##### Diplomacraft

User Manual

**Team Royale with Cheese**

Simon Brami

Ryan Cole

J.B. Colette

Eric Do

Marcus Herndon

Ryan Huey

Kenny Nguyen

Table of Contents

[**Introduction**](#_4663b472ijw8) **1**

[**Environmental Needs**](#_v4x25vqfzmvn) **2**

[**Test Items**](#_iq7h3mjqqufg) **2**

[**Features to be Tested**](#_kgrn8kysryu0) **2**

[**Features Not to be tested**](#_8efql7suq4kh) **3**

[**Approach**](#_vzustwk442hd) **3**

[**Item Pass/Fail Criteria**](#_m91rekuydun6) **3**

[**Suspension Criteria and Resumption Requirements**](#_617imgjpupkw) **4**

[**Test Deliverables**](#_si6ufzl1ts02) **4**

[**Test Tasks**](#_owf3tisihdli) **4**

[**Responsibilities**](#_szgor5clcko2) **5**

[**Staffing and Training Needs**](#_brlfolabd2tz) **6**

[**Schedule**](#_jwh645rkhg1s) **6**

[**Risks and Contingencies**](#_ktbp7n87pwt9) **7**

[**Approvals**](#_l8x04dxxfg11) **7**

##### Introduction

The purpose of this test plan is to layout how we at Royale With Cheese plan to test the individual modules of Diplomacraft, and ensure that proper testing conditions are in place at the time of each test. Other information related to Diplomacraft can be found in Program Flow Chart, Project Plan, Use Cases, Vision Document, and User Manual. Testing will be performed entirely in-house.

##### Environmental Needs

Testing will be performed on a simple “Does it work? Yes or no” methodology; as such, no special measures need to be taken there. There will be no need for restricted use of the system for testing. Any and all communication between team members will be done both in person during meeting time and over the Internet through Slack.

##### Test Items

The scope of the test plan will cover for the most part the testing of almost all of the product’s features covered in the use cases. Such features will range from the game mechanics (such as the hold, move, support, and convoy) to non-gameplay elements (such as the login and register features). The user stories will play a large role in constructing our test cases.

##### Features to be Tested

1. User being able to register an account
2. User logging in to the game
3. User can list his or her lobby
4. User being able to join a lobby
5. User can create a lobby
6. User can close the lobby
7. User can start the actual game
8. Users can have a unit stay in the same position (Holding).
9. Users can move their units to other tiles.
10. Users can have Navies(Fleet) convoy their army across water tiles.
11. Users can have their units support an attack made by another unit.
12. Users can submit their orders
13. Players publicly chat with each other
14. Players privately chat with each other
15. User can view move history
16. Players can request to surrender
17. Test the winning window when the players meet all conditions for victory.
18. If the player is able to move their unit or remove them from the game when they are dislodged.
19. Player can gain a unit
20. Player can remove a unit when conditions for disbanding are met
21. The failure of a unit to support another unit due to it being attacked by another unit (Cutting off support)
22. The transitioning from fall to spring and vice versa
23. Displaying to the users whether their move is valid or not

##### Features Not to be tested

Generally testing will not cover anything outside of our use cases:

* User’s hardware
  + How well their cpu can run the game
  + Whether their mouse or keyboard works when interacting with the game
* Internet connectivity
* The functionality of the server being used
* The hardware that is being used to manage the server
* The features of the React framework and Electron Framework
* The functionality of communication mediums like Slack
* The functionality of applications used to help write documentation and Diagrams (Google Docs, Excel, draw.io, etc)
* The functionality of organizing and planning applications like Trello
* The functionality of version control resources like Git

##### Approach

We will use various technologies and methodologies when testing our application. The development of our project will be facilitated by a scrum focused agile methodology and our testing practices will be modeled after white-box testing, because of the smaller scale of our project. We feel that these two methodologies will be advantageous due to our limited development time and small code-base. To help organize our growing codebase, we will be using github as our main version control system.

##### Item Pass/Fail Criteria

A test item is considered to be passing if it is capable of following the steps outlined in its associated use case with its end result matching what is to be expected from that test item, which is also described in the use cases. Each step is expected to flawlessly lead to the subsequent step. The test cases must also cover the alternate courses for the test item, making sure it can follow the steps there to reach the same end result.

If there are any unintended results from testing the test items that are not mentioned in the use cases, they are treated as defects. Whether the test case pass or fails depends on the nature of the defect. If the defect inconveniences the user or is an obstacle in reaching the test item’s end result, the test case fails testing. The test case also fails if the defects presents a danger to the user or compromises the product’s integrity, even if the test item is able to reach its expected result. If the defects provide an alternate course for achieving the test item’s end result, the test case for that item fails if those alternate steps greatly deviates from the actual steps. Usually defects can be overlooked if they are really trivial and do not get in the way achieving the test item’s expected end result.

