, 90% of testers understood the game mechanics.

UH4. **Users** should be able to play the game with no prior experience or training.

Fit Criterion: 90% of testers with no prior experience or training were able to navigate through the interface and play the game.

UH5. The **game** must provide a set of instructions describing the **game's** rules and objectives and controls.

Fit Criterion: 95% of testers were able to navigate to the instructions page and were able to play the game without any issues on how to play.

4.2.4 Understandability and Politeness Requirements

UH6. The **game** must use common symbols and game terms for buttons and functions.

Fit Criterion: 90% of testers understood the symbols and game terms for buttons and functions after reading the instructions and navigating through the interface.

4.2.5 Accessibility Requirements

~~The **game** must be playable for **users** with colour blindness.~~
~~N/A~~

4.3 Performance Requirements

4.3.1 Speed and Latency Requirements

PE1. The **system** must maintain a minimum of *MIN_FRAMERATE* during gameplay.

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

~~PE2. The **system** must respond to all user inputs within *MAX_LATENCY* milliseconds.~~

~~PE3. The **system** must upload **scores** to the leaderboard in less than *MAX_UPLOAD_TIME* seconds.~~

4.3.2 Safety-Critical Requirements

~~N/A~~

4.3.3 Precision or Accuracy Requirements

PE2. The leaderboard must upload **scores** as **positive integer** values

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

~~PE3. The **system** must be able to time **user** inputs accurate to the nearest frame, up to a maximum of *MAX_LATENCY* milliseconds.~~

4.3.4 Reliability and Availability Requirements

~~N/A~~

4.3.5 Robustness or Fault-Tolerance Requirements

~~N/A~~

4.3.6 Capacity Requirements

PE4. The **system** must store at least *MIN_USER_SCORE_SAVES* user scores.

Fit Criterion: The developers of **Rhythm Master** will create *MIN_USER_SCORE_SAVES_TEST* fake users and ensure that the data was recorded by checking the contents of the text file that stores the leaderboard information.

4.3.7 Scalability or Extensibility Requirements

~~PE7. The **system** must allow developers to add additional game tracks without changing other components of the game.~~

~~N/A~~

4.3.8 Longevity Requirements

PE5. The **game** must be functional with existing software and hardware until Spring 2021.

Fit Criterion: The game runs as expected with full functionality on all three developer's computers.

4.4 Operational and Environmental Requirements

4.4.1 Expected Physical Environment

PE6. The **system** must not require an Internet connection to function correctly.

Fit Criterion: The game runs as expected with full functionality on a developer's computer, disconnected from the internet.

4.5 Requirements for Interfacing with Adjacent Systems

~~PE7. The **system** must not make changes to files outside its main directory.~~

4.5.1 Productization Requirements

PE7. The **game** must be distributed as a .EXE file.

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

PE8. The **game** must be less than *MAX_STORAGE* gigabytes.

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

4.6 Release Requirements

PE9. The product will have a final release in April 1246, 2021.

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

4.7 Maintainability and Support Requirements

4.7.1 Maintenance Requirements

MA1. Source code must be fully documented, via commenting and class diagrams.

Fit Criterion: All three developers, after thorough review, must agree on documentation quality.

MA2. Source code must all adhere to the same standard style.

Fit Criterion: All three developers, after thorough review, must agree on style matching.

4.7.2 Supportability Requirements

MA3. The **project's** main repository is to be made public, to allow **users** to raise issues.

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

4.7.3 Adaptability Requirements

MS1. The **game** shall be supported by any machine running on Windows 7 or newer, ~~macOS Sierra 10.12 or newer, or Linux Ubuntu 16.04 or newer.~~

Fit Criterion: This requirement is stated with quantifiable terms and does not need a fit criterion to validate it.

