PROJE II: WEB TABANLI SOSYAL KÜTÜPHANE PLATFORMU

Ömer Faruk Toycu
230202040
Ali Berke Erenler
230202069

1. Giriş

Günümüzde dijital içeriklerin (film, kitap, dizi) sayısının hızla artması, kullanıcıların tükettikleri veya tüketmek istedikleri içerikleri organize etme ihtiyacını doğurmuştur. Mevcut platformlar genellikle ya sadece film ya da sadece kitap odağında hizmet vermekte, sosyal etkileşim özellikleri ise sınırlı kalmaktadır.

Bu projenin amacı, farklı içerik türlerini tek bir çatı altında toplayan, kullanıcı dostu ve sosyal etkileşimi merkeze alan bir platform geliştirmektir. Geliştirilen sistem, kullanıcıların içerik keşfetmesini kolaylaştırırken, "İzledim", "Okunacak" gibi listelerle kişisel arşiv yönetimini sağlamaktadır. Ayrıca, takipçi sistemi sayesinde kullanıcılar, benzer zevklere sahip diğer bireylerin aktivitelerinden haberdar olabilmektedir.

2. Yöntem ve Sistem Mimarisi

2.1. Yazılım Mimarisi

Proje, İstemci-Sunucu (Client-Server) mimarisine uygun olarak geliştirilmiştir.

- **Sunucu Tarafı (Backend):** Yüksek performanslı asenkron işlem yeteneği nedeniyle Python dilinde yazılmış **FastAPI** framework'ü kullanılmıştır. Veri doğrulama için **Pydantic** şemaları, güvenlik için **OAuth2** ve **JWT** (JSON Web Token) kullanılmıştır.
- **İstemci Tarafı (Frontend):** Modern HTML5, CSS3 ve Vanilla JavaScript kullanılarak geliştirilmiştir. Backend ile iletişim RESTful API uç noktaları üzerinden sağlanmaktadır.

2.2. Veri Tabanı Tasarımı

Sistemin veritabanı **MySQL** üzerinde kurgulanmıştır. Veri tutarlılığını sağlamak ve karmaşık sorguları yönetmek için ilişkisel veritabanı yapısı tercih edilmiştir.

Veritabanı tasarımında "Polimorfik" bir yapı kullanılmıştır. contents tablosu temel içerik bilgilerini tutarken, movies ve books tabloları bu tablodan türetilerek içerik tipine özel verileri saklar.

Sınıf Diyagramı (Class Diagram): Aşağıdaki diyagram sistemin temel varlıklarını ve aralarındaki ilişkileri göstermektedir:

```
classDiagram
    class User {
```

```
    +int id
    +string username
    +string email
    +string hashed_password
    +create_activity()
  }
  class Content {
    +int id
    +string title
    +string type
    +float average_rating
  }
  class Movie {
    +int runtime
    +string director
    +string cast
  }
  class Book {
    +string authors
    +string publisher
    +int page_count
  }
  class Activity {
    +int id
    +string type
    +datetime created_at
  }
  class Rating {
    +float score
  }
  class Review {
    +string text
    +int likes_count
  }
  class CustomList {
    +string name
    +bool is_public
  }
  User "1" --> "*" Activity : generates
  User "1" --> "*" Rating : gives
  User "1" --> "*" Review : writes
  User "1" --> "*" CustomList : owns
  User "1" --> "*" User : follows
  Content <|-- Movie : inherits
  Content <|-- Book : inherits
  Content "1" --> "*" Rating : has
  Content "1" --> "*" Review : has
  Activity --> Content : references
```

2.3. Kullanım Durumu (Use Case) Diyagramı

Sistemdeki temel aktör "Kayıtlı Kullanıcı"dır. Ziyaretçiler sadece kayıt ve giriş işlemlerini yapabilirken, kayıtlı kullanıcılar tüm fonksiyonlara erişebilir.

```
usecaseDiagram
  actor Kullanıcı as "Kayıtlı Kullanıcı"

  package "Web Library Sistemi" {
    usecase "İçerik Ara (Film/Kitap)" as UC1
    usecase "Kütüphaneye Ekle (İzlendi/Okunacak)" as UC2
    usecase "Puan Ver ve Yorum Yap" as UC3
    usecase "Özel Liste Oluştur" as UC4
    usecase "Kullanıcı Takip Et" as UC5
    usecase "Akışı (Feed) Görüntüle" as UC6
  }

  Kullanıcı --> UC1
  Kullanıcı --> UC2
  Kullanıcı --> UC3
```

Kullanıcı --> UC4
Kullanıcı --> UC5
Kullanıcı --> UC6

2.4. Temel Algoritmalar

2.4.1. Sosyal Akış (Feed) Algoritması

Kullanıcının ana sayfasında gördüğü akış, takip ettiği kullanıcıların aktivitelerinden oluşur. feed.py dosyasında uygulanan bu mantık, ilişkisel veritabanı sorguları ile optimize edilmiştir.

