Durham University MeditateVR
Test Plan Report

Group 1

Adam Seidel (nnng27)

Ben Hughes (krcc82)

Enego Comley (wdsz78)

Pad Pang (gjzt58)

Nathaniel Mosey (wjld56)

Tom Owen (zjmq47)

Contents

Section 1 – Introduction

In this document we will outline a testing plan for our group's project. These tests will cover both functional and non-functional requirements. We will outline multiple tests for each testing type and in the second section of this document we will further develop a selection of these tests. The final section will include testing context and the severity of any failed tests.

1.1 – Project Overview

We are working on developing a VR meditation environment that will enhance the adaptivity and immersion of users during meditation sessions. The project is being developed based on the requirements of our client Prof Alexandra Cristea. This project is based around the belief that modern VR technology can increase the effectiveness of meditation sessions. The project's objective is to create an immersive and abstract meditation environment for our client to use for research that utilizes visualization-based meditation techniques. This will include a relaxing

environment with a plant able to grow according to the quality of the users' mediation. To assist this, the environment will feature relaxing visuals as well as audio and visual cues to help with the meditation session.

1.2 - Testing Overview

Our project implements Test Driven Development (TDD) approach to software development, where test cases are written before the code components. This is to help mitigate risks we have identified in our Requirements Specification and ensure our project can meet our Functional and Non-Functional Requirements. By adopting continuous testing alongside development, we can reduce the emergence of bugs and make our program more robust.

Tests will be split into 4 types for this document's purposes: Unit (UNT), Integration (INT), System (SYS) and User Acceptance (UAT) testing. These tests will also be categorized based on their relation to functional or non-functional requirements (or both). An overview of each type of testing and what tests we have selected are outlined below.

1.2.1 Unit Testing Overview

Our Unit Testing plan involves testing the individual components or units of the system in isolation. Our units are functions, methods, or classes that perform specific tasks within the software. Unit tests are designed to verify that each unit of code behaves as expected, according to its design and specifications. They have been categorized into what type of test they are: Black box (where the function is assessed solely on the output) or White Box (where the internals of the system and observed to determine correctness).

Test ID	**Brief Description**	**Test Type**
unt_test-01	Biometrics output	Black Box
unt_test-02	User Telemetry to File	Black Box
unt_test-03	Config File loading	White Box
unt_test-04	Game music and sound	Black Box
unt_test-05	Startup script	Black Box

1.2.2 Integration Testing Overview

Integration testing is a key to ensuring all the individual components of the system work in tandem. It's where we combine different software modules and test them together as a group to ensure interactions are as expected and demonstrate overall system functionality. For our project we will use an incremental approach to combine parts of the system piece by piece.

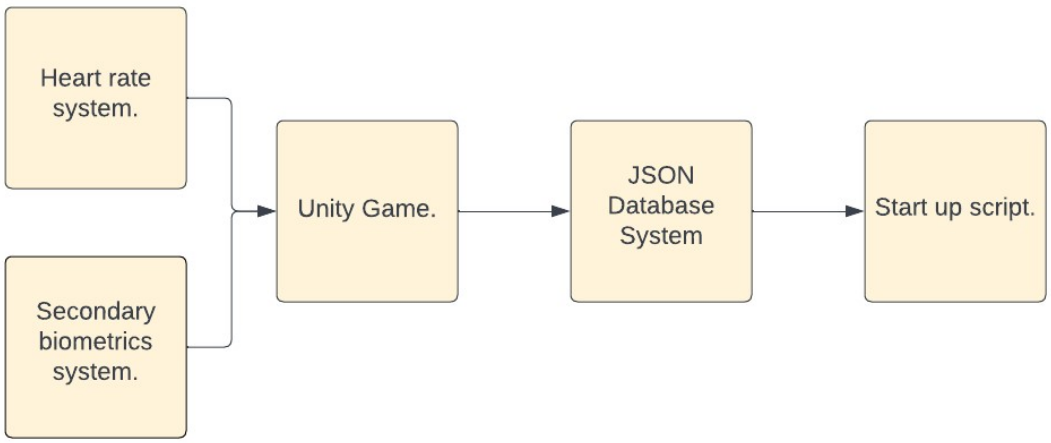

Figure 1 Integration Testing Diagram

Biometrics and Unity:

First, we'll connect the biometrics systems with the Unity system. We want them to communicate smoothly, ensuring the Unity project receives all the data correctly. Doing this incrementally helps us find and fix any problems as they come up and lets us ensure they are efficiently implemented.

Integrating the JSON Database:

Once the biometrics and Unity systems are working together, we'll integrate the database system. This means making sure the application can write all the research data to the JSON database. Testing here is about making sure data moves correctly between the application and the JSON database, and that everything in the database works as it should.

Integrating the Final system:

Finally, we'll make sure the final system runs correctly and all the files needed for research are accessible for the program and the researcher. This means making sure the application can find and use its configuration files, and that the database files are accessible. We'll also ensure the researcher can access the files they need for research.

1.2.3 System Testing Overview

Our System Testing involves testing the integrated system as a whole to ensure that all parts of the system function correctly together and meet the requirements specified in the system requirements specification. System testing verifies the behavior of the entire system against functional and non-functional requirements. Passing these tests is integral to verifying that our proposed system meets the minimum viable product agreed with the client, according to their vision.

Test ID	Brief Description
sys_test-01	Single Script
sys_test-02	High Framerate
sys_test-03	Non-identifiable Data
sys_test-04	Researcher Config
sys_test-05	Plant Growth reflects meditation

1.2.4 User Acceptance Testing Overview

User Acceptance Testing (UAT) is our final phase of testing performed by end-users and stakeholders (researchers and potential research subjects) to validate whether the system meets their requirements and is ready for deployment. Our

UAT focuses on assessing the system's usability, functionality, and overall user experience. This will be evaluated primarily through the client's evaluation, since they are the main end-user, and augmented with evaluation from research subjects as stakeholders in the system.

Test ID	**Brief Description**
usr_test-01	Calming Audio
usr_test-02	Cohesive Visuals
usr_test-03	Easy to use config
usr_test-04	Suitable Audio
usr_test-05	Growth feels reflective

Section 2 – Test Cases

In this section below are detailed the planned test cases in each of the 4 testing categories.

2.1 – Unit Testing (UNT)

Test Case ID	unt_test-01
Description of test	The biometrics system outputs a list of floats from 0 to 100 that show the user's stress level over time.
Related requirement spec/design spec details	FR 1.1, FR 3.4
Pre-requisites for test	None
Test procedure	1. Start all programs, those used for biometrics and the game. 2. Wear the biometric device. 3. Ensure the output data reflects the user's periods of high and low stress.
Test material used	C# code in Unity and BCI-OSC API
Expected result (test oracle)	The variable userStress should be a list of floating-point numbers ranging from 0 to 100. Each float value represents the stress experienced by the user, with its index within the list indicating the timestamp for that experience. When the user was thinking about something stressful, these should be lower.
Level of Severity	Extreme
Created by	Enego Comley
Test environment(s)	Windows 10

Test Case ID	unt_test-02
Description of test	User telemetry is stored in a log file.
Related requirement spec/design spec details	FR 3.5
Pre-requisites for test	The VR game and biometrics system generates user telemetry data and stores it in variables.
Test procedure	1. Start all software. 2. Complete a meditation session. 3. Open and inspect the log file, which stores telemetry data.
Test material used	BCI-OSC API, VR headset, PC
Expected result (test oracle)	There should be a new file in the logs folder called MeditationDATETIME where DATETIME will be replaced with the time and date of the meditation. The contents of this file should be in the JSON format. It should contain a dictionary whose keys are forms of telemetry, such as heart rate or the direction they are facing. Each of these will then be paired with an array of each telemetry item over time.
Level of Severity	Severe
Created by	Enego Comley
Test environment(s)	Windows 10

