Appendix 1:

Test Plan

Change Log:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | By | Description |
| 1.0 | 08/03/2021 | Jordan Davis | Document Created |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents:

1. Objectives

* 1. Purpose

This document describes the test scope, strategy and schedule for all testing activities on the Artemis Lite project.

This test plan supports the following objectives:

* List the test requirements.
* Detail the type of testing required.
* Provide an estimate of test timelines.
* Detail testing strategies.
* List of testing deliverables.
  1. Scope

1. Test Requirements

The following list details the areas/features of the Artemis Lite system that have been identified as a testing requirement. These requirements have been split into sections representing the test type required.

* 1. Functional Testing
* Verify the system can have between two and four players.
* Verify users can enter their own name.
* Verify the player can roll two virtual dice and move on the board accordingly.
* Verify that when a player lands on an unowned element they can purchase it.
* Verify the system accurately deducts player resources and displays an appropriate message.
* Verify a player gains resources when they land on or pass the start square.
* Verify than an element can have both major and minor developments.
* Verify a full system is owned before an element can be developed.
* Verify that there minor developments are required before a major development can be complete.
* Verify that a player can develop from any location on the board.
* Verify that when a player runs out of resources the game will end.
* Verify that a player can choose to quit and end the game for all players.
* Verify that a player is charged rent when landing on a square owned by another player.
* Verify that the element owned can decline a rent payment and no player is charged.
* Verify that the game is won when all elements are fully developed.
  1. Non-Functional Testing
* Verify there are four game systems.
* Verify there are two systems consisting of three adjacent elements.
* Verify there are two systems consisting of two adjacent elements.
* Verify one of the two element systems is the least expensive to acquire.
* Verify one of the two element systems is the most expensive to acquire.
* Verify there is a square where no action occurs.
* Verify element names align with the real Artemis project.
* Verify an end sequence is displayed with major event headlines when the game is won.
* Verify the user interface is functional in different console windows (e.g. Eclipse/IntelliJ).

1. Test Strategy

Testing of the system will consist of four main test types; static verification, unit testing, integration testing and system testing. The details for each of these test types are shown below. These tests will be derived from the customer requirements to ensure compliance to the specification. A bottom up testing strategy will be adopted where there lowest level subsystems will be tested individually. Once this has been complete any calling subsystems will then be tested. This method will repeat until all of the system has been included.

* 1. Static Verification

Static verification on the form of code walkthroughs will be complete on a weekly basis. These walkthroughs will help to identify and rectify defects early in the development process. This method will also identify defects that are not highlighted by automated system messages such as comparing objects using ‘==’ rather than ‘.equals’.

A walkthrough will be complete during the weekly team meeting to review all code that has been updated since the previous review. All faults identified during this meeting will be recorded and assigned to a team member to resolve.

* 1. Unit Testing

Unit testing will be carried out on individual modules of the Artemis Lite project. These tests will be conducted using the JUnit framework by the developer responsible for creating that module.

Testing for each use case flow or function will be created after the development of the module has been complete. After any updates or refactoring of the code the tests will be run with the goal of ensuring that no unexpected errors have occurred. All units to be tested

* 1. Integration Testing

Integration testing will be used to test the combination of individual modules. It will be complete through the use of both JUnit tests and code walkthrough by a member of the development team. These tests will be complete each time two or more modules are combined into a functional unit. This will enable errors to be identified and rectified as early as possible in the development.

* 1. System Testing

System testing is conducted on a complete software system to ensure compliance to the customer requirements. This testing will be complete by developers in which they will ensure each requirement in the specification document has been validated.

1. Test Completeness

The testing of the system will be considered complete when the below requirements have been met:

* All developers agree that the software system is stable, meets all customer requirements and are satisfied all testing is complete.
* All Junit test cases have been run and passed.
* Ad hoc testing in all areas of the system have been complete.
* A trial run of the system complete with no new bugs or non-conformance identified

