Collaboration Application - Feature Specification

Project Overview

Build a team collaboration web application for an internship project. The app should be simple, functional, and demonstrate core full-stack development skills.

Technology Stack

- **Frontend**: React with JS
- **Backend**: Node.js with Express
- **Database**: Your choice (SQLite recommended for simplicity)
- **Real-time**: Socket.io for chat functionality
- **File Storage**: Local file system
- **Styling**: styled components

Core Features

1. User Authentication

Users need to be able to create accounts and log into the application.

- Registration with email, password, and full name
- Login functionality
- User session management
- Basic logout functionality

2. Workspaces

Workspaces represent teams working on specific projects.

- Users can create new workspaces
- Users can see all workspaces they belong to
- Workspace owners can invite other users to join their workspace
- Each workspace has a name and description
- Users can switch between different workspaces they're part of

3. Kanban Boards

Visual task management system within each workspace.

- Each workspace can have multiple boards
- Each board uses the classic kanban approach with three columns: "To Do", "In Progress", and "Done"
- Users can create tasks (cards) with a title and description
- Tasks can be assigned to team members
- Users should be able to move tasks between columns (drag-and-drop preferred)
- Tasks show who they're assigned to and when they were created

4. Real-Time Chat

Simple messaging system for team communication.

- Each workspace has its own chat room
- Messages should appear instantly for all team members in that workspace
- Chat history should be saved and loaded when users enter a workspace

- Messages show sender name and timestamp
- Basic text messaging (no need for advanced features like file sharing or emojis)

5. Document Space

Shared file storage and organization system.

- Each workspace has a document section
- Users can create folders to organize files
- Users can upload files into folders
- All workspace members can view and download files
- Basic folder structure (folders can contain other folders)
- File list shows file name, size, who uploaded it, and when
- Support common file types (PDFs, Word docs, images, text files)
- Reasonable file size limits

User Experience Flow

1. User registers/logs in
2. User sees dashboard with all their workspaces
3. User can create new workspace or enter existing one
4. Inside workspace, user sees three main sections:
    - **Boards**: List of kanban boards, can create new boards and manage tasks
    - **Chat**: Real-time messaging with team members
    - **Documents**: File management with folders and uploads

Design Requirements

- Clean, modern interface
- Responsive design (works on desktop and tablets)
- Simple navigation between features
- Clear visual hierarchy
- Loading states and basic error handling

Technical Notes

- Keep file uploads reasonable (suggest 10MB limit per file)
- Store uploaded files locally on the server
- Use proper authentication middleware
- Implement real-time features with Socket.io
- Focus on functionality over complex features

Success Criteria

The application should demonstrate:

- Full-stack development skills
- Real-time functionality
- File handling
- User authentication and authorization
- Clean, functional UI
- Proper project structure and code organization

Implementation Freedom

You have complete freedom to:

- Choose specific database schema design
- Implement API endpoints as you see fit
- Structure the frontend components however makes sense
- Handle error cases and edge scenarios
- If you feel you need to add things and you can do it with no complexity add them
- Add reasonable validation and security measures
- Organize the codebase in a logical way