SE 3XA3: Test Plan
Mini-Arcade

Andrew Hum
huma3
400138826

William Lei
leim5
400125240

Arshan Khan
khana172
400145605

Jame Tran
tranj52
400144141

April 5, 2020

Contents

List of Tables

List of Figures

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
2/27/2020	1.3	William completed section 3.1
2/27/2020	1.4	Jame completed section 3.2 and section 4
2/28/2020	1.5	Revision 0
4/3/2020	1.6	Andrew - Revision 1
4/3/2020	1.7	William - Revision 1
4/4/2020	1.8	Arshan - Revised and edited document

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

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
FDM	Functional, Dynamic and Manual Testing
FPS	Frames per Second

Table 3: **Table of Definitions**

Term	**Definition**
Functional Testing	Testing derived from the functional requirements of the software.
Dynamic Testing	Testing through executing test cases during runtime.
Manual Testing	Testing conducted by providing manual inputs and people checking for outputs.

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 challenging and visually pleasing.

2.2 Test Team

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

2.3 Automated Testing Approach

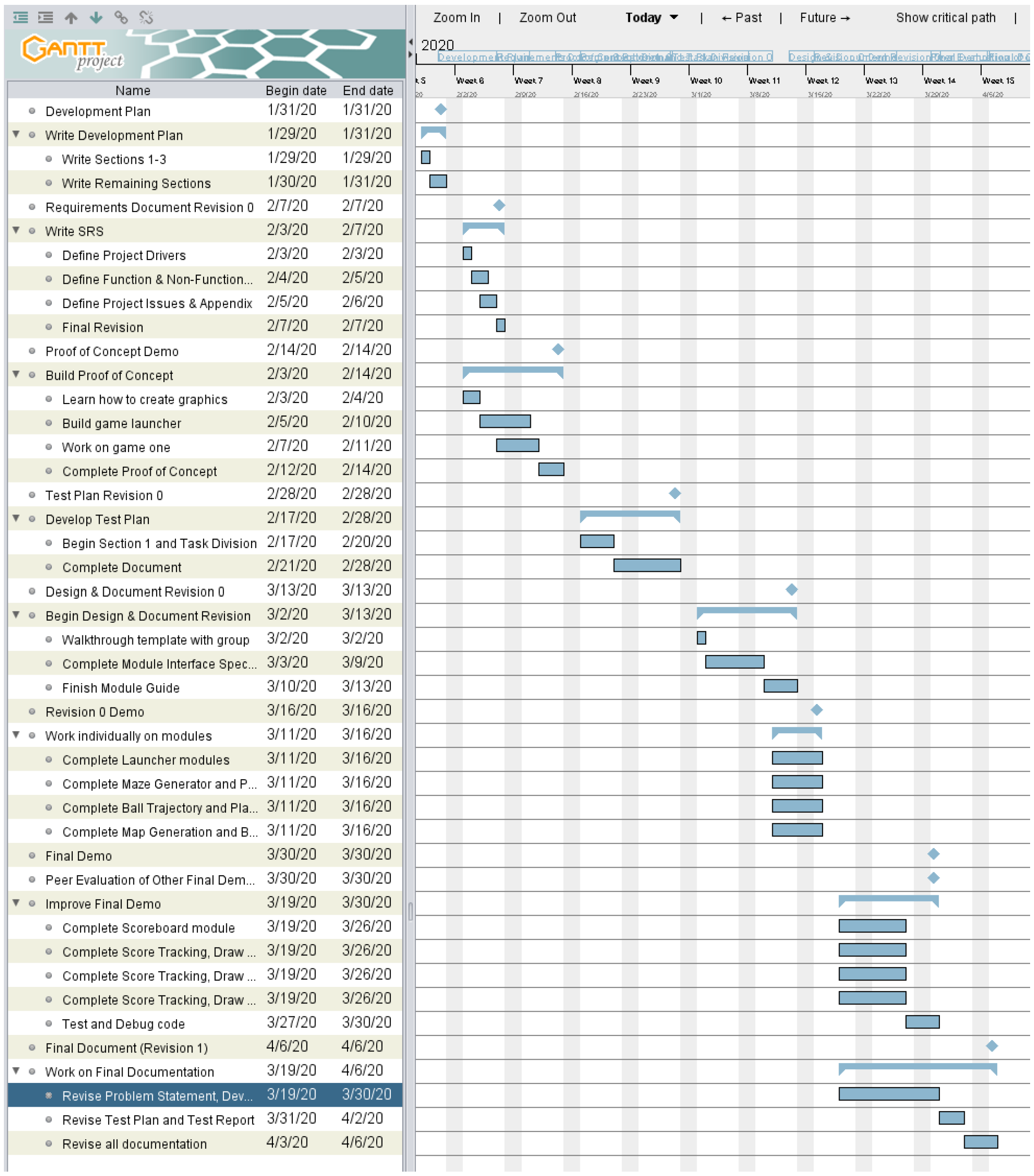
The tests will be automated by Pytest because it is a very popular test automation platform and provides detailed assertion error messages.