Test Case ID	unt_test-03
Description of test	The system must load a text file containing meditation configuration information. This data must then be parsed into an instance of the config class in the Unity game.
Related requirement spec/design spec details	FR 3.2
Pre-requisites for test	None
Test procedure	1. Change all parameters in the configuration file. 2. Start the unity game. 3. Inspect the public variables in the newly created config class to see that the changes in 1 were recorded.
Test material used	PC
Expected result (test oracle)	The values of the variables in the config class should match those specified in the .txt file.
Level of severity	Moderate
Created by	Enego Comley
Test environment(s)	Windows 10

Test Case ID	unt_test-04
Description of test	During the meditation game, music and atmospheric sound should be playing.
Related requirement spec/design spec details	FR 2.4
Pre-requisites for test	The config system must be in place so that the game knows to play music and atmospheric sound.
Test procedure	1. Make sure the pc system's sound is set to come out of the speakers. 2. Ensure that music and atmospheric sound are enabled in the config. 3. Start the unity game. 4. Listen for music and atmospheric sound.
Test material used	The Unity game.
Expected result (test oracle)	There should be both music and atmospheric sound such as birds chirping.
Level of Severity	Moderate
Created by	Enego Comley
Test environment(s)	Windows 10

Test Case ID	unt_test-05
Description of test	The startup script should start all software.
Related requirement spec/design spec details	FR 1.2
Pre-requisites for test	All the individual programs should be able to start up and run without crashing.
Test procedure	1. Run the startup script. 2. Check that all programs are running.
Test material used	Unity game, BCI-OSC API
Expected result (test oracle)	Both the programs for the unity game and the biometrics system should have started.
Level of Severity	Moderate
Created by	Enego Comley
Test environment(s)	Windows 10

2.2 – Integration Testing (INT)

After performing each Unit test we will then move onto Integration testing to ensure that all the atomized parts of the system work together and they all communicate correctly.

Test Case ID	int_test-01
Description of test	The Biometrics tracking system successfully integrates with the plant growth system
Related requirement spec/design spec details	
Pre-requisites for test	The Biometrics system outputs data which can be routed into the main unity program and the plant growth system is able to take parameters to adjust its growth rate
Test procedure	1. Run biometrics tracking system, with a live feed of the data it produces 2. Run the Unity game system and load into the game. 3. Have the user do things that would cause the flower to be small i.e. relax mentally or lower heart rate and see if the plant stays small 4. Have the user do stressful or physically intensive things to see if the plant growth increases
Test material used	VR Headset. BCI-OSC API. PC
Expected result (test oracle)	The plant growth rate is reflective of the data coming from the biometrics system.
Level of Severity	Severe
Created by	Ben Hughes
Test environment(s)	Windows 10, Unity, Oculus VR

Test Case ID	int_test-02
Description of test	The Researcher config data goes throughout the system to the different parts that it affects
Related requirement spec/design spec details	
Pre-requisites for test	The File loading and variable parsing system works and all configurable aspects take variable parameters.
Test procedure	1. Configure variables to have certain values and then run the program. 2. Open the Unity Variables view 3. Ensure the configurable parts of the system have changed in line with the configuration. 4. Change the variables to other values and repeat step 2
Test material used	PC
Expected result (test oracle)	The parts of the system that can be configured are shown
Level of Severity	Severe
Created by	Ben Hughes
Test environment(s)	Windows 10, Unity, Oculus VR

2.3 – System Testing (SYS)

Test Case ID	sys_test-01
Description of test	The system is started by a single script
Related requirement spec/design spec details	FR 1.2
Pre-requisites for test	System has not been started.
Test procedure	1. Launch the system using the start script.
Test material used	Start script
Expected result (test oracle)	System is operational and the user can begin a meditation session.
Severity	Minor
Created by	Adam Seidel
Test environment(s)	Windows 10, Unity, Oculus VR

