Test plan

Introduction

This section describes the nature of the system under test (SUT), the goals and scope of the testing, and the types of the testing that will be performed. The testing approach and the features to be tested are also included in this section.

Nature of the Project

The Water Cycle VR guide is being developed for students, teachers and for people who are interested in learning the earth’s water cycle. This virtual reality application will allow the user to have a visual view of the complete water cycle and its stages. The application is divided into two main features; one is the tour guide which shows the user all the stages completely and with details. Second is the game which will enhance user’s knowledge in the water cycle, making the learning process entertaining as well as useful. The system is expected to have features which are currently not present in much applications.

The user will be interacting with the application through the gaze control and Bluetooth controller. The gaze control will allow the user to perform many functionalities such as looking around in 360o and selecting options, whereas Bluetooth controller is mainly used to control the player in the app.

Testing objectives and scope

The main objectives to test the Water Cycle VR guide are to ensure that:

* All the required functionalities of the Water Cycle VR guide are well established and working correctly according to the requirements.
* The software meets its non-functional requirements.
* To identify and reveal as many errors as possible in the tested software.
* To perform the required tests efficiently and effectively, within budgetary and scheduling limitations.

Types of System Tests

The set of system tests that will be performed in this test plan includes:

* Functional testing: all the main functions of the application will be testing. Unit testing technique will be used to test each module individually to ensure that they perform as designed.
* Performance testing: to test that the software meets its non-functional requirements.

Features to be tested

The following features will be tested to ensure that they work as expected:

* Main menu: testing that the main menu performs its operations correctly such as proceeding to game or tour.
* Tour guide: testing that it performs and designed.
* Game: testing that it performs as designed.

Approach

In order to ensure the delivery of a high-quality application, several tests will be performed on Water Cycle VR guide prior to its releasing to the public to make certain that the app runs as smoothly as possible and to provide a good user experience. System tests will be performed extensively. The app will undergo functional testing to test requirements and features of the app in order to ascertain that they meet user requirements. All requirements must be covered by at least one test case.

Staff

The team will be responsible the testing phase of the project. They will design test cases and execute the tests. The students will be supervised by project supervisor.

Record Keeping

All tests that are performed in the testing phase will be recorded. The test failures will be kept separately to be fixed and tested again.

Stop-Test Criteria

The system testing will be concluded after covering every user requirement by at least one test case and making sure that there are no incidents or failures in the application. We acknowledge the fact that the system will still not be 100% defect free.

Testing techniques:

The functional requirements will be evaluated using the following black box testing techniques:

Equivalence class partitioning (ECP)

State transition testing

ECP will be mainly used in unit testing to insure that all classes of legal inputs are accepted, for example, gazing at tour guide for 2 seconds will take the user to the tour, and all classes of illegal input are rejected by the system. This types of black box testing eliminates the need for exhaustive testing, which is not feasible. Furthermore, it allows the tester to cover a large domain of input or output with a smaller subset that is selected from an equivalence class. State transition testing will also be used to cover all paths, to enter all states and to examine every possible state transition, for example, the state transition in our app from main menu to the game. This will help us to make certain that the flow between the states is on the mark. Many transitions that look ambiguous or haven’t been tried before will be tested in order to ensure that the system works as designed.

Test cases

Test cases designed for the functional testing are divided into two types; unit testing and module testing. Each type of testing has appropriate amount of test cases to ensure that the functionality and the features behave according to the requirements.

Unit testing:

In this test, we will be testing individual units of the system separately to ensure that each unit performs as designed. This system has been divided into three modules, so in unit testing each module will be tested individually.

