ULTIMATE ACTION RPG
STARTER PROJECT

Complete Technical Documentation — Updated & Revised

Unreal Engine 5.3+ | Blueprint-Only | No External Plugins

Support: discord.gg/cq464nBPRE

1. Quick Start Guide

New to this template? Follow these five steps to get up and running in under ten minutes.

1. Open the project in Unreal Engine 5.3 (recommended).
2. Go to Edit > Project Settings > Input and verify all Action and Axis Mappings listed in Section 5 are present. If any are missing, add them manually — missing mappings are the most common cause of controls not working.
3. Open the Maps folder and load one of the example levels from AllLevels or TestingMaps.
4. Press Play. The player should move, aim, fire, and interact out of the box.
5. If you see any errors on startup, check the Common Errors table in Section 3 before anything else.

> ℹ️ **If You Are Converting to UE 5.6 or 5.7**
> Read Section 3 (Common Errors) and Section 6 (Compatibility Notes) before opening the project.
> Engine upgrades require a few extra steps — knowing them in advance saves a lot of time.

1.1 What You Can Customise Immediately

- Change health, shield, and pulse energy values in BP_PlayerCharacter variables
- Swap weapon meshes in the Weapons folder
- Add or remove levels in the Maps folder
- Replace UI widget art in the UI/PNG folder
- Adjust AI detection range and damage values in BP_AI

1.2 Where to Go For Common Tasks

Goal	Go To
Add a new weapon	Section 9 (Combat & Interaction)
Add a new pickup type	Section 14 (Pickup Systems)
Modify AI behaviour	Section 15 (AI Combat System)
Change camera / view modes	Section 10 (Aiming & Camera)
Understand damage flow	Section 11 (Damage System)
Modify sword combat	Section 13 (Sword Attack System)
Understand the folder layout	Section 4 (Folder Structure)

2. Multiplayer — Important Disclaimer

🚫 **This Template Is Single-Player Only**

This project has NOT been designed, tested, or validated for multiplayer.

Specifically:

- No variables are replicated
- No RPCs (Remote Procedure Calls) are implemented
- No network prediction or lag compensation is present
- It has never been run in a multiplayer session

Do not purchase this template expecting ready-made multiplayer functionality.
Adding multiplayer requires a full replication setup — a significant engineering task that is outside the scope of this starter project.

The Blueprint architecture is clean enough that multiplayer can be added in the future — event dispatchers, interface calls, and centralised player state all support eventual replication. However, none of that work is done for you here.

✅ **What This Template Is For**

Single-player games, prototypes, learning projects, and game jams.
It is an excellent foundation for solo experiences and can be extended to multiplayer by an experienced developer, but that extension is not included.

3. Common Errors & Fixes

Check this table first before posting a support question. Most issues have a fast fix.

Error / Symptom	Cause	Fix
Skipped package /Engine/EngineMeshes/Humanoid	Engine content not fully installed	Epic Games Launcher → Verify UE installation → Restart engine
Material error in bottom-right after engine upgrade	Internal material node changed in UE 5.7+	Click error → Open material → Reconnect broken node → Compile → Save
Player does not move or controls do nothing	Input mappings missing from Project Settings	Edit > Project Settings > Input → Add all Action and Axis Mappings from Section 5
Weapon does not fire	Not aiming, Equipped? is false, or out of pulse energy	Aim first (hold right-click), pick up a weapon, wait for pulse to recharge
Duplicate UI elements on screen	Widget references created more than once	Widgets are created in Begin Play — ensure Begin Play fires only once per spawn
Blueprints show errors after engine upgrade	Stale redirectors or intermediate files	Delete Saved / Intermediate / DerivedDataCache → Reopen → Recompile Blueprints
AI does not attack or move	Begin Play events not wired, or detection range is zero	Open BP_AI → Check Begin Play connections → Verify sense range variable is > 0
Sword hits not registering	AttackStart / AttackOver notify timing is off	Open sword anim montage → Adjust AttackStart and AttackOver notify positions
Camera snaps during ADS / scope	MinScope value is too low	In BP_PlayerCharacter Begin Play → Increase the MinScope value
Mac: shaders recompile every launch	Shader cache not preserved; Mac not officially tested	Allow shaders to recompile fully once → Save all materials → Reopen project

