YAZILIM MİMARİLERİ VE TASARIM

DOÇ. DR. HAKAN GÜNDÜZ

DİJİTAL RESTORAN REZERVASYON VE MENÜ SİSTEMİ: YAZILIM MİMARİSİ VE TASARIM

230229034 Betül Eren
230229038 Irmak Yılmaz
230229059 Duygu Ersoy
230229074 Zehra Özdemir
220229053 Sena Nur Dülger

Dijital Restoran Rezervasyon ve Menü Sistemi

İçin

Yazılım Gereksinimleri Analizi

Versiyon 1.0

230229034 Betül Eren
230229038 Irmak Yılmaz
230229059 Duygu Ersoy
230229074 Zehra Özdemir
220229053 Sena Nur Dülger
tarafından hazırlanmıştır.

Kocaeli Üniversitesi
Yazılım Mühendisliği

08 / 05 / 2025

İçindekiler

İçindekiler
Revizyon Geçmişi

1. Giriş

1.1 Ürün Misyon Beyanı
1.2 Kapsam
1.3 Tanımlamalar ve Kısaltmalar
1.4 Kaynaklar
1.5 Dokümana Genel Bakış

2. Genel Tanım

2.1 Ürün Perspektifi
2.2 Sistem İşlevleri
2.3 Kullanıcı Özellikleri
2.4 Kısıtlar

3. Özel Gereksinimler

3.1 Dış Arabirimi Gereksinimleri
3.2 Fonksiyonel Gereksinimleri
3.3 İşlevsel Olmayan Gereksinimler
3.4 Ters Gereksinimleri
3.5 Tasarım Kısıtlamaları
3.6 Mantıksal Veritabanı Gereksinimleri
3.7 Diğer Gereksinimler

4. Arayüz Gereksinimleri

4.1 Kullanıcı Paneli
4.2 Yönetici Paneli
4.3 Donanım Arayüzleri
4.4 Yazılım Arayüzleri
4.5 İletişim Arayüzleri

5. Ek Gereksinimler

5.1 Akıllı Asistan
5.2 Veri Modeli ve Veritabanı Yapısı
5.3 Riskler
5.4 Geliştirme Takvimi

Revizyon Geçmişi

İsim	Tarih	Değişim Sebebi	Versiyon
Birleştirme ve Tamamlama	08/05/2025	Kişiler arasında paylaştırılan bölümler birleştirildi ve eksik kalan maddeler tamamlandı.	1.0

1. Giriş

1.1 Ürün Misyon Beyanı

Dijital Restoran Rezervasyon ve Menü Sistemi, kullanıcı dostu altyapısıyla müşterilerin restoran deneyimlerini dijital ortamda kolayca yönetmelerini, rezervasyon işlemlerini gerçekleştirmelerini ve menüleri detaylı bir şekilde incelemelerini hedefler. Sistem, müşteriler ile restoran yöneticileri arasında etkili bir etkileşim ortamı sağlayarak, restoran hizmetlerine erişimi kolaylaştırır ve müşteri memnuniyetini en üst düzeye çıkarmayı amaçlar.

Geliştirilen sistem; müşterilerin rezervasyon sürecinden siparişe kadar olan tüm işlemleri zahmetsizce tamamlamasına olanak tanırken, yöneticilere masa takibi, menü güncellemeleri ve rezervasyon yönetimi gibi operasyonel süreçleri etkin bir şekilde yürütme imkânı sunar. Bu sayede, hem müşteriler hem de restoran işletmecileri için verimli, düzenli ve kaliteli bir dijital restoran deneyimi oluşturulması hedeflenmektedir.

1.2 Kapsam

Sistem, kullanıcıların restoran bilgilerine, menü içeriklerine, masaların dolu olup - olmama durumuna ve rezervasyon takvimine hızlı ve kolay bir biçimde erişmesini sağlar. Kullanıcılar, sistem üzerinden restoranlara rezervasyon yapabilir, menüleri inceleyebilir, sipariş oluşturabilir ve ziyaret ettikleri restoranları puanlayarak yorum bırakabilirler. Bu sayede kullanıcı deneyimi sistemi sürekli gelişen ve değerlendirilen bir yapıya dönüştürür.

Aynı zamanda restoran yöneticileri, tüm bu verileri kolayca görüntüleyip düzenleyebilir; masa yönetimi, menü güncellemeleri ve rezervasyon takibini gerçekleştirebilir. Rezervasyonlar, kullanıcı profilleri, yorumlar ve sipariş bilgileri gibi işlemler sistemde uyumlu bir biçimde yürütülür.

Dijital Restoran Rezervasyon ve Menü Sistemi; müşteriler, restoran çalışanları ve yöneticiler arasındaki iletişimi sorunsuz bir şekilde sağlamayı hedefleyerek, restoran hizmetlerinin kalitesini artırır ve müşteri memnuniyetini üst düzeye taşır. Sistemin kullanıcı dostu yapısı sayesinde restoran yönetimi daha etkili ve yenilikçi çözümlerle gerçekleştirilir.

1.3 Tanımlamalar ve Kısaltmalar

Kullanıcı	Sistemi restoran müşterisi olarak kullanan kişi.
Yönetici	Restoranın sistemden sorumlu personeli.
Spam	Kullanıcının onayı olmadan gönderilen istenmeyen veya tekrarlayıcı içerikler.

JWT	JSON Web Token; kullanıcı kimlik doğrulaması ve veri iletimi için kullanılan güvenli bir token formatıdır.
İndeksleme	Veritabanında veri aramalarını hızlandırmak için verilerin belirli bir düzene göre sıralanması işlemi
Cacheleme	Sık kullanılan verilerin geçici olarak hızlı erişim için saklanması işlemi.
HTTPS	Hypertext Transfer Protocol Secure; web tarayıcıları ve sunucular arasında güvenli veri iletimi sağlamak için şifreleme kullanan HTTP protokolünün güvenli versiyonudur.
KVKK	Kişisel Verilerin Korunması Kanunu; Türkiye'de kişisel verilerin işlenmesi ve korunmasına ilişkin düzenlemeleri içeren kanundur.
GDPR	General Data Protection Regulation; Avrupa Birliği'nde kişisel verilerin korunmasına yönelik yasalar ve düzenlemeler getiren genel veri koruma yönetmeliğidir.
reCaptcha	Google tarafından geliştirilen, kullanıcıların insan olduğunu doğrulamak için kullanılan, robotlara karşı koruma sağlayan güvenlik sistemi.
SMS	Short Message Service; mobil cihazlar arasında kısa metin mesajlarının gönderilmesini sağlayan hizmettir
Chatbot	Kullanıcılarla otomatik olarak metin veya sesli sohbet yapan, genellikle yapay zeka ile desteklenen yazılımdır
E-posta	Elektronik posta; internet üzerinden yazılı mesajların gönderilip alınmasını sağlayan dijital iletişim aracıdır
MySQL	Açık kaynaklı bir veritabanı yönetim sistemi (DBMS); verilerin depolanması, yönetilmesi ve sorgulanması için kullanılan, SQL (Structured Query Language) dilini kullanan popüler bir sistemdir.
REST API	Representational State Transfer (REST) mimarisiyle tasarlanmış, web uygulamaları arasında veri iletişimini sağlayan, HTTP protokolünü kullanan bir API (Application Programming Interface) türüdür.
SQLite	Hafif, disk tabanlı, ilişkilendirilmiş veritabanı yönetim sistemi (DBMS); özellikle yerel ve gömülü uygulamalarda veri depolamak için yaygın olarak kullanılır.
PostgreSQL	İleri düzey, açık kaynaklı ilişkisel veritabanı yönetim sistemi (DBMS); karmaşık veri yapılarını ve büyük veri setlerini yönetmek için yüksek performans ve esneklik sağlar.
UI	User Interface; bir kullanıcının bir yazılım uygulamasıyla etkileşimde bulunduğu görsel ve işitsel öğelerin toplamıdır.
UX	User Experience; bir kullanıcının bir ürün veya hizmetle etkileşimi sırasında yaşadığı genel deneyimi ve memnuniyetini ifade eder.
Endpoint	Bir ağda veya API'de veri alışverişinin yapıldığı, genellikle belirli bir işlevi gerçekleştiren uç noktadır.
Mockup	Bir ürünün, özellikle bir yazılım veya web tasarımının, tasarım aşamasında görsel olarak temsil edilen, işlevsel olmayan prototipi.
Wireframe	Bir web sitesinin veya uygulamanın temel yapısını ve düzenini görsel olarak gösteren, işlevsellikten çok yapısal öğelere odaklanan tasarım şablonudur

