**Software Implementation and Testing Document**

**For**

**Group 4**

Version 1.0

**Authors**:

Andrew F

Bret S

William H

William S

# Programming Languages

For this project we used the C# programming language. C# was chosen for its heavy use with the gaming industry and being one of the languages that was more compatible with the Unity engine. We also felt a higher-level language might be easier to work with since none of our members know anything about unity. Working with low level language such as C++ on top of our unfamiliarity with unity seemed like a heavy load to carry for our first unity project. C# was mostly used in writing scripts that were imported into our game engine. These scripts were used for character interaction, enemy interaction, and various other aspects.

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

The game engine unity was used along with the C# programming language. No API’s or Databases were used as a of yet as there has not been a real need for such resources. We aimed for our second iteration to be a to have more enemies, GUI’s, and etc. But as of now none of these additions have called for such technologies as Databases or API’s external to our unity game engine have been needed. Also, we did not want to add in additional technologies when we were not completely sure how said technology might interact with our scripts as of yet.

# Execution-based Functional Testing (10 points)

We executed several system testing’s searching for functionality of shooting and killing enemies. In addition, we also tested for the ability of our enemies to detect and pursue the player for a certain distance. The testing took place by building and running the scenes and slowly adding additional features such as enemies, death animations, etc. We also tested by using serialized fields for values to determine how fast enemies should move, how high the fire rate for weapons should be, and how high our player could jump during certain situations.

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

There has not been a lot non-functional testing with our project. Within this iteration we felt that functional testing held a higher priority. That non-functional testing would come later down the project pipeline when we added more UI elements. As a result, we ended up focusing more on the functional testing. Moving the non-functional testing down the pipeline util we have achieved more elements into our game and once completion of that takes place we plan to move forward with more non-functional testing.

# Non-Execution-based Testing (10 points)

For the non-execution testing there was a lot of code inspection among our members. For example, since we wrote a lot of the code while in communication, we would often end up asking other members to review and even look over one another’s code before a push or pull was made to our GitHub repo. Another aspect of the non-execution testing was constantly doing code walkthroughs before adding scripts to the unity project. For this project it was very big issue that our logic was clear and sound. Such as when we created our state machine for the project it had to be walked over several times before we figure out the correct states for all our animations.