Verification and Validation Report: Farming Matters

Team #14, The Farmers
Brandon Duong
Andrew Balmakund
Mihail Serafimovski
Mohammad Harun
Namit Chopra

April 5, 2023

1 Revision History

Date	Version	Notes
06/03/2023	1.0	Recorded the results of some functional and non-functional tests
08/03/2023	1.0	Finished first version
03/04/2023	1.1	Addressed GitHub issues
05/04/2023	2.0	Finished second version

Contents

1 Revision History **i**

2 Functional Requirements Evaluation **1**
- 2.1 Account Testing 1
- 2.2 Game Mechanics Testing 2
- 2.3 Database Testing 4

3 Nonfunctional Requirements Evaluation **6**
- 3.1 Look and Feel 6
- 3.2 Usability and Humanity 7
- 3.3 Performance 8
- 3.4 Operational and Environmental 9
- 3.5 Maintainability and Support Requirements 9
- 3.6 Security 10
- 3.7 Access 11
- 3.8 Integrity 12

4 Unit Testing **13**
- 4.1 Game Mechanics Testing 13

5 Changes Due to Testing **18**

6 Automated Testing **19**

7 Trace to Requirements **19**

8 Trace to Modules **22**

9 Code Coverage Metrics **24**

10 Appendix **26**
- 10.1 Symbolic Parameters 26
- 10.2 Usability Survey Questions 26

List of Tables

1 Functional Requirements Evaluation Results for Account Testing 2

2 Functional Requirements Evaluation Results for Game Mechanics Testing 3

3 Cont. Functional Requirements Evaluation Results for Game Mechanics Testing 4

4 Functional Requirements Evaluation Results for Database Testing 5

5 Non-Functional Requirements Evaluation Results for Look and Feel 7

6 Non-Functional Requirements Evaluation Results for Usability and Humanity 8

7 Non-Functional Requirements Evaluation Results for Performance 8

8 Non-Functional Requirements Evaluation Results for Operational and Environmental 9

9 Non-Functional Requirements Evaluation Results for Maintainability and Support Requirements 10

10 Non-Functional Requirements Evaluation Results for Security 11

11 Non-Functional Requirements Evaluation Results for Access 12

12 Non-Functional Requirements Evaluation Results for Integrity 13

13 **Functional Requirements Evaluation Results** 14
14 **Functional Requirements Evaluation Results** 15
15 **Functional Requirements Evaluation Results** 16
16 **Functional Requirements Evaluation Results** 17
17 Traceability Matrix for Functional Test Cases 20
18 Traceability Matrix for Non-functional Test Cases 21
19 Trace Between Test cases and Modules 22
20 Cont. Trace Between Test cases and Modules 23
21 Cont. Trace Between Test cases and Modules 24
22 **Symbolic Parameter Table** 26

List of Figures

1 Test Coverage for Unit Testing Inventory, Game Logic, and Events 25
2 Test Coverage for Testing Accounts and Authentication 25

This document describes the test results of the verification and validation (VnV) plan for Farming Matters. The VnV plan was continuously updated as the project evolved. The following document records the results of the current version of the VnV plan. It provides results of functional and nonfunctional requirements tests, unit tests, changes that will be implemented in the system as a result of the tests, and various traceability tables.

2 Functional Requirements Evaluation

The following section outlines the results of functional testing. The process and test performed follow the VnV Plan. To summarize, all the functional tests passed, indicating that all the functional requirements in the Software Requirements Specification (SRS) document are covered.

2.1 Account Testing

Table 1 below demonstrates the functional requirements evaluation for account testing. The system requires you to have an account to play the game. The test cases include the following: creating an account, resetting the password, logging in, verifying if the user is human and account deletion

Table 1: **Functional Requirements Evaluation Results for Account Testing**

Id	Type	Description	Expected Result	Actual Result	Result
Test-AC1	Manual	User is not logged in and submits a correct email and password on the login form	User is logged in and can access game	Same as expected	Pass
Test-AC2	Manual	User is not logged in and submits an incorrect email or password on the login form	User login is denied	Same as expected	Pass
Test-AC3	Manual	User is not logged in and goes through 'reset password' flow	User is able to access password reset functionality through a link sent to email	Same as expected	Pass
Test-AC4	Manual	User is not logged in and tries to login with correct credentials	Access is granted if the user is able to pass a human verification step (eg. captcha) and denied if they fail	Same as expected	Pass
Test-AC5	Manual	User has an account and requests to delete it	User account and all user data is deleted	Same as expected	Pass

2.2 Game Mechanics Testing

Table 2 and Table 3 below demonstrate the functional requirements evaluation for game mechanic testing. The game mechanics compose of all the game logic and consist of any actions the player is able to perform in the context of the game. Examples of these test cases include successfully buying and selling crops.

