KÜTÜPHANE OTOMASYON SİSTEMİ PROJE RAPORU

Ders Adı: Veri Tabanı Yönetim Sistemleri **Öğrenci Adı Soyadı:** Furkan Lale

Öğrenci Numarası: 445843 **Bölüm:** Yazılım Mühendisliği

1. GİRİŞ VE PROJENİN AMACI

1.1. Problem Tanımı Günümüzde geleneksel kütüphane yönetim süreçlerinde, kitapların takibi ve üye işlemleri genellikle manuel yöntemlerle veya eski nesil masaüstü yazılımlarla yürütülmektedir. Bu durum, kitapların kaybolması, stok sayımlarında hata yapılması, ödünç alınan kitapların iade tarihlerinin gözden kaçması ve ceza hesaplamalarında tutarsızlıklar yaşanması gibi sorunlara yol açmaktadır. Ayrıca, kullanıcıların kütüphaneye gitmeden kitap durumunu sorgulayamaması, modern çağın gerekliliklerine ters düşmektedir.

1.2. Projenin Amacı Bu projenin temel amacı, bahsi geçen sorunları ortadan kaldıran, web tabanlı, güvenli ve sürdürülebilir bir **Kütüphane Otomasyon Sistemi** geliştirmektir. Sistem; kitap ekleme/silme, üye yönetimi, stok takibi, ödünç alma/verme süreçleri ve otomatik ceza hesaplama işlemlerini dijital ortamda hatasız bir şekilde gerçekleştirmeyi hedefler.

Sistem, **Yönetici (Admin)** ve **Üye (Kullanıcı)** olmak üzere iki temel rol üzerine kurgulanmıştır. Üyeler kitapları inceleyip ödünç alabilirken, yöneticiler sistemin genel işleyişinden ve onay süreçlerinden sorumludur.

2. KULLANILAN TEKNOLOJİLER VE ARAÇLAR

Projenin geliştirilmesinde, endüstri standardı olan **Katmanlı Mimari (Layered Architecture)** ve **Full-Stack** geliştirme yaklaşımı izlenmiştir:

- **Backend:** Java 17, Spring Boot Framework. (Hızlı API geliştirme ve güvenli altyapı için tercih edilmiştir).
- **Veritabanı:** MySQL. (İlişkisel veri tutarlılığı için).
- **Frontend:** HTML5, CSS3, JavaScript (Fetch API).
- **Güvenlik:** Spring Security, JSON Web Token (JWT) ve BCrypt Hashing.
- **Geliştirme Ortamı (IDE):** IntelliJ IDEA.
- **API Test Araçları:** Postman ve Google Chrome Developer Tools.

