Name: **Spencer Butler** Mark ____________________/50

1. Brief introduction __/3

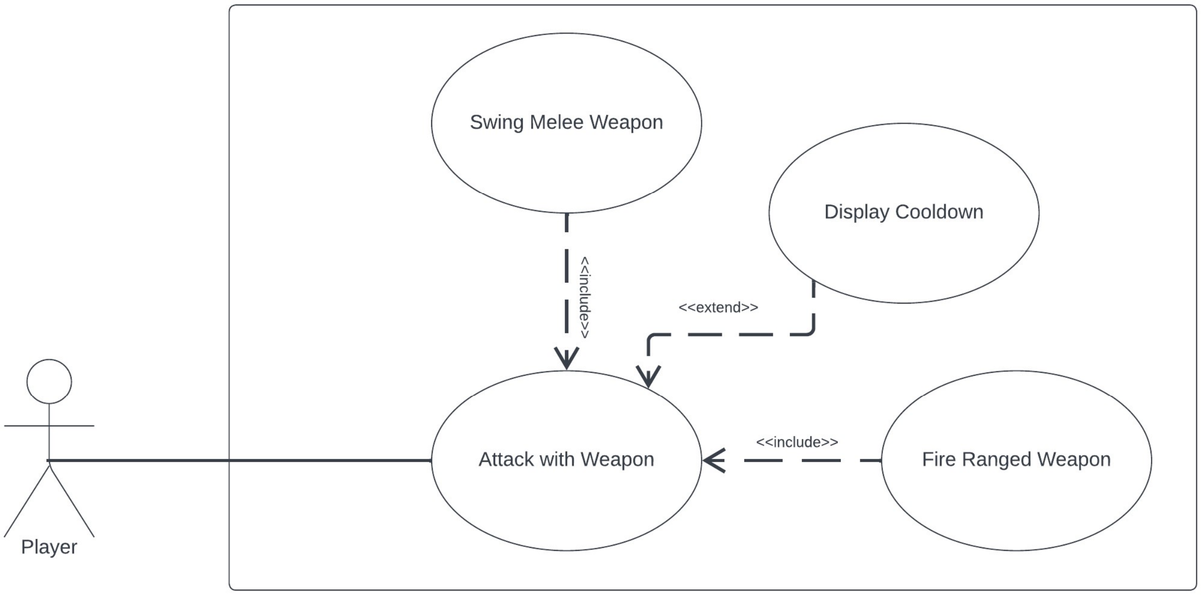
I am creating the weapons for our game. This will include the functionality when weapons are used to attack, as well as providing available weapons for purchase/pickup in shops/levels.

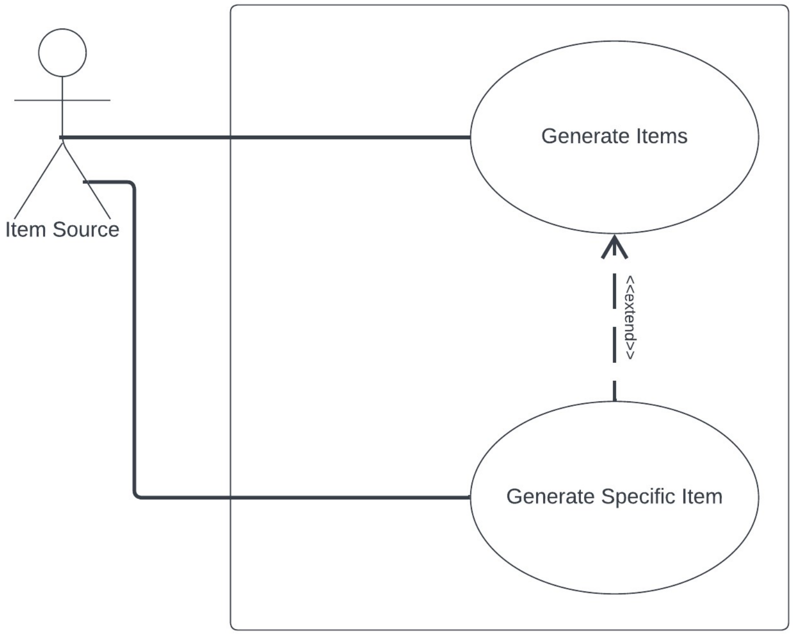
When the player attacks, depending on the type of weapon equipped, either a projectile will be created traveling away from the player, or an animation will play at the player's location. Any enemies hit by the projectile or animation will take damage.

