

SE 3XA3: Test Plan

Title of Project

Team 12, Team Name
Eric Thai thaie1
Student 2 name and macid
Student 3 name and macid

March 5, 2021

Contents

1	General Information	1
1.1	Purpose	1
1.2	Scope	1
1.3	Acronyms, Abbreviations, and Symbols	1
1.4	Overview of Document	1
2	Plan	2
2.1	Software Description	2
2.2	Test Team	2
2.3	Automated Testing Approach	2
2.4	Testing Tools	2
2.5	Testing Schedule	2
3	System Test Description	3
3.1	Tests for Functional Requirements	3
3.1.1	User Interface Testing	3
3.1.2	Game Mechanics Testing	5
3.2	Tests for Nonfunctional Requirements	7
3.2.1	Gaming Quality	7
3.2.2	System Quality	7
3.3	Traceability Between Test Cases and Requirements	9
4	Tests for Proof of Concept	9
4.1	Movement	9
4.2	Score	10
5	Comparison to Existing Implementation	11
6	Unit Testing Plan	11
6.1	Unit testing of internal functions	11
6.2	Unit testing of output files	12
7	Appendix	13
7.1	Symbolic Parameters	13
7.2	Usability Survey Questions?	13

List of Tables

1	Revision History	ii
2	Table of Abbreviations	1
3	Table of Definitions	1
4	Traceability Between Test Cases and Requirements . .	9

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1 General Information

1.1 Purpose

The purpose of this test plan is to ensure that all functional and non-functional requirements are met as stated in the SRS. It also describes the testing strategy for this project and the testing tools that will be used.

1.2 Scope

This document will outline the project's testing plan including the tools and approaches that will be used and a testing schedule. The document will also list the areas of the system that need to be tested to ensure the functional and non-functional requirements are satisfied. Additionally, there will be information on testing the POC and unit testing.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	Definition
SRS	Software requirement Specification
POC	Proof of Concept
UI	User Interface

Table 3: **Table of Definitions**

Term	Definition
Mocha	JavaScript testing library

1.4 Overview of Document

This Document will go through the test plan for our Improved version of the arcade game Asteroids. The document will go over our plans on testing the functional and non-functional requirements. It will also include a plan to test the POC and a unit testing plan.

2 Plan

2.1 Software Description

This software is a improved version of the classic arcade game Asteroids written in Javascript/HTML. This version of the game will refactor the code to improve the game's UI and allow for 2 players to play the game together.

2.2 Test Team

The test team includes Eric Thai, Tianzheng Mai, Linqi Jiang, and Junhong Chen

2.3 Automated Testing Approach

Using Mocha we will create a set of Unit tests in Javascript to test the function of the software. This will handle the testing of the game's core logic. The front-end will be manually tested as we concluded that given it is a game, manual testing with user interaction is necessary. A manual test suite will be created to make sure that the software visually displays the correct result given a set of user inputs.

2.4 Testing Tools

Mocha is a Javascript testing library that allows for the creation of unit tests. Mocha can help us determine if the correct output of a function is produced given different inputs. It also keeps track of the time each test takes to complete.

2.5 Testing Schedule

See Gantt Chart at the following url: [Gantt Chart](#)

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 User Interface Testing

Home Page

1. FR-UI-1

Type: Functional, Dynamic, Manual.

Initial State: A new empty tap on the browser.

Input: The project HTML link of the browser.

Output: The browser displays the home page of the Asteroid game.

How test will be performed: The testers must have a stable internet connection and a valid browser on the system to access this game.

Battlefield

1. FR-UI-2

Type: Functional, Manual

Initial State: Home Page

Input: The user selects a game mode and press space to start.

Output: The game page will display a battlefield with a suitable size on the screen.

How test will be performed: After the tester selects a game mode and press space to start, the game will generate a battlefield where the user can control the spaceship to fly on a battlefield with a suitable size on the screen.

Spaceship

1. FR-UI-3

Type: Functional, Manual

Initial State: Home Page

Input: The user selects a game mode and press space to start.

Output: The game page will display a spaceship on the screen.

How test will be performed: After the tester selects a game mode and press space to start, the game will generate a spaceship.

