Feature Specification: Advanced Search & Filtering

1. Introduction

- **1.1 Feature Name:** Advanced Search & Filtering
- **1.2 Feature ID:** FEAT-003
- **1.3 Version:** 1.0
- **1.4 Date:** June 12, 2025
- **1.5 Author(s):** Maximo Valencia
- **1.6 Stakeholders:** Product Management, UI/UX Team, Engineering Team, Data Science (if complex search), Customers.
- **1.7 Category:** Backend, Database

2. Overview

This feature will enhance the application's search capabilities by introducing advanced filtering options and improving the relevance of search results. Users will be able to refine their searches based on multiple criteria, leading to more precise and efficient data discovery within the application.

3. Goals and Objectives

- **3.1 Primary Goal:** To enable users to quickly and accurately find specific information within the application's data.
- **3.2 Objectives:**
    - Provide dedicated filter categories (e.g., by date range, status, category, assigned user).
    - Allow combination of multiple filters.
    - Implement a clear and intuitive search interface.
    - Improve search result relevance for keyword searches.
    - Enable users to save common search queries.

4. User Stories

- As a user, I want to filter my results by date range so I can find items from a specific period.
- As a user, I want to filter by item status (e.g., "Open", "Closed", "Pending") to quickly see relevant tasks.
- As a user, I want to search for keywords *and* apply filters simultaneously to narrow down results.
- As a user, I want to save my frequently used filter combinations so I don't have to set them up every time.
- As a user, I want to see how many results are returned based on my search and filter criteria.

5. Functional Requirements

- **FR-5.1 Search Bar Enhancement:** The existing universal search bar must integrate with the new filtering mechanism.
- **FR-5.2 Filter Panel:** A collapsible or expandable filter panel must be available next to search results.
- **FR-5.3 Filter Types:** Implement the following filter types for relevant data models (e.g., "Tasks", "Documents"):
    - **Text Search:** Keyword search across relevant fields.
    - **Date Range:** From (date) and To (date) picker.
    - **Dropdown/Multi-Select:** For predefined categories (e.g., Status: Open, In Progress, Done; Type: Bug, Feature, Task).
    - **User Assignment:** Select one or more users.
- **FR-5.4 Filter Combination:** Users must be able to select multiple filters across different categories (AND logic by default).
- **FR-5.5 Clear Filters:** A "Clear All Filters" button must reset all active filters.
- **FR-5.6 Saved Searches:** Users can save their current search query and filter combination for future access.
    - Provide an input field to name the saved search.
    - List saved searches in a dropdown or dedicated section.

- **FR-5.7 Result Count:** Display the number of results matching the current search and filter criteria.
- **FR-5.8 URL Parameter Integration:** Search and filter parameters should be reflected in the URL for shareability and bookmarking.

6. Non-Functional Requirements

- **NFR-6.1 Performance:** Search and filter operations must return results within 1-2 seconds, even for large datasets.
- **NFR-6.2 Scalability:** The search infrastructure should be able to handle increasing data volumes and user queries.
- **NFR-6.3 Accuracy:** Search results must be highly relevant to the queried terms and filters.
- **NFR-6.4 Security:** Ensure proper sanitization of search inputs to prevent injection attacks.

7. UI/UX Specifications

- **7.1 Filter Panel Layout:** Clearly organized filter sections, potentially with accordions for each filter category.
- **7.2 Input Controls:** Use appropriate UI components for each filter type (e.g., calendar pickers for dates, multi-select dropdowns).
- **7.3 Active Filters Display:** Clearly show which filters are currently applied (e.g., "chips" or tags above the results, with individual clear options).
- **7.4 Empty State:** A clear message and suggestions when no results are found for a given search/filter.
- **7.5 Loading Indicator:** Visual feedback (spinner/skeleton loader) during search operations.

8. Technical Design & Implementation Details (High-Level)

- **8.1 Technology Stack:** JavaScript, Node.js, SQL

- **8.2 Backend Search Engine:** Implement a robust search backend (e.g., Elasticsearch, Algolia, custom SQL-based full-text search) for efficient querying and indexing.
- **8.3 API Endpoints:** Dedicated API endpoints for searching and filtering, accepting various parameters.
- **8.4 Frontend State Management:** Manage search query and filter states in the frontend (e.g., React Context, Redux, Zustand).
- **8.5 URL Sync:** Utilize browser history API (pushState, replaceState) to update URL parameters without full page reloads.

9. Test Cases (Examples)

- **TC-9.1:** Verify search by keyword returns relevant results.
- **TC-9.2:** Apply a single filter (e.g., by status) and verify results are narrowed correctly.
- **TC-9.3:** Apply multiple filters (e.g., keyword + status + date range) and verify combined logic.
- **TC-9.4:** Verify "Clear All Filters" resets the search and displays all original results.
- **TC-9.5:** Save a search query, log out, log in, and verify the saved search is available and loads correctly.
- **TC-9.6:** Test performance with a large number of items in the database.

10. Open Questions / Dependencies

- Which specific data models will be searchable and filterable in V1?
- What fields within those models should be indexed for full-text search?
- Do we need fuzzy searching or typo tolerance? (Out of scope for V1, but good to consider).
- What is the maximum number of saved searches a user can have?