🎬 1. KULLANIM DURUM DİYAGRAMI (Use Case Diagram)

🎯 Amaç:

Sistemin kullanıcıyla etkileşim halinde olduğu senaryoları ve bu senaryoların birbirleriyle ilişkisini modellemek.

👤 Aktör(ler):

- **Kullanıcı (User):** Sisteme doğal dil sorgusu giren, sorgu sonuçlarını inceleyen, veri raporlayan kişi.
- **Sistem (Flask + OpenAI + Veritabanı):** Arka planda işlemleri yöneten otomasyon sistemi.

📌 Kullanım Durumları:

Use Case ID	Adı	Açıklama
UC01	Doğal Dil Sorgusu Gir	Kullanıcı, metin kutusuna sorgu cümlesi yazar
UC02	SQL Sorgusu Oluştur	Sistem, doğal dili OpenAI API aracılığıyla SQL'e dönüştürür
UC03	SQL Sorgusunu Çalıştır	Oluşan SQL komutu veritabanında çalıştırılır
UC04	Sonucu Görüntüle	Sorgu sonucu kullanıcıya tablo formatında sunulur
UC05	SQL Kodu Önizle	Kullanıcıya üretilen SQL komutu gösterilir
UC06	Hataları Gör	Sistem, hata varsa anlamlı uyarılar gösterir
UC07	Veriyi Dışa Aktar	Sonuçlar CSV/TXT formatında dışa aktarılabilir
UC08	Mobil Arayüz Kullan	Mobil uyumlu sistem, farklı cihazlardan kullanılabilir

İlişkiler:

- UC01 → UC02 → UC03 → UC04 + UC05
- UC03 → UC06 (opsiyonel)
- UC04 → UC07
- UC01 → UC08 (platform bağımsız erişim)

---

2. SINIF (CLASS) DİYAGRAMI

Amaç:

Sistemin yazılımsal bileşenleri arasındaki ilişkileri, verileri ve metodları nesne yönelimli olarak göstermek.

Temel Sınıflar ve Açıklamaları:

AppController

Ana Flask controller sınıfıdır. Rotaları yönetir.

- **Nitelikler:**
  - `app: Flask`
  - `DATABASE_PATH: str`
- **Metotlar:**
  - `home()`
  - `generate_sql_query()`
  - `response()`

QueryProcessor

Kullanıcının doğal dil sorgusunu işler.

- **Nitelikler:**
  - `client: OpenAI`
- **Metotlar:**
  - `generate_sql(query_text: str) → str`
  - `extract_sql(response_text: str) → str`

DatabaseHandler

Veritabanıyla etkileşimi yönetir (SQLite).

- **Nitelikler:**
  - `db_path: str`
- **Metotlar:**
  - `get_tables() → list`
  - `get_columns() → dict`
  - `execute_query(sql: str) → tuple`

SQLResultRenderer

SQL çıktısını biçimlendirir ve tabloya dönüştürür.

- **Metotlar:**
  - `render_table(results, headers) → str`

İlişkiler:

- `AppController` → kullanır → `QueryProcessor`
- `AppController` → kullanır → `DatabaseHandler`
- `AppController` → kullanır → `SQLResultRenderer`

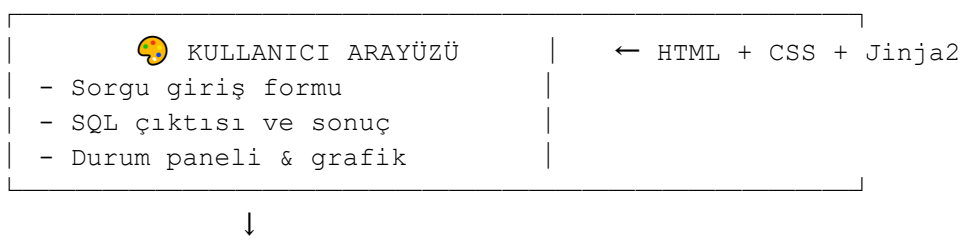
---

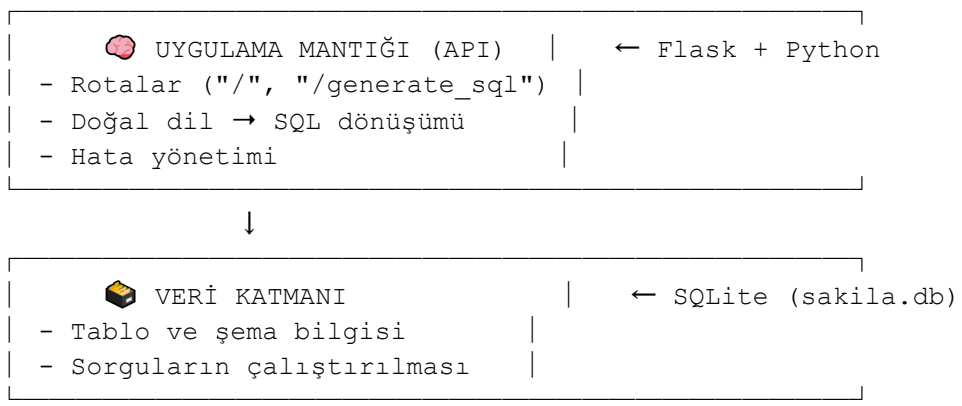
3. UYGULAMA MİMARİSİ

Amaç:

Sistemin bileşenlerini katmanlı olarak ayırarak, yapının nasıl çalıştığını göstermek.

3 Katmanlı Mimari:





🔌 DIŞ SERVİS ENTEGRASYONLARI

Servis	Amaç
OpenAI API	GPT-3.5 Turbo modeliyle doğal dil → SQL dönüşümü
Tabulate	Sorgu sonuçlarını okunabilir tabloya dönüştürür
Flask	Sunucu tarafı yönlendirme ve web servisi

🔍 GELECEKTEKİ GENİŞLETME NOKTALARI

- PostgreSQL / MongoDB gibi alternatif veritabanı katmanı
- Mobil uygulama arayüz katmanı (PWA veya Flutter)
- Grafik motoru (Chart.js, D3.js) için görsel analiz katmanı
- Kimlik doğrulama (login/register) modülü