SE 3XA3: Test Plan
Mini-Arcade

Team #104
Andrew Hum, 400138826
Arshan Khan, 400145605
Jame Tran, 400144141
William Lei, 400125240

February 27, 2020

Contents

List of Tables

List of Figures

1 General Information

1.1 Purpose

The purpose of testing our project is to verify that it meets the requirements outlined in the 'Software Requirements Specification' and ensure that it is implemented correctly.

1.2 Scope

The test plan develops a baseline for testing the functionality and correctness of Mini-Arcade. Its core objective is to verify that the games run correctly and efficiently all with a single click utilizing the launcher. The test plan documents will highlight what is to be tested of our project, testing methods and what resources we will use to test our software.

1.3 Acronyms, Abbreviations, and Symbols

1.4 Overview of Document

This document will outline a detailed testing plan with the tools that will be utilized and the approximated schedule of testing. It will also give in-depth test cases and the method of testing for the functional requirements, non-functional requirements, the proof of concept tests and the unit-testing plan.

2 Plan

2.1 Software Description

The software is a launcher for a selection of games for the user to play. These games are updated from their original versions to be more visually pleasing and challenging.

2.2 Test Team

The test team is composed of all team members: Andrew Hum, Arshan Khan, Jame Tran, and William Lei.

Table 1: **Revision History**

Date	**Version**	**Notes**
2/24/2020	1.0	Andrew and Arshan divided the project into workable parts for group members and began the rough draft of sections 1, 2, 5
2/26/2020	1.1	Andrew completed sections 1 and 5
2/27/2020	1.2	Andrew revised sections 1 and 5 for grammatical errors
Date 2	1.1	Notes

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
Abbreviation	Definition

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1

2.3 Automated Testing Approach

The tests will be automated by pytest because it is very popular and allows "assert rewriting".

2.4 Testing Tools

2.5 Testing Schedule

See Gantt Chart at the following url:
../../ProjectSchedule/3XA3-ProjSched.gan.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.1.2 Area of Testing2

...

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:
   Initial State:
   Input/Condition:
   Output/Result:
   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2 Area of Testing2

...

5 Unit Testing Plan

The pytest library will be used for the unit testing of our project.

5.1 Unit testing of internal functions

To efficiently use unit-testing for our project, we will use hard-coded, expected, and unexpected, inputs for individual functions and methods. These functions and methods will then provide output, and we will verify that the resulting output is correct or that the program handles the unexpected input correctly. For example, telling the game that the game was won, and the expected output should be the end-game screen. As games are more difficult to completely test with unit tests, we will only test the functions that can be tested by providing an expected and unexpected output with input values

relating to a current state or completed event. To cover a wide range of scenarios, the input variables will test both expected output, and reaction to incorrect/unexpected input values. There will be no need for stubs or drivers to test our project. To ensure high-quality coverage, we will be using testing coverage metrics. Our goal is to cover a minimum of 60% of the project with unit tests alone, derived by the total lines of code in the project divided by the number of lines covered by the test cases.

5.2 Unit testing of output files

In-depth testing of the output files using unit testing will be not applicable for our project, and any unit tests to test output files would prove to be not useful and ineffective in both coverage and effective use of time.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

This is a section that would be appropriate for some teams. A possible set of questions to ask beta testers would include:

- What game did you play first?
- What did you think of that game?
- How long did you play?
- Would you play again?
- Did you play any other games?
- Was it easy to get started?