3XA3 Test Report
Rhythm Master

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

April 12, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
April 10, 2021	1.0	Introduction and preliminary test cases
April 12, 2021	1.1	Complete test cases

Table 2: **Table of Definitions**

Term	**Definition**
C Sharp[C#]	The programming language used in this project.
Fret Board	A vertical musical staff upon which [Note]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.
Game track	The game track is where the gameplay happens. It consists of music track where the user interacts to score points.
Graphical User Interface	A visual representation of the program allowing user interaction
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]player to press.
Pause menu	The menu the user can open during a [Game track]game track
Player/User	The individual playing the [Game]game.
Python	The programming language used in Frets on Fire.
Score	A numerical value quantifying the [Player]player's performance in their last game.
Software Requirements Specification	A document that describes what the [System]system will do and the expected performance
System	The software of the [Game]game
Tester	An individual testing the game either via playing the game or inspecting the code.
Unity Test Framework	Automated software testing provided by the Unity game engine
Typeform	Website that builds surveys online surveys

This document details the complete testing process for Rhythm Master, as laid out in the project test plan. It contains an evaluation of the project's functional and non-functional requirements that are defined in the **Software Requirements Specification**, the changes made due to testing, and an analysis of the traceability between requirements and modules.

1 Functional Requirements Evaluation

1.1 Gameplay

Test #1:	**FR-GP-1**
Description:	Intial game track is empty
Type:	Manual
Initial State:	Empty screen within the Game track
Input:	An event of starting a new Game track
Output:	A blank fret board
Expected:	The fret board should be blank
Result:	PASS

Test #2:	**FR-GP-2**
Description:	Score should be set to 0 at the start of a game track
Type:	Automated
Initial State:	Initialization of the Game track
Input:	An event of starting a new Game track
Output:	Score should be set to 0
Expected:	Score should be set to 0
Result:	PASS

Test #3:	**FR-GP-3**
Description:	Notes are spawning
Type:	Manual
Initial State:	Game track should have been initialized
Input:	Game track has been initialized
Output:	[Note]Notes should be displayed
Expected:	Notes are displayed
Result:	PASS

Test #4:	**FR-GP-4**
Description:	Notes can be played using keyboard
Type:	Manual
Initial State:	Game track should have been initialized
Input:	An event of starting a new Game track
Output:	[Note]Notes are played
Expected:	The notes can be played using keyboard
Result:	PASS

Test #5:	**FR-GP-5**
Description:	Checking whether score goes up when a notes is played accurately
Type:	Manual
Initial State:	Game track should have been initialized
Input:	[Note]Notes are played accurately
Output:	Tester is awarded points
Expected:	Score goes up when note is played accurately
Result:	PASS

Test #6:	**FR-GP-6**
Description:	Checking whether score is displayed on the screen
Type:	Manual
Initial State:	Game track should have been initialized
Input:	Initialization and changes to [Score]score
Output:	[Score]score should be displayed
Expected:	Score is displayed on screen
Result:	PASS

Test #7:	**FR-GP-7**
Description:	Checking whether [Tester]Tester can save their score for a track
Type:	Manual
Initial State:	Game track should have been completed
Input:	[Tester]Tester chooses the option to save their score with a username
Output:	[Score]score should be saved locally with the given username
Expected:	Score is saved and can be viewed
Result:	PASS

Test #8:	**FR-GP-8**
Description:	Checking whether score does not change if invalid key is pressed
Type:	Manual
Initial State:	Game track should have been initialized
Input:	Tester presses an invalid key
Output:	Tester is not awarded points
Expected:	Score does not change
Result:	PASS

Test #9:	**FR-GP-9**
Description:	User tries to save the score with empty username
Type:	Manual
Initial State:	Game track should have been initialized
Input:	[Tester]Tester chooses the option to save their score with a username and enters empty username
Output:	The Game should provide a warning
Expected:	
Result:	PASS

1.1.1 User Interface

Test #10:	**FR-UI-1**
Description:	[Tester]Tester should be able to redo the Game track from the end game screen
Type:	Manual
Initial State:	Game track should have been completed
Input:	[Tester]Tester chooses the option to redo the Game track
Output:	Game track should be reinitialized
Expected:	
Result:	PASS

Test #11:	**FR-UI-2**
Description:	[Tester]Tester should be able to go to the main menu screen from the end game screen
Type:	Manual
Initial State:	Game track should have been completed
Input:	[Tester]Tester chooses the option to go back to the main menu
Output:	The game should display the main menu
Expected:	
Result:	PASS