Yalancı Kod (Pseudo-Code): Sosyal Akış Oluşturma

```
ALGORİTMA GetUserFeed(current_user_id, skip, limit):
  1. Takip Edilenleri Bul:
    followed_ids = SELECT followed_id FROM follows
                   WHERE follower_id = current_user_id

  2. Eğer takip edilen yoksa:
    DÖNDÜR Boş Liste

  3. Aktiviteleri Getir:
    activities = SELECT * FROM activities
                 WHERE user_id IN followed_ids
                 ORDER BY created_at DESC
                 LIMIT limit OFFSET skip

  4. Aktiviteleri Zenginleştir (Döngü):
    HER activity İÇİN:
        activity_data = activity temel bilgileri

        EĞER activity.type == 'review':
            Review tablosundan yorum metnini çek
            activity_data.review_text = review.text

        EĞER activity.type == 'rating':
            Rating tablosundan puanı çek
            activity_data.score = rating.score

        Beğeni Durumunu Kontrol Et:
            is_liked = CHECK IF current_user liked this activity
            activity_data.is_liked = is_liked

        Listeye Ekle:
        enriched_list.ADD(activity_data)

  5. DÖNDÜR enriched_list
```

2.4.2. Harici API Entegrasyonu

Sistem, yerel veritabanında olmayan bir içerik arandığında, tmdb_service.py ve books_service.py modülleri aracılığıyla anlık olarak dış servislere istek atar. Kullanıcı bir içeriğe tıkladığında, bu içerik yerel veritabanına (contents tablosuna) kaydedilir (caching pattern), böylece sonraki erişimlerde dış servise gidilmez.

3. Deneysel Sonuçlar ve Ekran Görüntüleri

Proje yerel ortamda başarıyla

test edilmiştir. Aşağıdaki test senaryoları gerçekleştirilmiştir:

1. **Kimlik Doğrulama:** Kullanıcılar başarıyla kayıt olup JWT token alabilmekte ve korumalı endpoint'lere (örn: /api/library/me) erişebilmektedir.
2. **Arama Performansı:** "Inception" araması yapıldığında TMDb API'den 200ms altında sonuç dönülmekte ve veriler doğru şekilde MovieResponse şemasına map edilmektedir.
3. **Veri Bütünlüğü:** Bir kullanıcı silindiğinde (Cascade Delete), ona ait puanlar, yorumlar ve kütüphane kayıtlarının da veritabanından temizlendiği doğrulanmıştır (ondelete="CASCADE").
4. **Kütüphane Yönetimi:** Kullanıcı bir kitabı "Okunacak" olarak işaretlediğinde, bu durum user_libraries tablosuna status='to_read' olarak işlenmektedir. Aynı içerik "Okundu"ya çekildiğinde durum güncellenmektedir.

Arayüz Sonuçları:

Frontend tarafında responsive tasarım ilkeleri uygulanmış (styles.css), mobil ve masaüstü cihazlarda uyumlu bir görünüm elde edilmiştir. Yükleme durumları (loading states) ve hata mesajları (toast notifications) kullanıcı deneyimini iyileştirmek için entegre edilmiştir.

4. Sonuç

Bu projede, modern web teknolojileri kullanılarak ölçeklenebilir bir sosyal kütüphane platformu geliştirilmiştir. Polimorfik veritabanı tasarımı sayesinde hem kitap hem de film verileri verimli bir şekilde yönetilmiştir. FastAPI'nin asenkron yapısı, dış API çağrılarında sistemin bloklanmadan çalışmasını sağlamıştır.

Gelecek çalışmalarda sisteme; kullanıcının geçmiş beğenilerine dayalı bir **öneri motoru (recommendation system)**, kullanıcılar arası mesajlaşma ve koyu/açık tema seçeneğinin eklenmesi planlanmaktadır.

5. Kaynakça

[1] FastAPI Documentation. (2023). "FastAPI". [Online]. Available: https://fastapi.tiangolo.com/
[2] SQLAlchemy. (2023). "The Python SQL Toolkit and Object Relational Mapper". [Online]. Available: https://www.sqlalchemy.org/
[3] The Movie Database (TMDb). (2023). "API Documentation". [Online]. Available: https://developer.themoviedb.org/docs
[4] Google Developers. (2023). "Google Books APIs". [Online]. Available: https://developers.google.com/books