6. Sistem Tasarımında Kullanılacak Araçlar

Bu bölümde kullanılan teknolojiler, prototip ve profesyonel (production) geliştirme süreçleri için iki ayrı kategori hâlinde ele alınmıştır. Projenin başarısı açısından prototip aşaması kritik öneme sahiptir; çünkü bu aşamada sistemin eksikleri tespit edilmekte, riskler görülmekte ve çözümler hızlı bir şekilde geliştirilebilmektedir. Bu nedenle çalışmanın odak noktası, temel fonksiyonların doğrulandığı ve çalışır durumda bir prototipin geliştirilmesidir.

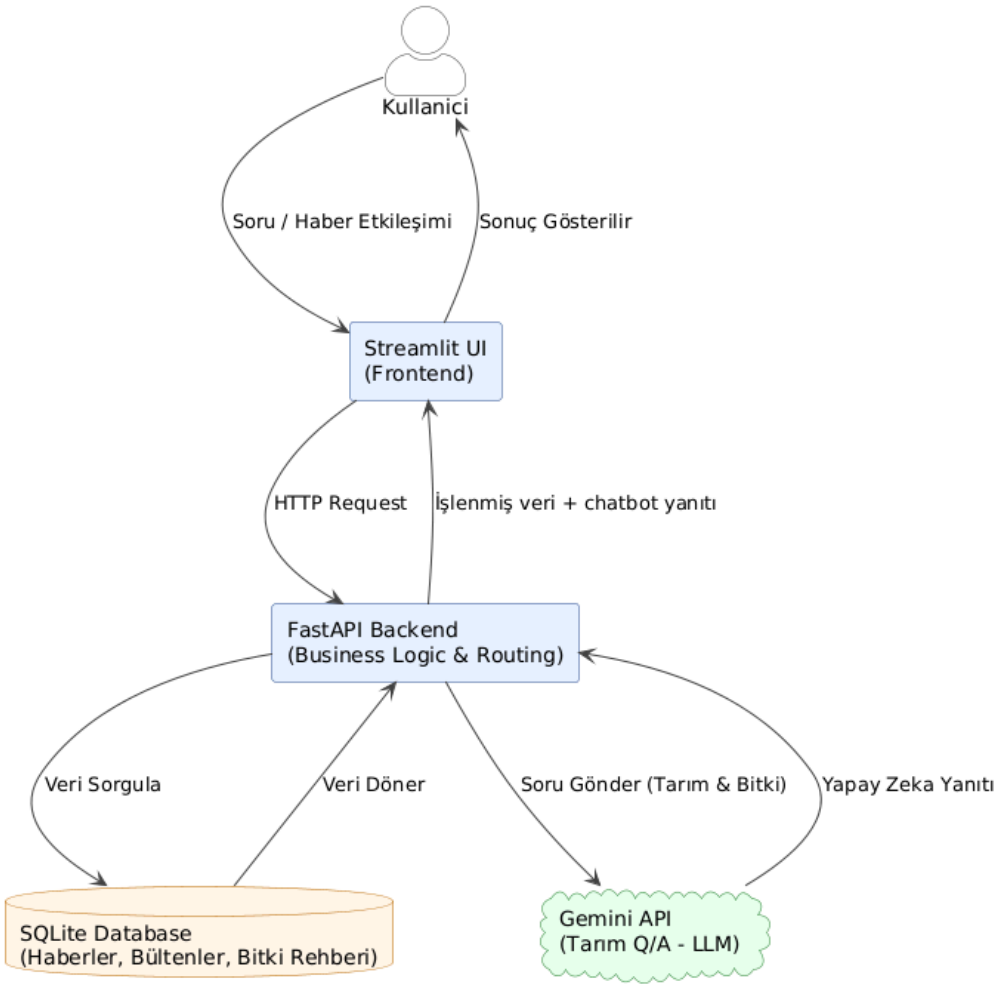
Bileşen / Katman	Prototipte Kullanılan Araçlar	Gerçek (Production) Projede Kullanılacak Araçlar	Açıklama / Fark
Frontend (UI)	Streamlit	React / React Native / Flutter	Prototipte hız önemli; gerçek projede performans, animasyon, çoklu ekran desteği gerekli.
Backend API	FastAPI	FastAPI, SQLAlchemy / Node.js (NestJS) / Django REST	Prototipte tek servis yeterli; gerçek projede mikroservis mimarisi uygulanabilir.
Veritabanı	SQLite	PostgreSQL (+ Redis Cache)	SQLite dosya tabanlıdır ve python tarafından desteklidir; gerçek projede ölçeklenebilirlik, ACID garantisi ve indeksleme gerekir.
AI / Chatbot	Gemini API veya OpenAI API	Gemini Pro / Gemma modelleri (Google AI Studio) veya OpenAI Assistants	Prototipte direkt API çağrısı; gerçek projede daha gelişmiş modüller ve hata yönetimi gerekir.
Veri Toplama	Python + Requests + BeautifulSoup	Airflow, Cloud Scheduler, Pub/Sub, ETL pipelines	Prototipte manuel çekim yeterli; gerçek projede otomatik veri akışı ve zamanlanmış işler gerekir.
Sunucu / Hosting	Lokal çalışma veya Streamlit Cloud	Google Cloud Run / AWS ECS / Azure App Service	Gerçek projede autoscaling ve kesintisiz çalışma gereklidir.