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

Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-GM1	Manual	User has over USER_MONEY and purchases a new land tile that costs USER_MONEY	User owns tile and USER_MONEY is subtracted from their total	Same as expected	Pass
Test-GM1	Manual	User does not have over USER_MONEY and tries to purchase a new land tile that costs USER_MONEY	User does not own tile and no money is subtracted from their total	Same as expected	Pass
Test-GM2	Manual	User tries to plant a seed that is in season on an owned land tile	Seed is removed from inventory and planted on land tile	Same as expected	Pass
Test-GM2	Manual	User tries to plant a seed that is not in season on an owned land tile	Seed is not removed from inventory and not planted	Same as expected	Pass
Test-GM3	Manual	User has at least 1 seed planted on a land tile and ends turn	All planted seeds grow by 1 turn	Same as expected	Pass
Test-GM4	Manual	User has at least 1 seed planted on a land tile, has at least 1 fertilizer in their inventory, and tries to use fertilizer on a planted seed	Planted seed grows by 1 turn and 1 fertilizer is removed from inventory	Same as expected	Pass
Test-GM5	Manual	User purchases a new land tile	User owns tile	Same as expected	Pass
Test-GM6	Manual	User just created a new account and is playing for the first time	The consultant prompt automatically appears with an option to purchase for a consultants advice	Same as expected	Pass
Test-GM6	Manual	User is currently in Fall and switches to the next season which is Winter	The consultant prompt automatically appears with an option to purchase for a consultants advice or not	Same as expected	Pass

Table 3: **Cont. Functional Requirements Evaluation Results for Game Mechanics Testing**

Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-GM7	Manual	User purchases consultant advice	The consultant type of advice is consistent throughout different seasons (each season you purchase a new consultant advice). Each user is either assigned a deterministic or probabilistic prompt type, upon creating a new account.	Same as expected	Pass
Test-GM7	Manual	User purchases consultant advice	The consultant advice is the same till a new season changes (i.e for 3 turns, the consultant advice does not change)	Same as expected	Pass
Test-GM8	Manual	User purchases a crop and is able to purchase insurance button	user sees insurance button after purchasing a crop	Same as expected	Pass
Test-GM9	Manual	Switching to several seasons to see that when a random event occurs	The random event occurs in the correct season which is visually observed by the change in the visual game environment	Same as expected	Pass
Test-GM10	Manual	User is in Fall and ends 3 turns	Season changes to Winter	Same as expected	Pass
Test-GM10	Manual	User is in Winter and ends 3 turns	Season changes to Spring	Same as expected	Pass
Test-GM10	Manual	User is in Spring and ends 3 turns	Season changes to Summer	Same as expected	Pass
Test-GM10	Manual	User is in Summer and ends 3 turns	Season changes to Fall	Same as expected	Pass

2.3 Database Testing

Table 4 below demonstrates the functional requirements evaluation for database testing. The database is critical as it will log decisions and the game state. The test cases ensure that the game decisions and game state will be successfully logged.

Table 4: **Functional Requirements Evaluation Results for Database Testing**

Test Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-DB1	Manual	The users play three turns (one season) of the game trying to maximize profit	All user actions are logged in the corresponding logging table with the most recent action as the last entry	Same as expected	Pass
Test-DB1	Manual	The users play three turns (1 season) without performing any actions	No actions are logged in the corresponding logging table	Same as expected	Pass
Test-DB2	Manual	Users play three turns of the game trying to maximize profit.	The game state table contains the most recent game state in the corresponding account.	Same as expected	Pass
Test-DB2	Manual	Users play three turns of the game trying to maximize profit and log out of the game. The user logs into the game using their account credentials.	The game saves the state prior to logout in the game state table to the corresponding account entry.	Same as expected	Pass
Test-DB3	Manual	Users log into the game for the first time and logs out without performing any actions. The user logs back into the game using their account credentials	The game loads in the new game state	Same as expected	Pass
Test-DB3	Manual	Users play three turns of the game trying to maximize profit and log out of the game. The user logs into the game using their account credentials	The game loads in the same state as prior to logout	Same as expected	Pass

3 Nonfunctional Requirements Evaluation

The section below summarizes the testing for the non-functional requirements. The tests for look and feel were crucial as the game being engaging was one of the main goals that were specified at the beginning of the project. The usability and humanity tests were also significant as the experiment should be relatively easy to conduct and the participants should have no issues understanding the game. Performance, maintainability, and support testing were also important as they could lead to a more immersive experience. Security, access, and integrity testing were performed to ensure the data derived from the experiment was valid.

3.1 Look and Feel

Table 5 below demonstrates the non-functional requirements evaluation for the look and feel testing. These tests will be an assessment of the user's overall experience such as if they find the user interface cluttered, the color theme of the system is coherent, and engaging graphics and audio.