If none of the above resolves your issue, visit the Discord server: discord.gg/cq464nBPRE

4. Folder Structure

All content is organised for scalability and clarity. Every system is separated by responsibility.

```
Content/
├── UltimateActionRPG/          ← main project content
│   ├── Arts/
│   ├── ALSV/
│   │   ├── PlayerAnimations
│   │   ├── Data
│   │   └── Environment         ← from ALSV package
│   ├── FX/
│   │   ├── WeaponFX
│   │   └── SwordSwingFX
│   ├── Mesh/
│   │   └── Weapons/
│   │       ├── WeaponMeshes
│   │       └── AdditionalMeshes
│   └── SFX/
│       ├── Cue/
│       └── Wave/
├── Maps/
│   ├── AllLevels
│   └── TestingMaps
├── VisualAssets/
│   ├── PhysicalMaterial/ → MetalBody_PhysMat
│   ├── Materials/
│   ├── Textures/
│   └── CameraShake/
├── UI/
│   ├── Widgets/
│   └── PNG/
└── Blueprint/
    ├── AI/ → BP_AI + Weapon/AIProjectiles
    ├── Menu/ → HomeMenu + GameInstance
    └── Player/
        ├── BP_PlayerCharacter
        ├── Weapons/ → Enums, Projectiles, Weapon_BP
        ├── Pickable/ → BP_Pickable, HealthPickup, HealthPickupBase
        └── GameplayFramework (Logic / Interfaces / Libraries)
```

5. Input System

This project uses the Legacy Input Mapping System — not Enhanced Input. All mappings must be configured in Project Settings.

> ℹ️ **How to Access**
> Edit > Project Settings > Input
> If any mapping below is missing, add it manually with the exact name shown.
> A missing or misnamed mapping will silently prevent that input from working.

5.1 Action Mappings

Action Name	Purpose
JumpAction	Jump
StanceAction	Change stance or mode
SprintAction	Sprint
WalkAction	Walk toggle
AimAction	Aim weapon
CycleOverlayUp	Cycle ALS overlay upward
CycleOverlayDown	Cycle ALS overlay downward
Fire	Attack / Shoot
Interact	Pickup and world interaction
PauseGame	Open pause menu
Heal	Use heal crystal
Task	Custom action slot
SwitchCamera	Toggle first-person / third-person

5.2 Axis Mappings

Axis Name	Purpose
MoveForward	W / S movement
MoveRight	A / D movement
LookUp	Camera up / down
LookLeft	Camera left / right

Axis Name	Purpose
Turn	Character turning

6. Requirements & Compatibility

6.1 Recommended Engine Version

Recommended: Unreal Engine 5.3. The template was originally built and validated in UE 5.3. Conversion to newer versions (5.6 / 5.7+) is possible but requires the additional steps described below.

6.2 Engine Content Dependency

The legacy ALS locomotion system depends on:

```
Engine/EngineMeshes/Humanoid
```

> ⚠️ **UE 5.6+ Change**
> Starter Content is no longer bundled in UE 5.6+, which may mean this asset is not installed.
> If you see: Skipped package /Engine/EngineMeshes/Humanoid
> This is NOT a broken template — it is a missing engine installation dependency.
> Fix: Epic Games Launcher → Verify your UE installation → Restart engine.

6.3 UE 5.7 Material Conversion

When converting from 5.3 to 5.7, a material error may appear in the bottom-right corner. This is caused by internal node changes in the engine — not a broken template.

Fix: Click the error > open the affected material > reconnect or replace the broken node > Compile > Save. The error disappears permanently after saving.

