SE 3XA3: Test Plan
Title of Project

Team #1, Team Rex
Anjola Adewale and adewaa1
Chelsea Maramot and maramotc
Sheridan Fong and fongs7

March 11, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

This document ...

1 General Information

1.1 Purpose

1.2 Scope

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
Abbreviation1	Definition1
Abbreviation2	Definition2

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1
Term2	Definition2

1.4 Overview of Document

2 Plan

2.1 Software Description

2.2 Test Team

The test team consists of Anjola Adewale, Sheridan Fong, and Chelsea Maramot. The testers will divide the tests according to the features of CDR and cover all the necessary types of testing.

2.3 Automated Testing Approach

The software required is a Python automated testing framework and a text editor to modify code. The personnel responsible for testing setup should have sufficient knowledge of the Python testing framework in order to write a series of test cases. In contrast, for manual testing, the user will not need previous knowledge of Python and the implementation of the game. The Python automated testing framework will be unittest which provides its own documentation found online. Unittest provides tools creating and running test case. Each test case will begin with "test". Each test case will implement an assert statement to check expected results and raise exceptions. The results of these tests will be displayed through the terminal.

2.4 Testing Tools

Asides from the output file, the game has a graphical user interface generated through PyGame. A user conducted black box testing will be done to test the interface and its functionalities. The user will play the game, testing each feature. If the game is successfully played by the user and all functionalities stated within the SRS have been achieved, then the user interface passes the test.

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Settings Page Tests

1. test-id1

   Type: Functional, Dynamic, Manual, unit.

   Initial State: audio variable = True

   Input: Keyboard Input N or A

Output: The audio variable will change based on the keyboard input.

How test will be performed: Unit and functional testing will be used to confirm that the audio variable is set correctly based on the user's keyboard input in the global_var.py file. Manual testing will be used for code inspection to ensure the code is running as planned.

2. test-id2

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: theme = 'default'

   Input: Keyboard Input "1", "2", or "3"

   Output: Based on the keyboard input the theme variable value should change."1" for the default dino theme, "2" for the student theme and "3" for the corona virus theme.

   How test will be performed: Unit and functional testing will be used to confirm that the theme variable is set correctly based on the user's keyboard input in the global_var.py file. Manual testing will be used for code inspection to ensure the code executes as desired.

3. test-id3

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: The settings page is displayed on the screen.

   Input: Keyboard Input "e"

   Output: The game will take you back to the main page and display the main page graphics.

   How test will be performed: Unit testing will be used to check if the variable main_flag is True and settings_flag is False. If it is true the main page is being displayed. Functional testing will be used to observe if the graphics have changed from the settings to main page. Manual testing will be used for code inspection to ensure the code executes as desired.

3.1.2 Gametrack Tests

1. test-id1

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: game images are set to default

   Input: theme variable

   Output: Appropriate themed images are displayed to the screen.

   How test will be performed: Unit testing will be used to confirm that the correct images are loaded. For example if theme variable = "corona" the images.RUNNING = images.RUNNING_THEME3 as opposed to images.RUNNING = images.RUNNING_THEME1. Functional testing will be verified by inspection to ensure that the correct images are displayed to the screen according to the theme. Manual testing will be used for code inspection to ensure the code executes as desired.

2. test-id2

   Type: Functional, Dynamic, Manual.

   Initial State: audio setting is set to true

   Input: audio variable

   Output: Audio is played if audio = True and not played if audio = False.

   How test will be performed: Functional testing will be verified by manual inspection to ensure that audio is playing out of the computer. Manual testing will be used to ensure that the correct audio track is loaded and played.

3. test-id3

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: Game track page

   Input: Keyboard input "p"

Output: Pauses the game track and displays the pause page.

How test will be performed: Functional testing will be verified by manual inspection to ensure that the game has stopped and that the appropriate text is displayed to the screen. Unit testing will be used to verify that game_track_flag = False and that game_track_pause_flag = True. This unit testing will ensure that the correct page elements are loaded to the screen. Manual testing will be used to ensure the code functions as desired.

4. test-id4

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: Pause game page

   Input: Keyboard input "u"

   Output: unpauses the game track and displays the game track page.

   How test will be performed: Functional testing will be verified by manual inspection to ensure that the game resumes as normal and that the appropriate graphics is displayed to the screen. Unit testing will be used to verify that game_track_flag = True and that game_track_pause_flag = False. This unit testing will ensure that the correct page elements are loaded to the screen. Manual testing will be used to ensure the code functions as desired.

5. test-id5

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: score = 0

   Input: n/a

   Output: score variable increases and is displayed to the screen

   How test will be performed: Functional testing will be verified by manual inspection to ensure that the score is displayed to the screen in the top right hand corner. Unit testing will be used to ensure that the score value is only increasing as game play occurs. Manual testing will be used to ensure the code functions as desired.

6. test-id6

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: character.run_img = True

   Input: keyboard input up arrow or spacebar

   Output: The character graphic will change from character.run_img to character.jump_img. The character will also increase it's position in the y-axis and therefore appear as if it is "jumping".

