Exercise: Build a Task Management Application

Overview
Create a task management application that allows users to manage projects and tasks within those projects. The application should showcase the ability to handle complex architecture while leveraging Vue 3 features effectively.

1) ask to prepare the project as
    a) a public git repository or a zip file of the project to easily review it
    b) prepare it as an html to be ran by you, without npm installations and so on.
2) add a bonus task, to use persistent storage of the browser

Requirements:
1. **Project Management**:
   - Users should be able to create, update, and delete projects.
   - Each project can contain multiple tasks.

2. **Task Management**:
   - Each task should have the following properties: title, description, priority (low, medium, high), status (pending, in-progress, completed), and due date.
   - Users should be able to create, update, and delete tasks.

3. **State Management**:
   - Use Vuex for global state management. The store should contain modules for projects and tasks.
   - Handle asynchronous actions for fetching data, e.g., simulating a backend service using Promises.

4. **Routing**:
   - Use Vue Router for navigation:
     - A homepage that lists all projects.
     - A project detail view that shows tasks within a selected project.
     - Forms for adding and editing both projects and tasks.

5. **UI/UX Features**:
   - Implement user feedback for actions (e.g., confirmations, alerts for errors).
   - Use a UI library (e.g., Vuetify, Element Plus, Tailwind CSS) for better design and styling.
   - Implement filters and sorting for tasks based on priority and status.

6. **Composition API**:
   - Utilize the Composition API to manage component logic.
   - Create reusable compositions to handle forms and models.

7. **Code Organization**:

- Organize components into a feature-based structure (e.g., `src/components/projects`, `src/components/tasks`, `src/store/modules`).
- Use TypeScript for type safety if familiar.

8. **Testing**:
- Write unit tests for Vuex store logic and a couple of key components (you can use Jest or Vue Test Utils).
- Consider setting up end-to-end tests using Cypress.

Implementation Outline:
1. **Setup the Project**:
- Create a new Vue 3 project with Vue Router and Vuex.
- Install your chosen UI library.

2. **Define Vuex Store**:
- Create a store with modules for `projects` and `tasks`.
- Implement actions, mutations, and getters.

3. **Create Components**:
- Create components for:
  - ProjectList (displays all projects)
  - ProjectDetail (displays tasks for a specific project, contains task management components)
  - TaskForm (for creating and editing tasks)
  - ProjectForm (for creating and editing projects)

4. **Setup Routing**:
- Define routes for the project list and project details.
- Implement navigation guards if needed (e.g., preventing access to certain routes).

5. **Use Composition API**:
- Refactor component logic to use the Composition API.
- Create a reusable `useForm` composition to handle form submissions.

6. **Add User Feedback**:
- Implement loading indicators and notifications for actions that take time (like saving data).

7. **Testing**:
- Write test cases for the Vuex store and components.

8. **Documentation**:
- Document your code and write a README that describes the features, how to run the project, and test it.

Deliverables:

- A fully functioning task management application.
- A clean and organized codebase.
- Well-structured and meaningful commit messages.
- Unit and integration tests per the above requirements.

Evaluation Criteria:
- Code quality and organization.
- Effective use of Vue 3 features (e.g., Composition API).
- Efficient state management with Vuex.
- Comprehensive routing implementation.
- User experience and feedback handling.
- Adequate test coverage.

This exercise should provide a comprehensive platform to show your skills in building a well-architected application with Vue 3! Make sure to explore best practices and keep code modular and reusable throughout the development process. Happy coding!