6.4 General Upgrade Maintenance

- Fix up Redirectors (right-click Content Browser root)
- Delete the Saved, Intermediate, and DerivedDataCache folders
- Reopen the project and recompile all Blueprints
- Re-save affected materials

6.5 Platform Support

Platform	Status	Notes
Windows	✅ Tested	Primary supported platform
Mac	⚠️ Not tested	May need shader recompile and material reconnection

Platform	Status	Notes
Linux	❌ Not tested	Not validated

6.6 Future Update Plan

- Remove legacy EngineMeshes dependency
- Improve UE 5.6+ / 5.7+ migration stability
- Expand Mac testing and compatibility
- Reduce manual setup steps

7. Core Systems Overview

The template ships with four fully blueprint-based gameplay systems. Each is modular and independently replaceable.

7.1 Player System

- Smooth locomotion (ALSV-based)
- Sprint with stamina regeneration
- Health and Shield (Defence) system — shield absorbs damage first
- Multi-directional Dodge / Roll (Forward, Back, Left, Right)
- First-person / third-person camera switching
- Energy-based weapon system (Pulse Charge — replaces traditional ammo)
- Sword combat with animation-driven combos
- Weapon cycling via mouse scroll
- Physical material hit detection (metal, wood, concrete, glass, body)
- Partial Save Game implementation

7.2 Combat System

- Single-shot and auto-fire weapon modes
- Pulse energy recharge (stops while shooting, resumes on release)
- Wall trace fire blocking — prevents shooting through geometry
- Melee fallback when pulse energy is depleted
- Centralised projectile spawning with per-weapon parameters
- Shield-first damage calculation
- Per-weapon spread and recoil

7.3 AI System

- Blueprint-only — no Behavior Trees required
- Player detection and sensing
- Smooth rotation to face the player
- Direct attack logic with cooldown
- Missile firing system
- Overheat system — creates attack windows for the player
- Dynamic damage scaling on overheat / recovery
- Clean death handling — no logic runs after death

7.4 UI System

All widgets are created once at Begin Play, stored as references, and added or removed from the viewport as needed.

- Full menu framework: Home, Main, Options, Controls, About, Store, Credits
- Pause Menu
- Dynamic Crosshair (expands with movement speed)
- Sniper Scope overlay
- Weapon Pickup prompt
- ALS HUD
- Damage Indicator (randomised timing, auto-removed)

8. BP_PlayerCharacter — Core Event Systems

This section covers how the player is initialised, how systems update each frame, and how all major gameplay inputs are handled.

8.1 Event Begin Play

EventBeginPlay — the Sequence node runs each initialisation step independently

A Sequence node runs each initialisation step independently, keeping startup logic readable and debuggable.

1. Parent Begin Play + On Begin Play

On Begin Play function inside BP_Base_CharacterBP — manages movement, anim instance, rotation, and stance

Calls the parent character class first, which handles core movement, rotation, and animation initialisation. This is inherited from the Base ALSV Character.

> ⚠️ **Do Not Remove**
> Removing this call without replacing the parent system will break movement and animations.

2. Create Widget References

Create Widget function — all UI widgets spawned once via Sequence, stored as named references

Spawns all UI widgets once and stores them as variables. Widgets are added or removed from the viewport via these references — never recreated. This prevents duplicate UI elements and makes dynamic updates reliable.

3–5. Weapon References, Scope Zoom & FPS Limit

The remaining Begin Play steps cache the weapon pickup actor reference, cache the base weapon actor and set the minimum scope FOV, and apply a 60 FPS frame rate cap via Game User Settings.

8.2 Event Tick

Event Tick — five systems isolated by Sequence node: pulse energy, shield, crosshair, attack reset, wall trace

Restore Pulse Energy

RestorePulseEnergy — recharges weapon energy over time, pauses while shooting

Recharges weapon energy over time. Pauses while IsShooting? is true. Uses a small initial delay to prevent instant recharge after firing. Loops until MaxPulseCharge is reached.

Restore Shield

RestoreShield — slowly restores Defence, pauses while GettingDamage is true

Slowly restores the Defence (shield) value. Pauses while GettingDamage is true. Loops until MaxDefence is reached.

Wall Trace, Crosshair Spread & CanAttack Reset

Three additional Tick steps handle the Wall Trace (detects geometry blocking fire), Dynamic Crosshair Spread (maps velocity to spread range), and CanAttack reset (delay-based cooldown to prevent attack spamming).