SSL	Secure Sockets Layer; internet üzerinden güvenli veri iletimi sağlamak için kullanılan eski bir şifreleme protokolüdür, günümüzde yerini TLS (Transport Layer Security) almıştır.
TLS	Transport Layer Security; internet üzerinden güvenli veri iletimi sağlamak için kullanılan şifreleme protokolüdür ve SSL'in yerini almıştır.
HSM	Hardware Security Module; kriptografik anahtarların güvenli bir şekilde depolandığı ve işlem yapıldığı fiziksel cihazdır
IDS	Intrusion Detection System; ağ veya sistemlere yapılan izinsiz girişleri tespit etmek ve raporlamak için kullanılan güvenlik sistemi.
IPS	Intrusion Prevention System; ağ veya sistemlere yapılan izinsiz girişleri tespit edip, bu tür saldırıları engelleyen güvenlik sistemi.
HTML5	HyperText Markup Language 5; web sayfalarını yapılandırmak için kullanılan, metin, medya ve etkileşimli öğeleri tanımlayan, modern web standartlarını destekleyen işaretleme dilidir.
CSS3	Cascading Style Sheets 3; web sayfalarının görünümünü, düzenini ve stilini tanımlamak için kullanılan, HTML ile birlikte çalışan modern stil sayfası dilinin üçüncü sürümüdür.
JavaScript	Web sayfalarında etkileşimli özellikler ve dinamik içerik sağlamak için kullanılan, tarayıcı tabanlı bir programlama dilidir.
ORM	Object-Relational Mapping; nesne tabanlı programlama dillerindeki nesneler ile ilişkisel veri tabanlarındaki veriler arasındaki dönüşümü sağlayan teknik ve kütüphanedir.
RBAC	Role-Based Access Control; kullanıcılara belirli roller atayarak, sistem kaynaklarına erişim izinlerini yönetmeye yönelik bir güvenlik modelidir.
CSRF	Cross-Site Request Forgery; bir kullanıcının kimliğiyle izinsiz işlem yapmaya yönelik bir saldırı türüdür, genellikle web uygulamalarında güvenlik açıklarından faydalanır.
XSS	Cross-Site Scripting; kötü niyetli kullanıcıların, güvenli olmayan web uygulamalarına zararlı JavaScript kodları enjekte ederek diğer kullanıcıların tarayıcılarında çalıştırılmasını sağladığı bir güvenlik açığıdır.
NLP	Natural Language Processing; bilgisayarların insan dilini anlaması, analiz etmesi ve işlem yapması için kullanılan yapay zeka alanıdır.

1.4 Kaynaklar

Requirements Engineering for Software and Systems (2017 Phillip A. Laplante)
IEEE Software Requirements Specification
Mevcut benzer sistem (Zomato / https://www.zomato.com)
Yapay zeka uygulaması (OpenAI GPT-3 / https://openai.com/gpt-3)
Çevrimiçi diyagram ve akış şeması yazılımı (Lucidchart / https://www.lucidchart.com)

1.5 Dokümana Genel Bakış

Bu belgede, Dijital Restoran Rezervasyon ve Menü Sistemine ait genel bir tanımlama sunulacak, sistemin yapısı, kullanıcı türleri ve kullanım senaryoları açıklanacaktır. Ayrıca, sistemin harici arayüz gereksinimleri, işlevsel ve işlevsel olmayan gereksinimleri detaylı biçimde ele

alınacaktır. Bu gereksinimlerin öncelik düzeyleri ve sistem üzerindeki etkileri de sınıflandırılarak değerlendirilecektir.

Belge, yazılım geliştirme sürecinde ekibe rehberlik etmek ve sistemin ne şekilde çalışacağını netleştirmek amacıyla hazırlanmıştır.

2. Genel Tanım

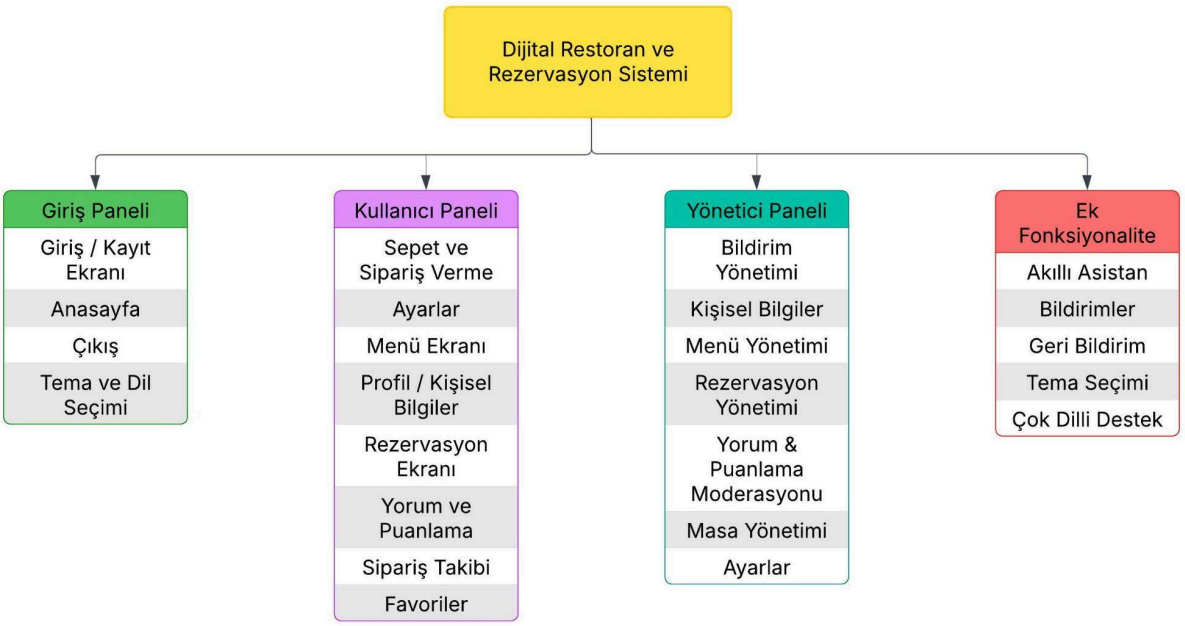
2.1 Ürün Perspektifi

Dijital Restoran Rezervasyon ve Menü Sistemi; Web tabanlı platformda ve mobil uygulamada; müşteri ve yönetici rollerine göre ayrılmış; kullanıcılara ait olduğu role göre kullanım olanağı sağlayan bir sistemdir. Sistem, modern yazılım mimarisi ilkelerine uygun olarak sunucu-istemci modelinde çalışmakta ve RESTful servisler aracılığıyla tüm kullanıcı etkileşimlerini yönetmektedir. Bu sistem, mevcutta kullanılan manuel veya parçalı yapıdaki yöntemlerin yerini alarak; rezervasyon, menü yönetimi, sipariş alma, kullanıcı yorum ve puanlamaları ile favori ekleme gibi işlemleri dijital ortamda tek bir platform üzerinden bütünleşik şekilde sunmayı hedeflemektedir. Hem masaüstü hem de mobil cihazlarda kolay kullanılabilir bir arayüzle kullanıcıya konforlu bir deneyim sağlamayı amaçlamaktadır.