Table 5: **Non-Functional Requirements Evaluation Results for Look and Feel**

Test Id	Type	Description	Expected Result	Actual Result	Result
Test-LF1	Manual	Survey was provided to the supervisor, Dr. Yiannakoulias, as well as 10 other people. These individuals are the target demographic: 18 years and older	Every aspect under the 'Look and Feel' has an average rating greater than or equal to EXPECTED_SURVEY_RATING	Ratings (rounded to 1 decimal point): • Minimalistic Design: 4.7 • Consistent Colour Theme: 4.1 • Engaging Audio: 4 • Engaging Graphics: 4.9	Pass
Test-LF2	Manual	The game will be played on different SCREEN_RESOLUTION	All visual elements are visible and accessible to the user	Same as expected	Pass
Test-LF3	Manual	Asked friends to play and recorded the number of turns they played	Average number of turns played to be over or equal to 12	Same as expected	Pass

3.2 Usability and Humanity

Table 6 below demonstrates the non-functional requirements evaluation for usability and humanity testing. This test will focus on the usability of the system by providing a survey to 11 users including the client. The results for the section "Easy to Understand" are averaged to determine how usable the system is on a scale of 1 to 5. If the rating is below 3, the UI must be improved to enhance the usability of the system.

Table 6: **Non-Functional Requirements Evaluation Results for Usability and Humanity**

Test Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-UH1	Manual	Survey was provided to the supervisor, Dr. Yiannakoulias, as well as 10 other people. These individuals are the target demographic: 18 years and older	The average rating for "Easy to Understand' is greater than or equal to EXPECTED_SURVEY_RATING	The average rating for ease of understanding was approximately 4.3	Pass

3.3 Performance

Table 7 below demonstrates the non-functional requirements evaluation for performance testing. These test will be an assessment of how will the system performs in certain areas. Mainly focusing on the load time and response time to a user's input.

Table 7: **Non-Functional Requirements Evaluation Results for Performance**

Test Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-P1	Manual	The user enters their account credentials and presses the login button	The system logs the user and displays the game within MAX_RESPONSE_TIME	Same as expected	Pass
Test-P2	Manual	The user plants a crop and harvests once ready	The game updates the interface (removing crop from and adding to inventory) within MAX_INTERFACE_UPDATE_TIME	Same as expected	Pass
Test-P3	Manual	The 3D models for pumpkin are replaced by a different one. The user buys a pumpkin seed and plants it. The crops are harvested by the user once ready	The game updates the interface for all steps (buying, planting, harvesting) within MAX_INTERFACE_UPDATE_TIME	Same as expected	Pass

3.4 Operational and Environmental

Table 8 below demonstrates the non-functional requirements evaluation for operational and environmental testing. These tests will be an assessment of how well the system can be run on different web browsers as well as different supported versions of each browser.

Table 8: **Non-Functional Requirements Evaluation Results for Operational and Environmental**

Test Id	Type	Description	Expected Result	Actual Result	Result
Test-OE1	Manual	The game will be played on different mainstream web browsers such Chrome, Firefox, Edge, Safari, and Brave	The game will be able to be played among different browsers	Same as expected	Pass
Test-OE2	Manual	The game will be played on different versions of a given browser for multiple browsers	Given that the game uses ReactJS which is only supported by web browsers that are ECMAScript 6 (also known as ES6) and above compliant. From a recent web version search, Versions of Chrome 34-113, Edge 12-110, Safari 9-16.3, and Firefox 32-112 are supported (Source)	Same as expected, not tested for all version (mostly the last 3 version of each web browser)	Pass

3.5 Maintainability and Support Requirements

Table 9 below demonstrates the non-functional requirements evaluation for maintainability and support testing. The test will be an assessment of how changing the game audio will impact the user's experience when playing the game.

Table 9: **Non-Functional Requirements Evaluation Results for Maintainability and Support Requirements**

Test Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-MS1	Manual	The user will switch the game audio to a different track and monitor the effect on the game after the change	The game still runs smoothly	Same as expected	Pass

3.6 Security

Table 10 below demonstrates the non-functional requirements evaluation for security. The test will be an assessment of how secure the system is. Mainly focusing on areas that can cause an impact on performance from an external source(e.g. automated scripts) and unauthorized account access.