1. Acceptance Tests (functional)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Use Case Ref | Description of Test | Test Initialisation | Test Inputs | Test Procedure | Expected Results | Passed? |
| 01 | UC01 | Testing min valid number of players (2). | Launch game and select “Start Game” | 1. p1  2. p2 | Enter name 1 and name 2 when prompted. | List of registered players shown on screen contains both players. |  |
| 02 | UC01 | Testing max valid number of players (4). | Launch game and select “Start Game” | 1. p1  2. p2  3. p3  4. p4 | Enter name 1 and name 2 when prompted.  Select add new Company.  Enter name 3.  Select add new Company.  Enter name 4. | List of registered players shown on screen contains all 4 players. |  |
| 03 | UC01 | Testing invalid number of players (5). | Launch game and select “Start Game” | 1. p1  2. p2  3. p3  4. p4  5. p5 | Enter name 1 and name 2 when prompted.  Select add new Company.  Enter name 3.  Select add new Company.  Enter name 4.  Try to add 5th player in the same manner. | User should not be able to add a 5th player to the game. |  |
| 04 | UC01 | Testing that invalid player names cannot be entered. | Launch game and select “Start Game” | 1. ‘p1’  2. ‘p1’  3. ‘ p1 ‘  4. ‘ ‘  5.’p-2’ | Enter name 1 when prompted.  Enter name 2 (duplicate).  Enter name 3 (take not of the additional space before and after).  Enter name 4 (blank space)  Enter name 5. | Name 1 and name 5 should be valid and added to the system.  Name 2, 3 and 4 should be identified as invalid and prompt user for new response. |  |
| 05 | UC01 | Testing that players can modify their name after input. | Launch game and enter two new players called ‘p1’ and ‘p2’ | 1. p1  2. p2  3. p3 | Select the option to modify existing player.  Select p1 to modify.  Select rename p1.  Enter new name (p2) to confirm error message is shown.  Enter new name (p3). | User should not be able to create duplicate name of ‘p2’.  Updated username ‘p3’ should be shown and ‘p1’ should no longer exist. |  |
| 06 | UC01 | Testing that players can be removed after input. | Launch game and enter two new players called ‘p1’ and ‘p2’ | 1. p1  2. p2 | Select the option to modify existing player.  Select p1 to modify.  Select delete p1. | P1 should no longer be listed as a registered player |  |
| 07 | UC01 | Testing correct resources are allocated to a normal game. | Launch game, enter two new players called ‘p1’ and ‘p2’ and begin game. | 1. p1  2. p2 | From the menu select normal game.  Play through game until starting resources are shown.  Record starting resources.  Play game until user passes the starting square.  Record the resources that are granted. | Both starting resources and resources gained passing the starting square should match.  Both resources should be XXX. |  |
| 08 | UC01 | Testing correct resources are allocated to a long game. | Launch game, enter two new players called ‘p1’ and ‘p2’ and begin game. | 1. p1  2. p2 | From the menu select long game.  Play through game until starting resources are shown.  Record starting resources.  Play game until user passes the starting square.  Record the resources that are granted. | Both starting resources and resources gained passing the starting square should match.  Both resources should be XXX. |  |
| 09 | UC01 | Testing the functionality of selecting the first player to move. | Launch game, enter four new players and begin the game. | 1. p1  2. p2  3. p3  4. p4 | Press enter to roll the dice.  Confirm that the winning player matches the player with the highest roll.  Repeat test until there is a tie for the highest roll.  Reroll dice after tie and confirm winner.  Continue to the game and record the order of players. | Player with the highest dice roll will take the first turn.  Game will continue from the winner based on the order that named were input into the system. |  |
| 10 | UC02 | Testing dice roll at start of turn. | Launch game, enter two new players and begin the game. | 1. p1  2. p2 | Confirm that player is prompted to roll dice at the start of their turn.  Press enter and dice should be rolled & displayed on screen.  Record the value of both dice rolls and the total. | Dice roll will be displayed on screen to the user.  Both individual dice rolls should add up to make the total displayed. |  |
| 11 | UC02 | Testing player movement matches dice roll. | Launch game, enter two new players and begin the game. | 1. p1  2. p2 | Record the player position at start of turn  Roll the dice and record the value.  Record the player position after movement.  Confirm the that dice roll matches the number of elements moved.  Repeat with each player for 10 turns. | Player should move the number of spaces shown on the dice.  System should display details of the element that the player has landed on. |  |
| 12 | UC02 | Confirmation that turn menu is displayed correctly. | Launch game, enter two new players and begin the game. | 1. p1  2. p2  3. ‘display all’  4. ‘end turn’ | Roll the dice to start players turn.  Confirm menu of possible options is displayed.  Select the option for ‘display all’.  Confirm that the menu reappears after all elements have been displayed.  Select the option for end turn.  Confirm that the menu is no longer visible and the game has moved to the next player | User menu should be shown after a dice roll.  User menu should be shown after each action with the exception of ‘end game’ and ‘quit game’ |  |
