



KOCAELİ ÜNİVERSİTESİ

Mühendislik Fakültesi
Yazılım Mühendisliği Bölümü

Java Programlama

ve

Veritabanı Yönetim Sistemleri

PROJE RAPORU

Kütüphane Yönetim ve Öneri Sistemi

Ad - Soyad: Sena Nur Dülger **No:** 220229053

Ad - Soyad: Irmak Yılmaz **No:** 230229038

Ad - Soyad: Zehra Gül Özdemir **No:** 230229068

Tarih: 8 Ocak 2026

İçindekiler

| | |
|--|-----------|
| 1. Giriş | 2 |
| 1.1. Projenin Amacı ve Önemi | 2 |
| 1.2. Projenin Kapsamı | 2 |
| 1.3. Kullanılan Teknolojiler | 2 |
| 2. Sistem Analizi ve Gereksinimler | 3 |
| 2.1. Fonksiyonel Gereksinimler | 3 |
| 2.2. Veritabanı Gereksinimleri | 3 |
| 3. Veritabanı Tasarımı | 3 |
| 3.1. ER Diyagramı (Entity–Relationship Diagram) | 3 |
| 3.2. Veritabanı Şeması ve Normalizasyon | 4 |
| 3.3. Veritabanı Tetikleyicileri (Triggers) | 5 |
| 4. Yazılım Mimarisi | 6 |
| 4.1. Katmanlı Mimari (Layered Architecture) | 6 |
| 4.2. Nesne Yönelimli Programlama (OOP) Yaklaşımı | 6 |
| 4.3. RESTful API Prensipleri ve HTTP Metotları | 6 |
| 5. Algoritmalar ve İş Mantığı | 7 |
| 5.1. Ödünç Alma ve Dinamik Envanter Yönetimi | 7 |
| 5.2. Gecikme Cezası Hesaplama Algoritması | 7 |
| 5.3. Kişiselleştirilmiş Öneri Algoritması | 7 |
| 6. Kullanıcı Arayüzü (UI) Geliştirmeleri | 8 |
| 6.1. Ortak Sayfalar | 8 |
| 6.2. Kullanıcı Sayfaları | 8 |
| 6.3. Admin Sayfaları | 9 |
| 7. Sonuç ve Değerlendirme | 9 |
| 8. Grup İçi İş Bölümü ve Sorumluluklar | 9 |
| 8.1. Sorumluluklar | 10 |
| 8.2. Proje Yönetimi ve Sürüm Kontrolü | 10 |
| 9. Kaynakça | 11 |

1. Giriş

1.1. Projenin Amacı ve Önemi

Bu proje, kütüphanelerde manuel olarak yürütülen yönetim süreçlerini dijitalleştirerek veri tutarlığını artırmayı, işlem sürelerini kısaltmayı ve kullanıcı deneyimini iyileştirmeyi amaçlamaktadır. Geleneksel yöntemler; kayıt tekrarları, işlem hataları ve verimsizlik gibi sorunlara yol açabilmektedir. Geliştirilen kütüphane yönetim sistemi, bu sorunlara modern yazılım teknolojileri kullanılarak çözüm sunmaktadır.

Projenin temel hedefleri şu şekilde özetlenebilir:

- Veri Yönetimi ve Bütünlüğü:** Kütüphane envanteri, üye bilgileri ve işlem kayıtlarının ilişkisel veritabanında, normalizasyon kurallarına (3NF) uygun şekilde saklanması.
- Süreç Otomasyonu:** Ödünç alma, iade, stok güncelleme ve gecikmeye bağlı ceza hesaplama işlemlerinin otomatik olarak gerçekleştirilmesi.
- Kullanıcı Odaklı Yaklaşım:** Kullanıcıların geçmiş işlemlerine dayalı olarak kişiselleştirilmiş kitap önerilerinin sunulması.

1.2. Projenin Kapsamı

Bu proje kapsamında geliştirilen sistem aşağıdaki temel işlevleri içermektedir:

- Üye kayıt ve yönetim işlemleri
- Kitap ve envanter yönetimi
- Ödünç alma, iade ve ceza hesaplama süreçleri
- Kullanıcıya özel kitap öneri mekanizması

Bu modüller, kütüphane süreçlerinin bütüncül ve etkin bir şekilde yönetilmesini sağlamaktadır.