Table 10: **Non-Functional Requirements Evaluation Results for Security**

Test Id	Type	Description	Expected Result	Actual Result	Result
Test-SR1	Manual	Use common exploit vulnerabilities with the current versions or past versions of the technology used within the system that can lead to automated attacks.	The system shall prevent the creation of accounts in suspicion of automated attacked	Same as expected	Pass
Test-SR2	Manual	Basic cross-site scripting (XSS) tech- niques will be used to see if the user is able to bypass the login screen. Some techniques include persistent, non-persistent, and DOM-based cross-site scripting	The system shall prevent the user from logging in through a malicious attack or from damaging the user database	Same as expected	Pass
Test-SR3	Manual	Manually query the passwords in the database and check to see if the user password is returned in plaintex	All user passwords are encrypted	Same as expected	Pass

3.7 Access

Table 11 below demonstrates the non-functional requirements evaluation for access. The tests will mainly focus on testing the access requirements that were mentioned above. Most of this pertains to what the user will be able to view and interact with when in different stages of the application.

Table 11: **Non-Functional Requirements Evaluation Results for Access**

Test Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-ACR1	Manual	The game was played at three locations. In two of those locations, the machine was connected to a private network. The machine was on a public network in a third location.	The user is able to access and play the game at all three locations	Same as expected	Pass
Test-ACR2	Manual	The user is initially not logged in and enters a URL that requires user credentials	User is redirected to login page	Same as expected	Pass
Test-ACR3	Manual	The user is initially logged in and enters a URL that requires user credentials	User is not redirected to the login page	Same as expected	Pass
Test-ACR4	Manual	The user is initially logged in and tries to log in on a separate computer	The new login is blocked	Same as expected	Pass

3.8 Integrity

Table 12 below demonstrates the non-functional requirements evaluation for integrity. The test will be an assessment for ensuring the user is able to retrieve their saved game state after signing back in and covering a case where users unexpectedly lose connection.

Table 12: **Non-Functional Requirements Evaluation Results for Integrity**

Test Id	**Type**	**Description**	**Expected Result**	**Actual Result**	**Result**
Test-IR2	Manual	The user is logged into the system and performs actions for one turn. The user logs out and logs back in	The database returns the game state information within MAX_RESPONSE_TIME	Same as expected	Pass
Test-IR3	Manual	The user is logged into the system and performs actions for one turn. The user losses an internet connection before logging out	The game state table contains the latest changes before the connection loss for the corresponding account	Same as expected	Pass

4 Unit Testing

The section below summarizes the testing for unit testing. The areas used for automation testing were the game mechanics of the system as this is crucial to ensure that over time as the internal game mechanics changes, the base and extreme cases still need to be verified.

4.1 Game Mechanics Testing

Table 13, Table 14, Table 15 and Table 16 below demonstrate the non-functional requirements evaluation for the game mechanics. The way the game mechanics will be assessed will be through the use of unit tests. These tests will cover the parts of the game mechanics which can be verified through automation.

Table 13: **Functional Requirements Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-GM11	Automated	User does not have an item in their inventory, User has USER_MONEY and purchases a single item that costs ITEM_MONEY and it is not in their inventory	User obtains purchased item	Same as expected	Pass
Test-GM11	Automated	User does not have any items in their inventory, User has USER_MONEY and purchases a single item that costs ITEM_MONEY and it is not in their inventory	ITEM-_MONEY is subtracted from their balance which is now USER_MONEY - ITEM_MONEY	Same as expected	Pass
Test-GM12	Automated	User does not have any items in their inventory, has a balance of $0 and purchases an item of ITEM_MONEY	User current balance remains the same of being $0	Same as expected	Pass
Test-GM12	Automated	User does not have any items in their inventory, has a balance of $0 and purchases an item of ITEM_MONEY	The chosen item to be purchased is not added to the user's inventory	Same as expected	Pass
Test-GM13	Automated	User has 1 item in their inventory, has a balance of USER_MONEY and purchases two items of ITEM_MONEY	The quantity of that same item purchased again is now increased by one which is now 2	Same as expected	Pass
Test-GM13	Automated	User has 1 item in their inventory, has a balance of USER_MONEY, and purchases two items of ITEM_MONEY	The user balance has decreased by ITEM_MONEY and now has a balance of USER_ MONEY - ITEM_MONEY	Same as expected	Pass

Table 14: **Functional Requirements Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-GM14	Automated	User has 1 crop in their inventory to sell and sold in the different seasons, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The item they want to sell is in their inventory	Same as expected	Pass
Test-GM14	Automated	User has 1 crop in their inventory to sell and sold in the different seasons, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The user balance increased by ITEM_MONEY and their balance is now USER_MONEY + ITEM_MONEY	Same as expected	Pass
Test-GM14	Automated	User has 1 crop in their inventory to sell and sold in the different seasons, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The quantity of this crop that is sold decreases by 1 which is zero	Same as expected	Pass
Test-GM15	Automated	User has 1 crop in their inventory to sell and sold in the same season, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The item they want to sell is in their inventory	Same as expected	Pass
Test-GM15	Automated	User has 1 crop in their inventory to sell and sold in the same season, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The user balance increased by ITEM_MONEY and their balance is now USER_MONEY + ITEM_MONEY	Same as expected	Pass

