**Software Implementation and Testing Document**

**For**

**Group <4>**

Version 1.0

**Authors**:

Matthew B

John R

Mina A

Ethan P

Jared G

# Programming Languages (5 points)

The programming language that is being used in this project is C# thus far because it is the best language for physics in Unity.

# Platforms, APIs, Databases, and other technologies used (5 points)

This project is being developed using the Unity game engine. Most assets for this project were obtained from the Unity Asset Store. The main character and enemy assets were obtained from a third party website. The level assets were also obtained from a free third party source.

# Execution-based Functional Testing (10 points)

**Collision Detection -** This was tested by initializing the game and controlling the player to jump on various objects, including the ground, floating surfaces, and enemies.

**Player Movement -** This was tested by initializing the game and using the keys ‘W’, ‘A’, ‘S’, and ‘D’ in order to move left, right, crouch, and jump.

**Attack Enemy -** This was tested by using the user’s arrow attack in order to implement a system where the enemy will take damage whenever an arrow hits it.

**Alter Enemy Health -** This was tested by damaging the enemy with arrows. Everytime an arrow would hit an enemy their health would drop until they were dead.

**Alter User Health -** To test this we implemented a function that would damage the users health bar if “tab” was pressed. This shows us the player could take damage.

**Enemy Movement -** We tested this by implementing enemy characters and making sure their movement performed as expected.

**Arrow Attack -** We tested this by having the player character perform a ranged attack within the game. And tested to make sure the arrow is destroyed on impact with an enemy.

# Execution-based Non-Functional Testing (10 points)

**Input Delay** - to test input delay we imputed commands while running our game such as jumping or shooting an attack.

**Crashes** - In order to make sure the game would not crash we played the game for a long amount of time.

**Game running on Windows and Mac** - we tested this by running the game on a windows machine and MacOS.

**Health Bar Accuracy** - We tested the health bar accuracy by making sure the enemy dies once their health reaches zero.

# Non-Execution-based Testing (10 points)

**Scripts -** The scripts were reviewed by another team member prior to execution to make sure there were no bugs that were overlooked.

**Button Combinations -** We made sure everyone had the correct button mapped to the scripts.

**Object Hierarchy -** The object hierarchy was reviewed to make sure game objects were organized in a logical structure within the project.

**Layer order -** The scene layers were reviewed to make sure they were in the correct order and that the layers are visible in the intended order.

**Scene Layout -** The layout of the levels in the scene was reviewed to ensure a balanced layout with regard to enemy frequency and varying platforms.