Project Management System

Business Objective:
To design and develop a scalable project management application that enables teams to efficiently plan, track, and execute projects by centralizing user authentication, project creation, task assignment, progress tracking, and collaboration within a single platform. The system aims to improve productivity, accountability, and visibility across projects while supporting integrations with developer tools and document management workflows.

Core Features:

1. Authentication
    - User registration and secure login (use JWT - sign and verify)
    - Role-based access control (Admin, Manager, Member)
    - Team and workspace support
2. Project Management
    - Create, update, archive projects
    - Project members and permissions
    - Project timelines and milestones
3. Task Management
    - Create, update, delete tasks
    - Task assignment to users
    - Task status workflow (Todo, In Progress, Review, Done)
    - Priority levels and deadlines
4. Collaboration
    - Comments and activity logs on tasks/projects
    - @mentions and notifications
    - File uploads for tasks/projects (docs, images, PDFs)
5. Developer Focused Features
    - GitHub/GitLab integration
6. Productivity & Tracking
    - Task search and filters
    - Due-date reminders and email notifications
    - Time tracking per task (manual start/stop)

Backend Requirements (C# .NET 8):

- Build a REST API with .NET 8Core and Entity Framework Core
- Use PostgreSQL
- Implement authentication using JWT
- Use DataAnnotations for input validation
- Apply separation of concerns(e.g., DTOs,services, models)

API Endpoints:

1. Authentication & User Management
    a. Auth
        i. POST /api/auth/register - Register a new user
        ii. POST /api/auth/login - Authenticate user and issue JWT
        iii. GET /api/auth/me - Get logged-in user profile (JWT required)
    b. Roles & Workspaces
        i. POST /api/workspaces - Create a workspace/team
        ii. GET /api/workspaces - List workspaces user belongs to
        iii. POST /api/workspaces/{workspaceId}/members - Add member to workspace (Admin only)
        iv. PUT /api/workspaces/{workspaceId}/members/{userId} - Update role
        v. DELETE /api/workspaces/{workspaceId}/members/{userId} - Remove member
2. Project Management
    a. Projects
        i. GET /api/projects - Get all projects
        ii. POST /api/projects - Create a project
        iii. GET /api/projects/{projectId} - Get project details
        iv. PUT /api/projects/{projectId} - Update project
        v. PATCH /api/projects/{projectId}/archive - Archive project
        vi. DELETE /api/projects/{projectId} - Delete project
    b. Project Members
        i. POST /api/projects/{projectId}/members

    ii. GET /api/projects/{projectId}/members
    iii. PUT /api/projects/{projectId}/members/{userId}
    iv. DELETE /api/projects/{projectId}/members/{userId}
  c. Milestones
    i. POST /api/projects/{projectId}/milestones
    ii. GET /api/projects/{projectId}/milestones
    iii. PUT /api/milestones/{milestoneId}
    iv. DELETE /api/milestones/{milestoneId}
3. Task Management
  a. Tasks
    i. POST /api/projects/{projectId}/tasks
    ii. GET /api/projects/{projectId}/tasks
    iii. GET /api/tasks/{taskId}
    iv. PUT /api/tasks/{taskId}
    v. DELETE /api/tasks/{taskId}
  b. Workflow
    i. PATCH /api/tasks/{taskId}/assign
    ii. PATCH /api/tasks/{taskId}/status
    iii. PATCH /api/tasks/{taskId}/priority
    iv. PATCH /api/tasks/{taskId}/deadline
4. Collaboration
  a. Comments
    i. POST /api/tasks/{taskId}/comments
    ii. GET /api/tasks/{taskId}/comments
  b. Activity
    i. GET /api/projects/{projectId}/activity
  c. Notifications
    i. GET /api/notifications
    ii. PATCH /api/notifications/{notificationId}/read
  d. Files
    i. POST /api/projects/{projectId}/files
    ii. POST /api/tasks/{taskId}/files
    iii. GET /api/files/{fileId}
    iv. DELETE /api/files/{fileId}

5. Developer Features
    a. Git Integration
        i. POST /api/projects/{projectId}/integrations/git
        ii. GET /api/projects/{projectId}/integrations/git
6. Productivity
    a. Search
        i. GET /api/tasks/search
    b. Time Tracking
        i. POST /api/tasks/{taskId}/time/start
        ii. POST /api/tasks/{taskId}/time/stop
        iii. GET /api/tasks/{taskId}/time
    c. Reminders
        i. POST /api/tasks/{taskId}/reminders

