T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

DERS :
YAZILIM TEST VE KALİTE
PROJE:
KÜTÜPHANE SİSTEMİ

ÖĞRENCİ ADI SILA SERDAR
ÖĞRENCİ NUMARASI 230502014

ÖĞRENCİ ADI MUHAMMED EMİR SARI
ÖĞRENCİ NUMARASI 230502054

ÖĞRENCİ ADI MURAT KAAN TEKELİ
ÖĞRENCİ NUMARASI 230502007

ÖĞRENCİ ADI MİRAY YALÇIN
ÖĞRENCİ NUMARASI 230502033

DERS SORUMLUSU:
PROF. DR./DR. ÖĞR. ÜYESİ
ELİF PINAR HACIBEYOĞLU

09.11.2025

İÇİNDEKİLER

1. GİRİŞ

1.1. Projenin Amacı Bu projenin temel amacı, geleneksel kütüphane süreçlerini dijitalleştirerek, kitap takibi, ödünç alma/iade işlemleri ve envanter yönetimini modern bir web tabanlı platform üzerinden gerçekleştirmektir. Ayrıca, yapay zeka (AI) destekli algoritmalar kullanılarak okuyuculara kişiselleştirilmiş kitap önerileri sunulması ve kütüphane personelinin iş yükünün hafifletilmesi hedeflenmiştir.

1.2. Projenin Önemi ve Kapsamı Günümüzde bilgiye hızlı erişim ve fiziksel kaynakların verimli yönetimi büyük önem taşımaktadır. Geliştirilen bu sistem, kitapların fiziksel raf konumlarını (Örn: A-3-2) nokta atışı belirleyerek kayıp zamanı en aza indirir. Müşteriler ve çalışanlar için ayrılmış özel paneller sayesinde, yetkilendirilmiş erişim ve veri güvenliği sağlanır. Proje kapsamı; kullanıcı kimlik doğrulama, kitap envanter yönetimi, kiralama süreçleri, satın alım öneri sistemi ve AI tabanlı öneri modülünü içermektedir.

2. PROJE YÖNETİMİ

2.1. Zaman Çizelgesi Projenin geliştirme süreci 4 haftalık bir zaman dilimine yayılmıştır.

hafta	Yapılan işlem	açıklama
1.hafta	Analiz Ve Tasarım	Gereksinimlerin belirlenmesi, veritabanı şemasının (ER) çıkarılması.
2.hafta	Backend Geliştirme	API uç noktalarının yazılması, veritabanı bağlantılarının kurulması.
3.hafta	Frontend Geliştirme	Arayüz tasarımı, React bileşenlerinin oluşturulması ve API entegrasyonu.
4.hafta	Test Ve Dokümantasyon	Sistem testlerinin yapılması, hataların giderilmesi ve raporlama.

2.2. Görev Dağılımı

Proje geliştirme süreci, uzmanlık alanlarına göre ayrılmış bir ekip tarafından yürütülmektedir. Proje ekibinin rol dağılımı ve sorumlulukları aşağıda detaylandırılmıştır:

- **Proje Yöneticisi (Murat Kaan Tekeli):** Proje planlamasının yapılması, ekip içi koordinasyonun sağlanması, gerekli kaynakların belirlenmesi ve süreç kontrolünün yürütülmesinden sorumludur.
- **Yazılım Geliştiriciler (Miray Yalçın, Sıla Serdar):** Uygulamanın hem sunucu tarafı (backend) hem de ön yüz (frontend) geliştirmelerini gerçekleştirmek, veri tabanı yapısını kurmak ve kullanıcı arayüzü ile sistemin uyumlu çalışmasını sağlamakla görevlidirler.
- **Test ve Dokümantasyon (Muhammed Emir Sarı):** Geliştirilen fonksiyonların testlerini gerçekleştirmek, test senaryoları hazırlamak, hata raporlamalarını yapmak ve projenin teknik dokümantasyonunu düzenlemekten sorumludur.