1.3. Kullanılan Teknolojiler

| Katman | Teknoloji | Açıklama |
|------------|----------------------------|--|
| Backend | Java 17, Spring Boot 3.5.9 | RESTful API geliştimi ve JPA/Hibernate veri yönetimi. |
| Veritabanı | PostgreSQL | İlişkisel veri yapısı, Trigger ve Stored Procedure desteği. |
| Frontend | HTML5, CSS3, JavaScript | Modern UI tasarımları ve fetch API ile asenkron veri iletişim. |
| Araçlar | GitHub, Postman, VS Code | Sürüm kontrolü ve API test süreçleri. |
| Yapay Zeka | Gemini, ChatGPT | Kod optimizasyonu, hata ayıklama ve raporlama süreçleri. |

2. Sistem Analizi ve Gereksinimler

2.1. Fonksiyonel Gereksinimler

Sistem, kullanıcı ve yönetici rollerine yönelik aşağıdaki temel fonksiyonel gereksinimleri karşılamaktadır:

- **Kitap Arama ve Filtreleme:** Kullanıcılar, kitaplara başlık, yazar veya kategori bilgilerine göre arayabilmektedir.
- **Ödünç Alma ve İade Süreci:** Müsait durumda kitaplar ödünç alınabilmekte, iade işlemi sonrası stok bilgileri otomatik olarak güncellenmektedir.
- **Gecikme Cezası Hesaplama:** İade süresi aşıldığında, gecikme süresi sistem tarafından hesaplanarak günlük ücret üzerinden ceza uygulanmaktadır.
- **Kullanıcıya Özel Öneriler:** Kullanıcının geçmiş ödünç alma verileri analiz edilerek, ilgili kategorilerden kitap önerileri sunulmaktadır.

2.2. Veritabanı Gereksinimleri

Veri bütünlüğünün sağlanması amacıyla aşağıdaki veritabanı prensipleri uygulanmıştır:

- **İlişkisel Yapı:** Loans tablosu, Members ve Books tabloları arasında ilişki kuran merkezi bir yapı olarak tasarlanmıştır.
- **Normalizasyon:** Yazar, kategori ve kitap bilgileri veri tekrarını önlemek amacıyla üçüncü normal forma (3NF) uygun şekilde ayrıstırılmıştır.
- **Trigger Kullanımı:** updated_at alanlarının ve stok tutarlığının korunması veritabanı tetikleyicileri ile sağlanmıştır.
- **Geçici Alanlar:** overdue ve remainingDays gibi hesaplanan alanlar, veritabanına kaydedilmeden çalışma anında oluşturulmaktadır.

3. Veritabanı Tasarımı

Sistemin veritabanı şeması, PostgreSQL üzerinde ilişkisel bütünlüğü sağlayacak şekilde tasarlanmıştır. Her tablo birincil anahtarlar (PK) ile benzersiz kılınmış, tablolar arasındaki ilişkiler yabancı anahtarlar (FK) aracılığıyla kurulmuştur.

3.1. ER Diyagramı (Entity–Relationship Diagram)

Temel ilişkiler aşağıdaki gibidir:

- **Çoktan-Bire (Many-to-One):** Birden fazla kitap, tek bir yazar ve kategori ile ilişkilendirilerek veri tekrarı önlenmiş ve yönetim merkezi hâle getirilmiştir.
- **Bire-Çok (One-to-Many):** Her kitap, farklı şubelere ait birden fazla envanter kaydı ile ilişkilendirilmiş; böylece stoklar şube bazında takip edilebilir hâle getirilmiştir.

- **Çoktan-Çoğa (Many-to-Many):** Üye ve kitap arasındaki ödünç alma ilişkisi, Loan ara tablosu ile modellenmiş; ödünç ve iade işlemleri ile ceza bilgileri bu yapı üzerinden yönetilmiştir.

