My Approach

I decided to start with the basics while thinking about how to implement some of the systems. So, I created a small scene with tilesets, using tilesheets from the Unity Assets Store and free resources from Kenney.nl, OpenGameArt.org, and Itch.io. After performing the main tasks from the editor, such as configuring the characters, importing basic packages, creating the InputActionAsset, i started with the code that i will try to explain it as clearly as possible below.

Explanation of the Systems

Utilities

To simplify tasks and due to the limited time to develop this project, I have heavily relied on the use of singletons, specifically for PlayerManager.cs, UIManager.cs, and LevelManager.cs. To achieve this, I used a script that allows any MonoBehavior to be easily managed as a singleton.

Data

As a database, I simply used ScriptableObjects, specifically to create coins (I ended up using just one), dialogues, and items. Within the components where references to this data are stored, I sometimes used serialized classes to maintain coherent groups of data.

Managers

A practice I always follow is to have a component as a manager, where I have most references to other important components, thus minimizing the use of “GetComponent<>()” calls and similar methods. In this specific implementation, I used singletons, discarding the use of events and delegates due to time constraints.

Inputs

I used the "New Input System" for the implementation, utilizing the PlayerInput component from this package. From each component that needed it, subscribed to each action with a declared “ActionName” string that allows changing the input configurations from the editor without having to program.

Inventory and Wallet

I implemented a single component for this, dedicated to storing a list of ScriptableObject elements as an inventory of items and a serialized class that

contains a type of currency and its quantity. Initially, I considered making the wallet more flexible, but I ultimately opted to keep it simple (you could say it's a bit hardcoded).

Equipment

For the equipment, I maintained a similar approach, having serialized classes that contain references to the SpriteRenderers and ScriptableObjects, which specify the item names, sprites, prices, etc.

Interactions

To achieve interactions, I used OnTriggerEnter2D and OnTriggerExit2D to allow player interaction with NPCs by pressing the "E" key.

Dialogues

When interacting with NPCs, all of them trigger dialogues. These dialogues are separated into dialogue lines within a ScriptableObject. If the NPC is marked as "isVendor," the store panel will open at the end of the dialogue.

UI

For the UI implementation, I kept everything under a main canvas. Each window is on its own canvas and has a canvas group. This allows me to not deactivate them but rather reduce the alpha to 0, making them stop blocking raycasts and being interactable. All canvases are referenced in the UIManager script for easy access.

Equipping Clothing

This is done from the Inventory Canvas. When clicking on an item, if it is of type SOEquippableItem, a panel will show with an equip button. Upon equipping it, a small icon will appear in the corner of the item to indicate that it is in use, and the equip button will disappear, replaced by an unequip button.

Store

I implemented the store to open after talking to certain NPCs marked as vendors. Initially, I considered implementing a drag & drop system for items, but I ended up discarding it due to the lack of time to implement it correctly as I would have liked. Instead, I implemented a simple system using "Buy" and "Sell" buttons that add to a basket and calculate the money needed for the transaction.

Audio:

I decided to add small SFX to the UI at the last minute, so they are added directly as OnClick events of the buttons. These are simply AudioSources placed in the scene.

Conclusions:

This task was very fun for me, especially due to the challenge presented by the limited time. Although many of the ideas I initially thought of implementing couldn't be realized, I am personally very satisfied with the result.

One of the things left out, besides the drag and drop implementation for the store, is a character menu ("C") from which different clothing pieces could be equipped. In the end, I implemented it with a simple button on each inventory item ("Tab").

Additionally, there are many other small details that I would have liked to improve and that you will surely notice when testing the game.

I hope I haven't left out any details and that you like the result.