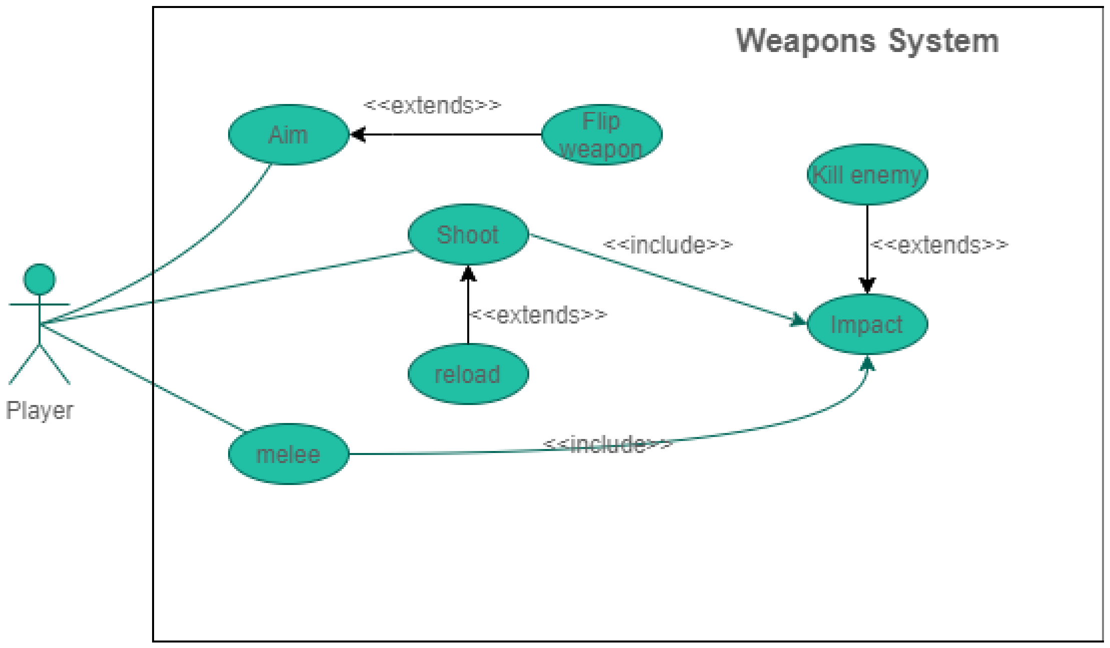
Name______**Alex Peña**___________________ Mark ______________________/50

1. Brief introduction __/3

My main feature for Super Spicy Gun Game is the weapons system. For this, my work is very broadly split into two categories. On one side of things, I'm responsible for the art and animation that pertains to the weapons system. This includes but is not limited to: firing, projectiles, impacts, explosions, and any other type of effect which we want to implement. However, while I wish to create some of my own pixel art for the weapons system, I understand that there are time constraints, and I am not an artist. For this reason, I'll also be using open-source assets as well. The main c1omponent of the weapons system, however, is programming. Our goal is to have projectile and melee weapons within the game (guns and swords, etc.), which will each have then unique weapons pertaining to each of those categories. For projectile weapons, I handle how they aim, shoot, reload, how the fired projectile traves, and what happens once it impacts (interacts with an enemy/boss; plays a particle effect, etc). Melee weapons will have the same general functionality, minus the "reloading". As our team pushes further into development, I will also be working with a game manager script, so that weapons can have a finite amount of ammo, which can be replenished from certain resources available to pick up, the exact details will be figured out later.

2. Use case diagram with scenario __14

Use Case Diagram



Scenarios

Name: Aim

Summary: The player aims the weapon equipped with the mouse.

Actors: Player.

Preconditions: Player has been placed in game, and has a weapon equipped.

Basic sequence:

Step 1: Get input from mouse.

Step 2: Aim weapon to point towards location of mouse pointer within game.

Exceptions:

None: (defined in own use case)

Post conditions: Weapon is aimed towards mouse pointer in correct orientation.

Priority: 3*

ID: Aim

*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Name: Flip Weapon

Summary: The weapon is flipped across its y-axis to maintain correct orientation, if the mouse cursor crosses the player's y-axis.

Actors: Player.