Ürün, aşağıdaki sistemlerle etkileşim kurabilir:

Veritabanı Sunucusu: Tüm kullanıcı verileri, rezervasyon kayıtları, menüler, yorumlar ve puanlama bilgileri burada tutulur.

Bildirim Sistemi : Rezervasyon onayları ve hatırlatmalar için e-posta ya da mobil bildirim desteği sağlanabilmektedir .



Resim 1. Dijital Restoran Rezervasyon ve Menü Sistemi Şeması

2.2 Kullanıcı Özellikleri

2.2.1 *Müşteri :*

Restorandan hizmet alan ,sisteme kayıt olup ve giriş yapan bireysel kullanıcılardır. Sipariş ve rezervasyon gibi temel hizmetleri kullanmakla birlikte, sistemde sadece kendi geçmiş bilgilerine ve aktivitelerine erişebilir. Görüntüleme ve işlem yetkileri kişisel kullanım ile sınırlıdır.Müşteriler kullanıcı dostu bir arayüzle sisteme erişir ve temel işlevleri sade bir yapıda kullanabilir.

2.2.2 *Yönetici:*

Sistemin tüm işleyişinden sorumlu olan yetkili kullanıcıdır. Yönetici rolü, sistemin sürdürülebilirliğini sağlamak ve gerektiğinde kullanıcı etkileşimlerine müdahale etmek amacıyla daha geniş kapsamlı erişim ve kontrol imkânına sahiptir. Yönetici arayüzü, operasyonel işlemlere yönelik detaylı kontrol seçenekleriyle donatılmıştır.

2.3. Sistem İşlevleri

2.3.1 *Müşteri(Kullanıcı):*

Oluşturulan sistemde müşteriler, rezervasyon yapma ,menü görüntüleme , sepetlerinde ekleme-çıkarma, sipariş verme işlemlerini gerçekleştirebilmektedir.
Ürüne(yemek/tatlı/içecek),restorana; yorum, puanlama gibi geri bildirimlerde bulunabilmekte; beğendikleri ürünü favorileyerek kolayca o ürüne ulaşabilmektedir .

2.3.2 *Yönetici:*

Sistemde yöneticiler rezervasyonları ,menüleri ve siparişleri görüntüleyebilmektedir. Rezervasyon ve menüler üzerinde düzenleme yapabilmektedir ayrıca müşterilerin yapmış olduğu yorumları görüntüleyebilmekte ve cevap verebilmektedirler.

2.4 Kısıtlar

Kullanıcı oturumları güvenli şekilde yönetilmelidir;yorumlar spam ve küfür içeriğine karşı filtrelenmelidir. Kullanıcı- yönetici rolleri ve website girişleri ayrılmalıdır. Rezervasyonlar çakışmalar önlenmelidir ve eş zamanlı yüksek kullanıcı trafiğinde sistemin yavaşlamaması için çözümler üretilmelidir.Sistem; mobil cihazlar, tabletler ve masaüstü tarayıcılarla uyumlu çalışmalıdır .

3. Özel Gereksinimler

Bu bölüm, sistemin tasarımı, gerçekleştirilmesi ve test edilmesi sürecinde temel alınacak teknik gereksinimleri içerir. Gereksinimler doğrulanabilir, izlenebilir ve tutarlı olacak şekilde yapılandırılmıştır.

3.1 Dış Arabirim Gereksinimleri

3.1.1 Kullanıcı Arayüzleri

3.1.1.1 Müşteri Arayüzü

3.1.1.1.1 Mobil ve web destekli olmalıdır.

3.1.1.1.2 Rezervasyon oluşturma, iptal etme ve görüntüleme ekranları bulunacaktır.

3.1.1.1.3 Menü kategorileri (yemek, içecek, tatlı) ve ürün detayları (malzemeler, alerjen bilgileri,fiyatlar) sunulacaktır.
3.1.1.1.4 Sipariş verme ve ödeme işlemleri desteklenecek, QR kod ile menü erişimi sağlanacaktır.
3.1.1.1.5 İsteğe bağlı olarak kullanıcı hesabı oluşturma ve giriş yapma ekranları bulunacaktır.

3.1.1.2 Yönetici / Personel Arayüzü
3.1.1.2.1 Rezervasyon takibi ile onay veya red yönetimi yapılabilecektir.
3.1.1.2.2 Menü düzenleme ve içerik yönetimi paneli sağlanacaktır.
3.1.1.2.3 Masa yönetimi (masa durumu ve kapasite bilgisi) yapılabilecektir.
3.1.1.2.4 Günlük raporlama ve veri analizi ekranları bulunacaktır.
3.1.1.3 Tasarım Özellikleri
3.1.1.3.1 Kullanıcı arayüzü mobil cihazlar ve tabletler için duyarlı (responsive) tasarım ilkelerine uygun olacaktır.
3.1.1.3.2 Erişilebilirlik (accessibility) kriterleri göz önünde bulundurularak geliştirilecektir.

3.1.2 İletişim Arabirimleri
3.1.2.1 İnternet Protokolleri
HTTP/HTTPS protokolleri kullanılacaktır. SSL sertifikası ile veri güvenliği sağlanacaktır.
3.1.2.2 Bildirim Sistemleri
E-posta ile rezervasyon onayı ve hatırlatma yapılacaktır.
Mobil cihazlara anlık bildirim (push notification) gönderilecektir.
3.1.2.3 Kullanıcıya Geri Bildirim
Sistem tarafından yapılan işlemlerde başarılı veya başarısız durum mesajları sunulacaktır.
Geribildirim formu ve yorum/review bölümü ile kullanıcı görüşleri restoran yönetimine iletilebilecektir.

3.2 Fonksiyonel Gereksinimler

Bu bölümde, Dijital Restoran Rezervasyon ve Menü Sistemi'nin işlevsel özellikleri tanımlanacaktır. Her gereksinim, belirli bir işlevi veya özellik setini kapsar.

3.2.1 Rezervasyon Yapma
3.2.1.1 Giriş
3.2.1.1.1 Kullanıcılar, sistem üzerinden restoranlara rezervasyon yapabilecektir. Rezervasyon işlemi, kullanıcıların istedikleri tarih ve saat için masa ayırmalarını sağlar.
3.2.1.2 Girişler
3.2.1.2.1 Kullanıcı rezervasyon yapmak istediği tarih ve saati seçer.
3.2.1.2.2 Kullanıcı kişi sayısını belirtir.
3.2.1.2.3 Kullanıcı rezervasyon yapılacak restoranı seçer.
3.2.1.2.4 Kullanıcı ad, soyad ve telefon numarası gibi kişisel bilgilerini girer.
3.2.1.3 İşleme
3.2.1.3.1 Sistem, belirtilen tarih ve saat için masa doluluk durumunu kontrol eder.

3.2.1.3.2 Sistem, masa kapasitesine göre rezervasyon uygunluğunu doğrular.
3.2.1.3.3 Sistem, kullanıcıdan kimlik doğrulama işlemini e-posta/ telefon üzerinden gerçekleştirir.