3. GEREKSİNİMLER ANALİZİ

Projenin başarıya ulaşması için belirlenen fonksiyonel ve fonksiyonel olmayan gereksinimler aşağıdaki tabloda sunulmuştur.

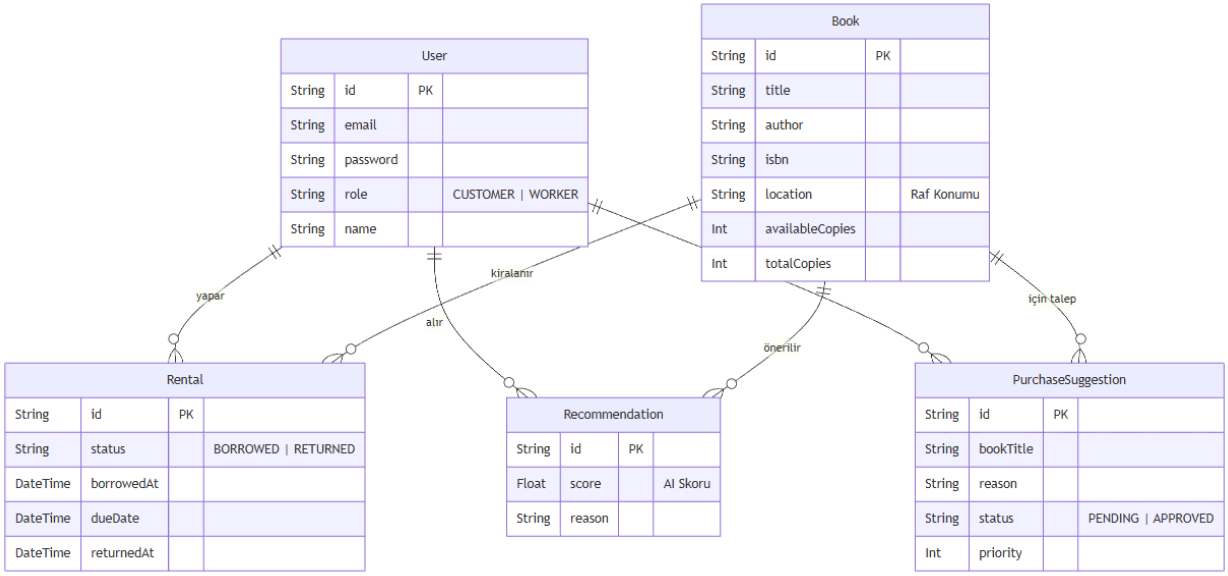
Gereksinim No	Gereksinim Türü	Açıklama
F-01	Fonksiyonel	Kullanıcılar (Müşteri/Çalışan) e-posta ve şifre ile sisteme giriş yapabilmelidir.
F-02	Fonksiyonel	Çalışanlar sisteme yeni kitap ekleyebilmeli ve konum bilgisi girebilmelidir.
F-03	Fonksiyonel	Müşteriler kitap ismine, yazarına veya ISBN numarasına göre arama yapabilmelidir.
F-04	Fonksiyonel	Sistem, kullanıcının geçmiş okumalarına dayalı kitap önerileri sunmalıdır.
F-05	Fonksiyonel	Çalışanlar kitap kiralama ve iade işlemlerini yönetebilmelidir.
NF-01	Fonksiyonel Olmayan	Kullanıcı şifreleri veritabanında şifreli (hashed) olarak saklanmalıdır.
NF-02	Fonksiyonel Olmayan	Arayüz responsive (mobil uyumlu) olmalıdır.

4. SİSTEM TASARIMI VE MİMARİ

4.1. Kullanılan Teknolojiler Proje, performans ve modern yazılım standartları gözetilerek "Full Stack" bir mimari ile geliştirilmiştir.