User Manual

1. FR-UI-4

Type: Functional, Manual

Initial State: Home Page

Input: The user selects the user manual option on the home page.

Output: The screen will display a user manual page.

How test will be performed: After clicking on the user manual option on the home page, the screen will display a list of game instructions and rules on the user manual page.

Stop Option

1. FR-UI-5

Type: Functional, Manual

Initial State: Game Page

Input: The user clicks on the stop button during the game.

Output: The game will pause after the user clicking on the stop button.

How test will be performed: The tester clicks on the stop button, and the game will pause.

Score Test

1. FR-UI-6

Type: Functional, Dynamic, Manual

Initial State: Game Page

Input: The user presses the space key to start the game.

Output: After the game starts, the game will display the scores of the game on the right-hand corner.

How test will be performed: The testers need to select one player game mode or two player game mode to view if the scores are displayed on the asteroid screen.

3.1.2 Game Mechanics Testing

Spaceship Speed Test

1. FR-GM-1

Type: Functional, Static

Initial State: Game Page

Input: The tester uses the keyboard to play the game.

Output: The spaceship should fly at the same speed as the same value of speed function in game.js and game2.js.

How test will be performed: The testers need to play the game multiple times to test if the speed of the spaceship is the same speed as the value of the speed function in game.js and game2.js.

Score Record

1. FR-GM-2

Type: Functional, Dynamic, Manual

Initial State: Game Page

Input: The tester controls the spaceship attack the asteroids

Output: The scoring board will update the score simultaneously when the spaceship attacks the asteroids.

How test will be performed: The tester controls the keyboard to make the spaceship attack the asteroids and check if the scoring board is updated simultaneously.

Asteroid Allocation:

1. FR-GM-3

Type: Functional, Dynamic, Manual

Initial State: Game Page

Input: The user selects game mode to start.

Output: After the game started, the asteroid will be distributed in random areas of the battlefield.

How test will be performed: The tester plays the game manually and tests if the asteroids are distributed in random areas of the battlefield.

Lives Test:

1. FR-GM-4

Type: Functional, Dynamic, Manual

Initial State: Game Page (Either 1 player mode or 2 player mode)

Input: Move the spaceship to hit an asteroid

Output: The lives of the spaceship will be reduced by 1 or display a game over message.

How test will be performed: The tester need to control the keyboard to move the spaceship to hit an asteroid. If the original lives are greater than 1, the number of lives will be reduced by 1 after getting hit by an asteroid. If the original lives is 1, the screen will display a game over message after the spaceship is hit by an asteroid.

3.2 Tests for Nonfunctional Requirements

3.2.1 Gaming Quality

Usability and Humanity Requirement

1. NFR-UH-1

Type: Non-functional, Manual

Initial State: Game Page

Input/Condition: Multiple users who play the game.

Output/Result: User experience of the game effectiveness, efficiency and the overall satisfaction.

How test will be performed: Multiple testers will be invited to play the game and provide feedback experience in the game effectiveness, efficiency and the overall satisfaction.

Performance Requirement

1. NFR-PR-1

Type: Non-functional, manual

Initial State: Game Page

Input/Condition: The user plays the game(either one player mode or two player mode) on the keyboard.

Output/Result: The game is displayed on the screen.

How test will be performed: Multiple testers will be invited to play the game in different modes and test whether the game can interact efficiently between the keyboard control and screen.

3.2.2 System Quality

Operation and environment requirement

1. NFR-OE-1

Type: Non-functional, Dynamic, Manual

Initial State: Valid browsers

Input/Condition: The user tries to open the game in different popular browsers such as Firefox, Google Chrome, Microsoft Edge.

Output/Result: The game should be operated and displayed in different browsers.

How test will be performed: The tester runs the game in different browsers, but the game in different browsers should have a similar performance.

Maintainability and Support Requirement

1. NFR-MS-1

Type: Non-functional, Dynamic, Manual

Initial State: Windows system or macOS system

Input/Condition: The user runs the game in the Windows system and macOS system separately.

Output/Result: The game should be operated correctly in both Windows systems and macOS systems.

How test will be performed: The tester runs the game in both the Windows and macOS systems and the game should have a similar performance.

