Verification and Validation Report: Software Eng

Team #11, Mac-AR
Student 1 Matthew Collard
Student 2 Sam Gorman
Student 3 Ethan Kannampuzha
Student 4 Kieran Gara

April 4, 2024

1 Revision History

Date	Version	Notes
March 6, 2024	1.0	Team worked all together on filling in all sections for revision 0 report

2 Symbols, Abbreviations and Acronyms

All symbols, abbreviations, and acronyms can be found in section 1.4 of the SRS.

Contents

List of Tables

List of Figures

This document outlines the results of implementing the verification and validation plan. Included is a summary of all manual and automated tests performed on the project, along with their outputs.

3 Functional Requirements Evaluation

3.1 Game Room Testing

The following section goes over tests related to starting a game, loading game assets, and determining/displaying puzzles.

Table 1 and 2 below demonstrates the functional requirements evaluation for the game room. The way the game room will be assessed will be through the use of manual testing.

Table 1: **Functional Requirements Evaluation Results for Game Room Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-CR1	Manual	User presses Host button.	User redirected to create game menu which has entries to set game room settings (game room name, game room capacity, game room password).	Same as expected	Pass
Test-CR2	Manual	Game room name inputted. The game room name will be a string consisting of letters, numbers, and ascii characters, and of character length greater than 3 and less than 64.	Game room with specified name is created in database.	Same as expected	Pass
Test-CR3	Manual	User tries to create a new game room without inputting a name for the game room	Game room is not created and error message tells user that game room name is not valid.	Same as expected	Pass
Test-CR4	Manual	User slides game room capacity slider to adjust game room capacity (min 2, max 10)	Game room with specified capacity is created in database.	Same as expected	Pass
Test-CR5	Manual	User inputs game room password. The password length will be between 8 and 64 characters, and consists of letters, number, and ASCII characters arranged in a random order.	Game room requiring password specified by user is created in database.	Same as expected	Pass
Test-CR6	Manual	User does not input anything for game room password, and hosts a game room.	Game room requiring no password is created in database.	Same as expected	Pass

Table 2: **Functional Requirements Evaluation Results for Game Room Testing 2**

Id	Type	Inputs	Expected Result	Actual Result	Result
Test-JR1	Manual	User presses Join Game button	User redirected to join room menu which lists all present game rooms in database that are available to be joined and are under capacity.	Same as expected	Pass
Test-JR2	Manual	User attempts to enter a game room at maximum capacity.	User is not able to join the game room.	Same as expected	Pass
Test-JR3	Manual	User enters room name of a created room that has no password set.	User joins game room and available capacity decreases by 1.	Same as expected	Pass
Test-JR4	Manual	User enters room name and password of created game room.	User joins game and available capacity decreases by 1.	Same as expected	Pass
Test-JR5	Manual	User enters room name and password of created game room	User is redirected to game room menu which displays the users present in the game room.	Same as expected	Pass
Test-RS1	Manual	User presses settings button, then processes to change the game room name, password, or capacity.	User is redirected to edit game settings menu which displays the current settings of the game room (ie. room capacity, and password).	Same as expected	Pass
Test-RS2	Manual	Password to game room is updated.	Game room is updated to require the new password.	Same as expected	Pass
Test-RS3	Manual	Game room settings are changed.	Game room settings menu is updated to show current setting values.	Same as expected	Pass
Test-ER1	Manual	Game room menu is present and user presses exit button.	User is removed from game room.	Same as expected	Pass
Test-ER2	Manual	User exits game room.	Game room menu updated to no longer display the user in the game room.	Same as expected	Pass
Test-ER3	Manual	User exits game room.	Total number of users in game room decreases by one.	Same as expected	Pass

3.2 Start Game Testing

The following section goes over tests related to starting a game, loading game assets, and determining/displaying puzzles.

Table 3: **Functional Requirements Evaluation Results for Start Game Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-ST1	Manual	User presses start button	Game is started	Same as expected	Pass
Test-ST2	Manual	User starts game	First puzzle GameObject is instantiated and loaded into the scene	Same as expected	Pass
Test-ST3	Automated	User starts game	Order of puzzles is determined and each user is assigned a part of the puzzle	Same as expected	Pass
Test-ST4	Automated	User starts game	Progress bar displayed on screen to let user know current progress	TBD	TBD
Test-ST5	Automated	User starts game	Puzzle is displayed to each user and game commences	Same as expected	Pass

3.3 Puzzle Interaction Testing

The following section goes over tests related to interacting with the puzzle UI elements and the corresponding updated on the back end.