Main menu (Module 1) testing:

In this test, we will perform scenario based test case design for Start Tour Guide Use-Case:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Results | Actual Result | Test  Result |
| TC\_001 | Starting the tour guide for the water cycle | Testing the start tour guide functionality by looking at the button for 2 seconds | Launch the app. | Look at the Tour Guide button for 2 seconds. | Gaze at the start tour button for 2 sec. | Tour guide Started | Tour Guide started | Pass |
| TC\_002 | Starting the tour guide for the water cycle | Testing the start tour guide functionality by not looking at the tour guide button | Launch the app | Look anywhere but not on the Tour Guide button | Gaze anywhere expect the start tour button. | Tour Guide does not start. i.e. no action taken | Tour guide does not start | Pass |
| TC\_003 | Starting the tour guide for the water cycle | Testing the Start tour guide functionality by looking at start Game button | Launch the app. | Looks at the start game button for 2 seconds. | Gaze at the start game button for 2 seconds. | Game starts | Game starts and tour guide does not start | Pass |

In this test, we will perform scenario based test case design for Start Game Use-Case:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Result | Actual Result | Test  Result |
| TC\_004 | Starting the Game. | Testing the Start Game functionality by looking at the start Game button for 2 seconds. | Launch the app. | Look at the Start Game button for 2 seconds. | Gaze at the Start Game button for 2 sec. | Game is started | Game is started | Pass |
| TC\_005 | Starting the Game. | Testing the Start Game functionality by not looking at the Start Game button | Launch the app | Look anywhere but not on the Start Game button | Gaze anywhere expect the Start Game button. | Game does not start. i.e. no action taken | Game does not start | Pass |
| TC\_006 | Starting the Game. | Testing Start Game functionality by looking at Tour Guide button. | Launch the app. | Looks at the Start Tour Guide button for 2 seconds. | Gaze at the Start Tour button for 2 sec. | Tour Guide starts | Game does not start but instead tour guide starts | Pass |

In this test, we will perform scenario based test case design for Exit Use-Case:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Results | Actual Result | Test  Result |
| TC\_007 | Exiting the app. | Testing the Exit functionality by looking at the Exit button for 2 seconds. | Launch the app. | Look at the Exit button for 2 seconds. | Gaze at the Exit button for 2 sec. | App terminated | App terminated. | Pass |
| TC\_008 | Exiting the app. | Testing the Exit functionality by not looking at the Exit button | Launch the app. | Look anywhere but not on the Exit button | Gaze anywhere expect the Exit button. | App does not terminate. i.e. no action taken | App does not terminate | Pass |

**Tour Guide (Module 2) testing**:

In this test, we will perform scenario based test case design for Repeat Tour Use-Case:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Results | Actual Result | Test  Result |
| TC\_009 | Restarting the tour | Testing the Restarting the tour functionality by pressing button C. | The user is in the tour. | The user presses Button C. | Button C | The tour is repeated. | The tour is repeated. | Pass |
| TC\_0010 | Restarting the tour | Testing the Restarting the tour functionality by pressing button A. | The user is in the tour. | The user presses button A. | Button A | The main menu appears. | The main menu appears. | Pass |
| TC\_0011 | Restarting the tour | Testing the Restarting the tour functionality by pressing button B. | The user is in the tour. | The user presses Button B. | Button B | No action taken. | No action taken. | Pass |
| TC\_0012 | Restarting the tour | Testing Restarting the tour functionality by pressing button D. | The user is in the tour. | The user presses Button D. | Button D | No action taken. | No action taken. | Pass |

In this test, we will perform scenario based test case design for Exit to Main Menu Use-Case:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Results | Actual Result | Test  Result |
| TC\_0013 | Exiting to main menu from tour. | Testing the exiting to main menu functionality by pressing button A. | The user is in the tour. | The user presses button A. | Button A | The main menu appears. | The main menu appears. | Pass |
| TC\_0014 | Exiting to main menu from tour. | Testing the exiting to main menu functionality by pressing button C. | The user is in the tour. | The user presses Button C. | Button C | The tour is repeated. | The tour is repeated. | Pass |
| TC\_0015 | Exiting to main menu from tour. | Testing the exiting to main menu functionality by pressing button B. | The user is in the tour. | The user presses Button B. | Button B | No action taken. | No action taken. | Pass |
| TC\_0016 | Exiting to main menu from tour. | Testing the exiting to main menu functionality by pressing button D. | The user is in the tour. | The user presses Button D. | Button D | No action taken. | No action taken. | Pass |

