SE 3XA3: Test Plan
Tetris Tussle

Nicholas Lobo
lobon3
400179304

Matthew Paulin
paulinm
400187147

David Carrie
carriedd
000661652

March 5, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2/26/2021	1.0	Matthew, Nicolas and David initial work on the template
2/28/2021	1.1	Matthew, Nicolas and David completed section 3
3/04/2021	1.3	Matthew, Nicolas and David Completed Document for first submission.

1 General Information

1.1 Purpose

The purpose of testing our project is to verify that the form of satisfies the functions that are necessitated by the software requirements specification. Additionally, there may arise additional requirements that are discovered throughout the testing process.

1.2 Scope

The test plan is used to create methods by which the correctness and functionality of the project can be verified. This document will include what components of the project are being tested, for what purpose they are being tested, and how to test them.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Definitions**

Term	**Definition**
Tetromino	A Tetromino is the geometric shape used in Tetris. It is composed of four squares connected at their edges to form 7 different shapes Definition1

1.4 Overview of Document

This document outlines the detailed testing plan that will be executed to give assurance that the system is behaving as stated in the requirements. It will lay out our plans for different types of testing including: acceptance, integration and unit testing as well as information regarding software tools to be used, personnel assignments, and scheduling.

2 Plan

2.1 Software Description

The software being created is an updated and improved version of Tetris with multiplayer functionality and revised graphics.

2.2 Test Team

All group members currently working on the project will be part of the testing team. This includes: Nicholas Lobo, Matthew Paulin, David Carrie.

2.3 Automated Testing Approach

All automated software tests will be completed through the use of the testing framework Jest.

2.4 Testing Tools

We will be using Jenkins to create a continuous integration testing environment and Jest to automate unit testing. Additionally, acceptance testing will be completed by some subset of the stakeholders to ensure that all the non functional requirements are sufficiently met.

2.5 Testing Schedule

See Gantt Chart at the following url ... (TO UPDATE LATER)

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Controller Input Tests

FR-CI-1. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and in the middle of the board

Input: The D key is pressed

Output: The Tetromino is shifted one board space to the right

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-2. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and in the middle of the board

Input: The A key is pressed

Output: The Tetromino is shifted one board space to the left

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-3. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and in the middle of the board

Input: The S key is pressed

Output: The Tetromino is shifted one board space down

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-4. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and on the right edge of the board

Input: The D key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-5. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and on the left edge of the board

Input: The A key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-6. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and on the bottom edge of the board

Input: The S key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-7. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is a Tetromino piece currently to the right of it.

Input: The D key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-8. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is a Tetromino piece currently to the left of it.

Input: The A key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-9. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is a Tetromino piece currently below it.

Input: The S key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-10. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in the middle of the board

Input: The W key is pressed

Output: The Tetromino rotates 90°

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-11. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and is currently on the edge of the board

Input: The W key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-12. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is a Tetromino piece blocking its rotation path.

Input: The W key is pressed

Output: The Tetromino stays in the same position

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-13. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there are no Tetromino pieces on the board

Input: The space-bar key is pressed

Output: The Tetromino is placed directly on the bottom edge of the board

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-14. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is a Tetromino piece directly below it.

Input: The space-bar key is pressed

Output: The Tetromino is placed directly on top of the other Tetromino

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-15. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is no Tetromino currently being held.

Input: The Z key is pressed

Output: The Tetromino piece is saved off of the board and new active Tetromino piece is on the board

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-16. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and there is a Tetromino currently being held.

Input: The Z key is pressed

Output: The Tetromino piece is saved off of the board and the Tetromino piece that was being held is the new active Tetromino piece and is on the board

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

FR-CI-17. **Type:** Dynamic, Manual

Initial State: The user is in control of a Tetromino that is currently in play and the user just held a Tetromino.