2.4 Testing Tools

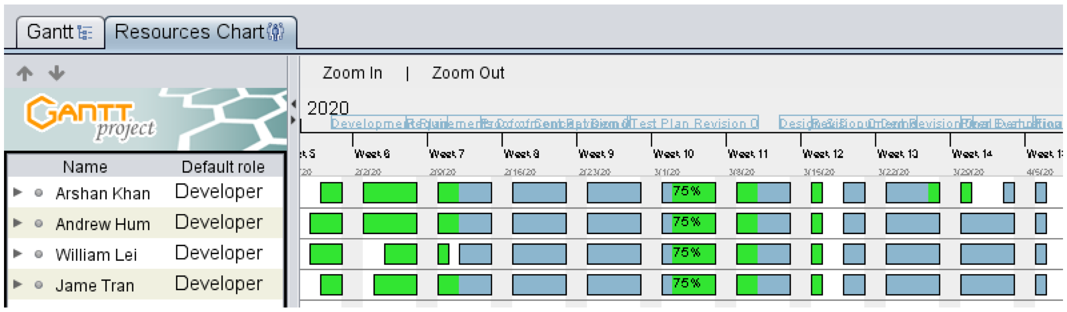
The main tool in our testing will be Pytest as it can cover a wide range of tests. Most of the testing will be done through the IDE but some testing will be done by asking users to install and play with the system.

2.5 Testing Schedule

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



Name	Begin date	End date
Development Plan	1/31/20	1/31/20
Write Development Plan	1/29/20	1/31/20
Write Sections 1-3	1/29/20	1/29/20
Write Remaining Sections	1/30/20	1/31/20
Requirements Document Revision 0	2/7/20	2/7/20
Write SRS	2/3/20	2/7/20
Define Project Drivers	2/3/20	2/3/20
Define Function & Non-Function...	2/4/20	2/5/20
Define Project Issues & Appendix	2/5/20	2/6/20
Final Revision	2/7/20	2/7/20
Proof of Concept Demo	2/14/20	2/14/20
Build Proof of Concept	2/3/20	2/14/20
Learn how to create graphics	2/3/20	2/4/20
Build game launcher	2/5/20	2/10/20
Work on game one	2/7/20	2/11/20
Complete Proof of Concept	2/12/20	2/14/20
Test Plan Revision 0	2/28/20	2/28/20
Develop Test Plan	2/17/20	2/28/20
Begin Section 1 and Task Division	2/17/20	2/20/20
Complete Document	2/21/20	2/28/20
Design & Document Revision 0	3/13/20	3/13/20
Begin Design & Document Revision	3/2/20	3/13/20
Walkthrough template with group	3/2/20	3/2/20
Complete Module Interface Spec...	3/3/20	3/9/20
Finish Module Guide	3/10/20	3/13/20
Revision 0 Demo	3/16/20	3/16/20
Work individually on modules	3/11/20	3/16/20
Complete Launcher modules	3/11/20	3/16/20
Complete Maze Generator and P...	3/11/20	3/16/20
Complete Ball Trajectory and Pla...	3/11/20	3/16/20
Complete Map Generation and B...	3/11/20	3/16/20
Final Demo	3/30/20	3/30/20
Peer Evaluation of Other Final Dem...	3/30/20	3/30/20
Improve Final Demo	3/19/20	3/30/20
Complete Scoreboard module	3/19/20	3/26/20
Complete Score Tracking, Draw ...	3/19/20	3/26/20
Complete Score Tracking, Draw ...	3/19/20	3/26/20
Complete Score Tracking, Draw ...	3/19/20	3/26/20
Test and Debug code	3/27/20	3/30/20
Final Document (Revision 1)	4/6/20	4/6/20
Work on Final Documentation	3/19/20	4/6/20
Revise Problem Statement, Dev...	3/19/20	3/30/20
Revise Test Plan and Test Report	3/31/20	4/2/20
Revise all documentation	4/3/20	4/6/20