Table 4: **Functional Requirements Evaluation Results for Puzzle Interaction Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-PI1	Manual	User Selects Puzzle	Puzzle Enlarges On UI	TBD	TBD
Test-PI2	Manual	User taps a key on the Combination Puzzle	If the digit was correct, the digit appears on the screen, if it was not, the combination puzzle turns red	Same as expected	Pass
Test-PI2	Manual	User presses a coloured button on the Simon Says Puzzle	If the colour pressed was the correct one, the system awaits the next button press. If it was not, the level is set to one, and the colour combination gets re-transmitted on the cube.	Same as expected	Pass
Test-PI2	Manual	User rotates their phone on the maze puzzle	The maze rotates in accordance with the phone rotation	Same as expected	Pass
Test-PI2	Manual	User performs drags a wire on the Wires Puzzle	The wire follows the drag, and locks onto a node if the user lets go of their finger over it	Same as expected	Pass
Test-PI2	Manual	User types on the input field of the Isometric Puzzle	The input field updates on all the user's screen. If the code was entered correctly, the puzzle is completed.	Same as expected	Pass
Test-PI3	Automated	User performs an action on a puzzle	Puzzle back end information is updated appropriately to action	Same as expected	Pass
Test-PI4	Manual	User performs an action on a puzzle	Puzzle UI updates appropriately for other members in game room	Same as expected	Pass
Test-PI5	Manual	Multiple users perform actions to complete puzzle	All users notified puzzle that has been completed and game room progresses	Same as expected	Pass
Test-PI6	Manual	Single user performs actions to complete puzzle	All users notified puzzle that has been completed and game room progresses	Same as expected	Pass
Test-PI7	Manual	User requests a hint	Hint for active puzzle is displayed	TBD	TBD
Test-PI8	Manual	User presses button to close hint	Hint display is removed	TBD	TBD
Test-PI9	Manual	User presses skip puzzle button	Game room progresses	Same as expected	Pass
Test-PI10	Manual	User selects skipped puzzle	Puzzle state is equal to the state it was prior	Same as expected	Pass
Test-PI11	Manual	User different than the one who skipped the puzzle selects the skipped puzzle	Puzzle state is equal to the state it was prior	Same as expected	Pass

3.4 Messaging Testing

The following section goes over tests related to sending and receiving messages.

Table 5: **Functional Requirements Evaluation Results for Messaging Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-MS1	Manual	User clicks messaging button	Messaging interface is displayed containing the messaging display window and input field	Same as expected	Pass
Test-MS2	Manual	User presses the ”X” button to close the messaging interface	The messaging interface is closed	Same as expected	Pass
Test-MS3	Manual	User taps the input field and types on the displayed keyboard to compose a message, presses the send button	Typed message appears in text input field, then once send button is pressed message is displayed in the message display port. The input field returns to the empty (default) state	Same as expected	Pass
Test-MS4	Manual	User sends a message (as in Test-MS3)	The UI of all the other users in the game room display a notification that a message has been received.	Same as expected	Pass
Test-MS5	Manual	User taps on the message button when there is a message received notification	The notification is removed. The received message is displayed in the display port	Same as expected	Pass
Test-MS6	Manual	Single user performs actions to complete puzzle	All users notified puzzle that has been completed and game room progresses	Same as expected	Pass

3.5 Individual Puzzle Testing

The following section goes over tests related to individual puzzle interactions.