Input: The Z key is pressed

Output: The current Tetromino stays on the board

How test will be performed: The inputs will be pressed manually by the user and the output will be observed on the screen

3.1.2 User Interaction

FR-UI-1. **Type:** Dynamic, Manual

Initial State: Title screen

Input: The view leaderboard button is pressed

Output: The leaderboard is displayed

How test will be performed: The menu button will be manually clicked and the output observed on the screen

FR-UI-2. **Type:** Dynamic, Manual

Initial State: Game is currently being played

Input: The game is sent to the game over state

Output: The option to view leader board is displayed

How test will be performed: A game will be ended by the user and the output on the screen verified.

FR-UI-3. **Type:** Dynamic, Manual

Initial State: Leaderboard displayed

Input: The user attempts to scroll through the leader board

Output: The display on the leader board is updated appropriately

How test will be performed: A user will access the leader board and manually attempt to scroll through the leader board and verify the result.

FR-UI-4. **Type:** Dynamic, Manual

Initial State: Singleplayer leaderboard displayed

Input: The user clicks view multiplayer leaderboard

Output: The display on the leader board is updated to multiplayer information

How test will be performed: A user will access the leader board and manually attempt to click the multiplayer leaderboard button.

FR-UI-5. **Type:** Dynamic, Manual

Initial State: multiplayer menu is displayed

Input: The user clicks to create a multiplayer lobby

Output: A multiplayer lobby is created and displayed and a unique room number is displayed **How test will be performed:** The menu button will be manually clicked and the output observed on the screen

FR-UI-6. **Type:** Dynamic, Manual

Initial State: multiplayer menu is displayed

Input: unique room code into join lobby field and join button pressed

Output: the correct lobby is joined

How test will be performed: The input will be done manually and the output observed.

FR-UI-7. **Type:** Dynamic, Manual

Initial State: multiplayer lobby is displayed with 2 players

Input: unique room code into join lobby field and join button pressed by a third client

Output: error messages

How test will be performed: The input will be done manually and the output observed.

FR-UI-8. **Type:** Dynamic, Manual

Initial State: User is in a multiplayer lobby

Input: User joins another multiplayer lobby

Output: user is removed from current lobby and placed into new lobby

How test will be performed: The input will be done manually and the output observed.

FR-UI-9. **Type:** Dynamic, Manual

Initial State: User is in the title screen

Input: User starts a sing player game

Output: user is brought to the active game board screen

How test will be performed: The input will be done manually and the output observed.

FR-UI-10. **Type:** Dynamic, Manual

Initial State: User is on the option screen

Input: User increases the volume level

Output: The sound coming from the system has increased

FR-UI-11. **Type:** Dynamic, Manual

Initial State: User is on the option screen

Input: User changes controls from default to arrow keys

Output: The controls of the game are switched to movement using the arrow keys

How test will be performed: The input will be done manually and the output observed.

3.1.3 Board Interactions

FR-BI-1. **Type:** Dynamic, Manual

Initial State: None

Input: The user starts a new game

Output: The board must be empty and be 10 by 20 board spaces.

How test will be performed: A test in our test suite will create the scenario and create a new game and verify the board state.

FR-BI-2. **Type:** Dynamic, Manual

Initial State: Empty game board

Condition: a piece is placed into a valid spot on the board

Result: The board state is updated with the required board space occupied.

How test will be performed: A test in our test suite will create the scenario and add a Tetromino to an empty game board and verify the board state.

FR-BI-3. **Type:** Dynamic, Manual

Initial State: Game board with board space occupied at target location

Condition: a piece attempts to be added to an invalid location

Result: The board state remains in valid state.

How test will be performed: A test in our test suite will create the scenario and add a Tetromino to an invalid spot on the game board and verify the board state.

FR-BI-4. **Type:** Dynamic, Manual

Initial State: Game board has 9/10 of the required spaces occupied for clearing a row.

Input: a piece is placed into the 10th spot on the board.

