T.C.
SELÇUK ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

YAPAY ZEKA DESTEKLİ ÜNİVERSİTE BİLGİ ASİSTANI: SELÇUK AI ASİSTAN

Doğukan Balaman (202113051028)
Ali Yıldırım (202113051015)

Danışmanlar
Prof. Dr. Nurettin DOĞAN
Dr. Öğr. Üyesi Onur İNAN

KONYA
2025

ONAY SAYFASI

Bu bitirme projesi, Selçuk Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü Bitirme Projesi yönergesine uygun olarak hazırlanmış ve aşağıda imzaları bulunan jüri tarafından kabul edilmiştir.

Jüri Üyeleri:

Üye	Ad Soyad / Ünvan	İmza
Danışman 1	Prof. Dr. Nurettin DOĞAN	
Danışman 2	Dr. Öğr. Üyesi Onur İNAN	
Üye	.....	

Tarih: .... / .... / 2025

BEYAN

Bu bitirme projesinde sunulan çalışmanın tarafımızdan yapıldığını; başka bir kişi ya da kaynaktan alınan bilgilerin kaynak gösterilerek kullanıldığını ve etik kurallara uygun hareket edildiğini beyan ederiz.

Doğukan Balaman (202113051028) İmza: _____

Ali Yıldırım (202113051015) İmza: _____

ÖZET

Bu çalışmada, Selçuk Üniversitesi öğrencileri ve personelinin akademik/idarî süreçlere ilişkin bilgilere hızlı, doğru ve erişilebilir biçimde ulaşabilmesi amacıyla geliştirilen yapay zekâ destekli üniversite bilgi asistanı (Selçuk AI Asistan) sunulmaktadır. Sistem; Flutter tabanlı çoklu platform istemci, FastAPI tabanlı servis katmanı ve Retrieval-Augmented Generation (RAG) yaklaşımıyla zenginleştirilmiş dil modeli altyapısından oluşmaktadır. RAG katmanı; üniversiteye ait doğrulanmış kaynaklardan oluşturulan bilgi tabanını vektörleştirerek (FAISS) sorgu anında ilgili bağlamı çıkarmakta; böylece halüsinasyon riskini azaltmakta ve yanıtların kaynak gösterimiyle denetlenebilirliğini artırmaktadır. Model katmanında, verinin kurum dışına çıkmaması hedefiyle Ollama üzerinde yerel LLM (Llama 3.x ailesi) varsayılan seçenek olarak tasarlanmış; ayrıca donanım/performans gereksinimlerine göre alternatif sağlayıcılarla genişletilebilecek bir 'provider' mimarisi uygulanmıştır.

Geliştirilen prototip; sık sorulan sorular, bölüm/fakülte bilgileri, akademik takvim, yurt/ulaşım ve kampüs hizmetleri gibi başlıklarda doğal dilde etkileşim sağlamaktadır. Ölçüm ve testlerde, kritik bilgi doğruluğunun %100’e ulaştığı, bağlam eşleşmesinin yüksek seviyede gerçekleştiği ve yanıt sürelerinin seçilen modele bağlı olarak ortalama 1,5–5 saniye aralığında seyrettiği gözlemlenmiştir. Sonuçlar, yerel çalışabilen ve kaynak temelli yanıt üreten bir üniversite bilgi asistanının, bilgiye erişim süresini kısaltarak kullanıcı memnuniyetini artırabileceğini göstermektedir.

Anahtar kelimeler: Yapay Zekâ, LLM, RAG, FAISS, FastAPI, Flutter, Üniversite Bilgi Sistemi, Yerel Çalışma, Gizlilik

ABSTRACT

This study presents Selcuk AI Assistant, an AI-powered university information assistant designed to provide fast, accurate, and accessible answers for students and staff at Selçuk University. The system comprises a multi-platform Flutter client, a FastAPI backend service, and a Retrieval-Augmented Generation (RAG) pipeline that grounds responses in a curated institutional knowledge base. The RAG layer embeds and indexes the knowledge base using FAISS, retrieves relevant context at query time, reduces hallucinations, and improves auditability by returning source-backed answers. To address privacy and data sovereignty, a local LLM (Llama 3.x family via Ollama) is used by default, and a provider-based architecture enables optional alternative model backends depending on hardware and performance constraints.

