

Product Requirements Prompt (PRP)

Feature: Live Search Filter

Version: 1.0

Objective

To modify the index.html file to include a live search filter. This involves adding a text input to the sidebar and implementing the JavaScript logic to filter the displayed plugin cards in real-time based on user input, matching against both plugin name and developer.

Execution Plan

Phase 1: HTML Structure Modification (index.html)

1. Add Search Input:

- Locate the <aside> element that serves as the sidebar.
- Below the <h2>Filters</h2> heading, add a <div> to wrap the search input for styling and spacing.
- Inside this div, create an <input> element with the following attributes:
 - id="search-input"
 - type="text"
 - placeholder="Search by name or developer..."
- Apply Tailwind CSS classes to the input for styling (e.g., w-full, p-2, rounded-md, theme-appropriate background and text colors).

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

The main script block will be refactored to make the rendering logic reusable and to add the new filtering functionality.

1. Refactor Card Rendering:

- The existing logic that iterates through window.vaultData and creates card elements will be extracted into a new,

reusable function named `renderCards(pluginToRender)`.

- This function will accept one argument: an array of plugin objects.
- The function's logic will be:
 - Get a reference to the `#plugin-grid-container`.
 - Clear the container's `innerHTML`.
 - Loop through the `pluginToRender` array and create/append a card for each plugin, just as the current code does.

2. Implement Filtering Logic:

- Inside the main `DOMContentLoaded` event listener:
 - Get a reference to the new `#search-input` element.
 - Call `renderCards(window.vaultData)` once initially to display all plugins when the page loads.
 - Attach an input event listener to the `#search-input` element.

3. Implement the Event Listener Callback:

- The function triggered by the input event will perform the following steps:
 - **a. Get Query:** Get the current value from the search input and convert it to lowercase for case-insensitive matching (`const query = searchInput.value.toLowerCase();`).
 - **b. Filter Data:** Use the `.filter()` array method on the original, complete `window.vaultData` array. The filter condition will return true if either the plugin's name (converted to lowercase) or the plugin's developer (converted to lowercase) includes the query string.
 - **c. Re-render:** Call the `renderCards()` function, passing the newly filtered array as its argument. This will cause

the UI to instantly update with only the matching plugins.

Final Review

- Confirm the search input field is correctly placed and styled in the sidebar.
- Verify that the card rendering logic has been successfully refactored into a reusable function.
- Test the input event listener to ensure it fires on every keystroke.
- Confirm the filtering is case-insensitive and correctly matches against both name and developer.
- Verify that clearing the search input results in the full list of plugins being displayed again.

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