**Assessment Report – Skateboarding Simulator Game**

**System Overview:**
The player system was implemented to provide smooth movement, including walking, jumping, and speed adjustments, with corresponding animation montages. A “speed-up” mechanic was integrated so that the player accelerates and the animation playback rate adjusts according to speed levels. Collectable actors were designed with sphere collisions that increment the player’s points only once per interaction, without destroying the actor. The system includes a modular score UI that updates incrementally with each collected point and displays the current player speed, categorizing it as “Stopped,” “Slow,” “Fast,” or “Very Fast” with distinct colors. The UI was also optimized to prevent unnecessary ticks and maintain performance.

**Thought Process During Implementation:**
I approached the project modularly. First, I focused on the player movement mechanics and jump/speed-up logic, ensuring they felt responsive and integrated with animation montages. Then, I implemented the collectable actor with controlled overlap detection to avoid repeated triggers. Next, I refactored the player’s point increment function for cleaner encapsulation and made the collision handling robust. Finally, I built the UI system, optimizing incremental updates for points and speed state, assigning colors dynamically and preparing it for future enhancements such as messages or effects.

**Personal Assessment:**
I believe my performance was methodical and solution-oriented. I successfully translated gameplay requirements into modular C++ classes, leveraging Unreal Engine best practices. While some challenges arose in managing UI updates and dynamic animation rates, I addressed them through clear logic separation and data-driven design. Overall, the project demonstrates functional and visually responsive mechanics and UI that are easy to extend or modify.

**Time Investment per Task:**

- Player movement, jump, speed-up, and animation montages: 4 hours
- Collectable actor and overlap logic: 2 hours
- Player point increment function and collision handling: 1 hour
- Score UI with incremental effect: 3 hours
- Testing, debugging, and adjustments: 2 hours

**Total Time Invested:** 12 hours