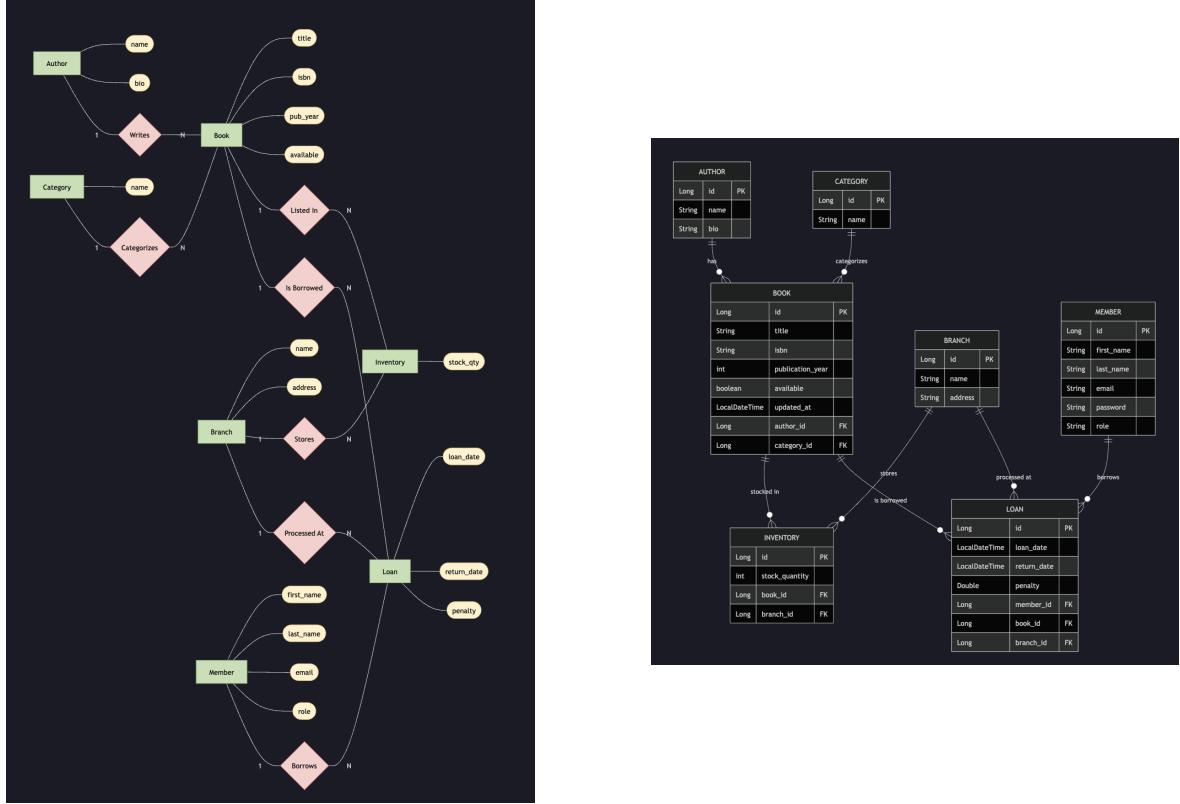


Figure 1: Kütüphane Yönetim Sistemi için ER Diyagramları

3.2. Veritabanı Şeması ve Normalizasyon

```

database > # schema.sql
1  -- Kütüphane Yönetim Sistemi Veritabanı Şeması
2
3  -- 1. Yazarlar (Authors) Tablosu
4  CREATE TABLE IF NOT EXISTS authors (
5    id BIGSERIAL PRIMARY KEY,
6    name VARCHAR(255) NOT NULL,
7    bio VARCHAR(1000)
8  );
9
10 -- 2. Kategoriler (Categories) Tablosu
11 CREATE TABLE IF NOT EXISTS categories (
12   id BIGSERIAL PRIMARY KEY,
13   name VARCHAR(255) UNIQUE NOT NULL
14 );
15
16 -- 3. Kitaplar (Books) Tablosu
17 CREATE TABLE IF NOT EXISTS books (
18   id BIGSERIAL PRIMARY KEY,
19   title VARCHAR(255) NOT NULL,
20   isbn VARCHAR(255) UNIQUE,
21   publication_year INTEGER,
22
23   -- İllükt Sütunları
24   author_id BIGINT NOT NULL,
25   category_id BIGINT,
26
27   -- Durum
28   available BOOLEAN DEFAULT TRUE,
29   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
30
31   -- Foreign Keys
32   CONSTRAINT fk_book_author FOREIGN KEY (author_id) REFERENCES authors(id),
33   CONSTRAINT fk_book_category FOREIGN KEY (category_id) REFERENCES categories(id)
34 );
35
36 -- 4. Şubeler (Branches) Tablosu
37 CREATE TABLE IF NOT EXISTS branches (
38   id BIGSERIAL PRIMARY KEY,
39   name VARCHAR(255) NOT NULL,
40   address VARCHAR(255)
41 );
42
43 -- 5. Üyeler (Members) Tablosu
44 CREATE TABLE IF NOT EXISTS members (
45   id BIGSERIAL PRIMARY KEY,
46   first_name VARCHAR(255) NOT NULL,
47   last_name VARCHAR(255) NOT NULL,
48   email VARCHAR(255) UNIQUE NOT NULL,
49   password VARCHAR(255) NOT NULL,
50   role VARCHAR(50) DEFAULT 'MEMBER', -- 'ADMIN' veya 'MEMBER'
51 );
52
53 -- 6. Ödünç İşlemleri (Loans) Tablosu
54 CREATE TABLE IF NOT EXISTS loans (
55   id BIGSERIAL PRIMARY KEY,
56   member_id BIGINT NOT NULL,
57   book_id BIGINT NOT NULL,
58   loan_date DATE NOT NULL,
59   return_date DATE,
60   penalty DECIMAL(10, 2) DEFAULT 0.0,
61
62   CONSTRAINT fk_loan_member FOREIGN KEY (member_id) REFERENCES members(id),
63   CONSTRAINT fk_loan_book FOREIGN KEY (book_id) REFERENCES books(id)
64 );
65
66 -- 7. Stoklar (Inventory) Tablosu
67 CREATE TABLE IF NOT EXISTS inventory (
68   id BIGSERIAL PRIMARY KEY,
69   book_id BIGINT NOT NULL UNIQUE,
70   stock_quantity INTEGER NOT NULL DEFAULT 0,
71
72   CONSTRAINT fk_inventory_book FOREIGN KEY (book_id) REFERENCES books(id)
73 );