| 13 | UC02 | Confirm that the menu matches the values entered by the user. | Launch game, enter two new players, begin the game and proceed to player menu. | 1. p1  2. p2  3. select menu option | Select a valid menu option.  Confirm that the option selected matches the response from the system.  Repeat for all menu options displayed | Menu options should correctly align with the user input and display the associated response. |  |
| 14 | UC02 | Confirm that the menu will display a warning message after an invalid menu input. | Launch game, enter two new players, begin the game, and proceed to player menu. | 1. p1  2. p2  3. invalid menu option (numeric)  4. invalid menu option (string) | Enter a number that is not shown on the menu list.  Confirm that invalid option prompt displayed, and user is given another opportunity to enter.  Enter a string value.  Confirm that invalid option prompt displayed, and user is given another opportunity to enter. | On invalid menu selection user should be prompted to try again. |  |
| 15 | UC03 | Test that develop element option will not be displayed unless the user has an element available for development. | Launch game, enter two new players, begin the game, and proceed to player menu. | 1. p1  2. p2 | Confirm development element option is not visible at the start of the game.  Play the game until you have purchased a full system of elements.  Confirm that develop element is now visible. | Develop element option will only be shown when a player owns all elements in a system.  User can develop from any location on the game board. |  |
| 16 | UC04 | Test the display of element details. | Launch game, enter two new players, begin the game, and proceed to player menu. | 1. p1  2. p2  3. ‘Get current element details’ | Select the option to display current element details.  Take note of element details displayed.  Repeat for five elements at random.  Compare all results to the game details and ensure they match | Each element will display the values shown in the game details. |  |
| 17 | UC05 | Test that players resources are correctly altered. | Launch game, enter two new players, begin the game, and proceed to player menu. | 1. p1  2. p2  3. ‘purchase element’ | Take note of starting resources.  Play the game for five rounds. Each time a players resources are to be changed take a note of the corresponding values.  End the game and ensure resources shown at the ending screen match calculated values. | Players resources should be modified by the value shown on screen. |  |
| 18 | UC05 | Test that a user is bankrupt when they go below 0 resources. | Launch game, enter two new players, begin the game, and proceed to player menu. | 1. p1  2. p2 | Play the game accepting all purchases until a player is in the position where they would lose all resources.  Accept a rent payment that would bankrupt a player. | Warning message should be displayed before the player is made bankrupt.  End credits should be shown and the game ends after a player is made bankrupt. |  |
| 19 | UC06 | Confirm that when a player does not want to purchase an element it is put up for auction. | Launch game, enter two new players, begin the game, and proceed to land on an available element | 1. p1  2. p2  3. ‘purchase element’ | Decline the purchase of an element.  Confirm auction started message is displayed. | When a purchase is rejected a message to show that the auction has started will be displayed. |  |
| 20 | UC06 | Test that after an auction the property is assigned to the correct player. | Launch game, enter four new players, begin the game, and proceed to an element auction. | 1. p1  2. p2  3. p3  4. p4  5. ‘view elements you own’ | P1 reject the purchase of an element.  P2 select the option to not take part in the auction.  P3 & p4 select the option to purchase the element for auction.  Take note of who wins the dice roll for the auction (p3 or p4).  Using the player that won the auction select the option to view your own elements. | Property should be assigned to the player that won the auction. |  |
| 21 | UC07 | When a player lands on an unowned element they have the option to purchase it. | Launch game, enter two new players, begin the game. | 1. p1  2. p2 | Play the game until a player lands on an unowned element.  Confirm that a message is displayed offering the element for purchase. | If a player lands on an unowned element a message will be displayed offering the element for purchase. |  |
| 22 | UC07 | When a player lands on an unowned element and they choose not to purchase an auction will begin. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. ‘decline purchase’ | Play the game until a player lands on an unowned element.  Select the option to decline the purchase of the element | When an element purchase is declined the auction starting message will be displayed. |  |
| 23 | UC07 | When a player lands on an unowned element and they do not have the required resources, an auction will begin. | Launch game, enter two new players, begin the game. | 1. p1  2. p2 | Play the game until a player lands on an unowned element which they do not have the resources to purchase. | Message will be displayed that the player cannot afford to purchase the element and the auction screen will be displayed. |  |
| 24 | UC08 | If a player lands on an owned element they are charged rent. | Launch game, enter two new players, begin the game. | 1. p1  2. p2 | Record both p1 and p2 resources.  Play the game until p1 lands on an element owned by p2.  Record the resources after rent has been taken | P1’s resources will be deducted by the amount shown for the element. P2 will game the same number of resources. |  |
