Diplomacy Desktop App - Test Plan

Jason Nguyen, Ellen Burger, Sadiq Sarwar, Sovathana Heng, Sophanna Ek, Vishant Khunti, Chelsea Marfil, Jessica Hilario

Group E

CECS 475

Professor Anthony Giacalone

February 28, 2019

1. Introduction **3**
1.1 References 3
2. Test Items **4**
3. Features To Be Tested **4**
4. Features Not To Be Tested **8**
5. Approach **9**
5.1 Tools to be used 9
5.2 Metrics to be collected 9
5.3 How configuration management will be handled 9
5.4 Configurations to be tested 9
5.5 Regression test rules 10
5.6 How elements in the requirements that are untestable will be processed 10
5.7 Overall project testing approach and coverage requirements 10
5.8 Grouping of features/components to be tested together 10
5.9 How meetings and other organizational processes will be handled 11
5.10 Constraints to testing - resource availability, deadlines 11
6. Item Pass/Fail Criteria **11**
7. Suspension Criteria and Resumption Requirements **11**
8. Test Deliverables **11**
9. Test Tasks **12**
10. Environmental Needs **12**
11. Responsibilities **13**
12. Staffing and Training Needs **14**
13. Schedule **14**
14. Risks and Contingencies **15**
14.1 Risk 15
14.2 Contingency Plan 15
15. Approvals **15**
16. Revision **15**

1. Introduction

This is the master plan for our Diplomacy desktop app detailing test items, features to be tested, features not to be tested, approach, an item pass/fail criteria, suspension criteria, test deliverables, test tasks, environmental needs, responsibilities, staffing and training needs, schedule, risks, and approvals. This test plan has no resource or budget constraints. Because this is a master test plan, this will be a high-level plan for our project which will detail the testing of all parts of the system. The success of our testing will determine whether the system will deliver our planned functionalities and be accepted by users. Changes to our test plan will be made in this document and the team will communicate and coordinate key activities through discord and waffle.io.

1.1 References

These are references to other plans and documents containing relevant information to the project:

- Project Authorization - Our project has been authorized by Professor Giacalone.
- Vision Document
- Project Plan
- Use Cases
- Test Cases
- Quality Assurance Plan - The test engineer, Jason Nguyen, will write all test cases, and the rest of the development team will act as QA engineers by going through the steps of those test cases and evaluating their success/failure.
- Configuration Management Plan - Version number and changes will be written on all project documentation. The most recent version will be located on the group's github.
- Relevant policies and standards - There are no policies or standards relevant to our project.
- For lower level plans, reference higher level plans - This is our master/high level test plan.

2. Test Items

Items to be tested:

1. Testing should be done on both the front-end and back-end of the Diplomacy Desktop Application on MacOS, Windows, and Linux environments.
2. User Manual
3. Use cases

3. Features To Be Tested

Testing the User Login (H)

- Detecting whether or not someone is using a specific username
- It switches to a game screen.

Testing the game setting functionality (M)

- Player should be able to edit, modify and save changes made to the settings.

Testing the game description (M)

- Player should be able to enter a description for the game session.

Testing entering the game name (M)

- Player should be able to change the name of the game session

Testing the adjudication period (H)

- The period should be saved by the session.

Testing the user's display of the game (H)

- The end user should be able to see the partitioned game map while given a unique name color and country color.
- The end users should be able to see the available orders (move, support, convoy, hold).
- End users should be able to exit out of the game at will.

Testing the chat to start conversation with a single player (L)

- Players should be able to see the list of players and a window should pop up to chat with them.

Testing the view chat with a single player (L)

- Players should show a public chat and user list.
- Players should be able to see the conversations between another player.

Testing the send chat with a single player (L)

- Players should be able to type a message and send it.

Testing the spectator views chat (L)

- Spectator should be able to view the public chat only.

Testing the view chat with all players (L)

- Players should be given different options and the general chat room is displayed.

Testing send chat to all players (L)

- Player should be able to type a message and send it to all players.

Testing the Start Game (H)

- A game session should be created after logging in.
- End users should be able to join through a game ID.

Testing the time limit (H)

- Given a time limit, orders will not be counted if they are not submitted.

Testing the join game as player (H)

- End user should be able to be redirected to the game once the game ID is entered.

Testing the join game as spectator (H)

- End user should be able to open the game and spectate it.

Test the create a game function (H)

- Application should open, request a username, and a game ID should be created.

Testing the End Game (H)

- If an end user has 18 or more supply centers, the game ends.
- End users can agree to end the game on their own terms.
- Majority rule can end the games.

Testing the exit game (H)

- Player should be able to disconnect from game while the game continues

Testing the Move order (H)

- End user should be able to move their army unit.
- End user wants to move their fleet unit.

Testing the Support Order (H)

- End users should be able to select an army or fleet to support and is validated.
- Supporting unit attacked by the original should not have support cut.

