Tomas Racciatti

I implemented a slot-based inventory system designed around Scriptable Objects to ensure flexibility, clean data separation, and designer-friendly workflows. Each item is defined as a Scriptable Object containing its properties, allowing the inventory to reference lightweight identifiers instead of full objects. The Inventory model stores an array of slot states, while the view and UI components react to changes through simple event calls. This separation ensures the system made it easy to extend to new item types, stack logic and equipment behavior without rewriting its core. Saving is handled by serializing the slot contents using item IDs, which allowed the load process to reconstruct the inventory reliably.

During the interview, my thought process was focused on demonstrating architectural discipline rather than building a feature-heavy prototype. With only 48 hours, I prioritized establishing a coherent structure using principles I rely on in production work: clear responsibilities, predictable data flow, and minimal coupling. My goal was to show that I could design a foundation with long-term scalability while still delivering the requested functionality in the short term. Whenever trade-offs appeared, I favored maintainability and clarity. For example, I simplified my usual item-pooling approach to avoid unnecessary complexity given the scope of the task.

In assessing my own performance, I'm satisfied with the stability and cleanliness of the implementation. The foundation is solid, easy to build upon, and demonstrably functional. That said, I believe I could have added more gameplay elements, such as basic combat interactions or consumables, if I had one additional day. My time investment leaned heavily toward establishing a robust architecture, which limited how much surface-level gameplay I could include. Still, I chose to respect the deadline and deliver a complete, reliable system rather than request an extension for optional features.