Shops and in-level item pickups will need weapons for the player to purchase or obtain. My system provides the weapons to these item sources.

2. Use case diagram with scenario __14

Use Case Diagrams





Scenarios

Scenario 1

Name: Attack with Weapon

Summary: The player presses a button to attack. Depending on the properties of the equipped weapon, any enemies hit by the attack take damage.

Actors: Player

Preconditions: The player has an equipped weapon.

Basic Sequence:

> **Step 1:** The player attacks.
>
> **Step 2:** The properties of the currently equipped weapon are fetched.
>
> **Step 3:** A visual indicator of the attack, either a projectile or a swinging animation, will begin.
>
> **Step 4:** Any enemies the visual indicator collides with will be damaged.

Exceptions:

> **Step 3:** If the weapon's cooldown from the last attack is not over, a different animation will be played, and no enemies will be damaged.

Post Conditions: The weapon is now on cooldown and cannot be used to attack until it ends.

Priority: 1

ID: AW1

Scenario 2

Name: Generate Items
Summary: Something which provides weapons to the player, such as a shop or an in-world pickup, requests weapons to provide.
Actors: Item Source
Preconditions: The registry of weapons contains at least one weapon.
Basic Sequence:

> **Step 1:** The item source requests one or more weapons.
> **Step 2:** The weapon registry is loaded.
> **Step 3:** Weapons are randomly selected from the registry, up to the number requested by the item source.
> **Step 4:** The selected weapons are transmitted to the item source.

Exceptions:

> **Step 3:** If the item source requests particular weapons, rather than selecting weapons from the registry randomly, the particular requested weapons will be selected.

Post Conditions: The item source has weapons to provide to the player.
Priority: 2
ID: GI1