Test Case ID	sys_test-02
Description of test	The system should have a high enough framerate to not cause nausea.
Related requirement spec/design spec details	NFR 1.1
Pre-requisites for test	The system is operational with the Oculus VR headset connected
Test procedure	1. Start the system 2. Enter a meditation session 3. Complete meditation session
Test material used	Oculus VR
Expected result (test oracle)	The framerate of the system does not fall below 60 for more than 5% of the meditation session.
Severity	Extreme
Created by	Adam Seidel
Test environment(s)	Windows 10, Unity, Oculus VR

Test Case ID	sys_test-03
Description of test	Data stored by the system must not be personally identifiable.
Related requirement spec/design spec details	NFR 4.1
Pre-requisites for test	System is operational.
Test procedure	1. Carry out and complete a meditation session. 2. Assess the JSON log file for the previous meditation session. No information that could identify the user can be stored in the file.
Test material used	JSON session file
Expected result (test oracle)	JSON session file does not contain any personal information about the user
Severity	Moderate
Created by	Adam Seidel
Test environment(s)	Windows 10

Test Case ID	sys_test-04
Description of test	Certain parameters can be set by the researcher via a config file.
Related requirement spec/design spec details	FR 3.2
Pre-requisites for test	System is not operational.
Test procedure	1. Adjust parameters in the config file. 2. Launch the system. 3. Begin a meditation session. 4. Verify that the parameters changed are being reflected in the meditation session.
Test material used	Config file
Expected result (test oracle)	The change of session parameters are reflected in the meditation session
Severity	Moderate
Created by	Adam Seidel
Test environment(s)	Windows 10

Test Case ID	sys_test-05
Description of test	The plant growth rate must be affected by the user's meditation quality.
Related requirement spec/design spec details	FR 1.3
Pre-requisites for test	The system is operational, and the user is participating in a mediation session.
Test procedure	1. Follow the on-screen guidance to carry out a period of high-quality meditation. 2. Ignore the on-screen guidance and purposely carry out a period of low-quality meditation.
Test material used	Oculus VR
Expected result (test oracle)	For procedure 1., the growth rate of the plant should be greater to reward meaningful meditation periods. For procedure 2., the growth rate of the plant should be less to reflect the low quality period of meditation.
Severity	Severe
Created by	Adam Seidel
Test environment(s)	Windows 10

2.4 - User Acceptance Testing (UAT)

Test Case ID	usr_test-01
Description of test	The audio is perceived as calming.
Related requirement spec/design spec details	NFR 3.3
Pre-requisites for test	The music system for the game is operational (ideally the game too).
Test procedure	Play the music to multiple users and ask them how relaxing the music is on a scale of 1 to 5.
Test material used	Oculus VR or headphones/ speaker
Expected result (test oracle)	The average score should be 4 or above to pass this test.
Level of severity	Severe
Comments	We have marked unt_test-04 as Moderate severity and this severity as Severe. This is because a lack of audio would not have a massive impact on the user experience, but an unnerving audio would have a major impact.
Created by	Thomas Owen
Test environment(s)	Windows 10

Test Case ID	usr_test-02
Description of test	The visual style of the game is cohesive.
Related requirement spec/design spec details	NFR 3.1
Pre-requisites for test	All parts of the game must be able to be run.
Test procedure	1. Ask users to navigate the game's title and menu screens and then do a meditation session. 2. Ask the users to rate the game on a scale of 1 to 5 based on how cohesive they think the style is.
Test material used	Oculus VR
Expected result (test oracle)	The average score should be 4 or above to pass this test.
Level of Severity	Minor
Comments	None
Created by	Thomas Owen
Test environment(s)	Windows 10