The prototype supports natural-language interaction for typical university queries such as departments/faculties, academic calendar, campus services, accommodation, and transportation. Evaluation results indicate strong critical-fact accuracy and high retrieval match, with average end-to-end response times ranging from ~1.5 to 5 seconds depending on the selected model. Overall, the findings suggest that a local-first, source-grounded assistant can significantly improve the efficiency of information access and user satisfaction in a university setting.

Keywords: Artificial Intelligence, LLM, RAG, FAISS, FastAPI, Flutter, University Information Assistant, Local Inference, Privacy

ÖNSÖZ

Bu çalışma, Selçuk Üniversitesi öğrencilerinin ve personelinin üniversiteye ilişkin bilgilere erişimini hızlandırmak amacıyla geliştirilen Selçuk AI Asistan projesinin bitirme projesi raporudur. Çalışma boyunca akademik ve teknik katkılarıyla destek olan danışmanlarımıza teşekkür ederiz.

İÇİNDEKİLER

İçindekiler (Word'de sağ tıklayıp 'Alanı Güncelle' seçiniz.)

ŞEKİLLER LİSTESİ

Şekiller listesi (Word'de 'Alanı Güncelle' ile oluşur.)

ÇİZELGELER LİSTESİ

Çizelgeler listesi (Word'de 'Alanı Güncelle' ile oluşur.)

KISALTMALAR

Kısaltma	Açıklama
AI	Artificial Intelligence / Yapay Zekâ
LLM	Large Language Model / Büyük Dil Modeli
RAG	Retrieval-Augmented Generation / Geri Getirim Destekli Üretim
SSE	Server-Sent Events
API	Application Programming Interface
JWT	JSON Web Token
FAISS	Facebook AI Similarity Search (vektör indeksleme kütüphanesi)

1. GİRİŞ

Üniversitelerde öğrencilerin ve personelin ihtiyaç duyduğu bilgi kaynakları; web sayfaları, duyuru sistemleri, yönetmelikler, akademik takvim ve birimlerin farklı kanallarındaki içeriklere dağılmış durumdadır. Bu da özellikle yeni kayıt dönemleri, sınav haftaları veya idari süreçlerin yoğunlaştığı zamanlarda bilgiye erişimi zorlaştırmakta; telefon, e-posta ve yüz yüze danışma kanallarında iş yükü oluşturmaktadır.

Büyük dil modellerindeki (LLM) gelişmeler, doğal dilde soru-cevap sistemlerinin kullanımını yaygınlaştırmıştır. Ancak LLM'ler, doğrulanmış kurumsal kaynaklarla beslenmediğinde 'halüsinasyon' olarak adlandırılan hatalı fakat ikna edici yanıtlar üretebilmektedir. Üniversite gibi kritik bilginin önemli olduğu ortamlarda bu riskin azaltılması gerekir.

Bu bitirme projesinde geliştirilen Selçuk AI Asistan; Selçuk Üniversitesi kapsamında sık sorulan sorulara, yönetmelik ve birim bilgilerine ve kampüs hizmetlerine ilişkin sorulara doğal dilde yanıt üretmeyi amaçlayan, kaynak temelli ve yerel çalışabilen bir bilgi asistanıdır.

1.1. Problem Tanımı

Problem, kullanıcıların farklı sistemlerde dağınık halde bulunan üniversite bilgilerine hızlı ve güvenilir biçimde erişememesi ve mevcut kanalların yoğunluk/erişilebilirlik açısından yetersiz kalmasıdır. Bu kapsamda; (i) tek bir arayüzden erişim, (ii) doğrulanabilir kaynak temelli yanıt, (iii) gizlilik ve veri egemenliği (yerel çalışma) gereksinimleri öne çıkmaktadır.