Preconditions: Player has been placed with weapon equipped, and mouse cursor has crossed the player's y-axis.

Basic sequence:

Step 1: As the player is moving the mouse cursor, it crosses the player's y-axis.

Step 2: The weapon equipped is flipped across the y-axis, so that its orientation remains correct.

Exceptions:

None

Post conditions: Weapon is displayed correctly when aiming in any direction.

Priority: 3*

ID: Flip

*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Name: Shoot

Summary: The player shoots the equipped weapon; in the direction they are aiming.

Actors: Player.

Preconditions: Player has been placed with weapon equipped.

Basic sequence:

Step 1: Based on unity's predefined inputs, the player presses a "fire" button (left click on PC, a button on joystick, etc.).
Step 2: A projectile (bullet, arrow, etc.) is created in front of the gun.
Step 3: The projectile flies forward in the direction the player was aiming.
Step 4: Calculate and show result.

Exceptions:
Step 1: The "r" key has been pressed, and the gun is reloading.
Step 2: The player cannot shoot the weapon, until the gun is done reloading.

Post conditions: A projectile flies forward in the direction the player is aiming.
Priority: 1*
ID: Shoot
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Name: Reload

Summary: The player reloads the equipped gun, and they can fire a determined number of shots again, before needing to reload later.
Actors: Player.
Preconditions: All preconditions for the player being able to shoot.
Basic sequence:
Step 1: Press "R" to begin the weapon reloading.
Step 2: An animation is played, and the player is unable to shoot the gun.
Step 3: Reloading finishes, and the player can now shoot again.

Exceptions:
Step 3: The player has no ammo left to reload the weapon.
Step 4: The reload animation is still played, the player cannot shoot their gun.
Step 5: The player can shoot, but nothing will come out, since they have no ammo.

Post conditions: Gun is reloaded, and player can shoot.
Priority: 2*
ID: Reload
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Name: Melee

Summary: The player swings their weapon (melee), and then returns to their state before swinging.
Actors: Player.
Preconditions: Player has been placed in the game, and they are equipped with a melee weapon (sword, staff, etc.).
Basic sequence:
Step 1: Press "fire" button/key which is predetermined by a set of inputs in unity (for cross-platform ease).

Step 2: The player swings their weapon.
Step 3: Any surface hit will lead to the "Impact" use case, and any Enemy hit will also lead to the "Kill enemy" use case.
Step 4: The player finishes swinging the weapon, and returns to the state before they swung.

Exceptions:

None

Post conditions: Player has swung their weapon, and returned to normal state.
Priority: 3
ID: Melee
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Name: Impact

Summary: For a projectile fired from a weapon, an impact animation or particle effect is placed at the point of impact, and the projectile is destroyed. For a melee weapon, an impact animation or particle effect is played at the point of impact, and the player then continues finishing their "melee" use case.
Actors: Player.
Preconditions: Either a projectile or a melee weapon has impacted with some object in the game.
Basic sequence:

Step 1: Projectile or melee weapon impacts an object with the game.
Step 2: A particle effect is played at the point of impact.
Step 3: If it is a projectile, the projectile is destroyed. If it is a melee weapon, the swing animation continues to play.
Step 4: The impact effect is finished, and the state of the weapon is back to where it was before this use case.

Exceptions:

Step 4: In the case of hitting an enemy, the "Kill Enemy" use case follows.

Post conditions: Projectile or melee swing has caused an impact effect to show, and the player can now fire or swing again.
Priority: 1*
ID: Impact
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

Name: Kill Enemy

Summary: The projectile or melee weapon from "Impact", collides with an enemy, and causes the enemy to die.
Actors: Player.
Preconditions: The projectile or melee weapon from "Impact", has collided with an enemy, and played its "Impact" effect.

Basic sequence:

Step 1: Interface with the enemy, to call a “die” function that is made for that enemy.

Step 2: Enemy dies.

Exceptions:

Step 2: Enemy is a boss and takes a certain amount of damage.

Step 3: Boss has taken its maximum amount of damage and dies.

Post conditions: Enemy has died.