Table 6: **Functional Requirements Evaluation Results for Messaging Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-FRMP1	Manual	Users start the maze puzzle	A random maze with an end goal and a ball is visible only to the user in control of viewing, and not the user in control of rotating the maze	Same as expected	Pass
Test-FRMP2	Manual	Users start the maze puzzle, user in control of rotating the maze rotates their phone	The maze visibly rotates on the controlling user's screen as well as the viewer's screen. The ball is observed to move in the direction of gravity by the viewing user	Same as expected	Pass
Test-FRMP3	Manual	Phone rotation by the controlling user, guiding the ball to the end goal	When the ball reaches the end goal, the puzzle is completed	Same as expected	Pass
Test-FRIP1	Manual	Users start the isometric puzzle	Each user is given randomly assigned letter and number combinations displayed on cubes.	Same as expected	Pass
Test-FRIP2	Manual	Users enter the correct code for the isometric puzzle	Each user observes the puzzle completing	Same as expected	Pass
Test-FRIP3	Manual	Users enter the incorrect code for the isometric puzzle	Each user sees the incorrect code on their input field, the puzzle does not complete	Same as expected	Pass

4 Nonfunctional Requirements Evaluation

4.1 Look and Feel

The following section goes over tests related to the non-functional look and feel requirements.

Table 7 below demonstrate the non-functional requirements evaluation for the look and feel criteria.

Table 7: **Non-Functional Requirements Evaluation Results for Look and Feel Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-LF1	Manual	Main menu is launched with simulated width and height of 1080 and 1920	x and y coordinates of menu elements are within the 1080 and 1920 bounds, allowing them to be visible and interacted with. This simulates the dimensions a Google Pixel would have.	Same as expected	Pass
Test-LF1	Manual	Main menu is launched with simulated width and height of 1440 and 2960	x and y coordinates of menu elements are within the 1440 and 2960 bounds, allowing them to be visible and interacted with. This simulates the dimensions a Samsung Galaxy S9 would have.	Same as expected	Pass
Test-LF1	Manual	Main menu is launched with simulated width and height of 1200 and 1920	x and y coordinates of menu elements are within the 1200 and 1920 bounds, allowing them to be visible and interacted with. This simulates he dimensions a Nexus 7 would have.	Same as expected	Pass
Test-LF1	Manual	Main menu is launched with simulated width and height of 1125 and 2436	x and y coordinates of menu elements are within the 1125 and 2436 bounds, allowing them to be visible and interacted with. This simulates he dimensions a iPhone X would have.	Same as expected	Pass
Test-LF2	Manual	Program launched on IPhone	All elements are clearly visible and able to be interacted with	App unable to be launched on IPhone due to stricter regulations on development builds and execution for IOS vs Android. IOS environment will be continued to be worked on throughout revision 1	Fail
Test-LF3	Manual	Program launched on Android	All elements are clearly visible and able to be interacted with	Same as expected	Pass
Test-LF4	Manual	Program launched in environment with harsh lighting	Most elements still clearly visible and able to be discerned by user	Same as expected	Pass

4.2 Usability

Results of usability survey can be found in the following table: Table 26

Table 7 below demonstrate the non-functional requirements evaluation for the Usability criteria.

Table 8: **Non-Functional Requirements Evaluation Results for usability Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-UH1	Manual	The user attempts to connect to a lobby with no internet connection.	A message appears to prompt the user to connect to the internet and no lobby is joined.	Same as expected	Pass
Test-UH2	Manual	The user loses connection to the internet while in the game room lobby	A message appears to prompt the user to connect to the internet.	Same as expected	Pass
Test-UH3	Manual	The user loses connection to the internet while in the game interacting with puzzles	A message appears to prompt the user to connect to the internet.	Same as expected	Pass
Test-UH4	Manual	Users below the age of 20 will install and use the app	The user determines that the app is easy to navigate	Same as expected (as shown through results of the usability survey)	Pass
Test-UH5	Manual	Users between the ages of 20 and 30 will install and use the app	The user determines that the app is easy to navigate	Same as expected (as shown through results of the usability survey)	Pass
Test-UH6	Manual	Users over the age of 30 will install and use the app	The user determines that the app is easy to navigate	TBD. No usability tests involving individuals over the age of 30 have been conducted yet. They will be conducted in the future	TBD
Test-UH7	Manual	The user disconnects from the internet then reconnect	The user should be prompted to rejoin the game after their connection is restore	TBD	TBD
Test-UH8	Manual	Users have completed playing the game	Average rating of usability survey is 3/5 or higher	Same as expected	Pass

4.3 Performance

The following section goes over tests related to the non-functional performance requirements.

Table 9 below demonstrate the non-functional requirements evaluation for the performance criteria.