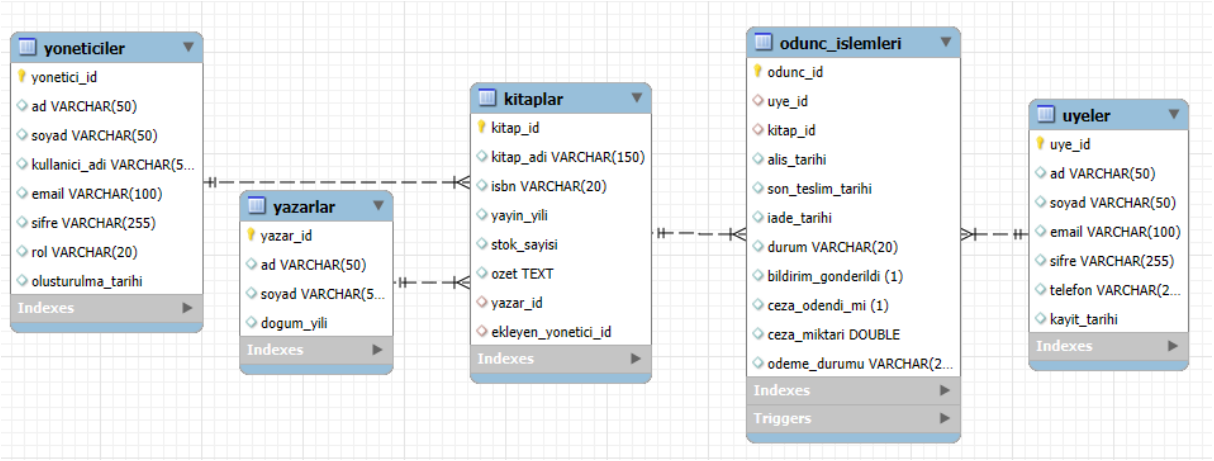
3. VERİTABANI TASARIMI VE SQL YAPISI

Projede veritabanı yönetimi için **"Schema-First" (Önce Şema)** yaklaşımı benimsenmiştir. Tablolar arasındaki ilişkiler (One-to-Many), veri bütünlüğünü (Data Integrity) sağlayacak şekilde kurgulanmıştır.

3.1. Tablo Yapıları

Veritabanı temel olarak 5 ana tablodan oluşmaktadır:

1. **Uyeler Tablosu: Kütüphaneyi kullanacak standart kullanıcıların verilerini tutar.**
    - ***Alanlar:*** **uye_id (PK), ad, soyad, email (Unique), sifre, telefon, kayit_tarihi.**
2. **Yoneticiler Tablosu: Sistem üzerindeki yetkili (Admin) kullanıcıları tutar.**
    - ***Alanlar:*** **yonetici_id (PK), ad, soyad, kullanici_adi, email, sifre, rol.**
3. **Yazarlar Tablosu: Kitap yazarlarının bilgilerini tutar. Veri tekrarını önlemek için kitap tablosundan ayrılmıştır.**
    - ***Alanlar:*** **yazar_id (PK), ad, soyad, dogum_yili.**
4. **Kitaplar Tablosu: Kütüphanedeki materyallerin bilgilerini barındırır. Her kitap bir yazara bağlıdır.**
    - ***Alanlar:*** **kitap_id (PK), kitap_adi, isbn, yazar_id (FK), yayin_yili, stok_sayisi, ozet.**
5. **Odunc_Islemleri Tablosu: Hangi kitabın, kim tarafından, ne zaman alındığını ve iade durumunu takip eder.**
    - ***Alanlar:*** **odunc_id (PK), uye_id (FK), kitap_id (FK), alis_tarihi, son_teslim_tarihi, iade_tarihi, durum, ceza_miktari.**



(Şekil 1: Veritabanı Varlık-İlişki Diyagramı)

3.2. Tetikleyici (Trigger) Mekanizması

Projede stok takibinin hatasız yapılması için veritabanı seviyesinde Trigger kullanılmıştır. Java kodunda bir hata olsa bile, veritabanı stok tutarlılığını garanti eder.

Kullanılan Trigger Mantığı (Örnek SQL):

```
DELIMITER //
CREATE TRIGGER stok_dusur
AFTER INSERT ON odunc_islemleri
FOR EACH ROW
BEGIN
    IF NEW.durum = 'ODUNC_ALINDI' THEN
        UPDATE kitaplar
        SET stok_sayisi = stok_sayisi - 1
        WHERE kitap_id = NEW.kitap_id;
    END IF;
END;
//
DELIMITER ;
```

Bu tetikleyici sayesinde, bir kitap ödünç verildiği anda kitaplar tablosundaki stok sayısı otomatik olarak 1 azaltılmaktadır.

4. YAZILIM MİMARİSİ

Proje, **Model-View-Controller (MVC)** tasarım desenine uygun olarak geliştirilmiştir. Bu yapı, kodun okunabilirliğini artırmış ve bakımını kolaylaştırmıştır.

