System Overview

The system is designed to simulate a functional shop within a game, using Unity’s features like ScriptableObjects and the MVP architecture. ScriptableObjects are utilized for managing inventories and handling item data. In this setup, Inventory and ItemData are implemented as ScriptableObjects, providing a clear and flexible structure to store items and their attributes. The ShopManager manages the transactions and updates the inventories, while UI components like ShopUI and InventoryUI display the items and handle user interactions.

The system uses Unity's UnityEvent to implement an event-driven architecture, which decouples the UI updates from the core transaction logic. Events like buyItem and sellItem are invoked in the ShopManager when items are bought or sold, ensuring that the UI components listen to these events and refresh their displays accordingly.

Thought Process During the Interview

Initially, I ensured the basic functionality of buying and selling items was correctly implemented using ShopManager. I then focused on the UI, ensuring it dynamically populated and updated based on the inventory data. A significant consideration was making the code modular and maintainable, which led to the refactoring of repeated code into more generic methods. This involved creating reusable methods for populating shop slots and handling button interactions. Additionally, implementing event-driven architecture ensured that UI updates were decoupled from the core logic, enhancing maintainability and scalability.

Personal Assessment

I think I did well in meeting the main goals and creating a system that works smoothly and can be easily expanded. At times, asking more questions could have helped clarify certain details, especially with how different types of items should be managed. Next time, I'll focus more on getting a thorough understanding of the requirements upfront to minimize changes during development.