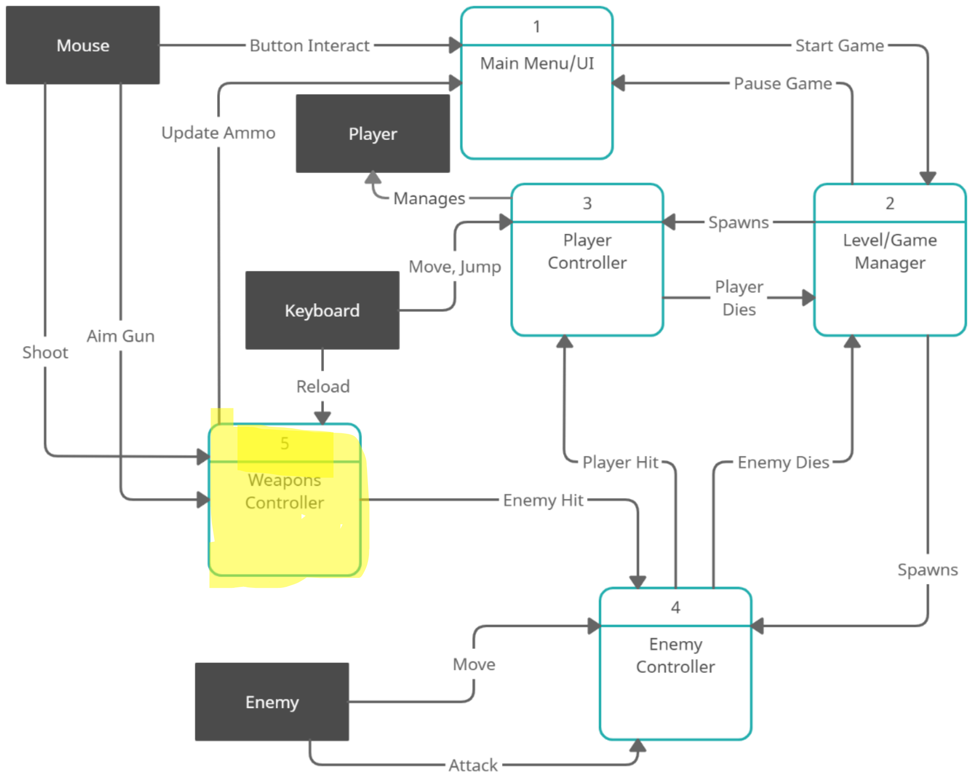
Priority: 1*

ID: Kill Enemy

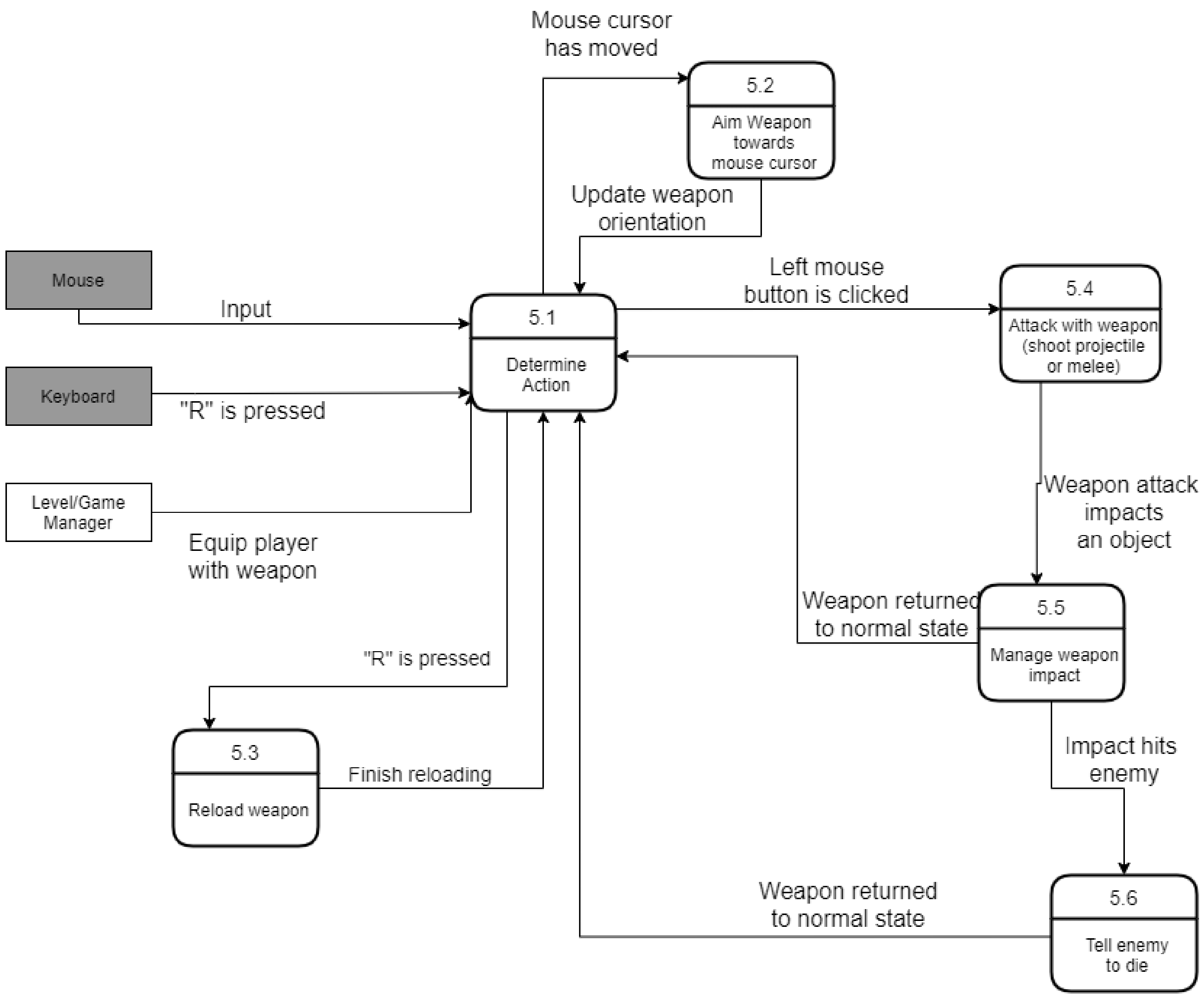
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

Data Flow Diagrams



Weapons Controller



Process Descriptions (Structured English)

```
Determine Action:
        WHILE input from mouse OR key “R” is pressed.
            Aim weapon towards mouse cursor (go to 5.2).
            IF “R” is pressed
                Reload weapon (go to 5.3).
            END IF
        END WHILE
Aim Weapon towards mouse cursor:
        Point weapon to face mouse cursor.
        Update orientation of weapon, to keep it upright.
```

```
Reload Weapon:
        Begin playing weapon reload animation.
        Stop weapon from shooting.
        Finish reloading animation.
        Allow player to shoot weapon again.
Attack with Weapon:
        IF player is holding projectile weapon
           Shoot weapon, sending projectile flying.
           Wait to shoot again for "FireRate" amount of time.
        ELSE IF player is holding a melee weapon
           Attack with melee weapon.
        ENDIF
Manage Weapon Impact:
        IF impact hits Enemy
           Play impact effect.
           Send message to enemy (go to Tell Enemy to die).
        ELSE
          Play impact effect.
        ENDIF
        IF impact is from projectile
           Destroy projectile.
        ENDIF
        Destroy impact effect.
        ENDIF
Tell Enemy to Die:
        Send message to "Enemy" to call its pre-defined "Die" function.
        Return the weapon to Its normal state.
```

Acceptance Tests _______9

Weapons System Tests

Test all processes within the weapons system, on each individual weapon. Check boundary cases by testing things such as reloading, on a melee weapon.

Run feature on each weapon in the game, testing for any inconsistencies. Each bullet point here names a part of my feature, and what output, that specific input must produce.

- Weapon Aim
  - Weapon maintains the correct orientation and rotates around the player to point towards the mouse.