   How test will be performed: Functional testing will be verified by manual inspection to ensure that the character is jumping and has an increased then decreased y-axis value that mimics jumping in the real world. Unit testing will be used to ensure that the character.run_img = False and character.jump_img = True. Manual testing will be used to ensure the code functions as desired.

7. test-id7

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: character.run_img = True

   Input: keyboard input down arrow.

   Output: The character graphic will change from character.run_img to character.duck_img. The character will also decrease it's position in the y-axis and therefore appear as if it is "ducking".

   How test will be performed: Functional testing will be verified by manual inspection to ensure that the character is "ducking" and has an decreased y-axis value that mimics ducking in the real world. Unit testing will be used to ensure that the character.run_img = False and character.duck_img = True. Manual testing will be used to ensure the code functions as desired.

8. test-id8

   Type: Functional, Dynamic, Manual, Unit.

   Initial State: game_speed = 20

Input: n/a

Output: The game_speed increases as time progresses and the images move across the screen faster.

How test will be performed: Functional testing will be verified by manual inspection to ensure that the character is moving faster. Unit testing will be used to ensure that the game speed is increasing with time. Manual testing will be used to ensure the code functions as desired.

3.1.3 Leaderboard Tests

1. test-id1

   Type: Functional, Dynamic, Manual

   Initial State: User is at the Restart screen waiting for an area to be selected(mouse event)

   Input: User clicks on an area within the area of the "Leaderboard" text

   Output: The Leaderboard page is displayed and contains up to five users and their scores.

   How test will be performed: After the menu function is called and the restart_flag variable is set to true, the appropriate user input(mouse event) is entered and the output is checked to ensure that the leaderboard is displayed.

2. test-id2

   Type: Automated, Dynamic, Structural

   Initial State: None, this function is called on a unit basis. However the score.txt file must be populated

   Input: Leaders_text (Python list containing top users and their score)

   Output: Unit test passes if the users in the leader board have the highest recorded scores

   How test will be performed: The score.txt file will be manually populated with users and their scores. After calling the get_leaders() function, each index of the leader_txt output is validated using assert statements.

3. test-id2

   Type: Functional, Manual, Dynamic

   Initial State: display_leaderboard flag is set to true and the Leader board page is displayed

   Input: N/A

   Output: Text elements of the Leaderboard page are properly displayed

   How test will be performed: When the Leaderboard page is displayed, manually inspect the page to ensure that there is no text area overlap between the individual texts containing the user name and score.

4. test-id2

   Type: Functional, Automated, Dynamic

   Initial State: Restart flag is set to true and game point is higher than the current user's highest score.

   Input: Game points

   Output: Unit test passes if the user's high score is updated to the game points.

   How test will be performed: After the user has completed a game run, use an assert statement to ensure that the value of the current user's high score is updated to the value of game points if the game point is higher than the current user's previous high score

5. test-id2

   Type: Functional, Manual, Dynamic

   Initial State: User has completed a game run (Death count is 0)

   Input: Game points and username

   Output: Unit test passes if the user's username and score is appended to score.txt

   How test will be performed: After the user has completed a game run, manually inspect score.txt to ensure that the user's username and game point is entered into score.txt in the correct format.

3.1.4 Instructions Page Tests

1. test-id1

   Type: Functional, Dynamic, Unit and Manual test.

   Initial State: Current page displayed within interface is the main menu page and instructions = False.

   Input: User selects the area within the "How to play" text

   Output: The current page displayed on the interface will be the instructions page and instructions = True.

   How test will be performed: The program will be given a specific user input. The response will be compared to the expected output: user is taken to the settings page. For unit testing, the state of the instructions variable will be asserted.

2. test-id2

   Type: Functional, Dynamic, and Manual test.

   Initial State: Current page displayed within the interface is the instructions page.

   Input: The user enters the associated keyboard exit key("e").

   Output: Current page displayed within the interface will transition to the main menu page.

   How test will be performed: The user will enter the appropriate input. The response will be compared to the expected output: user is taken back to the main menu page.

3. test-id2

   Type: Functional, Dynamic, and Manual test.

   Initial State: The game is paused and instructions = False.

   Input: The user enters the associated keyboard("i") input.

   Output: Current page displayed within the interface will transition to the instructions page and instructions = True.

How test will be performed: The user will enter the appropriate input. The response will be compared to the expected output: user is taken back to the instructions page. For unit testing, the state of the instructions variable will be asserted.

4. test-id2

   Type: Functional, Dynamic, Unit and Manual test.

   Initial State: The current clock time is between 7:00 p.m. and 7:00 a.m and the instructions page is currently displayed on the interface.

   Input: N/A

   Output: Current page displayed within the interface will be the instructions page in dark mode.

   How test will be performed: The displayed instructions page will be visually inspected to check if it is in dark mode. Similarly, the state of the background colour and font colours will be asserted to their expected values in unit testing.

5. test-id2

   Type: Automated, manual, dynamic, and functional test.

   Initial State: Current page displayed within the interface is the instructions page.

