Raytracer Project Test Plan:

Requirements:

1. The raytracer should generate scenes based on JSON input files.
2. The rendered images are produced within 30 seconds of execution

Requirement 1 Testing Specification: JSON Input Processing:

JSON Reading Tests:

- jsonFileExistsTest: Validates that the raytracer can correctly detect the existence of the specified JSON input file.
- jsonFileContentValidationTest: Checks that the parser identifies the correct number of each object in the input file

Integration Generation Tests:

- sceneObjectCreationTest: Ensures that the raytracer can create scene objects accurately based on the parsed JSON content.
- sceneComplexityTest: Validates the raytracer's ability to handle scenes of varying complexities, including scenes with a large number of objects.

Requirement 2 Testing Specification: Rendering Performance:

Rendering Time Tests:

- basicSceneRenderingTest: Evaluates the rendering time for a basic scene with a few objects to ensure it falls within acceptable limits.
- standardSceneRenderingTest: Tests the rendering time for the given example scenes, ensuring performance scalability.

Load Testing:

- shapesRenderingTest: Introduces stress tests by rendering larger scenes with more and more shapes to assess if or when the ray tracer cannot render the scene under 30 seconds
- lightSourcesRenderingTest: A similar stress test but increasing the number of light sources in the scene.

Test-Driven Design Evolution:

Use Case: Implemented optional requirement for multiple samples per pixel
Impact Analysis: Any image can have the samples per pixel increased without a limit. 2 samples per pixel will invoke the same amount of rendering as 2 images since each pixel is calculated twice.
Iteration of test plan: Define threshold for 30 second rendering with 1 sample per frame. Observe and predict impact of multiple samples per pixel on render times.

Use case: Mirrors reflecting mirrors
Impact Analysis: When rays are caught between mirrors they might end up needing hundreds of traces to find the final destination of the ray. This has been addressed in the code by adding a number for depth which stops the ray after n bounces. This hasn’t been addressed in the render time tests which could lead to inconsistent findings. The standard use of the raytracer should

have the depth set quite low so this case should not happen in typical usage unless the user is looking for unusually deep mirror reflections.
Iteration of test plan: Proceed with all tests using a standard depth around 8 reflections. Impact of increasing it further can only be observed when increasing depth much larger and in specific scenes. For full coverage it should also be investigated when this occurs but for the scope of this it will be left as an edge case.