Table 15: **Functional Requirements Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-GM15	Automated	User has 1 crop in their inventory to sell and sold in the same season, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The quantity of this crop that is sold decreases by 1 which is zero	Same as expected	Pass
Test-GM16	Automated	User has 1 crop in their inventory to sell, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The user balance increased by ITEM_MONEY and their balance is now USER_MONEY + ITEM_MONEY	Same as expected	Pass
Test-GM17	Automated	User has 1 crop in their inventory to sell, this crop has a market value of ITEM_MONEY and has a balance of USER_MONEY and sell that one item	The quantity of this crop that is sold decreases by 1 which is zero	Same as expected	Pass
Test-GM18	Automated	The current season is Fall and the next season is changed to Winter	The ambient light changes	Same as expected	Pass
Test-GM18	Automated	The current season is Winter and the next season is changed to Spring	The ambient light changes	Same as expected	Pass
Test-GM18	Automated	The current season is Spring and the next season is changed to Summer	The ambient light changes	Same as expected	Pass
Test-GM18	Automated	The current season is Summer and the next season is changed to Fall	The ambient light changes	Same as expected	Pass

Table 16: **Functional Requirements Evaluation Results**

Id	**Type**	**Inputs**	**Expected Result**	**Actual Result**	**Result**
Test-GM18	Automated	The current season is Fall and the next season is changed to Winter	The camera angle changes	Same as expected	Pass
Test-GM18	Automated	The current season is Winter and the next season is changed to Spring	The camera angle changes	Same as expected	Pass
Test-GM18	Automated	The current season is Spring and the next season is changed to Summer	The camera angle changes	Same as expected	Pass
Test-GM18	Automated	The current season is Summer and the next season is changed to Fall	The camera angle changes	Same as expected	Pass
Test-GM18	Automated	The current season is Fall and the next season is changed to Winter	The base environment changes	Same as expected	Pass
Test-GM18	Automated	The current season is Winter and the next season is changed to Spring	The base environment changes	Same as expected	Pass
Test-GM18	Automated	The current season is Spring and the next season is changed to Summer	The base environment changes	Same as expected	Pass
Test-GM18	Automated	The current season is Summer and the next season is changed to Fall	The base environment changes	Same as expected	Pass

5 Changes Due to Testing

Source	**Feedback**	**Changes**
Supervisor (Survey)	Closing the shop or inventory is unintuitive	Added an X button on the top right of the shop or inventory modal
Supervisor (Survey) and TA	The tab buttons in the shop are unclear	Added hover and active effects to the tab buttons
Supervisor and TA	The tab buttons in the shop are out of place (i.e fixed to the top of the page)	Moved tab buttons to inside the shop modal
Supervisor (Survey)	The overall UI is not consistent	Changed the shop and inventory background colors to match the info header. Changed shop, inventory, and consultant buttons to match info header's
Supervisor and TA	The seeds in the shop are unclear (i.e what seeds can be planted in what season, how long do they take to grow)	Added hover to seed options that display the seasons it can be planted in and growth length. Seed options changed to have a background color that matches what season it can be planted in
Supervisor (Survey) and TA	The UI is unnecessarily cluttered when trying to sell items in the shop	Only items that you have at least 1 of are displayed when in the sell tab of the shop
Supervisor	The surrounding foliage on the farm is unattractive	Replaced tree model with grass model and flower model
Supervisor	There seems to be a possibility that the consultant options are overlooked	The consultant modal changed to automatically popup at the beginning of every season
Supervisor (Survey)	Needs more engaging audio	Added different background music for each of the 4 seasons. Added audio for button clicks, incrementing or decrementing money, harvesting crops, and planting the crop.
TA	Buying seeds are unclear. It seems like you are buying the full-grown plant rather than just the seed	The seeds in the shop have their icon and name changed to indicate they are seeds
TA	It is unclear what buying insurance on a crop does	Added a tooltip hover to summarize what buying insurance does
Test-DB3	Loading a saved game not working properly	Fixed database schema to allow for loading of a saved game
Test-AC3	Reset password flow does not provide email to reset password	Added reset password email notification that provides a link to reset password
Test-AC4	Even if the user is flagged as a bot during a verification step, they are still permitted to login	Debugged login system and the integrated result of verification to deny login if failing

There was a lot of feedback concerning the user interface and usability elements of the system. The team focused mostly on the functionality and core gameplay mechanics prior to the revision 1 demonstration. As such, the user interface elements were unpolished and many assets were placeholders. The supervisor and teacher assistants critiqued the elements, detailed in the table above, and provided insight into increasing the system's usability. The comments gave the team an initial direction for fixing the aforementioned issues. The team implemented these changes before the final demonstration and continues to work on improving the user interface. Although these issues have been resolved, this does not indicate that the user interface is perfect or near complete. As the product is used by more people, user testing will be conducted once again. The iterative process of feedback from the user and improving the interface will ensure the system is accessible and easy to use for everyone.

