Inventory Management App Documentation

Developer: Nadal Bardisa Quintero (documentation by ChatGPT)

Technology Stack: Pure HTML, CSS, and JavaScript (No Frameworks)

Source Control: GitHub API for database updates

Project Overview

The **Inventory Management App** is a web-based application designed to help users interact with and manage inventory data. Developed entirely by Nadal, this app is built with pure HTML, CSS, and JavaScript without relying on frameworks. The app enables two main types of users:

1. **Authorized Users** who can modify the inventory database.
2. **Public Users** who can view and search the inventory data.

Key Functionalities

1. **Data Management**: Allows authorized users to modify data in the database using GitHub's API.
2. **Data Visualization**: Enables all users to view and filter inventory data based on various attributes.
3. **Finder**: Provides a dynamic search and filtering tool to quickly locate specific inventory items.
4. **Documentation**: Contains an in-app documentation page with technical information about the project.
5. **About Page**: Displays information about the developer, Nadal, and project background.

Project Goals

1. Provide a user-friendly interface for viewing and searching inventory data.
2. Enable secure and controlled access to database modification for authorized users.

3. Deliver a responsive and visually appealing layout without any frameworks.
4. Utilize GitHub's API for version-controlled database management.
5. Offer comprehensive documentation within the application to guide users and developers.

1. Application Structure

The app consists of four primary sections: **Inventory Management**, **Finder**, **Documentation**, and **About**. Each section is built with JavaScript for interactivity, HTML for structure, and CSS for styling.

1.1 Inventory Management

The **Inventory Management** section displays inventory items in a structured table or grid view. This section is accessible to all users for viewing, but only authorized users can perform database updates. The inventory data includes attributes such as item ID, name, category, quantity, location, and supplier.

1.2 Finder

The **Finder** is a dynamic search tool that allows users to filter and locate items based on specific attributes (e.g., name, category, location). This functionality leverages JavaScript to dynamically filter displayed inventory data as the user enters search criteria, providing instant feedback on the filtered results.

1.3 Documentation

The **Documentation** section contains detailed information on the application, including this file. It explains the app's purpose, technology stack, usage guidelines, and an in-depth look at the functionality and code structure. This section is accessible to all users and acts as a reference for understanding how the app works.

1.4 About Page

The **About Page** provides information about the developer, Nadal, including the app's purpose and background. This section enhances transparency and provides users with insight into the project's motivations and design philosophy.

2. Core Functionalities and Code Explanation

2.1 Data Visualization and Dynamic Filtering

- **Grid and Table Display**: The main inventory display consists of a grid or table layout, which organizes inventory items based on specific attributes.
- **Filtering Logic**: JavaScript is used to apply filtering criteria. When a user selects filtering options, the JavaScript function `applyFilters()` dynamically hides or displays rows based on the matching criteria.
- **Hover Effects**: Interactive hover effects are applied to highlight specific tiles or rows as the user navigates through the inventory data.

2.2 Inventory Management with GitHub API

- **Purpose**: The app uses GitHub's API to manage the database, making it easy to update inventory data and maintain version control.
- **Access Control**: Only authorized users have access to the data-modifying functions, ensuring data security.
- **API Integration**: Authorized users can push changes to the database via GitHub's API, enabling a simple and secure way to edit the database in real-time. This feature is implemented using JavaScript's `fetch()` function to communicate with the GitHub API, allowing the app to perform GET and POST requests for database updates.

2.3 Finder Function

- **Search and Filter Options**: The Finder tool allows users to search for items based on attributes like item name, location, and category. It uses a combination of dropdowns and search input fields to let users specify multiple criteria at once.
- **Real-Time Filtering**: As users interact with the Finder inputs, JavaScript immediately filters the displayed items in real time without requiring a page refresh, making it easy to locate specific inventory items quickly.

2.4 Documentation Page

- **Content**: The Documentation Page includes this comprehensive file, detailing each section of the app, how it works, and the technologies used.
- **Accessibility**: The documentation is embedded directly within the app, ensuring that users and developers have immediate access to it when needed.

