Project Management Application

By Yuval Mandler

This is a lightweight project and task management application built with **Vue.js**, designed for smooth navigation and real-time updates. The application interacts with a backend via **API calls** for creating, updating, and deleting projects or tasks while using **WebSockets** to receive live updates when changes occur.

Core Concept & Approach

The main idea behind this application is its simplicity—it doesn't handle a large dataset. Because of this, all projects and tasks are fetched on mount, reducing the number of API requests and ensuring a seamless user experience. This means:

- Only two API requests are made at the start (one for projects and one for tasks).
- Navigation throughout the app remains smooth and does not require additional API calls.
- Any changes made by other users are instantly reflected using WebSockets.

For a larger application, the approach would be different. Instead of fetching all data on mount, searches and filters would trigger API requests with debounce for better efficiency.

Key Features

1. State Management with Vuex

    - After fetching, projects and tasks are stored in Vuex for global access across components.

2. API Requests & WebSockets in a Helper File

    - API interactions and WebSocket logic are placed in a helper file, ensuring reusability and preventing excessive prop drilling.

3. Error Handling

    - All API and WebSocket errors are logged and handled inside the helper file.
    - A global error message component is used to notify users of any failures.

4. Form Validation

    - Task and project name fields are validated before submission, preventing unnecessary API requests on invalid forms.

5. Search & Filter for Tasks

    - Users can search and filter tasks for a better experience.
    - In a larger application, search would have been handled via API calls with debounce.

6. Visual Cues for Due Dates

- Task due dates turn red as they approach to alert users.

7. Reusable Confirmation Popup & Error Message Component

- A confirmation popup for deletions ensures users don't accidentally remove items.
- Both the popup and error message components are reusable, receiving texts and events as props from parent components.

8. Minimalist & Intuitive Design

- The UI is simple and clean, with pastel colors and icons for an easy user experience.

9. Storybook for Component Testing

- Storybook was used to design and test components efficiently during development.

This project showcases an optimized approach for small-scale applications while considering scalability for larger datasets.