Result: The board state in the row that the line was cleared is set vacant

How test will be performed: A test in our test suite will create the scenario and add a Tetromino to complete the line and verify the board is updated

FR-BI-5. **Type:** Dynamic, Manual

Initial State: Game board has 9/10 of the required spaces occupied for clearing a row.

Input: a piece is placed into the 10th spot on the board.

Result: The score of player is increased.

How test will be performed: A test in our test suite will create the scenario and add a Tetromino to complete the line and verify the score is increased by 100

FR-BI-6. **Type:** Dynamic, Manual

Initial State: Row 1 - 19 of the board state is occupied for all columns.

Input: a piece is added that requires the board to exceed the 20 row height limit

Result: The game over screen is shown.

How test will be performed: The user will manually create the situation and observe the outcome.

FR-BI-7. **Type:** Dynamic, Manual

Initial State: Game is active

Input: None

Output: Tetrominos appear in a random order

How test will be performed: The user will manually observe the generated Tetrominos.

FR-BI-8. **Type:** Dynamic, Manual

Initial State: Game is active

Input: None

Output: The currently active Tetromino falls at a predictable rate from the top to the bottom of the board

How test will be performed: The user will manually observe the game state

FR-BI-9. **Type:** Dynamic, Manual

Initial State: Game is active

Input: None

Output: The rate of the Tetrominos decent will increase as periodically

How test will be performed: The user will manually observe the rate of falling of the Tetrominos

FR-BI-10. **Type:** Dynamic, Manual

Initial State: Game is active

Input: None

Output: The next Tetromino will be displayed in the ”next tetromino section of the board” and the next Tetromino to become active on the board will be of that type.

How test will be performed: The user will observe the next Tetromino and manually verify it is correct.

FR-BI-11. **Type:** Dynamic, Manual

Initial State: Active game

Input: None

Output: There must be displayed on screen an option to close the application, and when clicked the game is terminated.

How test will be performed: The user will verify that the option to close the application is visible and when pressed it closes the application.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel Requirement

NFR-LFR1. **Type:** Dynamic, Manual

Initial State: Single player game of Tetris is running on a computer with the minimum recommended hardware

Input: 60 successive inputs of the up arrow in one second to rotate the Tetromino

Output: There are at least 30 display updates per second, reflecting the updated Tetromino rotation

How test will be performed: A macro will perform the keyboard input and a variable will keep track of the difference in the times to display the first and last frames

NFR-LFR2. **Type:** Dynamic, Manual

Initial State: Single Player Tetris Game running

Input: Users are asked to play one game of Tetris

Output: A majority of users find the graphics of the game to be visually appealing

How test will be performed: The users will be given a survey to rate the extent to which they like the game's graphics

NFR-LFR3. **Type:** Dynamic, Manual

Initial State: The game is running on a computer and the main menu of the game is being displayed

Input: Users with good vision or are wearing prescription corrective lenses are asked to read the different options being displayed on the menu

Output: A majority of users will be able to read all options being displayed on the menu

How test will be performed: The users will be asked to real aloud the text to ensure that it matches the given text.

NFR-LFR4. **Type:** Static, Manual

Initial State: The game's css and processing code (for the graphics) are completed

Input: The completed code

Output: The typeface, iconography, and colouring are consistent

How test will be performed: All the typeface, iconography, and colouring in the css and processing code will be analysed to check for consistency

3.2.2 Usability and Humanity Requirements

NFR-UHR1. **Type:** Dynamic, Manual

Initial State: None

Input: User interacts and plays game

Output: 9/10 users must be able to properly use game functions with provided controls.

How test will be performed: Users will be asked to test the game with a developer present. The developer will verify that the user is able to properly access all functions without difficulty.

NFR-UHR2. **Type:** Static, Manual

Initial State: None

Input: Tetromino RGB colors and Web Content Accessibility Guidelines

Output: The Tetromino RGB colors will conform to the given guidelines

How test will be performed: A developer will compare the RBG colors used by the Tetrominos and ensure it falls within the Web Content Accessibility Guidelines.