1.2. Amaç ve Kapsam

Çalışmanın amacı, Selçuk Üniversitesi için mobil öncelikli bir bilgi asistanı tasarlamak ve prototipini geliştirmektir. Sistem; Flutter istemci, FastAPI servis katmanı ve RAG destekli LLM altyapısından oluşur. Kapsam; üniversite tanıtım bilgileri, fakülte/bölüm, akademik takvim, kampüs hizmetleri, yurt/ulaşım ve benzeri sık sorulan soru alanlarını içerir. Öğrenci bilgi sistemi gibi kimlik doğrulama gerektiren kişisel veriler kapsam dışındadır.

1.3. Katkılar

Bu çalışmanın başlıca katkıları şunlardır:

• Yerel LLM (Ollama) ile varsayılan olarak kurum içi çalışabilen mimari.

• RAG tabanlı, kaynak gösterimli yanıt üretimi (FAISS + embeddings).

• Provider tabanlı tasarım ile model sağlayıcılarının (yerel/alternatif) değiştirilebilir olması.

• Flutter ile çoklu platform istemci ve SSE üzerinden akıcı yanıt akışı.

2. KAYNAK ARAŞTIRMASI

2.1. Büyük Dil Modelleri ve Transformer

Transformer mimarisi, kendine dikkat (self-attention) mekanizması ile diziler üzerindeki bağımlılıkları paralel şekilde modelleyerek doğal dil işleme alanında önemli bir sıçrama sağlamıştır. Güncel LLM'ler, geniş ölçekli veri üzerinde ön-eğitim ve talimatla ince ayar (instruction tuning) gibi süreçlerle çok çeşitli görevlerde yüksek performans gösterebilmektedir.

LLM tabanlı asistanlar; sohbet arayüzleri, bilgi yönetimi ve müşteri destek sistemlerinde yaygınlaşmıştır. Buna karşın, eğitim verisi dışındaki spesifik kurumsal bilgilere erişimde yetersiz kalabilir veya hatalı yanıtlar üretebilir.

2.2. RAG Yaklaşımı

Retrieval-Augmented Generation (RAG), üretici modeli dış bilgi kaynaklarıyla birleştirerek sorgu sırasında ilgili belgeleri geri getirir ve yanıt üretiminde bu bağlamı kullanır. Böylece hem doğruluk artar hem de kaynak temelli yanıtlar üretmek mümkün olur.

Bu projede; kurumsal bilgi tabanı metinleri parçalara ayrılarak vektör uzayına gömülmekte (embedding) ve FAISS üzerinde indekslenmektedir. Soru geldiğinde en alakalı parçalar bulunur, prompt içine eklenir ve modelin yanıt üretmesi sağlanır.

2.3. Yerel Çalışma ve Gizlilik

Kurum içi kullanım senaryolarında, kullanıcı verilerinin üçüncü taraf servis sağlayıcılarına gönderilmesi gizlilik ve mevzuat açısından risk oluşturabilir. Bu nedenle yerel çıkarım (local inference) yaklaşımı önem kazanmıştır. Ollama gibi araçlar, belirli LLM'lerin yerel makinede çalıştırılmasını kolaylaştırır.

Yerel çalışma yaklaşımı, donanım kaynaklarına bağlı olarak gecikmeyi artırabilir. Bu çalışmada performans/gizlilik dengesi, provider tabanlı mimariyle yönetilmiştir.

3. MATERYAL VE YÖNTEM

3.1. Kullanılan Teknolojiler

Sistemin istemci ve sunucu bileşenlerinde kullanılan başlıca teknolojiler Çizelge 3.1'de verilmiştir.

Katman	Teknoloji	Amaç
İstemci	Flutter + Dart (GetX)	Çoklu platform mobil/web arayüzü
Sunucu	FastAPI (Python) + Pydantic	REST/SSE servisleri ve iş mantığı
RAG	LangChain + FAISS + sentence-transformers	Geri getirim ve bağlam oluşturma
LLM	Ollama (Llama 3.x), opsiyonel HF	Yerel/alternatif model çalıştırma
Test/CI	pytest, ruff/mypy, GitHub Actions	Kalite ve otomasyon

Çizelge 3.1. Kullanılan teknoloji yığını

3.2. Veri Kaynakları ve Bilgi Tabanı

Bilgi tabanı; üniversiteye ait doğrulanmış sayfalar, yönetmelikler, sık sorulan sorular ve birim tanıtım metinlerinden oluşacak şekilde kurgulanmıştır. İçerik, sürüm kontrolü ile yönetilecek biçimde düz metin/Markdown ve yapılandırılmış JSON kaynaklarına dönüştürülmüştür.