3.2.1.4 Çıkışlar
3.2.1.4.1 Rezervasyon onayı kullanıcıya sistem üzerinden ve e-posta/SMS ile iletilir.
3.2.1.4.2 Kullanıcı, onaylanan rezervasyon bilgilerini görüntüleyebilir.

3.2.1.5 Hata İşleme
3.2.1.5.1 Geçersiz bilgi girilirse sistem kullanıcıyı hata mesajı ile bilgilendirir.

3.2.2 Menü Görüntüleme

3.2.2.1 Giriş
3.2.2.1.1 Kullanıcılar, restoran menüsünü detaylı şekilde inceleyebilir. Menüdeki ürünler isim, açıklama ve fiyat bilgileriyle sunulur.

3.2.2.2 İşleme
3.2.2.2.1 Sistem, restoranın güncel menüsünü veri tabanından çeker.
3.2.2.2.2 Menü öğeleri kategorilere göre kullanıcıya sunulur.
3.2.2.2.3 Açıklama, içerik bilgisi (alerjenler) ve fiyat bilgisi erişilebilir şekilde düzenlenir.

3.2.2.3 Çıkışlar
3.2.2.3.1 Menü öğelerinin adı, açıklaması ve fiyatı kullanıcıya gösterilir.
3.2.2.3.2 Kullanıcı, istediği öğeleri sepete ekleyebilir veya sipariş verebilir.

3.2.2.4 Hata İşleme
3.2.2.4.1 Menü verisi bulunamazsa kullanıcıya bilgilendirme yapılır.
3.2.2.4.2 Fiyatlandırma veya içerik hatalarında sistem yöneticisine uyarı gönderilir.

3.2.3 Sipariş Verme

3.2.3.1 Giriş
3.2.3.1.1 Kullanıcılar menüden seçim yaparak sipariş oluşturabilir. Sipariş, restoranın mutfak sistemine iletilir.

3.2.3.2 Girişler
3.2.3.2.1 Kullanıcı, menüden yemek, içecek veya tatlı seçer.
3.2.3.2.2 Kullanıcı, sipariş miktarını belirtir.
3.2.3.2.3 Kullanıcı, ödeme yöntemini (kredi kartı, mobil ödeme) seçer.

3.2.3.3 İşleme
3.2.3.3.1 Sistem, siparişi alır ve mutfak sistemine yönlendirir.
3.2.3.3.2 Ödeme işlemi, seçilen yönteme göre işlenir.
3.2.3.3.3 Sipariş durumu (hazırlanıyor, hazır, teslim edildi) takip edilir.

3.2.3.4 Çıkışlar
3.2.3.4.1 Kullanıcıya sipariş özeti ve toplam ücret sunulur.
3.2.3.4.2 Sipariş durumu kullanıcıya bildirilir.
3.2.3.4.3 Sipariş başarıyla tamamlandığında kullanıcıya bildirim gönderilir.

3.2.3.5 Hata İşleme
3.2.3.5.1 Ödeme başarısız olursa kullanıcı bilgilendirilir ve yeniden denemesi istenir.
3.2.3.5.2 Sistemsel bir hata oluşursa kullanıcıya özür mesajı ve destek seçeneği sunulur.

3.3 İşlevsel Olmayan Gereksinimler

İşlevsel olmayan gereksinimler, sistemin performansını, güvenilirliğini, kullanılabilirliğini, güvenliğini, sürdürülebilirliğini ve taşınabilirliğini tanımlar. Bu gereksinimler genellikle sistem genelinde uygulanmalı ve ölçülebilir şekilde belirtilmelidir.

3.3.1 Performans

3.3.1.1 Sistem, kullanıcı isteklerinin %95'ini 1 saniyeden kısa sürede işlemelidir.
3.3.1.2 Aynı anda en az 100 eşzamanlı kullanıcı desteklenmelidir.
3.3.1.3 Veritabanı sorgularının ortalama cevap süresi 500 ms'yi aşmamalıdır.
3.3.1.4 Sipariş oluşturma işlemi maksimum 2 saniye içinde tamamlanmalıdır.

3.3.2 Güvenilirlik

3.3.2.1 Sistemin MTBF (arızasız geçen ortalama süre) değeri en az 30 gün olmalıdır.
3.3.2.2 Günlük sistem kesinti süresi toplamda 1 dakikayı geçmemelidir.
3.3.2.3 Herhangi bir sistem hatası durumunda, sistem 3 saniye içinde yeniden başlatılabilir durumda olmalıdır.
3.3.2.4 Veri kaybı riski olan durumlarda otomatik yedekleme sistemi çalışmalıdır.

3.3.3 Kullanılabilirlik

3.3.3.1 Sistemin erişilebilirlik oranı %99.9 olmalıdır.
3.3.3.2 Yeni bir kullanıcı, sistem arayüzünü ilk defa kullanırken 5 dakikadan kısa sürede temel işlevleri (rezervasyon, sipariş, menü görüntüleme) gerçekleştirmelidir.
3.3.3.3 Kullanıcı arayüzü, farklı cihaz boyutlarına (mobil, tablet, masaüstü) uyumlu (responsive) olmalıdır.
3.3.3.4 Yardım dökümanları ve rehber içerikler, kullanıcılar için sistemde 7/24 erişilebilir olmalıdır.

3.3.4 Güvenlik

3.3.4.1 Tüm kullanıcı verileri AES-256 şifreleme ile korunmalıdır.
3.3.4.2 Giriş ve şifre işlemleri SSL/TLS protokolü üzerinden gerçekleştirilmelidir.
3.3.4.3 Sistem, yetkisiz erişimlere karşı koruma sağlayan güvenlik duvarı ve kullanıcı rol denetimi içermelidir.
3.3.4.4 Kullanıcı oturumları, 15 dakikalık hareketsizlikten sonra otomatik olarak sonlandırılmalıdır.

3.3.5 Sürdürülebilirlik

3.3.5.1 Sistem mimarisi, en az 3 yıl boyunca bakım ve güncelleme yapılabilir yapıda olmalıdır.
3.3.5.2 Kaynak kullanımı optimize edilerek sistemin CPU ve bellek tüketimi %70'in altında tutulmalıdır.
3.3.5.3 Yazılım güncellemeleri sırasında kullanıcı verisi ve sistem kararlılığı korunmalıdır.
3.3.5.4 Açık kaynak veya belgelenmiş teknolojilerin kullanılması tercih edilerek gelecekteki geliştirme maliyetleri azaltılmalıdır.

3.3.6 Taşınabilirlik

3.3.6.1 Sistem, farklı işletim sistemlerinde (Windows, Linux) çalışabilir yapıda olmalıdır.

3.3.6.2 Veritabanı bağımsızlığı gözetilerek, sistemin PostgreSQL, MySQL gibi veritabanlarıyla uyumlu olması hedeflenmiştir.

3.3.6.3 Web tabanlı sistem, farklı tarayıcılarda (Chrome, Firefox, Edge, Safari) %100 uyumlu şekilde çalışmalıdır.

3.3.6.4 Sistem bileşenleri, bulut altyapısına kolayca entegre edilebilir ve taşınabilir olmalıdır.

3.4 Ters Gereksinimler (Negative Requirements)

Ters gereksinimler, sistemin yapmaması gereken davranışları tanımlar. Bu tür gereksinimler, kullanıcı güvenliğini sağlamak, hatalı kullanımın önüne geçmek ve sistemin sınırlarını belirlemek açısından önemlidir.

3.4.1 Yetkisiz Erişim

3.4.1.1 Sistem, yetkilendirilmemiş kullanıcıların özel verilere erişmesine izin vermemelidir.

3.4.2 Mükerrer İşlem

3.4.2.1 Sistem, aynı işlemi birden fazla defa yapmaya çalışan kullanıcılara mükerrer işlem gerçekleştirmemelidir (örneğin: aynı siparişin birden fazla kaydedilmesi).