```

Figure 2: Veritabanı şemasına ait kısıtler (schema.sql)

Sistem, veri tekrarını azaltmak ve güncelleme anomalilerini önlemek amacıyla 3NF kurallarına göre tasarlanmıştır. Her tablo tek birincil anahtara sahiptir ve tüm alanlar bu anahtara tam bağımlıdır.

Tabloların temel görevleri aşağıda özetlenmiştir:

- **Member:** Üyelerin kimlik, iletişim ve rol (ADMIN/MEMBER) bilgilerini saklar.
- **Book:** Kitaba ait statik bilgileri ve ödünç verilebilirlik durumunu tutar.
- **Loan:** Ödünç alma ve iade süreçlerini, gecikme durumlarını ve ceza bilgilerini yönetir.
- **Inventory:** Kitap stok bilgisini bağımsız olarak yönetir.
- **Branch:** Kütüphane şubelerini tanımlar.
- **Author:** Yazar bilgilerini merkezi ve tutarlı biçimde saklar.
- **Category:** Kitap türlerini benzersiz olarak depolar.

Yazar ve kategori bilgilerinin **Book** tablosundan ayrılarak yabancı anahtarlar aracılığıyla ilişkilendirilmesi, veri tutarlığını artırmış ve 3NF seviyesinde bir yapı elde edilmesini sağlamıştır.

3.3. Veritabanı Tetikleyicileri (Triggers)

Veri bütünlüğünü otomatik olarak korumak amacıyla veritabanı seviyesinde tetikleyiciler kullanılmıştır. Kitap tablolarında yapılan güncellemelerde, `update_updated_at_column` fonksiyonu sayesinde `updated_at` alanı otomatik olarak güncellenmektedir.

```
database > E triggers.sql
 1  -- OTOMATİK TARİH GÜNCELLEME TETİKLEYİCİSİ
 2
 3  -- 1. Fonksiyonu Tanımla
 4  -- Bu fonksiyon, bir satır güncellendiğinde 'updated_at' sütununa şimdiki zamanı basar.
 5  CREATE OR REPLACE FUNCTION update_updated_at_column()
 6  RETURNS TRIGGER AS $$%
 7  BEGIN
 8      NEW.updated_at = CURRENT_TIMESTAMP;
 9      RETURN NEW;
10  END;
11  $$ language 'plpgsql';
12
13  -- 2. Books Tablosu İçin Trigger Oluştur
14
15  -- Eğer trigger daha önce varsa önce sil!
16  DROP TRIGGER IF EXISTS update_books_updated_at ON books;
17
18  -- Trigger'i oluştur
19  CREATE TRIGGER update_books_updated_at
20  BEFORE UPDATE ON books
21  FOR EACH ROW
22  EXECUTE PROCEDURE update_updated_at_column();
```

Figure 3: `triggers.sql` dosyasında `updated_at` alanını otomatik güncelleyen tetikleyici tanımı

Bu tetikleyici, her güncelleme işleminden hemen önce çalışarak kayıtların güncellliğini veritabanı seviyesinde garanti altına almaktadır.