6 Automated Testing

Because we initially used create-react-app to bootstrap our React application, Jest was already set up as the test runner by default. Because of this, we decided to fully use Jest for all automated testing. It greatly simplified the setup of automated tests as all that was required was to create stubs of libraries that were incompatible with Jest.

In order to automatically test functional requirements such as a user logging in, we used react-testing-library to implement end-to-end testing. The library can be used to render components and simulate user interaction such as typing into an input field or clicking a button.

7 Trace to Requirements

The following Table 17 is a traceability matrix that maps the individual functional test cases to the functional requirements.

	FR1	**FR2**	**FR3**	**FR4**	**FR5**	**FR6**	**FR7**	**FR8**	**FR9**	**FR10**	**FR11**	**FR12**	**FR13**	**FR14**	**FR15**	**FR16**	**FR17**	**FR18**	**FR19**	**FR20**	**FR21**	**FR22**	**C1**	**C2**	**C3**	**C4**
Test-AC1	X	X			X																					
Test-AC2	X	X			X																					
Test-A3		X																								
Test-AC4	X				X																					
Test-AC5																										X
Test-GM1			X									X				X	X	X								
Test-GM2			X	X		X	X									X	X	X					X			
Test-GM3			X	X		X	X	X								X	X	X					X			
Test-GM4			X	X		X	X	X			X					X	X	X								
Test-GM5			X									X				X	X	X					X			
Test-GM6			X										X			X	X		X	X		X	X			
Test-GM7			X										X			X	X		X	X		X				
Test-GM8			X			X				X				X		X	X									
Test-GM9																X	X			X	X					
Test-GM10																X	X			X	X		X			
Test-GM11															X											
Test-GM12			X	X		X										X	X						X			
Test-GM13			X	X		X										X	X									
Test-GM14			X	X		X										X	X									
Test-GM15			X	X					X	X						X	X				X					
Test-GM16			X	X					X	X						X	X									
Test-GM17			X	X					X	X						X	X									
Test-GM18			X	X	X				X	X						X	X									
Test-GM19																X	X			X	X					
Test-DB1																X										
Test-DB2																	X									
Test-DB3																X	X									
Test-PG1																									X	
Test-PG2																										X

Table 17: Traceability Matrix for Functional Test Cases

The following Table 18 is a traceability matrix that maps the individual non-functional test cases to the non-functional requirements.

	LF1	**LF2**	**LF3**	**LF4**	**LF5**	**LF6**	**UH1**	**PR1**	**PR2**	**PR3**	**OE1**	**OE2**	**MS1**	**SR1**	**SR2**	**SR3**	**LR1**	**LR2**
test-LF1	X	X	X	X		X	X										X	
test-LF2					X						X	X						
test-LF3						X	X											
test-UH1						X	X											
test-P1								X		X					X			
test-P2									X	X					X			
test-P3								X	X	X			X					
test-OE1											X	X						
test-OE2												X						
test-MS1			X					X										X
test-SR1														X		X		
test-SR2														X	X	X		
test-SR3															X	X		
test-ACR1								X	X	X					X			
test-ACR2															X			
test-ACR3								X	X	X					X			
test-ACR4								X	X	X					X			
test-IR1									X	X							X	
test-IR2									X	X					X			

Table 18: Traceability Matrix for Non-functional Test Cases

8 Trace to Modules

Test	Modules
Test-AC1	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User
Test-AC2	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, AuthError, User
Test-AC3	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User
Test-AC4	CreateAccount
Test-AC5	GameSettings, DatabaseOperation, GameController
Test-GM1	GameController, FarmGrid, FarmTile
Test-GM2	GameController, FarmGrid, FarmTile, Item, Inventory
Test-GM3	GameController, FarmGrid, FarmTile, Item
Test-GM4	GameController, FarmGrid, FarmTile, Item
Test-GM5	GameController, FarmGrid, FarmTile
Test-GM6	GameController, AvatarMenu, Avatar, Consultant
Test-GM7	GameController, AvatarMenu, Avatar, Consultant
Test-GM8	GameController, Market, Item, Inventory
Test-GM9	GameController, GenerateStatistic, SeasonalEvents, FarmGrid, FarmTile, Inventory
Test-GM10	GameController, SeasonalEvents
Test-GM11	GameController, Inventory, Item
Test-GM12	GameController, Item, Inventory, Market
Test-GM13	GameController, Item, Inventory, Market
Test-GM14	GameController, Item, Inventory, Market
Test-GM15	GameController, Item, Inventory, Market
Test-GM16	GameController, Item, Inventory, Market

