Product Requirements Prompt (PRP)

Feature: Detail View Modal

Version: 1.0

Objective

To modify the index.html file to include a fully functional "detail view" modal. This involves adding the static HTML for the modal structure, creating the JavaScript logic to dynamically populate and control its visibility, and attaching click event listeners to the plugin cards to trigger it.

Execution Plan

Phase 1: HTML Structure Modification (index.html)

1. **Add Modal Container:**
    - At the end of the <body> tag, after the main application , add a new container with id="detail-modal-container".
    - This container will be hidden by default (hidden) and will act as an overlay using fixed positioning (fixed, inset-0, z-50).
2. **Structure Modal Internals:**
    - Inside the container, create two child elements:
        - **Backdrop:** The first with id="modal-backdrop" will be a full-screen, semi-transparent background (bg-black/75).
        - **Content Area:** The second will be the main modal panel. It will be centered and styled with theme-appropriate colors, padding, and shadows (bg-white dark:bg-gray-800, rounded-lg, p-6).
3. **Populate Modal Content Area:**
    - Inside the content , create a header section with an <h2> for the plugin name (id="modal-title") and a close <button> (id="modal-close-button").

- Below the header, create a content body . This div will contain placeholder elements that will be populated by JavaScript. For example:
  - A for the developer: Developer:
  - A for the families list:
  - A for the tags list:
  - A for notes:

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

The main script block will be updated to manage the modal's state and interactivity.

1. **Add New Element References:**
   - At the top of the DOMContentLoaded listener, get references to the new modal elements: #detail-modal-container, #modal-backdrop, #modal-close-button, and all the content placeholder spans/divs.
2. **Update renderCards Function:**
   - Modify the loop that creates each plugin card.
   - Inside the loop, attach a click event listener to each card element.
   - The click listener will call a new function, openModal(), and pass the corresponding plugin object to it.
3. **Create openModal(plugin) Function:**
   - This new function will accept a single plugin object as an argument.
   - Its logic will be:
     - **a. Populate Text Fields:** Set the textContent of #modal-title, #modal-developer, etc., with the data from the plugin object.

        - **b. Populate Array Fields:** For families and tags, the function will dynamically create and append styled "pill" or "badge" elements for each item in the array into the #modal-families and #modal-tags containers.
        - **c. Show Modal:** Remove the hidden class from the #detail-modal-container to display the modal and backdrop.
4. **Create closeModal() Function:**
    - This function will simply add the hidden class back to the #detail-modal-container.
5. **Attach Close Event Listeners:**
    - Attach click event listeners to the #modal-close-button and the #modal-backdrop. Both will call the closeModal() function.
    - Attach a global keydown event listener to the window object. Inside its callback, check if event.key === 'Escape'. If it is, and the modal is not hidden, call closeModal().

Final Review

- Confirm clicking a plugin card opens the modal.
- Verify all data fields from the clicked plugin are correctly displayed in the modal, with arrays formatted as lists or pills.
- Test that the modal can be closed by clicking the 'X' button, clicking the backdrop, and pressing the 'Escape' key.
- Ensure the modal's styling conforms to the "Classic Pearl" / "Subscotia Dark" themes.

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