SE 3XA3: Test Plan
BigTwo

Team 06, Aplus^3
Jiaxin Tang, tangj63
Manyi Cheng, chengm33
Senni Tan, tans28

March 5, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
March 3rd	1.0	Initial Draft
March 4th	1.1	Detailed Revision

1 General Information

1.1 Purpose

The purpose of the test plan is to ensure our project BigTwo to be tested systematically and thoroughly. We will build suitable test suites to outline the goals that BigTwo should meet according to the functional requirements and non-functional requirements provided in the SRS.

1.2 Scope

The test plan mainly focuses on checking if each functional unit of BigTwo provided functionalities as expected based on the rules of BigTwo. Besides, it also tests if all the non-functional requirements were met.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
SRS	Software Requirements Specification
PoC	Proof of Concept

Table 3: **Table of Definitions**

Term	**Definition**
BigTwo	Name of our project

1.4 Overview of Document

BigTwo would be re-implemented for web browsers using Python instead of Java for the existing project. This document specifies what types of tests will be used for testing and gives a detailed description of how the test will be performed. All tests will be built according to SRS and PoC to make sure BigTwo achieves its functionalities.

2 Plan

2.1 Software Description

The software will allow users to play the BigTwo game on a website with a web browser on the computer. The implementation of the game will be completed in Python and the implementation of the web page will be completed in HTML, CSS, and JavaScript, along with modern web technologies.

2.2 Test Team

The test team will consist of all Group 6 members who are involved in the development of the BigTwo project. The test team members are Senni Tan, Manyi Cheng, and Jiaxin Tang. The members will test all aspects of the software evenly covering the different types and methods of testing.

2.3 Automated Testing Approach

2.3.1 Unit Testing

The automated unit testing will be used to check the correctness of the modules such as functions and methods in classes for the game. The functions will be given various inputs to cover normal, boundary, and edge cases. The results of the tested functions will be compared with expected outputs. The results of each unit test case and function must match the requirement specification document for the project to ensure the correctness and robustness of the program.

2.3.2 Integration Testing

Integration testing will be used to test the connections between classes and modules to ensure they produce the desired and correct output. This integration testing will require the components to work together to update the state of the game based on various input combinations.

2.3.3 System Testing

System testing will be used for testing the entire game system with various user inputs to ensure the application meets all functional and non-functional

requirements for the user.

2.4 Testing Tools

The PyTest will be used for unit testing for the functions of the modules and the methods of the classes of the game. The JEST framework will be used for unit testing for JavaScript. For the system testing, we will have invited volunteers and ourselves to manually play the game to test it.

Table 4: **Testing Schedule**

Due Date	Task	Tester
Mar 22, 2021	unit testing on Python functions	Group 6 members
Mar 24, 2021	unit testing on Python classes' methods	Group 6 members
Mar 26, 2021	unit testing on JavaScript files	Group 6 members
Mar 29, 2021	system testing on the entire game	Group 6 members and volunteers

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 User Interface

3.1.1.1 Game Mechanics

1. FR-UI-1

   Type: Functional, Dynamic, Manual

   Initial State: The user is connected to the BigTwo website.

   Input: User selects an appropriate game mode from the navigation.

   Output: Display a window of game of BigTwo

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

2. FR-UI-2

   Type: Functional, Dynamic, Manual

   Initial State: Before a game has started or during a round of game.

   Input: User clicks the "Help" button for game rules.

   Output: The interface displays the game rule on the screen.

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

3. FR-UI-3

   Type: Functional, Dynamic, Manual

   Initial State: A round of game is started.

   Input: A round is finished.

   Output: The interface displays a message to the game screen informing the result of the round, showing the scores of the player and dealers.

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

4. FR-UI-4

   Type: Functional, Dynamic, Manual

   Initial State: The user is connected the BigTwo website.

   Input: The user starts a round of game.

   Output: The game interface must display the interactive deck of cards the user possesses such that each card can be selected by the user upon clicking it.

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

5. FR-UI-5

   Type: Functional, Dynamic, Manual

Initial State: The user interface must include an image to represent the background of the game.

