Todo Application – Requirements Document

Project Overview

Build a Todo Management System where users can create, assign, and track todos with file attachments and profile features. The system should demonstrate strong architecture, clean code practices, and modern web development standards.

Functional Requirements

User Management

- User registration and login with JWT authentication
- User profile management with name, profile picture (uploadable)
- Secure logout and session management
- Admin manager role with privileges to:
  - View all users
  - Enable or disable user accounts

Todo Management

- Full CRUD operations for todos
- Todo attributes:
  - Title & description
  - Assignee(s)
  - Due date
  - Attached files (uploaded at creation)
- Ability to assign todos to self or other users
- Notification system for assigned/updated todos
- Todo Views/Filters:
  - "Todos for me" (assigned to the logged-in user)
  - "Todos for others" (assigned by the logged-in user)
- Drag-and-drop support in the UI for changing todo order/status

File Management

- Attach files while creating a todo
- Dedicated Files page listing all uploaded files with related todos

Collaboration

- Todos accessible to assigned users only
- Admin can manage users and monitor todo activity

Technical Requirements

Backend Architecture
- RESTful API with clear endpoint structure
- PostgreSQL as primary database
- Prisma ORM for schema and migrations
- Authentication with JWT and password hashing via bcrypt
- Role-based authorization (User vs Admin)
- Secure file storage (cloud bucket or local storage with access control)
- Middleware for:
  - Authentication
  - Input validation (express-validator)
  - Error handling
  - Logging (Winston or similar)

Database Design
Required tables:
- users (with profile picture path, status active/disabled)
- todos (with attributes, due date, assignee references)
- files (linked to todos)

Frontend Architecture
- Next.js 14 (App Router) + TypeScript
- Tailwind CSS for styling
- shadcn/ui for UI components
- React Query or SWR for data fetching
- React Hook Form for form management
- Axios or Fetch for API calls
- Pages:
  - Authentication (Login/Register)
  - Dashboard (overview)
  - Todos list + filters
  - Files list
  - User profile (with profile pic)
  - Admin dashboard (user management)

UI/UX Requirements
- Responsive design for web + mobile
- Drag-and-drop for todos
- Clear separation of “My Todos” vs “Assigned Todos”
- Notification alerts (in-app + optional email)
- Intuitive navigation for files and todos
- Form validation with real-time feedback

Deliverables

1. Codebase (GitHub) with clear README
2. Architecture Documentation:
   - ER diagram for users, todos, and files
   - API documentation
   - System architecture diagram
3. UI/UX:
   - Wireframes for todos, files, profile, and admin pages
   - Responsive layouts

Evaluation Criteria

- Architecture & Code Organization (40%): clean code, modular design
- Functionality & UX (30%): todos, file uploads, notifications, drag-and-drop
- Security & Best Practices (20%): auth, validation, error handling
- Collaboration & Documentation (10%): Git workflow, README, clarity