Test #12:	**FR-UI-3**
Description:	[Tester]Tester should be able to view the instructions screen from the main menu screen
Type:	Manual
Initial State:	The [Tester]tester is viewing the main menu
Input:	Tester chooses the option to view instructions
Output:	The game should display instructions for the gameplay
Expected:	
Result:	PASS

Test #13:	**FR-UI-4**
Description:	[Tester]Tester can choose the option to go to the main menu from the instructions menu
Type:	Manual
Initial State:	The [Tester]tester is viewing the instructions
Input:	Tester chooses the option to return to the main menu
Output:	The game should display the main menu
Expected:	
Result:	PASS

Test #14:	**FR-UI-5**
Description:	Opening the pause menu should be pause the game
Type:	Manual
Initial State:	The [Tester]tester is playing a [Game track]game track
Input:	Tester opens the pause menu
Output:	The game should pause the game
Expected:	
Result:	PASS

Test #15:	**FR-UI-6**
Description:	The pause menu should allow you to open the settings menu or go back to the main menu
Type:	Manual
Initial State:	The [Tester]tester is playing a [Game track]game track
Input:	Tester opens the pause menu
Output:	The game should show the tester the options to opening the settings menu, going back to main menu, or restarting the [Game track]game track
Expected:	
Result:	PASS

Test #16:	**FR-UI-7**
Description:	The pause menu should continue the game when the pause menu has been closed
Type:	Manual
Initial State:	The pause menu has been opened
Input:	[Tester]Tester closes the pause menu
Output:	The game should show stop showing the pause menu and resume the gameplay
Expected:	
Result:	PASS

Test #17:	**FR-UI-8**
Description:	The game volume can be changed using the settings menu
Type:	Manual
Initial State:	The settings menu has been opened
Input:	[Tester]Tester specifies the volume of the game to some level using the tester interface
Output:	The game should change the volume of the game accordingly
Expected:	
Result:	PASS

Test #18:	**FR-UI-9**
Description:	The settings menu should display the version number
Type:	Manual
Initial State:	The settings menu has been opened
Input:	[Tester]Tester opened the settings menu
Output:	The game should display the version of the game
Expected:	
Result:	PASS

Test #19:	**FR-UI-10**
Description:	The settings menu should have the option to rebind the gameplay keys to the one preferred by the [User]User
Type:	Manual
Initial State:	The settings menu has been opened
Input:	[Tester]Tester chooses to rebind their input keys to some specific key using the user interface
Output:	The game should change the input keys to the one specified by the user
Expected:	
Result:	PASS

Test #20:	**FR-UI-11**
Description:	The settings scene accessed from the main menu should have the option to go back to the main menu
Type:	Manual
Initial State:	The settings menu has been opened
Input:	[Tester]Tester chooses to go to the main menu
Output:	The game should display the main menu screen
Expected:	
Result:	PASS

Test #21:	**FR-UI-12**
Description:	Leaderboard scene where player rankings can be scene should be accessible from the main menu
Type:	Manual
Initial State:	The leaderboard screen is being displayed
Input:	[Tester]Tester chooses to view the leaderboard
Output:	The game should display a list of players and their respective score
Expected:	
Result:	PASS

Test #22:	**FR-UI-14**
Description:	There should be an option to return to the main menu scene from the leaderboard scene
Type:	Manual
Initial State:	The leaderboard screen is being displayed
Input:	[Tester]Tester chooses to return to the main menu
Output:	The game should display the main menu screen
Expected:	
Result:	PASS

Test #23:	**FR-UI-15**
Description:	Viewing a leaderboard when no score was saved should not throw an error and shod display an empty leaderbaord
Type:	Manual
Initial State:	Game is on the main menu
Input:	[Tester]Tester chooses to view the leaderboard
Output:	The game should display an empty list
Expected:	
Result:	PASS

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel Requirements

Test #24:	**NFR-1-LF1**
Description:	Tests that the user interface only contains essential information using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 90%

Test #25:	**NFR-2-LF2**
Description:	Tests that the design was heavily inspired by Frets on Fire using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 100%

Test #26:	**NFR-2-LF3**
Description:	Tests that the background of the [Game track]game trackis not too distracting using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 82%

2.2 Usability

Test #27:	**NFR-3-UH1**
Description:	Tests that the cotrols are reachable at the same time using at most two hands
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 100%

Test #28:	**NFR-3-UH2**
Description:	Tests that all the game is easily playable for children greater or equal than MIN_AGE using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group made of children of age of MIN_AGE or greater
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 85%

Test #29:	**NFR-3-UH2**
Description:	Tests that all the game is easily playable for children greater or equal than MIN_AGE using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group made of children of age of MIN_AGE or greater
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 85%

Test #30:	**NFR-3-UH3**
Description:	Test that the user should be able to understand the game mechanics within $MAX\_PLAYTHROUGHS$. Test was done using a suvey result whether the players understood the game after $MAX\_PLAYTHROUGHS$ runs
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 100%

