REQ4: Crafting at the Arcane Benches

This rationale outlines the design and implementation of the Arcane Crafting mechanic. The feature enables players to craft weapons and magical items using relics mined from the Hollow Roots. The crafting interaction is triggered when a player stands adjacent to an Arcane Bench while holding the required materials.

Design Implementations and Evaluations

Implementation A: CraftAction class to handle crafting logic

Pros	Cons
Encapsulates all crafting behaviour in a dedicated Action class	Slight duplication in inventory checking and message generation
Clearly separates execution and menu description	Requires RecipeHandler to be used in other classes (e.g., ArcaneBench)
Easy to extend for additional crafting effects or output logic	Resulting item names must match the crafting logic in Recipe

Example Scenario: If a new weapon is added (e.g., “Deadly Dagger”), only a new Recipe and optionally a new weapon class need to be created. CraftAction will automatically generate crafting logic and descriptions using the recipe data.

Maintainability: The logic for executing crafting and describing the option is completely encapsulated, making it easy to test or replace.

Improvement: The logic for dynamic feedback to the player (e.g., missing items) could be extracted into a reusable utility.

Implementation B: ArcaneBench class providing proximity-based crafting access

Pros	Cons
Encapsulates crafting trigger within a dedicated Ground type	Checks all surrounding tiles every turn, which can be inefficient if scaled
Prevents non-player actors from using crafting	CraftActions are created fresh every turn (slight overhead)

Extensibility: New variants of benches (e.g., DayBench, EnchantedBench) can be created with specialised constraints.

Benefits: Supports Single Responsibility and Open/Closed Principles.

Implementation C: Recipe class for defining item requirements and crafting logic

Pros	Cons
Allows easy configuration of recipes and materials	Uses string comparison on class names, which is fragile if refactored
Easily expandable for new items	No validation on item compatibility with player classes/stats

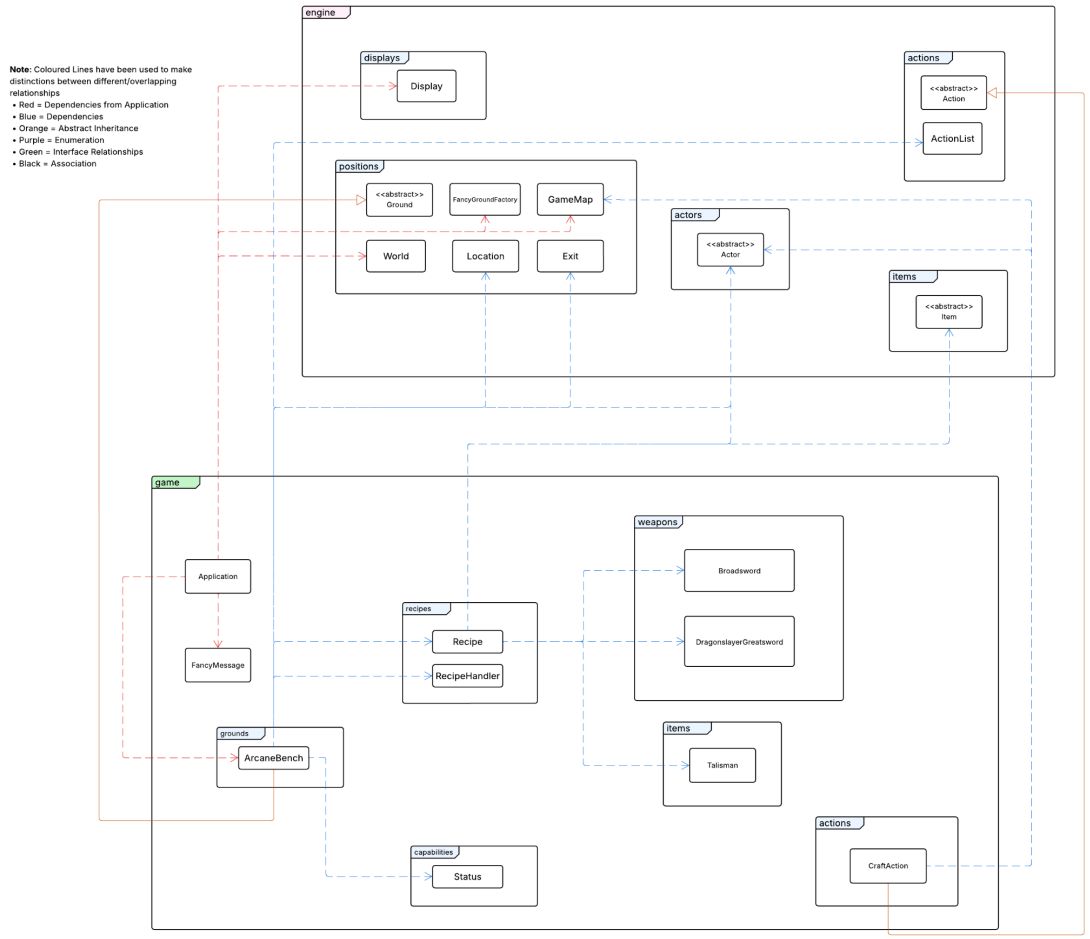
Maintainability: Clear separation of concerns: crafting recipes know how to check and produce items.

Implementation D: RecipeHandler for recipe lookup and central management

Pros	Cons
Stores all known crafting recipes in one place	Uses string comparison on class names, which is fragile if refactored
Can easily be extended to load recipes from external config	Lacks sorting or filtering logic

Example Scenario: If crafting is moved to a dynamic system where recipes are unlocked gradually, RecipeHandler can be modified to return only known recipes.

UML Design Reflection



The UML diagram above illustrates the structure and interactions of the crafting system that was introduced in Requirement 4.

The crafting functionality is well distributed across CraftAction, Recipe, and RecipeHandler, which adheres to the Single Responsibility Principle by separating crafting logic, data, and validation. We have here is the RecipeHandler, which can manage all available recipes centrally and enable new crafting options to be added without modifying core logic, which also supports the Open/Closed Principle. The system can check the inventory dynamically, keeping crafting loosely coupled to specific item types. Additionally, the crafted items are designed with polymorphism to reuse the existing weapon and item classes. This implementation has made the system compliant with the Liskov Substitution Principle, which is very crucial. Lastly, the ArcaneBench class has implemented contextual detection, offering available CraftAction when the player is nearby. This decentralized interaction model supports cohesion and avoids menu transitions, which maintains a seamless in-world crafting experience.

In conclusion, this design was chosen for its maintainability, scalability, and clean separation of responsibilities. It also allows for future recipes and features to be easily integrated with minimal impact on the existing system.