

Use Case Narrative and Functional Test Plan

Lindsey A. Monyelle

DAT 445 Predictive Analytics and Machine Learning (WI)

Dr. Jess Schwartz

September 15, 2026

Project Overview

The proposed system is an independent structured catalog for combat items in Kingdom of Loathing (KoL). Currently, information about combat items is spread across wiki pages, item descriptions, player knowledge, and other sources, which makes it difficult for players and script authors to quickly compare items based on their actual combat effects. The proposed system would organize combat items into consistent categories and searchable properties, including elemental damage, healing effects, combat effects, unusual behaviors, inventory quantity, and Mall price information. The primary users would be KoL players who want to make informed decisions about which combat items to use, as well as script authors and community maintainers who need structured and consistent item information.

Use Case Narratives

Use Case 1: Search Combat Items by Element

Primary Actor	Mafia User
Supporting Actors	• KoLmafia • Combat Item Catalog
Goal	Find combat items that cause a specific type of elemental damage.
Trigger	The player needs a combat item that causes a particular elemental damage type.
Preconditions	• The combat item catalog is available. • Combat items have been entered into the catalog. • The player can access the search function of the script.
Postconditions	• The system displays combat items matching the selected criteria. • The player can review the available items and their relevant properties.
Main Success Scenario	1. The player runs the search script in Mafia 2. The player searches for an elemental damage type. 3. The system searches the catalog for combat items with that elemental effect flagged as true. 4. The system returns matching combat items. 5. The player reviews the results to select an appropriate item.
Alternate Flows or Exceptions	• If no combat items cause the selected elemental damage type, players are prompted to double check the input or report a bug. • If an item causes multiple elemental damage types, it appears when searching for each applicable element.

Related Requirement or User Story	<i>As a KoLMafia user, I want to search for combat items by element so that I can quickly find items that cause specific elemental damage.</i>
-----------------------------------	--

Use Case 2: Identify and Exclude Unwanted Combat Behaviors

Primary Actor	KoL Player
Supporting Actors	Combat Item Catalog
Goal	Find combat items without accidentally selecting an item with an unwanted secondary effect or behavior.
Trigger	The player searches for combat items and needs to avoid one or more unwanted behaviors.
Preconditions	• Combat items have all known behaviors categorized • Built-in exclusion flags are toggleable for common unwanted behaviors.
Postconditions	• The system returns items that meet the player's behavior requirements • Players are warned of potentially disruptive behaviors if exclusions are not toggled on in case of error
Main Success Scenario	1. The player searches for combat items based on their desired combat effect. 2. The system identifies known secondary effects and behaviors for matching items. 3. The system applies built-in exclusions for behaviors that are excluded by default, but this can be turned off. 4. The player can add or remove behavior exclusions granularly. 5. Excluded behaviors are likewise excluded from the results. 6. KoLmafia displays a warning when a potentially relevant behavior may have been missed. 7. Items with unknown, conditional, or difficult-to-categorize behaviors are flagged for review.
Alternate Flows or Exceptions	• If an item has a behavior that cannot be confidently categorized, the system does not silently place it into an inaccurate category. • Users are given the ability to do ad hoc categorization of an item with unique behaviors for the criteria of the search. • If a potentially relevant exclusion may have been forgotten, KoLmafia warns the user.

Related Requirement or User Story	<i>As a KoL player, I want to know that any combat item I use is not going to have unforeseen circumstances like accidentally banishing a particular monster.</i>
-----------------------------------	---

Use Case 3 Compare Combat Items

Primary Actor	Mafia User
Supporting Actors	• KoLmafia • Combat Item Catalog • KoL Mall
Goal	Compare combat items based on their effects, relative strength, inventory quantity, and Mall price.
Trigger	The player wants to determine which of several combat items is appropriate to use.
Preconditions	• The combat items being compared exist in the catalog. • Item effect information is available. • Inventory and/or Mall price information is available when applicable.
Postconditions	• The system displays the resulting items and their relevant comparison information. • The player can use the information to inform their choice.
Main Success Scenario	1. The player searches for combat items based on a criteria. 2. The player chooses the option to analyze. 3. The script retrieves the relevant properties for each item. 4. KoLMafia displays all available information about each returned item in a table format. 5. The player reviews the comparison and determines which item meets their needs.
Alternate Flows or Exceptions	• If an item does not have a current Mall price, the system indicates that the item is untradeable. • If their inventory quantity for everything is none, the user is asked if they want to check in their closet or Hagnks instead.. • If the selected items have additional effects, the script will return that information in verbose mode.
Related Requirement or User Story	<i>As a KoL player, I want to compare my available combat items to make the best choice.</i>

