Pre-Meeting Task List

Good morning.

I want you to do 3 things before our next meeting:

1. Check your code for potential issues, bugs and inconsistencies. I've used copilot to compile a list of things for you to check

- Error handling: Are all errors (network, server, user input, etc.) properly caught and communicated to the user?
- Loading and empty states: Does the app handle loading and no-data situations gracefully (e.g., spinners, "no data" messages)?
- Data consistency: After creating, updating, or deleting items, does the UI always reflect the latest data?
- Async operations: Are there any race conditions, missed awaits, or potential for requests to overlap or fail silently?
- Permissions and access control: Can users only see and modify what they are allowed to?
- Input validation: Are all forms and user inputs being validated both on the client and the backend?
- Type safety: Is TypeScript (or types) being used consistently, and are there any type errors or unsafe casts?
- Unfinished features: Are there any TODOs, placeholders, or feature stubs that might confuse users or developers?
- Inefficient or repeated logic: Is there code that could be refactored to be more efficient or reusable?
- Error logging: Are errors only logged to the console, or are users informed appropriately when something goes wrong?
- Component communication: Are events between components properly handled, and do emitted

events have listeners?

- Security practices: Are sensitive data and operations protected? Are passwords and tokens handled securely?

- Edge cases: Are uncommon user actions (like rapid clicks, double submissions, or invalid URLs) handled gracefully?

- Comments and documentation: Is the codebase well-commented and easy to understand for someone new?

2. Implement roles for users.

Users can be assigned roles, such as admin, project manager, contributor. I want you to implement 3 things:

1. a migration to add a new table with the roles. the table needs at least 3 columns (id, key and description, we'll usually work with the key of the role). we'll also need a seeder for this, so we will always have the roles available, when migrating to a fresh db. then add another new table user_roles, where we link the users to the roles. hint: users can have MULTIPLE roles. e.g. an admin also has the roles project manager and contributor.

2. a composable that will check if the currently loggedin user has a specific role. e.g. you call it like userHasPermission('admin') and it will either return true or false.

3. a page where you can assign roles to a user. important: only administrators should be able to access this page!

3. look at existing project management tools. for example linear.app - create an account there for yourself, create a project and add the tasks that i just gave you to that project. in general, play around and get familiar with linear. then, come up with a list of features that linear has, but our app currently does not have. we will talk about that list in our next meeting.

Hope this is not too much, but i think this is a good step into the right direction

---

1. Code Quality Review

Error Handling

[ ] All network/server/user input errors are caught

[ ] User-friendly messages are shown (not just console logs)

Loading & Empty States

[ ] Spinners or skeletons for loading states

[ ] Clear messages for empty/no-data scenarios

Data Consistency

[ ] UI reflects changes after create/update/delete

[ ] Real-time sync or proper refetch logic implemented

Async Operations

[ ] All promises are awaited

[ ] No race conditions or overlapping requests

Permissions & Access Control

[ ] Users can only access what they're authorized to

[ ] Role-based logic is respected

Input Validation

[ ] All inputs are validated client-side

[ ] Server-side validation confirms and sanitizes data

Type Safety

[ ] TypeScript or types are consistent

[ ] No `any`, unsafe casts, or type errors

Unfinished Features

[ ] No TODOs or placeholders in production code

Reusability & Efficiency

[ ] No copy-pasted logic

[ ] Extracted functions where appropriate

Error Logging

[ ] Errors are not just in console - user is notified

[ ] Logging to a service like Sentry (if needed)

Component Communication

[ ] Events emitted by components are properly handled

[ ] No unused or unlistened emits

Security Practices

[ ] No sensitive data in frontend code

[ ] Tokens/passwords are stored and transmitted securely

Edge Cases

[ ] Handles rapid clicks, double submissions, broken URLs

Comments & Documentation

[ ] Code is easy to understand for newcomers

[ ] Complex parts are commented

---

2. Implement User Roles

a) Database

[ ] Add `roles` table: id (UUID) | key | description

[ ] Add `user_roles` table: id | user_id (FK) | role_id (FK)

[ ] Seeder: Insert roles like `admin`, `project_manager`, `contributor`

b) Composable

[ ] Create `useHasRole` or `userHasPermission(roleKey: string): boolean`

[ ] Pull from user's roles and match the key

c) Roles Management Page

[ ] Create `/admin/user-roles` page

[ ] Show list of users + their roles

[ ] Form to assign/remove roles

[ ] Only accessible to users with `admin` role

---

3. Research Linear.app

[ ] Create a Linear account

[ ] Set up a project ("Permissions + QA Tasks")

[ ] Add tasks from above

[ ] Prepare a comparison list of features in Linear vs. our app