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

Data Flow Diagrams

Diagram 0

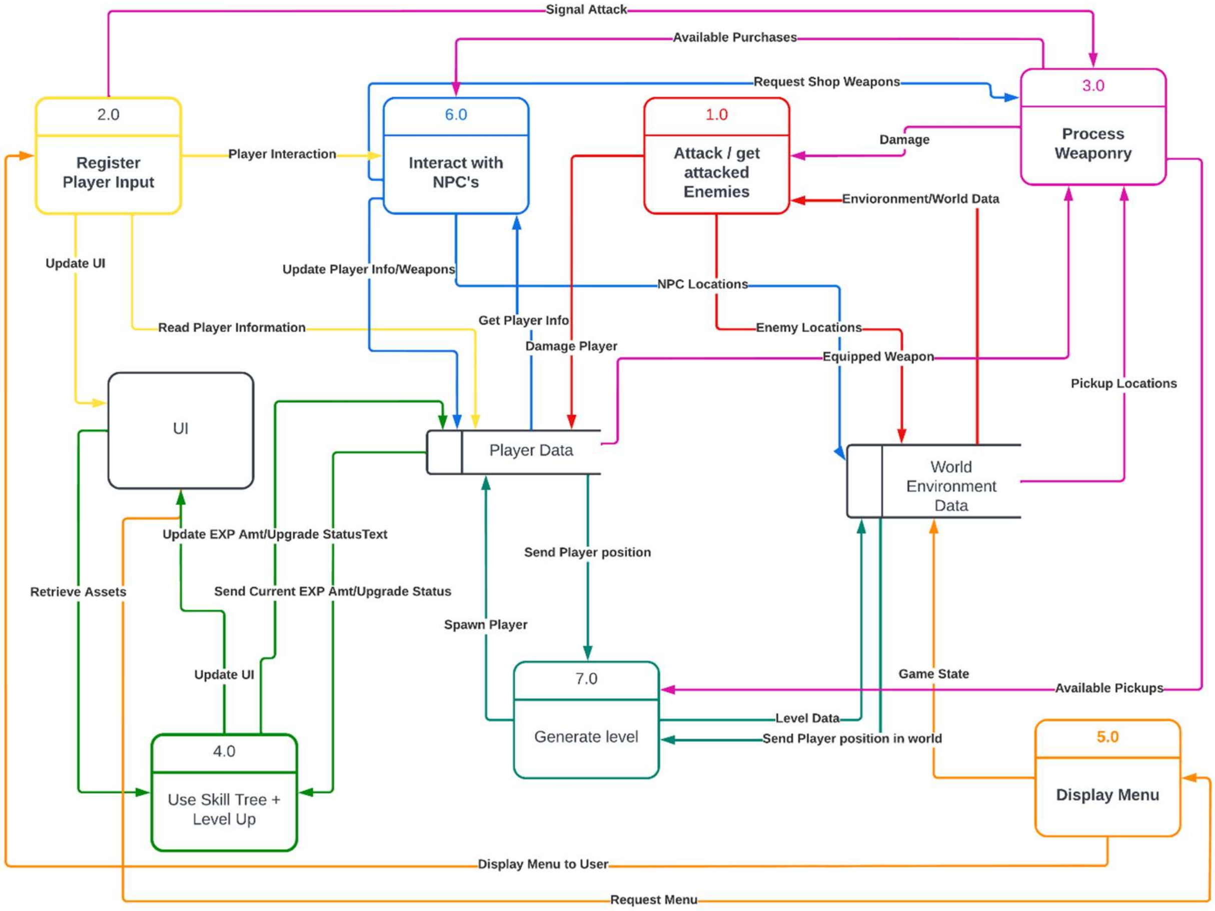
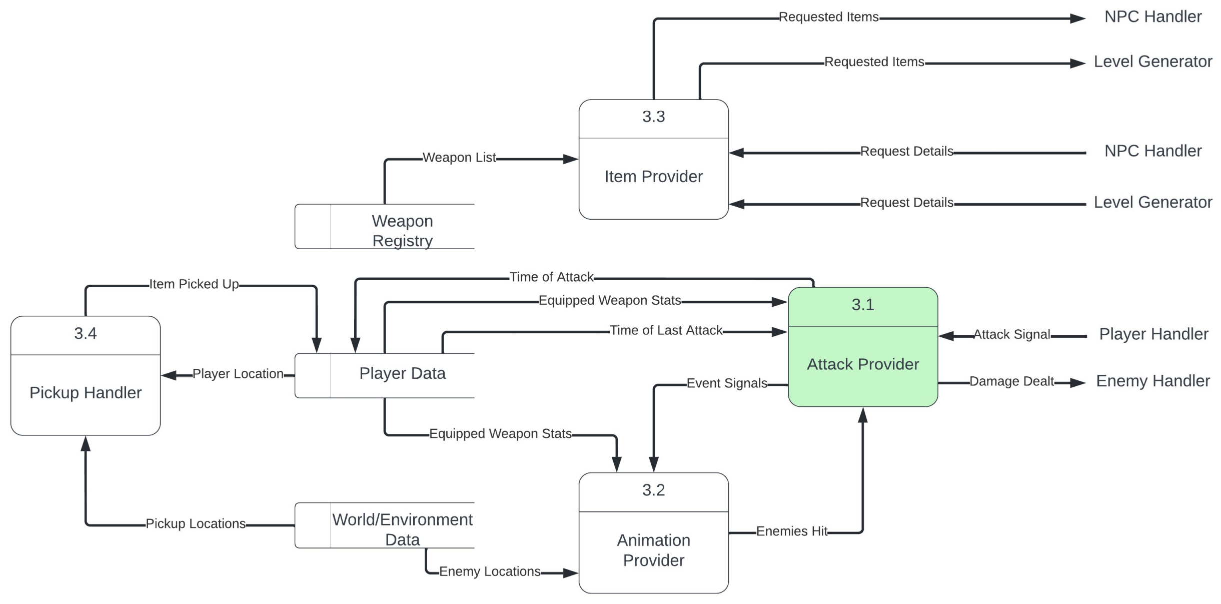