NFR-UHR3. **Type:** Dynamic, Manual **Initial State:** The game is running on a computer and the main menu of the game is being displayed **Input:** User is asked to start a single player game of Tetris **Output:** A majority of users will be able to locate the options needed to start a single player game of Tetris. How tests will be performed: A select group of users will be given the game in the main menu state. They will be asked to start a single player game of Tetris. If the majority of users are able to intuitively start a game of Tetris the test will be considered successful.

NFR-UHR4. **Type:** Static, Manual

Initial State: None

Input: Access all game menus and functions

Output: The text in all cases will be legible and understandable

How test will be performed: A developer will ask a user with at least a fifth grade reading comprehension and ensure they can read and understand the text.

3.2.3 Performance Requirements

NFR-PR1. **Type:** Dynamic, Manual

Initial State: Active single player game

Input: Move Tetromino

Output: The Tetromino must visibly move within 0.25 seconds

How test will be performed: A developer set up a timer and send the command to move the Tetromino. They will then verify it moves within 0.25 seconds.

NFR-PR2. **Type:** Dynamic, Manual

Initial State: None

Input: 20 different multiplayer games running from 40 clients

Output: The games must function without error

How test will be performed: A developer will start the server and then start 20 multiplayer games and connect them to 40 different game clients manually.

NFR-PR3. **Type:** Dynamic, Manual

Initial State: Active multiplayer game

Input: A drop block command is input 10 times

Output: All inputs are processed within 30 seconds

How test will be performed: Google Lighthouse will record the time it takes for network requests to be processed by the server.

NFR-PR4. **Type:** Dynamic, Manual

Initial State: Game service running

Input: none

Output: The server will be available 99.99

How test will be performed: A developer will access the server logs to ensure the downtime is under 0.01

NFR-PR5. **Type:** Dynamic, Manual

Initial State: Game menu

Input: Start a multiplayer game

Output: The game will play successfully over the internet

How test will be performed: A developer will start a multiplayer game with another developer on separate networks and will verify the results.

3.2.4 Operational and Environmental Requirements

NFR-OER1. **Type:** Dynamic, Manual

Initial State: The software is running on 3 different computers each running a different operating system(Windows 10, Mac OSX, Linux)

Input: The single player game mode is selected

Output: The single player game of Tetris is running on all 3 operating systems

How test will be performed: The software will be run on the Windows 10, Mac OSX, and Linux operating systems. If a full game of Tetris is able to be played without crashing the test will be considered successful.

NFR-OER2. **Type:** Dynamic, Manual

Initial State: None

Input: Start a single player game

Output: The game will launch successfully

How test will be performed: A developer start one single player game in each of Firefox, Safari and Google Chrome and verify the game completes successfully.

3.2.5 Maintainability and Support Requirements

NFR-MSR1. **Type:** Static, Manual

Initial State: None

Input: Server log

Output: The server is down for less than 10 minutes due to maintenance

How test will be performed: A developer will access the log files after and verify maintenance takes less than 10 minutes.

NFR-MSR2. **Type:** Static, Manual

Initial State: None

Input: Source code

Output: The connection between clients uses HTTPS.

How test will be performed: A developer will verify in the source code that the Hypertext Transfer Protocol Secure (HTTPS) is being used to communicate with clients from the server.

3.2.6 Security Requirements

NFR-SR1. **Type:** Static, Manual

Initial State: The project's source code is available on Gitlab with commit logs

Input: A developer will inspect the commit logs for all the braches

Output: The commit logs demonstrate that only the developers have modified the code

How test will be performed: The Git commit logs will be inspected to check the author(s) of all the commits to the source code.

NFR-SR2. **Type:** Dynamic, Manual

Initial State: An instance of the game is running on a browser and connected to the server

Input: A developer will connect to the server on a supported browser

Output: The https prefix is contained within the url and the browser shows that an encrypted connection was achieved

How test will be performed: A developer will connect to the server and will inspect the website to ensure that an SSL certificate is obtained.