Frontend Requirements:

1. Technology Stack Requirements:
    a. **Framework**: React
    b. **Language**: TypeScript
    c. **Styling**: Tailwind CSS
    d. **State Management**: Redux Toolkit
    e. **Routing**: React Router
    f. **API Communication**: Axios
    g. **Auth Handling**: JWT stored securely (HttpOnly cookie or memory)
    h. **Build Tooling**: Vite
    i. **Linting & Formatting**: ESLint + Prettier
2. Authentication & Authorization UI
    a. Features
        i. User **Register** and **Login** screens
        ii. JWT-based session handling
        iii. Protected routes (no token = no entry)
        iv. Auto logout on token expiry
        v. Role-based UI rendering:
            1. **Admin**: full access
            2. **Manager**: project & task control

3. **Member**: task execution only

b. Screens
    i. Login Page
    ii. Register Page
    iii. Unauthorized / Access Denied Page

3. Workspace & Team Management
    a. Features
        i. Create and switch between workspaces
        ii. View workspace members with roles
        iii. Add / remove members (Admin only)
        iv. Update member roles dynamically
    b. Screens
        i. Workspace Selector
        ii. Workspace Settings
        iii. Members Management Panel
4. Project Management UI
    a. Features
        i. Create, edit, archive, and delete projects
        ii. Assign members to projects
        iii. View project timeline and milestones
        iv. Project-level permissions
    b. Screens
        i. Project Dashboard (list + status)
        ii. Project Detail Page
        iii. Project Settings Page
        iv. Milestones View (timeline style)
5. Task Management Interface
    a. Features
        i. Create, update, delete tasks
        ii. Assign tasks to users
        iii. Priority levels (Low, Medium, High)
        iv. Deadline picker
        v. Status workflow: Todo → In Progress → Review → Done

    b. Screens
        i. Task List View
        ii. Task Detail Drawer / Page
6. Collaboration & Communication
    a. Features
        i. Comment system on tasks
        ii. Real-time activity logs
        iii. @mentions highlighting users
        iv. File uploads (docs, images, PDFs)
        v. File preview & download support
    b. Screens
        i. Task Comments Panel
        ii. Project Activity Feed
        iii. File Manager Section
7. Notifications System
    a. Features
        i. In-app notifications list
        ii. Mark notifications as read
        iii. Due-date reminders
        iv. Role-based notification filtering
    b. UI Components
        i. Notification Bell
        ii. Notification Drawer / Page
8. Developer Integrations
    a. Features
        i. GitHub/GitLab integration UI
        ii. Connect repository to project
        iii. Display commit history & links
    b. Screens
        i. Integrations Page
        ii. Repo Connection Modal
9. Productivity & Tracking
    a. Features
        i. Time tracking (start/stop per task)

        ii. Time summary per task
        iii. Global task search
        iv. Filters by:
            1. Status
            2. Priority
            3. Deadline
            4. Assignee
    b. Screens
        i. Search Results Page
        ii. Time Tracking Panel
10. Error Handling & UX Standards
    a. Global error boundary
    b. API error messages mapped cleanly to UI
    c. Loading skeletons (not spinners everywhere)
    d. Empty states with guidance
    e. Responsive design (desktop-first, tablet-ready)
11. Non-Functional Frontend Requirements
    a. Modular component structure
    b. Reusable UI components
    c. Accessibility (ARIA labels, keyboard navigation)
    d. Optimized API calls (debounce search, pagination)
    e. Environment-based configs (dev/prod)
12. Folder Structure
    a. Frontend

```
src/
 ├── api/
 ├── components/
 ├── pages/
 ├── hooks/
 ├── store/
 ├── routes/
 ├── utils/
 ├── styles/
 └── assets/
```

        i.

b. Backend

i.
```
backend/
  ├── Controllers/
  ├── Services/
  ├── Models/
  ├── DTOs/
  ├── Program.cs
  └── appsettings.json
```

13. Security Requirements
    a. No sensitive data in localStorage
    b. JWT refreshed securely
    c. Input validation on frontend (basic sanity checks)
    d. XSS-safe rendering for comments