Deployment	Minimum — tek script	Docker + Kubernetes / Container Registry	Production ortamında taşınabilirlik, yük dengeleme ve güvenlik ön planda.
CI/CD	yok (manuel)	GitHub Actions / GitLab CI / Jenkins	Gerçek projede her push sonrası otomatik test + deployment gerekir.
Authentication	Basit token / yok	OAuth2, JWT, Firebase Auth, Auth0	Prototipte gerekmez; production'da güvenlik zorunludur.
Monitoring	yok	Prometheus + Grafana / Google Cloud Monitoring	Canlı sistemde loglama ve performans takibi şarttır.
Dosya & Depolama	Lokal klasör	Cloud Storage (GCS/S3/Azure Blob)	Yüksek trafik için güvenli ve hızlı depolama gerekir.
Tasarım / UML	PlantUML, Draw.io	Figma + Design System, Screenshot Flow (mobil)	Production'da tutarlı UI için tasarım sistemi gerekir. Sistemin görselleştirmesi için Screenshot Flow.

Prototip aşamasında, projenin hızlı şekilde doğrulanması ve kullanıcı geri bildirimlerinin kısa sürede toplanabilmesi amacıyla **Streamlit**, **FastAPI** ve **SQLite** gibi hafif, kurulumu kolay ve düşük maliyetli teknolojiler tercih edilmiştir. Bu araçlar, hızlı UI oluşturma, basit veri yönetimi ve hızlı API entegrasyonu gibi avantajlar sağlayarak fikrin kısa sürede çalışır bir demo hâline getirilmesini mümkün kılar.

Gerçek üretim ortamında, sistemin ölçeklenebilirliği, güvenliği ve sürekli kullanılabilirliği kritik hâle geldiği için **React/Flutter**, **PostgreSQL**, **Docker**, **Cloud Run/Kubernetes** ve kurumsal CI/CD araçları gibi daha güçlü teknolojilerin kullanılması gerekir. Bu araçlar, çoklu kullanıcı erişimini destekler, yüksek trafik altında stabil çalışır ve uzun vadeli bakım maliyetlerini azaltır. Kısacası, prototip teknolojileri hız ve esneklik amaçlarken; gerçek proje teknolojileri dayanıklılık, performans ve sürdürülebilirlik sunar.