Test #31:	**NFR-3-UH4**
Description:	Test that the game is playable with no prior experience or training. Test done using using a binary survey with the options of agree and disagree. type
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 100%

Test #32:	**NFR-3-UH5**
Description:	Test that the game provides a set of instructions describing the game's rules and objectives. Test done using using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 100%

Test #33:	**NFR-4-UH6**
Description:	Test that the game uses common symbols and game terms for user interfaces. Test done using using a binary survey with the options of agree and disagree
Type:	Manual
Tester(s):	Testing group
Pass:	Average survey score of at least Θ
Result:	PASSED with an agreement of 91%

2.3 Performance

To test if the test case NFR-5-PE1 where the system should maintain a fps of $MIN\_FRAMERATE$ during gameplay, a fps tracker was used to track the fps of the game, and that data was then graphed which can be found in the figure below.

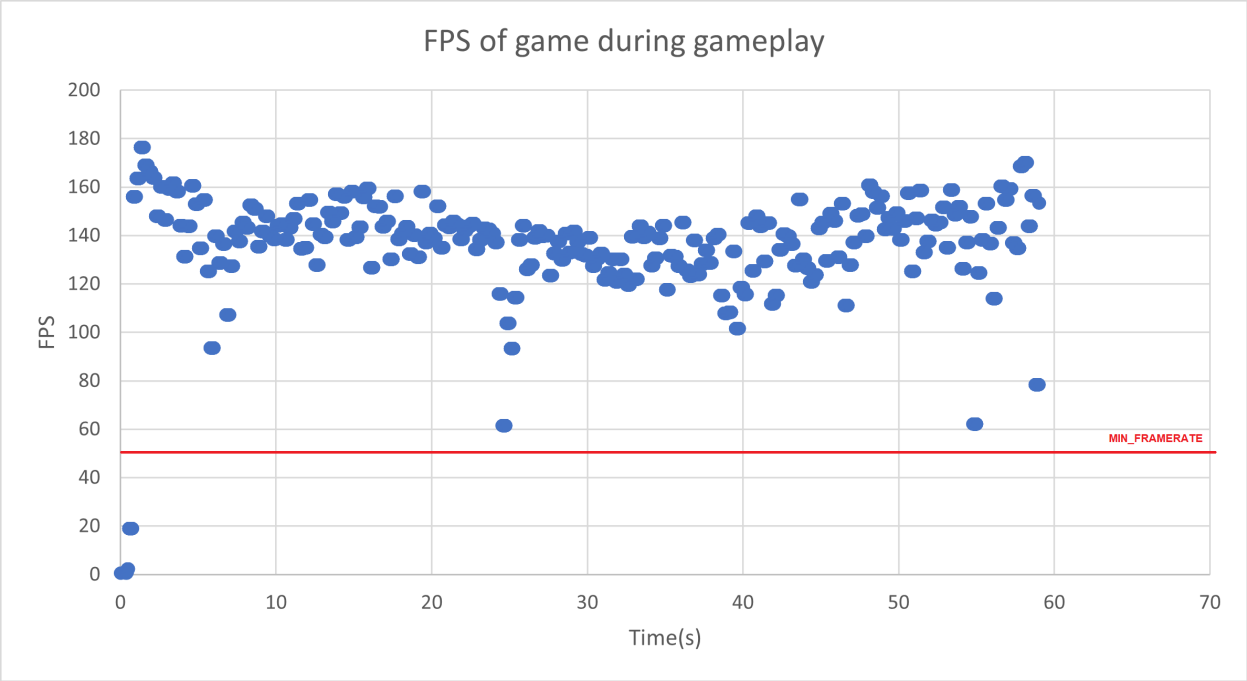

Figure 1: Framerate of game during a game track playthrough

The tests were run using the systems specified in the table below.

Table 3: Systems used in performance testing

System	**Hardware**
Medium performance	i7 8550U @ 2.4 GHz

The test is considered successful as the system has been able to maintain frame-rates above the $MIN\_FRAMERATE$ during gameplay.

Test #34:	**NFR-10-PE4**
Description:	Test that the system can store atleast $MIN\_USER\_SCORE\_SAVES\_TEST$ user scores. Test was carried out using automated script where $MIN\_USER\_SCORE\_SAVES\_TEST$ scores were saved, and checked whether all scores still existed
Type:	Automated
Tester(s):	Testing group
Pass:	All scores still persists
Result:	PASSED

3 Comparison to Existing Implementation

In the original implementation, Frets on Fire, there were 18 test cases that tested for each component without referencing any documentation that would aid in the traceability of the test cases to functional and non-functional requirements. The test cases were not thorough as it did not test for all components of the system and it lacked testing for non-functional requirements. With Rhythm Master, the development team wrote a software requirements specification, test plan document, and design documentation to aid in the traceability to requirements and modules. Moreover, this document also serves as a way to keep track of all the tests that were done and to provide The team considered and used a variety of different testing including specification testing, functional testing, and exploratory testing and completed a total of 50 test cases prior to the final demonstration of the project.