Table 9: **Non-Functional Requirements Evaluation Results for Performance Testing**

Id	Type	Inputs	Expected Result	Actual Result	Result
Test-PR1	Manual	Request made to create lobby from main menu	Response time of system measured to be $\leq$ 5 seconds	Same as expected	Pass
Test-PR2	Manual	App launched between the hours of 6am and 6pm	All functionality is available	Same as expected. Occasional Vivox outages have occurred which have hindered voice chat support, but general reliability is found to be $> 95\%$, which is within bounds of tolerance	Pass
Test-PR3	Manual	App launched between the hours of 6pm and 6am	All functionality is available	Same as expected. Occasional Vivox outages have occurred which have hindered voice chat support, but general reliability is found to be $> 95\%$, which is within bounds of tolerance	Pass
Test-PR4	Manual	A request is made to join a lobby during a simulated server outage	A warning is generated, notifying the user of the server outage and the unavailable functionality	TBD	TBD

4.4 Maintainability and Support

The following section goes over tests related to the non-functional maintainability and support requirements.

Table 10 below demonstrate the non-functional requirements evaluation for the maintainability and support criteria.

Table 10: **Non-Functional Requirements Evaluation Results for Maintainability and Support Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-MT1	Manual	Static walk through of all documents	Requirements verified by code	Same as expected	Pass
Test-MT2	Manual	N/A	All code comments are verified by third party to be valid and understandable	Same as expected	Pass

4.5 Security

The following section goes over tests related to the non-functional security requirements.

Table 11 below demonstrate the non-functional requirements evaluation for the security criteria.

Table 11: **Non-Functional Requirements Evaluation Results for Security Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-SR1	Manual	External networking request for user IP address	No IP returned	Same as expected	Pass
Test-SR2	Manual	N/A	No data is shown to the user that is not relevant to their experience with the app	Same as expected	Pass

4.6 Cultural

The following section goes over tests related to the non-functional cultural requirements.

Table 12 below demonstrate the non-functional requirements evaluation for the cultural criteria.

Table 12: **Non-Functional Requirements Evaluation Results for Cultural Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-CU1	Manual	N/A	No offensive images, text, or sound are displayed or heard during playing the game	Same as expected	Pass
Test-CU2	Manual	N/A	All available text and audio clues are in Canadian English	Same as expected	Pass
Test-CU3	Manual	N/A	Static code inspection ensuring all assets are in Canadian English	Same as expected	Pass

4.7 Legal

The following section goes over tests related to the non-functional legal requirements.

Table 13 below demonstrate the non-functional requirements evaluation for the legal criteria.

Table 13: **Non-Functional Requirements Evaluation Results for Legal Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-LR1	Manual	N/A	Confirmation that all the assets in the game are either created by MacAR developers or are open source resources that are properly attributed to.	Same as expected	Pass

4.8 Health and Safety

The following section goes over tests related to the non-functional health and safety requirements.

Table 14 below demonstrate the non-functional requirements evaluation for the health and safety criteria.

Table 14: **Non-Functional Requirements Evaluation Results for Health and Safety Testing**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-HS1	Manual	Game Started	The system will prompt the user to check their environment and make sure its suitable for playing the game	Same as expected	Pass

5 Unit Testing

5.1 Puzzle Manager

Table 15 below demonstrates the functional requirements evaluation for the Multiplayer Puzzle Manager. The way the Multiplayer Puzzle Manager will be assessed will be through the use of unit tests. These tests will cover the parts of the Multiplayer Puzzle Manager which can be verified through automation.

Table 15: **Multiplayer Puzzle Manager Unit Test Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-PM1	Automated	Simple serialization of client data	Data is serialized into the correct bytes	Same as expected	Pass
Test-PM2	Automated	Complex serialization of client data	Data is serialized into the correct bytes	Same as expected	Pass
Test-PM3	Automated	Simple deserialization of client data	Data is deserialized into the correct bytes	Same as expected	Pass
Test-PM4	Automated	Complex deserialization of client data	Data is deserialized into the correct bytes	Same as expected	Pass
Test-PM5	Automated	Client data undergoes full serialization - deserialization cycle	Initial data is retrieved on client side	Same as expected	Pass

5.2 Maze Puzzle Testing

Table 16 below demonstrates the functional requirements evaluation for the maze puzzle. The way the maze puzzle will be assessed will be through the use of unit tests. These tests will cover the parts of the maze puzzle which can be verified through automation.