Test Case ID	usr_test-03
Description of test	The researcher configuration file should be easy to use.
Related requirement spec/design spec details	NFR 2.2
Pre-requisites for test	A fully operational config file.
Test procedure	1. Ask the client to use the configuration file to set up their desired meditation environment. 2. Ask the client if they like the design and setup of the config file.
Test material used	PC
Expected result (test oracle)	The client must be happy with the config file.
Level of Severity	Minor
Comments	None
Created by	Thomas Owen
Test environment(s)	Windows 10

Test Case ID	usr_test-04
Description of test	The audio must fit the setting.
Related requirement spec/design spec details	NFR 3.2
Pre-requisites for test	The music system and the meditation environment must both work.
Test procedure	1. Ask multiple users to participate in a meditation experience. 2. After the session, ask them to rate how well the audio fits with the setting on a scale of 1 to 5.
Test material used	Oculus VR, headphones/ speaker
Expected result (test oracle)	The average score should be 4 or above to pass this test.
Level of Severity	Moderate
Comments	None
Created by	Thomas Owen
Test environment(s)	Windows 10

Test Case ID	usr_test-05
Description of test	The user should feel that the plant growth represents how relaxed they are.
Related requirement spec/design spec details	None
Pre-requisites for test	There must be a fully operational game with plant growth and an environment.
Test procedure	1. The user must experience at least one meditation session. 2. After the session the user will be asked if they thought the plant growth represented how relaxed they felt during the session.
Test material used	Oculus VR
Expected result (test oracle)	The overall feedback should be that the plant growth did represent how relaxed they felt.
Level of Severity	Minor
Comments	This level of severity is Minor because the user should not know that the plant growth is representing their stress level during the meditation session.
Created by	Thomas Owen
Test environment(s)	Windows 10

Section 3 - Testing context

This section includes details of the testing process and what is required for them to be carried out. It will further expand upon testing prerequisites and testing procedure.

3.1 - Testing Environment

All tests require the project running through Unity; we will use Unity version 2022.3.14f1 running on a Windows 10 operating system. The device will be connected to an Oculus Quest 2 as well as an Emotiv BCI EPOC X 14 channel wireless EEG headset. This will be set up prior to any testing being carried out.

The Unity program will receive data from the BCI Headset through the BCI-OSC API which will be recorded within the Unity program.

3.2 - Testing Procedure

Unit, System, and Integration tests will be carried out by group members, with some observing whilst another utilizes the equipment to provide results. User Acceptance Testing will be done with the client and volunteer test subjects, with their feedback documented in survey format for each test focus.

All tests will be undertaken manually with the startup script being run to initialize the BCI application and Unity game. Those tests which require editing of the configuration file (namely: unt-test-03; unt-test-04; and int-test-02) will need further user input before commencing the test.

3.3 - Testing Success

Success of Unit, System and Integration tests will be dictated by the group members observing if they sufficiently satisfy the expected outcomes.

User Acceptance Tests will be considered a success if the majority of those volunteering give positive responses to the criteria specified.

3.4 – Severity of Failure Overview

Each test is categorized with the severity in the case of failure. These metrics are ranked in severity in a descending order from least to most Severe. This table also includes reference to the relevant tests.

Level of Severity	Description	Relevant Tests
1. Minor	These errors are not critical to the overall function of the program and are low priority to be rectified.	sys-test-01, usr-test-02, usr-test-03, usr-test05
2. Moderate	These errors can provide issues for the overall function and user experience. Any failure should be corrected	unt-test-03, unt-test-04, unt-test-05, int-test-02, sys-test-03, sys-test-04, usr-test-04
3. Severe	These errors provide significant issues toward the function of the program or greatly impact the user experience. Failures must be corrected before handover.	unt-test-02, int-test-01, sys-test-02, sys-test-05, usr-test-01
4. Extreme	These errors are a critical fault in the operation of the program. Any failure in this level requires immediate attention.	unt-test-01