3.2.7 Cultural Requirements

NFR-CR1. **Type:** Static, Manual

Initial State: None

Input: Source code, Anti Defamation League definitions(ADL).

Output: There is no use of hate symbols defined by the ADL.

How test will be performed: A developer will read the source code to alongside the ADL definitions and verify there are no violations.

3.2.8 Legal Requirements

NFR-LR1. **Type:** Static, Manual

Initial State: None

Input: System knowledge and source code.

Output: There are no known breaches of known laws.

How test will be performed: A developer will use their general knowledge of Federal, provincial and local law and verify the system does not violate any known laws.

3.2.9 Health and Safety Requirements

HSR1. **Type:** Static, Manual

Initial State: The system is running on the browser and on the main menu screen.

Input: The system is left running for 60 minutes

Output: A text warning is displayed on the menu stating to take a break from the game

How test will be performed: A developer will run the game for 60 minutes. If the text is displayed on the screen the test is successful.

HSR2. **Type:** Dynamic, Manual

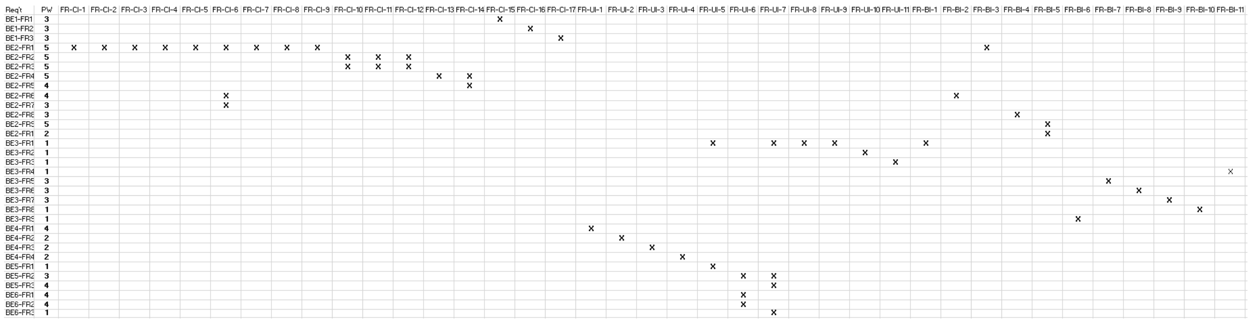
Initial State: Active single player game

Input: Tetromino movement commands.

Output: The system will not violate guidelines set out by WCAG for epilepsy.

How test will be performed: A developer will run the game and compare the results to the WCAG guidelines.

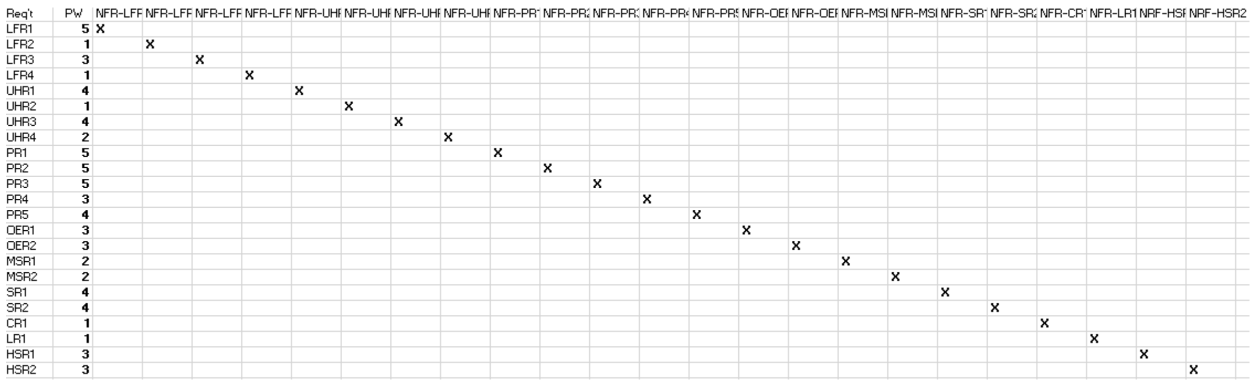
3.3 Traceability Between Test Cases and Requirements