- Weapon Attack

    - If projectile weapon, the fired projectile will come from the correct side of the weapon, with the correct orientation and speed. If melee weapon, an animation for attacking will play. When the weapon is done attacking, the player returns to its normal state, ready to attack again.
- Weapon Reload
    - No melee weapon will reload. For any projectile weapon, it shall reload, and the player will not have the ability to fire the weapon, while it is reloading.
- Weapon Impact
    - All attacks from weapons will have some sort of effect when they impact any object in the game. If the attacks hit an enemy, that enemy must be sent a message to die. Any particle effects generated from the attack must be deleted after they play.
- Weapon Tells Enemy to Die
    - After an enemy is hit, and a message is sent to it, there must be a pre-defined function on that enemy, so that It can die. No enemy (except possible bosses later on) shall take more than 1 hit to die.

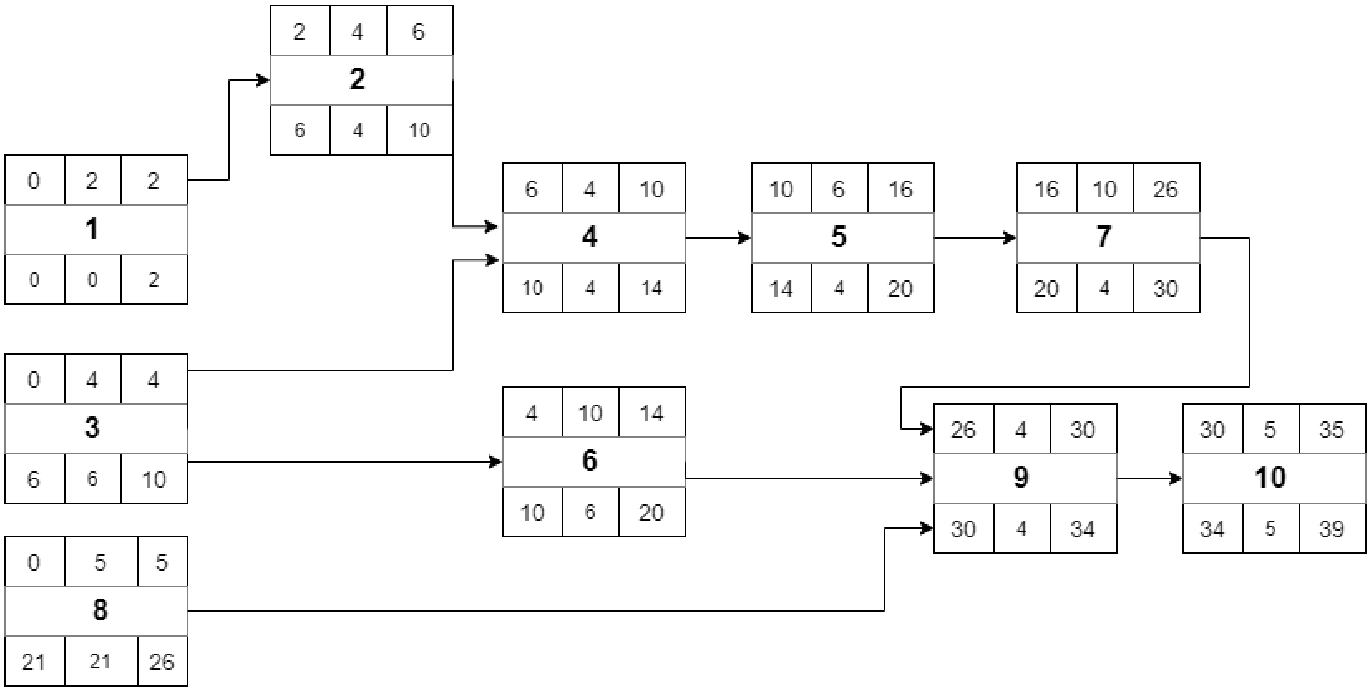
4. Timeline _______/10

Work items