Table 16: **MazePuzzle Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-MP1	Automated	Spawn in the maze puzzle using NUnit unit testing	The maze puzzle is successfully spawned in, and mazePuzzle.activeSelf is true.	Same as expected	Pass
Test-MP2	Automated	Instantiate maze puzzle at (0,0,0) with rotation (0,0,0)	The maze puzzle object is not null and is available in the scene	Same as expected	Pass
Test-MP3-1	Automated	A two dimensional array of size 3x3 input into the To1DArray function	The output 1-D array is the same as the 2D array arranged row by row	Same as expected	Pass
Test-MP3-2	Automated	A two dimensional array of size 5x4 input into the To1DArray function	The output 1-D array is not the same as the 2D array arranged column by column	Same as expected	Pass
Test-MP4-1	Automated	A one dimensional array of size 9 is input with size parameters 3 for width and 3 for length into the Make2DArray function	The output 2D array is the same as arranging the 1D array into a 2D array of size 3x3 by rows	Same as expected	Pass
Test-MP4-2	Automated	A one dimensional array of size 20 is input with size parameters 5 for width and 4 for length into the Make2DArray function	The output 2D array is not the same as arranging the 1D array into a 2D array of size 5x4 by columns	Same as expected	Pass
Test-MP5	Automated	Two adjacent maze cube are input into the ClearWalls function. Cube 1 is spawned at (1,0,0) and cube 2 is spawned at (0,0,0)	The Left wall of the first cube and the Right wall of the second cube are both invisible	Same as expected	Pass
Test-MP6	Automated	A random array of size 100 with elements between 0 and 3 input into the convertLayoutToGrid function	A Maze that matches the removed elements	Same as expected	Pass
Test-MP7	Automated	A completely filled maze is massed into the GetUnvisitedCells function	An empty list of maze cells is returned	Same as expected	Pass

5.3 Isometric Puzzle Testing

Table 17 below demonstrates the functional requirements evaluation for the Isometric puzzle. The way the Isometric puzzle will be assessed will be through the use of unit tests. These tests will cover the parts of the Isometric puzzle which can be verified through automation.

Table 17: **Isometric Puzzle Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-IP1	Automated	Spawn in the Isometric puzzle using NUnit unit testing	The Isometric puzzle is successfully spawned in, and IsometricPuzzle.activeSelf is true.	Same as expected	Pass
Test-IP2	Automated	Instantiate Isometric puzzle at (0,0,0) with rotation (0,0,0)	The Isometric puzzle object is not null and is available in the scene	Same as expected	Pass
Test-IP3	Automated	Call isometric puzzle InitializePuzzle without first setting up a multiplayer environment	Null Reference Exception	Same as expected	Pass
Test-IP4	Automated	Call SetCubeLayoutSet function of the Isometric Puzzle class with parameter ”1T” passed in	Cube layout changes to match the input combination ”1T	Same as expected	Pass

5.4 Simon Says Puzzle Testing

Table 18 below demonstrates the functional requirements evaluation for the Simon Says puzzle. The way the Simon Says puzzle will be assessed will be through the use of unit tests. These tests will cover the parts of the Simon Says puzzle which can be verified through automation.

Table 18: **Simon Says Unit Test Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-SP1	Automated	Spawn in Simon Says Puzzle	Simon Says Puzzle successfully spawns in, and attribute activeSelf is true.	Same as expected	Pass
Test-SP2	Automated	Spawn in Simon Says Puzzle	Simon Says Puzzle object is not null	Same as expected	Pass
Test-SP3	Automated	Increment Level function called on Simon Says Puzzle	Simon Says Puzzle level increases by 1 (from 1 to 2)	Same as expected	Pass
Test-SP4	Automated	Colour sequence list contains blue. TrackUserInput(blue) is called which represents user pressing blue button	Colour sequence list and Player Sequence list are equal	Same as expected	Pass
Test-SP5	Automated	Player input list contains colour values. BeginSimonSays() function is called.	Player input list is cleared	Same as expected	Pass
Test-SP6	Automated	Colour sequence list contains colour values. BeginSimonSays() function is called.	Colour sequence list is cleared	Same as expected	Pass

5.5 Wires Puzzle Testing

Table 19 below demonstrates the functional requirements evaluation for the Wires puzzle. The way the Wires puzzle will be assessed will be through the use of unit tests. These tests will cover the parts of the Wires puzzle which can be verified through automation.