Name	Default role
Arshan Khan	Developer
Andrew Hum	Developer
William Lei	Developer
Jame Tran	Developer

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 General Navigation

1. FR-N-1
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Scoreboard
   Output: Scoreboard opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

2. FR-N-2
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Maze
   Output: The mini-game Maze opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3. FR-N-3
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Flappy
   Output: The mini-game Flappy opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

4. FR-N-4
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Pong
   Output: The mini-game Pong opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

5. FR-N-5
   Type: FDM
   Initial State: Main Screen

Input: User clicks on close button
Output: The software will be terminated.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

6. FR-N-6
Type: FDM
Initial State: Scoreboard Screen
Input: User clicks on Maze
Output: The scoreboard screen will display the scoreboard for Maze.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

7. FR-N-7
Type: FDM
Initial State: Scoreboard Screen
Input: User clicks on Pong
Output: The scoreboard screen will display the scoreboard for Pong.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

8. FR-N-8
Type: FDM
Initial State: Scoreboard Screen
Input: User clicks on Flappy
Output: The scoreboard screen will display the scoreboard for Flappy.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

9. FR-N-9
Type: FDM
Initial State: Maze - Menu Screen
Input: User clicks on Help
Output: The screen will display the instructions for how to play the mini-game.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

10. FR-N-10
Type: FDM

Initial State: Flappy - Menu Screen
Input: User clicks on Help
Output: The screen will display the instructions for how to play the mini-game.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

11. FR-N-11
~~Type: FDM~~
~~Initial State: Pong - Menu Screen~~
~~Input: User clicks on Help~~
~~Output: The screen will display the instructions for how to play the mini-game.~~
~~How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.~~

12. FR-N-12
Type: FDM
Initial State: Maze - Menu Screen
Input: User clicks on ~~Back~~ Home
Output: The Main Screen opens and is displayed on the screen.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

13. FR-N-13
Type: FDM
Initial State: Flappy - Menu Screen
Input: User clicks on Back
Output: The Main Screen opens and is displayed on the screen.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

14. FR-N-14
Type: FDM
Initial State: Pong - Menu Screen
Input: User clicks on Quit Game
Output: The Launcher Screen opens and is displayed on the screen.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3.1.2 Mini-Game - Maze

1. FR-MGM-1
   Type: FDM
   Initial State: Maze - Menu Screen
   Input: User ~~clicks on a difficulty level~~ selects Play
   Output: ~~A maze will be displayed on the screen~~ Three different difficulties will be displayed on the screen: Easy, Medium. Hard.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

2. FR-MGM-2
   Type: FDM
   Initial State: Maze - Game Screen
   Input: User clicks on ~~home~~ menu
   Output: Menu screen of Maze will be displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3. FR-MGM-3
   Type: FDM
   Initial State: Maze - Menu Screen
   Input: User clicks on a specific difficulty level, then clicks home, and repeats this for 5 times in total
   Output: A maze will be displayed on the screen every time the user clicks a difficulty level, and there should be no patterns for when a specific maze will be displayed.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

4. FR-MGM-4
   Type: FDM
   Initial State: Maze - Game Screen
   Input: User clicks a movement key from the keyboard
   Output: The object will move according to the key-movement mapping and the movement will be displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

5. FR-MGM-5
   Type: FDM

Initial State: Maze - Game Screen
Input: Object reaches the end of the maze through a movement
Output: A score (base on time elapsed) along with high score will be displayed on the end game screen.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

6. ~~FR-MGM-6~~
   ~~Type: FDM~~
   ~~Initial State: Maze - End Game Screen~~
   ~~Input: User clicks on Next~~
   ~~Output: A maze will be displayed on the screen~~.
   ~~How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.~~

7. FR-MGM-7
   Type: FDM
   Initial State: Maze - End Game Screen
   Input: User clicks on Return
   Output: The Menu Screen opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3.1.3 Mini-Game - Flappy

1. FR-MGF-1
   Type: FDM
   Initial State: Flappy - Menu Screen
   Input: User clicks on start
   Output: The game will be initialized/started and the game screen will be opened and displayed
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

2. FR-MGF-2
   Type: FDM
   Initial State: Flappy - Game Screen
   Input: User controlling the character to make sure it will not collide with any object
   Output: There will be randomly generated objects approaching toward the character, and their speed and amount generated will be increased as time elapses.

