Camilo Andres Rojas Hernandez | :

System Description: All the scripts and prototype has been created in 48 hours. The developed clothing store system is based on an MVC (Model-View-Controller) architecture, organizing code in a modular and scalable manner. Key components include:

- **Tools such as Sprite Loader and Data Storage.**.
- **GameInstaller and Managers:** Installs the game and registers essential services such as economy, inventory, and UIFacade for the store.
- **Playable:** Interfaces and classes enabling player interaction with in-game objects like InteractableObject, DresserInteraction, ShopInteraction, ShopPayInteraction, and ShopSellInteraction; and Store System.

Architecture and Components:

- **MVC UI Architecture-**
    - **UI Components:** Manage UI visualization and behavior.
    - **Controllers:** Handle logic for interaction between models and views.
    - **Models:** Represent game data such as inventories, clothes, and transactions.
- **Service Locator and IOC.**
    - Services and managers are stored on a Service Locator.

System Execution: The GameInstaller serves as the starting point, registering essential services and configuring the user interface. Various interaction components allow players to interact with the store, select clothes, add items to the cart, and complete transactions. The UIFacade centralizes UI operations, displaying initial data and managing player interactions.

Development Process:

- **Architecture Design:** A modular MVC architecture was defined to separate responsibilities, ensuring scalability and maintainability.
- **Implementation of Essential Services:** Economy, inventory, and trade services were configured using a ServiceLocator pattern to centralize access.
- **Development of Interaction Components:** Interfaces and classes were implemented to facilitate player interactions with objects like DresserInteraction and ShopInteraction.
- **UI Management:** The UIFacade was developed to centralize UI management, enabling easy interaction with various UI components and controllers.

Self-Assessment: The system development was successful, meeting all requested features within the set timeline. The code maintained a clean and organized structure, adhering to programming best practices such as separation of concerns and modularity. Areas for improvement include initial resource search, MVC architecture implementation iterations, and character animation challenges due to time constraints and SpriteSheet compatibility issues.

Code Iterations and Refactoring: Following several iterations, the UI was integrated into the service architecture using a ServiceLocator to centralize services in MonoBehaviour, allowing for code reuse. Implementing the item purchase system post-architecture creation facilitated the development of sales systems and clothing organization, demonstrating the effectiveness of the implemented architecture.