Başvurunuz için teşekkür ederiz. Aşağıda tarafınıza değerlendirme için teknik task'ı iletiyoruz. Önemli olan ne kadarını yaptığınız değil, süreci nasıl ele aldığınız ve yaklaşımınızın doğruluğudur.

Başarılar dileriz. 🙌

📌 BACKEND DEVELOPER TASK

Görev: Çift Aşamalı Onaylı Rezervasyon Sistemi Geliştirme

Bu görev, gerçek bir senaryoya dayanan bir rezervasyon sürecinin backend tarafını tasarlamanı ve geliştirmeni bekler. Amaç; mimari tasarım becerisi, veri tutarlılığı, işlem yönetimi, eşzamanlılık (concurrency) ve background job kullanımına hâkimiyeti ölçmektir.

Sistem Özeti

Bir etkinlik platformu düşünün. Kullanıcılar etkinliklere yer ayırtabiliyor ancak rezervasyon süreci **iki aşamalı** çalışıyor:

1. **Hold (Geçici Ayırma – 5 dakika)**
2. **Confirm (Onaylama)**

Hold süresi dolarsa rezervasyon otomatik olarak düşmeli ve kapasite geri alınmalı.

İstenilen Özellikler

1) Etkinlik Yönetimi

Admin şu işlemleri yapabilmelidir:

- Etkinlik oluşturma
- Alanlar:
- title
- capacity
- startDate
- endDate
- isActive (true/false)

2) Rezervasyon Süreci

Aşama 1 – HOLD

- Kullanıcı rezervasyon talebi gönderir.
- Kapasite uygunsa:
- Kullanıcıya **5 dakikalık bir hold** oluşturulur.
- Hold kaydı veritabanında "HOLD" state olarak saklanır.
- 5 dakikalık süre boyunca kapasite geçici olarak düşer.
- Aynı etkinliğe birden fazla kullanıcı aynı anda hold atabilir → veri tutarlılığı sağlanmalı.

Aşama 2 – CONFIRM

- Kullanıcı 5 dakika içinde rezervasyonu onaylarsa:
- HOLD → CONFIRMED olarak güncellenir.
- Kapasite kalıcı olarak düşer.

Hold Süresi Dolarsa

- 5 dakika sonunda hold kaydı otomatik silinmeli (background job).
- Kapasite geri iade edilmeli.

3) Arka Plan İşlemleri (Background Job)

Süresi dolmuş HOLD kayıtlarını otomatik kaldıran bir mekanizma eklenmelidir.
Aşağıdaki yöntemlerden biri kullanılabilir:

- Cron job
- Background worker
- Queue (bonus)
- DB trigger (opsiyonel)

Amaç:

- Hold süresi dolan rezervasyonları silmek
- Kapasiteyi geri arttırmak

4) Listeleme

Etkinlik detay endpoint'inde:

- Kalan kapasite
- Kaç HOLD olduğu
- Kaç CONFIRMED olduğu

doğru şekilde gösterilmelidir.

Teknik Gereklilikler

- JWT veya benzeri bir auth uygulanmalı.
- Tüm kritik işlemlerde **transaction** kullanılmalı.
- Eşzamanlı gelen rezervasyon taleplerinde veri tutarlılığı korunmalı.
- Katmanlı/temiz bir mimari tercih edilmeli.

Değerlendirme Kriterleri

- Mimari tasarımın kalitesi
- Kodun okunabilirliği
- Transaction ve concurrency yönetimi
- HOLD → CONFIRM sürecini doğru uygulama
- Background job tasarımı
- README açıklığının yeterliliği
- Test endpointleri ve Postman collection

Teslimat:

- GitHub repo linki
- Postman Collection
- README dosyası