   Input: N/A

   Output: Text elements of the instructions page are displayed.

   How test will be performed: A manual test will be performed to inspect if the text elements are in their correct positions. Unit testing will assert the element positions with their expected positions.

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

Proof of Concept verifies and validates the method by which automated testing is performed. The testing for proof of concept will test the functionality of the game and game flow logic.

4.1 Game Play

1. test-id1

   Initial State: Game track page and score = 0
   Input: n/a

Output: Game track is shown to the screen and updated score is in the top right corner.

How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Manual inspection will check if the score variable is a positive value. Visual inspection will validate if the screen is displaying the game play page and that the score is updated in the top right hand corner.

2. test-id2

   Type: Functional, Dynamic, Manual.

   Initial State: Game Track Page

   Input: theme variable

   Output: theme graphics to the game page

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the correct images in accordance to the theme variable.

3. test-id3

   Type: Functional, Dynamic, Manual.

   Initial State: Game track page

   Input: Keyboard input "p"

   Output: Page that displays options to unpause the game or go to the instructions.

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the correct page and that the game track is paused.

4. test-id4

   Type: Functional, Dynamic, Manual.

Initial State: Game track page

Input: Keyboard input down arrow

Output: Character images changes to img_duck

How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the correct "ducking" image to the screen.

5. test-id5

   Type: Functional, Dynamic, Manual.

   Initial State: Game track page

   Input: Keyboard input up arrow

   Output: Character images changes to img_jump

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the correct jumping image to the screen and if the character jumps on screen.

Pause Page

1. test-id1

   Initial State: Pause page

   Input: Keyboard input "U"

   Output: Game track continues on screen.

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the game play page. To verify the game is running again, the character will be inspected and the score variable.

2. test-id2

   Initial State: Pause page

   Input: Keyboard input "I"

   Output: Instructions page displayed on screen

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the instruction page.

4.2 Additional Game Pages

Home Page

1. test-id1

   Type: Functional, Dynamic, Manual.

   Initial State: Main Page and default settings (theme and audio)

   Input: Any keyboard input

   Output: Game track is shown to the screen.

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the game play page.

2. test-id2

   Type: Functional, Dynamic, Manual.

   Initial State: Main page and default settings (theme and audio)

   Input: Mouse click on "Game Settings" text

   Output: Settings page.

   How test will be performed:Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the settings.

3. test-id3

   Type: Functional, Dynamic, Manual.

   Initial State: Main page and default settings (theme and audio)

   Input: Mouse click on "How to Play" text

   Output: Instructions page

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the screen is displaying the instructions page.

Instructions Page

1. test-id1

   Type: Functional, Dynamic, Manual.

   Initial State: Instructions Page

   Input: Keyboard input "1", "2" or "3"

   Output: Graphics on the main page and game track change.

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Visual inspection will validate if the graphics have changed according to the theme. On the main page the icon will change and on the game track the obstacles and character will change based on the theme.

2. test-id2

   Type: Functional, Dynamic, Manual.

   Initial State: Instructions Page

   Input: Keyboard input "N" or "A"

   Output: if "N" audio = True, if "A" audio = False

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Manual inspection testing will be used to check if the variable has changed in the appropriate python file.

3. test-id3

   Type: Functional, Dynamic, Manual.

   Initial State: Instructions Page

   Input: Keyboard input "E"

   Output: Main Page

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Manual inspection testing will be used to check if the home page is displayed.

Instructions Page

1. test-id1

   Type: Functional, Dynamic, Manual.

   Initial State: Settings

   Input: Keyboard input "E"

   Output: Main page

   How test will be performed: Manual and dynamic testing will be used to ensure the program functions as expected. Manual inspection testing will be used to check if the home page is displayed.

5 Comparison to Existing Implementation

6 Unit Testing Plan

The unittest unit testing framework will be used to conduct unit tests.

6.1 Unit testing of internal functions

To validate internal functions, test cases will be created for each class, as well as specific functions that return a value. Test cases will cover edge cases to validate the extreme boundaries of the various input. Additionally, exceptions and basic tests will be inputs to testing internal functions. The tests

will examine the correctness and robustness of the program. Furthermore, no stubs or drivers are needed in the execution of test cases. The classes should import the necessary libraries and modules needed for testing. We will ensure that majority i.e. at least 80% of the functions will be covered by the test cases through coverage metrics.

6.2 Unit testing of output files

The only output file generated by the game is the score.txt file which stores the high scores displayed on the leader board. To validate the file output, a combination of unit testing and manual testing will be done. A dynamic manual test is conducted by having the user play the game and beat the high score. Their current score should be displayed in the leaderboard and found on top of the score.txt file. On the other hand, unit testing will verify that the high score displayed on the interface is the same as the high score in the score.txt file.

Asides from the output file, the game has a graphical user interface generated through PyGame. A user conducted black box testing will be done to test the interface and its functionalities. The user will play the game, testing each feature. If the game is successfully played by the user and all functionalities stated within the SRS have been achieved, then the user interface passes the test.

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

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

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.