9. Combat, Interaction & Weapon Systems

Design rule: One Fire Event. One Interact Event. Everything else reacts.

9.1 Fire Event — Safety Checks

FireEvent (part 1) — safety checks gate: CanAttack, Death?, Sword mode, NearWall?, PulseCharge, IsOutOfPower?, Equipped?, Aiming?

Check	Blocks Firing If...
CanAttack	Attack cooldown has not reset yet
Death?	Player is dead
Equipped?	No weapon is currently held
Aiming?	Player is not currently aiming
IsOutOfPower?	Pulse energy is at zero
NearWall?	Wall trace detected geometry directly ahead

9.2 Fire Event — Weapon Output

FireEvent (part 2) — Gate node controls auto-fire, Retriggerable Delay sets fire rate, SpawnBullets handles all output

After all checks pass, a Gate node controls continuous fire via Retriggerable Delay. SpawnBullets handles projectile spawning, muzzle flash, sound, recoil, spread, and material hit effects. Each weapon feeds only its own values into this shared function.

9.3 Interact System

Event Dispatcher: Intract — the single dispatcher that all interactable objects bind to

IntrectAction — InputAction Interact calls the dispatcher only; no checks, no logic

The Interact input calls the Interact Event Dispatcher only — no checks, no logic, no hard references. Pickups, NPCs, and any interactable actor bind to this dispatcher and react independently.

9.4 Weapon Pickup Variables

Pickup boolean flags — one per weapon type, controls equip logic and prevents duplicate pickups

- PistolPickup
- RiflePickup
- ShotGunPickup
- SniperPickup
- FlameThrowerPickup

✅ **Adding a New Weapon**

1. Add a boolean to BP_PlayerCharacter (e.g. RocketLauncherPickup)
2. Add a branch in the Weapon Decision Layer
3. Create a pickup actor

No interaction logic needs to be rewritten.

10. Aiming, ADS & Camera Systems

10.1 Aiming System

Aiming — checks NearWall?, Aiming?, Equipped?, then sets rotation mode and handles sniper / non-sniper UI swap

Before aiming activates, four conditions must all be true: not near a wall, weapon equipped, not in sword-only mode, CanAttack is true. When all pass, Aiming is set to true, rotation mode updates via BPI Set Rotation Mode, and sniper-specific scope logic applies.

10.2 ADS Zoom

ADS Zoom custom event — reads camera FOV, adjusts incrementally, clamps between MinScope and MaxScope

Reads the current camera FOV, adjusts it incrementally by the Forward input value (+/- 10.0), then clamps the result between MinScope and MaxScope. Gives smooth gradual zoom instead of instant snap.

10.3 Mouse Scroll — Dual Behaviour

MouseScroll — CycleOverlayUp / Down check Aiming? and route to either ADS Zoom or Roll Weapon

Context	Scroll Result
Player IS aiming with scope	Calls ADS Zoom — forward = zoom in, back = zoom out
Player is NOT aiming	Calls Roll Weapon — cycles through available weapons

10.4 FPS / TPS Camera Switch

SwitchCamera — FlipFlop node toggles First Person / Third Person via BPI Set View Mode interface call

A FlipFlop node toggles between First Person and Third Person by calling the BPI Set View Mode interface function. Using an interface keeps camera logic decoupled from the character, allowing reuse across multiple character types.

10.5 Sword Attack Camera (Optional)

CamSwitch — on sword hit, FlipFlop cycles camera angle via Set View Target with Blend (0.5s); TimeTrigger drives slow-motion via Set Global Time Dilation

When enabled, each sword hit triggers a camera angle switch via FlipFlop and a slow-motion effect via timeline-driven time dilation. Camera blends smoothly using Set View Target with Blend. This entire system can be disconnected without affecting any other gameplay system.

11. Damage System

11.1 Damage & Feedback

GetDamage — AnyDamage event checks Dodging?, runs TakeDamage, triggers camera shake, and spawns a Damage Indicator widget via Do Once

The AnyDamage event only applies damage if Dodging? is false and the damage value is valid. When damage lands: Defence is reduced first; if Defence reaches zero, Health takes the remainder. The GettingDamage flag is set briefly to pause shield regeneration.