3.4.3 Eşzamanlı Giriş

3.4.3.1 Sistem, bir kullanıcı oturumu aktifken, başka bir cihazdan eşzamanlı giriş yapılmasına izin vermemelidir.

3.4.4 Hatalı Veri Girişi

3.4.4.1 Sistem, eksik veya hatalı veri girişinde, işlemi gerçekleştirmemelidir ve kullanıcıyı uygun şekilde uyarmalıdır.

3.4.5 Veri Kaybı

3.4.5.1 Sistem, beklenmeyen sistem hatalarında veri kaybına neden olmamalıdır.

3.5 Tasarım Kısıtlamaları

Sistem geliştirme sürecinde aşağıdaki tasarım kısıtlamaları göz önünde bulundurulmalıdır:

3.5.1 Donanım Kısıtlamaları

3.5.1.1 Sistem, düşük donanımlı cihazlarda (4GB RAM, çift çekirdekli işlemci) performanslı şekilde çalışacak şekilde optimize edilmelidir.

3.5.2 Yazılım Kısıtlamaları

3.5.2.1 Kullanılacak olan işletim sistemi Windows 10 ve üzeri ile Linux dağıtımları (Ubuntu, Fedora) ile uyumlu olmalıdır.

3.5.3 Veritabanı

3.5.3.1 Veritabanı olarak PostgreSQL kullanılacaktır.

3.5.4 Geliştirme Ortamı

3.5.4.1 Yazılım geliştirme süreci için .NET Core 8 kullanılacaktır.

3.5.5 Geliştirme Süresi

3.5.5.1 Proje geliştirme süresi maksimum 3 ay ile sınırlıdır.

3.5.6 Standartlar

3.5.6.1 Kod yazımı ve test süreçleri, IEEE 830 ve ISO/IEC 25010 standartlarına uygun olmalıdır.

3.5.7 Güvenlik Politikası

3.5.7.1 Üniversite veya kurum politikalarına uygun olarak KVKK (Kişisel Verileri Koruma Kanunu) ile uyumlu veri işleme süreci izlenecektir.

3.6 Mantıksal Veritabanı Gereksinimleri

Projede ilişkisel bir veritabanı kullanılacaktır. Veritabanı, uygulamanın kullanıcı, ürün, sipariş, rezervasyon, yorum ve sistem logları gibi temel bileşenlerini depolayacaktır.

3.6.1 Veri Formatları

3.6.1.1 Metin verileri için VARCHAR veri tipi kullanılacaktır.

3.6.1.2 Sayısal değerler için INT ve FLOAT veri tipleri kullanılacaktır.

3.6.1.3 Tarih ve saat bilgileri için TIMESTAMP veri tipi kullanılacaktır.

3.6.2 Veri Saklama ve Yedekleme

3.6.2.1 Tüm veriler günlük olarak yedeklenecek ve 30 gün süreyle saklanacaktır.

3.6.2.2 Silinen veriler, geri yüklenebilir şekilde arşivlenmelidir.

3.6.3 Veri Bütünlüğü

3.6.3.1 Tüm tablolar arasında ilişkisel bütünlük (foreign key) sağlanacaktır.

3.6.3.2 Kullanıcı şifreleri, hashlenmiş şekilde veri tabanında saklanacaktır.

3.6.3.3 Zorunlu alanlar, NOT NULL kısıtlamalarıyla tanımlanacaktır.

3.6.4 Veri Erişimi ve Yetkilendirme

3.6.4.1 Kullanıcı rollerine göre (yönetici, çalışan, müşteri) veritabanı erişimi sınırlandırılacaktır.

3.6.4.2 Veritabanı sorguları için ORM (Entity Framework Core) kullanılacaktır.

3.7 Diğer Gereksinimler

Aşağıdaki ek gereksinimler proje kapsamına dahil edilmiştir:

3.7.1 Çok Dilli Destek

3.7.1.1 Uygulama, başlangıçta Türkçe ve İngilizce dil desteğine sahip olacaktır.

3.7.2 Bildirim Sistemi

3.7.2.1 Kullanıcılar, rezervasyon, onay veya değişiklik durumlarında e-posta ve uygulama içi bildirim alacaktır.

3.7.3 API Desteği

3.7.3.1 Dış sistemlerle entegrasyon için RESTful API desteği sunulacaktır.

3.7.4 Tema Desteği

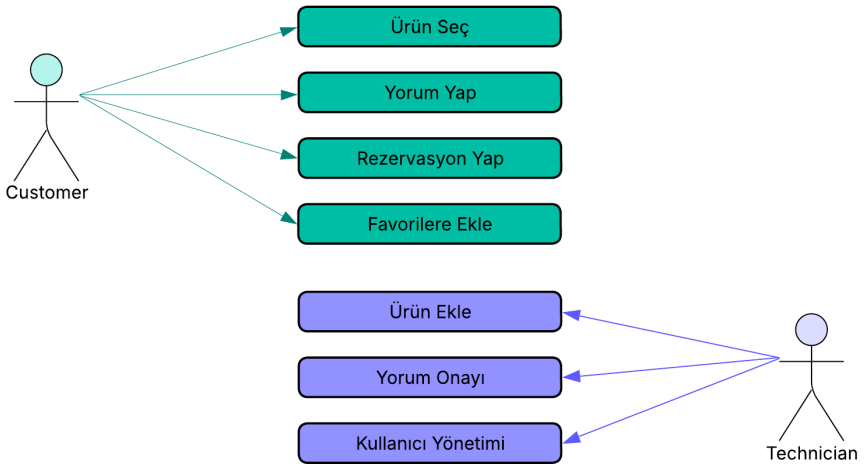
3.7.4.1 Kullanıcı arayüzü, açık ve koyu tema seçenekleri içerecektir.

3.7.5 Test Süreci

3.7.5.1 Sistem, geliştirme sonunda birim testi, entegrasyon testi ve kullanıcı kabul testine tabi tutulacaktır.

3.7.6 Erişilebilirlik

3.7.6.1 Arayüz, renk körlüğü ve benzeri özel ihtiyaçlara sahip kullanıcılar için WCAG 2.1 standartlarına uyumlu tasarlanacaktır.



Resim 2. Dijital Restoran Rezervasyon ve Menü Sistemi Use Case Diyagramı

4. Arayüz Gereksinimleri

4.1 Kullanıcı Paneli

4.1.1 Kullanıcı Girişi

4.1.1.1 Giriş ekranında e-posta ve şifre giriş alanları yer alacaktır.

4.1.1.2 Giriş butonu, alanlar geçerli şekilde doldurulup reCaptcha doğrulaması tamamlandığında aktif hale gelir.

4.1.1.3 “Şifremi unuttum” butonu ile parola sıfırlama ekranına yönlendirme yapılır.

4.1.1.4 "Beni hatırla" seçeneğiyle tarayıcıda oturum bilgisi belirli süreyle saklanabilir.

4.1.1.5 Yeni kullanıcılar için “Kayıt Ol” butonu görünür şekilde yer alır.

4.1.2 Anasayfa

4.1.2.1 Restoranlar, kampanyalar ve önerilen menüler gösterilir.

4.1.2.2 Arama çubuğu ile restoran veya yemek araması yapılabilir.

4.1.2.3 Hızlı rezervasyon seçeneği bulunur.

4.1.3 Kişisel Bilgiler

4.1.3.1 Kullanıcılar ad, soyad, e-posta gibi bilgilerini görüntüleyip güncelleyebilir.

4.1.3.2 Geçmiş rezervasyonlar ve siparişler listelenir.

4.1.3.3 Favori yemekler ve yapılan yorumlar bu alanda bulunur. Favorilere ekledikleri yemekler, kullanıcı profillerinde liste halinde gösterilir.

