First, I got some skateboarding animations from Mixamo and created a new Animation Blueprint using these animations (I used Unreal's default Skeletal Mesh and model).

I added a 3D skateboard model and attached it to the character's blueprint, linking it to the foot bone.

I used Unreal's ThirdPersonCharacter, which already includes movement, jumping, and camera logic, to speed up the process.

I modified the code and added a Sprint (Speed Up) function, as well as an input binding (Shift) using Unreal's Input Actions system. The Sprint function takes the Max Walk Speed value set in Character Movement and multiplies it by 1.5, making it easier to configure speed while keeping a consistent ratio between walk speed and sprint speed, regardless of the base value.

Following the same logic, I also added a Brake (Slow Down) function to the character's code and created an input binding (Ctrl). In this case, I decided to keep the slow-down acceleration fixed at 100.

Both Sprint and Brake modify the Max Walk Speed value when pressed and revert to the original value when released.

I temporarily used Unreal's default jump animation.

Most of the time was spent working on the points system, where I created new C++ classes called PointCollector, MainHUD, and used Unreal's default GameMode class (it had no relevant code, it was just to speed up the process).

PointCollector is an Actor where I created a box collision with Begin/End Overlap functions. When the player passes through it, they earn points. The points variable is exposed in the editor, allowing easy customization of the points each collider grants to the player (default value: 1). When a collision occurs, the score value is sent to GameMode, which is responsible for creating and adding the HUD widget to the screen (MainHUD class).
The GameMode adds the received value to the current score and sends the updated value to the Widget, which updates the TextBlock.

I worked on level design and added the colliders.

Performed code cleanup.

Added informational texts to the HUD.

Made some modifications to the level.

In the Animation Blueprint, I retrieve the PlayerCharacter using TryGetPawnOwner and access CharacterMovement. This way, I can use:

- VectorLength to get the character's speed (Float variable).

- IsFalling to check if the character is on the ground (Boolean variable).
These two variables are used in the BlendSpace for movement animations and in the State Machine for transitioning the jump animation.

Since the logic was simple, I initially implemented it in Blueprints for speed, but later I converted everything to C++.

I updated the jump animation, replacing Unreal’s default one, and made some improvements to other movement animations.

Finally, I cleaned up and organized the assets.

Around 4 hours were spent on the movement system, 8 hours on the scoring system and another 4 hours on other processes, a total of 16 hours spent from the beginning to delivery.