# Technical Test Summary

## Gameplay & Character Logic

- Implemented a basic movement system using Unity’s Input System, combined with a simple state-driven animation setup. The animation controller is tied to a PlayerState enum, and an observer system notifies other systems whenever the player state changes. NPC interaction was also added via a coroutine-based dialog system.

## Inventory System

- The inventory was built using a slot-based UI, with draggable components handling pointer events for item movement, swapping, usage, and removal. Each slot uses ScriptableObjects and the Observer pattern to sync state between the main inventory and the hotbar, allowing decoupled communication and real-time UI updates. Items are designed as extensible ScriptableObjects with support for equippable and consumable behaviors using polymorphism and interfaces.

## UI Design

- The UI dynamically reflects inventory changes. A clean layout ensures usability, and tooltips appear on item click, showing detailed information in a side panel. The interface is designed to be intuitive and responsive.

## Save & Load

- Inventory data is serialized to JSON and saved in Unity's persistent path. Each item is stored with a unique ID and its corresponding slot index to ensure slot-based persistence. Upon loading, the inventory state is fully restored.

## Optimization

- To improve performance, I implemented a lightweight pooling system for GameObjects and MonoBehaviours. This system is based on utility scripts from a previous project, which I've refined and reused over time to ensure efficiency and maintainability.

## Self-Assessment

- I'm satisfied with the modularity and overall structure of the system. Given more time, I would focus on improving polish by adding better UI feedback, more refined animations, and a bit more environmental detail. I would also integrate Unity's Sprite Libraries to achieve more dynamic and maintainable 2D animations.
  Due to a bug in the selected Unity version, I had to slightly adapt the animation system, switching from the Animator to a custom sprite-swapping system controlled via code.

## Asset Packages Used:

- Ninja Adventure Assets - by Pixel-Boy
- Slash Effects FREE - by Matthew Guz