The Herbalist - Technical Assessment Document

System Architecture Overview

The Herbalist project represents a comprehensive Unity-based game implementation featuring a modular architecture built on Singleton Manager patterns and Data-Driven Design principles. The system was designed to address all technical requirements while maintaining scalability and maintainability for future development.

Core System Architecture

The project employs a **Modular Game Architecture with Singleton Managers** that centralizes control while maintaining clear separation of concerns. Located within the Assets/Custom/Scripts directory, the architecture consists of six primary systems:

Interaction System: Manages player interactions with environmental elements and NPCs through a standardized IInteractable interface and Mouse Interaction controller. This system provides a unified approach to object interaction, ensuring consistent behavior across different interactive elements.

Inventory System: Implements a comprehensive slot-based inventory management solution featuring dynamic UI updates, drag-and-drop functionality, item swapping, and usage mechanics. The system supports both consumable items (health potions) and equipment, with real-time visual feedback and tooltip integration.

Player System: Controls character movement, animations, and input handling through Unity's New Input System via PlayerControllerInput. Additionally manages camera controls, permanent speed buff systems, and pause functionality, providing a complete player experience framework.

SaveSystem: Delivers robust persistence functionality using JSON serialization for inventory states, acquired buffs, and inventory upgrades. The system ensures data integrity and supports seamless save/load operations across game sessions.

Vendor System: Encompasses complete NPC trading functionality including purchase/vending logic, dedicated UI systems, IInteractable extension for merchant NPCs, and integrated economy management with corresponding user interface.

Environment System: Administers plant collection mechanics based on ItemData assignments, enabling dynamic resource gathering aligned with the game's herbalist theme.

Data-Driven Design Implementation

The architecture emphasizes data-driven design through extensive use of ScriptableObjects for item definitions and JSON structures for save data. This approach allows for content modification, game balancing, and feature additions without core code alterations, significantly enhancing development flexibility and iteration speed.

Technical Integration

All systems utilize Unity's New Input System for consistent input handling, while the Singleton pattern ensures global accessibility and centralized state management. The modular design facilitates independent system evolution and potential code reusability across projects.

Development Process and Thought Process

Initial Analysis and Planning

When presented with open-ended requirements, the primary challenge was establishing a concrete implementation strategy without predetermined specifications. The broad nature of the technical requirements necessitated careful architectural planning to ensure all components would integrate seamlessly while maintaining individual system integrity.

The decision to implement Singleton Managers with Data-Driven Design emerged from the need to create a scalable single-player game system that could accommodate future expansions. This architectural choice provided the foundation for building interconnected systems that remain loosely coupled and independently maintainable.

Resource Management Strategy

To optimize development time allocation, I utilized Unity Asset Store resources for visual assets, allowing complete focus on code implementation and system architecture. This strategic decision enabled deeper concentration on meeting technical requirements while maintaining visual quality standards necessary for prototype demonstration.

Development Methodology

The implementation process followed an iterative approach, beginning with core system foundations and progressively adding complexity. Each system was developed with extensibility in mind, ensuring that future features could be integrated without requiring architectural refactoring.

The emphasis on modular design meant that systems could be developed and tested independently before integration, reducing debugging complexity and enabling parallel development streams when time constraints allowed.

Technical Challenges and Solutions

Key challenges included implementing seamless drag-and-drop inventory mechanics, ensuring save system reliability across different data types, and maintaining UI responsiveness during dynamic content updates. These were addressed through careful event-driven programming, robust error handling, and comprehensive testing protocols.

Performance Assessment

Project Management and Execution

The development period coincided with personal commitments, including proximity to my birthday, which created discontinuous development schedules. However, effective external task management through structured to-do lists enabled successful prototype completion and achievement of personal satisfaction with the final result.

Despite time constraints, all core requirements were successfully implemented within the specified timeframe. The modular architecture proved beneficial during periods of interrupted development, as individual systems could be resumed without extensive context rebuilding.

Technical Achievement Evaluation

The final implementation successfully addresses all technical requirements while exceeding baseline expectations through additional features such as economic systems, buff management, and comprehensive save functionality. The architecture demonstrates professional-level planning and execution, with clear evidence of scalable design principles.

The integration of Unity's New Input System showcases adaptability to modern Unity development practices, while the Singleton pattern implementation demonstrates understanding of design patterns appropriate for game development contexts.

Future Development Plans

The prototype serves as a foundation for continued personal and professional development. Plans include expanding the herbalist gameplay mechanics, implementing additional NPC interactions, and enhancing the economic simulation aspects. This continued development will serve both as a portfolio piece and a platform for exploring advanced Unity features and game design concepts.

The successful completion of this technical assessment demonstrates capability in Unity development, architectural design, and project management under constrained timelines. The resulting system provides a solid foundation for further development and showcases practical application of modern game development principles.