21.05.2025

Görev 3: Dosya Yükleme ve Yönetimi Uygulaması

EMİR KANAT emirkanat49gmail.com

İçindekiler

[**1. Proje Tanımı** 1](#_Toc198762756)

[**2.2. Proje Yapısı** 2](#_Toc198762757)

[**3. Backend Geliştirme Aşamaları** 3](#_Toc198762758)

[**4. Frontend Geliştirme Aşamaları** 3](#_Toc198762759)

[**5.Çalıştırma ve Test** 4](#_Toc198762760)

# **1. Proje Tanımı**

Bu proje, **MERN Stack** (MongoDB, Express.js, React, Node.js) teknolojileri kullanılarak geliştirilen bir **dosya yönetim uygulamasıdır**. Uygulamanın temel amacı, kullanıcıların **PDF, PNG ve JPG** formatındaki dosyalarını sisteme yükleyebilmelerini, yüklenen dosyaları listeleyebilmelerini ve gerektiğinde silebilmelerini sağlamaktır.

**Uygulamanın Temel İşlevleri:**

1. **Dosya Yükleme:**
   * Kullanıcılar, **10MB'a kadar** olan PDF, PNG veya JPG dosyalarını sisteme yükleyebilir.
   * Yüklenen dosyalar, **sunucuda fiziksel olarak** (uploads/ klasörü) ve **MongoDB veritabanında** kaydedilir.
2. **Dosya Listeleme:**
   * Yüklenen tüm dosyalar, kullanıcı dostu bir arayüzde listelenir.
   * Her dosya için **dosya adı, türü, boyutu ve yükleme tarihi** görüntülenir.
   * Kullanıcılar, dosyaları **tıklayarak görüntüleyebilir** (PDF'ler tarayıcıda açılır, görseller önizlenir).
3. **Dosya Silme:**
   * Kullanıcılar, istedikleri dosyayı listeden silebilir.
   * Silme işlemi hem **veritabanından** hem de **fiziksel dosya sisteminden** gerçekleştirilir.
4. **Hata Yönetimi:**
   * Geçersiz dosya türü veya boyut aşımı durumunda kullanıcıya **anında hata mesajı** gösterilir.
   * Sunucu kaynaklı hatalar (veritabanı bağlantısı kesintisi gibi) kullanıcıya **anlaşılır bir şekilde** iletilir.

**Teknik Hedefler:**

* **Full-Stack Mimari:**
  + **Frontend:** React.js ile dinamik ve kullanıcı dostu bir arayüz.
  + **Backend:** Node.js + Express.js ile RESTful API geliştirilmesi.
  + **Veritabanı:** MongoDB ile dosya meta verilerinin saklanması.
* **Dosya Yönetimi:**
  + **Multer** ile dosya yükleme işlemlerinin yönetimi.
  + **FS Modülü** ile fiziksel dosya silme işlemleri.
* **Güvenlik:**
  + **CORS** ayarları ile sadece belirlenen kaynaklardan gelen isteklere izin verilmesi.
  + Dosya türü ve boyutu kontrolleri ile kötü niyetli yüklemelerin engellenmesi

# **2.2. Proje Yapısı**

Projenin genel seması ve yapısı



# **3. Backend Geliştirme Aşamaları**

**3.1. Proje Kurulumu**

* express, mongoose, multer, cors ve dotenv paketleri yüklendi.
* nodemon geliştirme bağımlılığı olarak eklendi.

**3.2. Temel Sunucu Yapısı**

* **Dosya:** server.js
  + MongoDB bağlantısı (mongoose.connect).
  + Middleware'ler: cors(), express.json(), statik dosya servisi (uploads dizini).
  + API rotaları: /api/files üzerinden dosya işlemleri.

**3.3. Veritabanı Modeli**

* **Dosya:** models/File.js
  + filename, originalname, mimetype, size, path ve uploadDate alanları tanımlandı.

**3.4. Dosya Yönetimi**

* **Multer Ayarları:**
  + Dosyalar uploads/ dizinine kaydedilir.
  + Dosya türü filtresi (fileFilter): Yalnızca JPEG, PNG ve PDF kabul edilir.
  + Boyut sınırı: 10MB.

**3.5. API Endpoint'leri**

* **POST**/upload**:** Dosya yükleme (Multer middleware ile).
* **GET**/**:** Tüm dosyaları listeleme.
* **DELETE**/:id**:** Dosya silme (hem veritabanından hem fiziksel olarak).

# **4. Frontend Geliştirme Aşamaları**

**4.1. React Projesi Kurulumu**

* create-react-app ile başlatıldı.
* axios (API istekleri) ve react-bootstrap (UI) eklendi.

**4.2. Ana Bileşenler**

1. **App.js**
   * State yönetimi: files, loading, error.
   * API fonksiyonları: fetchFiles, handleFileUpload, handleDeleteFile.
2. **FileUploadForm.js**
   * Dosya seçme ve validasyon (tür/boyut kontrolü).
   * Form gönderimi: axios.post ile multipart/form-data.
3. **FileList.js**
   * Tablo formatında dosya listesi.
   * Dosya önizleme linkleri (http://localhost:5000/uploads/[dosya-adı]).
   * Silme butonu ile axios.delete çağrısı.

**4.3. Kullanıcı Arayüzü**

* **Bootstrap** ile responsive tasarım.
* Hata mesajları (Alert), yükleme göstergesi (Spinner).

# **5.Çalıştırma ve Test**

5.1. Backend



5.2. Frontend