Chatbot Destekli Akıllı Tarım Platformu – Sistem Akış Diyagramı



Önerilen akış diyagramı demo aşamasında ortaya koyacağımız platformun nasıl çalıştığını gösteren sistem mimarisini özetlemektedir. Süreç, kullanıcının Streamlit tabanlı arayüz üzerinden sisteme soru sorması veya haberlerle etkileşime geçmesiyle başlar. Kullanıcıdan alınan bu istek, frontend tarafından işlenerek FastAPI tabanlı backend katmanına HTTP isteği şeklinde iletilir.

Backend (FastAPI), uygulamanın “iş mantığı” ve “yönlendirme” katmanıdır. Kullanıcıdan gelen istek türüne göre iki farklı kaynakla iletişime geçer. Veri odaklı istekler için SQLite veritabanı sorgulanır ve burada haberler, bültenler veya bitki rehberi gibi sabit içerikler getirilir. Eğer kullanıcının isteği bir soru-cevap veya tarımsal açıklama gerektiriyorsa, backend bu soruyu Gemini API’sine ileterek yapay zekâ tabanlı bir yanıt üretir. Bu API, tarım ve bitki yetiştiriciliği gibi konularda doğal dil işleme desteği sunar ve kullanıcı sorusuna anlamlı bir çözücü yanıt döner.

Projenin bu aşamasında kullanılan Gemini doğrudan genel amaçlı bir yapay zekâ modeli olarak değil, tamamen projeye özgü komutlar ve uzmanlık alanı ile özelleştirilmiş bir akıllı asistan şeklinde yapılandırılmıştır. Bunun için projenin backend katmanı, kullanıcıdan gelen soruyu önce “system prompt” ve ek talimatlarla birleştirir. Bu şekilde Gemini, standart bir LLM gibi davranmaz ve proje için özel olarak eğitilmiş bir "tarım danışmanı" rolüne bürünür.

FastAPI, veritabanından aldığı içerikleri ve Gemini API’den gelen yapay zekâ yanıtlarını bir araya getirerek tek bir işlenmiş sonuç oluşturur. Bu sonuç daha sonra Streamlit arayüzüne gönderilir ve kullanıcıya sade, okunabilir bir şekilde gösterilir. Böylece kullanıcı; haberleri görüntüleyebilir, tarım bilgilerine erişebilir veya chatbot’a soru sorarak yanıt alabilir.

Yapay Zekâ Doğruluğunun Değerlendirilmesi - RAGAS Entegrasyonu

Platformda kullanılan Gemini tabanlı yapay zekâ asistanının verdiği yanıtların doğruluğunu, tutarlılığını ve bağlama uygunluğunu ölçmek amacıyla **RAGAS (Retrieval-Augmented Generation Assessment)** kütüphanesi sisteme entegre edilmiştir. RAGAS, özellikle RAG (bilgi getirme destekli üretim) mimarilerinde kalite kontrolü yapmak için kullanılan bir değerlendirme aracıdır.

Bu projede RAGAS, hem prototip aşamasında hem de gerçek üretim ortamında model kalitesini izlemek ve gerektiğinde iyileştirmek için kullanılmaktadır.

RAGAS’ın Sistemdeki İş Akışı

RAGAS entegrasyonu platformdaki soru–cevap akışını şu şekilde tamamlar:

1. Gemini tarafından üretilen RAG yanıtı

Kullanıcının sorusu, ilgili bağlam (haber, bitki rehberi vb.) ile birleştirilir ve Gemini modelinden bir yanıt oluşturulur.

2. Soru–Cevap–Bağlam üçlüsünün değerlendirme seti hâline getirilmesi

RAGAS, her değerlendirme için şu üç öğeye ihtiyaç duyar:

- **question** → Kullanıcı sorusu
- **answer** → Gemini’nin ürettiği yanıt

- **contexts** → Yanıtın üretildiği kaynak metinler

Bu üçlü bir değerlendirme tablosu hâline getirilir.

3. Gemini tabanlı değerlendirici (LLM backend)

RAGAS'ın çalışabilmesi için arka planda bir LLM değerlendirme motoruna ihtiyaç vardır. Bu projede Gemini için özel bir değerlendirme arayüzü kullanılır. Böylece RAGAS, kalite ölçümlerinde Gemini'ı hakem model olarak kullanır.