Katmanlar

- **Entity Katmanı:** Veritabanı tablolarının Java sınıfları halindeki karşılığıdır.
- **Repository Katmanı:** Veritabanı ile konuşan katmandır. JpaRepository arayüzü sayesinde karmaşık SQL sorguları yazmadan veri çekme işlemleri sağlanmıştır.
    - *Örnek:* findByKitapAdiContainingIgnoreCase(String isim) metodu ile SQL yazmadan arama fonksiyonu geliştirilmiştir.
- **Service Katmanı:** İş mantığının kurulduğu yerdir. Stok kontrolü, ceza hesaplama, mükerrer kitap alımını engelleme gibi kurallar burada kodlanmıştır.
- **Controller Katmanı:** Frontend'den gelen HTTP isteklerini karşılar.

5. GÜVENLİK ÖNLEMLERİ

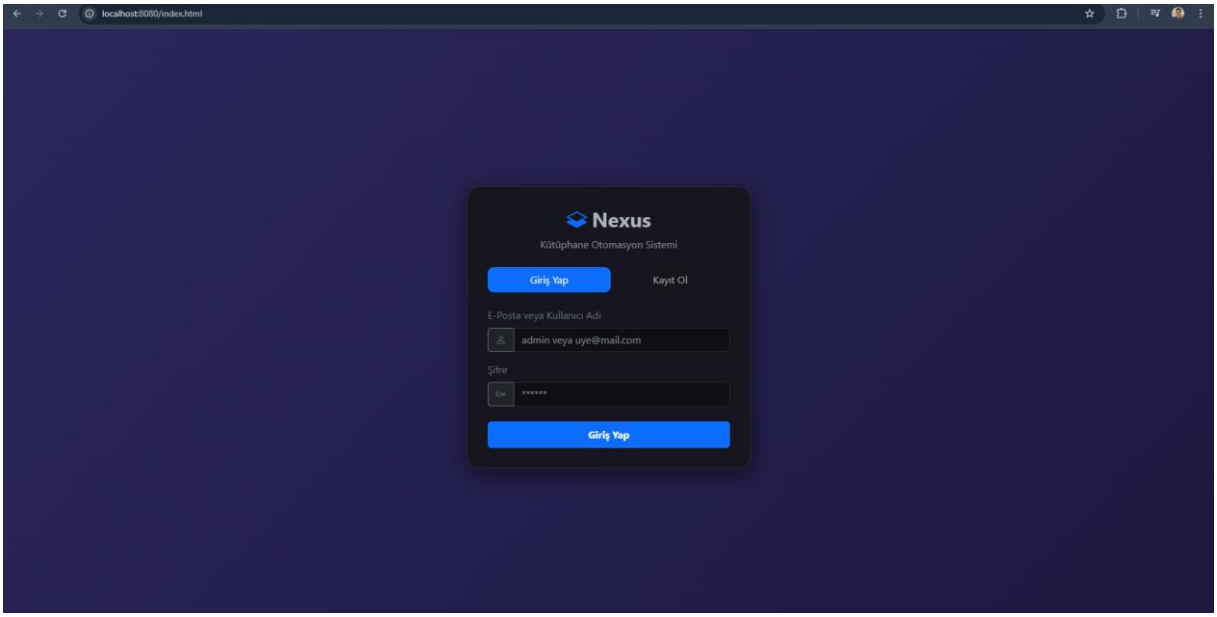
Sistemin güvenliği modern web standartlarına uygun olarak tasarlanmıştır:

1. **JWT (JSON Web Token):** Klasik oturum (Session) yönetimi yerine, ölçeklenebilir olması için JWT kullanılmıştır. Kullanıcı giriş yaptığında sunucu şifreli bir Token üretir ve kullanıcı sonraki isteklerinde bu kimlik kartını gösterir.
2. **Şifreleme (Hashing):** Kullanıcı şifreleri veritabanında asla açık metin olarak saklanmaz. BCryptPasswordEncoder kullanılarak hashlenmiş formatta tutulur.
3. **CORS Politikaları:** Frontend ve Backend farklı portlarda çalıştığı için (Localhost:5500 ve Localhost:8080), SecurityConfig dosyasında gerekli izinler tanımlanmıştır.

