The Test Plan is Due January 26.

Group Assignment in 3 Parts.

Part I. Description of Overall Test Plan

Our project consists of two parts, the AI image creation and the gallery itself. These two sections will be tested individually first and then together as they are integrated. For the AI image creation, Python will be used, and it is expected that the code will run and produce images before it is integrated into the gallery. For the gallery, Unity and C# will be used and it is expected that it will function and display images before the Python code is incorporated. After the two parts have been incorporated there will be a third focus of testing, on the integration. This is to ensure that all code continues to run correctly and as expected once run together.

Part II. Test Case Descriptions

List a series of 10-25 tests for validating your project outcomes. For each test case provide the following:

1. test case identifier (a number or unique name)
2. purpose of test
3. description of test
4. inputs used for test
5. expected outputs/results
6. normal/abnormal/boundary case indication
7. blackbox/whitebox test indication
8. functional/performance test indication
9. unit/integration test indication

Note that some of these categories may be inappropriate for your project and may be omitted if you can justify doing so. For items 6-9, only one term should apply.

1. **Picture Display Test**
2. Test whether an image appears correctly in the provided spot.

3. An image will be loaded in from a file to Unity using C# and displayed in the gallery. The image size or displayed picture is not known before executing.

4. Picture file location

5. Display of the picture in the gallery

6. Normal

7. Blackbox test

8. Functional test

9. Unit test

1. Python Integration Test

2. Test to ensure that python code is integrated correctly into the c# code.

3. From Unity, using C#, the python code will be run, either as an executable or as direct code. The python code functionality is not known before executing.

4. Python code file location or python executable file location

5. It is expected that the python code will run and not cause problems with other C# code

6. Normal

7. Blackbox

8. Functional test

9. Integration test

1. Motion Sickness Test

2. Test for how long it takes an average player to become motion sick in program

3. Have multiple people run the VR program and determine when they begin to feel motion sick and need to stop with and without preventative measures.

4. This is not a technical test, so there aren't really any inputs besides preventative measures

5. Time it takes for a person to become sick while using the program, we want this time to be below the time it takes for a user to finish using the program.

6. Boundary test

7. Blackbox test

8. Performance test

9. Integration test

As this isn't a technical test and is more focused on player interaction, some of the categories don't fit perfectly with this test.

1. Graphical Initialization Test

2. Does the application initialize all graphics/graphical related entities correctly?

3. This test is to check whether textures, meshes, lighting, AI generated photos have loaded correctly upon launching the app.

4. Inputs: Preloaded (or proactively loaded [this is less important for this particular test - but in case we cache anything, it is still good to check]) textures, photos, lighting

5. Output: An accurately rendered scene

6. Normal

7. Blackbox

8. Functional

9. Integration (technically based on the definition the assignment gave, this would be integration)

1. Overall Performance Test

2. This will test the overall performance of the application.

3. This test will inform us of the proper values to use that will enhance performance as much as possible, while also keeping crashes, halts, etc. to a minimum

4. Inputs: A build (any) of the application, user's choices for AI image generation

5. Output: Raw performance data that we will then use to inform future decisions on performance/optimization

6. Boundary

7. Blackbox

8. Performance

9. System Testing

1. VR Actions Test

2. This will test the functionality of the virtual reality components

3. This test is to make sure the user can perform all expected VR actions (movement, selection, vision, etc.)

4. Inputs: Buttons from VR controller(s), Motion from VR controller(s)/Headset

5. Output: Performance of the expected action based on the input (i.e. joystick to move, look left, pointing at and clicking on a button)

6. Normal

7. Blackbox

8. Functional

9. Integration (technically System test, but based on definition of the assignment it is integration)

1. Image Save Test

2. Test whether images are saved correctly to users computer

3. This test ensures that images users may want to save are saved from the program to their computer for later access.

4. Save location for the images

5. Images saved to the save location

6. Normal
7. Blackbox
8. Functional
9. Unit

1. Word Selection Test
2. Tests the user's ability to select a word from the dropdown menu of words.
3. The user will start the application and attempt to open the dropdown menu and select a word to use for AI art generation.
4. Inputs:
5. Output: The word is selected.
6. Normal
7. Blackbox
8. Functional
9. Integration

1. Image Generation Test
2. Tests the application's ability to generate an image.
3. The user will select a word to use for AI art generation and prompt the program to generate an image. The test will pass if the program is able to generate an image.
4. Inputs: The selected word.
5. Output: The AI-generated image.
6. Normal
7. Blackbox
8. Functional
9. Integration

1. Image Accuracy Test
2. Tests the application's ability to generate accurate representations of the word selected by the user.
3. The user will select a word to use for AI art generation and prompt the program to generate an image. The test will pass if the AI-generated image accurately represents the word selected.
4. Inputs: The selected word.
5. Output: The AI-generated image.

6. Normal
7. Blackbox
8. Performance
9. Integration

Part III. Test Case Matrix: summarizes the test case coverage (items 1, 6-9 in a tabular format)

	Normal/ Abnormal/ Boundary	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
Picture Display Test	Normal	Blackbox	Functional	Unit
Python integration test	Normal	Blackbox	Functional	Integration
Motion sickness test	Boundary	Blackbox	Performance	Integration
Graphical Initialization Test	Normal	Blackbox	Functional	Integration
Overall Performance Test	Boundary	Blackbox	Performance	System Test
VR Actions Test	Normal	Blackbox	Functional	Integration Test
Word Selection Test	Normal	Blackbox	Functional	Integration Test
Image	Normal	Blackbox	Functional	Integration Test

Generation Test				
Image Accuracy Test	Normal	Blackbox	Performance	Integration Test
Image Save Test	Normal	Blackbox	Functional	Unit Test

Execution of Tests:

Many of the features we needed to test were dependent on separate things that other tests cover, creating a sort of hierarchical chain of tests. As you can see below, with the 'highest level' test at the top, as long as that specific test passes, you can logically assume all tests under it pass as well. Of course, on other occasions, these child tests are executed individually, but to save time, that wasn't always the case.

If Picture Display Test works, then we can assume that the Python and the Graphical Init worked, since Python is needed to generate a picture, and of course in order to see anything at all, the Graphics for everything needs to be initialized.

Below are all related tests:

Picture Display Test / Image Generation Test
|_ Word Selection Test
|_Python Integration Test
|_Graphical Initialization Test

Motion Sickness Test / VR Actions Test
|_Overall Performance Test
Note: Either one could be parent here, as better performance keeps Motion Sickness low, but low motion sickness causes a perceived better performance.

Image Accuracy Test
|_Image Save Test
Note: Either one could be parent here, as these were frequently tested together, though are technically separate features

Below are results of tests:

TEST NAME	RESULT	NOTES
Picture Display Test	Passed	Worked even without the Python, easy to test with placeholder images.

Python integration test	Passed	This was the most challenging feature to add. Testing was frequent
Motion sickness test	Passed	As optimization improved, motion sickness decreased
Graphical Initialization Test	Passed	This test technically was executed each time we launched, and of course as long as we saw something, it passed.
Overall Performance Test	Passed	Letting the application run for an extended period of time revealed some major design flaws that ended up being changed for the expo version.
VR Actions Test	Passed	N/A, straightforward testing
Word Selection Test	Passed	Word was successfully identified by the program and returned an expected output
Image Generation Test	Passed	Typing in a prompt returned an image from python
Image Accuracy Test	Passed	After some testing and including negative prompts, typing in a prompt gave us an image related to the prompt. Still does poorly with abstract ideas or not well known items.
Image Save Test	Passed	Image successfully saved to the computer