NeonPM - Project Management Application Documentation

Overview

NeonPM is a modern, feature-rich project management application built with React, TypeScript, and Vite. Inspired by Jira, it provides a comprehensive solution for team collaboration, project tracking, task management, and communication. The application features a sleek dark/light theme with neon cyan accents and is fully responsive across all devices.

Live Demo: https://ruchika216.github.io/NeonPM/

Technology Stack

Frontend Framework & Build Tools

- **React 19.1.1** - Core UI library with latest features
- **TypeScript 5.8.3** - Type safety and better developer experience
- **Vite 7.1.2** - Fast build tool and development server
- **React Router DOM 7.8.0** - Client-side routing and navigation

UI & Styling

- **Tailwind CSS 4.1.11** - Utility-first CSS framework with latest features
- **Framer Motion 12.23.12** - Smooth animations and transitions
- **Custom CSS Variables** - Dynamic theming system

State Management & Data

- **Zustand 5.0.7** - Lightweight state management with persistence
- **LocalStorage** - Client-side data persistence
- **date-fns 4.1.0** - Modern date utility library

Charts & Visualization

- **Recharts 3.1.2** - Responsive chart library for React

Development Tools

- **ESLint 9.33.0** - Code linting and quality enforcement
- **TypeScript ESLint 8.39.1** - TypeScript-specific linting rules
- **PostCSS** - CSS processing and optimization
- **gh-pages 6.3.0** - GitHub Pages deployment

Project Architecture

Directory Structure

```
src/
├── components/           # Reusable UI components
│   ├── ErrorBoundary.tsx    # Error handling wrapper
│   ├── Layout.tsx          # Main app shell (sidebar, header, navigation)
│   ├── UserAvatar.tsx      # User profile avatar component
│   └── modals/             # Modal dialogs
│       ├── MeetingModal.tsx
│       ├── ProjectModal.tsx
│       └── TaskModal.tsx
├── pages/                # Page components (route handlers)
│   ├── Landing.tsx         # Authentication & welcome page
│   ├── Dashboard.tsx       # Main dashboard with analytics
│   ├── Projects.tsx        # Project management interface
│   ├── ProjectDetails.tsx  # Individual project details
│   ├── Tasks.tsx          # Kanban task board
│   ├── Meetings.tsx       # Meeting scheduling & management
│   ├── People.tsx         # User management (admin)
│   ├── Chat.tsx           # Team communication
│   └── Settings.tsx       # User preferences & app settings
├── store/                # State management
│   ├── data.ts            # Main data store (Zustand)
│   └── theme.ts           # Theme management store
├── assets/               # Static assets
├── App.tsx              # Main application component & routing
├── main.tsx             # Application entry point
├── index.css            # Global styles & theme variables
└── vite-env.d.ts        # Vite type definitions
```

Core Features

1. Authentication System

- **Local Authentication** - Stores user data in localStorage
- **Demo Mode** - "Try Demo" button for instant access with sample data
- **Protected Routes** - Automatic redirect to landing page for unauthenticated users
- **User Profiles** - Name, email, role, and avatar management

2. Dashboard Analytics

- **Interactive Charts** - Area charts, pie charts, and bar graphs using Recharts
- **Key Metrics** - Project progress, task distribution, team performance
- **Activity Feed** - Real-time updates on project and task activities
- **Quick Actions** - One-click creation of projects, tasks, and meetings
- **Recent Projects** - Quick access to recently worked on projects

3. Project Management

- **CRUD Operations** - Create, read, update, delete projects
- **Project Properties**:
  - Name, description, status (planning/active/on-hold/completed)
  - Priority levels (low/medium/high/critical)
  - Start/end dates, assignee, reporter
  - Team members, progress tracking
  - Labels and attachments
- **Search & Filtering** - Filter by status, priority, search by name
- **Sorting Options** - Sort by recent, progress, or alphabetical
- **Progress Tracking** - Visual progress bars and completion percentages

4. Task Management (Kanban)

- **Kanban Board** - Four columns: To Do, In Progress, Review, Done
- **Drag & Drop** - Move tasks between columns to update status
- **Task Properties**:
  - Title, description, story points
  - Priority, type (story/bug/task/epic)
  - Assignee, reporter, due date
  - Time estimation and logging
  - Labels and comments
- **Project Filtering** - View tasks for specific projects
- **Real-time Updates** - Automatic status updates on drag/drop