Table 19: Trace Between Test cases and Modules

Test	Modules
Test-GM17	GameController, Item, Inventory, Market
Test-GM18	GameController, Item, Inventory, Market
Test-GM19	GameController, SeasonalEvents, FarmGrid, FarmTile, Item, Inventory
Test-DB1	GameController, DatabaseOperations, Server
Test-DB2	GameController, DatabaseOperations, Server
Test-DB3	GameController, DatabaseOperations, Server
Test-PG1	CreateAccount
Test-PG2	CreateAccount
Test-LF1	AvatarMenu, Consultant, SeasonalEvents, Inventory, Market, GameSettings, CreateAccount, Login, FarmGrid, FarmTile
Test-LF2	AvatarMenu, Consultant, SeasonalEvents, Inventory, Market, GameSettings, CreateAccount, Login, FarmGrid, FarmTile
Test-LF3	AvatarMenu, Consultant, SeasonalEvents, Inventory, Market, GameSettings, CreateAccount, Login, FarmGrid, FarmTile, SeasonalEvents, GenerateStatistics
Test-UH1	AvatarMenu, Consultant, SeasonalEvents, Inventory, Market, GameSettings, CreateAccount, Login, FarmGrid, FarmTile, SeasonalEvents, GenerateStatistics
Test-P1	DatabaseOperations, GameController
Test-P2	DatabaseOperations, GameController, AvatarMenu, Consultant, SeasonalEvents, Inventory, Market, GameSettings
Test-P3	AvatarMenu, SeasonalEvents, Inventory, Market
Test-OE1	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User, GameController, FarmGrid, FarmTile
Test-OE2	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User, GameController, FarmGrid, FarmTile

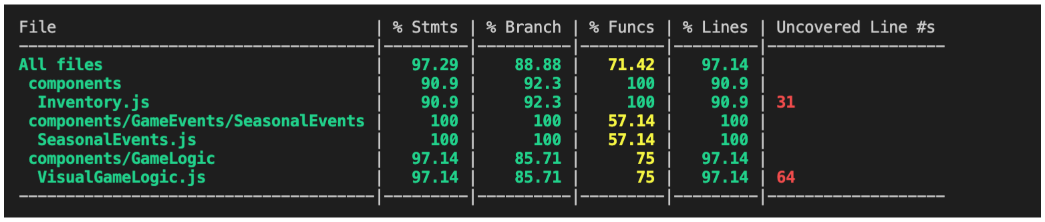
Table 20: Cont. Trace Between Test cases and Modules

Test-MS1	GameController, GameSettings, MusicPlayer
Test-SR1	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User
Test-SR2	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User
Test-SR3	DatabaseOperations, ServerFirebase
Test-ACR1	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User
Test-ACR2	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, AuthError, User
Test-ACR3	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, AuthError, User, GameController, FarmGrid, FarmTile, Market, Inventory, Consultant, AvatarMenu, GameSetting, SeasonalEvents
Test-ACR4	Login, DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, AuthError, User
Test-IR1	DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User, GameController
Test-IR2	DatabaseOperations, ServerFirebase, ClientFirebase, Socket, Server, AuthState, User, GameController

