Front-end requirements for the new "AI Tools" view

Goal: Create a new table view under "Library Management" to display, search, and filter AI tools stored in the database.

Location: Within the "Library Management" section, alongside "Job Roles" and "AI Capabilities".

UI Components & Functionality:

1. **Tab Navigation:**

   * Add a new tab labeled "AI Tools" next to "AI Capabilities".

   * Clicking this tab should display the AI Tools table view and update the active state visually, consistent with the existing tabs.

2. **Header & Controls Bar:**

   * Maintain the "Library Management" title.

   * Include the controls bar below the tabs, consistent with the other views[1]:

     * **Search Input:** Placeholder text "Search tools...". Search functionality should trigger on input change (debounced).

     * **Filter Button:** Opens a dropdown or modal for filtering options.

     * **Import Button:** (Optional, if needed) Functionality TBD, but visually consistent.

     * **+ Add New Button:** (Optional, if manual addition is needed) Opens a form/modal to add a new AI tool.

3. **Data Table (`ai_tools`):**

   * Display data fetched from the backend API endpoint querying the `ai_tools` table.

   * **Columns (based on `ai_tools.csv` [2] and UI consistency[1]):**

     * `Tool Name` (from `tool_name`)

        * `Primary Category` (from `primary_category`)

        * `License Type` (from `license_type` - potentially styled with badges/tags, e.g., "Open Source", "Commercial", "Freemium")

        * `Description` (from `description` - likely truncated with a tooltip or link to view full details)

        * `Website` (Optional - display `website_url` as a clickable link)

        * `Actions` (Icons for View/Edit/Delete, consistent with other views[1])

    * **Loading State:** Display a loading indicator (e.g., spinner or skeleton rows) while data is being fetched.

    * **Empty State:** Display a message like "No AI tools found" if the table is empty or search/filter yields no results.

4. **Search Functionality:**

    * The search input should filter the displayed tools based on matches in `tool_name`, `description`, `primary_category`, or potentially `tags`.

    * API requests should be triggered with a search parameter (`?search=query`).

    * Search should reset pagination to the first page.

5. **Filter Functionality:**

    * The Filter button should reveal options to filter by:

        * `Primary Category`: Dropdown populated with unique categories present in the `ai_tools` data. Allows selecting one or multiple categories.

        * `License Type`: Checkboxes or multi-select dropdown for "Open Source", "Commercial", "Freemium", "Unknown".

    * Applying filters should trigger an API request with filter parameters (e.g., `?category=CRM&license=OpenSource`).

    * Filters should reset pagination to the first page.

    * Active filters should be clearly indicated near the filter button or search bar.

6. **Row Actions:**

   *   **View (Eye Icon):** Opens a modal or navigates to a detail page showing all fields for the selected tool, including the full description and tags.

   *   **Edit (Pencil Icon):** Opens a modal or navigates to a form pre-filled with the tool's data, allowing modification. Requires a backend endpoint to handle updates.

   *   **Delete (Trash Icon):** Prompts the user with a confirmation dialog before deleting the tool via a backend API call. The table should refresh upon successful deletion.

7. **Pagination:**

   *   Implement pagination controls at the bottom of the table, consistent with the existing views[1].

   *   Include "Previous" and "Next" buttons, page numbers (if applicable), and a display showing the range of items currently visible (e.g., "Showing 1 to 10 of 50 results").

   *   Pagination state should interact correctly with search and filter parameters.

8. **Styling and Consistency:**

   *   All UI elements (tabs, search bar, buttons, table, pagination, modals) must strictly adhere to the styling (colors, fonts, spacing, component library like Shadcn UI if used) of the existing "Job Roles" and "AI Capabilities" views[1].

   *   Maintain responsiveness for different screen sizes.

Backend API Requirements (Implied):

*   An endpoint `GET /api/ai-tools` that supports:

   *   Pagination (`page`, `pageSize`)

   *   Searching (`search`)

   *   Filtering (`category`, `licenseType`)

    * Sorting (optional, e.g., by `tool_name`)

* Endpoints for CRUD operations:

    * `GET /api/ai-tools/{id}`

    * `POST /api/ai-tools`

    * `PUT /api/ai-tools/{id}`

    * `DELETE /api/ai-tools/{id}`

* (Optional) An endpoint to fetch distinct `primary_category` values for the filter dropdown.