3.3 Traceability Between Test Cases and Requirements

Table 4: Traceability Between Test Cases and Requirements

Test Cases	Requirement
FR-UI-1	Test whether the home page is displayed correctly on the browser
FR-UI-2	Test whether the battlefield is constructed as a suitable size on the screen
FR-UI-3	Test whether the game will generate a spaceship once it starts
FR-UI-4	Test whether the user manual is accessible on the main page
FR-UI-5	Test whether the stop option can pause the game
FR-UI-5	Test whether the scoring board is displayed properly on the game screen
FR-GM-1	Test whether the speed of the spaceship is the same as the value of the speed function in game.js and game2.js
FR-GM-2	Test whether the scoring board record the scores correctly
FR-GM-3	Test whether the asteroid is distributed correctly in the random areas of the battlefield
FR-GM-4	Test whether the lives of the spaceship displayed correctly on the screen
NFR-UH-1	Test the Usability and Humanity of the game
NFR-PR-1	Test the Performance of the game
NFR-OE-1	Test the Operation and Environment of the game
NFR-MS-1	Test the Maintainability and Support of the game

4 Tests for Proof of Concept

4.1 Movement

Movement Testing

1. T-M-1

Type: Dynamic, Manual

Initial State: pre-game stage

Input: User presses W-A-D, Shift, Up arrow, Right arrow, Left arrow and Space keys on the keyboard.

Output: User successfully controls spaceship's movement and changes spaceship's position.

How test will be performed: In the single-player mode, W-key pressed will let the spaceship accelerate and move forward. A-key will let the spaceship rotate to the left, and D-key pressed will let the spaceship rotate to the right. Space-key pressed will let the spaceship shoot a bullet forward. In two-player mode, the method of controlling the spaceship1 is the same as described above, while spaceship2 will be controlled by the Up arrow, Down arrow, Right arrow, Left arrow, and Shift keys. Up arrow key pressed will let the spaceship2 accelerate and move forward. Right arrow key pressed and left arrow key pressed will let the spaceship2 rotate to the left and rotate to the right, respectfully. Shift key pressed will let the spaceship2 shoot a bullet forward. Two McMaster University students will collaborate with us to do the testing. After we introduce how to play the game, they will play the game by pressing the keys described above on the keyboard. We will make sure each key will be pressed at least 20 times.

4.2 Score

Score Testing

1. T-S-1

Type: Functional, Manual

Initial State: The Game is started by the user interface.

input: The Game is played by clicking the play button from the user.

output: The final scores is based on the number of asteroids that the spaceship has destroyed.

How the test will be performed: A sample playthrough is made by the test at the beginning, the number of asteroids that destroyed will be recorded. The scores will be based on this number and the test will check if the output score matches this score.

5 Comparison to Existing Implementation

The existing implementation had no testing on the project. Thus, it is essential to create a set of unit test cases to validate the existing project and it can be also applied to the new features developed by our project. As extensive unit test cases are done to every component of the product, the usability and maintainability will be significantly raised which makes the product more reliable.

6 Unit Testing Plan

6.1 Unit testing of internal functions

The javascript testing framework Jasmine will be applied for unit testing for internal functions in our project. There are many different types of functions that need to be tested. First of all, for the functions that return a value, the assertion method(`expect.toBe`)will be applied to check the output, exceptions will be raised according to the functions. Secondly, for the functions that do not return a value but include some kind of transitions, the test will initialize different state variables, applied to the functions, and then the assertion method can be applied to the expected results to check whether the transition functions work properly. The results will be shown in the console by running `jest`. Once the file is saved, `jest` will inform the developer which test is failed, which is helpful to find potential issues before committing changes. For example, to check if the “HELP” button works properly, the

test will apply few state variables and use the assertion method to check if the variable's state is changed since it is hard to test the “HELP” button directly.

6.2 Unit testing of output files

There are no output files from our project.

7 Appendix

Additional information.

7.1 Symbolic Parameters

N/A

7.2 Usability Survey Questions?

- Is the difficulty of the game acceptable for you?
- How long does it take for the user to load the game on the browser?
- Are you satisfied with the control of the keys?
- Does the user manual provide enough information for you to understand the game?
- Do you enjoy the User Interface (UI) of the game?
- Does the game have good performance on your browser? Is there any delay in the game process?