Feedback: a Damage Indicator widget is created and auto-removed via delay. Camera shake plays with its epicentre at the player's world location.

11.2 Character Stats Reference

Three variable groups define the entire player state. These are what every system reads from and writes to.

Character Stats

Character Stats — Defence, MaxDefence, Health, MaxHealth, CamBlend, HealCrystalCount, ExpCoin, Death?, GettingDamage, Dodging?

Weapon Stats (Booleans)

Weapon Stats — PrimaryWeapon, SecondaryWeapon, Equipped?, CanAttack?, SwordAttack?, IsAttacking?, ContinousFire?, IsShooting?, IsOutOfPower?, IsAutoFire, Scope, SwordHit?, Hit?, SpreadBullets?, NearWall?, MaxDamage, Damage, MaxPulseCharge

Weapon Stats (Floats & Enum)

Weapon Stats (cont.) — PulseCharge, FireRate, SpreadRange, DecalLifeTime, DecalSize, MinScope, MaxScope, ShotAmount, E_WeaponType

12. Foundational Systems

12.1 Weapon Socket Names

Weapon Socket Name variables — SwordSocketName, PrimaryWeaponSocketName, SecondaryWeaponSocketName

Weapons are attached to the character skeleton via named sockets, not hard-wired mesh references. To move a weapon to a different position (back, hip, shoulder), change only the socket name variable. No Blueprint rewiring required.

12.2 Weapon Switch / Roll

Weapon Roll / Switch — checks pickup boolean flags, skips unavailable weapons, updates equipped state, meshes, and weapon type

The Roll Weapon function cycles through available weapons by checking pickup boolean flags and skipping any that are false. It updates equipped state, weapon meshes, and weapon type variables, then syncs with the firing and aiming systems. Mouse scroll, UI buttons, and keyboard input all call the same function.

12.3 Character Components

BpPlayerCharacter component hierarchy — CapsuleComponent, Mesh, weapon socket components, VisualMeshes, HeldObjectRoot, AttackArms with child cameras

The character component hierarchy is intentionally structured: weapon holding components (MeleeWeaponSocket, PrimaryWeaponSocket, SecondaryWeaponSocket) each match a skeleton socket. Multiple AttackArm components under the Attacks hierarchy each carry a scene component, child actor, and camera attachment — used by the optional sword CamSwitch system.

12.4 Core Functions

Core function list — ConstructionScript, Create Widget, SwitchWeapon, SpawnBullets, WeaponType, TakeDamage, StopAnim, Wall Trace, SwitchToSword, Death, SheathWeapon

Function	Responsibility
SwitchWeapon	Cycles and equips available weapons
SpawnBullets	Centralised projectile and hitscan output
WeaponType	Resolves current weapon behaviour category
TakeDamage	Processes damage against shield then health
WallTrace	Detects walls blocking fire
SwitchToSword	Transitions to melee mode
SheathWeapon	Hides weapon meshes when holstered
Death	Handles player death state and cleanup
StopAnim	Safely interrupts any active animation
Create Widget	Spawns and stores all UI widget references

13. Sword Attack System

Melee combat is isolated from gunplay so the two systems never conflict. All sword logic is driven by Animation Notifies — the Blueprint does not guess at timing.

SwordAttack — input disabled, random montage selected from SwordAttackAnim array, AttackStart / AttackOver notifies control the damage window

13.1 SwordAttack Custom Event

6. Player input is temporarily disabled
7. A random animation is selected from the SwordAttackAnim array
8. The animation montage plays
9. The system waits for the montage to finish
10. Input is re-enabled

13.2 Animation Notify Events

AttackStart

Fires when the animation reaches the hit window. Sets SwordAttack? to true. Damage can only register while this flag is true — no ghost hits.

AttackOver

Fires when the animation exits the hit window. Sets SwordAttack? to false, resets SwordHit?, and re-enables input.