Input: The user starts a round of game.

Output: The user interface display an image to simulate a real game of BigTwo.

How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

6. FR-UI-6

   Type: Functional, Dynamic, Manual

   Initial State: The user starts a round of game.

   Input: The user selects a set of valid combinations of cards from their deck to make a deal.

   Output: Cards selected are removed from their deck, and displayed to the other players.

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

7. FR-UI-7

   Type: Functional, Dynamic, Manual

   Initial State: The game proceeds to user's turn after entering the game by clicking the play game button.

   Input: The user clicks the pass button.

   Output: The turn goes to the next player to the right of user.

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

8. FR-UI-8

   Type: Functional, Dynamic, Manual

   Initial State: The user is connected the BigTwo website.

Input: The user clicks on the play game button.

Output: The Back of each player's card set is displayed (number cards in a card deck).

How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

9. FR-UI-9

   Type: Functional, Dynamic, Manual

   Initial State: The user is connected the BigTwo website.

   Input: The user clicks on the play game button.

   Output: The user interface must indicate who is the current dealer by labelling the dealer's icon.

   How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

10. FR-UI-10

    Type: Functional, Dynamic, Manual

    Initial State: The user starts a round of game.

    Input: The user clicks the pause button.

    Output: The game should paused and the current game status should be saved.

    How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

11. FR-UI-11

    Type: Functional, Dynamic, Manual

    Initial State: The user starts a round of game.

    Input: The user clicks the new game button.

    Output: The interface should immediately return to the default window of a new game and clear any metadata from last game.

How test will be performed: A game of BigTwo will be simulated by a group member to perform manual testing of this functionality.

3.1.2 Game Rules

1. FR-GR-12

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state.

   Input: The method startGame() is called.

   Output: The user object will start the game with a random STARTING DECK OF CARDS, with 4 points and more, point counting rules: J=1, Q=2, K=3, A=4, 2=5, others=0.

   How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

2. FR-GR-13

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state, each player with a random STARTING DECK OF CARDS.

   Input: Each player receives a random STARTING DECK OF CARDS, the one with the Diamonds 3 invokes the new trick.

   Output: Trick 1 starts, with the player with the Diamonds 3 being the first one to deal.

   How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

3. FR-GR-14

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state, a trick has started.

Input: The first player deals.

Output: The next player in the counter-clockwise direction is the dealer.

How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

4. FR-GR-15

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state, a trick has started.

   Input: Player selects a valid combinations of cards to deal.

   Output: The cards are dealt after verifying the combination.

   How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

5. FR-GR-16

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state, a trick has started.

   Input: Player selects a valid combinations of cards to deal.

   Output: The cards are dealt after verifying the combination, verifying the combination is higher than the one before, with the same number of cards.

   How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

6. FR-GR-17

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state, two players have passed in succession.

Input: One of the player clicked the pass button in succession.

Output: The current trick is over, all cards are gathered up and a new trick will be started.

How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

7. FR-GR-18

   Type: Unit Test, Dynamic, Automated

   Initial State: Custom in-game state, a trick has started.

   Input: A player runs out of cards.

   Output: The game ends, and the scoring session begins.

   How test will be performed: A group member will write test cases for corresponding functions to perform unit testing of this functionality using PyTest.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel Requirements

3.2.1.1 Appearance Requirements

1. NF-L1

   Type: Static, Manual

   Initial State: When the BigTwo game enters the splash screen

   Input/Condition: Starting the game

   Output/Result: A splash screen will display the name of the game and the logo of the company.

   How test will be performed: The test will be performed by manually starting the game to check if the splash screen displays as expected.

2. NF-L2

Type: Dynamic, Manual

Initial State: After the game starts up, on the main menu screen

Input: Pressing the "Start"/"Help" button

Output: After pressing the "Start" button, a new round of BigTwo will start.
After pressing the "Help" button, a brief instruction of BigTwo will be provided to the player.

How test will be performed: The test will be performed by manually pressing the "Start" button to see if a new round of game is loaded and pressing the "Help" button to check if the screen displays instructions for the game.