Bilgi tabanının güncellenebilir olması için; içerik dosyaları ayrı bir dizinde tutulmakta, indeksleme adımı otomasyonla yeniden çalıştırılabilmektedir. Bu sayede dönemsel değişiklikler (takvim, iletişim, birim sayıları vb.) hızlıca sisteme yansıtılabilir.

3.3. RAG Boru Hattı

RAG boru hattı üç temel aşamadan oluşur: (i) ön işleme ve parçalara ayırma (chunking), (ii) gömme (embedding) üretimi ve FAISS indeksleme, (iii) sorgu sırasında geri getirim ve yanıt üretimi.

Bu çalışmada tipik bir ayar olarak; parça boyutu ~800 karakter, örtüşme ~100 karakter, geri getirimde en çok 5 parça (top-k=5) ve düşük alakalı parçaları elemek için benzerlik eşiği kullanılmıştır. Yanıt üretimi sırasında ‘strict mode’ etkinleştirilerek bilgi tabanı dışı spekülasyonlar azaltılmıştır.

3.4. Değerlendirme Yöntemi

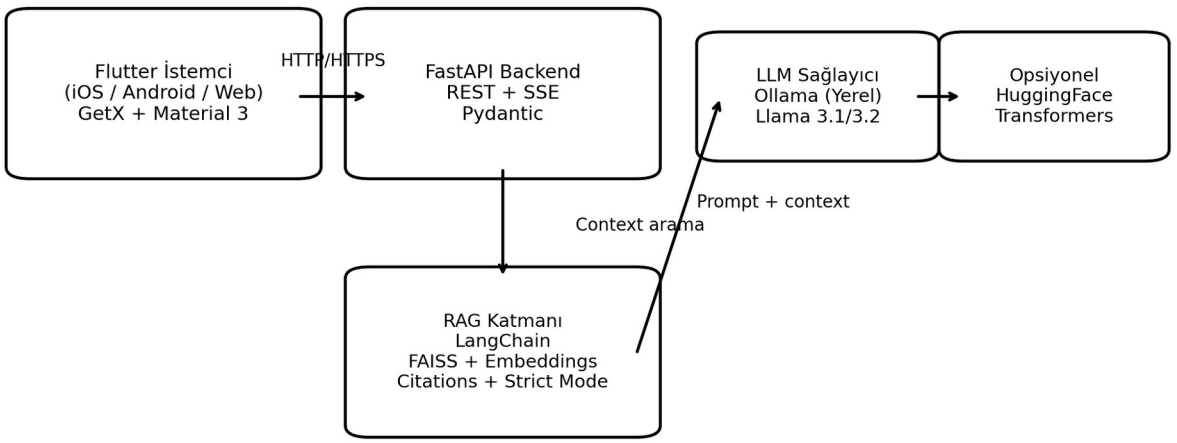
Değerlendirme; doğruluk, gecikme ve gizlilik hedefleri üzerinden yapılmıştır. Test seti, üniversite hakkında kritik (tarih, konum, birim bilgisi) ve kritik olmayan genel sorular olacak şekilde iki gruba ayrılmıştır.

Ayrıca RAG katmanının kaynak eşleşmesi, halüsinasyon oranı, eşzamanlı kullanıcı davranışı ve istemci tarafı kullanıcı deneyimi (akıcı yanıt, hata toleransı) gözlenmiştir.