| 25 | UC08 | Test that an element owner can choose not to charge rent to a player that has landed on their element. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. ‘do not charge rent’ | Record both p1 and p2 resources.  Play the game until p1 lands on an element owned by p2.  Record the resources of both  Confirm option is displayed on screen allowing p2 to not charge rent.  Select the option for p2 not to charge rent.  Record the resources of both players | Player should be given the option to not charge rent.  Resource balance for before and after landing on the square should be the same. |  |
| 26 | UC09 | Test that a player can have three minor developments on an element. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. develop element | Play the game until a player controls all elements in a single system.  Select the option to develop element.  Confirm that the element minor development level changes by 1.  Repeat until the element is at development 3.  Try to increase the element an additional tine once at level 3. | Player can develop an element when they own a full system.  Each time develop element is selected the dev level will increase by 1.  When an element is at minor dev level 3 it can no longer be increased, further development will instead increase major dev. |  |
| 27 | UC09 | Test that a player can have one major development on an element. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. develop element | Play the game until a player controls all elements in a single system.  Select the option to develop element.  Repeat until the element is at development 3.  Increase the element an additional time once at level 3 and confirm that the major development increases by 1.  Try to increase the major development level past 1.  Check that a max level message is displayed | Each time develop element is selected the dev level will increase by 1.  When an element is at major dev level 1 it can no longer be increased, further development will instead display a warning message |  |
| 28 | UC09 | Test that a player cannot develop an element if they do not have the required resources. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. develop element | Play the game until a player controls all elements in a single system and does not have enough resources to cover a development level.  Select the option to develop element.  Confirm that user does not have the option to increase development due to the lack of resources. | Message will be displayed showing the user that they do not have the resources required to develop the element. |  |
| 29 | UC10 | A player can choose to quit the game and it will end for all players and a loss message will be displayed. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. ‘end game’ | Record both players resources.  On player 1 select the option to end game.  Select the confirm end game option.  Confirm loss message is displayed.  Confirm that player 1’s resources have been set to 0 and p2 resources remain unchanged. | Ending messages will be displayed.  The player that selected end game will be set to bankrupt and all other players resources will be shown. |  |
| 30 | UC10 | A player can go bankrupt and the game will end for all players and a loss message will be displayed. | Launch game, enter two new players, begin the game. | 1. p1  2. p2 | Play the game until one player is in a position where they are going to be made bankrupt.  Confirm that ending message displays. | Ending messages will be displayed. |  |
| 31 | UC10 | When all elements are fully developed the game will end and a winning message will be displayed. | Launch game, enter two new players, begin the game. | 1. p1  2. p2 | Play the game until all elements are fully developed.  As soon as all elements are fully developed the winning credits should be displayed. | Gameplay will stop and message displayed as soon as the last element reaches max development. |  |
| 32 | UC11 | At any stage of the game a player can choose to review the game rules. | Launch game, enter two new players, begin the game. | 1. p1  2. p2  3. ‘rules’ | Play the game until the player menu.  Type ‘rules’ and select enter.  Confirm the rules are displayed.  Play the game until the player is at develop element menu.  Type ‘rules’ and select enter.  Confirm the rules are displayed.  Play the game until the player is at auction element menu.  Type ‘rules’ and select enter.  Confirm the rules are displayed. | Players are able to type ‘rules’ at any point in the game and the rules will be displayed. |  |

1. Acceptance Tests (non-functional)

|  |  |  |
| --- | --- | --- |
| ID | Description of Test | Passed? |
| 36 | Verify that the game board has four systems. |  |
| 37 | Verify that two of the systems consist of three adjacent elements. |  |
| 38 | Verify that two of the systems consist of two adjacent elements. |  |
| 39 | Verify that one of the two element systems is the least expensive to acquire. |  |
| 40 | Verify that one of the two element systems is the most expensive to acquire. |  |
| 41 | Verify that there is a square where no action occurs. |  |
| 42 | Verify that all element and system names can be found on the real Artemis project. |  |
| 43 | Verify that the end sequence will display major event headlines. |  |
| 44 | Verify that the game is functional in different console windows (IntelliJ). |  |