# **Technical Document: Skill System & HUD**

## **1. Overview**

This document provides a technical breakdown of the Skill System and HUD implementation. It describes the systems created, the workflow used during development, and the reasoning behind design decisions. The goal of the system is to provide a modular, extensible, and efficient way to manage skills, cooldowns, and user interface feedback for the player.

## **2. Created Systems**

### **2.1 Skill System Component**

* **Type**: Actor Component (attached to player character).
* **Core Variable**:  
   Skills → Map with:
  + **Key**: Enum (ESkillNames) → defines available skills (e.g., Dash, Force Push, Invisibility).
  + **Value**: Struct (ST\_SkillDetails) with:
    - Is\_Active (bool) → whether the skill is currently active.
    - Cooldown\_Duration(float) → amount of cooldown time set for skill.
    - Cooldown\_RemainingTime(float) → remaining cooldown time.
* **Core Functions**:
  + **ActivateSkill(ESkillNames)** → checks availability, toggles IsActive, sets cooldown.
  + **Cooldown Management** → using multiple timer events to manage cooldown progress.
* **Extensibility**: New skills can be added by extending the enum and updating data in the map, without rewriting core logic. While the core system is reusable, each new skill does require the addition of a dedicated timer event in the Blueprint to manage its cooldown. However, this step is mostly procedural, essentially copying an existing timer setup and renaming it for the new skill. This keeps the workflow consistent and avoids introducing new or unique logic for each skill.

### **2.2 Cooldown Material**

* **Purpose**: Visually represent cooldown on skill icons.
* **Implementation**:
  + Custom material with circular + radial mask.
  + Uses a scalar parameter (Cooldown\_Parameter) to animate wipe progress.
* **Integration**:
  + Dynamic Material Instance updated from the Skill System each tick, reflecting current cooldown progress.

### **2.3 HUD / UI System**

* **Widgets**:  
  + W\_SkillIcon: Displays:
    - Background skill image (Skill\_Texture variable, instance-editable).
    - Cooldown overlay (cooldown material applied on top).
  + W\_PlayerHUD: Contains multiple W\_SkillIcon widgets, each representing one skill.
* **Functionality**:
  + Each W\_SkillIcon can be configured individually (different skill textures, cooldown bindings).
  + Updates dynamically when cooldown state changes.

## **3. Workflow / Process**

1. **System Design**
   * Started by defining a modular approach: using enums + structs + maps to represent skills in a scalable way.
   * However, upon reflection, I realized that using Data Tables would have been a more efficient and flexible choice. Data Tables would make it easier to connect the skill system with other systems (such as progression, balancing, or external configuration) without needing to hardcode or manually expand maps inside Blueprints.
2. **Skill Activation & Cooldown**
   * First tried using Delay → discarded due to inability to handle multiple parallel cooldowns.
   * Adopted **Timer Handles**, enabling independent cooldown tracking per skill.
3. **Material Creation**
   * Iterated from a square mask → radial wipe.
   * Added circular mask for clean UI.
   * Added alpha transparency to avoid background color clutter.
4. **HUD Implementation**
   * Created reusable W\_SkillIcon widget to avoid duplicating logic for each skill.
   * Used instance-editable variables (Skill\_Texture) to allow flexibility.
   * Set Brush dynamically in Construct to ensure textures update properly

#### **Use of Interfaces for Efficiency**

* + Leveraged **interfaces** instead of casting to streamline interactions between components.
  + This reduced reliance on large maps while improving both **efficiency** and **code readability**.

## **4. Reasoning Behind Decisions**

* **Actor Component for Skills**:  
   Keeps skill logic independent of the player class → reusable across different characters or AI.
* **Enum + Struct Map**:
  + Enum ensures clarity of available skills.
  + Map allows O(1) lookup by skill name.
  + Struct centralizes skill state (active + cooldown).
* **Cooldown Handling Notes:**Originally considered using **Timer Handles** instead of **Delays** to support multiple simultaneous cooldowns. Eventually switched to using **separate timer events** per skill, which felt simpler and more manageable. Also thought about spawning **dedicated timer manager actors**, but dropped that idea due to overhead. For a larger-scale skill/ability system, relying solely on multiple timer handles/events probably isn’t scalable. A more robust cooldown management solution will be needed later.
* **Dynamic Material for Cooldown**:  
   More flexible than static textures. Enables real-time updates, reusable across multiple icons.
* **Recurring Logic as Functions:**To maintain clarity and reduce repetition, I encapsulated commonly used logic into reusable functions. This approach kept the blueprint/codebase organized, avoided clutter, and made adjustments easier when tweaking or expanding functionality.

## **5. Future Improvements**

* Add skill descriptions & hotkey bindings inside W\_SkillIcon.
* Use DataTables to drive skill metadata (instead of hardcoding values in Blueprint).
* Add animation feedback (pulse / glow) when skill becomes ready again.  
  Replace temporary **PrintString** debugging with proper **visual indicators** (e.g., greying out skill icon or overlaying a cooldown clock).
* Refactor timer logic into a **centralized cooldown manager** (instead of multiple scattered timer events).
* Add **sound effects** or subtle UI clicks when activating skills or when cooldown finishes.
* Optimize Blueprint logic by moving parts of cooldown handling to **C++ for performance** (if the project gets heavier).

## **6. References Used**

During development, I studied existing tutorials and breakdowns to understand different implementation approaches for abilities in Unreal Engine. These resources helped me shape my own system while extending functionality beyond what was covered:

1. **Force Ability** –<https://youtu.be/CtDoiym_pEg?si=0J1bl1ZwD3rPWRT_>
2. **Dash Ability** – <https://youtu.be/TkSiasE0p_I?si=O_0iSw_mTd7DSSwZ>
3. **Invisibility Ability** – <https://youtu.be/THZwtxhKovE?si=6B85v6u_XKOX2scI>
4. **Cooldown Material Effect** –<https://youtu.be/4bWkYzEJ-Cc?si=4FQkQQhAh1Z4ZQ2o>

Note: There may be a few additional references that I missed, but the above are the major ones I relied on.