**Game Mode (Module 3) testing**:

In this test, we will perform scenario based test case design for Move Player Use-Case:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Expected Results | Actual Result | Test  Result |
| TC\_0017 | Moving the player. | Testing the functionality of moving the player by moving 360˚ rocker to right | The user must be in game. | Move the 360 ˚ Rocker to right | Player moved to right. | Player moved to right. | Pass |
| TC\_0018 | Moving the player. | Testing the functionality of moving the player by moving 360˚ rocker to right | The user must be in game. | Move the 360 ˚ Rocker to left | Player moved to left. | Player moved to left. | Pass |
| TC\_0019 | Moving the player. | Testing the functionality of moving the player by moving 360˚ rocker to right | The user must be in the game. | Move the 360 ˚ Rocker upwards. | Player moved to forward. | Player moved forward. | Pass |
| TC\_0020 | Moving the player. | Testing the functionality of moving the player by moving 360˚ rocker to right | The user must be in the game. | Move the 360 ˚ Rocker downward. | Player moved to backward. | Player moved to forward. | Fail |

In this test, we will perform scenario based test case design for Play Again Use-Case:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Results | Actual Result | Test  Result |
| TC\_0021 | Playing the game again. | Testing the play again functionality by pressing button C. | The user must be in game. | Pressing the Button C. | Button C. | Game is loaded again. | Game is loaded again. | Pass |
| TC\_0022 | Playing the game again. | Testing the play again functionality by pressing button A. | The user must be in game. | Pressing the Button A. | Button A. | User returned to main menu. | User returned to main menu. | Pass |

**System Testing**

After performing unit testing, in which all the modules were tested one by one, we will perform system testing. System testing is the testing conducted on the system that is integrated and completed. This type of testing allows us to determine whether the system’s modules interact with each other properly or not and whether the inputs given to the system result in the desired output.

The first test case in the following table is to test the flow of the user entering the tour and exiting it.

The second test case in the following table is to test the flow of the user entering the game, playing the game and exiting it.

The third use case in the following table is to test the flow of the user entering the tour, exiting the tour, entering the game and exiting the game and then exiting the app.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case Description | Precondition | Test Steps | Test Data | Expected Results | Actual Result | Test  Result |
| STC\_001 | Testing the flow of the user entering the tour and exiting it | The user will enter the tour by gazing at the start tour button for 2 seconds, then the user will the tour with button A. | App is launched. | User gazes at the start tour button for 2 seconds. | Gaze at the start tour button for 2 seconds. | User enters the tour. | User enters tour. | Pass |
| User presses Button A to exit the tour | Button A | User exits the tour. | User exits the tour. |
| STC\_002 | Testing the flow of the user entering the game, playing the game and exiting it. | The user will enter the game by gazing at the Start Game for 2 seconds, plays the game and then the user exits the Game with button A. | App is launched. | User gazes at the Start Game button for 2 seconds. | Gaze at the start game button for 2 seconds. | User enters the game. | User enters the game. | Pass |
| User moves the player right. | Move the 360 ˚ Rocker to right. | Player moves right. | Player moves right. |
| User moves the player left. | Move the 360 ˚ Rocker to left. | Player moves left. | Player moves left. |
| User exits the Game. | Button A | Player exits the game. | Player exits the game. |
| STC\_003 | Testing the flow of the user entering the tour, exiting the tour, entering the game and exiting the game and then exiting the app. | The user will enter the tour by gazing at the start tour button for 2 seconds, then the user will exit the tour by pressing button A, then the user will enter the game by gazing at the start game button, then the user will exit the game by pressing button A, then the user will exit the app by gazing at the exit button for 2 seconds | App is launched | User gazes at the start tour button for 2 seconds. | Gaze at the start tour button for 2 seconds | User enters the tour. | User enters the tour. | Pass |
| User presses button A to exit the tour. | Button A | User exits the tour | User exits the tour |
| The user gazes at start game button for 2 seconds. | Gaze at the start game button for 2 seconds. | User enters the game. | User enters the game. |
| The user exits the game by pressing button A. | Button A | User exits the game. | User exits the game. |
| The user Exits the app by gazing exit button for 2 seconds. | Gaze at the exit button for 2 seconds. | App is terminated | App is terminated |