ProjectHub - Intro

ProjectHub

Internal Client-Project Management Tool

Project Requirements Document (PRD)

TABLE OF CONTENTS

Summary

ProjectHub is an internal tool designed to centralize client-project management within the organization. It aims to replace fragmented communication and data storage with a unified platform that combines project tracking, team collaboration, and resource management. Think of it as a streamlined alternative to Slack, built specifically for project-centric workflows with only the features that matter.

Objectives

- Centralize all client and project data in a single platform
- Enable efficient assignment and tracking of developers across projects
- Streamline daily reporting and project documentation
- Facilitate real-time team communication within project context
- Organize project assets and resources systematically

Success Criteria

Phase 1 will be considered complete when:

- All Phase 1 features are implemented and functional
- Dynamic role system correctly restricts access based on configuration
- Real-time chat operates without latency issues
- File uploads work within defined size limits
- Developer dashboard accurately reflects pending items and actions
- Application meets coding standards as mentioned.

Tech Stack

Technical Stack

Frontend

Layer	Technology
Framework	Next.js 16.1.0
Language	TypeScript
Styling	Tailwind CSS
UI Components	shadcn/ui
Forms & Validation	React Hook Form + Zod

Backend

Layer	Technology
API Layer	Next.js API Routes
ORM	Drizzle
Database	PostgreSQL
Real-time	Socket.io

Authentication & Security

Layer	Technology
Auth Library	Better Auth

External Services

Layer	Technology
Media Storage	Cloudinary
Email Templates	React Email
Email Service	Nodemailer
SMTP Provider	Gmail

Infrastructure

Layer	Technology
Hosting	Vercel

Development Tools

Tool	Purpose
pnpm	Package Manager
ESLint + Prettier	Code Quality & Formatting
Husky + lint-staged	Pre-commit Hooks

Core Features — Phase 1

Core Features — Phase 1

4.1 Dynamic Role & Access Management

Admins can create custom roles and define which pages/features each role can access. Route protection ensures users only see authorized content. This provides flexibility to adapt the permission structure as the organization evolves.

4.2 Client Management

A centralized registry of all clients. Each client entry includes relevant business information and serves as the parent entity for one or more projects. Clients can have multiple active projects simultaneously.

4.3 Project Management

Projects are created under clients and serve as the primary organizational unit. Each project can have multiple developers assigned. All project-related features (memos, EODs, tasks, chats, assets, links) are scoped to the project level.

4.4 Developer Assignment

Admins can assign multiple developers to a single project. Developers gain access to all project features upon assignment and can be reassigned or removed as needed.

4.5 Project Memos

- 140 chars max
- One per dev, per project, per day
- Editable only on the same day
- Admin view: All memos with date + project filters
- Dev view: Own memo history (read-only for past days)

4.6 EOD Reports

End-of-Day reports are a mandatory daily submission for developers. Two types are supported:

- **Internal EOD:** Detailed technical updates for internal team consumption
- **Client-facing EOD:** Summarized progress updates suitable for sharing with clients via external channels

4.7 Project Chat

Real-time messaging within each project context. All assigned developers and admins can participate. Chat history is persistent and searchable, providing a communication record tied to the project.

4.8 Links Management

A repository for project-related URLs (documentation, third-party services, references). Each link includes a label/description for easy identification and categorization.

4.9 Assets Management

File storage for project assets using Cloudinary. Supports uploads up to 5-10 MB per file. Assets can be images, documents, or other relevant files that need to be shared within the project team.

4.10 Developer Dashboard

A personalized landing page for developers after login. Features include:

- Quick action buttons (e.g., Add Memo, Submit EOD)
- Pending/incomplete tasks list
- Reminders for EOD/memo submissions after specified intervals

4.11 Email Notifications

Triggered email alerts for key events such as project assignments, critical updates, or system notifications. Chat will have its own real-time notification system.

Future Scope — Phase 2

Future Scope — Phase 2

The following features are planned for future implementation:

1. **EOD Compliance Tracking:** Monitor and report on EOD submission patterns, identify missed submissions
2. **Custom Task Statuses:** Allow per-project customization of task status options beyond the default set.
3. **Deadline Management:** Automated handling of missed deadlines including rollover logic and notifications
4. **Analytics & Reporting:** Dashboards and reports for project progress, team productivity, and resource utilization
5. **Task Management:** Developers create and manage their own tasks within assigned projects. Tasks include deadlines and follow predefined statuses (To Do, In Progress, Done). Tasks are self-assigned and help developers organize their daily work.

Quality Standards & PTR

Quality Standards

This is a production-grade application. All development must adhere to industry best practices with no compromises on quality.

1. **Security:** Proper authentication, authorization, input validation, and secure data handling
2. **Performance:** Optimized load times, efficient real-time communication, responsive UI
3. **Scalability:** Architecture should support growing number of users, projects, and data
4. **Maintainability:** Clean code structure, proper documentation, consistent coding standards
5. **Reliability:** Error handling, data integrity, graceful failure modes
6. **Usability:** Intuitive interface, consistent design patterns, accessibility considerations

Points to remember :

1. Make the components re-usable.
2. Loading states should be properly handled.
3. Proper use of types in every place. Don't use "unknown" or "any" types. Create proper interfaces and types and use them.
4. Must follow the below NEXTJS 15+ patterns.
    a. Use of loading.tsx, error.tsx, not-found.tsx, layout.tsx
    b. Usage of suspense boundaries in addition to hydrateClient to prevent hydation errors
    c. Server-side prefetching wherever possible.
5. Handle large datasets enlisting in the table format with proper pagination controller rather than cards.