How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3. FR-MGF-3
   Type: FDM
   Initial State: Flappy - Game Screen
   Input: User clicks space key for 5 times separated by a short period
   Output: The character will move up a constant amount every time the space key is being clicked.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

4. FR-MGF-4
   Type: FDM
   Initial State: Flappy - Game Screen
   Input: User controls the character to collide with an object
   Output: A score (base on time elapsed) along with high score will be displayed on the end game screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

5. FR-MGF-5
   Type: FDM
   Initial State: Flappy - End Game Screen
   Input: User clicks on Restart
   Output: The game will be initialized/started and the game screen will be opened and displayed.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

6. FR-MGF-6
   Type: FDM
   Initial State: Flappy - End Game Screen
   Input: User clicks on Return
   Output: The Menu Screen opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3.1.4 Mini-Game - Pong

1. FR-MGP-1
   Type: FDM
   Initial State: Pong - Main Menu Screen
   Input: User clicks on ~~Single Player~~ Difficulty Level and Max Score
   Output: The game will display the selected difficulty level and maximum score on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

2. FR-MGP-2
   Type: FDM
   Initial State: Pong - Main Menu Screen
   Input: User clicks on 'Begin'
   Output: The game will start, taking no longer than 10 seconds to begin.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

3. FR-MGP-3
   ~~Type: FDM~~
   ~~Initial State: Pong - Game Screen~~
   ~~Input: User inputs an integer between 1 to 10.~~
   ~~Output: The game will be initialized or started. How test will be performed: Assuming difficulty and max score have been chosen, the application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.~~

4. FR-MGP-4
   Type: FDM
   Initial State: Pong - Game Screen
   Input: User clicks a movement key
   Output: The corresponding paddle will move according to the key-movement mapping and the movement will be displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen.

5. FR-MGP-5
   Type: FDM
   Initial State: Pong - Game Screen

Input: User controls the paddle to hit the ball until the ball reaches the boundary on either side (i.e. without hitting a paddle)
Output: The score of the scoring side will be increased by 1 and the change will be displayed on the game screen.
How test will be performed: The application will be opened and the user will manually provide inputs to the software and observes for the output of the software on the screen. Once for the user to score and once for the AI to score.

6. FR-MGP-6
   Type: FDM
   Initial State: Pong - Game Screen
   Input: User control the paddle to hit the ball until either side reaches the max score
   Output: The score between the two players will be displayed on the end game screen.
   How test will be performed: The user will play the game until the max score is reached.

7. FR-MGP-7
   Type: FDM
   Initial State: Pong - End Game Screen
   Input: User clicks on New Game
   Output: The Main Menu Screen will be displayed.
   How test will be performed: While on the End Game Screen, the user clicks on the New Game button.

8. FR-MGP-8
   Type: FDM
   Initial State: Pong - End Game Screen
   Input: User clicks on Quit Game
   Output: The Launcher opens and is displayed on the screen.
   How test will be performed: While on the End Game Screen, the user clicks on the Quit Game button.

9. FR-MGP-9
   Type: FDM
   Initial State: Pong - Game Screen
   Input: The user pauses the game
   Output: This will display the Pause Game Screen.
   How test will be performed: While the game is running, the user presses the key to pause the game.

10. FR-MGP-10
    Type: FDM
    Initial State: Pong - Pause Game Screen
    Input: The user chooses to resume the current game
    Output: This will redisplay the Game Screen at the same point where the Pause Game Screen was invoked.
    How test will be performed: While on the Pause Game Screen, the user clicks on the Resume Game button.

11. FR-MGP-11
    Type: FDM
    Initial State: Pong - Pause Game Screen
    Input: The user chooses to start a new game
    Output: This will take the user back to the Main Menu Screen.
    How test will be performed: While on the Pause Game Screen, the user clicks on the New Game button.

12. FR-MGP-12
    Type: FDM
    Initial State: Pong - Pause Game Screen
    Input: The user chooses to return to the Launcher
    Output: This will take the user back to the Launcher Screen.
    How test will be performed: While on the Pause Game Screen, the user clicks on the Quit Game button.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel

1. NFT-1

   Type: FDM

   Initial State: The program is launched on default settings, with default performance settings on the computer, at the start screen of one of three games (Flappy, Maze, or Pong).

   Input/Condition: Users will be asked to play the game for 5 minutes.

   Output/Result: Average FPS displayed by pygame get_fps() is higher than 30.

   How test will be performed: A test group of people who have graduated high school or have an equivalent GED will be asked to play the games (Maze, Pong, and Flappy

Bird) for a total of 5 minutes. The average FPS will be recorded by using the built-in method get_fps() in pygame. The majority of the test group must have an average FPS of 30 or above for the test to be considered successful.

