SE 3XA3: Test Plan
Chrome-Dino-Runner

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

March 12, 2022

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
March 11, 2022	0	All team members

1 General Information

1.1 Purpose

This document describes the testing, validation and verification process that will be implemented by Team Rex to test the SFWRENG 3XA3 project, Chrome Dino Runner (CDR). These test cases were determined before the complete implementation of the project and will be used as a guide to test and maintain the code. Since this document is being created with test cases before complete implementation, there will be revisions to our test cases. These revisions will be shown in revision 1 of this document.

1.2 Scope

Our project, Chrome Dino Runner, is a Python implementation of the original Chrome Dinosaur Game with additional features. The scope of testing will cover: graphics, leaderboard and settings logic, and game functionality. This document will outline the system and unit tests for functional and non-functional requirements.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
CDR	Chrome Dino Runner
FST	Functional System Test
NFST	Non-Functional System Test

Table 3: **Table of Definitions**

Term	**Definition**
game run (run)	Refers to a single iteration of the game play(Game start to game over). Every time the player exits the game or restarts the game, another game run begins.
instructions	A Boolean variable indicating the display status of the instructions page.
main	A Boolean variable indicating the display status of the main menu page.
points	An integer variable indicating the user score during the game.
score.txt	A text file which contains each player's username and score. After every game run, the current players username and score is appended to a new line in this text file.
test group	A group of 5 individuals ranging in technical ability and are proficient in English who participate in user testing. None of the members of this group are on the test team.
points	An integer variable indicating the user score during the game.
global$_v ar.py$	A python module which contains all global variables.

2 Plan

2.1 Software Description

This software project implements the Chrome Dino Game in Python using the pygame library. The game has additional features such as various themes, a leaderboard and additional pages to help with game flow. Due to the nature of the software we will need to perform structural, functional, dynamic, static, manual and automated testing to ensure that the game functions as desired.

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 for creating and running test case. Each test case will begin with the word "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

To conduct unit tests, the **Python unittest** unit testing framework will be used.

2.5 Testing Schedule

Table 4: **Table of Testing Schedule**

Task	Team member	Due Date
Setting Page Tests	Anjola	March 28th 2022
Gametrack Tests	Chelsea	March 28th 2022
Leaderboard Tests	Sheridan	March 28th 2022
Instructions Page	Anjola	March 28th 2022
Operating Systems and Compatibility Test	All	March 30th 2022
Look and Feel	Chelsea	March 30th 2022
Usability	Sheridan	March 30th 2022
Precision Tests	Anjola	March 30th 2022
Security Requirements	Sheridan	March 30th 2022
Cultural and Political Requirements	Chelsea	March 30th 2022

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Settings Page Tests

1. FS-SPT-1

   Type: Functional, Dynamic, Manual.

   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 tests will be performed to assert that the vale of the audio variable changes in accordance with the keyboard input. The user will listen to the audio to judge if the game audio changes in response to the input.

2. FS-SPT-2

   Type: Functional, Dynamic, Manual.

   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 tests will be performed to assert that the vale of the theme variable changes in accordance with the keyboard input. Manual visual tests will also be performed to verify that the game theme display also changes to the corresponding theme input.

3. FS-SPT-3

   Type: Functional, Dynamic, Manual.

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

Input: Keyboard Input “e”

Output: The current setting page will transition 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. Visual inspection will be used to observe if the graphics have changed from the settings to main page.

3.1.2 Leaderboard Tests

1. FST-LT-1

   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 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. FST-LT-2

   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 (leaders_text) 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 leaders_txt output is validated using assert statements.

3. FST-LT-3

   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, the test team will manually inspect the page to ensure that there is no text area overlap between the individual texts containing the user name and score.

4. FST-LT-4

   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: If game points is higher than the current user high score, 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, an assert statement will be used. This will ensure that the value of the current user's high score is updated to the value of game points appropriately.

5. FST-LT-5

   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, the test team will manually inspect score.txt to ensure that the user's username and game point is entered into score.txt in the correct format.

3.1.3 Gametrack Tests

1. FST-GT-1

   Type: Functional, Dynamic, Manual.

   Initial State: Game images are set to default

   Input: Theme variable (e.g Corona, Student)

   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. FST-GT-2

   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. FST-GT-3

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. FST-GT-4

Type: Functional, Dynamic, Manual.

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. FST-GT-5

Type: Functional, Dynamic, Manual.

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. FST-GT-6

   Type: Functional, Dynamic, Manual.

   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. FST-GT-7

   Type: Functional, Dynamic, Manual.

   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. FST-GT-8

Type: Functional, Dynamic, Manual.

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.

3.1.4 Instructions Page Tests

1. FST-IPT-1

   Type: Functional, Dynamic, Manual.

   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 instructions page. For unit testing, the state of the instructions variable will be asserted.

2. FST-IPT-2

   Type: Functional, Dynamic, and Manual test.

   Initial State: The 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. FST-IPT-3

   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. FST-IPT-4

   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. FST-IPT-5

   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.