3. NF-L3

Type: Static, Manual

Initial State: Any state of the game

Input: N/A

Output: The game will display in a proper size fitting on a web.

How test will be performed: The test will be performed visually by confirming that the size of the game is suitable for the player with texts and images on the screen shown clearly.

4. NF-L4

Type: Static, Manual

Initial State: Any state of the game

Input: N/A

Output: The layout and interface of the game will be clear for people want to play BigTwo.

How test will be performed: The test will be performed by comparing the interface of our BigTwo game with the existing projects and asking different BigTwo players if the appearance of our BigTwo looks attractive to them.

3.2.1.2 Style Requirements

1. NF-L5

   Type: Dynamic, Manual

   Initial State: On the main menu screen

   Input: After pressing the "Start" button

   Output: The game will generate thirteen cards automatically for each of the four players. Each group of cards will be placed on one of the four sides.

   How test will be performed: The test will be performed by checking if the generated cards are mess up with each other.

2. NF-L6

   Type: Static, Manual

   Initial State: after generating cards for a new round of the game

   Input: N/A

   Output: The color of the cards will be strongly different from the color of the background.

   How test will be performed: The test will be performed visually by confirming the color the cards are different from the background.

3.2.2 Usability and Humanity Requirements

3.2.2.1 Ease of Use Requirements

1. NF-U1

   Type: Dynamic, Manual

   Initial State: Any state after game's initial start.

   Input: Any mouse input

   Output: Any response from the game according to each mouse input

How test will be performed: The test will be performed by manually using the mouse to select cards or press buttons in the game to check if the game responds as expected.

2. NF-U2

   Type: Dynamic, Manual

   Initial State: After game's initial start.

   Input: N/A

   Output: Users complete the game

   How test will be performed: The test will be performed by inviting different users to play the game to check if they could complete a round of the game easily.

3.2.2.2 Personalization and Internationalization Requirements

1. NF-U3

   Type: Static, Manual

   Initial State: Any state of the game

   Input: N/A

   Output: The screen displays texts in English

   How test will be performed: The test will be performed by visually checking if the game shows English texts.

3.2.2.3 Learning Requirements

1. NF-U4

   Type: Dynamic, Manual

   Initial State: On the main menu screen

   Input: Pressing the "Help" button

   Output: A brief instruction of BigTwo will be provided to the player.

How test will be performed: The test will be performed by manually pressing the "Help" button to check if the screen displays instructions on how to play the game.

3.2.3 Accessibility Requirements

1. NF-A1

   Type: Dynamic, Manual

   Initial State: Before starting the game

   Input: Loading the game on a web browser on a computer

   Output: The game is executable on the web browser.

   How test will be performed: The test will be performed by manually loading the game on various computers with different operation systems and on different web browsers(Firefox, Google Chrome, etc.) to check if the game is executable.

3.3 Performance Requirements

3.3.0.1 Speed and Latency Requirements

1. NF-P1

   Type: Dynamic, Automated

   Initial State: Any state of the game

   Input: Any mouse input

   Output: Any response from the game according to the input

   How test will be performed: The test will be performed by having a watcher keep track of the main game thread to ensure the game responds immediately to the user input. If the game is not responding, the watcher will time out and the test fails.

3.3.0.2 Precision or Accuracy Requirements

1. NF-P2

Type: Dynamic, Manual

Initial State: Any state of any turn of a round of the game

Input: Selecting invalid combination of cards

Output: The game will display warning messages and refuse the invalid cards

How test will be performed: The test will be performed by manually selecting cards violating the rules of BigTwo to see if the game displays warning messages and refuse the cards.

3.3.0.3 Longevity Requirements

1. NF-P3

Type: Dynamic, Manual

Initial State: Before starting the game

Input: Loading the game on a web browser

Output: The game is executable on the web browser.

How test will be performed: The test will be performed by manually loading the game on different web browsers(Firefox, Google Chrome, etc.) to check if the game is executable.