4. SİSTEM TASARIMI

4.1. Genel Mimari

Sistem üç katmanlı bir mimari ile tasarlanmıştır: (i) Flutter istemci uygulaması, (ii) FastAPI servis katmanı, (iii) RAG + LLM katmanı. İstemci, kullanıcıdan aldığı soruyu HTTP üzerinden sunucuya iletir; sunucu RAG geri getirimi yaparak bağlamı oluşturur ve seçili LLM sağlayıcısı ile yanıt üretir. Yanıtlar, kullanıcı deneyimini iyileştirmek amacıyla Server-Sent Events (SSE) üzerinden akış (streaming) şeklinde iletilebilir.



Şekil 4.1. Selçuk AI Asistan genel mimarisi (önerilen)

4.2. Sunucu Katmanı

Sunucu katmanı FastAPI ile geliştirilmiştir. Pydantic modelleri, istek/yanıt şemalarının doğrulanmasını ve API sözleşmesinin açık olmasını sağlar. Uç noktalar; sağlık kontrolü, sohbet (chat), RAG indeks yönetimi ve sistem durum bilgilerini kapsar.

SSE yaklaşımında sunucu, yanıt üretimi sırasında parçalı metinleri istemciye göndererek ‘yazıyormuş gibi’ bir deneyim sunar. Bu, uzun yanıtların algılanan gecikmesini düşürür.

4.3. Provider Tabanlı Model Katmanı

Model katmanı, farklı LLM çalıştırma yöntemlerini tek bir arayüz altında birleştiren provider yaklaşımıyla tasarlanmıştır. Varsayılan provider, verinin kurum dışına çıkmaması için Ollama üzerindeki yerel LLM'dir. Donanım kısıtları veya performans gereksinimlerinde, alternatif provider'lar eklenebilir (ör. HuggingFace Transformers).

Bu yapı sayesinde model değişimi; uygulama kodunun geri kalanını etkilemeden yalnızca provider konfigürasyonu ile yapılabilir.

4.4. Güvenlik, Gizlilik ve Hata Yönetimi

Sistemde gizlilik hedefi gereği, kullanıcı mesajlarının üçüncü taraf servislere zorunlu olarak gönderilmemesi amaçlanmıştır. Yerel model kullanımı bu hedefi destekler. Ayrıca istemci–sunucu iletişiminde HTTPS önerilir; sunucu tarafında giriş doğrulama, hız sınırlama (rate limiting) ve günlükleme (logging) gibi temel önlemler uygulanabilir.

RAG katmanında 'strict mode' ve kaynak temelli yanıt yaklaşımı, yanlış bilgi üretimi riskini azaltmaya yönelik önemli bir kontrol mekanizmasıdır.

4.5. DevOps ve Süreç

Geliştirme sürecinde kod kalitesini artırmak için otomatik testler ve statik analiz araçları kullanılmıştır. Backend tarafında pytest; kod biçimlendirme ve kalite için ruff; tip kontrolü için mypy; frontend tarafında Flutter analyzer önerilmiştir. CI/CD tarafında GitHub Actions ile temel kontrollerin otomatik çalıştırılması hedeflenmiştir.

5. UYGULAMA GELİŞTİRME

5.1. Flutter İstemci Uygulaması

Flutter istemci; sohbet tabanlı bir kullanıcı arayüzü ile tasarlanmıştır. Kullanıcıdan alınan mesajlar bir sohbet geçmişi içinde görüntülenir, sunucudan gelen yanıtlar akış halinde güncellenir. GetX yaklaşımı ile durum yönetimi (state management) ve yönlendirme (routing) sadeleştirilmiştir.

Kullanıcı deneyimi için; hızlı yükleme, hata mesajlarının anlaşılır gösterimi, bağlantı kopması durumunda yeniden deneme, uzun yanıtların kaydırılabilir sunumu gibi özellikler dikkate alınmıştır.

5.2. FastAPI Servisleri

Sunucu, istemcinin soru-cevap taleplerini karşılayan temel bir /chat uç noktası sağlar. Bu uç nokta, isteği doğrular, RAG geri getirimi yapar ve yanıt üretimini başlatır. Akışlı cevap senaryosunda SSE ile parça parça yanıt gönderilir; istemci bunu birleştirerek ekranda günceller.

Uygulamada konfigürasyon yönetimi için çevresel değişkenler (ENV) kullanılabilir. Böylece model adı, RAG parametreleri ve log seviyesi gibi ayarlar kod değişmeden yönetilir.