4 Unit Testing

Unit testing was only used to test individual components of modules because of the highly interconnected nature of video games such as Rhythm Master. The numerous modules of the project must be correct in their interactions with each other, and it was too costly to conduct unit testing on each one to confirm correct behaviour. Large, experience-based projects such as this are better served via manual and exploratory testing, in order to ascertain correct user experiences.

5 Changes Due to Testing

Formal testing did not reveal any necessary changes in terms of module interfacing, decomposition, or internal design. Changes made to code were to address bugs and logical errors revealed by the testing plan. In terms of the gameplay experience, graphical improvements were made throughout the development process in response to feedback from developers and informal testers.

6 Automated Testing

Automated testing was only a minimal part of testing due to the fact that the software being tested is a game which is GUI based. The testing of the software required user inputs and exploration of various scenarios in the game. However, unit tests that test each game component's functionality was automated using Unity Test Framework,

7 Trace to Requirements

7.1 Traceability Between Test Cases and Requirements

Traceability matrices can be found on the next few pages.

Table 4: **Traceability Matrix for Gameplay Requirements**

		Requirements						
		FR1	FR2	FR3	FR4	FR5	FR6	FR7
Test Cases	FR-GP-1	X						
	FR-GP-2		X					
	FR-GP-3			X				
	FR-GP-4				X			
	FR-GP-5					X		
	FR-GP-6						X	
	FR-GP-7							X
	FR-GP-8					X		
	FR-GP-9						X	

Table 5: **Traceability Matrix for UI Requirements**

		Requirements													
		FR8	FR9	FR10	FR11	FR12	FR13	FR14	FR15	FR16	FR17	FR18	FR19	FR20	FR21
Test Cases	FR-UI-1	X													
	FR-UI-2		X												
	FR-UI-3			X											
	FR-UI-4				X										
	FR-UI-5					X									
	FR-UI-6						X								
	FR-UI-7							X							
	FR-UI-8								X						
	FR-UI-9									X					
	FR-UI-10										X				
	FR-UI-11											X			
	FR-UI-12												X		
	FR-UI-13													X	
	FR-UI-14														X
	FR-UI-15												X		

Table 6: **Tracability Matrix for Non-Functional Requirements**

		Requirements										
		LF1	LF3	UH2	UH5	PE1	LF2	PE6	UH1	UH5	UH6	PE4
Test Cases	NFR-1-LF1	X										
	NFR-2-LF2						X					
	NFR-2-LF3		X									
	NFR-3-UH1								X			
	NFR-3-UH2			X								
	NFR-3-UH5									X		
	NFR-4-UH6										X	
	NFR-4-UH5				X							
	NFR-5-PE1					X						
	NFR-6-PE2											
	NFR-7-PE4							X				X
	NFR-8-PE7											
	NFR-9-PE9											

8 Trace to Modules

Req.	Modules
FR-GP-1	m6
FR-GP-2	m6, m15
FR-GP-3	m10, m6
FR-GP-4	m1
FR-GP-5	m6, m15, m13
FR-GP-6	m6
FR-GP-7	m6, m6
FR-GP-8	m1
FR-GP-9	m6
FR-UI-1	m6
FR-UI-2	m6
FR-UI-3	m9
FR-UI-4	m9
FR-UI-5	m11, m6
FR-UI-6	m11, m4
FR-UI-7	m11, m6
FR-UI-8	m4
FR-UI-9	m4, m2
FR-UI-10	m4
FR-UI-11	m5
FR-UI-12	m8
FR-UI-13	m5
FR-UI-14	m5
FR-UI-15	m5

Table 7: Trace Between Requirements and Modules

9 Code Coverage Metrics

9.1 Symbolic Parameters

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

$MIN\_FRAMERATE = 30$
$INITIAL\_SCORE = 10$
$MAX\_LATENCY = 33$
$MAX\_PLAYTHROUGHS = 2$
$MAX\_STORAGE = 2$
$MAX\_UPLOAD\_TIME = 2$
$MIN\_USER\_SCORE\_SAVES = 100$
$MIN\_AGE = 10$
$TIME\_COST = 60$
$MIN\_USER\_SCORE\_SAVES\_TEST = 101$
$\Theta = 80$