4. Yazılım Mimarisi

Proje, kurumsal yazılım geliştirme yaklaşımlarına uygun olarak **Katmanlı Mimari** prensibiyle tasarlanmıştır. Bu yapı, kodun okunabilirliğini, test edilebilirliğini ve bakım kolaylığını artırmaktadır.

4.1. Katmanlı Mimari (Layered Architecture)

Sistem, istemciden gelen isteklerin veritabanına ulaşmasına kadar aşağıdaki katmanlı akışı izlemektedir:

- **Controller Katmanı:** REST API uç noktalarını karşılar ve istemci ile sunucu arasındaki iletişimini yönetir.
- **DTO Katmanı:** API üzerinden taşınan verilerin yapısını tanımlar ve entity sınıflarının doğrudan dış dünyaya açılmasını engeller.
- **Service Katmanı:** İş mantığını içerir; gecikme cezası hesaplama ve kitap öneri algoritmaları bu katmanda gerçekleştirilir.
- **Repository Katmanı:** Spring Data JPA aracılığıyla veritabanı işlemlerini ve CRUD operasyonlarını yönetir.
- **Model/Entity Katmanı:** Veritabanı tablolarının Java tarafından arasındaki karşılıklarını ve ilişkilerini temsil eder.

4.2. Nesne Yönelimli Programlama (OOP) Yaklaşımı

Sistem tasarımindında Java'nın temel OOP prensipleri etkin şekilde kullanılmıştır:

- **Encapsulation:** Entity alanları `private` tanımlanmış, erişim getter/setter metotlarıyla sağlanmıştır.
- **Abstraction:** Repository katmanında arayüzler kullanılarak veritabanı işlemleri soyutlanmıştır.
- **Inversion of Control (IoC):** Spring Boot'un bağımlılık enjeksiyonu mekanizması ile sınıflar arası bağımlılıklar yönetilmiştir.

4.3. RESTful API Prensipleri ve HTTP Metotları

Frontend ve backend arasındaki veri传递 RESTful prensiplere uygun olarak aşağıdaki HTTP metotları üzerinden sağlanmaktadır:

- **GET:** Veri listeleme ve öneri alma işlemleri
- **POST:** Yeni kayıt ve ödünç alma işlemleri
- **PUT:** Güncelleme ve iade işlemleri
- **DELETE:** Yetkili kullanıcılar tarafından silme işlemleri

Bu mimari yapı, sistemin modüler, sürdürülebilir ve ölçeklenebilir olmasını sağlamaktadır.

5. Algoritmalar ve İş Mantığı

Sistemin operasyonel zekası ve karar alma süreçleri Service katmanı içerisinde kapullenmiştir. Bu bölümde, sistemin temel işleyişini belirleyen ödünç alma–iade süreci, gecikme cezası hesaplama mantığı ve kişiselleştirilmiş öneri algoritması özetlenmektedir.

5.1. Ödünç Alma ve Dinamik Envanter Yönetimi

Ödünç alma işlemi, Book, Member, Loan ve Inventory tablolarını etkileyen bütünselik bir iş akışı şeklinde gerçekleştirilir. İşlem öncesinde kitabıın stok durumu kontrol edilir; stok bulunmaması durumunda süreç sonlandırılır. Başarılı bir ödünç alma işleminde stok miktarı azaltılır ve stok sıfıra ulaştığında kitabıın erişilebilirlik durumu otomatik olarak “ödünçte” olarak güncellenir.

5.2. Gecikme Cezası Hesaplama Algoritması

```
// CEZA HESAPLAMA
LocalDateTime dueDate = loan.getLoanDate().plusDays(days: 2);

if (loan.getReturnDate().isAfter(dueDate)) {
    // Tam 24 saatlik döngüler halinde ceza hesapla
    // Örn 1 saat gecikirse bile 1 gün sayılır
    java.time.Duration diff = java.time.Duration.between(dueDate, loan.getReturnDate());
    long overdueDays = (long) Math.ceil(diff.toMinutes() / (24.0 * 60.0));
    // Math.ceil 0.1 -> 1.0 verir.
    if (overdueDays < 1)
        overdueDays = 1;

    double penaltyAmount = overdueDays * 5.0; // GÜNLÜK 5 TL
    loan.setPenalty(penaltyAmount);
} else {
    loan.setPenalty(penalty: 0.0);
}
```

Figure 4: Gecikme cezası hesaplama algoritması

İade işlemi sırasında sistem, teslim süresini aşan durumları otomatik olarak tespit eder. Ödünç alma tarihine tanımlı teslim süresi eklenerek son teslim tarihi belirlenir. Gecikme olması hâlinde, geciken gün sayısı günlük ceza bedeli ile çarpılarak ceza tutarı Loan nesnesine kaydedilir.