- **Backend:** Bun Runtime üzerinde ElysiaJS Framework. Yüksek performans ve düşük gecikme süresi sağladığı için tercih edilmiştir.
- **Frontend:** React (Vite) ve Tailwind CSS. Hızlı arayüz geliştirme ve kullanıcı deneyimi için kullanılmıştır.
- **Veritabanı:** PostgreSQL. Veri bütünlüğü ve ilişkisel yapı (Relational Database) gereksinimi nedeniyle seçilmiştir.
- **ORM:** Prisma. Tip güvenliği (Type-safety) ve veritabanı şema yönetimi için kullanılmıştır.

4.2. Veritabanı Tasarımı Sistemin veritabanı; Kullanıcılar, Kitaplar, Kiralamalar, Öneriler ve Satın Alım Talepleri olmak üzere 5 ana tablodan oluşmaktadır.



5. UYGULAMA VE KODLAMA

Bu bölümde, projenin temel fonksiyonlarını gerçekleştiren kritik kod bloklarına ve açıklamalarına yer verilmiştir.

5.1. Backend (Sunucu) Kodlaması

Aşağıdaki kod bloğu, veritabanı şemasını tanımlayan `schema.prisma` dosyasından alınmıştır. Burada kitapların konumu ve stok durumlarının nasıl tutulduğu görülmekte

```
model Book {
  id              String   @id @default(cuid())
  title           String
  author          String
  isbn            String?  @unique
  description     String?
  category        String?
  publishedYear   Int?
  totalCopies     Int      @default(1)
  availableCopies Int      @default(1)
  location        String   // Rafta bulunan konum (örn: "A-3-2" = A bölümü, 3. raf, 2. sıra)
  imageUrl        String?
  createdAt       DateTime @default(now())
  updatedAt       DateTime @updatedAt

  rentals         Rental[]
  recommendations Recommendation[]
  purchaseSuggestions PurchaseSuggestion[]

  @@map("books")
}
```

Aşağıdaki blok ise yapay zeka destekli öneri algoritmasının mantığını göstermektedir. Sistem, kullanıcının daha önce okuduğu kategorilere ve yazarlara göre bir skorlama (0 ile 1 arasında) yapmaktadır.

```
// Tablo 4: AI Öneri Algoritması (index.ts)
// Önerileri skorla ve neden ekle
const recommendationsWithScore = await Promise.all(
  recommendations.map(async (book) => {
    let score = 0.5; // Taban puan
    let reason = "";

    if (readCategories.includes(book.category || "")) {
      score += 0.3; // Kategori eşleşmesi
      reason += `${book.category} kategorisinden kitaplar okumuşsunuz. `;
    }

    if (readAuthors.includes(book.author)) {
      score += 0.2; // Yazar eşleşmesi
      reason += `${book.author} yazarından kitap okumuşsunuz. `;
    }

    return { ...book, score: Math.min(score, 1.0), reason };
  })
);
```

5.2. Frontend (Arayüz) Kodlaması

Müşterilerin kitap aradığında anlık sonuç alabildiği ve kitabın raf konumunu görebildiği arayüz kodu `WorkerDashboard.tsx` içerisinde aşağıdaki gibi kurgulanmıştır.

```
// Tablo 5: Kitap Arama Fonksiyonu (WorkerDashboard.tsx)
const handleSearch = async () => {
  if (!search.trim()) return
  setLoading(true)
  try {
    const response = await api.get(
      `/api/books/search/location?search=${encodeURIComponent(search)}`,
      token
    )
    setBooks(response.books)
  } catch (error: any) {
    console.error('Arama hatası:', error)
  } finally {
    setLoading(false)
  }
}
```

6. TESTLER VE SONUÇLAR

Proje geliştirme sürecinde aşağıdaki test senaryoları uygulanmış ve başarıyla sonuçlanmıştır:

1. **Kimlik Doğrulama Testi:** Yanlış şifre ile girişte sistem hata mesajı vermiş, doğru girişte JWT token üretilmiştir.
2. **Kiralama Testi:** Stokta olmayan (0 adet) bir kitap kiralanmak istendiğinde sistemin işlemi engellediği görülmüştür.
3. **Öneri Testi:** "Roman" kategorisinde kitap okuyan bir kullanıcıya, sistemin "Roman" kategorisindeki diğer kitapları önerdiği doğrulanmıştır.
4. **Konum Testi:** Arama sonuçlarında kitabın raf bilgisinin (Örn: A-1-1) doğru şekilde listelendiği teyit edilmiştir.

