Name: Bidhi Paudel Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

My primary responsibility will be to define and implement the weapon mechanics and power-ups that enhance the player’s combat capabilities. Weapons will be key items that players can equip from their inventory for combat, contributing significantly to the overall gameplay experience. Each weapon will feature distinct characteristics, such as damage output, projectile type, firing speed, cooldown rates, and range, offering players a variety of strategic options.

Weapons will not require ammunition, allowing players to use them freely as long as they are equipped. I will focus on designing a system where players can discover and obtain weapons by interacting with the level environment—finding them hidden in the world or unlocking them through progression. Once equipped, weapons will allow players to engage in combat by interacting with defined inputs, such as a key press (e.g., spacebar), to shoot projectiles at enemies.

Each weapon's **projectile behavior** will be carefully defined, including range, hitbox interaction with enemies, and damage stats. The goal is to ensure that projectiles behave consistently and cause the appropriate damage when hitting enemy targets, contributing to a balanced and satisfying gameplay loop. For example, weapons may include guns with wide bullet spreads for crowd control or precision projectiles for targeting specific enemies, all tailored to the game’s dystopian narrative and fast-paced action.

## Use case diagram with scenario \_\_14

### Use Case Diagrams

A diagram of a weapon effects

AI-generated content may be incorrect.

### Scenarios

**Name:** Engage Combat

**Summary:** The player uses an equipped weapon to attack enemies and engage in combat.

**Actors:** Player, Enemy/Boss

**Preconditions:**

The player has a weapon equipped

The enemy/boss is within range of the player’s weapon

**Basic sequence:**

1. Player initiates an attack by pressing the assigned input
2. The weapon type is checked to determine projectile behavior (speed, range, damage)
3. A projectile is generated and fired from the players position toward the target
4. Collision detection evaluates whether the projectile hits the enemy.
5. If a hit is detected:
   1. Calculate the damage based on the weapon’s stats and any bonuses or modifiers.
   2. Apply damage or effects (e.g., poison, burn) to the enemy.
6. Update the enemy’s health bar or status based on the calculated damage.
7. If the enemy’s health reaches zero, trigger a defeat sequence (e.g., death animation, loot drop).

**Exceptions:**

1. Player initiates an attack with no weapon equipped: Display an error message or ignore the input.
2. Enemy is outside the weapon’s range: Projectile disappears after traveling the maximum distance.
3. Weapon cooldown is active: Ignore the attack and display a "weapon cooling down" message.
4. Collision detection fails or misses the target: Projectile continues or disappears without causing damage.

**Post conditions:**

1. The projectile’s impact is resolved (hit or miss).
2. Enemy health and status are updated, reflecting damage or effects from the projectile.
3. If the enemy is defeated, trigger relevant events (e.g., animations, loot drops, progression updates).

**Priority:** 1

**ID:** P01

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

[Get the Level 0 from your team. Highlight the path to your feature]

Example:

### A diagram of a system AI-generated content may be incorrect.Data Flow Diagrams

A diagram of a system

AI-generated content may be incorrect.

### Process Descriptions

**(3p.1) Combat Manager**  
 function EngageCombat():

IF PlayerInitiatesAttack():

HandleProjectile()

End

function HandleProjectile():

IF DetectCollision(Projectile, Enemy):

ApplyDamage(Enemy, CalculateDamage(Projectile))

ELSE:

DestroyProjectile(Projectile)

End

**(3p.2) Projectile Handler**

function CreateProjectile():

ProjectileType = GetWeaponType(PlayerWeapon)

SetProjectileStats(ProjectileType)

Return Projectile

End

function LaunchProjectile(Projectile):

SetProjectileVelocity(Projectile)

MoveProjectile(Projectile)

End

**(4p.3) Damage and Collision Manager**

function ApplyDamage(Enemy, Damage):

Enemy.Health -= Damage

IF Enemy.Health <= 0:

TriggerDeathSequence(Enemy)

ELSE:

TriggerEnemyReaction(Enemy)

End IF

End

function CheckCollision(Projectile, Enemy):

IF DetectEnemyHit(Projectile, Enemy):

Return True

ELSE:

Return False

End

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

• Weapon Type Behavior:

o Correct projectiles are fired for each weapon type.

o Switching weapons during combat must immediately reflect the change.

o Correct weapon type behavior is observed in 99% of cases.

• Powerup Effects:

o Powerup effects are applied correctly and last for the defined duration.

o Effects do not persist after their timer expires.

o Powerup effects behave as expected 100% of the time

**Example for divide feature**

|  |  |  |
| --- | --- | --- |
| Output | Input Command | Notes |
| Projectile Fired | Player Initiates Attack | Projectile launches from the player’s position when the attack button is pressed. |
| Damage Applied | Projectile Hits Enemy | Damage is calculated based on the projectile’s stats and is applied to the enemy’s health. |
| Projectile Missed | Projectile Out of Range | The projectile disappears when it travels beyond its maximum range without hitting anything. |
| Enemy Reaction | Enemy Takes Damage | The enemy reacts to damage with a stagger or death animation based on the remaining health. |
| Projectile Destroyed | Projectile Hits Object | When the projectile collides with an obstacle, it disappears without affecting enemies. |

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

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Hours) | Prerequisite |
| 1. Define Weapon/Powerup System Scope | 2 | - |
| 2. Research Reference Games | 1 | 1 |
| 3. Create Design Document for Weapons/Powerups | 3 | 1,2 |
| 4. Develop Weapon Types and Stats | 5 | 3 |
| 5. Implement Weapon Behavior Logic | 4 | 4 |
| 6. Create Powerup Effects and Attributes | 2 | 4 |
| 7. Implement Powerup Activation System | 2 | 6 |
| 8. Test Weapon and Powerup Functionality | 3 | 5, 7 |
| 9. Adjust Weapon Balance (Damage, Range, Speed) | 3 | 8 |
| 10. Adjust Powerup Timers and Effects | 2 | 8 |
| 11. Add Visual Effects for Weapons/Powerups | 3 | 9,10 |
| 12. Integrate Audio Effects for Weapons/Powerups | 4 | 9,10 |
| 13. Conduct Playtests for Weapons/Powerups | 2 | 11,12 |
| 14. Bug Fixing and Final Adjustments | 2 | 14 |

### Pert diagram

A white paper with black text

AI-generated content may be incorrect.

### Gantt timeline

A graph with green squares

AI-generated content may be incorrect.