5.3. Kişiselleştirilmiş Öneri Algoritması

```
public List<Book> recommendBooks(Long memberId) {
    // 1. Kullanıcının geçmiş ödünç aldığı kitapları getir (Optimize edilmiş DTO)
    List<com.library.library_system.dto.LoanDTO> history = loanRepository.findDTOByMemberId(memberId);

    // Okunan kitapların ID'lerini listele
    List<Long> readBookIds = history.stream()
        .map(loan -> loan.getBookId())
        .collect(Collectors.toList());

    List<Book> finalRecommendations = new java.util.ArrayList<>();

    // 2. Geçmiş varsa favori kategori mantığı uygula
    if (!history.isEmpty()) {
        Map<Long, Integer> categoryFrequency = new HashMap<>();
        for (com.library.library_system.dto.LoanDTO loan : history) {
            Long catId = loan.getCategoryId();
            if (catId != null) {
                categoryFrequency.put(catId, categoryFrequency.getOrDefault(catId, defaultValue: 0) + 1);
            }
        }

        Long favoriteCategoryId = null;
        int maxCount = -1;
        for (Map.Entry<Long, Integer> entry : categoryFrequency.entrySet()) {
            if (entry.getValue() > maxCount) {
                maxCount = entry.getValue();
                favoriteCategoryId = entry.getKey();
            }
        }

        if (favoriteCategoryId != null) {
            // Favori kategoriden okumanmış kitapları getir
            List<Book> categoryRecs = bookRepository.findByCategoryIdAndNotIn(favoriteCategoryId, readBookIds);
            // En fazla 4 tane
            finalRecommendations.addAll(categoryRecs.stream().limit(maxSize: 4).collect(Collectors.toList()));
        }
    }

    // 3. Kullanıcıya özel kitap önerisi
    List<Book> randomFill;
    if (finalRecommendations.size() < 4) {
        int needed = 4 - finalRecommendations.size();

        // Haric okunacaklar: Zaten okundukları + Su anasızı liste'ne eklenenler
        List<Long> excluded = new java.util.ArrayList<>(readBookIds);
        finalRecommendations.forEach(b -> excluded.add(b.getId()));

        List<Book> randomFill;
        if (excluded.isEmpty()) {
            // MİDORI'ye comunitàye overline'meye direkt rastgele al
            randomFill = bookRepository.findRandomBooks();
        } else {
            randomFill = bookRepository.findRandomBooksExcept(excluded, needed);
        }

        finalRecommendations.addAll(randomFill);
    }

    // Yine de 4'ten fazla olursa (randomBooks 4'dense vs) karp
    return finalRecommendations.stream().limit(maxSize: 4).collect(Collectors.toList());
}
```

Figure 5: Kullanıcıya özel kitap öneri algoritmasının temel adımları

Sistemde, içerik tabanlı filtreleme yaklaşımı kullanılarak kullanıcıya özel kitap önerileri sunulmaktadır. Kullanıcının geçmiş ödünç alma verileri analiz edilerek en sık okuduğu kategori belirlenir. Bu kategoriye ait, kullanıcının daha önce ödünç almadığı kitaplar filtrelenerek öneri listesi oluşturulur.

6. Kullanıcı Arayüzü (UI) Geliştirmeleri

Bu bölümde, geliştirilen kütüphane yönetim sisteminin kullanıcı ve yönetici arayüzerine ait ekran görüntüleri sunulmuştur. Arayüzler, kullanıcıların temel işlemleri hızlı şekilde yapabilmesini; yöneticilerin ise envanter, üye ve ödünç süreçlerini yönetmesini sağlayacak şekilde tasarlanmıştır.

