My objective throughout the test was to design systems with low coupling and high reusability, enabling the easy addition of new features without affecting existing behavior. The movement and attack systems were built using a component-based structure, allowing independent development and refinement of these features while keeping them modular and scalable.

The interaction system follows a weight-based prioritization, where interactables are dynamically detected using overlap checks and sorted based on their priority type — such as picking up items before interacting with NPCs. This decoupled approach supports the addition of new interactable types without modifying the core logic, aligning with the Open/Closed Principle.

The inventory system was developed using a slot-based UI architecture. It supports adding, removing, using, and moving items between slots. The drag-and-drop mechanic was built using Unity's Event System interfaces and handles cases like swapping, stacking, and contextual usage. The logic for item usage was delegated to a separate handler class through an event-driven architecture, allowing for clear responsibility separation and future extensibility (e.g., differentiating between consumables and equipment). Items are only removed from the inventory when successfully used.

A tooltip system was also implemented using the dependency injection concept and reusable trigger components, allowing easy integration across UI elements. Additionally, the inventory state is saved and loaded using a BinaryFormatter-based system and serialized to disk, comparing loaded item data with existing ScriptableObjects at runtime to accurately restore state.

During the test, my focus remained on writing clean, maintainable code that adheres to SOLID principles. As time allowed, I continuously refactored components as new features were introduced to ensure each system could evolve without breaking others. I believe my performance reflects a solid understanding of game architecture and a strong problem-solving mindset, though I acknowledge opportunities to further optimize, input management, game states and reduce manual component linking.