Diagram 3.0 – Process Weaponry



Process Descriptions

Process 3.1 – Attack Provider:

```
ON AttackSignal FROM PlayerHandler
    RETRIEVE EquipStats FROM PlayerData
    RETRIEVE LastAttackTime FROM PlayerData
    IF Time MINUS LastAttackTime GREATER THAN Cooldown IN EquipStats
        SEND Time to PlayerData
        SEND AttackSignal TO AnimationProvider
        RECEIVE EnemiesHit FROM AnimationProvider
        FOR EACH EnemyHit IN EnemiesHit DO
            SEND Damage IN EquipStats TO EnemyHit
        END FOR EACH
    ELSE
        SEND CooldownSignal TO AnimationProvider
    END IF ELSE
```

4. Acceptance Tests _______9

Item Provider

The item provider process will fulfill the Generate Items use case. As part of this, it will receive requests for certain numbers of random items, as well as specific items. It fulfills these requests with weapons pulled from the Weapon Registry, returning the selected weapons to whence the request came. Requests made for the sake of testing this shall be:

- 0 or less random items. This should return nothing, and log an error.
- A number of random items equal to the number contained in the Registry. This should return every weapon contained in the Registry, without duplicates.
- More random items than exist in the Registry. This should return every weapon in the Registry, with enough items duplicated enough times that the requested number is provided.
- Requesting any number of items, or any specific items, when the Registry is empty. This should return nothing, and log an error.
- Requesting a specific item which is in the Registry. This should return that weapon, and nothing more.
- Requesting a specific item which does not exist in the Registry. This should return nothing, and log an error.

Attack Provider

The attack provider will, along with the animation provider, fulfill the Attack with Weapon use case. In doing so, it will check the equipped weapon, make sure the cooldown for using that weapon is over, use the animation provider to find enemies hit, and damage those enemies. Tests to verify functionality shall include:

- Attack when no weapon is equipped. This should do nothing.
- Attack repeatedly, such that the attacks are occurring less frequently than the equipped weapon’s cooldown. This should function as above.
- Attack repeatedly, such that the attacks are occurring with equal frequency to the equipped weapon’s cooldown. This should function as above.
- Attack repeatedly, such that the attacks are occurring more frequently than the equipped weapon’s cooldown. This should use the animation provider to produce a special “weapon on cooldown” animation on some attacks, while others function normally. No two attacks should be fulfilled regularly save that the time between them be greater than the equipped weapon’s cooldown, and no two consecutive attacks should produce a cooldown animation save the time between them be less than the equipped weapon’s cooldown.

