

1. PlayerController

Responsibilities: - Handle movement (8-directional, isometric) via Unity New Input System. - Handle weapon switching and sigil activation inputs. - Forward ability input to active WeaponController. - Manage player states (stunned, shield broken, grappling). - Track global resources (player HP, cooldowns, etc.).

Inputs: - Movement → Vector2 - Attack → Button press (SwordController) - Block → Hold button (ShieldController) - Grapple → Aim + Fire (GrapplingHookController) - Switch Weapon → Button press

Flow: 1. Read input. 2. Update movement (unless movement is restricted). 3. Send ability commands to active weapon controller. 4. Update global state (stun, knockback, cooldowns).

2. WeaponControllers

Each weapon reads Ability ScriptableObjects to execute logic.

A. SwordController

- Core mechanic: Swing → damage + knockback.
- Sigil effects: Projectiles, AoE, status effects, increased damage/knockback.
- Flow:
 - Receive attack input.
 - Determine direction (8-way).
 - Detect collisions with enemies.
 - Apply damage + knockback + sigil effects.
 - Trigger VFX/SFX.
 - Handle cooldown.

B. ShieldController

- Core mechanic: Shield energy absorbs damage.
- Break mechanic: Energy = 0 → disable defense temporarily.
- Regen mechanic: Energy restores after cooldown.
- Sigil effects: Modify energy, regen speed, add block-based effects.
- Flow:
 - On damage → reduce shield energy first.
 - If energy = 0 → break state + optional stun.
 - Handle cooldown before energy regen.
 - Block/Parry input → mitigate damage or trigger counterattack.

C. GrapplingHookController

- Core mechanic: Fire grapple → small damage on attach → pull enemies or objects.
- Restrictions: Player cannot move while firing.
- Sigil effects: Modify pull strength, range, multi-target, extra effects.
- Flow:
 - Receive grapple input + target.
 - Fire hook and detect collision.
 - On attach → apply damage + pull logic.
 - Release → restore movement.

3. Ability ScriptableObject

Purpose: Drive weapon abilities in a data-driven way.

```
using UnityEngine;

public enum AbilityType { Damage, Knockback, ShieldBlock, Grapple, Utility }
public enum AbilityDirection { None, Cardinal, EightWay, AllAround }

[CreateAssetMenu(fileName = "NewAbility", menuName = "Abilities/Ability")]
public class Ability : ScriptableObject
{
    public string abilityName;
    public AbilityType type;

    public float cooldown;
    public float resourceCost;

    public AbilityDirection direction;

    public float damage;
    public float knockbackForce; // sword
    public float shieldEnergy; // shield
    public float grappleForce; // grapple
    public GameObject vfxPrefab;

    public virtual void Activate(GameObject user, Vector2 direction)
    {
        // WeaponController handles logic
    }
}
```

Notes: - Sigils modify these values dynamically. - Base weapons can operate even without equipped sigils.

4. Sigil System

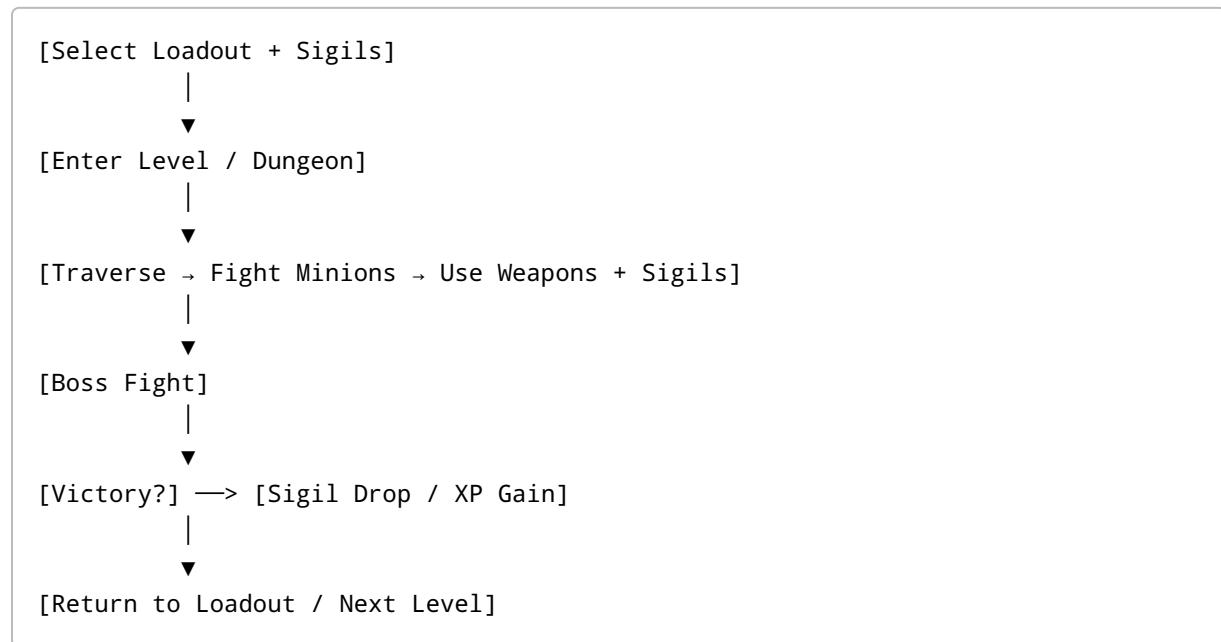
Responsibilities: - Track sigil XP, level, and effects. - Provide loadout selection before levels/dungeons. - Persist progression across story and dungeon modes. - Handle duplicate sigil XP instead of redundant drops.

Flow: 1. Player equips sigils → assign to weapon slots. 2. During combat → WeaponControllers read sigil-modified values from Ability SOs. 3. XP awarded per sigil based on usage or drops. 4. Leveling up improves sigil effects.

5. Level / Dungeon Loop

Mechanics: - Pre-constructed level layouts for story mode. - Dungeon/Mythic mode: scalable minion counts, boss mechanics, timed objectives. - Victory conditions: - Clear required minions. - Defeat boss. - Optionally complete within time. - Rewards: - Sigil drop or bonus XP. - Sigil XP progression persists across modes.

Flowchart:



6. Player Experience Goals

- Sword: Aggressive, skillful melee. Feels impactful even without sigils.
- Shield: Strategic, tension-driven defense. Timing and energy management matter.
- Grappling Hook: Skillful mobility and tactical control. Timing and positioning critical.

- Sigils: Encourage experimentation and progression without replacing base mechanics.
- Story vs Dungeon Mode:
 - Story → structured progression, loadout changes at checkpoints.
 - Dungeon → timed, repeatable, skill-based challenge with scalable difficulty.