7. KARŞILAŞILAN ZORLUKLAR VE ÇÖZÜMLER

Proje sürecinde karşılaşılan temel zorluklar ve üretilen çözümler şunlardır:

- **Zorluk:** Yapay zeka öneri algoritmasının, soğuk başlangıç (cold start - hiç kitap okumamış kullanıcı) durumunda nasıl davranacağı sorunu.
    - **Çözüm:** Hiç okuması olmayan kullanıcılara "En Çok Okunanlar" veya rastgele popüler kitapların önerilmesi mantığı yerine, başlangıçta sistemin boş liste döndürmesi ve kullanıcının etkileşime geçtikçe öneri alması sağlanmıştır.
- **Zorluk:** Kitapların raf konumlarının standart bir formatta tutulması.
    - **Çözüm:** Konum verisi için "Blok-Raf-Sıra" (A-3-2) formatı zorunlu kılınarak standartizasyon sağlanmıştır.

8. SONUÇ VE ÖNERİLER

8.1. Projenin Geleceği Bu proje, temel kütüphane işlevlerini başarıyla yerine getirmektedir. Ancak projenin gelecekte geliştirilmesi ve ticarileşmesi adına şu eklemeler yapılabilir:

- **Mobil Uygulama Entegrasyonu:** React Native kullanılarak sistemin bir mobil uygulamaya dönüştürülmesi ve kullanıcıların telefonlarından karekod (QR) ile kitap ödünç alabilmesi.
- **Gelişmiş Yapay Zeka:** Mevcut kural tabanlı öneri sistemi yerine, "Collaborative Filtering" (İşbirlikçi Filtreleme) kullanan ve benzer zevklere sahip kullanıcıları eşleştiren bir makine öğrenmesi modeline geçilmesi.
- **Ceza Sistemi:** İadesi geciken kitaplar için otomatik ceza hesaplama ve ödeme ağ geçidi (Payment Gateway) entegrasyonu.
- **Dış Kütüphanelerle Bağlantı:** Kütüphaneler arası ödünç alma protokolünün eklenmesi.

Sonuç olarak, geliştirilen Kütüphane Yönetim Sistemi, modern web teknolojilerinin gücünü kullanarak kütüphane operasyonlarını hızlandırmış ve kullanıcı deneyimini iyileştirmiştir.

KAYNAKÇA

1. **React Documentation.** (2024). *React: The library for web and native user interfaces*. Meta Platforms. Erişim adresi: https://react.dev/reference/react
2. **Prisma Data.** (2024). *Prisma ORM Documentation: Next-generation Node.js and TypeScript ORM*. Erişim adresi: https://www.prisma.io/docs
3. **ElysiaJS.** (2024). *Elysia: Ergonomic Framework for Humans*. Erişim adresi: https://elysiajs.com/
4. **Oven-sh.** (2024). *Bun: A fast all-in-one JavaScript runtime*. Erişim adresi: https://bun.sh/docs
5. **The PostgreSQL Global Development Group.** (2023). *PostgreSQL 15.0 Documentation*. Erişim adresi: https://www.postgresql.org/docs/15/index.html
6. **Tailwind Labs.** (2024). *Tailwind CSS: Rapidly build modern websites without ever leaving your HTML*. Erişim adresi: https://tailwindcss.com/docs
7. **Docker Inc.** (2024). *Docker Documentation: Containerization technology*. Erişim adresi: https://docs.docker.com/
8. **Vite.** (2024). *Vite: Next Generation Frontend Tooling*. Erişim adresi: https://vitejs.dev/guide/