Testing the cut support (H)

- Supporting unit should be dislodged and support cut if attacker's strength is greater than the supporter's strength

Testing the Convoy Order (H)

- Fleet unit is able to select the army across a body of water to convoy, then select a target area the enemy has targeted.

Testing attacking the convy (H)

- The convoy should be cancelled and fleet should be dislodged.

Testing multiple convoys on same unit (H)

- Convoy fails if all convoys are cutted off.

Testing disrupting a convoy (H)

- Convoy order is not successful if convoy order fails.

Testing the holding position (H)

- Players should be able to keep units in place whether it was specified or not.

Testing dislodged unit causes standoff (H)

- None of the units can move if both have equal strength that moves to same province as dislodged unit.

Testing multiple units occupy same province (H)

- The unit should enter the province if one unit has greater strength than all others, unless otherwise stated.

Testing exchange places via convoy (H)

- Swap should work when both armies are ordered into each others' provinces.

Testing units retreat to the same province (H)

- Both units should disband regardless of ownership or strength.

Testing civil disorder (H)

- When a player leaves game early or doesn't submit orders, unit is disbanded instead of dislodged.

Testing civil order disbandment (H)

- Disbanded based on alphabetical order of province names.

Testing the strength comparison (H)

- Attack should fail if both units have equal strength.

Testing the support action (H)

- Supported unit should gain power is adjacent.
- No support is given if unit supports a nonadjacent unit.
- Support unit loses its gained by when attacked.

Testing the Resolve order (H)

- At the end of everyone's turn, all orders will be resolved.

Testing the move order into a non-adjacent province (M)

- Unit should hold instead.
- Unit should be able to convoy if valid.

Testing the Army from going into the water (M)

- Army unit should hold when ordered and convoyed.

Testing the move order for fleet into an inland province (M)

- Fleet should hold when order is to move inland.
- Fleet should move when order is to move coastland.

Testing the supported attack cuts a convoyed attack (M)

- Support should not be cut.

Testing the end turn functionality (H)

- End user should be able to end their turn after putting in their orders.

Testing the Retreat (M)

- End user should be able to move troops in a designated region.
- End user's troops should be disbanded if no province is available.

Testing the unit rotation (H)

- None of the units involved should move whether a swap/attack or 3+ players are involved.

Testing the seize supply center (H)

- Supply center should remain in ownership of the country.

Testing the reorder move (M)

- The order to move or hold should be given to the unit.

Testing the Disband Units (H)

- A country should be disbanded if it has fewer supply centers than units.
- A country should be disbanded if there is no available province to retreat.
- The end user should be able to disband on its own.

Testing the Request Draw (M)

- End users should be able to accept the request for a draw and have others notified.
- End users should be able to reject the request to draw and have others notified.

Testing the Gain Troops (H)

- End user should be able to select an army or fleet units to build on their original supply centers.

- End user should not be able to select units if their units is greater than the number of supply centers.
- End user should not be able to select units if the country's original supply centers are occupied or no longer owned.
- End user should not be able to create a selected fleet if the end user chooses to build a fleet on a supply center whose land is not adjacent to a coast.

Testing an illegal move: unit moves into a province held by another unit without support (H)

- Attack should fail and the unit holds

Testing an illegal move: unit attempts position trade without convoy.

- Neither unit should move if each unit has equal strength.

Testing the assign power (H)

- All players in game should be assigned power.

Testing advance year (H)

- Year should be advanced once Winter is over.

Testing the change of seasons to Spring (H)

- Players should be able to build new units, retreat or disband units.

Testing the change of seasons to Fall (H)

- All orders should be resolved.

Testing the change of seasons to Winter (H)

- All orders should be resolved.

4. Features Not To Be Tested

Testing the desktop app on the end user's computer.

Testing the end user's connection

- The developers are not responsible for testing the connection of the end users.

5. Approach

5.1 Tools to be used

The Electron framework will be used for the development and testing of the Diplomacy desktop app. This will require each team member to learn how to use the framework, JavaScript, HTML, and CSS.

5.2 Metrics to be collected

Metrics to be collected include the percentage of individual test cases passed, as well as percentage of total test cases passed. The results of each use case test will be documented in the test cases. Metrics include date tested, tester, and test results.

5.3 How configuration management will be handled

Changes in the software will be documented on the projects github. Changes on the documentation will be displayed at the top of every document with the version number and a short description of the changes.

5.4 Configurations to be tested

Configurations to be tested	**Software**	**Hardware**
1.	Diplomacy desktop app, Linux OS	Laptop or Desktop
2.	Diplomacy desktop app, Windows OS	Laptop or Desktop
3.	Diplomacy desktop app, MacOS	Laptop or Desktop

5.5 Regression test rules

We will do regression testing when new features or code are introduced into our existing, functioning code base. Old test cases will be run against the new version to make sure that old existing capabilities still work with the newly introduced features/code. Regression testing will ensure that even little changes don't break our software.

