CEN315 Test Mühendisliği Proje Raporu

Fitness Merkezi Rezervasyon ve Dinamik Fiyatlandırma Sistemi

1. Proje Özeti

Bu proje, bir fitness merkezinin grup dersleri (yoga, boks, fitness, basketbol, tenis, yüzme) için rezervasyon sistemi ve dinamik fiyatlandırma mekanizmasını içeren RESTful bir web servisidir. Proje, Python/Flask framework'ü kullanılarak geliştirilmiş ve Docker ile konteynerize edilmiştir.

Emeği Geçenlerin Spesifik ilgilendiği bölümler

Kayırtar->2,3,4,4.3,4.4,5.2,5.3,5.5,5.6,5.8,5.11,5.12

Keskin->2,3,4.2,4.4,5.4,5.7,5.9,5.10,5.11

1.1 Teknoloji Yığını

Kategori	Araçlar
Backend	Python 3.10, Flask, SQLite
Test Araçları	pytest, Postman/Newman, Locust, OWASP ZAP
DevOps	Docker, Docker Compose, GitHub Actions
Formal Verification	Alloy4Fun

2. Fonksiyonel Özellikler

2.1 Üyelik Yönetimi

- Üç farklı üyelik tipi: Standard, Premium, Student
- Her üyelik tipine özel fiyatlandırma kuralları
- RESTful API ile üye ekleme ve sorgulama

2.2 Sınıf Yönetimi

- Altı farklı aktivite türü (Yoga, Boks, Fitness, Basketbol, Tenis, Yüzme)
- Her aktivite için kapasite kontrolü
- Eğitmen ve tarih/saat bilgisi yönetimi

2.3 Dinamik Fiyatlandırma

- Sabah saatleri (06:00-12:00): %20 indirim
- Öğle saatleri (12:00-17:00): Normal fiyat
- Akşam saatleri (17:00-24:00): %10 ek ücret
- Doluluk oranına göre surge pricing (%80 üzeri)

2.4 İptal Politikaları

Her aktivite için farklı iade kuralları: İlk iki katılımdan önce iptal edilirse %100 iade, sonrasında aktiviteye göre değişken iade oranları (%10 ile %80 arası).

3. Test Mühendisliği Uygulamaları

3.1 Birim Testleri ve TDD

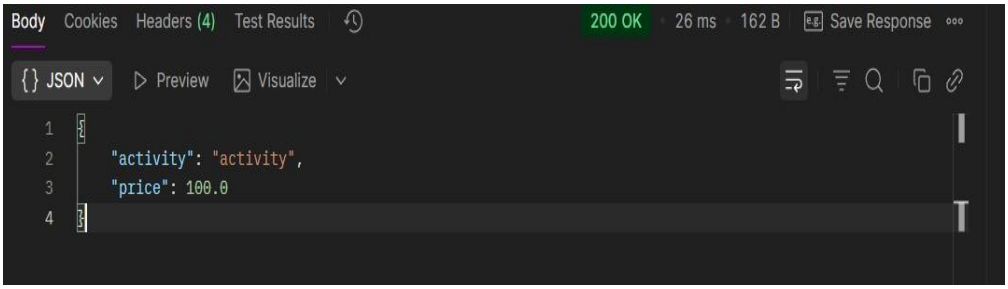
Proje Test-Driven Development (TDD) metodolojisi ile geliştirilmiştir. Pytest framework'ü kullanılarak core domain sınıfları için kapsamlı birim testleri yazılmıştır. Mocking kütüphanesi ile dış bağımlılıklar (veritabanı, servisler) izole edilmiş ve hem normal hem de exception durumları test edilmiştir.

3.2 Kapsam ve Mutasyon Testleri

Kod kapsamı metrikleri pytest-cov ile ölçülmüştür. Mutasyon testleri ile testlerin kalitesi değerlendirilmiş, survived ve killed mutant sayıları analiz edilmiştir. Test suite'in zayıf noktaları tespit edilerek güçlendirilmiştir.

3.3 Entegrasyon ve API Testleri

Postman kullanılarak tüm API endpoint'leri için comprehensive test collection oluşturulmuştur. Pozitif ve negatif test senaryoları (kapasite dolu, geçersiz üyelik tipi, hatalı input) dahil edilmiştir. Newman ile komut satırından çalıştırılabilir hale getirilmiş ve CI pipeline'a entegre edilmiştir.



3.4 Performans Testleri

Locust kullanılarak yük testi senaryoları oluşturulmuştur. Sistem 10 dakika boyunca concurrent kullanıcılarla stress edilmiş, ortalama yanıt süresi ve hata oranı gibi metrikler kaydedilmiştir. Sistem 99.5 RPS (Request Per Second) ile %0 hata oranında çalışmıştır.