5. Meeting Scheduler

- **Meeting Types** - Standup, planning, review, retrospective, client, other
- **Scheduling Options**:
  - Date and time selection
  - All-day meeting toggle
  - Recurring meeting patterns
  - Attendee management
- **Meeting Links** - Auto-generate or manual entry of meeting URLs
- **Agenda Management** - Define meeting objectives and topics
- **Quick Start** - One-click meeting launch

6. Team Communication (Chat)

- **Conversations** - 1:1 and group chat capabilities
- **User Management** - Drag users into conversations
- **Message Types** - Text messages with system notifications
- **Persistent History** - Messages saved per conversation
- **Active Conversation** - Clear indication of current chat
- **Notification Integration** - Chat activity appears in notification center

7. People Management (Admin)

- **User Profiles** - Complete user information management
- **Role-Based System** - Admin, manager, developer, designer, QA roles
- **User Properties**:
  - Name, email, role, status (active/inactive)
  - Job title, department
  - Avatar (initial-based)
- **Statistics Tracking** - Projects, tasks, and hours per user
- **Inline Editing** - Quick updates to user information
- **User Actions** - Add demo users, delete users

8. Theming System

- **Dual Themes** - Light and dark mode with smooth transitions
- **Dynamic Switching** - Header toggle with instant theme changes
- **CSS Variables** - Comprehensive color system for theming
- **System Preference** - Automatic detection of OS theme preference
- **Persistent Storage** - Theme preference saved in localStorage
- **Neon Aesthetics** - Cyan accent colors with glow effects in dark mode

9. Notifications & Activity

- **Real-time Notifications** - Project, task, meeting, and chat updates
- **Notification Center** - Header bell with unread indicators
- **Activity Types** - New projects, task updates, meeting schedules, chat messages
- **Mark as Read** - Individual and bulk notification management
- **Persistent Notifications** - Stored until explicitly cleared

10. Data Management

- **Local Storage Persistence** - All data stored in browser localStorage
- **Demo Data** - Rich sample data for immediate exploration
- **Data Import/Export** - Clear data functionality in settings
- **Type Safety** - Full TypeScript interfaces for all data models
- **State Management** - Zustand for efficient state updates

Data Models

Core Entities

Project

```
interface Project {
  id: string
  name: string
  description: string
  status: 'planning' | 'active' | 'on-hold' | 'completed'
  priority: 'low' | 'medium' | 'high' | 'critical'
  startDate: string
  endDate: string
  assignee: string
  reporter: string
  team: string[]
  progress: number
  labels: string[]
  attachments: string[]
  createdAt: string
  updatedAt: string
  comments?: Comment[]
}
```

Task

```
interface Task {
  id: string
  title: string
  description: string
  status: 'todo' | 'in-progress' | 'review' | 'done'
  priority: 'low' | 'medium' | 'high' | 'critical'
  type: 'story' | 'bug' | 'task' | 'epic'
  assignee: string
  reporter: string
  projectId: string
  storyPoints: number
  labels: string[]
  dueDate: string
  estimatedHours: number
  timeLogged: number
  attachments: string[]
  comments: Comment[]
  createdAt: string
  updatedAt: string
}
```

Meeting

```
interface Meeting {
  id: string
  title: string
  description: string
  date: string
  startTime: string
  endTime: string
  attendees: string[]
  location: string
  type: 'standup' | 'planning' | 'review' | 'retrospective' | 'client' | 'other'
  agenda: string[]
  meetingLink: string
  isRecurring: boolean
  recurrencePattern?: string
  createdBy: string
  createdAt: string
  allDay?: boolean
}
```

User Profile

```
interface UserProfile {
  id: string
  name: string
  email: string
  role: 'admin' | 'manager' | 'developer' | 'designer' | 'qa'
  title?: string
  department?: string
  status: 'active' | 'inactive'
  avatarUrl?: string
}
```

State Management

Zustand Stores

Data Store (`src/store/data.ts`)