Table 21: Cont. Trace Between Test cases and Modules

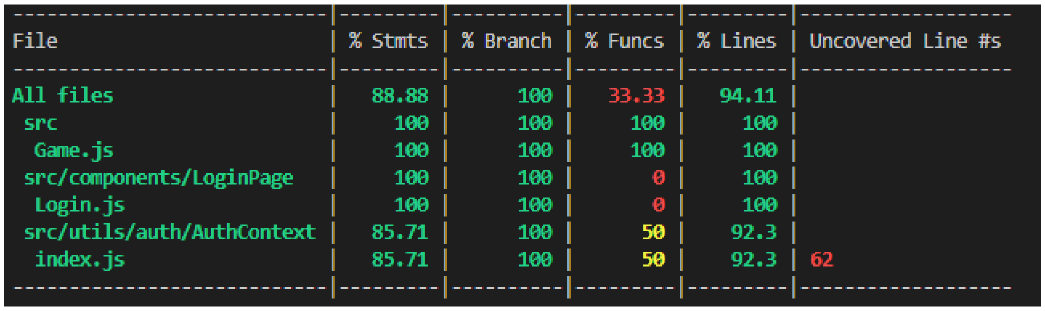
9 Code Coverage Metrics

Jest was used as a unit testing tool that also comes with an analysis of code coverage. As seen in Figure 1 and Figure 2, at least 85% code coverage has been obtained for the automated testing that was done. Thus, building confidence that most majority of the key areas in the code were run when testing these individual components. However, there was poor function coverage that was revealed in our automated testing because the team had prioritized testing more modules more thoroughly than others because some modules were much more significant in gathering research than others.



```
File                                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------------------------|---------|----------|---------|---------|-------------------
All files                             |   97.29 |    88.88 |   71.42 |   97.14 |
 components                           |    90.9 |     92.3 |     100 |    90.9 |
  Inventory.js                        |    90.9 |     92.3 |     100 |    90.9 | 31
 components/GameEvents/SeasonalEvents |     100 |      100 |   57.14 |     100 |
  SeasonalEvents.js                   |     100 |      100 |   57.14 |     100 |
 components/GameLogic                 |   97.14 |    85.71 |      75 |   97.14 |
  VisualGameLogic.js                  |   97.14 |    85.71 |      75 |   97.14 | 64
--------------------------------------|---------|----------|---------|---------|-------------------
```

Figure 1: Test Coverage for Unit Testing Inventory, Game Logic, and Events



```
-----------------------------|---------|----------|---------|---------|-------------------
File                         | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------------|---------|----------|---------|---------|-------------------
All files                    |   88.88 |      100 |   33.33 |   94.11 |
 src                         |     100 |      100 |     100 |     100 |
  Game.js                    |     100 |      100 |     100 |     100 |
 src/components/LoginPage    |     100 |      100 |       0 |     100 |
  Login.js                   |     100 |      100 |       0 |     100 |
 src/utils/auth/AuthContext  |   85.71 |      100 |      50 |    92.3 |
  index.js                   |   85.71 |      100 |      50 |    92.3 | 62
-----------------------------|---------|----------|---------|---------|-------------------
```

Figure 2: Test Coverage for Testing Accounts and Authentication

10 Appendix

10.1 Symbolic Parameters

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

Table 22: **Symbolic Parameter Table**

Symbolic Parameter	**Description**	**Value**
NEW_ACCOUNT_STATE	The initial state of a new account	Turn 0, 9 tiles of land owned, USER_MONEY
SCREEN_RESOLUTIONS	The list of screen resolutions that are supported by the user interface; given as a width and height	1920 pixels x 1080 pixels, 1366 pixels x 768, pixels, 1536 pixels x 864 pixels, 1440 pixels x 900 pixels, 1280 pixels x 720 pixels
MIN_TURNS	The minimum amount of turns played needed for a study participant to be a significant data point	12 turns
CONSULTANT_PRICE	The default price to purchase a consultant's advice	$150
ITEM_PRICE	The default price of a item	$150
USER_MONEY	The money a user has	USER_MONEY
EXPECTED_SURVEY_RATING	The expected survey rating to result as a pass	4
MAX_RESPONSE_TIME	The maximum time allowed for the system to respond by	5 seconds
MAX_INTERFACE_UPDATE_TIME	The maximum time allowed for the interface to respond by	1 second

10.2 Usability Survey Questions

User Experience Survery						
The following survey will be completed upon playing the game after 20 minutes						
Look and Feel						
Minimalistic Design	0	1	2	3	4	5
	[0 = too much clutter of elements, 5 = no clutter and minimal feel]					
Consistent Color Theme	0	1	2	3	4	5
	[0 = inconsistent color theme, 5 = consistent color theme]					
Engaging Audio	0	1	2	3	4	5
	[0 = audio is terrible to listen to, 5 = audio is enjoyable to listen to]					
Engaging Graphics	0	1	2	3	4	5
	[0 = graphics are not pleasing, 5 = graphics are pleasing and comfortable]					
Usability						
Age Group	0	1				
	[0 = age less than 18, 1 = age is 18 or above]					
Easy to understand	0	1	2	3	4	5
	[0 = hard to understand, 5 = easy enough to understand]					

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 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.)

One way the VnV Plan was different from what we actually conducted was that some tests were added and some tests were removed due to requirements changing. For example in the VnV Plan, we had planned to allow users to buy buildings. However, this functionality was removed later on in development as the team didn't see a need for them in terms of game mechanics. With this removal, the tests associated with this functionality were removed as well. Another change was with where we would store game logic. In the

VnV Plan, we first had it such that clients would have to make requests to the server to do any game action such as buying or planting seeds, harvesting or selling crops, etc. The architecture now is that the game logic is stored and handled client side, and so performance testing of the game logic API requests was no longer needed.

These changes mostly arose from the fact that the team did not fully plan and discuss all game mechanics and how they would be intertwined with each other. They were mostly discussed in a vacuum rather than what role they would play in the encompassing game loop.

For future projects, we will further emphasize thorough planning of the entire product. This can be done by a number of visualization or planning methods such as sketches, storyboards, paper prototypes, or any other low-fidelity prototype. More communication with the client such as more meetings would also help iron out all the details of the project and reduce the number of changes or clarifications later on in development. However, even with all of this, there is still no guarantee that all possible changes are anticipated, which further highlights how important it is to minimize these possibilities.