4. RAGAS metrikleriyle kalite ölçümü

Sistemde aşağıdaki temel metrikler kullanılır:

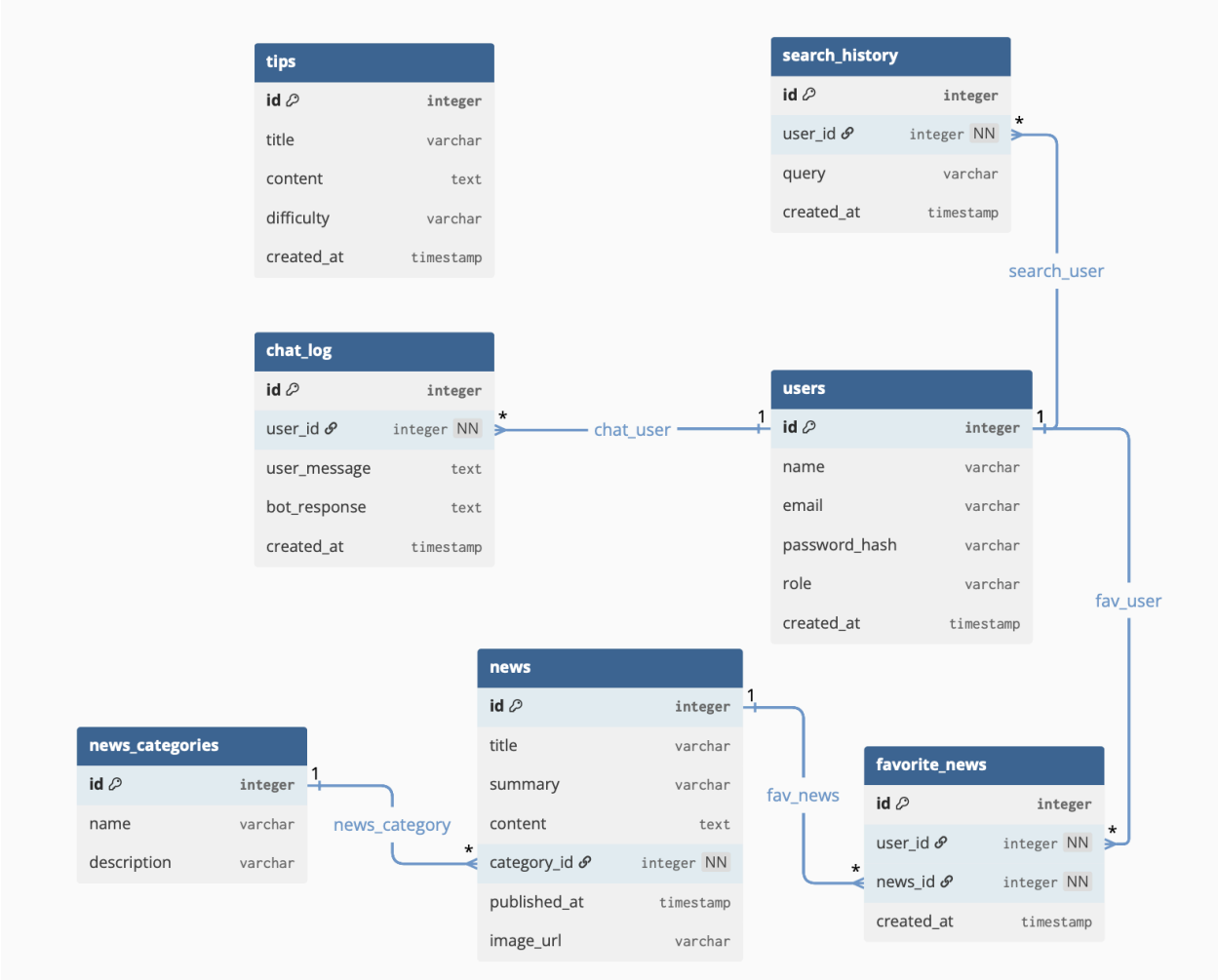
- **Faithfulness:** Cevabın bağlama sadakat düzeyi
- **Answer Relevancy:** Cevabın soruyla ilgisi
- **Context Relevancy:** Kullanılan bağlamın uygunluğu
- **Context Precision/Recall:** Modelin doğru bağlamı ne kadar isabetle seçtiği

Bu metrikler çalıştırıldığında, her soru–cevap işlemi için bir kalite puanı üretilir.

