**Overview**

The \*\*Chemical Inventory Manager\*\* is designed to facilitate the management of chemical supplies. It provides a user-friendly interface for viewing, sorting, editing, and adding chemical data records. The application's design is focused on creating an intuitive experience for managing large datasets efficiently.

**Design Approach**

**1. User-Centric UI Design:**

The design prioritizes ease of use and intuitive navigation. The interface follows a minimalistic layout that helps users focus on the task at hand. Key design principles considered:

**Clarity:** Use of clean lines, readable font sizes, and well-defined sections.

**Consistency:** Uniform styling across buttons, tables, and interactive elements.

**Accessibility:** Sufficient contrast and large touch targets ensure a smooth experience for users of varying abilities.

**2. Interface Layout:**

**1.Header Section:** Contains the application title and a brief description of its purpose.

**2.Toolbar Section:** Positioned at the top-right side, providing quick access to core operations like adding, deleting, saving, and refreshing data.

**3. Data Table Section:** The main content area where chemical records are displayed in a tabular format. The table supports interactive features like sorting and in-place editing.

**3. Color Palette and Typography:**

The application uses a professional color palette with a focus on neutral tones like white and light gray, complemented by a primary blue accent color. Typography choices include:

- Font Family: Arial, sans-serif for a clean and readable style.

- Font Sizes: Larger font sizes for headings and buttons to ensure easy readability.

**4. Iconography:**

Icons play a critical role in conveying the purpose of actions. FontAwesome icons are used for their versatility and visual clarity. Examples include:

Add (`fa-plus`): Represents the addition of a new chemical.

Delete (`fa-trash`): Removes the selected chemical row.

Save (`fa-save`): Saves the modified data.

**Code Design Choices**

**1. Modular JavaScript Functions:**

The JavaScript code follows a modular approach to ensure that each function has a single responsibility, making the codebase easy to maintain and extend.

Table Management Functions: Handle adding, deleting, and sorting rows.

Event Handling Functions: Centralize event listener logic to avoid code duplication.

Data Manipulation Functions: Update chemical data based on user input.

**2. CSS for Responsive Design:**

The `sty.css` file ensures that the app is responsive across devices:

**Flexbox Layouts:** Used for the toolbar and table containers to dynamically adjust based on screen size.

**Media Queries:** Ensure that buttons and table elements resize appropriately on smaller screens.

**Key Code Snippets**

**1. Event Handling for Toolbar Buttons:**

document.getElementById('add-row').addEventListener('click', function() {

openAddChemicalModal();

});

document.getElementById('save').addEventListener('click', function() {

saveTableData();

});

**2. Table Sorting Function:**

function sortTable(columnIndex) {

const table = document.getElementById('chemical-table');

const rows = Array.from(table.tBodies[0].rows);

rows.sort((a, b) => {

const cellA = a.cells[columnIndex].innerText.toLowerCase();

const cellB = b.cells[columnIndex].innerText.toLowerCase();

return cellA.localeCompare(cellB);

});

table.tBodies[0].append(...rows);

}

**3. Modal Handling for Adding a New Chemical:**

function openAddChemicalModal() {

const modal = document.getElementById('add-chemical-modal');

modal.style.display = 'block';

document.getElementById('submit-new-chemical').onclick = () => {

const newChemical = {

id: chemicalData.length + 1,

chemicalName: document.getElementById('chem-name').value,

vendor: document.getElementById('chem-vendor').value,

density: parseFloat(document.getElementById('chem-density').value),

viscosity: parseFloat(document.getElementById('chem-viscosity').value),

packaging: document.getElementById('chem-packaging').value,

packSize: parseInt(document.getElementById('chem-pack-size').value),

unit: document.getElementById('chem-unit').value,

quantity: parseInt(document.getElementById('chem-quantity').value)

};

chemicalData.push(newChemical);

displayTable(chemicalData);

modal.style.display = 'none';

};

}

This document outlines the core design principles and code implementation strategies used in the Chemical Inventory Manager.