- **Primary Store** - Contains all application data and business logic
- **Persistence** - Automatically syncs with localStorage
- **Actions** - CRUD operations for all entities
- **Getters** - Computed data and filtered results
- **Notifications** - Automatic notification generation for actions

Theme Store (`src/store/theme.ts`)

- **Theme State** - Current theme (light/dark)
- **Theme Actions** - Toggle and set theme functions
- **System Integration** - Automatic OS preference detection
- **DOM Updates** - Automatic CSS class toggling

Key Store Actions

Project Actions

- `addProject()` - Create new project with notification
- `updateProject()` - Update project properties
- `deleteProject()` - Remove project and associated tasks
- `addProjectComment()` - Add comment to project

Task Actions

- `addTask()` - Create new task
- `updateTask()` - Update task properties (status, assignee, etc.)
- `deleteTask()` - Remove task

Meeting Actions

- `addMeeting()` - Schedule new meeting
- `updateMeeting()` - Modify meeting details
- `deleteMeeting()` - Cancel meeting

Chat Actions

- `createConversation()` - Start new chat conversation
- `addConversationMessage()` - Send message to conversation
- `setActiveConversation()` - Switch active chat

Routing & Navigation

Route Structure

```
/landing              # Authentication page
/                    # Dashboard (default)
/dashboard           # Main dashboard
/projects            # Project list
/projects/:id        # Project details
/tasks               # Kanban board
/meetings            # Meeting management
/people              # User administration
/chat                # Team communication
/settings            # User preferences
```

Protected Routes

- All routes except `/landing` require authentication
- `ProtectedRoute` component handles authentication checks
- Automatic redirect to landing page for unauthenticated users

Navigation Components

- **Sidebar Navigation** - Desktop persistent sidebar with navigation items
- **Mobile Drawer** - Collapsible sidebar for mobile devices
- **Breadcrumbs** - Dynamic page titles in header
- **Active States** - Visual indication of current page

Responsive Design

Breakpoint Strategy

- **Mobile First** - Base styles for mobile devices
- **Tablet** - Medium screen optimizations
- **Desktop** - Large screen layouts
- **Ultra-wide** - Extra large screen support

Mobile Features

- **Drawer Navigation** - Collapsible sidebar menu
- **Touch Interactions** - Optimized for touch devices
- **Fluid Typography** - Responsive font sizes
- **Compact Layouts** - Space-efficient mobile interfaces

Desktop Features

- **Persistent Sidebar** - Always-visible navigation
- **Multi-column Layouts** - Efficient use of screen real estate
- **Keyboard Shortcuts** - Ctrl+K for search (placeholder)
- **Hover States** - Interactive feedback for mouse users

Animation & Interactions

Framer Motion Integration

- **Page Transitions** - Smooth page change animations
- **Modal Animations** - Slide-in/fade effects for modals
- **Sidebar Drawer** - Spring-based mobile menu animations
- **Notification Popups** - Smooth notification appearance

Interaction Patterns

- **Drag & Drop** - Kanban task movement
- **Hover Effects** - Button and card interactions
- **Loading States** - Visual feedback during actions
- **Micro-interactions** - Subtle animation details

Performance Optimizations

React Optimizations

- **Strict Mode** - Development mode optimizations
- **Error Boundaries** - Graceful error handling
- **Component Splitting** - Modular component architecture

- **Efficient Re-renders** - Optimized state selectors

Build Optimizations

- **Vite Build System** - Fast development and production builds
- **Tree Shaking** - Unused code elimination
- **Code Splitting** - Dynamic imports where applicable
- **Asset Optimization** - Optimized images and fonts

Data Management

- **Local Storage** - Fast client-side data access
- **Efficient State Updates** - Minimal re-renders with Zustand
- **Computed Values** - Cached derived data

Development Workflow

Available Scripts

```
npm run dev        # Start development server
npm run build      # Build for production
npm run preview    # Preview production build
npm run typecheck  # TypeScript type checking
npm run lint       # ESLint code linting
npm run deploy     # Deploy to GitHub Pages
```

Development Setup

1. **Install Dependencies** - `npm install`
2. **Start Dev Server** - `npm run dev`
3. **Type Checking** - `npm run typecheck`
4. **Code Quality** - `npm run lint`

