

Chemical Supplies Table - Design Documentation

1. Overview

This document provides a detailed explanation of the design approach and decisions made during the development of the "Chemical Supplies" table. The table is a part of a web application that allows users to manage a list of chemical supplies, including functionalities such as sorting, adding, editing, deleting rows, and saving data. The table is mobile-responsive and designed to be Progressive Web App (PWA) compliant.

2. Design Goals

- **User-Friendly Interface:** The table should be easy to navigate and operate, even on mobile devices.
- **Dynamic Data Handling:** Allow users to manipulate table data (add, edit, delete) and persist these changes.
- **PWA Compatibility:** Ensure the page is responsive and functions well in a PWA environment.
- **Data Persistence:** Use localStorage to store data so that user changes are retained across sessions.

3. UI Design Choices

3.1 Layout

- **Table Structure:** The table layout is simple, with headers corresponding to the columns: ID, Chemical Name, Vendor, Density, Viscosity, Packaging, Pack Size, Unit, and Quantity.
- **Toolbar:** A toolbar with buttons for adding rows, moving rows up/down, deleting rows, refreshing data, and saving data is placed at the top for easy access.

3.2 Responsiveness

- **Mobile-Friendly Design:** The table uses a fluid layout with percentage-based widths, ensuring it adjusts to different screen sizes.
- **Touchable Areas:** Buttons and interactive elements are large enough to be easily tapped on touchscreens.

3.3 PWA Compliance

- **Meta Tags:** Proper meta tags are included to ensure mobile responsiveness and PWA compliance, such as the viewport tag.
- **Responsive Layout:** The table and its elements are designed to scale and rearrange according to the screen size.

4. Functional Design Choices

4.1 Data Handling

- **Default Data:** A `DefaultData()` function initializes the table with pre-defined chemical supply data, making it easier for users to get started.
- **Dynamic Row Creation:** Rows are dynamically created based on the data in the `DefaultData()` function. This allows for flexibility and scalability in handling large datasets.

4.2 Sorting

- **Column Sorting:** Each column header is designed to be clickable, allowing the user to sort the table data by that column. The sort order (ascending/descending) is indicated by an icon next to the column header.
- **Flexible Data Types:** The sorting function is designed to handle both numeric and text-based data, ensuring accurate sorting regardless of the column type.

4.3 Row Manipulation


- **Add Row:** The `addRow()` function allows users to add a new row to the table. The new row includes an automatically generated ID, and all other fields are editable.
- **Edit Functionality:** Users can edit the data directly in the table cells, providing a seamless user experience.
- **Delete Row:** The `deleteRow()` function removes the selected row from the table and updates `localStorage` accordingly.

4.4 Data Persistence

- **LocalStorage:** All data manipulations are stored in `localStorage`, ensuring that user changes persist even after the browser is closed. This makes the application more reliable and user-friendly.

5. User Interaction Design

5.1 Visual Feedback

- **Selected Rows:** When a row is selected, it is highlighted in blue, providing a clear visual cue to the user. This is achieved by toggling the `.selected` class on the row.
- **Checkmark for Selection:** Instead of a checkbox, a  mark is used to indicate row selection. This is more visually engaging and aligns with modern UI design practices.

5.2 Input Validation

- **Numeric Fields:** For fields such as Density, Viscosity, Pack Size, and Quantity, input validation ensures that only numeric values (integers or decimals) can be entered.

5.3 Sorting Icons

- **Sort Order Indicators:** Small icons (arrows) are used to indicate the current sort order of the table columns, providing clear feedback to the user about the state of the data.

6. Code Structure

6.1 HTML Structure

- **Table:** The HTML structure is centered around a `<table>` element with `<thead>` for headers and `<tbody>` for the dynamic data rows.
- **Toolbar:** The toolbar is implemented as a `<div>` with buttons that trigger various table operations.

6.2 CSS

- **Responsive Layout:** CSS is used to ensure the table is responsive, with media queries applied for different screen sizes.
- **Visual Styling:** Basic styling is applied to the table, buttons, and other elements to make the interface visually appealing and consistent.

6.3 JavaScript

- **DOM Manipulation:** JavaScript is used extensively to manipulate the DOM, allowing for dynamic row creation, sorting, and data handling.
- **Event Listeners:** Event listeners are attached to buttons and table headers to handle user interactions like sorting, adding rows, and saving data.
- **Data Handling:** JavaScript manages data storage and retrieval from `localStorage`, ensuring data persistence across sessions.

7. Conclusion

The design of the Chemical Supplies table is focused on providing a user-friendly, responsive, and functional interface that is easy to use and integrates well with modern web technologies. By using `localStorage` for data persistence and ensuring PWA compliance, the design is both robust and future-proof.