

# Project Collaboration Hub - Master Technical Documentation

**Date:** October 26, 2023 **Status:** MVP Phase Completed (Auth, Projects, Roles, Issues, Comments, Likes, Public Access)

## 1. Technology Stack

- **Frontend:** React.js, Tailwind CSS, Context API, React Router v6
- **Backend:** Node.js, Express.js
- **Database:** PostgreSQL
- **Authentication:** JWT (JSON Web Tokens), bcryptjs

## 2. Database Schema (PostgreSQL)

Bu SQL komutları, projenin veritabanı yapısının son halidir.

```
-- UUID eklentisi
CREATE EXTENSION IF NOT EXISTS "uuid-oss";

-- Users Table (Kullanıcılar)
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    username VARCHAR(100) NOT NULL UNIQUE,
    email VARCHAR(255) UNIQUE, -- Sonradan eklendi
    password_hash VARCHAR(255) NOT NULL,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

-- Projects Table (Projeler)
CREATE TABLE projects (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    owner_id UUID NOT NULL REFERENCES users(id),
    is_public BOOLEAN DEFAULT false NOT NULL, -- Public/Private durumu
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    last_updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

-- Project Members Table (Proje Üyeleri)
CREATE TABLE project_members (
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    role VARCHAR(50) NOT NULL DEFAULT 'viewer', -- 'owner', 'editor', 'viewer'
    joined_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (project_id, user_id)
);

-- Issues Table (Sorunlar/Görevler)
CREATE TABLE project_issues (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    created_by_id UUID NOT NULL REFERENCES users(id),
    text TEXT NOT NULL,
    status VARCHAR(50) DEFAULT 'Open', -- 'Open', 'Closed'
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);
```

```

-- Comments Table (Yorumlar - Hem Proje hem Issue için)
CREATE TABLE comments (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    issue_id UUID REFERENCES project_issues(id) ON DELETE CASCADE, -- NULL ise Proje yorumu
    author_id UUID NOT NULL REFERENCES users(id),
    text TEXT NOT NULL,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    likes_user_ids UUID[] DEFAULT ARRAY[]::UUID[] -- Beğenenlerin listesi
);

-- Trigger for last_updated_at (Otomatik Güncelleme)
CREATE OR REPLACE FUNCTION trigger_set_timestamp()
RETURNS TRIGGER AS $$
BEGIN
    NEW.last_updated_at = NOW();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER set_project_timestamp
BEFORE UPDATE ON projects
FOR EACH ROW
EXECUTE FUNCTION trigger_set_timestamp();

```

### 3. Backend Architecture & Rules

#### A. Middleware ( checkProjectMember )

Projelere erişimi kontrol eden "Kapı Bekçisi".

1. Proje **Public (is\_public=true)** mi? -> Evet ise, giriş yapan herkes `public_viewer` rolüyle girebilir.
2. Kullanıcı **Üye ( project\_members )** mi? -> Evet ise, veritabanındaki rolü ( `owner` , `editor` , `viewer` ) ile girebilir.
3. İkisi de değilse -> `403 Access Denied` .

#### B. Role Permissions Matrix (Yetki Tablosu)

Action	Owner	Editor	Viewer	Public Viewer
View Project	✓	✓	✓	✓
Create Issue	✓	✓	✗	✗
Edit/Close Issue	✓	✓	✗	✗
Comment on Project	✓	✓	✓	✓
Comment on Issue	✓	✓	✓	✓
Like Comment	✓	✓	✓	✓

Edit/Delete Own Comment	✓	✓	✓	✓
Delete Any Comment	✓	✗	✗	✗
Invite Member	✓	✓	✗	✗
Remove Member	✓	✗	✗	✗
Change Visibility	✓	✗	✗	✗
Delete Project	✓	✗	✗	✗

### C. Key API Routes

- GET /api/projects/my-projects : Sadece sahip olduğum projeler.
- GET /api/projects/shared-projects : Paylaşılanlar (üye olup sahibi olmadığım) + Public olanlar.
- POST /api/projects/:id/members : Üye ekleme (Invite).
- DELETE /api/projects/:id/members/:userId : Üye çıkarma.
- POST /api/projects/:id/issues/:issueId/comments : Issue yorumu ekleme (Issue "Closed" ise engeller).

## 4. Frontend Architecture

### A. Context API

- AuthContext : Kullanıcı giriş/çıkış işlemlerini yönetir. localStorage kullanarak oturumu hatırlar.
- ProjectContext : Proje verilerini, üyeleri, issue'ları ve yorumları yönetir. Backend ile tüm iletişimi ( axios ) burası kurar.

### B. Page Structure

- LoginPage / RegisterPage : Giriş ve Kayıt (Public Routes).
- DashboardLayout : Sidebar ve Header'ı içeren ana iskelet (Protected Route).
  - DashboardOverviewPage : Proje listeleri (My Projects).
  - SharedProjectsPage : Paylaşılan projeler listesi.
  - ProjectDetailPage : Tekil proje sayfası. 4 Sekmeli yapı:
    1. [Files] : Dosyalar (Default).
    2. [Issues] : Görev takibi.
    3. [Discussions] : Sohbet odası (Üyelere özel).
    4. [Edit] : Ayarlar (Sadece Owner görür).

### C. Key Components

- RightSidebar :
  - Dashboard'da: İstatistikleri gösterir.

- Proje Sayfasında: Proje bilgilerini ve Üye Listesini gösterir.
- IssueItem : Tek bir sorunu gösterir. Genişletilebilir (yorumları gösterir). 3-nokta menüsü (Edit/Done) vardır.
- Comment : Tek bir yorumu gösterir. Beğeni, Kopyalama, Düzenleme ve Silme özelliklerine sahiptir.
- ProjectDiscussion : Sohbet odası tarzı arayüz.

## 5. User Stories (Summary)

### Student:

- Can register, login, logout.
- Can create private projects.
- Can invite members (Editor/Viewer) via username.
- Can make projects Public.
- Can track tasks via Issues (Create, Comment, Close).
- Can chat in Discussions.

### Faculty Advisor:

- Can view student projects (Read-Only).
- Can leave feedback comments.

### Admin:

- Can manage users and projects system-wide.

**\*\*Sıradaki Adımlar (Öneri):\*\***

Eğer projeye devam edecekseniz, bir sonraki oturumda bu dosyayı bana vererek doğrudan **\*\*"Do**

Harika bir iş çıkardınız! 🚀