4.1.4 Rezervasyon Ekranı

4.1.4.1 Restoran seçimi, tarih ve saat belirlenerek uygun masa seçimi yapılır.

4.1.4.2 Kullanıcılar, dolu olan restoranlarda masa boşalınca bilgilendirilmek üzere "Boş Masa Bildirimi Al" butonuyla sistem üzerinden talep oluşturabilir; bu talep sistemde kaydedilmelidir.

4.1.4.3 Rezervasyon durumu (Bekliyor, Onaylandı, Reddedildi) izlenebilir.

4.1.4.4 Kullanıcıya rezervasyon hatırlatmaları gönderilir.

4.1.5 Menü Ekranı

4.1.5.1 Menü ürünleri kategoriye (ana yemek, içecek, tatlı) göre filtreleyebilir.

4.1.5.2 Her ürün için görsel, açıklama, fiyat, kullanıcı puanı ve yorumlar yer alır.

4.1.5.3 Favorilere ekleme ve sipariş verme butonları bulunur. Favorilere ekleme, yemek görselinin sağ üst köşesine içi boş kalp butonu ile yapılmalıdır. Sipariş verme ise yemek görselinin altında bulunan sepete ekle butonu ile yapılmalıdır.

4.1.5.4 Kullanıcılar, istedikleri zaman favorilerinden yemekleri kaldırabilmelidir.

4.1.6 Sipariş Takibi

4.1.6.1 Kullanıcılar, verdikleri siparişleri (hazırlanıyor, hazır, teslim edildi) görüntüleyebilir.

4.1.6.2 Siparişe ait tahmini teslim süresi gösterilir.

4.1.6.3 Sipariş geçmişi kullanıcı profiline kaydedilir.

4.1.7 Yorum Yapma ve Puanlama

4.1.7.1 Kullanıcılar, sipariş edilen yemek için yorum yapabilir.

4.1.7.2 Yorum yapmayan kullanıcı, yemeklerin altındaki yorumlara erişebilir. Yorumları okur fakat değişim yapamaz.

4.1.7.3 Kullanıcılar, kendi yazdıkları yorumları düzenleme ve silme işlemlerini gerçekleştirebilir.

4.1.7.4 Kullanıcılar, yemeklerin yanına yerleştirilen 1 ile 5 arasında bir yıldızla, yemeği puanlayabilir. 1 yıldız en düşük, 5 yıldız en yüksek puan anlamına gelir. Kullanıcılar puan verirken, ekran üzerinde boş veya dolu yıldız simgeleri görecektir.

4.1.8 Ayarlar

4.1.8.1 Kullanıcı kişisel tercihine göre temayı (koyu/açık) seçebilir.

4.1.8.2 Hesap silme ve parola değiştirme işlemleri yapılabilir.

4.1.9 Çıkış

4.1.9.1 Kullanıcı çıkış butonuna bastığında sistemden çıkış yapılır.

4.1.9.2 Oturum süresi dolunca otomatik çıkış gerçekleşir.

4.2 Yönetici Paneli

4.2.1 Rezervasyon Yönetimi

4.2.1.1 Tüm rezervasyonlar listelenir veya filtrelenebilir.

4.2.1.2 Zaman değişikliği, iptal etme gibi işlemler yapılabilir.

4.2.1.3 Müşterilere rezervasyon onaylandığında ve rezervasyon saatinden 2 saat önce e-posta veya SMS bildirimi gönderilir.

4.2.2 Masa Yönetimi

4.2.2.1 Masa ekleme, düzenleme ve silme yetkileri sağlanır.

4.2.2.2 Masa kapasitesi, konumu ve doluluk durumu güncellenebilir.

4.2.2.3 Bildirim talebi oluşturan kullanıcılara masa boşaldığında sistem üzerinden bildirim (e-posta veya SMS ile) gönderebilir.

4.2.3 Menü Yönetimi

4.2.3.1 Yönetici; ürün ekleme, çıkarma, fiyat ve içerik düzenleme yapabilir.

4.2.3.2 Yönetici ürünleri kategorilere göre sıralayabilir.

4.2.3.3 Yönetici kampanya ve indirim etiketleri tanımlayabilir.

4.2.4 Yorum ve Puan Moderasyonu

4.2.4.1 Yönetici kullanıcı yorumlarını inceleyip onaylayabilir veya silebilir.

4.2.4.2 Uygunsuz içerikler filtrelenebilir (küfür, spam, reklam).

4.2.4.3 Yorumlar tarihe ve puana göre sıralanabilir, analiz yapılabilir.

4.2.4.4 Yemeklerin yanında, o yemeğe verilen ortalama puan ve toplam puan sayısı görüntülenebilir.

4.3 Donanım Arayüzleri

4.3.1 Kullanıcı Cihazları

4.3.1.1 Akılı telefon, tablet ve bilgisayar üzerinden erişim desteklenir.

4.3.2 Tarayıcı Uyumluluğu

4.3.2.1 Sistem Google Chrome, Opera, Microsoft Edge, Opera ve Safari tarayıcılarının en güncel sürümlerinde tam uyumlu olarak çalışacaktır.

4.3.3 Ekran Çözünürlüğü

4.3.3.1 Sistem, minimum 360x640 piksel çözünürlüğe sahip mobil cihazlarda işlevsel ve kullanıcı dostu bir şekilde çalışacak şekilde tasarlanacaktır.

4.3.3.2 Sistem, masaüstü ve dizüstü bilgisayarlarda 1366x768 piksel ve üzeri çözünürlüklerde tam uyumlu ve optimize edilmiş kullanıcı arayüzü sağlayacaktır.

4.3.3.3 Uygulama arayüzü, responsive tasarım ilkelerine uygun olarak farklı cihaz boyutlarında tutarlı bir görünüm ve kullanılabilirlik sunacak şekilde geliştirilecektir.

4.3.4 Güvenlik Donanımı

4.3.4.1 Sistem sunucuları, istemci ile sunucu arasındaki veri iletimini SSL/TLS protokolleri ile güvence altına almalı; şifreleme anahtarları, güvenli şekilde HSM (Hardware Security Module) aracılığıyla saklanmalı ve yönetilmelidir.

4.3.4.2 Sunucu altyapısı, zararlı yazılımlara ve yetkisiz erişimlere karşı donanımsal güvenlik önlemleriyle korunmalıdır. Bu kapsamda; donanımsal güvenlik duvarı (hardware firewall), IDS/IPS sistemleri, güncel antivirüs çözümleri uygulanmalıdır.

4.4 Yazılım Arayüzleri

4.4.1 Web Arayüz Teknolojileri

4.4.1.1 Kullanıcı arayüzü, web tarayıcılarında çalışacak şekilde HTML5, CSS3 ve JavaScript teknolojileri kullanılarak geliştirilecektir. Responsive tasarımı desteklemek amacıyla gerekli framework ve kütüphaneler (örn. Bootstrap, React) entegre edilecektir.

4.4.2 Veritabanı Bağlantısı

4.4.2.1 Sistem, ilişkisel veritabanı olarak MySQL kullanılarak yapılandırılacaktır.

4.4.2.2 ORM (Object-Relational Mapping) kullanımıyla veritabanı işlemleri yönetilir.

4.4.3 Mobil Uygulama Entegrasyonu

4.4.3.1 Mobil uygulama, REST API ile sunucuya entegre olur.

4.4.4 Güvenlik Katmanları

4.4.4.1 Kullanıcı kimlik doğrulama işlemleri, JWT (JSON Web Token) teknolojisi ile yapılacaktır.

4.4.4.2 Sistemde, kullanıcıların erişim yetkileri rol bazlı erişim kontrolü (RBAC) prensibine göre belirlenecek ve yönetilecektir.