2.5 About Page

- **Developer Information**: The About Page showcases Nadal's background as the developer of the project. This personal touch helps users understand the context of the app and recognize the effort that went into its creation.

3. Code Components and Structure

3.1 HTML Structure

The app's HTML structure is straightforward, containing sections for the inventory table, filter options, documentation, and about information. Key HTML elements include:

- **Filter Controls**: Dropdowns and input fields for applying filter criteria.
- **Table**: Displays inventory data with columns for each attribute.
- **Grid**: A div-based grid layout where each item is represented as a tile, designed for responsive data visualization.

3.2 CSS Styling

The CSS file includes:

- **Responsive Design**: Ensures the app works across devices and screen sizes.
- **Hover Effects**: Enhances user interaction by highlighting tiles or rows based on filter criteria or row hover.
- **Dynamic Classes**: Uses `.enlarged` and other classes to visually indicate filtered or hovered states.

3.3 JavaScript Functionality

JavaScript is at the core of the app's interactivity. Key functions include:

`applyFilters()`

- **Purpose**: Applies the selected filters to the inventory data.
- **Process**: Loops through each table row, checks if it matches the selected criteria, and hides or displays rows based on matching results.
- **Tile Highlighting**: Enlarges and highlights specific tiles that match the filter criteria.

addRowHoverEffect()

- **Purpose**: Handles interactive hover effects for the table rows.
- **Process**: Adds `mouseenter` and `mouseleave` event listeners to each row. On hover, only the tile corresponding to the hovered row is enlarged, prioritizing it over other filtered tiles.

fetchGitHubData()

- **Purpose**: Uses GitHub's API to interact with the database.
- **Authorization**: Checks if a user is authorized and enables them to update the database via POST requests.
- **Process**: Fetches and updates the inventory data in a version-controlled way, allowing secure modifications.

initializeApp()

- **Purpose**: Initializes the app by building the grid, setting up filter controls, and loading initial data.
- **Process**: Runs once the DOM is fully loaded, creating the grid layout and attaching event listeners to filter controls.

4. User Interactions and Behaviors

4.1 Authorized Users (Database Modification)

- **Action**: Authorized users have access to database modification controls.
- **Behavior**: They can update inventory data using the GitHub API integration, allowing controlled, versioned edits to the data directly from the app.

4.2 Public Users (Data Viewing and Filtering)

- **Action**: Public users can view inventory data, apply filters, and use the Finder to locate items.
- **Behavior**: JavaScript dynamically updates the displayed items as users apply filters or interact with the Finder, providing an intuitive search experience.

4.3 Finder and Filtering

- **Action**: Users apply filtering criteria or type keywords to locate specific items.

- **Behavior**: As filters are applied, JavaScript instantly updates the displayed rows and tiles to match the criteria, ensuring a smooth and responsive search experience.

4.4 Row Hover Highlighting

- **Action**: When a user hovers over a table row, the corresponding tile in the grid is enlarged.
- **Behavior**: Only the tile for the hovered row is highlighted. When the mouse leaves the row, all matching tiles based on the filter criteria are re-highlighted.

5. Future Enhancements

The app provides a strong foundation for inventory management, but there are several potential enhancements to consider:

1. **Advanced Authorization Controls**: Implement role-based access for different levels of authorized users.
2. **Sorting Functionality**: Add clickable headers to sort the inventory data by columns like name, category, or quantity.
3. **Enhanced Search Options**: Allow users to specify partial matches or use multiple criteria within a single filter.
4. **Improved Mobile Experience**: Optimize the layout further for smaller screens.
5. **Historical Data Access**: Use GitHub's version history to view previous versions of the database or track changes over time.

Conclusion

The **Inventory Management App** is a feature-rich, responsive, and accessible application that serves as a powerful tool for inventory management. Designed with simplicity and interactivity in mind, it effectively combines real-time filtering, secure database management through GitHub, and a clear user interface. Nadal's attention to detail in crafting an app using only HTML, CSS, and JavaScript highlights the potential of pure web technologies to deliver robust applications without relying on modern frameworks.