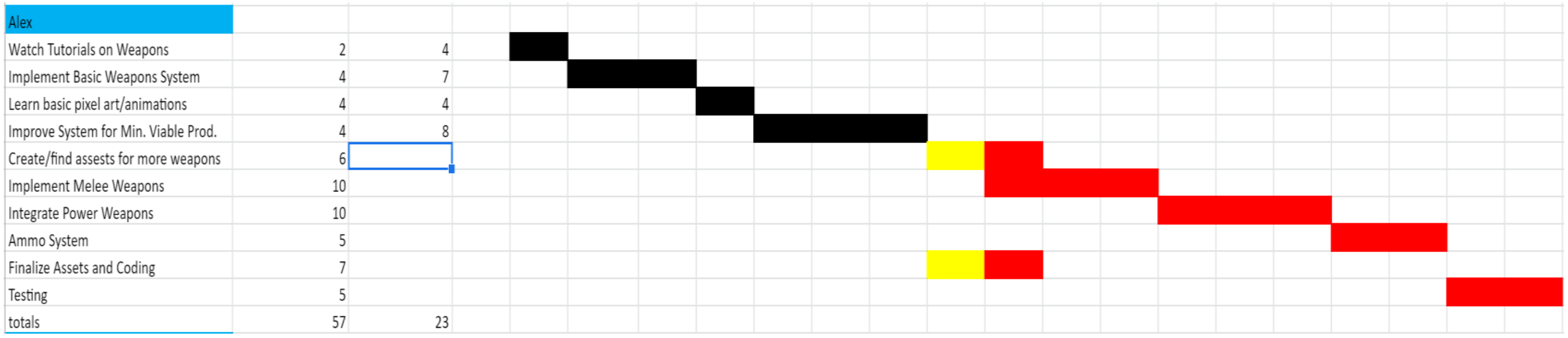
Task	Duration (Hours)	Predecessor Task(s)
1. Watch Tutorials on Weapons	2	-
2. Implement Basic Weapons system	4	1
3. Learn basic pixel art and animations	4	-
4. Improved system for minimum viable product	4	2, 3
5. Create/find assets for more weapons	6	4
6. Implement Melee Weapons	10	1,3
7. Integrate more “special” power weapons	10	5
8. Ammo system	5	-

9. Finalize assets and coding for weapons system	7	6, 7, 8
10. Testing	5	9

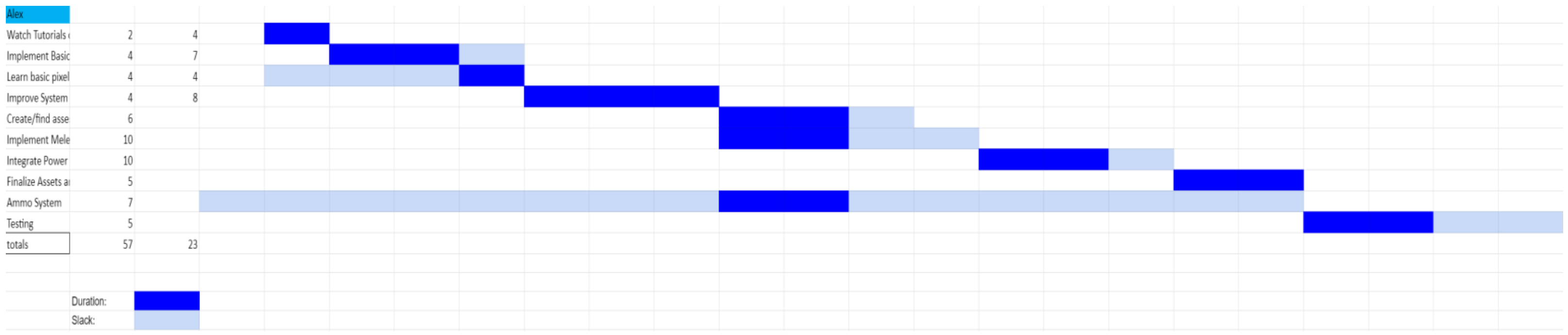
Pert diagram



Gantt timeline



Alex		
Watch Tutorials on Weapons	2	4
Implement Basic Weapons System	4	7
Learn basic pixel art/animations	4	4
Improve System for Min. Viable Prod.	4	8
Create/find assests for more weapons	6	
Implement Melee Weapons	10	
Integrate Power Weapons	10	
Ammo System	5	
Finalize Assets and Coding	7	
Testing	5	
totals	57	23



Alex		
Watch Tutorials o	2	4
Implement Basic	4	7
Learn basic pixel	4	4
Improve System	4	8
Create/find asse	6	
Implement Mele	10	
Integrate Power	10	
Finalize Assets ar	5	
Ammo System	7	
Testing	5	
totals	57	23

Duration:	🟦
Slack:	▨