6.1. Ortak Sayfalar



Figure 6: Kullanıcı kimlik doğrulama arayüzleri

6.2. Kullanıcı Sayfaları

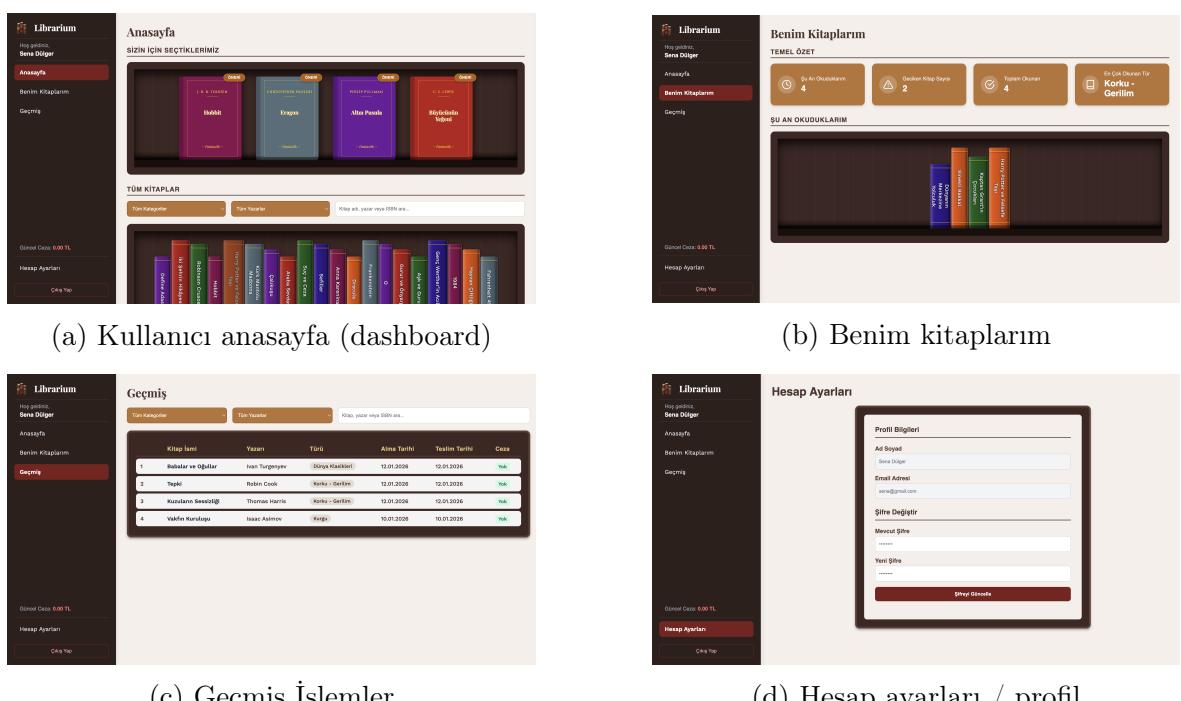
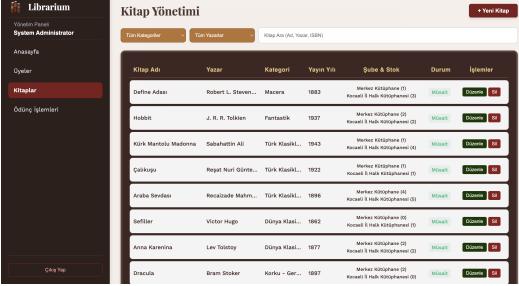


Figure 7: Kullanıcı arayüzüne ait temel ekran görüntüleri

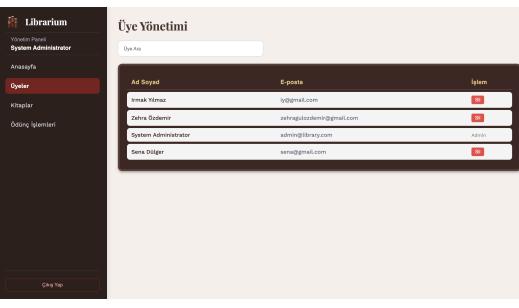
6.3. Admin Sayfaları



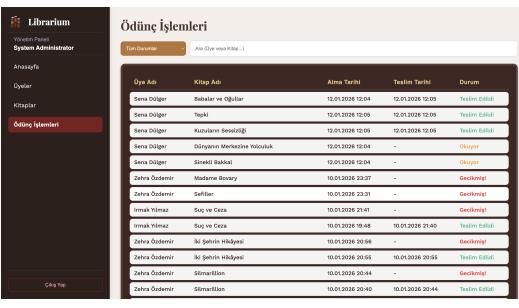
(a) Yönetici paneli (dashboard)



(b) Kitap yönetimi



(c) Üye yönetimi



(d) Ödünç işlemleri

Figure 8: Yönetici arayüzüne ait temel ekran görüntüleri

7. Sonuç ve Değerlendirme

Kütüphane Yönetim ve Kişiiselleştirilmiş Öneri Sistemi, ödev dokümanında belirtilen fonksiyonel ve teknik gereksinimleri eksiksiz şekilde karşılayacak biçimde geliştirilmiştir. Spring Boot tabanlı katmanlı mimari ile sürdürülebilir bir backend yapısı kurulmuş, PostgreSQL üzerinde kullanılan tetikleyiciler sayesinde veri bütünlüğü güvence altına alınmıştır. İçerik tabanlı filtreleme yaklaşımı ile kullanıcıların okuma geçmişi analiz edilerek kişiselleştirilmiş kitap önerileri sunulmuştur. Sonuç olarak proje, nesne yönelimli programlama ve katmanlı mimari prensiplerine uygun, modern kütüphane yönetimi için etkin bir yazılım çözümü ortaya koymuştur.