6. KULLANICI ARAYÜZÜ VE SENARYOLAR

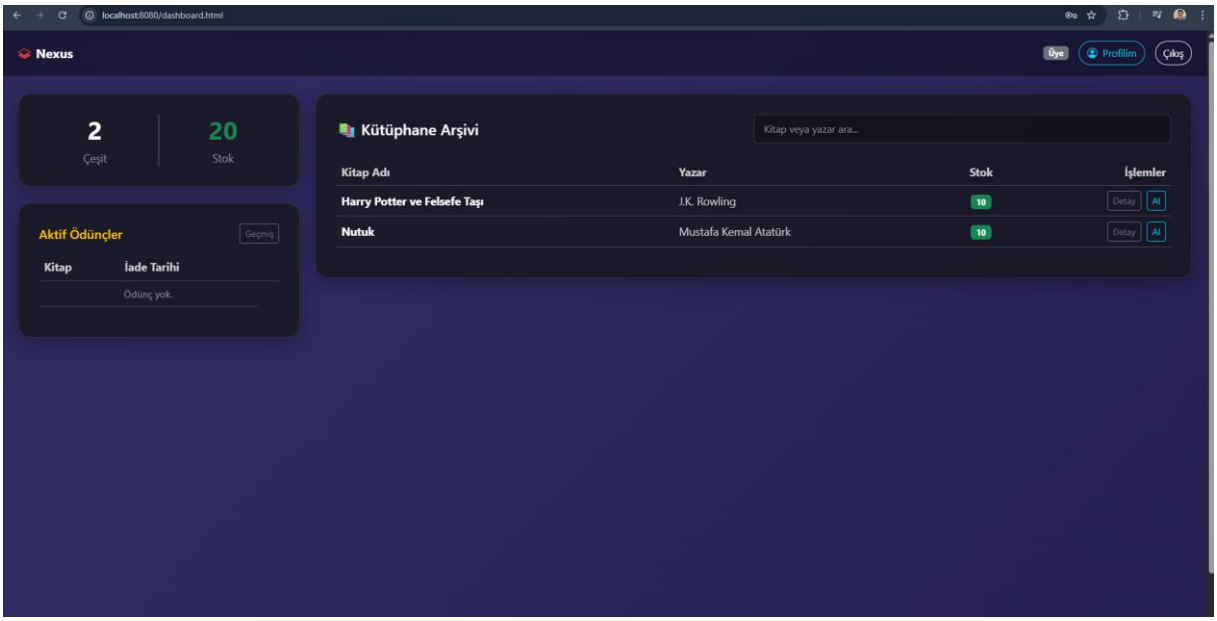
Sistemin arayüzü kullanıcı dostu (User Friendly) bir yapıda, HTML/CSS ve JavaScript kullanılarak tasarlanmıştır.

6.1. Giriş ve Kayıt Ekranı Kullanıcılar sisteme e-posta ve şifreleri ile giriş yaparlar. Giriş başarılı olduğunda sunucudan alınan JWT Token, tarayıcının localStorage hafızasına kaydedilir.



(Şekil 2: Kullanıcı Giriş Paneli)

