Technical Challenge: Salvit Client Configuration Console

Context

Salvit Advisors is transitioning from fragmented, spreadsheet-based workflows to a centralized, web-based SKU Management Platform. Because our clients manage thousands of SKUs across multiple brands, we need a "Pro-grade" interface that solves the problem of **scale**. The goal is to move away from editing items one-by-one and toward a powerful, rule-based configuration system.

The Goal

Build a **Dynamic SKU Management & Rule Engine** that allows users to manage product catalogs across different e-commerce companies. You must demonstrate how a user can efficiently manage thousands of SKUs using **Bulk Operations** and a **Centralized Rule Builder**.

1. Requirements & Business Logic

A. Multi-Tenant & High-Density Interface

- **Company Switcher:** Toggle between different e-commerce companies. Changing the company must refresh the entire data context.
- **Pro-Data Table:** An interactive table capable of handling many SKUs.
    - **Filtering:** The ability to filter SKUs by Category or Title.
    - **Multi-Select:** Implement checkboxes to select multiple SKUs for bulk editing.

B. The Rule Builder & Hierarchical Resolution

Instead of just editing a single SKU, the user needs to define logic that applies at scale.

- **The Rule Builder:** An interface to create a cost rule that targets:
    - **General/Global:** Applies to all SKUs.
    - **Category-Level:** Applies to all SKUs within a specific category.
    - **Title-Level:** Applies to all SKUs sharing a Title.
    - **Individual SKU:** A specific override for one item.
- **Hierarchical Resolution (The "Engine"):** You must implement a selector that calculates the **"Effective COGS"** based on this priority:
    - *SKU Override > Title Rule > Category Rule > General Rule.*

- **Date Sensitivity:** Rules can have an optional `startDate` and `endDate`. A rule matching "Today" always overrides a "General" rule (no dates) at the same level.

C. Bulk Actions & UX

- **Bulk Edit Bar:** When multiple SKUs are selected, a "Bulk Action" menu should appear (e.g., "Set Category," "Assign Title," or "Override COGS").
- **Status Badges:** Clearly indicate for each SKU if its cost is "Inherited" (from Category/Title) or "Resolved" (Specific SKU rule), or “Missing Cost” if there’s no cost defined for the SKU.
- **Detail Drawer:** Clicking a SKU opens a drawer to view its metadata and its specific **Resolution Trace** (showing exactly which rule is currently defining its cost and why).
- **Persistence:** For the purpose of this test, all changes should be **saved locally (State/LocalStorage) or simulated**. No external database connection is required.

2. Tech Stack & Tools

- **Framework:** React.js
- **State Management:** Redux (Strictly Required).
- **Styling:** Tailwind CSS (Following the provided Salvit Brand System).
- **Leverage AI:** You are **expected** to use AI tools (Cursor, Copilot, Codex, etc.) to accelerate your workflow. We are evaluating your ability to prompt, architect, and validate AI-generated code.

3. Styling Guidelines

Please use the following brand constants in your Tailwind implementation:

- **Primary Blue:** `#002D72` (`brand-blue`) — Use for headers and authority.
- **CTA Orange:** `#F47C1F` (`brand-orange`) — Use for primary actions.
- **Accent Indigo:** `#4f46e5` (`brand-indigo`) — Use for data highlights.
- **Fonts:** `Barlow` for headings, `Inter` for UI/Body.

Note

Understand carefully how mock_data.json is structured to understand how relationships are handled between SKUs, titles, and categories.