3.1.5 Operating System Compatibility Test

1. FST-IPT-3

Type: Structural, Manual, Dynamic

Initial State: Game is stored in an executable file and uploaded to Google drive

Input: Download and launch the game on computers running on Windows 10, macOS Sierra 10.12 and Linux Ubuntu 16.04.

Output: Game should run successfully on any of these operating systems.

How test will be performed: The executable file containing our program will be uploaded to Google drive. The test team will then download and execute the programs on systems running on Windows 10, macOS Sierra 10.12 and Linux Ubuntu 16.04. The test team will verify the successful run of the game on these systems

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel

Style Requirements

1. NFS-1

Type: Functional, Manual, Dynamic

Initial State: The game executable file is loaded and is currently on the main page.

Input/Condition: The test group are asked to inspect the interface and its design.

Output/Result: Atleast 80% of the test group will report that the game colours are not distracting and the 90's arcade appearance is maintained.

How test will be performed: The test group will play the game. After playing, they will then be asked to comment on the user interface design (i.e. colour and theme).

3.2.2 Usability

Ease of Use

1. NFS-2

   Type: Functional, Manual, Dynamic

   Initial State: Game is newly launched from an executable file and set up on a computer (Main Menu Page)

   Input/Condition: Users are asked to play the game

   Output/Result: Majority of the users are able to successfully play the game without any assistance from the development team

   How test will be performed: The test group will be asked to play the game, which will already be launched on a computer. Users should be able to successfully navigate the different pages and play the game without additional instructions from the development team.

2. NFS-3

   Type: Structural, Manual, Dynamic

   Initial State: Game is stored in an executable file and uploaded to Google drive

   Input/Condition: The test group is asked to download and launch the game on their computers running Windows 10, macOS Sierra 10.12 or Linux Ubuntu 16.04.

   Output/Result: Users from the test group are successfully able to launch the game on their computers without assistance from the development team.

How test will be performed: The executable file containing our program will be uploaded to Google drive. We will then ask the test group to download and execute the programs on their computers running on Windows 10, macOS Sierra 10.12 or Linux Ubuntu 16.04. The test team will verify the successful run of the game on these systems

3.2.3 Precision Test

1. NFS-4

   Type: Functional, Dynamic, Manual

   Initial State: The game is launched and user-input (keyboard or mouse) is expected

   Input: Using a stop watch, time how long it takes for the game to response (change page) after an input is entered.

   Output: The average response time should be less than two seconds

   How test will be performed: The game is launched to the homepage. Someone from the test group enters a valid input and the game's response time is measured by another member of the test group. A similar test in performed for the response time of pause and unpause inputs on the game play page.

3.2.4 Security Requirements

Access Requirements

1. NFS-5

   Type: Functional, Dynamic, Manual

   Initial State: The interface is currently displaying the leaderboard page

   Input: User mouse and keyboard input

   Output: The user is not able to change the leaderboard names and high score.

   How test will be performed: The user will attempt to change the leaderboard manually, while in the leaderboard page.

3.2.5 Cultural and Political Requirements

Cultural Requirements

1. NFS-6

   Type: Functional, Dynamic, Automated

   Initial State: The game prompts the user for a username

   Input: Illegal username (e.g a swear word)

   Output: An error message should be displayed to inform the user of an illegal username

   How test will be performed: The game developers will implement an Illegal username dictionary. When a username is entered, it will be compared to the Illegal username dictionary. If it is an illegal word an Illegal-Argument error is thrown. An assert statement will be used to validate this error handling method.

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

Game Track

1. POC-1

   Initial State: Game track page is display to the screen and game points = 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 points 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. POC-2

   Type: Functional, Dynamic, Manual.

   Initial State: Game Track Page

   Input: Theme variable

   Output: Theme of the game page changes in accordance with the input theme variable.

   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. POC-3

   Type: Functional, Dynamic, Manual.

   Initial State: Game track page

   Input: Keyboard input ("p")

   Output: Game is paused and a "Pause page" which contains un-pause instructions and game instructions is displayed.

   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. POC-4

   Type: Functional, Dynamic, Manual.

   Initial State: Game track page

   Input: Keyboard input(down arrow)

Output: Character images changes to img_duck (Image of the dinosaur(game character) ducking)

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. POC-5

   Type: Functional, Dynamic, Manual.

   Initial State: Game track page

   Input: Keyboard input up arrow

   Output: Character images changes to img_jump (Image of the dinosaur(game character) jumping)

   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. POC-6

   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 and the game points will be inspected.

2. POC-7

   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

Main Page

1. POC-8

   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. POC-9

   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. POC-10

   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.

Home Page

1. POC-11

   Type: Functional, Dynamic, Manual.

   Initial State: Settings 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. POC-12

   Type: Functional, Dynamic, Manual.

   Initial State: Settings 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. POC-13

   Type: Functional, Dynamic, Manual.

   Initial State: Settings 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. POC-14

   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

The implementation performed by Team Rex can be compared to the open source implementation using the Usability tests. The usability tests will allow the user to execute the game and play. Completing the functional requirement tests for the main page will allow users to navigate through the additional pages implemented by Team Rex.

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 90% 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 username and scores of the game players. To validate the file output, please refer to test FST-LT-5.

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.