5.6 How elements in the requirements that are untestable will be processed

We will not worry much about requirements that are untestable such as those in our features not to be tested section. The features not to be tested deal with how our application will operate on the end user's specific machine configuration which we cannot exactly control. What we can control is carrying out tests on each of the machine configurations available to us and see how our test cases stand.

5.7 Overall project testing approach and coverage requirements

Since this is a master test plan, we will be testing all of the features and functionality of our system. Our test cases will detail the testing criteria for features mentioned in our features to be tested section.

5.8 Grouping of features/components to be tested together

Group #	Features/components to be tested together
1	User login, create game, enter username, display game, chat feature
2	User login, join game, enter username, display game, chat feature
3	Game play which includes display game, chat feature, move order, support order, convoy order, hold order, resolving order, end turn, retreat, disband, request draw, gain troops, and change seasons.

Eventually, we will have to test our completed application which all features and components are bundled up together.

5.9 How meetings and other organizational processes will be handled

Meetings of the project's team will be held every Tuesday and Thursday at 3:00PM-4:15PM until May 9, 2019.

5.10 Constraints to testing - resource availability, deadlines

Deadline for testing is on May 09, 2019.

6. Item Pass/Fail Criteria

For each test case, an item is considered pass if all steps defined in the test case are passed. A test case is considered to have failed if at least one of the test conditions of the test case fails.

7. Suspension Criteria and Resumption Requirements

Testing on a test case will be suspended after a fail criteria has been identified and documented. In such a case, the tester will then inform developers and tests will be resumed at the developer's preferred time.

8. Test Deliverables

No.	**Description**	**Author**	**Delivery Date**
1.	Test Plan	Project Team	02/26/19
2.	Test Cases	Project Team	02/26/19
3.	Procedure Specifications	Test/QA Engineers	02/26/19

4.	Test Logs	Test/QA Engineers	05/09/19

9. Test Tasks

Task No.	Description	Required Resources	Completion Date	Corresponding Task No.
1	Creating test scenario	System use cases	02/26/19	2
2	Creating test case	Excel Spreadsheet	02/26/19	1, 3
3	Creating test script	Electron Framework	05/01/19	2, 4
4	Executing test case	Electron Framework	05/01/19	3, 5, 6
5	Report bugs	Waffle.io	05/01/19	4, 6
6	Creating issues log	Waffle.io	05/01/19	4, 5

10. Environmental Needs

Needed items:

- Access to the internet
- Git to log errors for the team
- Ensure all testers use the same dependencies and versions. (package.json)
- Use the Electron framework
- Node.js module

- Test data will be provided by testers through test cases. Testers will specify a pass/fail/not applicable result for each test scenario and steps for that test case scenario.

11. Responsibilities

Role	Team Member
Test Engineer	Jason Nguyen
QA Engineer	Sadiq Sarwar Jessica Hilario Ellen Burger Vishant Khunti Sovathana Heng Chelsea Marfil Sophanna Ek

Responsibilities:
As the test engineer, Jason, will be in charge of selecting which features are to be tested and not to be tested. He will also be assigning other team members to test certain areas as needed. All other members in the team will be working along with the test engineer to check all processes and ensure that all test cases pass successfully. These tests will be done following a strict schedule that is to be made by the test engineer. Any scheduling conflicts with regards to testing will need to be discussed with the test engineer, as he will be making the critical decisions necessary for which items will and will not be covered in the test plans. The finalized test items will be delivered by the test engineer in the end.

12. Staffing and Training Needs

Each student team member will work on the project and may need to be trained in using the Electron Framework, JavaScript, HTML, and CSS.

13. Schedule

No.	Description	Start Date	End Date
1.	Writing Test Plan	02/19/19	02/26/19
2.	Writing Test Cases	02/19/19	02/26/19
3.	Creating Test Script	02/26/19	03/05/19
4.	Executing Test Case	02/26/19	03/05/19
5.	Report Bugs	03/05/19	05/09/19
6.	Writing Incident Report	03/05/19	05/09/19
7.	Writing Summary Report	03/05/19	05/09/19

Testing takes place during and after coding development. Due to the uncertainty surrounding the time required to build the project, slippages in the schedule are a possibility. Testing may be crunched, but we will always find time for essential features and systems. If necessary, other features will be evaluated on which ones are in need of testing the most before the final deadline.

14. Risks and Contingencies

14.1 Risk

In case of a miscalculated project time estimation, the project won't be done as we projected.

14.2 Contingency Plan

Establish the scope before starting the testing tasks and pay attention in the project planning and also track the project deadline constantly throughout the project development.

15. Approvals

Upon the completion of the project, the professor should agree and determine the steps to proceed further.

16. Revision

Date	Version	Description
2/19/2019	<1.0>	Original document