Req't	PW	FR-CI-1	FR-CI-2	FR-CI-3	FR-CI-4	FR-CI-5	FR-CI-6	FR-CI-7	FR-CI-8	FR-CI-9	FR-CI-10	FR-CI-11	FR-CI-12	FR-CI-13	FR-CI-14	FR-CI-15	FR-CI-16	FR-CI-17
BE1-FR1	3															x		
BE1-FR2	3																x	
BE1-FR3	3																	x
BE2-FR1	5	x	x	x	x	x	x	x	x	x								
BE2-FR2	5										x	x	x					
BE2-FR3	5										x	x	x					
BE2-FR4	5													x	x			
BE2-FR5	4														x			
BE2-FR6	4						x											
BE2-FR7	3						x											
BE2-FR8	3																	
BE2-FR9	5																	
BE2-FR1	2																	
BE3-FR1	1																	
BE3-FR2	1																	
BE3-FR3	1																	
BE3-FR4	1																	
BE3-FR5	3																	
BE3-FR6	3																	
BE3-FR7	3																	
BE3-FR8	1																	
BE3-FR9	1																	
BE4-FR1	4																	
BE4-FR2	2																	
BE4-FR3	2																	
BE4-FR4	2																	
BE5-FR1	1																	
BE5-FR2	3																	
BE5-FR3	4																	
BE6-FR1	4																	
BE6-FR2	4																	
BE6-FR3	1																	

Req't	FR-UI-1	FR-UI-2	FR-UI-3	FR-UI-4	FR-UI-5	FR-UI-6	FR-UI-7	FR-UI-8	FR-UI-9	FR-UI-10	FR-UI-11	FR-BI-1	FR-BI-2	FR-BI-3	FR-BI-4	FR-BI-5	FR-BI-6	FR-BI-7	FR-BI-8	FR-BI-9	FR-BI-10	FR-BI-11
BE1-FR1																						
BE1-FR2																						
BE1-FR3																						
BE2-FR1														x								
BE2-FR2																						
BE2-FR3																						
BE2-FR4																						
BE2-FR5																						
BE2-FR6													x									
BE2-FR7																						
BE2-FR8															x							
BE2-FR9																x						
BE2-FR1																x						
BE3-FR1					x		x	x	x			x										
BE3-FR2										x												
BE3-FR3											x											
BE3-FR4																						x
BE3-FR5																		x				
BE3-FR6																			x			
BE3-FR7																				x		
BE3-FR8																					x	
BE3-FR9																	x					
BE4-FR1	x																					
BE4-FR2		x																				
BE4-FR3			x																			
BE4-FR4				x																		
BE5-FR1					x																	
BE5-FR2						x	x															
BE5-FR3							x															
BE6-FR1						x																
BE6-FR2						x																
BE6-FR3							x															

Figure 1: Traceability Matrix Fig.1



Req't	PW	NFR-LFF	NFR-LFF	NFR-LFF	NFR-LFF	NFR-UHF	NFR-UHF	NFR-UHF	NFR-UHF	NFR-PR1	NFR-PR2	NFR-PR3	NFR-PR4	NFR-PR5	NFR-OEF	NFR-OEF	NFR-MSI	NFR-MSI	NFR-SR1	NFR-SR2	NFR-CR1	NFR-LR1	NRF-HSF	NRF-HSR2
LFR1	5	X																						
LFR2	1		X																					
LFR3	3			X																				
LFR4	1				X																			
UHR1	4					X																		
UHR2	1						X																	
UHR3	4							X																
UHR4	2								X															
PR1	5									X														
PR2	5										X													
PR3	5											X												
PR4	3												X											
PR5	4													X										
OER1	3														X									
OER2	3															X								
MSR1	2																X							
MSR2	2																	X						
SR1	4																		X					
SR2	4																			X				
CR1	1																				X			
LR1	1																					X		
HSR1	3																						X	
HSR2	3																							X