Table 19: **Wires Puzzle Unit Test Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-WI1	Automated	Puzzle is initialized with trivial sequence	Wire/ Node sequence is set internally	Same as expected	Pass
Test-WI2	Automated	Puzzle is initialized with complex sequence	Wire/ Node sequence is set internally	Same as expected	Pass
Test-WI3	Automated	Puzzle is left uninitialized	Wire/ Node sequence is set to impossible to achieve values	Same as expected	Pass
Test-WI4	Automated	User connects wire	Wire/ Node sequence is properly updated	Same as expected	Pass
Test-WI5	Automated	User creates incorrect sequnce	Wire/ Node sequence is updated but puzzle is not marked as complete	Same as expected	Pass

5.6 Combination Puzzle Testing

Table 18 below demonstrates the functional requirements evaluation for the Combination puzzle. The way the Combination puzzle will be assessed will be through the use of unit tests. These tests will cover the parts of the Combination puzzle which can be verified through automation.

Table 20: **Combination Puzzle Unit Test Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-CP1	Automated	Spawn in Combination Puzzle	Combination Puzzle successfully spawns in, and attribute activeSelf is true.	Same as expected	Pass
Test-CP2	Automated	Spawn in Combination Puzzle	Combination Puzzle object is not null	Same as expected	Pass
Test-CP3	Automated	KeyPadPress(”3”) is called to simulate a correct input	The first character of the current code is returned as ”3”	Same as expected	Pass
Test-CP4	Automated	KeyPadPress(”4”) is called to simulate an incorrect input	The first character of the current code is returned as ”_”	Same as expected	Pass
Test-CP5	Automated	KeyPadPress is used to simulate entry of the entire correct code	completePuzzle is true	Same as expected	Pass
Test-CP6	Automated	The puzzle is initialized	The text of the instruction page is not still the default instruction page text	Same as expected	Pass

6 Changes Due to Testing

Currently, our group has done one usability testing session, however, there are plans to do more in the future. As a result, we will most likely receive more feedback and make changes based on this.

Source	**Feedback**	**Changes**
Usability Testing	Time to play the game is <30 minutes	Our group has decided to remove the Save/Load game requirements after usability testing, as the game only takes 15-30 minutes to complete, and doesn't really need a save game option.
Usability Testing	Time to play the game is <30 minutes	Since game length is max 30 minutes, our group has decided to have only 5 puzzles since this will keep the game within the time limit
Usability Survey and TA meeting	Calibration step not necessary	Removed FR-ST3 since there is no need for a calibration setup step for our technology to be used.
Usability Testing	Difficulty creating lobby as no notification was given for password not meeting requirements	Added pop-up notifications to give information to users (such as if password is not long enough)
Usability Testing	Join lobby button had issues as when pressed multiple times consecutively, users were unable to join the game room	Disabled join lobby button once it has been pressed to prevent consecutive server calls
Dr. Yuan Feedback	Isometric Puzzle spawns too high, resulting in users being unable to see letter/number on top of the puzzle	Instantiation of the Isometric Puzzle prefab adjusted to spawn in a lower position, and size of puzzle decreased

7 Automated Testing

The automated test suite is, due to limitations with Unity, split into two parts. Both parts, the Edit Mode and Play Mode test suites, utilize NUnit to create and run the tests.

The Play Mode tests use Unity's InputSystem library to simulate user inputs (clicks, rotations, etc.), and consist of the majority of tests that simulate user interaction or multiplayer aspects.

The Edit Mode tests are more basic unit tests that test individual script functions. These tests are run on the CI/CD pipeline. In addition to the tests that the Mac-AR team has created, certain unity packages come with built in unit tests that are run alongside this suite. These additional tests currently number 682 in total.