4.4.4.3 Kullanıcı arayüzleri üzerinden gelen formlarda CSRF (Cross-Site Request Forgery) koruma mekanizması uygulanarak isteklerin güvenilirliği sağlanacaktır.

4.4.4.4 Tüm giriş alanlarında XSS (Cross-Site Scripting) saldırılarına karşı sunucu tarafında filtreleme ve kaçış karakterleri kullanılarak veri güvenliği sağlanacaktır.

4.5 İletişim Arayüzleri

4.5.1 Geri Bildirim

4.5.1.1 Kullanıcılar sistemle ilgili ne düşünüyor? Bunun için geri bildirim formu doldurabilir.

4.5.1.2 Geri bildirimler sistem yöneticisine iletilir ve analiz edilebilir şekilde saklanır.

4.5.1.3 Geri bildirimler kategori (öneri, hata bildirimi, memnuniyet vb.) bazlı sınıflandırılır.

4.5.2 İletişim Bildirimleri

4.5.2.1 Rezervasyon onayı, iptal bilgisi ve sipariş durumu sistem tarafından kullanıcıya e-posta veya uygulama içi bildirim olarak iletilir.

4.5.2.2 Bildirim tercihleri kullanıcı profilinde ayarlanabilir.

5. Ek Gereksinimler

5.1 Akıllı Asistan

Sistem, kullanıcı deneyimini artırmak amacıyla yapay zeka destekli bir akıllı asistan içerecektir. Bu modül, kullanıcıların önceki etkileşimlerinden ve tercihlerinden öğrenerek daha kişiselleştirilmiş hizmetler sunar. Aynı zamanda chatbot entegrasyonu sayesinde anlık bilgilendirme ve yönlendirme imkânı sağlar.

5.1.1 Kişiselleştirilmiş Öneriler Sunma

5.1.1.1 Sistem, kullanıcıların önceki rezervasyonları, tercih ettikleri masa konumları (pencere kenarı, dış mekan) ve favori yemek seçimlerini analiz ederek, yeni rezervasyonlarda otomatik önerilerde bulunur.

5.1.1.2 Menü görüntüleme sırasında, geçmiş sipariş verilerine göre kişiselleştirilmiş yemek önerileri sunulur.

5.1.1.3 Kullanıcının sıklıkla tercih ettiği zaman dilimlerine göre, uygun tarih ve saatlerde önerilen rezervasyon zamanları gösterilir.

5.1.2 Chatbot Destekli Etkileşim

5.1.2.1 Sistem içerisinde yer alan chatbot, doğal dil işleme (NLP) teknolojisi sayesinde kullanıcılardan gelen yazılı talepleri anlayabilir ve uygun yanıtları anında verebilir.

5.1.5.2.2 Kullanıcılar, chatbot aracılığıyla aşağıdaki işlemleri gerçekleştirebilir:

5.1.2.2.1 Anlık rezervasyon sorgulama ve oluşturma

5.1.2.2.2 Güncel menüyü görüntüleme

5.1.2.2.3 Kampanya ve indirim bilgilerini öğrenme

5.1.2.2.4 Genel bilgilere ulaşma (çalışma saatleri, oturma düzeni, adres bilgileri)

5.1.2.3 Chatbot, hem web hem de mobil arayüzlerde entegre çalışarak 7/24 hizmet verecektir.

5.2 Veri Modeli ve Veritabanı Yapısı

5.2.1 Tablolar

5.2.1.1 Users Tablosu

5.2.1.1.1 id : Birincil anahtar, her kullanıcıya benzersiz kimlik atar

5.2.1.1.2 name : Kullanıcının tam adı

5.2.1.1.3 email : Kullanıcının e-posta adresi (benzersiz)

5.2.1.1.4 phone : Telefon numarası

5.2.1.1.5 role : Kullanıcı rolü (customer, admin)

5.2.1.1.6 password : Hash’lenmiş şifre

5.2.1.2 Reservations Tablosu

5.2.1.2.1 id : Rezervasyon kaydının birincil anahtarı

5.2.1.2.2 user_id : Rezervasyonu yapan kullanıcıyı işaret eden yabancı anahtar

5.2.1.2.3 table_id : Rezervasyonun ilişkilendirildiği masa

5.2.1.2.4 date : Rezervasyon tarihi

5.2.1.2.5 time : Rezervasyon saati

5.2.1.2.6 guests : Misafir sayısı
5.2.1.2.7 note : İsteğe bağlı açıklama

5.2.1.3 Tables Tablosu
5.2.1.3.1 id : Masa kaydının birincil anahtarı
5.2.1.3.2 name : Masanın tanımlayıcı adı/numarası
5.2.1.3.3 capacity : Maksimum kişi sayısı
5.2.1.3.4 status : available / reserved / out_of_service

5.2.1.4 MenuItems Tablosu
5.2.1.4.1 id : Menü öğesinin birincil anahtarı
5.2.1.4.2 name : Ürün adı
5.2.1.4.3 category : başlangıç / ana yemek / tatlı / içecek
5.2.1.4.4 price : Ürün fiyatı
5.2.1.4.5 description : Kısa açıklama
5.2.1.4.6 is_available : Stok durumu (boolean)

5.2.2 İlişkiler
5.2.2.1 Users → Reservations (1:N) : Bir kullanıcı birden fazla rezervasyon yapabilir.
5.2.2.2 Tables → Reservations (1:N) : Her masa farklı zaman dilimlerinde birden çok rezervasyonla çalışabilir.
5.2.2.3 MenuItems : Bağımsızdır; henüz rezervasyon veya kullanıcıyla doğrudan ilişkisi yoktur.

5.3 Riskler

5.3.1 SQLite Performans Kısıtları
Yüksek eş zamanlı kullanımda sorgu yanıt süreleri uzayabilir
Önlem: İleride PostgreSQL/ MySQL'e geçiş, indeksleme ve sorgu optimizasyonu

5.3.2 Yönetici Paneli Güvenliği
Zayıf doğrulama, yetkisiz erişime yol açabilir
Önlem: JWT/Session doğrulama, çok faktörlü kimlik doğrulama, IP filtreleme

5.3.3 Mobil Uyumluluk Sorunları
Responsive tasarım hataları kullanıcı deneyimini bozar
Önlem: Erken mobil testleri, CSS framework'leri ile uyumluluk

5.3.4 Yetersiz Kullanıcı Geri Bildirimi
Hatalı işlemler sonrası kullanıcı ne yapacağını bilemez
Önlem: Açıklayıcı hata mesajları, form doğrulama, kullanıcı rehberi

5.4 Geliştirme Takvimi

5.4.1 1. Hafta – Planlama & UI/UX
Kullanıcı hikâyeleri, wireframe'ler ve mockup'lar hazırlanır

5.4.2 2–3. Haftalar – Backend & Veritabanı
Model tanımlamaları, API endpoint'leri, SQLite entegrasyonu

5.4.3 4. Hafta – Frontend Entegrasyonu
React/Vue ile arayüz geliştirme, REST API bağlantıları

5.4.4 5. Hafta – Test & Hata Ayıklama
Birim, entegrasyon ve kabul testleri; tespit edilen hataların düzeltilmesi

5.4.5 6. Hafta – Yayınlama & Sunum
Üretim ortamına geçiş, dokümantasyon, demo ve sunum hazırlığı