Functional Test Checklist

Use Case 1: Search Combat Items by Element

Confirm that:

- ☐ the user can run the search script successfully from KoLmafia.
- ☐ the user can select or enter a valid elemental damage type.
- ☐ Mafia returns combat items with the selected element.
- ☐ items without the selected elemental damage phylum are not included in the results.
- ☐ an item with multiple elemental damage types appears when searching for each applicable element.
- ☐ searching for an element with no matching items prompts the user to double-check their input or report a possible catalog issue.
- ☐ the results display a list of relevant items.

Use Case 2: Identify and Exclude Unwanted Combat Behaviors

Confirm that:

- ☐ the database identifies all known behaviors associated with returned combat items.
- ☐ behaviors that are commonly unwanted as secondary behaviors are excluded by their built-in exclusion flags, which are enabled by default.
- ☐ the user can disable any built-in exclusion as desired.
- ☐ the user can granularly create exclusions based on any categorized data.
- ☐ the system removes items with an excluded behavior from the applicable results.
- ☐ KoLmafia prints a warning when a potentially disruptive behavior may have been missed in the search criteria.
- ☐ items with conditional, or difficult-to-categorize behaviors are flagged for review..
- ☐ the user can create ad hoc inclusions or exclusions for uniquely behaving items relative to the current search without modifying its permanent catalog classification.
- ☐ users can return a list of items with ad hoc information and reset some or all to default.

Use Case 3: Compare Combat Items

Confirm that:

- ☐ the user can search for combat items using a defined search criterion.
- ☐ the user can select the option to analyze the returned combat items.
- ☐ the script retrieves the relevant properties for each returned item.
- ☐ KoLmafia displays the returned items and their available information in a table format.
- ☐ inventory quantity is displayed correctly for items the user currently possesses.
- ☐ Items flagged as untradable are displayed as such without searching the Mall.
- ☐ the system asks whether they want to check their closet or Hagnk's if no items are returned.
- ☐ additional combat effects are displayed when verbose mode is enabled.
- ☐ the comparison displays enough information for the user to evaluate the available combat items.

Writing these use cases helped clarify the scope and nature of a project that I have been thinking about tackling for quite a while. Breaking the idea down into specific user interactions made it easier to distinguish the core functionality from features that could be added later. The use cases showed that the project is not simply a searchable list of combat items, but a structured system for understanding item behaviors and helping users make decisions about what to use. I also identified the need to distinguish between behaviors that are known and categorized and behaviors that are uncertain or difficult to classify, because the game's content is rich and complex. The alternate flows also clarified how the system should handle no results, multiple effects, default exclusions, and user overrides. These use cases give me a much clearer idea of what the initial version of the project needs to accomplish and provide a useful foundation for later process modeling, prototyping, and system design.

References

- A KoL Wiki. (2026, July 8). *Banishing*. A KoL Wiki. Retrieved August 30, 2026, from <https://wiki.kingdomfloathing.com/Banishing>
- A KoL Wiki. (2024, January 18). *Combat Items*. A KoL Wiki. Retrieved September 1, 2026, from https://wiki.kingdomfloathing.com/Combat_Items
- gausie. (2026, August 22). *kolmafia/src/data/modifiers.txt*. Github. Retrieved August 30, 2026, from <https://github.com/kolmafia/kolmafia/blob/4b19681c000308d08c4e55452c246883847a4d6c/src/data/modifiers.txt>
- KoLmafia Wiki. (2026). CLI reference. KoLmafia Wiki. Retrieved September 1, 2026, from https://wiki.kolmafia.us/index.php?title=CLI_Reference
- KoLmafia Wiki. (2021, March 15). *Custom Combat Script - Kolmafia*. KoLmafia Wiki. Retrieved September 1, from https://wiki.kolmafia.us/index.php?title=Custom_Combat_Script
- KoLmafia Wiki. (2023, March 10). Modifier Maximizer - Kolmafia. KoLmafia Wiki. Retrieved September 1, 2026, from https://wiki.kolmafia.us/index.php?title=Modifier_Maximizer