3.3.1 Operational and Environmental Requirements

3.3.1.1 Release Requirements

1. NF-O1

Type: Dynamic, Manual

Initial State: Any state of the game.

Input: Any input.

Output: Any response from the game according to the input.

How test will be performed: The test will be performed by manually checking if the new release of the game passes the test cases built for the previous version.

3.3.2 Cultural Requirements

1. NF-C1

   Type:Static, Manual

   Initial State: Any state of the game.

   Input: N/A

   Output: N/A

   How test will be performed: The test will be performed by manually checking if there are any words or graphics that are offensive to people with any culture.

3.3.3 Legal Requirements

3.3.3.1 Compliance Requirements

1. NF-Le1

   Type:Static, Manual

   Initial State: Any state of the game.

   Input: N/A

   Output: N/A

   How test will be performed: The test will be performed by manually checking if the game violates any law.

3.3.3.2 Standard Requirements

1. NF-C2

   Type:Static, Manual

   Initial State: Any state of the game.

   Input: N/A

   Output: N/A

   How test will be performed: The test will be performed by manually checking if the game follow the MIT Open license.

3.3.4 Health and Saftey Requirements

1. NF-H1

   Type:Static, Manual

   Initial State: Any state of the game.

   Input: N/A

   Output: N/A

   How test will be performed: The test will be performed by manually checking if there are any gambling elements involved in the game.

3.4 Traceability Between Test Cases and Requirements

Table 5: Functional Traceability Matrix

Requirement #	Description	Test ID(s)
FR1	The user must be able to play a game of BigTwo.	FR-UI-1
FR2	he user interface must allow user to see the game rules before or during the game.	FR-UI-2
FR3	The user interface must notify the user the results of a round.	FR-UI-3
FR4	The game interface must display the interactive deck of cards the userpossesses	FR-UI-4
FR5	The user interface must include an image to represent the backgroundof the game	FR-UI-5
FR6	The user interface must include a button for the user to indicate theiraction to deal.	FR-UI-6
FR7	The user interface must include a button for the user to indicate theiraction to pass	FR-UI-7
FR8	he user interface must display how many cards each other player possesses without revealing actual card sets.	FR-UI-8
FR9	The user interface must indicate who is the current dealer.	FR-UI-9
FR10	The user interface must include a button for the user to indicate theiraction to pause	FR-UI-10
FR11	The user interface must include a button for the user to indicate theiraction to start a new game	FR-UI-11
FR12	The user will start the game with STARTING DECK OF CARDS with4 points and more, point counting rules: J=1, Q=2, K=3, A=4, 2=5,others=0.	FR-GR-12
FR13	At beginning of each game, the player with the Diamonds 3 is the firstone to deal in first trick.	FR-GR-13

Table 6: Functional Traceability Matrix (continued)

Requirement #	Description	Test ID(s)
FR14	Play proceeds counter-clockwise, with normal climbing-game rules applying: each player must play a higher card or combination than theone before, with the same number of cards.	FR-GR-14
		FR-GR-15
		FR-GR-16
FR15	A trick is over when three players have passed in succession.	FR-GR-17
FR16	When a trick is over, all cards are gathered up and a new trick is startedwith all players, initiated by the last player to play.	FR-GR-17
FR17	The game ends when one player runs out of cards, the scoring session will begin automatically	FR-GR-18

Table 7: Non-Functional Traceability Matrix

Requirement #	Description	Test ID(s)
3.1.1-1	The product shall display the name of the product and the logo of thecompany upon starting.	NF-L1
3.1.1-2	The product shall have clearly labelled buttons.	NF-L2
3.1.1-3	The product shall have a proper size to fit on a web page.	NF-L3
3.1.1-4	The product shall appear attractive to users.	NF-L4
3.1.2-1	The product shall generate cards in a clear way.	NF-L5
3.1.2-2	The color of the cards shall be distinct from the background.	NF-L6
3.2.1-1	The product shall only require the users' mouse for navigating the menu and selecting cards to play the game.	NF-U1
3.2.1-2	The product shall be easy for users with rules of the game provided	NF-U2
3.2.2-1	The product shall be provided in English.	NF-U3
3.2.3-2	The product shall provide a simple instruction of the game for users.	NF-U4
3.3	The product shall be executable on the majority of computers and web browsers.	NF-A1
3.4.1	The product shall respond to each user input within 2 seconds.	NF-P1
3.4.3	The product shall respond to each user input correctly according to therules of BigTwo game	NF-P2
3.4.8	The product shall always be functional with a relevant web browser.	NF-P3
3.5.4	The new release of the product shall not cause the previous version to fail.	NF-O1