Figure 2: Traceability Matrix Continued

4 Tests for Proof of Concept

4.1 Multiplayer

Multiplayer Test

POC-1. **Type:** Dynamic, Manual

Initial State: Multiplayer Game active with 2 users

Input: Tetromino movement command

Output: For User 2: There must be displayed on screen two different boards one which is not controlled by user 2 and which shows the movement of user 1's Tetromino.

How test will be performed: User will manually open 2 clients and connect to a multiplayer game. user will then input keystrokes to move a Tetromino in one client which observing if it moves in the other client.

4.2 User Interface and Gameplay

User Interface Test

POC-2. **Type:** Dynamic, Manual

Initial State: Active Single player game

Input: Tetromino movement command right

Output: The Teteromino will move right one unit

How test will be performed: User will manually start a game and press the keys to move the tetromino and verify the result.

Gameplay Test

POC-3. **Type:** Dynamic, Manual

Initial State: Active Single player game

Input: Movement commands to cause a full row of spaces to be occupied by tetrominos.

Output: The row will be removed from the board, the above rows will be lowered by 1 space and a new empty row placed on top.

How test will be performed: User will manually start and cause the board to be in a state through movement commands where an entire row is occupied, then verify the results.

5 Comparison to Existing Implementation

N/A No test plan is available for the current implementation

6 Unit Testing Plan

Unit testing will be facilitated through the Jest framework.

6.1 Unit testing of internal functions

To increase the efficiency of the testing process, we will use a hard-coded test oracle that encodes the expected results of the test cases as well as the code to set up the test cases. Once the test suite is run, the framework will indicate the whether the tests pass or fail, context about the failure as well as code coverage metrics. The results of the test will indicate where more developmental effort needs to be placed to increase the level of robustness. Additionally, the code coverage indicator will drive the placement of additional tests to get a higher level of coverage. Since games can be difficult to test, testing will be focused on aspects of the code that can be tested against an expected output. Therefore, only certain functions and methods will be able to be tested. That being said, the goal is to reach at least statement coverage for 85% of the code.

6.2 Unit testing of output files

Our system will not use unit testing of output files. Our system does not have any output files to test and creating some for the sake of testing seems like an inefficient use of development time. Instead of creating and unit testing output files, we will supplement with a more rigorous unit testing plan described above.

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

SYMBOLIC CONSTANT	**MEANING**
FR-CI	Functional Requirement - Controller Input
FR-UI	Functional Requirement - User Interaction
FR-BI	Functional Requirement - Board Interactions
NFR-LFR	Nonfunctional Requirement - Look & Feel Requirement
NFR-UHR	Nonfunctional Requirement - Usability & Humanity Requirement
NFR-PR	Nonfunctional Requirement - Performance Requirement
NFR-OER	Nonfunctional Requirement - Operational & Environmental Requirement
NFR-MSR	Nonfunctional Requirement - Maintainability & Support Requirement
NFR-UHR	Nonfunctional Requirement - Security Requirement

7.2 Usability Survey Questions

- On a scale from 1-10, how would you rate the modernity and appeal of the game's graphics?
- Are you satisfied with the performance of the game on your computer?
- How long was your longest game session playing this game?
- Would you recommend Tetris Tussle to a friend?
- Did you enjoy playing the game?
- How does this version of Tetris align with your expectations?
- Were the graphical elements easy to see/read?

- On a scale from 1-10, how easy was this game to play?
- What improvements would you make to this game?
- Was the user interface intuitive in your opinion?