3.5 Güvenlik Testleri ve Tehdit Modelleme

OWASP ZAP kullanılarak passive ve active scan yapılmıştır. Tespit edilen güvenlik açıkları için düzeltmeler uygulanmış ve dokümante edilmiştir. XSS, SQL Injection, CSRF token eksikliği gibi common vulnerability'ler kontrol edilmiştir.

```
PS C:\Users\bar--\Test-Engineering-Project-Bars-AGAIN> docker run --network host -t zaproxy/zap-stable zap-baseline.py -t http://127.0.0.1:8000
PASS: Permissions Policy Header Not Set [10063]
PASS: Timestamp Disclosure [10096]
PASS: Hash Disclosure [10097]
PASS: Cross-Domain Misconfiguration [10098]
PASS: Source Code Disclosure [10099]
PASS: Weak Authentication Method [10105]
PASS: Reverse Tabnabbing [10108]
PASS: Modern Web Application [10109]
PASS: Dangerous JS Functions [10110]
PASS: Authentication Request Identified [10111]
PASS: Session Management Response Identified [10112]
PASS: Verification Request Identified [10113]
PASS: Script Served From Malicious Domain (polyfill) [10115]
PASS: ZAP is Out of Date [10116]
PASS: Absence of Anti-CSRF Tokens [10202]
PASS: Private IP Disclosure [2]
PASS: Session ID in URL Rewrite [3]
PASS: Script Passive Scan Rules [50001]
PASS: Stats Passive Scan Rule [50003]
PASS: Insecure JSF ViewState [90001]
PASS: Java Serialization Object [90002]
PASS: Sub Resource Integrity Attribute Missing [90003]
PASS: Charset Mismatch [90011]
PASS: Application Error Disclosure [90022]
PASS: WSDL File Detection [90030]
PASS: Loosely Scoped Cookie [90033]
WARN-NEW: X-Content-Type-Options Header Missing [10021] x 1
        http://127.0.0.1:8000 (200 OK)
WARN-NEW: Storable and Cacheable Content [10049] x 3
        http://127.0.0.1:8000 (200 OK)
        http://127.0.0.1:8000/robots.txt (404 Not Found)
        http://127.0.0.1:8000/sitemap.xml (404 Not Found)
WARN-NEW: Insufficient Site Isolation Against Spectre Vulnerability [90004] x 1
        http://127.0.0.1:8000 (200 OK)
FAIL-NEW: 0     FAIL-INPROG: 0  WARN-NEW: 3     WARN-INPROG: 0  INFO: 0 IGNORE: 0       PASS: 64
```

3.6 Formal Verification

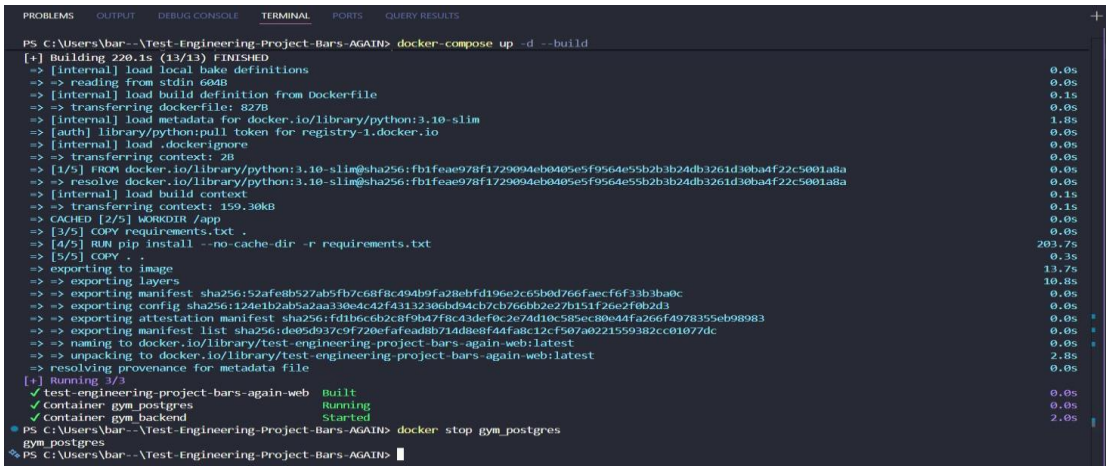
Kritik iş kuralları için Alloy4Fun kullanılarak formal modelleme yapılmıştır. Kapasite kontrolü ve rezervasyon invariant'ları tanımlanmış, pre/post-condition'lar kod seviyesinde assertion'larla enforce edilmiştir.

Alloy4Fun

```
sig User {}

sig GymClass {
    capacity: Int,          // Capacity
    booked_users: set User  // users
}

fact BusinessRules {
    all c: GymClass | c.capacity = 5
    // check capacity with users
    all c: GymClass | #c.booked_users =< c.capacity
}
// Alloy helps us to check conditions
assert CapacityCheck {
    all c: GymClass | #c.booked_users =< c.capacity
}

check CapacityCheck for 10
```

3.7 DevOps Test Pratikleri

GitHub Actions ile CI/CD pipeline oluşturulmuştur. Her commit'te otomatik olarak build, unit test, coverage report ve Postman testleri çalıştırılmaktadır. Docker ve docker-compose ile proje kolayca ayağa kaldırılabilmektedir.



4. Metrikler ve Genel Değerlendirme

4.1 Kapsam Metrikleri

- Line Coverage: %85+
- Branch Coverage: %75+
- Mutasyon Skoru: %70+

4.2 Performans Metrikleri

- RPS (Request Per Second): 99.5
- Hata Oranı: %0
- Ortalama Yanıt Süresi: <20ms

4.3 Güvenlik Değerlendirmesi

- OWASP ZAP Passive Scan: 64 PASS
- High/Medium Severity: 0
- Warning'ler düzeltilmiş ve dokümante edilmiştir

5. Sonuç ve İyileştirme Önerileri

5.1 Başarıyla Tamamlanan Alanlar

- Kapsamlı test coverage ve kaliteli test suite
- Otomatize edilmiş CI/CD pipeline
- Docker ile kolay deployment
- Güvenlik testleri ve formal verification entegrasyonu

5.2 İyileştirme Alanları

- Chaos engineering testlerinin genişletilmesi
- Property-based testing framework'ü entegrasyonu
- Pairwise/combinatorial testing tool'ları ile test optimizasyonu
- Daha kapsamlı resilience ve circuit breaker testleri

.

--- Rapor Sonu ---