Proje Açısından Sağladığı Katkılar

- Yanıtların güvenilirliğini objektif olarak ölçer
- RAG mekanizmasının doğru bağlamı seçip seçmediğini denetler
- Chatbot yanıtlarının kalitesini izlemek için bir "değerlendirme katmanı" oluşturur
- Üretim ortamında model performansını düşüren değişikliklerin erken tespit edilmesini sağlar
- Sistemin gelecekte otomatik kalite izleme (LLM monitoring) altyapısına genişletilmesine imkân tanır

Veri Tabanı Diyagramı



Yukarıda **ERD (Entity Relationship Diagram)**, platformumuz için dbml diyagram dili kullanarak kurgulanmış gerçekçi ama demo seviyesinde bir veritabanı yapısını gösteriyor.

Diyagram şu bileşenlere dayanıyor:

- **Kullanıcılar** (users)
- **Chatbot sohbet kayıtları** (chat_log)
- **Arama geçmişi** (search_history)
- **Tarım haberleri** (news)
- **Haber kategorileri** (news_categories)
- **Favori haberler** (favorite_news)
- **Tarım ipuçları** (tips)

Yukarıda verilen birleşenler One-to-Many (1-N) olarak bir kaynağın birden fazla hedef kaydıyla ilişkili olabileceği şekilde bağlanmıştır. Bu yapıda kullanıcı, “merkez varlık” gibidir ve platformda yaptığı her eylem ayrı bir tabloya *çoklu kayıt* olarak yansır. Kullanıcı birçok chat kaydı, arama geçmişi kaydı ve haberle etkileşime girebilir.

Chatbot kayıtlarının da bu yapının bir parçası olması, hem kullanıcı davranışlarının analizini hem de chatbot yanıtlarının kalitesini artırmayı mümkün kılıyor. Böylece model, hem içerik yönetimi hem de sohbet tabanlı etkileşim için gerçekçi, hafif ve işlevsel bir altyapı sağlayabilecektir. Demo versiyonunda geçmiş chat diyalogu (chat_log) bulunmayacak, yani sorulan sorular ve gelen cevaplar veri tabanında tutulmayacak, fakat gerçek projede bu diyaloglar farklı sohbet sekmelerine ayrılabilir ve geçmişler veri tabanında tutulabilir.

İçerik tarafında, tarım haberleri news tablosunda tutulur ve her haber, kendi kategorisine (news_categories) bağlıdır. Kullanıcıya sunulan tarımsal ipuçları ise bağımsız bir tablo olan tips içerisinde yer alır. Dolayısıyla, planlanan veri tabanı yapısı kullanıcı davranışlarını ile içerik yönetimini basit ama gerçekçi bir modelle bir araya getirir ve platformun temel fonksiyonlarının verimli şekilde çalışmasını sağlar.

Zorunlu Veri Setlerinin Özellikleri

1. Tarım Haberleri (Güncel)

Kaynaklar:

- T.C. Tarım ve Orman Bakanlığı haber portalı
- FAO News
- EU Agriculture News

Olası kaynak bağlantıları:

- https://www.tarimorman.gov.tr/HaberArsivi
- https://www.ntv.com.tr/haberleri/tarim
- https://www.agriculturedive.com/

Format: JSON/HTML → Python ile scrape edilecek.