8 Trace to Requirements

	CG1	**CG2**	**CG3**	**CG4**	**CG5**	**JG1**	**JG2**	**JG3**	**JG4**	**JG5**	**RS1**	**RS2**	**RS3**	**RS4**	**RS5**	**RS6**	**ER1**	**ER2**	**ER3**	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
Test-CR1	X																								
Test-CR2	X	X			X																				
Test-CR3	X		X		X																				
Test-CR4	X		X		X																				
Test-CR5	X			X	X	X	X																		
Test-CR6	X			X	X																				
Test-JR1						X				X															
Test-JR2						X																			
Test-JR3						X		X																	
Test-JR4						X			X																
Test-JR5						X																			
Test-RS1											X	X													
Test-RS2											X	X	X												
Test-RS3											X	X		X											
Test-RS4											X	X			X										
Test-RS5											X	X				X									
Test-ER1																	X								
Test-ER2																		X	X						
Test-ER3																			X						
Test-ST1																				X					
Test-ST2																				X	X				
Test-ST3																				X			X		
Test-ST4																				X				X	
Test-ST5																				X	X		X		X
Test-PI5																							X		

Table 21: Functional Traceability Matrix Pt. 1

	PI1	**PI2**	**PI3**	**PI4**	**PI5**	**PI6**	**PI7**	**HR1**	**HR2**	**HR3**	**SP1**	**SP2**	**SP3**	**SM1**	**SM2**	**SM3**	**SM4**	**SM5**	**RM1**	**RM2**	**RM3**
Test-PI1	X	X																			
Test-PI2	X	X	X	X																	
Test-PI3	X		X																		
Test-PI4				X	X																
Test-PI5	X	X	X	X	X	X	X														
Test-PI6	X	X	X	X	X	X	X														
Test-PI7								X	X												
Test-PI8										X											
Test-PI9	X										X										
Test-PI10	X										X	X									
Test-PI11	X				X						X	X									
Test-PI12											X	X	X								
Test-MS1														X		X					
Test-MS2															X		X				X
Test-MS3															X	X		X			
Test-MS4																X			X		
Test-MS5																X			X	X	
Test-PM1	X					X	X														
Test-PM2	X					X	X														
Test-PM3	X					X	X														
Test-PM4	X					X	X														
Test-PM5	X					X	X														
Test-WI1			X	X	X																
Test-WI2			X	X	X																
Test-WI3			X	X	X																
Test-WI4			X	X	X																
Test-WI5			X	X	X																
Test-MP1			X	X	X																
Test-MP2			X	X	X																
Test-MP3			X	X	X																
Test-MP4			X	X	X																
Test-MP5			X	X	X																
Test-MP6			X	X	X																
Test-MP7			X	X	X																
Test-SP1			X	X	X																
Test-SP2			X	X	X																
Test-SP3			X	X	X																
Test-SP4			X	X	X																
Test-SP5			X	X	X																
Test-SP6			X	X	X																
Test-IP1			X	X	X																
Test-IP2			X	X	X																
Test-IP3			X	X	X																
Test-IP4			X	X	X																
Test-CP1			X	X	X																
Test-CP2			X	X	X																
Test-CP3			X	X	X																
Test-CP4			X	X	X																
Test-CP5			X	X	X																
Test-CP6			X	X	X																

Table 22: Functional Traceability Matrix Pt. 2

	LF1	LF2	UH1	UH2	UH3	UH4	UH5	UH6	UH7	UH8	PR1	PR2	OE1	MS1	MS2	SR1	SR2	CR1	CR2	LR1	HS1
Test-LF1	X																				
Test-LF2	X	X	X										X								
Test-LF3	X	X	X										X								
Test-LF4		X			X																
Test-UH1				X																	
Test-UH2				X		X															
Test-UH3				X		X															
Test-UH4					X																
Test-UH5					X																
Test-UH6					X			X													
Test-UH7							X		X												
Test-UH8										X											
Test-PR1											X										
Test-PR2												X									
Test-PR3												X									
Test-PR4												X									
Test-MT1														X	X						
Test-MT2														X	X						
Test-SR1																X					
Test-SR2																	X				
Test-CU1																		X			
Test-CU2																		X	X		
Test-CU3																			X		
Test-LR1																				X	
Test-HS1																					X

Table 23: Non-Functional Traceability Matrix

9 Trace to Modules

Module descriptions can be found in section 5 of the MG. Module M10 (Math Puzzle module) will not be implemented, and therefore does not have any traceability to tests. In the future, documents will be revised to update module numbering.

