Unity Technical Task: System Design and Implementation Review

This technical task was approached with a strong focus on clarity, maintainability, and scalability, while keeping the overall solution simple and easy to understand. The project is structured around two main scenes: a Main Menu scene and a Game scene. This separation allows for a clean flow between initialization and gameplay, while keeping responsibilities clearly defined.

In the Game scene, player controls are intentionally basic and accessible. Character movement supports keyboard input using WASD and arrow keys, as well as gamepad input via the left analog stick. Input handling is implemented using Unity's New Input System with a generated C# input class, and player intent is communicated through an event-driven approach. This decouples input from gameplay logic and allows the movement system to remain flexible and reusable.

Item interaction is designed to be straightforward: items are collected automatically when the player character moves over them. For the scope of this task, two consumable items are implemented—Health Potions and Mana Potions—which are added to a slot-based inventory system and can be used by the player. The inventory UI supports adding, removing, moving, swapping, and using items, with the UI dynamically updating based on inventory state.

To improve usability, both the inventory and pause menu are accessible via buttons located in the bottom-right corner of the screen. This avoids complex input combinations and ensures clarity for the player. The save and load system persists the inventory state on a per-slot basis, allowing accurate restoration of items when the game starts.

Throughout development, I intentionally avoided overengineering. I applied design patterns and architectural principles I am familiar with—such as controller–view separation and observer-style events—only where they provided clear benefits. From a personal assessment perspective, I am satisfied with the overall structure and readability of the project. With additional time, I would further polish UI feedback, improve edge-case handling, and add automated tests for core systems. Overall, this task reflects my practical approach to Unity development in a production-oriented environment.