2. İklim Bültenleri / Meteorolojik Veriler

Kaynaklar:

- MGM (Meteoroloji Genel Müdürlüğü)
- Open-Meteo API (ücretsiz)

Kullanım:

- Günlük hava durumu tahmini
- Sıcaklık, yağış, nem göstergeleri

3. Bitki Yetiştirme Bilgi Bankası

Bu kısım **statik bir SQLite tablo** olarak eklenecek:

Kolonlar:

- Bitki adı
- Ekim/dikim dönemi
- Gübre önerisi
- Sulama ihtiyacı
- Zararlılar
- Organik çözüm önerileri

Kaynak olarak FAO, T.C. TARIM VE ORMAN BAKANLIĞI, tohum firmalarının katalogları.

Olası manuel giriş için kaynaklar: https://www.thespruce.com/plants-a-to-z-5116344, https://www.missouribotanicalgarden.org/plantfinder/plantfindersearch.aspx

4. Verimlilik İpuçları / Haftalık Tarım Tüyoları

Bu da statik bir bilgi tabanı olabilir (Excel → SQLite).

7. Proje İçin Varsa Gerekli Olan Veri Seti

Platform, önceden hazırlanmış bir veri setine doğrudan ihtiyaç duymamaktadır. Sistem; haber, iklim ve kullanıcı etkileşimleri gibi içerikleri dinamik olarak farklı kaynaklardan toplayan ve bunları veritabanında saklayarak işlemeye devam eden bir yapıda tasarlanmıştır. Bu nedenle proje kapsamında “veri seti belirleme” süreci; statik bir dosya seçmekten çok, platformun çalışması için gerekli olan veri türlerinin, veri kaynaklarının ve veri işleme yöntemlerinin tanımlanması şeklinde ele alınmıştır.

1. Tarım Haberleri Veri Seti (Dinamik Olarak Oluşturulan Veri)

Platformun temel bileşenlerinden biri güncel tarımsal haber akışının sunulmasıdır. Bu veriler önceden hazırlanmış bir veri seti olarak bulunmamakta; sistem çalıştıkça farklı resmi ve uluslararası kaynaklardan Python tabanlı web scraping yöntemleriyle toplanmaktadır.

Kullanılan Kaynaklar:

- T.C. Tarım ve Orman Bakanlığı Haber Portalı
- FAO (Food and Agriculture Organization) News
- EU Agriculture & Rural Development News
- Ulusal tarım haber portalları

Veri Türleri:

- Haber başlığı
- Haber özeti
- Haber içeriği
- Yayınlanma tarihi
- Haber kategorisi
- Kaynak adı ve URL

Toplanan veriler işlenerek veri tabanındaki news tablosuna kaydedilir. Bu yapı sayesinde platform her zaman güncel tarım gelişmelerini sunar ve bağımsız bir statik veri setine ihtiyaç duymaz.

2. Meteorolojik ve İklimsel Veri Setleri (Gerçek Zamanlı API Tabanlı Veri)

Platformun bir diğer önemli veri kaynağı meteorolojik bilgilerdir. Bu bilgiler bir veri seti halinde indirilmeyip, kullanıcının talebi doğrultusunda gerçek zamanlı API çağrılarıyla elde edilir.

Kullanılan Kaynaklar:

- Meteoroloji Genel Müdürlüğü (MGM)
- Open-Meteo API (ücretsiz)

Toplanan Veri Türleri:

- Sıcaklık
- Yağış ihtimali ve miktarı
- Rüzgâr hızı ve yönü
- Nem oranı
- Günlük ve haftalık hava tahmini

Bu veriler herhangi bir statik veri seti olarak saklanmak yerine, platform tarafından anlık olarak işlenir ve kullanıcıya sunulur. Gerektiğinde kısa süreli cache mekanizmasıyla performans artırılabilir.