8. Grup İçi İş Bölümü ve Sorumluluklar

Proje geliştirme sürecinde, kod tabanı ve veritabanı tasarımlı modüler bir yapıda ele alınmış ve 3 kişilik ekibimiz arasında aşağıdaki gibi paylaştırılmıştır. Her üye, sorumlu olduğu modülün veritabanı tasarımindan (Entity), iş mantığına (Service) ve kullanıcı arayüzüne (Frontend) kadar uçtan uca (end-to-end) geliştirmesini üstlenmiştir.

8.1. Sorumluluklar

| Üye / Modül | Veritabanı ve Backend Sorumlulukları | Geliştirilen Özellikler ve Frontend |
|---|--|--|
| Irmak Yılmaz Platform Yönetimi ve Stok (Core & Inventory) | <p>Yetki Alanı: Kimlik Doğrulama, Üyeler, Envanter</p> <p>Veritabanı: members inventory</p> <p>Backend: Spring Security, JWT tabanlı stateless authentication Rol bazlı yetkilendirme (Admin / User) Transactional stok yönetimi (ödünç alma / iade)</p> | login.html, register.html settings.html Stok giriş ve güncelleme arayüzleri Gerçek zamanlı stok düşme/artma mantığı |
| Zehra Gül Özdemir Ana Katalog ve Şube Yönetimi (Main Catalog & Branches) | <p>Yetki Alanı: Kitaplar, Yazarlar, Şubeler</p> <p>Veritabanı: books authors branches</p> <p>Backend: ISBN doğrulama ve mükerrer kayıt engellemesi JPA Specification / Criteria API ile dinamik filtreleme BranchService ile şube–envanter ilişkisi</p> | admin/books.html admin/branches.html Kitap ekleme/düzenleme modalları Yazar ve yayın yılına göre filtreleme |
| Sena Nur Dülger Sınıflandırma, Dolaşım ve Algoritmalar (Classification & Circulation) | <p>Yetki Alanı: Kategoriler, Ödünç Verme, Öneri Sistemi</p> <p>Veritabanı: categories loans</p> <p>Backend: Ödünç alma iş kuralları (es zamanlı kitap limiti) Ceza hesaplama algoritmaları Öneri motoru (okuma geçmişsi ve popülerlik)</p> | user/dashboard.html user/history.html admin/loans.html Kullanıcı bazlı öneri ve analiz ekranları |

8.2. Proje Yönetimi ve Sürüm Kontrolü

Proje geliştirme sürecinde sürüm kontrol sistemi olarak Git, proje barındırma ve iş birliği platformu olarak ise GitHub aktif bir şekilde kullanılmıştır. Grup üyeleri ortak bir repository üzerinden çalışarak geliştirdikleri özellikleri düzenli commit'ler aracılığıyla entegre etmiş ve proje süreci sürüm kontrolü altında yönetilmiştir.

Proje Bağlantısı: <https://github.com/Irmakyil/library-management-system>

9. Kaynakça

- [1] Spring Boot Project, *Spring Boot Documentation*, Erişim adresi: <https://spring.io/projects/spring-boot>
- [2] PostgreSQL Global Development Group, *PostgreSQL Documentation*, Erişim adresi: <https://www.postgresql.org/docs/>
- [3] Spring Data JPA Project, *Spring Data JPA Documentation*, Erişim adresi: <https://spring.io/projects/spring-data-jpa>
- [4] Oracle Corporation, *Java SE 17 Documentation*, Erişim adresi: <https://docs.oracle.com/en/java/javase/17/>
- [5] Mozilla Developer Network (MDN), *Fetch API Documentation*, Erişim adresi: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- [6] OpenAI, *ChatGPT*, Erişim adresi: <https://chatgpt.com/>
- [7] Google, *Gemini AI*, Erişim adresi: <https://gemini.google/>