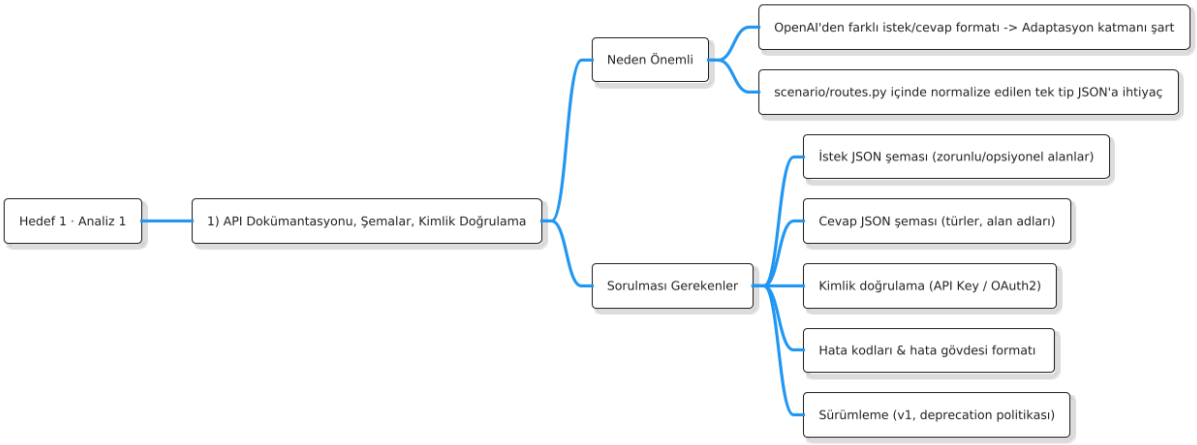



Hedef.1 – Analiz.1 – Soru 1

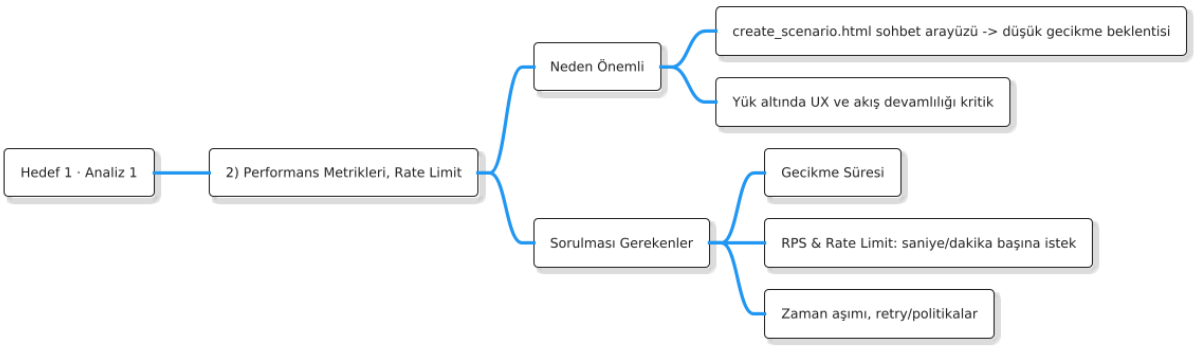
***API Dokümantasyonu, Veri Şemaları ve Kimlik Doğrulama Yöntemi Nelerdir?***

Bu soruyu sormamın sebebi, görev tanımında yerel AI servisinin OpenAI'den farklı bir istek/yanıt formatı kullandığının özellikle belirtilmiş olması. Geçişi sağlıklı şekilde yapabilmem için önce bu yeni servisin kullandığı yapıyı net olarak anlamam gerekiyor. Çünkü scenario/routes.py içinde kuracağımız adaptasyon katmanı, tamamen bu yapıya göre şekillenecek.

Bu noktada netleştirmem gerekenler:

- İstek (request) formatı ve cevap (response) JSON şeması tam olarak nasıl?
- Kimlik doğrulama mekanizması nedir?
  (Basit bir API key mi gerektiriyor, yoksa OAuth 2. gibi daha kapsamlı bir akış mı?)
- Servis hangi hata kodlarını döndürüyor ve bu hataların cevap formatı nasıl görünüyor?