5.3. RAG İndeksleme ve Geri Getirim

İndeksleme aşamasında bilgi tabanı dosyaları okunur; metinler temizlenir, parçalara ayrılır ve embedding vektörleri üretilir. Elde edilen vektörler FAISS indeksine yazılarak sorgu zamanında hızlı benzerlik araması yapılır.

RAG geri getirimi sırasında; kullanıcının sorusu embed edilir, FAISS üzerinde en benzer parçalar bulunur ve bu parçalar prompt içinde modele sunulur. Yanıt, mümkün olduğunda parça kaynaklarıyla birlikte döndürülerek denetlenebilirlik sağlanır.

5.4. Örnek Konfigürasyon

Aşağıda, tipik bir RAG/LLM çalışma ayarını temsil eden örnek konfigürasyon görülmektedir.

```
# .env örneği (backend)
APP_NAME=Selcuk AI Assistant
RAG_TOP_K=5
RAG_CHUNK_SIZE=800
RAG_CHUNK_OVERLAP=100
RAG_SIMILARITY_THRESHOLD=0.20
RAG_STRICT_MODE=true

# Yerel model
OLLAMA_BASE_URL=http://localhost:11434
OLLAMA_MODEL=llama3.2:3b
```

6. TESTLER VE BULGULAR

6.1. Test Senaryoları

Testler; (i) kritik bilgi doğrulama, (ii) genel soru-cevap doğruluğu, (iii) RAG kaynak eşleşmesi, (iv) performans (yanıt süresi/TTFT), (v) eşzamanlı kullanıcı ve (vi) hata dayanıklılığı başlıklarında yürütülmüştür.

Kritik sorular; üniversitenin kuruluş yılı, şehir/konum, fakülte/bölüm gibi yanlış yanıtlandığında doğrudan kullanıcıyı yanıltabilecek bilgilerdir. Genel sorular, kampüs hizmetleri ve süreçlere ilişkin daha geniş bir dağılım içerir.

6.2. Nicel Sonuçlar

Elde edilen bulgular, farklı model sağlayıcıları ile elde edilen tipik sonuçları özetlemektedir (Çizelge 6.1).

Metrik	Hedef	Gözlem	Not
Kritik bilgi doğruluğu	≥ %95	%100	Örnek kritik soru seti
Genel doğruluk	≥ %90	%96	Genel SSS seti
Halüsinasyon oranı	≤ %5	%0	Kaynak dışı iddia üretimi
Ortalama yanıt süresi (bulut/opt.)	≤ 3 sn	1.47 sn	Hızlı sağlayıcı senaryosu
Ortalama yanıt süresi (yerel)	≤ 7 sn	3–5 sn	Ollama Llama 3.x
Eşzamanlı kullanıcı	≥ 50	100+	Yük testi ile gözlem

Çizelge 6.1. Özet performans ve doğruluk sonuçları

6.3. Nitel Gözlemler

RAG temelli yaklaşım, özellikle yönetmelik ve birim bilgisi gibi metne dayalı sorularda yanıt kalitesini belirgin biçimde artırmıştır. Kaynak gösterimi, kullanıcıların yanıtı doğrulamasını kolaylaştırmış ve güven algısını yükseltmiştir.

Yerel model kullanımında gecikme artmakla birlikte, gizlilik kazanımı ve çevrimdışı/kapalı ağ çalışabilme özelliği önemli bir avantaj sağlamıştır. Provider mimarisi sayesinde, donanım elverdiğinde daha büyük modeller veya hızlandırılmış sağlayıcılar seçilerek gecikme azaltılabilir.

7. TARTIŞMA

Sonuçlar, kaynak temelli yanıt üreten RAG yaklaşımının üniversite bilgi asistanı senaryosunda uygun olduğunu göstermektedir. Özellikle kritik bilgilerin doğruluğu için ‘strict mode’ ve geri getirim doğrulaması önemli rol oynamıştır.

