SE 3XA3: Software Requirements Specification
Frets On Fire

Team #16, Rhythm Master
Almen Ng, nga18
David Yao, yaod9
Veerash Palanichamy, palanicv

February 13, 2021

Contents

List of Tables

List of Figures

Table 1: Revision History

Date	**Version**	**Notes**
February 2, 2021	1.0	Initial Document
February 8, 2021	1.1	Finished Project Drivers
February 10, 2021	1.2	Finished Non-Functional Requirements Section
February 12, 2021	1.3	Added Functional Requirements Section and respective diagrams

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 Additionally, we will be programming the **game** in a modern language, C#, improving on the outdated graphics, as well as providing meaningful test cases using Unity Test Framework.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 Maryan 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

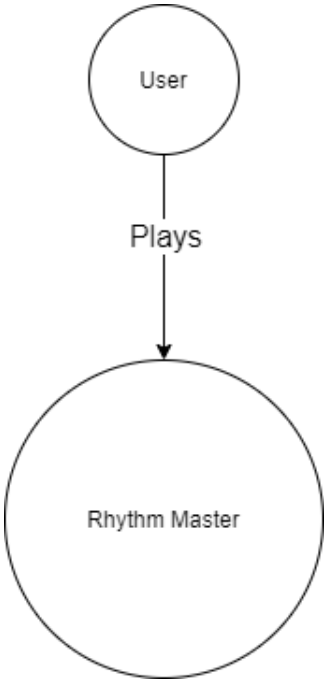

Figure 1: Context diagram for Rhythm Master

Rhythm Master is a standalone application that does not interact with other systems. It consists of one actor which is the user/player.

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 gameplay	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 2: 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 consists of only essential information relevant to the gameplay.

4.1.2 Style Requirements

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

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

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

UH3. The **game** must be easy for children age MIN_AGE to use

4.2.2 Personalization and Internationalization Requirements

UH4. The **game** must provide themes for the **user** to choose from based on their preferences.

4.2.3 Learning Requirements

UH5. **Users** should be able to understand game mechanics within $MAX\_PLAYTHROUGHS$ play-throughs.

UH6. **Users** should be able to play the game with no prior experience or training.

UH7. The **game** must provide a set of instructions describing the **game's** rules and objectives and controls.

4.2.4 Understandability and Politeness Requirements

UH8. The **game** must use common symbols and game terms for buttons and functions.

4.2.5 Accessibility Requirements

UH9. The **game** must be playable for **users** with colour blindness.

4.3 Performance Requirements

4.3.1 Speed and Latency Requirements

PE1. The **system** must maintain a minimum of $MIN\_FRAMERATE$ during gameplay.

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

PE4. The leaderboard must upload **scores** as integer values

PE5. 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

PE6. The **system** must store $MAX\_USER\_SCORE\_SAVES$ **user scores** on the leaderboard.

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.

4.3.8 Longevity Requirements

PE8. The **game** must be functional with existing software and hardware until Spring 2021.

4.4 Operational and Environmental Requirements

4.4.1 Expected Physical Environment

PE9. The **system** must not require an Internet connection to function correctly.

4.5 Requirements for Interfacing with Adjacent Systems

PE10. The **system** must not make changes to files outside its main directory.

4.5.1 Productization Requirements

PE11. The **game** must be distributed as a .EXE file.

PE12. The **game** must be less than $MAX\_STORAGE$ gigabytes.

4.6 Release Requirements

PE13. The product will have a final release in April 16, 2021.

4.7 Maintainability and Support Requirements

4.7.1 Maintenance Requirements

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

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

4.7.2 Supportability Requirements

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

4.7.3 Adaptability Requirements

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

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**.

4.8.2 Integrity Requirements

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

4.8.3 Privacy Requirements

SR4. The **user** must not be able to view any information about other **player's** other than the disclosed names and **scores** 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.

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 langauges asides from English.

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.

$MIN\_FRAMERATE = 30$
$INITIAL\_SCORE = 10$
$MAX\_LATENCY = 33$
$MAX\_PLAYTHROUGHS = 3$
$MAX\_STORAGE = 2$
$MAX\_UPLOAD\_TIME = 2$
$MAX\_USER\_SCORE\_SAVES = 100$
$MIN\_AGE = 10$
$TIME\_COST = 60$