Build Process

1. **TypeScript Compilation** - `tsc -b`
2. **Vite Production Build** - Bundle optimization
3. **Asset Processing** - Image and CSS optimization
4. **Static Generation** - Ready for deployment

Deployment

GitHub Pages

- **Automated Deployment** - `npm run deploy` script
- **Base Path Configuration** - `/NeonPM/` base path in Vite config
- **Static Hosting** - All static assets and SPA routing support

Alternative Deployments

- **Vercel** - Drop-in deployment with automatic builds
- **Netlify** - Continuous deployment from Git
- **Traditional Hosting** - Any static file hosting service

Browser Support

Modern Browsers

- **Chrome/Edge** - Full feature support
- **Firefox** - Complete compatibility
- **Safari** - WebKit optimizations
- **Mobile Browsers** - Responsive design support

Progressive Enhancement

- **CSS Grid/Flexbox** - Modern layout techniques
- **CSS Variables** - Dynamic theming
- **ES6+ Features** - Modern JavaScript syntax
- **Local Storage** - Client-side persistence

Security Considerations

Client-Side Security

- **No Sensitive Data** - Demo application with mock data
- **Local Storage Only** - No external data transmission
- **Input Sanitization** - Safe handling of user inputs
- **Error Boundaries** - Graceful error handling

Future Considerations

- **Authentication System** - JWT/OAuth implementation
- **Data Validation** - Server-side validation
- **Rate Limiting** - API request throttling
- **HTTPS Enforcement** - Secure data transmission

Demo Data

Sample Projects

1. **E-commerce Platform** - Full-stack web application
2. **Mobile App Redesign** - UI/UX improvement project
3. **API Integration** - Third-party service integration
4. **Analytics Dashboard** - Data visualization project
5. **QA Automation Suite** - Testing infrastructure

Sample Users

- **Sarah Chen** - Product Manager
- **John Miller** - Senior Engineer
- **Mike Johnson** - Frontend Engineer
- **Emma Davis** - UI/UX Designer
- **Alex Garcia** - Backend Engineer

Sample Tasks

- Various tasks across different projects
- Different priority levels and statuses
- Realistic story points and time estimates
- Sample comments and attachments

Future Enhancements

Planned Features

- **Real-time Collaboration** - WebSocket integration for live updates
- **File Attachments** - Document and image upload capabilities
- **Advanced Reporting** - Detailed project analytics and reports
- **Time Tracking** - Built-in time logging and reporting
- **Mobile App** - Native mobile application
- **API Integration** - Backend service integration

Technical Improvements

- **Performance Monitoring** - Application performance tracking
- **Accessibility Enhancements** - WCAG compliance improvements
- **Internationalization** - Multi-language support
- **Advanced Animations** - More sophisticated UI animations
- **PWA Features** - Offline functionality and app-like experience

Scalability Considerations

- **Database Integration** - PostgreSQL/MongoDB backend
- **Authentication System** - User management and permissions
- **Real-time Features** - WebSocket or Server-Sent Events
- **File Storage** - Cloud-based asset management
- **Microservices** - Distributed architecture for large teams

Troubleshooting

Common Issues

Theme Issues

- **Dark Mode Not Working** - Check CSS variable definitions
- **Theme Not Persisting** - Verify localStorage functionality

Data Issues

- **Data Not Saving** - Check localStorage permissions
- **Demo Data Missing** - Clear localStorage and refresh

Build Issues

- **TypeScript Errors** - Run `npm run typecheck`
- **Linting Failures** - Run `npm run lint --fix`

Development Tips

- Use React Developer Tools for debugging
- Check browser console for errors
- Verify localStorage data in DevTools
- Test responsive design with device emulation

Contributing

Code Standards

- **TypeScript** - Strict typing for all new code
- **ESLint** - Follow configured linting rules
- **Component Structure** - Follow established patterns
- **CSS Classes** - Use Tailwind utility classes

Testing

- **Manual Testing** - Test all features across devices
- **Type Checking** - Ensure TypeScript compilation
- **Cross-browser** - Verify functionality across browsers

Documentation

- **Code Comments** - Document complex logic
- **README Updates** - Keep documentation current
- **Type Definitions** - Maintain accurate interfaces

Last Updated: January 2025 Version: 1.0.0 Author: NeonPM Development Team