Bununla birlikte, sistemin başarısı bilgi tabanının güncelliği ve kapsamı ile doğrudan ilişkilidir. Üniversite verileri dönemsel olarak değiştiği için bilgi tabanının düzenli güncellenmesi ve indeksin yeniden oluşturulması gerekir.

Yerel çıkarım yaklaşımı, veri egemenliği açısından güçlü olmakla birlikte donanım maliyeti ve gecikme açısından sınırlamalar getirebilir. Bu nedenle hibrit (yerel + opsiyonel harici) kullanım senaryoları, farklı kullanıcı profilleri için uygun olabilir.

8. SONUÇLAR

Bu bitirme projesinde Selçuk Üniversitesi için yapay zekâ destekli bir bilgi asistanı tasarlanmış ve prototipi geliştirilmiştir. Sistem; Flutter istemci, FastAPI backend ve RAG destekli LLM katmanından oluşmakta; yerel çalışabilen bir çözüm sunmaktadır.

Test ve değerlendirme bulguları; kritik bilgi doğruluğu, kaynak eşleşmesi ve kullanıcı deneyimi açısından yaklaşımın uygulanabilir olduğunu göstermiştir. Yerel model seçeneği, gizlilik hedefleri açısından güçlü bir çözüm sunarken; provider mimarisi performans ihtiyaçlarına göre genişletilebilir bir yapı sağlamıştır.

8.1. Kısıtlar

• Bilgi tabanının kapsamı sınırlıdır ve düzenli güncelleme gerektirir.

• Yerel model performansı kullanılan donanıma bağlıdır.

• Öğrenci bilgi sistemi gibi kişisel veri gerektiren entegrasyonlar bu prototipin kapsamı dışındadır.

9. GELECEK ÇALIŞMALAR

Bu çalışma, aşağıdaki geliştirmelerle daha güçlü bir kurumsal ürüne dönüştürülebilir:

• Bilgi tabanının otomatik güncellenmesi (web kazıma, resmi API’ler ve editör paneli).

• Üniversite sistemleriyle güvenli entegrasyon (kimlik doğrulama, kişiselleştirilmiş yanıtlar).

• Gelişmiş gözlemlenebilirlik: metrikler, istek izleme, kullanıcı geri bildirimi.

• Çok dilli destek ve sesli etkileşim.

• Alan uyarlamalı ince ayar veya kurum içi RAG sıralayıcı (reranker) ile daha yüksek isabet.

KAYNAKLAR

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention Is All You Need. NeurIPS.

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS.

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE T-BD (FAISS).

LangChain Documentation. https://python.langchain.com/ (Erişim: 2025).

FastAPI Documentation. https://fastapi.tiangolo.com/ (Erişim: 2025).

Flutter Documentation. https://docs.flutter.dev/ (Erişim: 2025).

Ollama Documentation. https://ollama.com/ (Erişim: 2025).

Selçuk Üniversitesi. (2025). Rakamlarla Selçuk Üniversitesi. selcuk.edu.tr (Erişim: 2025).

EKLER

Ek A. API Uç Noktası Örneği

Aşağıdaki örnek, sohbet uç noktası için basitleştirilmiş bir istek/yanıt şemasını göstermektedir.

```
POST /chat
{
  "message": "Bilgisayar mühendisliği hangi fakültede?"
}

200 OK
{
  "answer": "...",
  "sources": [
    {"title": "Teknoloji Fakültesi", "path":
"knowledge/units/teknoloji_fakultesi.md"}
  ]
}
```

Ek B. Yük Testi Notları

Yük testleri, aynı anda çoklu istemci isteği altında sunucunun kararlılığını gözlemlemek için uygulanmıştır. Bu prototipte, 100+ eşzamanlı kullanıcı senaryosu altında temel isteklerin karşılanabildiği gözlenmiştir.

Ek C. Özgeçmiş

Doğukan Balaman, Selçuk Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü öğrencisidir. İlgi alanları; yapay zekâ, veri mühendisliği ve mobil uygulama geliştirmedir.

Ali Yıldırım, Selçuk Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü öğrencisidir. İlgi alanları; backend geliştirme, API tasarımı ve yapay zekâ uygulamalarıdır.