Bu bilgileri netleştirirsem, adaptasyon katmanının hem doğru veri dönüşümünü yapabilmesini hem de sorunsuz hata yönetimini yapmasını sağlayabilirim. Ayrıca config.py içerisine eklenecek ayarların ne olması gerektiğini de doğrudan belirlenecek.



Hedef.1 – Analiz.1 – Soru 2

***Servisinizin Performans Metrikleri (Rate Limit, Gecikme) Nedir?***

Bu soruyu yöneltme nedenim, sistemin create_scenario.html üzerinden çalışan bir sohbet arayüzüne dayanması. Kullanıcı burada anlık yanıt bekliyor, dolayısıyla geçiş sırasında en kritik risklerden biri gecikme (latency) ve istek sınırları (rate limit) olabilir. Eğer yeni model OpenAI'ye kıyasla daha yavaş cevap veriyorsa ya da belirli bir istek sınırına sahipse, bu doğrudan kullanıcı deneyimini ve sürecin akıcılığını etkiler.

Bu yüzden şu bilgileri netleştirmek istiyorum:

Ortalama ve gecikme süreleri nedir?

Platformun saniye/dakika başına izin verdiği istek sayısı ne kadar?

"Servisinizin Performans Metrikleri (Rate Limit, Gecikme) ve SLA Bilgisi Nedir?"

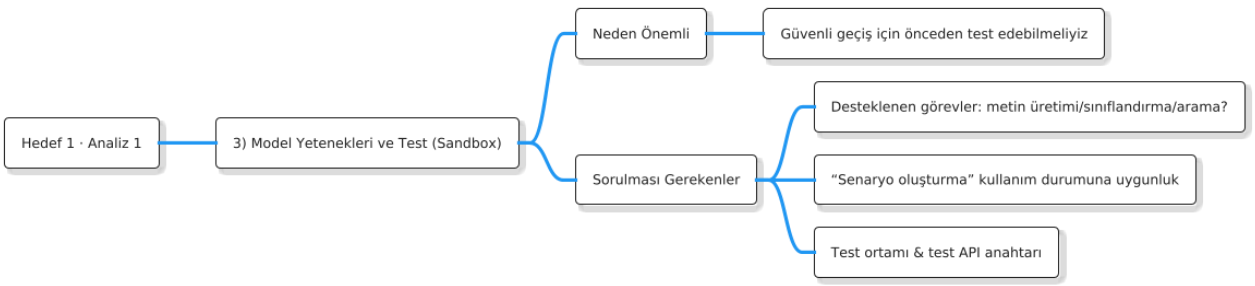
Bu soruyu yöneltme nedenim, sistemin create_scenario.html üzerinden çalışan bir sohbet arayüzüne dayanması. Kullanıcı burada anlık yanıt bekliyor, dolayısıyla geçiş sırasında en kritik risklerden biri gecikme (latency) ve istek sınırları (rate limit) olabilir. Eğer yeni model OpenAI'ye kıyasla daha yavaş cevap veriyorsa ya da belirli bir istek sınırına sahipse, bu doğrudan kullanıcı deneyimini ve sürecin akıcılığını etkiler.

Bu yüzden şu bilgileri netleştirmek istiyorum:

Ortalama ve gecikme süreleri nedir?

Platformun saniye/dakika başına izin verdiği istek sayısı ne kadar?

(Limit aşıldığında hangi hata dönüyor? Örn: HTTP 429)




Hedef.1 – Analiz.1 – Soru 3

***Modelin Generative Yetenekleri ve Test Ortamı Var mı?***

Bu soruyu sormanın nedeni, mevcut sistemde OpenAI'nin etkinlik senaryosu üretme görevinde kullanılıyor olması. Yani ihtiyacımız sadece metin üretimi değil; aynı zamanda bağlamı anlayıp anlamlı, tutarlı ve kategorize edilebilir bir senaryo çıktısı üretme kapasitesi.

Bu nedenle şunları öğrenmek istiyorum:

Model hangi kullanım türlerini destekliyor? (Metin üretimi, sınıflandırma, semantik arama vb.)

Bizim "senaryo oluşturma" kullanım durumumuza uygunluğu test edilmiş mi?