4.8 Security Requirements

4.8.1 Access Requirements

~~SR1. The **user** must have read-only access to other **high scores** on the leaderboard.~~

~~SR2. The **user** must have read-only access to their own **scores**.~~

~~N/A~~

4.8.2 Integrity Requirements

~~SR3. The **user** must not be able to modify any previously-submitted **scores**.~~

4.8.3 Privacy Requirements

SR1. The **user** must not be able to view any information about other **player's** other than the disclosed names ~~and, scores, and the date they achieved that scores~~ on the leaderboard.

Fit Criterion: 100% of testers must not be able to see any information about other **player's** other than the disclosed names, **scores**, and the date they achieved that score on the leaderboard.

4.8.4 Audit Requirements

N/A

4.8.5 Immunity Requirements

~~SR5. The **system** must not be vulnerable to attacks from intruders.~~

N/A

4.9 Cultural and Political Requirements

4.9.1 Cultural Requirements

~~CP1. The **system** shall not allow users to input names that are culturally offensive/inappropriate.~~

~~CP2. The **system** shall not allow users to input names that are in languages besides from English.~~

N/A

4.9.2 Political Requirements

N/A

4.10 Legal Requirements

4.10.1 Compliance Requirements

N/A

4.10.2 Standards Requirements

N/A

4.11 Health and Safety Requirements

N/A

5 Project Issues

5.1 Open Issues

There are no open issues on **Frets on Fire**'s official repository. The last commit was made in 2014.

The version of the game installed with the downloadable installer crashes immediately upon launching. Furthermore, the open-source repository requires an old version of **Python**, as well as slight code modifications, to compile.

5.2 Off-the-Shelf Solutions

Frets on Fire is the only notable open-source **Guitar Hero**-type game. **Clone Hero** is the best-known and most complete game of this genre on **PC**, but its source code is not public. **Clone Hero** contains many of the advantages **Rhythm Master** is intended to have over **Frets on Fire**, namely improved visuals and ease of installation on modern PCs.

Outside of **Frets on Fire**, there is not much code specific to this type of game to reference in the creation of **Rhythm Master**. Most adapted code, should any be used, will originate from **Frets on Fire**.

5.3 New Problems

Game performance must be a focus of the development team in order to maintain responsive gameplay. It is unknown how complicated replicating **Python** code behaviour in **C#** will be.

5.4 Tasks

Tasks are scheduled and delegated as per the project [Gantt Chart](#).

5.5 Migration to the New Product

N/A. **Rhythm Master** will work independently of **Frets on Fire**.

5.6 Risks

There is minimal risk with this type of project, because it is intended to run as a standalone program with little to no interaction with external systems. Risks originating from the program itself include excessive resource usage, unexpected crashes, and poor performance. In order to minimize these risks, the program should be tested on multiple hardware configurations of varying levels of performance.

5.7 Costs

This project will not have any monetary cost, because it will use open-source development software and resources. Time cost is estimated to be approximately *TIME_COST* hours of development and testing.

5.8 User Documentation and Training

5.8.1 Documentation

Player instructions will be included as an option in the game's main menu. These instructions will highlight the game's control scheme, and the logic behind scoring.

The game's directory will contain a README file, providing installation instructions.

5.8.2 Training

No specific training is necessary to play **Rhythm Master**. Controls should be intuitive, and practice should only be needed to improve player skill.

5.9 Waiting Room

Low priority additions, given extra development time, include:

1. Importing custom-made **game tracks**, from within the game.
2. Display resolution and graphics quality controls.

5.10 Ideas for Solutions

N/A.

References

- [1] James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. 16th ed. 2012.

6 Appendix

6.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

MIN_FRAMERATE = 30

INITIAL_SCORE = 10

MAX_LATENCY = ~~33~~

MAX_PLAYTHROUGHS = 2

MAX_STORAGE = 2

MAX_UPLOAD_TIME = ~~2~~

MIN_USER_SCORE_SAVES = 100

MIN_AGE = 10

TIME_COST = 60

MIN_USER_SCORE_SAVES_TEST = 101