# Yazılım Laboratuvarı 1  2.Proje Raporu

Çağatay ALTINTOPAÇ
Kocaeli Üniversitesi
*Bilgisayar Mühendisliği*
cagatayaltintopac@gmail.com

Sadık GÖLPEK
*Kocaeli Üniversitesi*
*Bilgisayar Mühendisliği*
sadikgolpek@gmail.com

## I. ÖZET

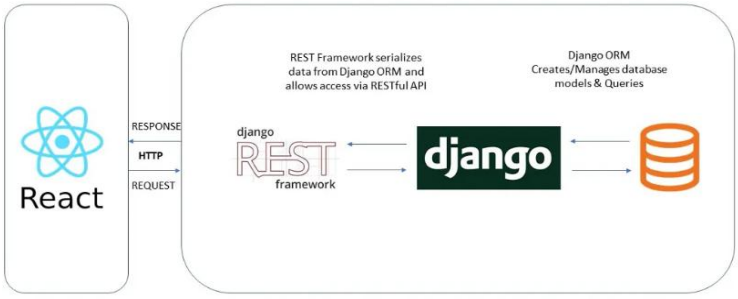
Bu projede, kullanıcıların kitap ve film içeriklerini keşfedebildiği, puanlayıp yorumlayabildiği ve sosyal etkileşim kurabildiği mobil uyumluluğu olan web tabanlı bir **Platform** geliştirilmiştir. Sistem, modern web teknolojilerine dayalı **RESTful mimaride** tasarlanmıştır.

Platformun istemci tarafı (frontend) **React** ile, sunucu tarafı (backend) ise **Django REST Framework** kullanılarak geliştirilmiş; böylece bileşenler arası gevşek bağlı, ölçeklenebilir ve standartlaştırılmış bir iletişim modeli elde edilmiştir. Verilerin kalıcı ve güvenilir biçimde saklanması için veritabanı motoru olarak da **PostgreSQL** tercih edilmiştir.

Film verileri **The Movie Database (TMDb)** API'si üzerinden, kitap verileri ise **Google Books API** üzerinden gerçek zamanlı olarak REST istekleri ile alınmıştır. Bu sayede manuel veri girişine ihtiyaç duyulmamış, platformun içerik kalitesi ve kapsamı önemli ölçüde artırılmıştır.

Geliştirme süreci boyunca proje, **Git tabanlı sürüm kontrol sistemi** kullanılarak yönetilmiş; tüm aşamalar düzenli commit mesajlarıyla belgelenmiş ve GitHub üzerinde barındırılmıştır. Yapılan her güncelleme, özellik ekleme, hata düzeltme ve yapılandırma değişikliği ayrı commit'ler hâlinde kayıt altına alınmış; böylece projenin gelişim süreci izlenebilir, geri alınabilir ve ekip içi iş birliğine uygun bir yapıya kavuşturulmuştur. Git kullanımı, kod bütünlüğünü korumanın yanı sıra, geliştirme sürecinin şeffaf ve düzenli ilerlemesini sağlamıştır.

Sonuç olarak sistem; kullanıcı dostu arayüzü, RESTful API yapısı, harici servis entegrasyonları, Git tabanlı sürüm yönetimi ve sosyal etkileşim mekanizmalarıyla dinamik, izlenebilir ve genişletilebilir bir dijital kültür platformu sunmaktadır.



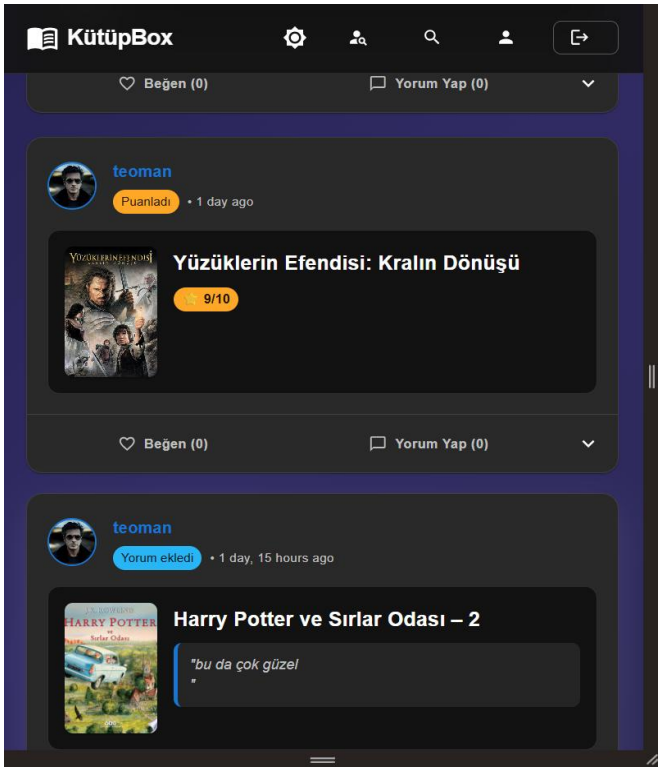

## II. GİRİŞ

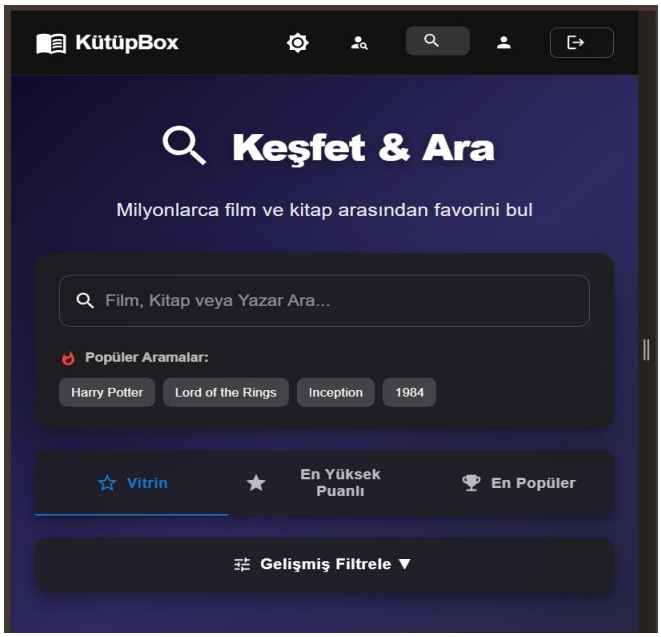
Dijital içerik platformlarının yaygınlaşmasıyla birlikte kullanıcıların medya deneyimlerini kişiselleştirebildiği, içerik hakkında yorum yapabildiği ve diğer kullanıcılarla etkileşim kurabildiği sistemlere olan ihtiyaç artmıştır. Bu proje, bu ihtiyaca yönelik olarak geliştirilmiş web tabanlı bir sosyal kütüphane platformudur.

Sistem, modern web geliştirme standartlarına uygun olacak şekilde **RESTful istemci–sunucu mimarisini**

temel almıştır. REST mimarisi sayesinde frontend ve backend birbirinden bağımsız geliştirilebilmekte, API endpoint'leri evrensel standartlarda iletişim sağlamakta ve sunucu tarafı yatayda kolayca ölçeklendirilebilmektedir.

Frontend tarafında, JavaScript ekosisteminin en yaygın kullanılan kütüphanelerinden biri olan **React** tercih edilmiştir. React'ın bileşen tabanlı yapısı, duruma duyarlı ekran güncellemeleri ve yeniden kullanılabilir UI modülleri sayesinde platformun modern, dinamik ve kullanıcı dostu bir arayüze sahip olması sağlanmıştır.

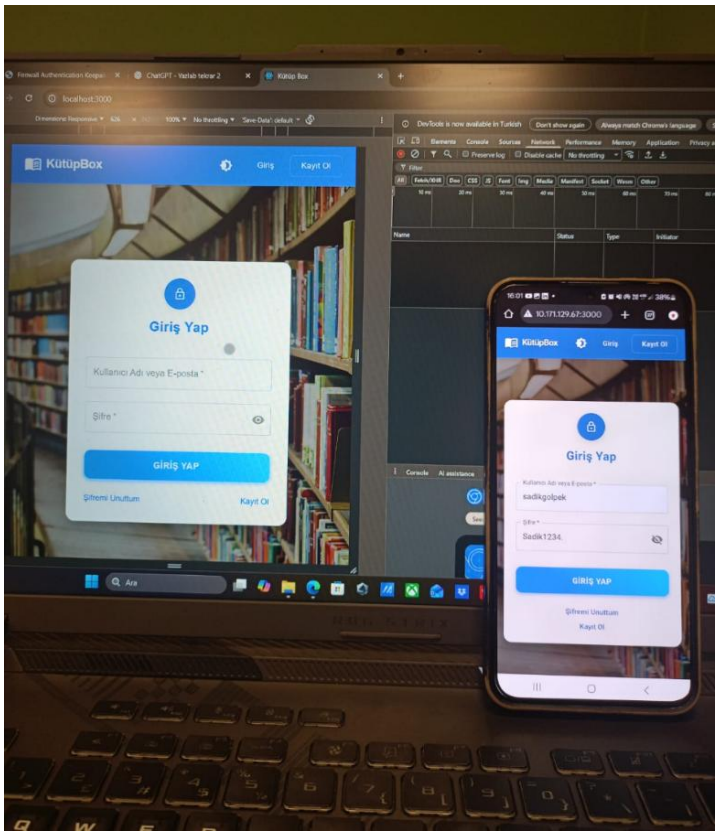



Backend tarafında **Django REST Framework** kullanılarak tüm iş mantığı, veri işleme süreçleri ve REST API uç noktaları oluşturulmuştur. Verilerin saklanması için kullanılan **PostgreSQL**, ilişkisel ve ölçeklenebilir bir yapı sunmasının yanında güçlü veri bütünlüğü sağlamaktadır. Projede veritabanı yapıları doğrudan Python kodu kullanılarak Django ORM üzerinden tanımlanmış olup, bu yaklaşımın sağladığı önemli avantajlar şunlardır: veritabanı şemalarının kod üzerinden yönetilebilmesi, migration işlemleri ile şemanın esnek şekilde güncellenebilmesi, SQL sorgularının Python nesneleriyle güvenli ve yüksek seviyeli bir biçimde gerçekleştirilebilmesi ve farklı veritabanlarına minimum kod değişikliğiyle geçiş yapılabilmesi.


Platformun temel yapı taşlarından biri, kullanıcı kimlik doğrulama sistemidir. Uygulamada **kayıt olma, giriş yapma ve şifremi unuttum** süreçleri tamamen işlevsel hâle getirilmiştir. Kayıt sırasında kullanıcıdan temel bilgiler alınmakta, giriş aşamasında ise JWT tabanlı kimlik doğrulama yapılmaktadır. "Şifremi Unuttum" özelliği kapsamında kullanıcıya gerçek bir **e-posta gönderilmekte**, şifre sıfırlama isteği üzerine **doğrulama kodu** ile güvenli bir doğrulama adımı gerçekleştirilmektedir. Bu yapı, hem kullanıcı deneyimini güçlendirmekte hem de sistemin güvenilirliğini artırmaktadır.

Film ve kitap bilgilerinin manuel girişle değil, **RESTful üçüncü parti API'ler** (TMDb ve Google Books) aracılığıyla dinamik olarak elde edilmesi, platformun veri doğruluğunu ve güncelliğini önemli ölçüde artırmıştır. Aynı zamanda bu yapı, platformun zamanla yeni içerik türleri veya ek medya kaynaklarıyla genişletilebilmesine olanak sağlayan modüler bir entegrasyon mimarisine sahiptir.




## III. YÖNTEM

Bu bölümde sistemin geliştirilme süreci, kullanılan teknolojiler, çalışma prensipleri, veritabanı tasarımı ve sürüm kontrol mekanizmaları bütüncül bir yapıda açıklanmaktadır.

Sistemin istemci tarafı, JavaScript ekosisteminde en yaygın kullanılan kütüphanelerden biri olan **React** ile geliştirilmiştir. React'ın bileşen tabanlı mimarisi, kullanıcı arayüzünün modüler, dinamik ve yeniden kullanılabilir bileşenlerle oluşturulmasına imkân sağlamıştır. Frontend tarafının çalıştırılması için proje dizininde `npm start` komutu kullanılmış olup, bu komut geliştirme sunucusunu başlatmakta ve hot-reload özelliği sayesinde yapılan değişikliklerin eş zamanlı olarak tarayıcıya yansıtılmasını sağlamaktadır.

Sunucu tarafında ise **Django REST Framework** tercih edilmiş, tüm iş mantığı ve API uç noktaları REST prensiplerine göre tasarlanmıştır. Backend geliştirme ortamı, Python bağımlılıklarının diğer projelerden izole edilmesi için bir **virtual environment (venv)** içerisinde oluşturulmuştur. Sanal ortamın etkinleştirilmesi `venv\Scripts\activate` komutu ile yapılmış, böylece projeye ait paketlerin global Python kurulumundan bağımsız çalışması sağlanmıştır. Django sunucusunun çalıştırılması ise `python manage.py runserver` komutu ile gerçekleştirilmiştir. Bu yapı sayesinde front–back ayrımı netleşmiş, sistem REST tabanlı iletişim kuran bağımsız katmanlara ayrılmıştır.

Projenin başlangıç aşamasında veritabanı olarak **Microsoft SQL Server (MSSQL)** ve SSMS 21 kullanılmaya çalışılmış ancak hem arayüz performansının yavaş olması hem de Django ORM ile yaşanan uyumsuzluklar nedeniyle bu tercih sürdürülebilir bulunmamıştır. Migration işlemlerinin uzun sürmesi ve veri tipleri üzerindeki kısıtlar, daha esnek ve hızlı bir yapıya geçilmesini gerekli kılmıştır. Bu nedenle proje **PostgreSQL** üzerine taşınmış ve veritabanı yönetimi **pgAdmin 4** üzerinden gerçekleştirilmiştir. PostgreSQL'in Django ORM ile doğal uyumluluğu, JSONField desteği ve açık kaynak yapısının sağladığı esneklik, projeyi daha kararlı ve ölçeklenebilir bir hâle getirmiştir.

Veritabanı tasarımı, Django ORM kullanılarak `models.py` dosyasında oluşturulmuş ve migration mekanizmasıyla PostgreSQL'e aktarılmıştır. ORM yaklaşımı sayesinde tablo yapıları, ilişkiler, kısıtlamalar ve veri tipleri doğrudan Python kodu üzerinden yönetilmiştir. Sistem genelindeki modeller, Django'nun yerleşik kimlik doğrulama altyapısı tarafından sunulan **Auth User** tablosu üzerine inşa edilmiştir. Kullanıcı bilgileri (kullanıcı adı, e-posta, şifre gibi temel alanlar) Django'nun varsayılan `User` modelinden sağlanmış; proje içerisinde tanımlanan tüm ilişkiler bu model üzerinden gerçekleştirilmiştir.

Sistem genelindeki modellerin işlevleri aşağıdaki gibidir:

---

**Profil modeli:**

Kullanıcının avatar ve biyografi bilgilerinin saklanmasını sağlar.

Django'nun sağladığı `User` tablosuyla bire bir ilişkilendirilmiştir (OneToOneField).

**Takip modeli:**

Kullanıcıların birbirini takip edebilmesini sağlar.

`takip_eden` ve `takip_edilen` alanları Auth User tablosuna bağlanır.

Aynı takip ilişkisinin tekrar oluşturulmasını engellemek için `unique_together` kullanılır.

**Puan modeli:**

Kullanıcıların film veya kitap içeriklerine verdikleri puanları tutar.

`content_id`, TMDB veya Google Books API'den gelen benzersiz içerik kimliğini temsil eder.

Bir kullanıcının aynı içeriğe tekrar puan vermesini önlemek için tekillik kısıtı bulunmaktadır.

**Yorum modeli:**

Kullanıcıların içeriklere yazdığı yorumları saklar.

Tarih alanı otomatik olarak güncellenir (`auto_now=True`).

Her yorum, Auth User tablosundaki bir kullanıcıya bağlıdır.

**Kütüphane modeli:**

Kullanıcının bir içeriği "izledim", "izlenecek", "okudum", "okunacak" şeklinde işaretlemesini sağlar.

Aynı içeriğin bir kullanıcıya ikinci kez eklenmesini önlemek için `unique_together` tanımlanmıştır.

**Özel Liste modeli:**

Kullanıcıların kişisel film/kitap koleksiyonları oluşturmasını sağlar.

Her liste Auth User tablosundaki bir kullanıcıya aittir.

**Özel Liste İçeriği modeli:**

Bir listenin içinde bulunan film/kitap içeriklerini tutar.

Aynı içeriğin aynı listeye iki kez eklenmesini engelleyerek veri tekrarını önler.

**Aktivite modeli:**

Sosyal akışta gösterilen tüm kullanıcıeylemlerini (puan verme, yorum yapma, takipetme, listeye ekleme vb.) kaydeder.

`meta` alanı JSON formatında esnek veri saklamayı mümkün kılar.

**AktiviteLike modeli:**

Kullanıcıların bir aktiviteyi beğenme işlemlerini kaydeder.

Aynı kullanıcının aynı aktiviteyi tekrar beğenmesini önlemek amacıyla tekillik kısıtı vardır.

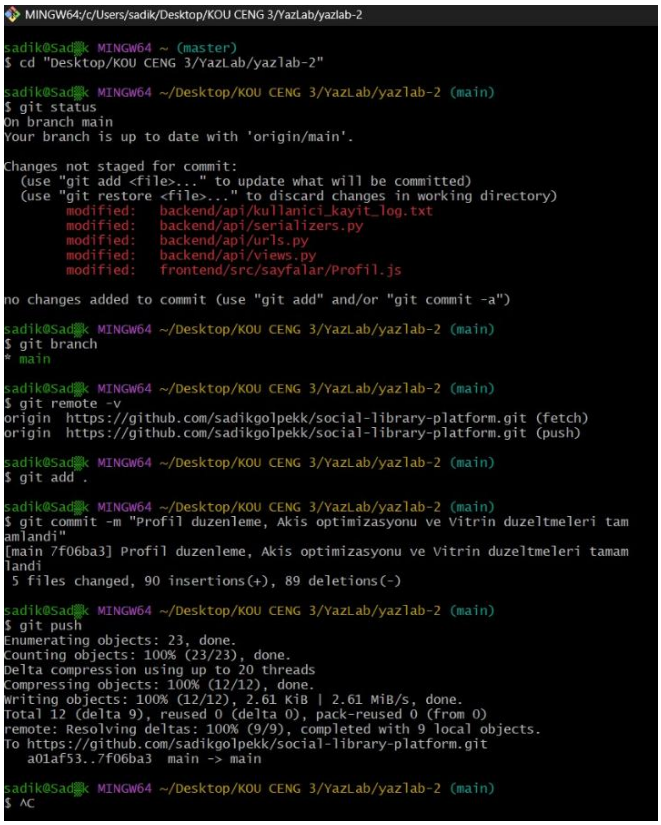
**AktiviteYorum modeli:**

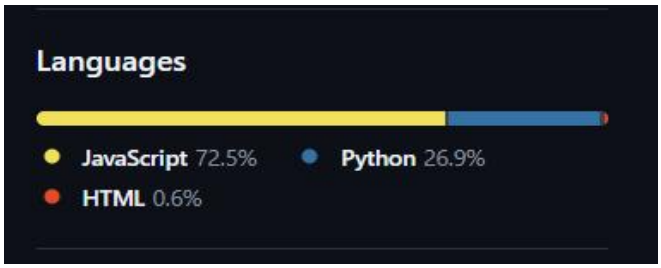
Aktivite kartları üzerinde yapılan yorumları saklar.

Her yorum belirli bir aktiviteye bağlıdır (ForeignKey).

## Versiyon Kontrol Sistemi

Projede sürüm kontrolü için **Git** kullanılmış, geliştirme sürecinin tüm aşamaları düzenli commit mesajlarıyla belgelenmiştir. Git deposu Git Bash üzerinden yönetilmiş; git add ., git commit -m "..." ve git push origin main komutlarıyla yapılan güncellemeler GitHub üzerinde saklanmıştır. Bu süreç, yazılımın değişim geçmişinin izlenebilir olmasını, hatalı değişikliklerin kolayca geri alınabilmesini ve projenin sürdürülebilir şekilde yönetilmesini mümkün kılmıştır.




Platformun çalışma prensibi frontend, backend ve veritabanı olmak üzere üç temel katmanda gerçekleşmektedir. Bu süreç aşağıdaki adımlarla özetlenebilir:

### *A. 1. Frontend Akışı (React)*

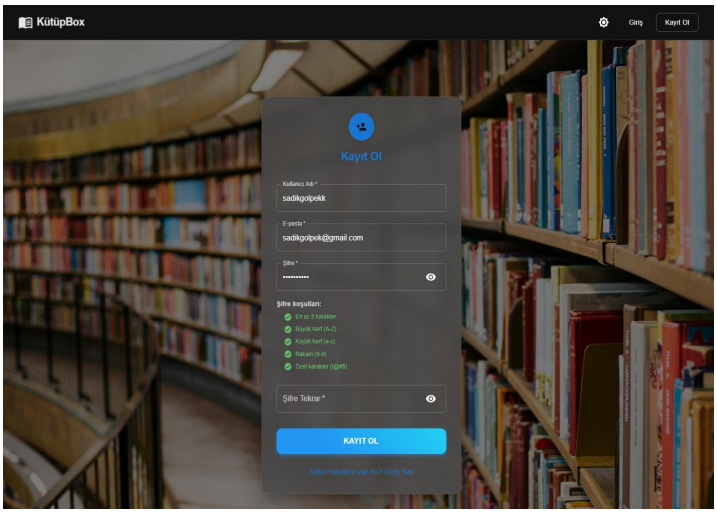
Kullanıcı, tarayıcı üzerinden React arayüzüne ulaşır.

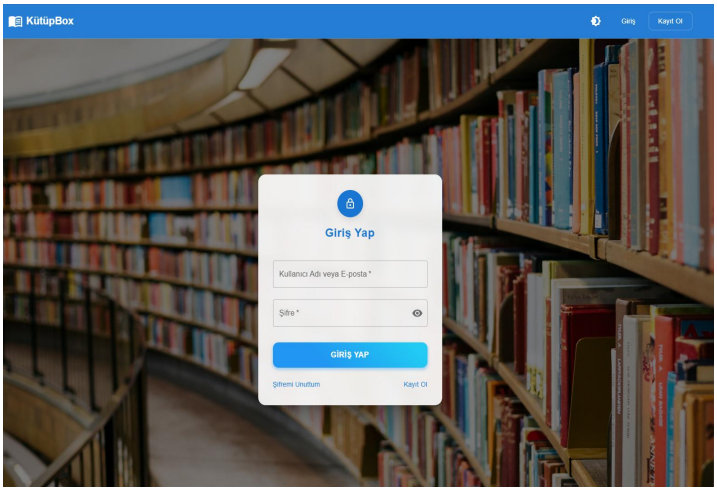
React Router doğru sayfayı yükler (ör. içerik detay, profil, arama).

Arayüzün ihtiyaç duyduğu veri, Axios ile REST API endpoint'lerine gönderilen HTTP istekleri üzerinden alınır.

Backend tarafından dönen JSON verisi React state yönetimi ile işlenir.

Ekran, state güncellemeleri sayesinde dinamik olarak yenilenir ve kullanıcıya anlık etkileşim sunar.




### *B. 2. Backend Akışı (Django REST Framework)*

Her istek öncelikle kimlik doğrulaması (token) açısından kontrol edilir.

Veri, serializer yapıları ile doğrulanır; eksik veya hatalı veri tespit edilirse API uygun hata yanıtı döner.

Doğrulanan veri üzerinde ilgili model işlemleri (kayıt oluşturma, güncelleme, silme, listeleme) Django ORM kullanılarak gerçekleştirilir.

İşlem tamamlandıktan sonra sonuç JSON formatında frontend'e yanıt olarak iletilir.

Backend tamamen REST mimarisine uygun çalışır: stateless, URI tabanlı ve HTTP metodlarına bağlı bir yapıdadır.

### C. *3. Veritabanı Akışı (PostgreSQL + ORM)*

Modeller `models.py` içinde Python sınıfları olarak tanımlanır.

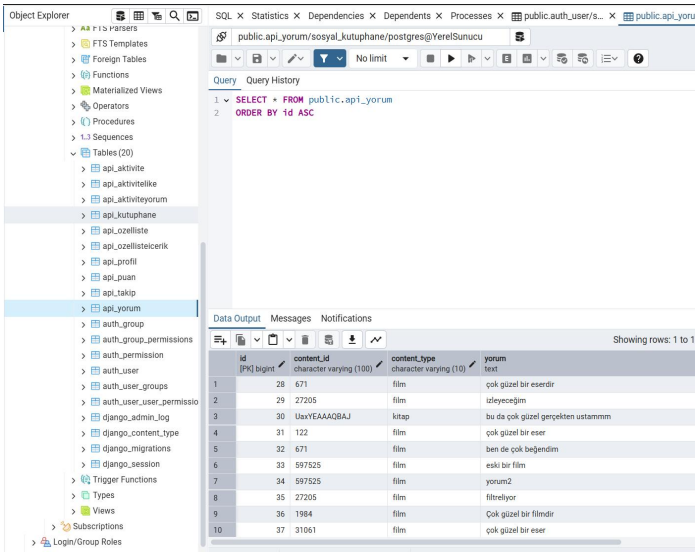
Django migration mekanizması ile tanımlamalar PostgreSQL'e aktarılır:

`python manage.py makemigrations` → değişiklikleri oluşturur

`python manage.py migrate` → veritabanına uygular

Django ORM, Python nesnelerini SQL sorgularına otomatik dönüştürerek güvenli bir veri erişim katmanı oluşturur.

Tüm veriler PostgreSQL üzerinde ilişkisel, tutarlı ve performanslı bir şekilde saklanır.




## IV. SONUÇ

Bu çalışmada, kullanıcıların kitap ve film içeriklerini keşfedebildiği, puanlayabildiği, yorumlayabildiği ve sosyal etkileşim kurabildiği web tabanlı bir Sosyal Kütüphane Platformu başarıyla geliştirilmiştir. Sistem, modern web mimarisinin gerektirdiği şekilde frontend ve backend katmanlarının ayrıldığı, RESTful API prensipleri üzerine kurulu bir yapı sunmaktadır. React tabanlı kullanıcı arayüzü sayesinde dinamik, hızlı ve kullanıcı dostu bir etkileşim sağlanırken; Django REST Framework ile oluşturulan backend katmanı güvenilir veri işleme, yetkilendirme ve API yönetimi gerçekleştirmiştir.

Projenin başlangıcında MSSQL kullanımı hedeflenmiş olsa da, performans ve yapılandırma açısından yaşanan sınırlamalar nedeniyle PostgreSQL'e geçilmiş ve veritabanı göç işlemleri Django ORM üzerinden sorunsuz bir şekilde yönetilmiştir. PostgreSQL'in sunduğu esnek veri tipleri, JSONField desteği ve Django ile doğal uyumu sistemin ölçeklenebilirliğini artırmıştır. Modele dayalı veritabanı tasarımı sayesinde hem içerik yönetimi hem de sosyal etkileşim bileşenleri tutarlı ve bütünleşik bir yapı hâline getirilmiştir.

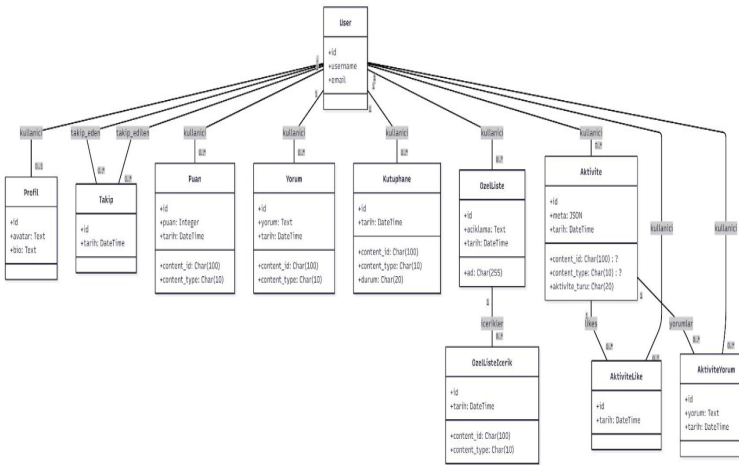
Harici API entegrasyonları (TMDb ve Google Books) platformun veri zenginliğini artırmış, manuel veri girişi gereksinimini ortadan kaldırmıştır. Kullanıcı aktivitelerinin akış (feed) yapısında gösterilmesi; yorum, beğeni, takip ve özel liste mekanizmalarının sorunsuz çalışmasıyla sosyal medya benzeri bir deneyim sağlanmıştır.

Geliştirme sürecinin tamamı Git tabanlı sürüm kontrol sistemi ile yönetilmiş; yapılan commit'ler sayesinde projenin evrimi izlenebilir hâle gelmiştir. Bu yaklaşım hem iş birliğini kolaylaştırmış hem de kod kalitesini artırmıştır.

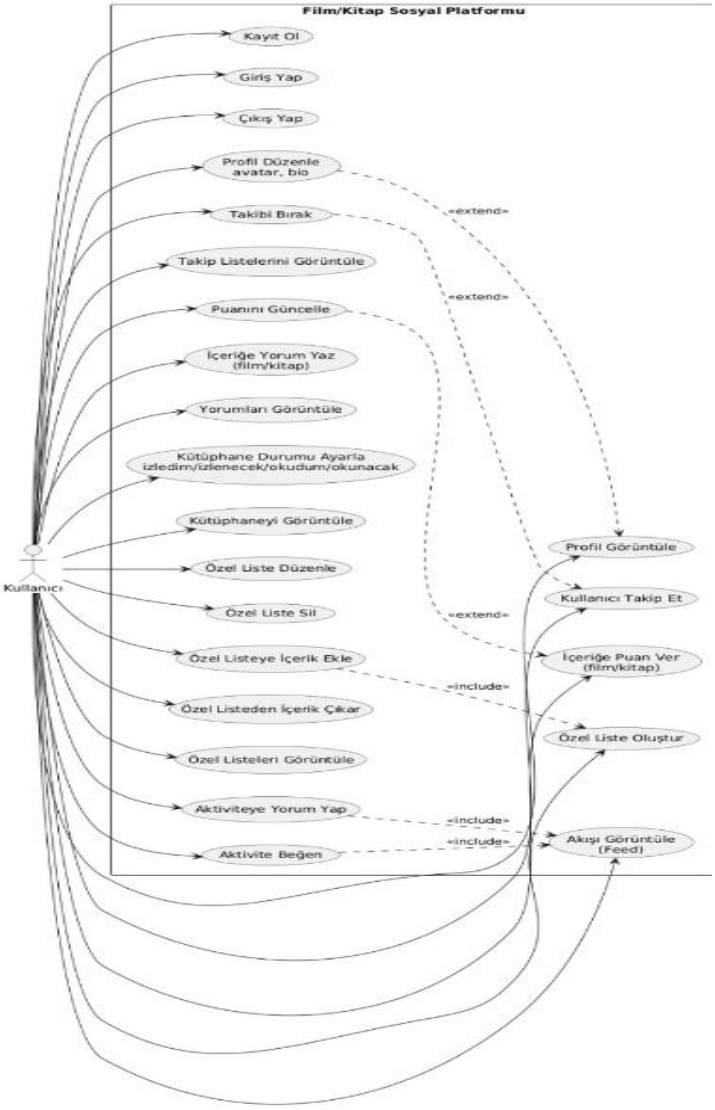
Sonuç olarak geliştirilen sistem; modüler mimarisi, güçlü API yapısı, modern arayüzü ve sağlam veritabanı altyapısıyla işlevsel, genişletilebilir ve gerçek bir sosyal platform nitelikleri taşıyan başarılı bir web uygulaması ortaya koymuştur. Gelecek çalışmalarda gerçek zamanlı bildirim sistemi, öneri algoritmaları (recommendation system), sohbet modülü veya mobil uygulama entegrasyonu gibi özelliklerle platformun daha da geliştirilmesi mümkündür.

## V. CLASS DİYAGRAMLARI VE USE CASE DİYAGRAMLARI

Class Diyagramı




Use Case diyagramı




## VI. KAYNAKÇA


Class diyagramı için https://mermaid.js.org/

Use Case diyagramı için https://www.planttext.com/

Django https://www.djangoproject.com/

ChatGPT https://chatgpt.com/

Veritabanı için https://www.postgresql.org/

Gemini https://gemini.google.com/

React Kütüphanesi https://tr.react.dev/