Table 8: Non-Functional Traceability Matrix (continued)

Requirement #	Description	Test ID(s)
3.8	The product shall not use any words or graphics that are offensive to people with any culture	NF-C1
3.9.1	The product shall not violate any laws.	NF-Le1
3.9.2	The product shall follow the MIT Open License.	NF-C2
3.10.2	The product shall not involve any gambling elements.	NF-H1

4 Tests for Proof of Concept

4.1 Game testing

4.1.0.1 Main Menu Section

1. POC-G-1

   Type: Functional, Dynamic, Manual

   Initial State: Game web page loaded

   Input: N/A

   Output: The game automatically initialize and show the main menu section on the web page.

   How test will be performed: The user will attempt to go to the web page of the game. Once the web page is connected with the web server, the game will automatically initialize and show the main menu section with two buttons: "Start the game" and "Help".

4.1.0.2 Help Module

1. POC-G-2

   Type: Functional, Dynamic, Manual

   Initial State: Main Menu Section loaded

   Input: The user click on "Help" button on the main menu section

Output: The game will go to the Help section page and display the rules of how to play BigTwo on the screen.

How test will be performed: The user will attempt to click the "Help" button on the main menu section of the game. Once the button is click, the game will go to the help page and display the rule of BigTwo in words on the screen.

4.1.0.3 Play the game

1. POC-G-3

   Type: Functional, Dynamic, Manual

   Initial State: Main Menu Section loaded

   Input: The user click on "Start the game" button on the main menu seciton

   Output: The game will allow the player to enter and play the game.

   How test will be performed: The user will attempt to click on the "Start the game" button on the main menu section. Once the button is clicked, the application will run the game modules at back end in the web server for the player to enter and play the game.

5 Comparison to Existing Implementation

Compare with the existing implementation of the web game on http://www.onlinesologames.com/bigtwo, our project will have similar functionalities with this web game except for the following differences:

- We will have a main menu section once the web page is loaded, instead of entering the game directly.
- In the main menu section, we will have two buttons; "Help" to direct the player to the rules of BigTwo, and "Start the game" for the user to start the game.

6 Unit Testing Plan

This project will incorporate the PyTest framework for unit testing.

6.1 Unit testing of internal functions

Internal functions of the program will be methods that will have return values or execute a specific task. Unit tests for internal functions will involve having various inputs for the methods and comparing the method output to the expected output. Unit tests will include various inputs for methods consisting of normal inputs, boundary inputs, and inputs that will generate exceptions and errors for the program. The unit tests will display the test cases that have passed, and the test cases that have failed with the traceback to the error. Our goal is to cover as much as possible in order to make sure that we test all functions adequately. The goal is to achieve 80% for branch coverage.

6.2 Unit testing of output files

This project generates no output files, thus no unit testing will be performed for this section.

References

7 Appendix

Not applicable. There are no added resources that are included in this documentation.

7.1 Symbolic Parameters

1. **NUMBER_OF_CARDS_IN_DECK**: **13**
2. **NUMBER_OF_PLAYERS**: 4
3. **Suit** : Spades, Hearts, Clubs, Diamonds
4. **Card Rank** :

Card Rank	Value
2	12
A	11
K	10
Q	9
J	8
10	7
9	6
8	5
7	4
6	3
5	2
4	1
3	0

7.2 Usability Survey Questions

1. Were you able to successfully play a game of BigTwo from start to finish?
2. Was the game easy to play?
3. Were the response times to your actions acceptable?
4. Did you run into any problems during the game?
5. Provide any feedback to the developers.