Geçiş sürecinde deneme yapabilmem için bir test ortamı (sandbox) veya test API anahtarı sağlanabilir mi?

Cevap kalitesini korumak için temperature gibi kontrol parametreleri destekleniyor mu?

Bu bilgiler, yerel modelin senaryo kalitesinin mevcut OpenAI akışına yakın veya daha iyi olup olamayacağını önceden değerlendirmeme yardımcı olacak.

---

## Hedef 1 – Plan 2

---

Projeye ai_service.py adında yeni bir dosya ekleyeceğiz. Bu dosya, scenario/routes.py ile yeni Yerel AI API'si arasında bir ara katman olacak.

- scenario/routes.py artık direkt API ile konuşmayacak.
- ai_service.py:
  - Prompt'u alacak
  - Yerel AI API'sine uygun hale getirecek
  - API'ye isteği gönderecek
  - Gelen cevabı eski sistemin beklediği standart JSON formatına dönüştürüp geri verecek

Böylelikle

- Kodun geri kalanında hiçbir yer değişmeyecek
- AI motoru değiştiğinde tek değiştireceğimiz dosya ai_service.py olacak.

Bu değişikliği yaptığımızda üç dosya etkilenir. İlk olarak, ***scenario/routes.py*** içindeki mevcut OpenAI ile ilgili kodları kaldırıyoruz. Artık bu dosya doğrudan API ile konuşmayacak; bunun yerine sadece ai_service.py dosyasını çağıracak. Yani tüm AI işlemleri ai_service.py üzerinden yapılacak.

İkinci olarak, ***config.py*** dosyasına yeni Yerel AI servisinin ***URL*** ve ***API_KEY*** bilgilerini ekleyeceğiz. ai_service.py bu bilgilere buradan erişerek API çağrılarını yapacak.

Son olarak, yeni bir dosya olan ***ai_service.py*** oluşturuyoruz. Tüm istek gönderme ve cevap dönüştürme işlemleri burada gerçekleşecek. Böylece ileride AI motoru tekrar değişirse, yalnızca ai_service.py dosyasını güncellememiz yeterli olacak; uygulamanın diğer bölümleri etkilenmeyecek.

---

## Hedef 1 – Risk 3

Bu geçişte en büyük teknik risk cevap kalitesidir. Yani sistemin ürettiği cevap kullanıcının isterlerine uygun olmalı.

Çünkü sistemin en önemli özelliği, kullanıcının yazdığı metinden etkinlik senaryosu üretmek. Yani bu platformu sıradan sitelerden ayıran şey tam budur.

Kullanıcı, sohbet ekranına hayalini yazıyor ve karşılığında anlamlı ve güzel bir senaryo görmek istiyor. Eğer yeni Yerel AI motoru bu senaryoyu kaliteli üretemezse, kullanıcı sonuçtan memnun kalmaz ve "Teklif Al" akışına hiç geçmez. Bu da sistemin diğer süreçlerinin başlamadan durması demektir.

Diğer riskler (örneğin hız veya maliyet) bir şekilde çözülebilir:

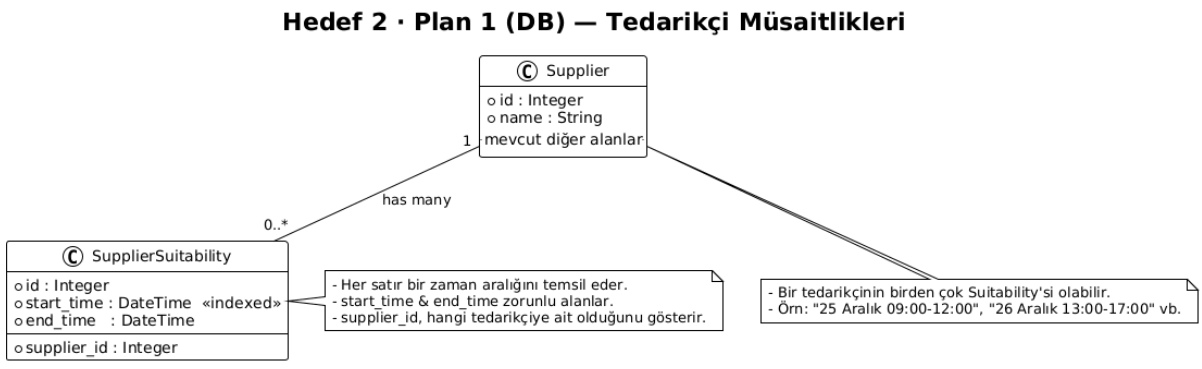
Hız yavaşsa, optimize edilebilir.

