Product Requirements Prompt (PRP)

Feature: Multi-Select Checkbox Filters

Version: 1.0

Objective

To modify the index.html file to include two groups of dynamically generated, multi-select checkboxes for families and tags. This requires significant refactoring of the JavaScript to create a unified filtering pipeline that combines the existing text search with the new checkbox filters.

Execution Plan

Phase 1: HTML Structure Modification (index.html)

1. **Add Filter Containers:**
    - Locate the <aside> element (the sidebar).
    - Below the existing search input , add two new container elements.
    - The first will have id="families-filter-container".
    - The second will have id="tags-filter-container".
    - Each container will have a heading (<h3>) to label the section (e.g., "Families", "Tags").

Phase 2: JavaScript Logic Refactoring & Implementation (index.html)

The main script block will be substantially refactored to support the new, more complex filtering logic.

1. **State Management:**
    - At the top of the script, initialize two Set objects to track the active filters: const activeFilters = { families: new Set(), tags: new Set() };. Using a Set provides fast add/delete/check operations.
2. **Dynamic Filter Generation (New Function):**
    - Create a new function, populateFilters().

- This function will:
  - **a. Extract Unique Values:** Iterate through window.vaultData once. Use Set objects to collect all unique families and tags. Handle cases where these fields might be missing or not arrays.
  - **b. Sort Values:** Convert the Sets to arrays and sort them alphabetically.
  - **c. Generate Checkboxes:** For each unique family and tag, create a containing an <input type="checkbox"> and a <label>.
  - **d. Append to DOM:** Append the generated checkboxes into their respective containers (#families-filter-container, #tags-filter-container).
  - **e. Attach Event Listeners:** As each checkbox is created, attach a change event listener to it. This listener will update the activeFilters Set (adding the value if checked, deleting it if unchecked) and then call the main applyAllFilters() function.
3. **Create a Unified Filtering Pipeline (New Function):**
   - Create a new master function, applyAllFilters(). This function will be the single source of truth for updating the UI.
   - Its logic will be:
     - **a. Get Search Query:** Get the current value from the #search-input.
     - **b. Start with Full Data:** Create a variable let filteredPlugins = window.vaultData;.
     - **c. Apply Text Search:** Filter filteredPlugins based on the search query, as the current handleSearch function does.
     - **d. Apply Families Filter:** If activeFilters.families is not

empty, further filter the results. Use the .filter() and .every() array methods to ensure a plugin has **all** the families listed in the activeFilters.families set.
        - **e. Apply Tags Filter:** If activeFilters.tags is not empty, apply the same .filter() and .every() logic for the selected tags to the results of the families filter.
        - **f. Render:** Call the existing renderCards() function with the final, fully filtered array of plugins.
4. **Update Existing Event Listeners:**
    - The input event listener on the #search-input will now simply call applyAllFilters().
    - The initial page load logic will now call populateFilters() first, and then applyAllFilters() to render the initial full set of cards.

Final Review

- Confirm that two distinct sets of checkboxes for families and tags are dynamically generated and displayed in the sidebar.
- Verify that the lists are correctly alphabetized.
- Test that checking a box filters the grid, and unchecking it reverts the filter.
- Test the "AND" logic by selecting multiple checkboxes and confirming that only plugins matching all criteria are shown.
- Test the full pipeline by typing in the search box and then applying checkbox filters to the already-filtered results.

Approval Request: The Product Director is requested to review this PRP. Upon approval, the Code Engine will proceed with the execution phase.