	M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**	**M11**	**M12**	**M13**	**M14**
Test-CR1		X												
Test-CR2		X												
Test-CR3		X												
Test-CR4		X												
Test-CR5		X												
Test-CR6		X												
Test-JR1		X												
Test-JR2		X												
Test-JR3		X												
Test-JR4		X												
Test-RS1		X												
Test-RS2		X												
Test-RS3		X												
Test-ER1		X												
Test-ER2		X												
Test-ER3		X												
Test-ST1		X												
Test-ST2		X			X									
Test-ST3					X									
Test-ST4					X									
Test-ST5					X									
Test-PI1					X									
Test-PI2					X									
Test-PI3					X									
Test-PI4					X									
Test-PI5					X									
Test-PI6					X									
Test-PI7					X									
Test-PI8					X									
Test-PI9					X									
Test-PI10					X									
Test-PI11					X									
Test-PI12					X									
Test-MS1			X											
Test-MS2			X											
Test-MS3			X											
Test-MS4			X											
Test-MS5			X											
Test-MS6			X											
Test-PM1					X									
Test-PM2					X									

Table 24: Module Traceability Matrix

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
Test-PM3					X									
Test-PM4					X									
Test-PM5					X									
Test-MP1									X					
Test-MP2									X					
Test-MP3									X					
Test-MP4									X					
Test-MP5									X					
Test-MP6									X					
Test-MP7									X					
Test-SP1						X								
Test-SP2						X								
Test-SP3						X								
Test-SP4						X								
Test-SP5						X								
Test-SP6						X								
Test-WI1								X						
Test-WI2								X						
Test-WI3								X						
Test-WI4								X						
Test-WI5								X						
Test-IP1							X							
Test-IP2							X							
Test-IP3							X							
Test-IP4							X							
Test-CP1											X			
Test-CP2											X			
Test-CP3											X			
Test-CP4											X			
Test-CP5											X			
Test-CP6											X			
Test-LF1		X												
Test-LF2		X			X									
Test-LF3		X			X									
Test-LF4		X			X									
Test-UH1		X											X	
Test-UH2		X											X	
Test-UH3		X											X	
Test-UH4		X												
Test-UH5		X												
Test-UH6		X												
Test-UH7		X												
Test-UH8		X												
Test-PR1		X										X		
Test-PR2		X	X	X								X		
Test-PR3		X	X	X								X		
Test-PR4		X	X	X								X	X	
Test-MT1														X
Test-MT2														X
Test-SR1												X		
Test-SR2												X		
Test-CU1														X
Test-CU2														X
Test-CU3														X
Test-LR1														X
Test-HS1		X											X	

Table 25: Module Traceability Matrix Pt. 2

10 Code Coverage Metrics

Our project did not have any strict requirements on meeting a particular code coverage percentage. Furthermore, lots of manual testing was done due to the use of the Unity framework.

Module	Branch Coverage	Line Coverage
Multiplayer Puzzle	11%	33%
Wires	45%	63%
Maze	23%	30%
Isometric	60%	67%
Simon Says	27%	40%
Combination	20%	65%

References

11 Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection. Please answer the following question:

1. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

The main difference between the original plan and how the validation and verification was conducted was the amount of tests that we thought we could automate. All of the members of the teams are very familiar with unit testing and therefore a large part of the VnV plan revolved around creating a comprehensive testing suite that could be run before every new feature to ensure correctness. What we failed to account for was the how strict Unity's environment was, and therefore how much work would have to be shifted to manual verification. This was fully a product of our inexperience with the program, and on future projects it is likely that we would be able to create a more accurate plan from the start.

11.1 Usability Survey

Table 26 below showing the results of the Usability survey

Table 26: **Results of Usability Survey**

Question	**Average Rating (out of 5)**
I find the controls of the game to be straightforward	3.25
I find the game's interface to be easy to navigate	3.25
I feel detached from the outside world while playing the game	3
I do not care to check events that are happening in the real world during the game	3
I think the game is fun	4.25
I find the game supports social interaction between players	5
I enjoy playing this game with other users	4.5
I feel the game allows me to be imaginative	4
I feel like the puzzles are challenging	4
I feel like the puzzle interactions are fun	4.25
I enjoy the game's graphics	2.75
I think the game is visually appealing	3.25