Maliyet yüksekse, ölçeklendirme veya model ayarı yapılabilir.

Ama kötü senaryo üretimi çözülemezse bu durum doğrudan kullanıcı kaybına yol açar. Kalite düşerse, kullanıcı sistemi kullanmayı bırakır. Hız veya maliyet sorunları ise sonradan iyileştirilebilir.

---

## Hedef 2 – Veritabanı 1



Tedarikçilerin müsaitlik zaman aralıklarını tutmak için veritabanına SupplierSuitability tablosu ekleyebiliriz. Bu tablo, her tedarikçi için “başlangıç zamanı” ve “bitiş zamanı” bilgisini barındırabilir. Mevcut Supplier tablosu ile bire-çok (one-to-many) ilişki kuracağız; yani bir tedarikçinin birden fazla müsaitlik kaydı olabilir. Uygulamada amaç basit: bir tedarikçinin hangi gün/saat aralıklarında uygun olduğunu sorgulayabilmek.

Bu yapıyla birlikte uygulama tarafında:

- Bir tedarikçinin tüm müsaitliklerini “tedarikçi → müsaitlik listesi” şeklinde basitçe çekebiliriz.
- Belirli bir tarih/saat aralığında hangi tedarikçilerin uygun olduğunu filtrelemek kolaylaşır.
- İleride ihtiyaç olursa “çakışan aralık” kontrolü gibi iş kurallarını servis katmanında ekleyebiliriz.

---

## Hedef 2 – Workflow 2

- Yeni eklenen bu özellik, ön filtreleme adımı gibi davranır. Yeni akışta bir talebe sadece uygun olan kişiler listelenir teklif gönderme rotasını ve booking_detail .html sayfasını değişmesine sebep olur.
- Yönetici, müsait olmayan tedarikçilere boş yere talep göndererek zaman kaybetmez.
- Sadece "gerçekten" müsait olan tedarikçilerden teklif alacağı için , gelen teklifleri onaylama süreci çok hızlanır. Daha az "red" cevabı ile uğraşır.
- Tedarikçiler, zaten dolu oldukları günler için sürekli "reddetmek" zorunda kalacakları teklif talepleri almazlar. Sistem üzerine binen yük azalmış olur.
- Sadece gerçekten yapabilecekleri işler için teklif verirler, bu da işe dönüşme oranlarını artırır.
- Yöneticinin müsait tedarikçileri bulması ve fiyatları toplaması hızlandığı için, müşterinin "Teklif Al"a basması ile "Teklif Gönderildi" e-postasını alması arasındaki toplam süre **önemli ölçüde kısalır**.

---

## Hedef 2 – Teknoloji Önerisi 3

Bu proje için Flatpickr + Custom yaklaşımını tercih ederim. Çünkü sistem burada ağır bir tam takvim yönetim sistemi değil, tedarikçinin müsaitlik aralıklarını kolay girip düzenleyebileceği hızlı ve sade bir arayüz istiyor. Bundle size'ı 260KB üzerinde olan FullCalendar (core + gerekli pluginler), basit bir müsaitlik işaretleme işlemi için fazlasıyla ağır. FullCalendar çok büyük, yüklenmesi zaman alıyor ve öğrenmesi daha zor. Buna karşılık Flatpickr hem daha hafif, hem de kullanması çok daha kolay. Bağımlılıkları az olduğu için projeyi yükünü arttırmıyor ve mobilde de daha doğal bir kullanım sunuyor. Ayrıca kendi kurallarımıza göre özelleştirmek istediğimizde bizi kısıtlamıyor. Kısacası, burada hız, sadelik ve esneklik önemli olduğu için Flatpickr + Custom benim için daha uygun bir seçim olur.