##### Suspension Criteria and Resumption Requirements

In the process of testing, we need to know when it will be best to simply end testing and attempt to fix any defects we find. Should the game, at any point, fail to load or crash, we will suspend testing, and attempt to find and fix the errors. Once we believe that have squashed all of the relevant bugs, we will resume testing. However, we cannot assume that additional errors have not arisen in the bug-fixing process. So, we will test any and all actions relating to the feature that we had just attempted to fix.

##### Test Deliverables

* Test Plan
* Test Case Specifications
* Test Tools
* Test Incident Report
* Test Summary Report

##### Test Tasks

* **Test Plan**
  + Complete the following.
* **Test Case Specifications**
  + Completion of Test Cases.
* **Test Tools**
  + None, all will be done by hand.
* **Test Incident Report**
  + Completion of all test cases and recording of results.
* **Test Summary Report**
  + Completion of all test cases and recording of results.

##### Responsibilities

* Eric
  + Responsibility: Assigning responsibilities to each member in the group.
  + Responsible test tasks:
    - User can close the lobby
    - User can start the actual game
    - Users can have a unit stay in the same position (Holding).
* JB
  + Responsibility: Ensuring all required elements are in place for testing.
  + Responsible test tasks:
    - User can list his or her lobby
    - User being able to join a lobby
    - User creates a lobby
    - User closes the lobby
    - User starts the game
* Simon
  + Responsibility: Makes critical go/no go decision for items not covered in the test plans
  + Responsible test tasks:
    - Users can have their units support an attack made by another unit.
    - Users can submit their orders
    - Players publicly chat with each other
    - Players privately chat with each other
* Ryan C.
  + Responsibility: Providing for resolution of scheduling conflicts.
  + Responsible test tasks:
    - Creation of new military units.
    - Disbandment of old military units.
    - Transitions between yearly seasons.
* Ryan H.
  + Responsibility: Delivers each item in the test items section.
  + Responsible test tasks:
    - User can view move history
    - Players can request to surrender
    - Test the winning window when the players meet all conditions for victory.
    - If the player is able to move their unit or remove them from the game when they are dislodged.
* Kenny
  + Responsibility: Selecting features not to be tested and setting risks
  + Responsible test tasks:
    - Player can gain a unit
    - Player can remove a unit when conditions for disbanding are met
    - The failure of a unit to support another unit due to it being attacked by another unit (Cutting off support)
* Marcus
  + Responsibility: Setting overall strategy for this level of plan
  + Responsible Test Tasks:
    - Users can move their units to other tiles.
      * Test different situations listed in use cases

##### Staffing and Training Needs

The server-side developers will need to learn C# since it is the main language for the back-end. They will need to learn how the server works and how it communicates with the client so that they know if the correct data is being sent/fetched.

The client-side developers will need to know how to use ReactJS and other libraries to know the logic behind the functions of all the use cases . They will also need to know ElectronJS to test if the application can work as both a desktop app and a web app.

##### Schedule

|  |  |
| --- | --- |
| **TASK** | **ESTIMATE EFFORT** |
| CREATE TEST SPECIFICATION | 20 MAN-HOUR |
| TEST EXECUTION | 10 MAN-HOUR |
| TEST REPORT | 5 MAN-HOUR |
| TOTAL | 35 MAN-HOUR |

|  |  |  |
| --- | --- | --- |
| **TASK NAME** | **START DATE** | **END DATE** |
| MAKING TEST SPECIFICATION | 03/20/2019 | 04/10/2019 |
| PERFORM TEST EXECUTION | 04/10/2019 | 04/30/2019 |
| TEST REPORT | 04/30/2019 | 05/05/2019 |

##### Risks and Contingencies

Making test specification might require to start developing some features before being able to know what will the test do exactly. That is why the test specification will take more time and will be evolutive as the project grows.

##### Approvals

Unit testing will involve testing all the individual units of the software, such as all the use cases and all the components of the software. This part of the testing will be mainly be done and approved by the front-end developers who have implemented the units themselves.

Integration Testing will involve testing the interaction between individual units. This will also be tested and approved by the front end developers who have implemented the functions of the software.

System Testing will involve evaluating the system’s compliance with its specified system requirements. The back end developers will be in charge of testing and approving this level of testing.

Acceptance Testing will involve evaluating the system’s compliance with its business requirements. This makes sure that the application we build is acceptable to be deployed in the market. This will involve all parties in the team.