Name Zach Haynie Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

## Brief introduction \_\_/3

My feature for InBetween will be the leveling / experience system to scale the character’s ability as the game continues.

The player will gain experience as enemies are defeated. This experience will scale the character’s stats such as health, damage, and movement and attack speed variables. I will design a UI to display the user’s experience and determine the rate in which it is earned and scales.

Additionally, I will oversee the upgrade system. In contrast to percentage-based scaling, upgrades will refer to alterations to the characters abilities entirely. This will include changing a sword strike to something such as an axe, dual wielded swords, or items that provide an advantage.

## Use case diagram with scenario \_\_14

### Use Case Diagrams

A diagram of a game

Description automatically generated

### Scenarios

**[You will need a scenario for each use case]**

**Name:** Experience System

**Summary:** If the player defeats enough enemies, the experience variable increments and levels up the player. Upon level up, the player is given a choice between possible upgrades.

**Actors:** Player.

**Preconditions:** Game has begun

**Basic sequence:**

**Step 1:** Enter room generated with enemy spawns.

**Step 2:** Earn XP upon dealing a finishing blow to an enemy.

**Step 3:** The XP counter increments until >= the next integer.

**Step 4:** The Player’s level count and UI increment, displaying a canvas with upgrades to choose from.

**Exceptions:**

**Step 1:** Player chooses to skip upgrades

**Step 2:** Player health reaches zero, game and experience variables reset.

**Post conditions:** Calculated value is displayed.

**Priority:** 2\*

**ID:** C04

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

Level 0 diagram

A diagram of a game

Description automatically generated



Level 1 Diagram within Section 4: Leveling

A diagram of a computer program

AI-generated content may be incorrect.

### Process Descriptions

initializeGame()

enteredRoom()

If (enteredRoom = true){

spawnEnemy()

EarnXP();

If (enemy.healthPoints = 0)

player.xp += enemy.xpTable( x )

LevelUp()

If(player.xp += levelup.threshold)

Console.Writeline(“Level up! Your stats have increased”)

Player.stats += (stats \* 1.15)

Console.Writeline(“Choose an upgrade”)

OfferUpgrades()

OfferUpgrades();

upgradeChoices = upgrades.OrderBy( upgrades[x] ) // list 3 upgrades

for( int i = 0 ; i < 3 ; i++)

Console.WriteLine(“{upgradeChoices[i]}”)

Int choice = GetPlayerChoice(user.input)

ApplyUpgrade(player.upgrade(choice[x])

## Acceptance Tests \_\_\_\_\_\_\_\_9

Most of the testing will revolve around ensuring that the relative upgrades and increases are applied correctly. First, the XP / UI should accurately depict the player’s current level, and the progress toward the next. When the player defeats an enemy, XP variable is incremented and updated in UI. When XP threshold is reached, LevelUp UI should pause the game, notify the user of the level increase, and offer upgrades.

Ensure the upgrades offered are truly pseudorandom to ensure variability. When an upgrade/skip is chosen, apply the chosen upgrade to the user’s playthrough. Ensure the upgrade persists and works as designed. Repeat testing with each upgrade applied, and all possible combinations of upgrades in unison to ensure they all work in cohesion. Check for all contradictions and interactions with all enemy types, sprites, etc.

|  |  |  |
| --- | --- | --- |
| Input | Output | Notes |
| Enemy slain | Player awarded XP resonant of difficulty |  |
| Player passes upgrade threshold | Player level UI increases  Upgrades offered  Player stats increase |  |
| Player Health reaches 0 | XP value reset  Upgrades removed | Reset all stats/upgrades to base character |
| Player Chooses Upgrade | Character awarded proper buff | “Skip Upgrade” should always be available |
| Player leaves current room | Upgrades persist through various rooms |  |
| Player completes game | Display Victory screen with all current upgrades, remove all upgrades for next playthrough |  |

## Timeline \_\_\_\_\_\_\_\_\_/10

|  |  |  |
| --- | --- | --- |
| Tasks | Hours to complete | Predecessor Task |
| XP table mapping | 2 | - |
| Upgrade Drafting | 3 | - |
| XP UI | 2 | 1 |
| Upgrades UI | 3 | 1,2,3, |
| Upgrade Integration in Unity | 8 | 1,2,3,4 |
| Compatibility Testing | 4 | 4,5 |
| Error Documentation and Fixing | 6 | 1-6 |
| Artwork | 4 | 7 |

### Pert diagram

### A diagram of a number AI-generated content may be incorrect.

### Gantt timeline

A close-up of a graph

AI-generated content may be incorrect.