**Software Implementation and Testing Document**

**For**

**Group 2**

Version 3.1

**Authors**:

Zachary Ortiz

Rafe Ewert

Joe Sahl

# Programming Languages (5 points)

We are using GDScript for the entirety of the project, from the player and enemy code to the level design. We chose this language because it was the only one that would work properly with Godot that didn’t require us to download any external libraries. (Z)

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

The game is still being built in Godot, a game designer engine. (Z)

The sprites for the environment are still used from the Spriters Resource. (R)

# Execution-based Functional Testing (10 points)

I tested the player’s control over Mario through constantly writing code and testing if the key inputs worked correctly. I have also tested Mario’s collision through code writing and slinging him in to numerous objects, such as enemies and items, before eventually getting it to work right. (Z)

I also tested Mario’s interactions with enemies and Bowser, as well as their interactions with the environment. I also tested the transitions between levels and the UI for the game. (J)

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

I have tested Mario’s behavior across levels, and it works as intended. The game does not crash, so there is hopefully no unexpected behavior anywhere in Mario’s code. (Z)

I have tested to make sure behavior is as expected across all three levels. I did not experience any bugs or crashes. (J)

I have tested on Microsoft Windows through parallels to ensure the game works on that OS. (R)

# Non-Execution-based Testing (10 points)

I have done numerous walkthroughs and inspections of my own code while testing both functional and non-functional requirements. (Z)

I made sure that there were no conflicts or bugs across all three levels by reading through the code. (J)

I have done walkthroughs of my code as well as double checking all three levels look and operate as expected. (R)