13.3 Dodge Variables

Dodge variable group — DodgeF?, DodgeB?, DodgeR?, DodgeL? (Boolean) and DodgeAnim (Anim Montage)

13B. Player Action Systems

Dodge System

Dodge — Shift+SpaceBar checks velocity > 0, then W/S/A/D keys set DodgeF?, DodgeB?, DodgeL?, DodgeR? flags

Input: Shift + Movement Keys. The system checks velocity is greater than zero, then directional key state sets the correct dodge boolean. The Dodge montage plays the corresponding directional animation. Works in both FPS and TPS.

Heal System

Heal — requires HealCrystalCount > 0 and Health < MaxHealth; decrements crystal count, adds MaxHealth/4.0, clamps result

Requirements: at least 1 Heal Crystal and Health below maximum. When used: crystal count decrements, Health increases by MaxHealth / 4.0, result is clamped, and a heal sound plays.

Pause System

Pause — PauseGame input calls Set Game Paused, then adds the Pause Menu widget to the viewport

PauseGame input calls Set Game Paused then adds the Pause Menu widget reference to the viewport. Centralised logic — no duplicate pause menus possible.

14. Pickup Systems

14.1 Base Weapon Pickup

On Component Begin Overlap validates the overlapping actor is BP_PlayerCharacter, binds to the Interact dispatcher, and shows the pickup UI. On End Overlap unbinds and hides the UI. Does not decide which weapon is given — that lives in the Weapon Decision Layer.

14.2 Health, Crystal & Pulse Pickups

This blueprint is generic and self-contained. It validates before consuming, clamps all values, plays feedback, and destroys itself.

Pickup Type	Condition	Effect
Health	Health < MaxHealth	Adds HP, clamped to MaxHealth, plays Heal SFX
Heal Crystal	Always consumable	Increments HealCrystalCount
Pulse Energy	PulseCharge < MaxPulse	Adds energy, clamped to MaxPulseCharge, plays SFX

> ✅ **Extend to New Pickup Types**
> Add a new branch, a new variable on BP_PlayerCharacter, and feedback (sound / VFX).
> No redesign of collision or interaction logic is needed.

15. AI Combat System

The AI is designed to fight, not just exist. It senses the player, rotates to face them, attacks with weapons and missiles, overheats, recovers, and dies cleanly. Every behaviour is a custom event — replacing or rebalancing any one of them does not affect the others.

15.1 System Summary

Custom Event	Behaviour
Begin Play	Initialises all AI state; kicks off custom events cleanly
Sense Player	Detects player within range and valid combat conditions
Face Player	Smoothly rotates AI to face player each frame
Attack	Applies damage; respects cooldown and overheat state
Missile Firing	Fires projectiles after conditions met; each shot adds heat
OverHeat	Stops firing; triggers recovery timer; creates player attack window
Increase/Dec Damage	Scales damage output on overheat state or difficulty triggers
AnyDamage	Reduces AI health; centralised across all damage types
Death	Stops all logic; destroys actor cleanly when Health <= 0

16. Architecture Philosophy

Every system in this template follows the same core set of rules.

Principle	Practical Meaning
One Fire Event	All firing logic lives in a single custom event
One Interact Event	Interaction is a dispatcher — no hard references
Player owns weapon state	Weapons feed values; the player holds all state flags
Data-driven weapons	Each weapon provides values to shared logic — no duplication
Reference-based UI	All widgets created once and reused via stored variables
Independent regen	Energy and shield regenerate on separate, isolated loops
State-driven logic	Systems react to boolean flags, not random triggers
Optional camera FX	Cinematic systems can be disconnected without breaking combat

16.1 Safe Customisation

- Replace guns with magic spells
- Replace pulse energy with stamina
- Remove the sword system entirely
- Add new weapon types
- Replace the entire UI
- Remove FPS or TPS mode
- Disable cinematic sword camera
- Add new pickup types
- Adjust all stat values (health, shield, energy, fire rate)

Nothing is hard-locked. Nothing fights you.

17. Support

Discord Community

For bug reports, questions, and update announcements:
discord.gg/cq464nBPRE

Before posting, please check Section 3 (Common Errors & Fixes) — most issues have a documented solution.

All Marketplace assets included in this project are fully integrated and cannot be extracted separately. All logic and systems are editable for your own commercial games.