**Software Requirements and Design Document**

**For**

**Group <32>**

Version 1.0

**Authors**:

Merrick M

Derek T

Dillon M

Isaac M

Jacob C

# Overview (5 points)

Undead Alliance is a Co Op 2D top-down zombie survival shooter. The player and up to 3 other teammates will survive waves of spawning zombies that they must shoot, until all members are dead. There will be a variety of different guns used, and new guns will be acquired as the game goes on. Upon killing zombies, the players will gain XP points which can be used to level up to increase stats as well as gain abilities. There is no end to the game, and it will be endless, and the goal is to last as long as possible.

# Functional Requirements (10 points)

User should be able to change game difficulty in the settings menu - Low

Zombies should chase players continuously until eliminated- High

Zombies should deal melee damage when at striking distance of a player - High

Player should be able to deal damage to zombies with weapons - High

Player should be able to pick up different weapons – Medium

Player should be able to level up from a given number of kills – Medium

Weapons should display remaining ammo – Low

Weapons should stop firing when they run out of ammo - Medium

Players should be able to reload weapons if ammo reserve is not empty – Medium

Players health should deplete upon damage from enemies – High

Players should have a standard health that regenerates over time and an additional amount of health that comes with leveling up - Low

Non functional requirements will include things like quick updates for important numbers and information like position of players, enemies, updating inventory upon picking up a gun, and bullet collision recognition. Quality is also important; the game should feel smooth and the player should never be confused on what to do.

# Use Case Diagram (10 points)

As this is a video game, the only real use case is to play for fun. The more fun the game is, the more successful the product is

# Class Diagram and/or Sequence Diagrams (15 points)

Currently, our main classes are Player, Enemy, Weapon, and UI.

These all have scripts that interface with each other. Essentially the main paradigm is object orientated, although a lot of that overhead in terms of inheritance are handled by the unity editor, which automatically organizes a lot of the game elements that we are building off of. In the only use case which is playing the game, all of these classes, or Game Objects as they are referred to are scripted to interact with each other and given behavior using C# code. A sequence diagram is hard to really create in this context, as the classes are all inherited from the Unity Engine’s overhead. But our main classes at this point are stated above.

# Operating Environment (5 points)

The operating environment is any PC running the game. Any Windows PC should be able to play this game without issues.

# Assumptions and Dependencies (5 points)

We are using Unity Engine, a 3rd party software and are relying on free assets for graphics. We are assuming we will be able to have a smooth first version of the game with working mobs and guns, and that our code will run together in unity when we combine things like our zombie asset and our shooting script.