2. NFT-2

   Type: FDM

   Initial State: The program is launched on default settings, with default performance settings on the computer, at the start screen of one of three games (Flappy, Maze, or Pong).

   Input: Users will be asked to play the game for 5 minutes.

   Output: Users in the test group will be asked to evaluate the existing implementation of the games' visual appeal.

   How test will be performed: A test group of people who have graduated high school or have an equivalent GED will be asked to play the games (Maze, Pong, and Flappy Bird) for a total of 5 minutes. They will then complete a survey that has a list of criteria related to each games' visual design, with each criterion attached to a Likert scale. The users will be asked to rate the program based on the criteria provided using the scales provided. The average rating for each criterion will be calculated. The average must be above 3 on each criterion for the test to be considered successful.

3.2.2 Usability and Humanity

1. NFT-3

   Type: FDM

   Initial State: The program is launched on default settings, with default performance settings on the computer, at the start screen of one of three games (Flappy, Maze, or Pong).

   Input: Users will be asked to play the game for 5 minutes

   Output: Users in the test group will be asked to evaluate the existing implementation and intrusiveness of the games' UI.

   How test will be performed: A test group of people who have graduated high school or have an equivalent GED will be asked to play the games (Maze, Pong, and Flappy Bird) for a total of 5 minutes. They will then complete a survey that has a list of criteria related to each games' UI design, with each criterion attached to a Likert scale. The users will be asked to rate the program based on the criteria provided using the scales provided. The average rating for each criterion will be calculated. The average must be above 3 on each criterion for the test to be considered successful.

2. NFT-4

   Type: FFDM

   Initial State: The program is launched on default settings, with default performance settings on the computer, at the start screen of one of three games (Flappy, Maze, or Pong).

   Input: Users will be asked to play the game for 5 minutes.

   Output: Users in the test group will be asked to evaluate the controls and game-play of the existing implementation of the games.

   How test will be performed: A test group of people who have graduated high school or have an equivalent GED will be asked to play the games (Maze, Pong, and Flappy Bird) for a total of 5 minutes. Afterwards, each user will complete a small quiz on the respective game they played, with questions about core game-play mechanics. The quizzes will be marked, and an average score of 80% on the quizzes is required for success.

3.3 Performance

1. NFT-5

   Type: FDM

   Initial State: The program is launched on default settings, with default performance settings on the computer.

   Input: The user will be asked to launch one of three games (Flappy, Maze, or Pong).

   Output: The requested action is performed in less than 30 seconds.

   How test will be performed: Either a group of users will be asked to perform the task or the task will be iterated multiple times. The average time elapsed between launching the game and the game being fully functional will be recorded. The requested action must be performed under 30 seconds for 80% of the times the test is performed.

2. NFT-6

   Type: FDM

   Initial State: The program is launched on default settings, with default performance settings on the computer, currently playing one of three games (Flappy, Maze, or Pong).

   Input: The user will be asked to make an input into the game.

   Output: The game will be updated within a quarter second of user input.

How test will be performed: Either a group of users will be asked to perform the task or the task will be iterated multiple times. The average time elapsed between making an input and the game updating will be recorded. The requested action must be performed within a quarter second for 80% of the times the test is performed.

3.4 Operation and Environmental Requirements

1. NFT-7

   Type: FDM

   Initial State: A computer powered on, without Mini-Arcade currently installed.

   Input: The user will be asked to install the program.

   Output: The majority of users can install the program without outside assistance.

   How test will be performed: Either a group of users will be asked to perform the task or the task will be iterated multiple times. The program installation success and the amount of time it took will be recorded. The test is considered successful if 80% of users can install the program without assistance. This test will be repeated on multiple computers with varying hardware and operating systems

4 Tests for Proof of Concept

4.1 Area of Testing

Since many of the above tests for non-functional requirements also cover issues in the Proof of Concept, these tests will focus on testing external modules

1. PC-1

   Type: FDM

   Initial State: A computer with VSCode and Python 3 installed.

   Input: The user will attempt to install the Pygame module

   Output: The Pygame module will be available for use in VSCode

   How test will be performed: The user will use Pip to install the Pygame module in a virtual environment. The user will then attempt to import the Pygame module and run a sample program that uses its functionality. If the program can run and display full functionality, the test is a success. This test will be repeated across multiple computers.

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

6.1 Usability Survey Questions

A set of questions to ask potential system 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?
- How would you improve the experience?
- Was the system easy to understand?
- Was the system easy to navigate?