System Explained:

In this project, I implemented an inventory and trading system. The process is divided into several key parts, each managed by a different class.

First, I defined the game item using the Item_SO class. This class inherits from ScriptableObject and allows me to define essential attributes such as cost, name, and icon. This makes it easy to create various items in a modular and reusable way within the Unity editor.

To handle inventory logic, I created the Inventory class. This class uses a dictionary to track the item and its quantity, and provides methods to add and remove an item, as well as manage the player's money. It also includes mechanisms to verify if the player can buy or sell an item based on its cost and quantity.

For the user interface, I implemented several classes. Panel is an abstract class that serves as the base for all UI panels, allowing an element to be shown or hidden on the screen. ItemDisplay manages the display of a single item, showing its icon and handling its visibility. InventoryDisplay manages the inventory display, updating the UI as items are added or removed.

The InteractionWidget is another UI panel that displays interaction messages to the player, such as when entering a vendor's area. This widget updates dynamically to reflect the available action.

The NPCSeller class represents the vendor in the game. It initializes its inventory with a predetermined list of items and manages interactions with the player when they enter or leave its area of influence.

InventoryPanel, a subclass of Panel, displays the player's inventory and allows item equipment. It implements the IInventoryOwner interface to ensure that any class handling an inventory provides access to it.

The GameManager acts as a singleton, centralizing the game logic and managing the visibility of UI panels. This class controls when to show or hide the inventory and shop, ensuring a consistent user experience.

Finally, the Player class handles player movement and interactions with the inventory and shop. It detects player input and allows opening or closing these panels using specific keys, enhancing gameplay.

However, I didn't manage to implement player equipment of purchased items from the shop or reflect prices in the user interface. Instead, I focused on creating solid and reusable structures, which will allow me to be more efficient in the future. This modular approach ensures that the system's foundations are robust and can easily expand, facilitating future improvements and additions.

In summary, this inventory and trading system is robust and modular, providing a solid foundation for managing items and economy within the game. Each component is designed to interact cohesively, facilitating both development and user experience, with a focus on efficiency and long-term scalability.