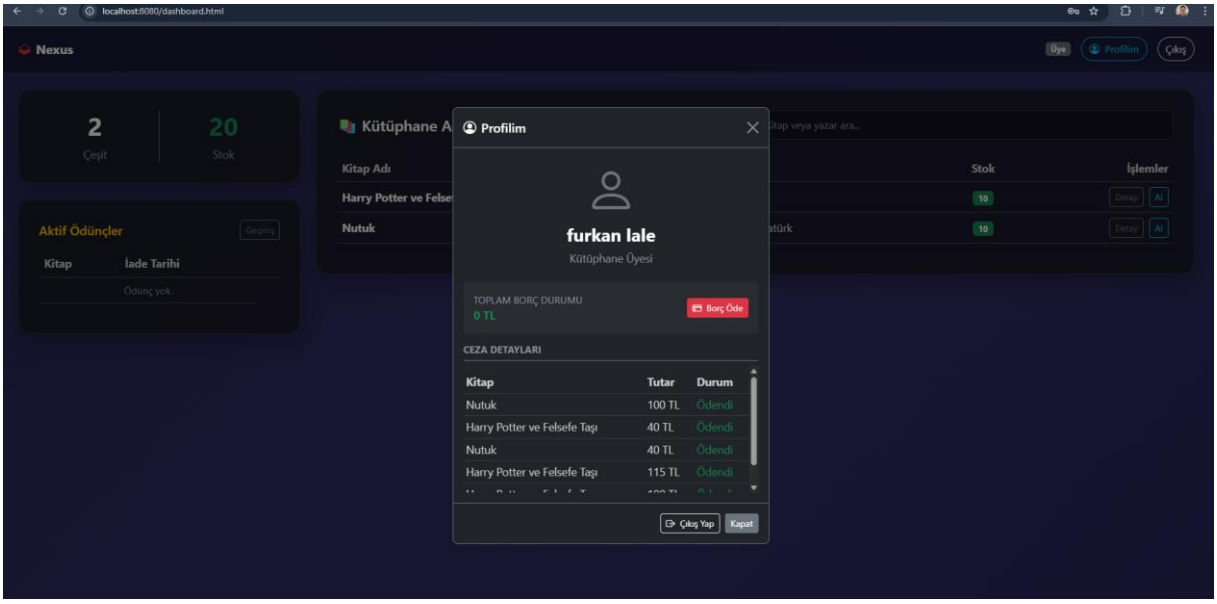
6.2. Kitap Listeleme ve Arama Kullanıcılar kütüphanedeki tüm kitapları görebilir. Arama kutusuna yazılan her harfte Backend'e dinamik istek atılarak veritabanından anlık filtreleme yapılır. Stokta olmayan kitapların butonu pasif hale gelir.



(Şekil 3: Kitap Kataloğu ve Arama Fonksiyonu)

6.3. Ödünç Alma ve Ceza Simülasyonu (Demo Modu) Projenin sunumunda işleyişin hızlı gösterilebilmesi için özel bir "Demo Modu" kurgulanmıştır.

- Normalde 15 gün olan ödünç süresi, sunum için **1 Dakika** olarak ayarlanmıştır.
- 1 dakikayı geçiren kullanıcılar için sistem otomatik olarak **dakika başı 5 TL** gecikme cezası hesaplar.
- Cezası olan kullanıcı, borcunu ödemeden yeni kitap alamaz.



(Şekil 4: Profil ve Borç Takip Ekranı)

7. KARŞILAŞILAN ZORLUKLAR VE ÇÖZÜMLER

Proje geliştirme sürecinde bazı teknik zorluklarla karşılaşılmış ve şu şekilde çözülmüştür:

- **CORS Hataları:** Frontend'in Backend API'ye erişiminde "Access-Control-Allow-Origin" hatası alınmıştır. Bu sorun, Spring Security konfigürasyonunda CorsConfigurationSource Bean'i tanımlanarak çözülmüştür.
- **İlişkisel Veri Silme (Foreign Key):** Bir kitabı silmeye çalışırken, eğer o kitap geçmişte bir üyeye verilmişse veritabanı hata vermekteydi. Bu durum, Controller katmanında try-catch blokları ile yakalanarak kullanıcıya "Bu kitap işlem gördüğü için silinemez" uyarısı verilerek yönetilmiştir.

8. SONUÇ

Bu proje kapsamında, bir kütüphanenin ihtiyaç duyabileceği temel otomasyon fonksiyonları başarıyla dijitalleştirilmiştir. Proje sayesinde;

- İlişkisel veritabanı tasarımı,
- Trigger mantığı,
- Spring Boot ile RESTful API geliştirme,
- Frontend-Backend haberleşmesi konularında derinlemesine yetkinlik kazanılmıştır.

Sistem, gelecekte barkod okuyucu entegrasyonu ve mobil uygulama desteği ile geliştirilmeye açık, modüler bir mimariye sahiptir.