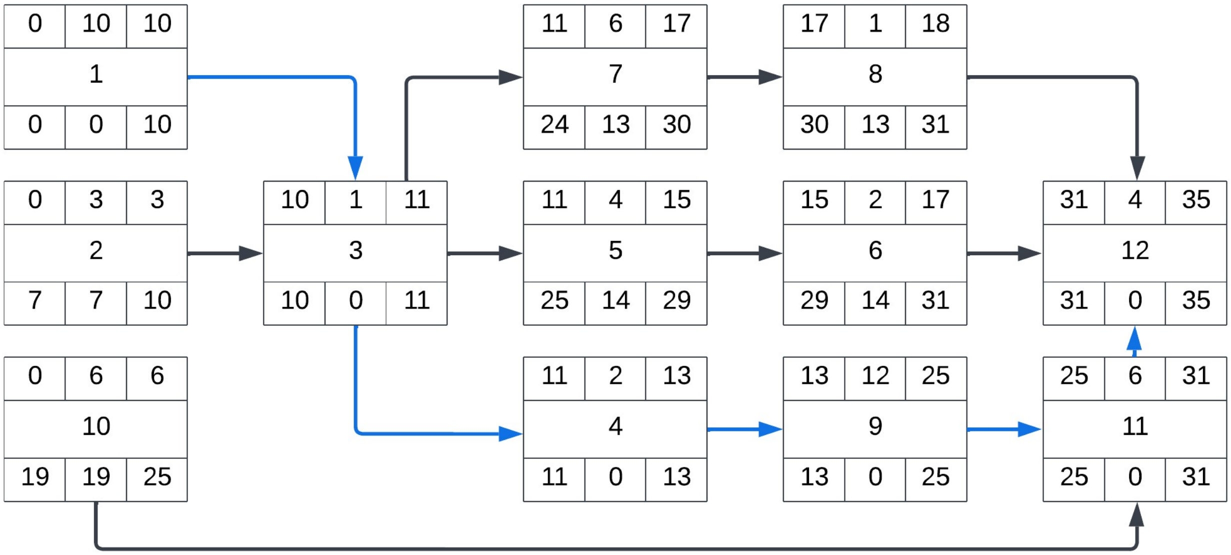
5. Timeline ________/10

Work items

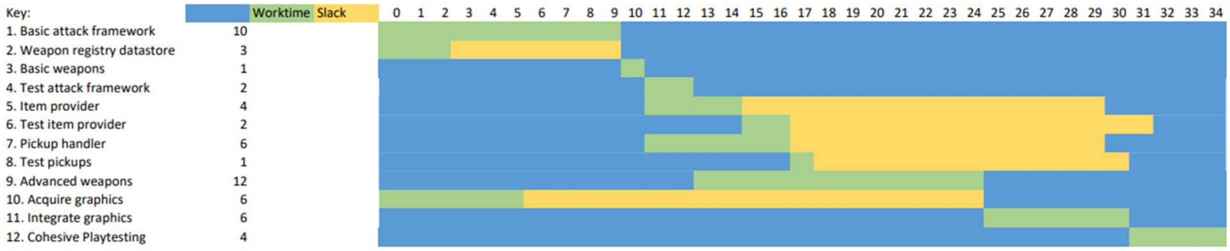
Task	Duration (PHrs)	Predecessor Task(s)

1. Create basic framework classes for weapon attack and animation providers	10	-
2. Create structure for the weapon registry datastore	3	-
3. Create basic sample weapons for testing functionality	1	1, 2
4. Test attack framework	2	3
5. Create item provider	4	3
6. Test item provider	2	5
7. Create in-world pickup handler	6	3
8. Test pickups	1	7
9. Create specific weapons	12	4
10. Acquire graphics	6	-
11. Attach graphics to weapons	6	9, 10
12. Cohesive Playtesting	4	6, 8, 11

Pert diagram



Gantt Timeline



Key:	🟦	Worktime	Slack	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
1. Basic attack framework	10			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
2. Weapon registry datastore	3			🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
3. Basic weapons	1			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
4. Test attack framework	2			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
5. Item provider	4			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟦	🟦	🟦	🟦
6. Test item provider	2			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟦	🟦
7. Pickup handler	6			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟦	🟦	🟦	🟦
8. Test pickups	1			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟦	🟦	🟦
9. Advanced weapons	12			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
10. Acquire graphics	6			🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
11. Integrate graphics	6			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦
12. Cohesive Playtesting	4			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟩