Çözüm Mimarisi

Bu doküman projenin mimari bakışını, kullanılan teknolojileri, hangi problemi çözdüğünü ve **RAG (Retrieval-Augmented Generation)** yaklaşımının nasıl uygulandığını açıklar.

Kullandığımız Teknolojiler

- Python 3.11
- LangGraph — workflow / state-graph motoru (düğümler ve geçişlerin yönetimi)
- LangChain modülleri (modüler bileşenler): langchain_core, metin bölme, provider adaptörleri
- HuggingFace Embeddings (langchain_huggingface) — Türkçe için önceden eğitilmiş BERT tabanlı embedding
- Chroma (langchain_community.vectorstores.Chroma) — lokal vektör veritabanı
- OpenRouter/OpenAI uyumlu LLM adapter (projedeki open_router.py ile sarılmış ChatOpenAI sınıfı)
- Streamlit — hızlı web arayüzü (frontend)
- ReportLab — dokümantasyon PDF üretimi (yardımcı araç)
- tqdm, pydantic vb. yardımcı kütüphaneler

Çözmeye Çalıştığımız Problem

Kullanıcıların Türkçe doğal dilde sorduğu film sorularına (ör. öneri, özet, inceleme analizi) bağlamsal ve doğru yanıtlar vermek.

Zorluklar:

- Açıklamalar, kısa özetler ve değişken kalitede kullanıcı yorumları
- Türkçe dilinde yüksek kaliteli embedding ve LLM yetenekleri
- Gerçek zamanlı kullanıcı sorgularına hızlı yanıt

Mimari Yaklaşım (Genel)

1. Veri Ingest ve Indexleme

- ingestion.py ile all_movies_reviews.json dosyasındaki film açıklamaları ve kullanıcı yorumları **Document** olarak hazırlanır.
- RecursiveCharacterTextSplitter ile metinler parçalara bölünür.
- Her parça için HuggingFaceEmbeddings kullanılarak embedding üretilir.
- Embedding'ler **Chroma**'ya kaydedilir ve ./.chroma/movie dizininde persist edilir.

2. Query Yönlendirme (Routing)

- Kullanıcı mesajı önce question_router aracılığıyla sınıflandırılır (FILM_QUERY veya GENERAL_CHAT).
- Film sorgusu ise retriever üzerinden ilgili parçalar çekilir; genel sohbet ise doğrudan LLM'e iletilir.

3. RAG (Retrieval-Augmented Generation)

- Retriever (Chroma) ilgili bağlam parçalarını getirir (örneğin k=6).
- Getirilen parçalar prompt içinde LLM'e verilir.
- LLM (ChatOpenRouter) bağlamı kullanarak net, kısa ve güvenilir cevap üretir.
- Bu yöntem, modelin halüsinasyon üretme olasılığını azaltır ve bağlama dayalı yanıt üretimini güçlendirir.

4. Workflow / StateGraph

- langgraph ile **MovieRetrieve**, **Generate**, **GeneralChat** gibi düğümler tanımlanır ve akış yönlendirilir.
- GraphState nesnesi boyunca context, message, answer gibi bilgiler taşınır.

Veri Akışı (Örnek)

Kullanıcı → Streamlit UI → GraphState(message) → detect_intent →
(MovieRetrieve → Generate) **veya** (GeneralChat) → Sonuç (UI)

Ölçeklenebilirlik ve Dağıtım Önerileri

- Chroma yerine daha büyük üretim iş yükü için **Weaviate**, **Milvus** veya yönetilen **Chroma/Chromadb** hizmetleri tercih edilebilir.
- Embedding üretimi ve ingest işlemleri **Celery**, **Prefect** veya **Airflow** ile batch-pipeline olarak paralelleştirilebilir.
- LLM çağrıları için **cache**, **rate-limiting** ve **fallback** stratejileri uygulanmalıdır.

Güvenlik ve Gizli Anahtar Yönetimi

- API anahtarları (OPENROUTER_API_KEY vb.) .env dosyasında saklanmalı, repoya eklenmemelidir.
- Gerektiğinde anahtar rotasyonu ve erişim denetimi uygulanmalıdır.