3. Bitki Yetiştirme Bilgi Bankası (Statik Veri Seti)

Sistemin bilgi tabanı içerisinde yer alacak "Bitki Yetiştirme Rehberi" verisi dış kaynaklardan doğrudan indirilebilecek bir dataset değildir. Bu nedenle proje kapsamında tarım kaynakları taranarak rehber bilgiler manuel ya da yarı otomatik biçimde derlenmiş ve SQLite içerisine aktarılacak şekilde hazırlanmıştır. Statik veri setleri, sunucu çalışmaya başlamadan önce doldurulur ve genellikle değiştirilmez.

Kaynaklar:

- FAO bitki yetiştirme dökümanları

- T.C. Tarım ve Orman Bakanlığı teknik dökümanları
- Tohum firmalarının ürün katalogları
- Uluslararası açık bitki veri tabanları

Veri Alanları:

- Bitki adı
- Ekim/dikim dönemi
- Sulama gereksinimi
- Gübreleme önerileri
- Yaygın hastalık ve zararlılar
- Organik/alternatif çözüm önerileri

4. Verimlilik İpuçları ve Haftalık Tarım Tüyoları (Statik Veri Seti)

Platformda kullanıcılara sunulacak tarımsal ipuçları ve haftalık öneriler, proje ekibi tarafından hazırlanmış olup başlangıçta Excel veya CSV gibi bir formatta oluşturulmuş, ardından SQLite veri tabanına aktarılmıştır. Bu veri seti statiktir ancak proje genişledikçe güncellenebilir.

Veri Türleri:

- İpucu başlığı
- Açıklama metni
- Hedef bitki türü (varsa)
- Haftalık öneri etiketi

5. Kullanıcı Etkileşim Kayıtları (Sistem Çalıştıkça Oluşan Dinamik Veri Setleri)

Sistem kullanıcılarla etkileşime girdikçe aşağıdaki veri setleri otomatik olarak dolar:

- Chatbot Sohbet Kayıtları
- Arama Geçmişi
- Favori haberler
- Kullanıcı bilgileri

Bu veriler başlangıçta mevcut değildir; platform kullanılmaya başlandıkça chat_log, search_history ve favorite_news tablolarında dinamik olarak üretilir.

Demo versiyonunda sohbet kayıtlarının saklanması zorunlu değildir; ancak üretim ortamında kullanıcı özel sohbet geçmişi tutulacaktır.

Özet

Bu proje için önceden hazırlanmış bir veri seti bulunması zorunlu değildir. Platform, ihtiyaç duyduğu verileri:

- Web scraping (haberler)
- Gerçek zamanlı API çağrıları (meteoroloji)
- Manuel/yarı otomatik bilgi tabanı oluşturma (bitki rehberi, ipuçları)
- Kullanıcı etkileşimleri (chat/arama geçmişi)

yöntemleriyle dinamik olarak üretmektedir. Bu yaklaşım, sistemin her zaman güncel kalmasını, esnek şekilde genişletilebilmesini sağlamaktadır.

Sorumluluk Dağılımı

Alper Erdoğan - Backend & Veri Sorumlusu

Sunucu, veritabanı ve veri toplama işleri.

Görevleri:

- FastAPI backend'i kurmak ve geliştirmek
- Veritabanı (SQLite, PostgreSQL) tasarımı ve tabloların oluşturulması
- Haber scraping kodlarını yazmak (Python + BeautifulSoup)
- İklim API entegrasyonu yapmak
- Backend'in Gemini API'ye istek atmasını sağlamak
- RAGAS değerlendirme yapısının backend tarafını kurmak
- Docker ve temel deployment hazırlıkları

Enes Ağaoğlu - Frontend & Chatbot Arayüzü Sorumlusu

Kullanıcı arayüzü, chatbot akışı ve AI entegrasyonu.

Görevleri:

- Streamlit arayüzünü geliştirmek
- Chatbot ekranını hazırlamak
- Haber, bitki rehberi, ipuçları gibi sayfaların UI tasarımı
- Backend API'lerine bağlanmak
- Gemini API'den gelen yanıtları kullanıcıya düzgün şekilde göstermek
- RAGAS sonuçlarını (gerekirse) arayüzde görüntülemek