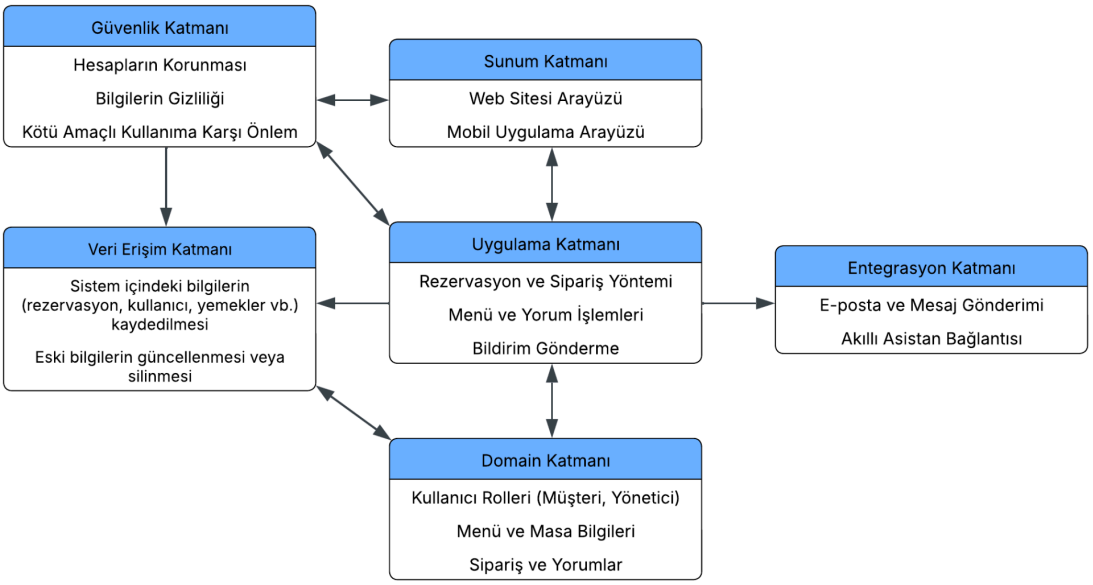
Mimari Seçimi ve Gerekçelendirilmesi

Bu projede, **katmanlı mimari (Layered Architecture)** yapısı seçilmiştir.

Katmanlı mimari, Dijital Restoran Rezervasyon ve Menü Sistemi'nin daha sürdürülebilir olmasını sağlar. Bu mimaride, sunum katmanı yalnızca kullanıcı arayüzünü ve form doğrulamalarını yönetir. İş mantığı katmanı, rezervasyon çakışması, puan hesaplama ve kullanıcı yetkilendirmesi gibi işlemleri yönetir. Veri erişim katmanı ise veritabanı işlemleri (ekleme, güncelleme, silme) ve ORM entegrasyonundan sorumludur.

Bu şekilde, her katman birbirinden ayrılmış olur ve eğer veritabanı değişirse, sadece veri katmanında değişiklik yapılır. Yeni özellik eklerken sadece iş mantığı katmanına odaklanılabilir. Ayrıca, bu yapı birim testlerinin hızlı çalışmasını sağlar, ekiplerin aynı anda farklı görevlerde çalışabilmesini sağlar ve güvenlik ile performans optimizasyonlarını belirli katmanlarda yapmayı mümkün kılar.

Sonuç olarak, bu mimari sayesinde kullanıcı arayüzü veya altyapı değişiklikleri, diğer katmanları etkilemeden yapılabilir. Bu da sistemin bakımını kolaylaştırır, daha hızlı büyümesini sağlar ve gelecekte yeni modüllerin eklenmesini daha kolay hale getirir.



Kullanım Senaryo Örnekleri

Senaryo 1: Müşterinin restoran rezervasyonu yapması

1. Kullanıcı, sisteme e-posta ve şifresiyle giriş yapar.
2. Ana sayfada yer alan filtreleme butonuyla puanlama sırasına göre restoranları listeler.
3. Listeyi inceledikten sonra beğendiği restoranın detay sayfasına girerek boşluk bulunan tarihlere göre uygun tarih ve saat seçer.
4. Uygun masa listesinden bir masa ve kişi sayısı seçer.
5. Rezervasyon Yap butonuna tıklar.
6. Rezervasyon durumu "Bekliyor" olarak veritabanına kaydedilir.
7. Kullanıcı rezervasyon durumunu profil sayfasındaki Rezervasyonlarım bölümünden görüntüleyebilir.
8. Rezervasyon saatine yaklaştığında sistem, kullanıcıya e-posta, SMS veya uygulama içi bildirim gönderir.

Senaryo 2: Müşterinin Yorum Yapması

1. Kullanıcı, sisteme e-posta ve şifresiyle giriş yapar.
2. Profil sayfasındaki "Geçmiş Siparişlerim" bölümüne girer.
3. Daha önce sipariş verdiği bir restoran veya yemeği seçer.
4. Yemeğin detay sayfasında yer alan “Yorum Yap” butonuna tıklar.
5. Açılan yorum formuna metin olarak görüşünü yazar.
6. Aynı form üzerinden, 1 ile 5 arasında yıldız seçerek hem yemek hem de restoran için ayrı ayrı puanlama yapar.
7. “Gönder” butonuna basarak yorumunu sisteme iletir.
8. Yorum, sistem tarafından “onay bekliyor” olarak veritabanına kaydedilir.
9. Sistem yöneticisi, gönderilen yorumu inceler ve onayladıktan sonra yorum herkese açık şekilde yayınlanır.
10. Kullanıcı, yaptığı yorumu profil sayfasındaki “Yorumlarım” alanında görüntüleyebilir, dilerse düzenleyebilir veya silebilir.

Dijital Restoran Rezervasyon ve Menü Sistemi

İçin

Yazılım Geliştirme Süreci ve Mimarisi

Versiyon 2.0

230229034 Betül Eren
230229038 Irmak Yılmaz
230229059 Duygu Ersoy
230229074 Zehra Özdemir
220229053 Sena Nur Dülger
tarafından hazırlanmıştır.

Kocaeli Üniversitesi
Yazılım Mühendisliği

04 / 06 / 2025

İçindekiler

İçindekiler

1. Detaylı Yazılım Mimarisi

1.1 Mimari Yaklaşımın Tanıtımı
1.2 Katmanların Ayrıntılı Açıklamaları
1.3 Katmanları Arası Etkileşim

2. Kullanım Senaryoları ve Akış Diyagramları

2.1 Senaryo 1: Kullanıcının Rezervasyon Yapması
2.2 Senaryo 2: Kullanıcının Sipariş Vermesi
2.3 Senaryo 3: Kullanıcının Yorum Yapması
2.4 Senaryo 4: Yöneticinin Menü Güncellemesi

3. Tasarım Desenleri

3.1 Singleton Tasarım Deseni
3.2 Observer Tasarım Deseni
3.3 Factory (Fabrika) Tasarım Deseni
3.4 Strategy (Strateji) Tasarım Deseni
3.5 MVC (Model-View_Controller)

4. Veri Modeli ve Veritabanı Yapısı

4.1 Tablolar
4.2 İlişkiler

5. Test Planı

5.1 Amaç ve Kapsam
5.2 Test Yaklaşımı ve Strateji
5.3 Test Ortamı ve Kullanılacak Araçlar
5.4 Kullanılacak Araçlar ve Çerçeveler
5.5 Test Süreçleri ve Zamanlama
5.6 Test Senaryoları ve Örnek Matrisler
5.7 Başarı Kriterleri ve Ölçütler

6. Karşılaşılan Zorluklar ve Çözüm Yolları

6.1 Rezervasyon Çakışma ve Eşzamanlılık Sorunları
6.2 Gerçek Zamanlı Masa Durumu ve Bildirim Entegrasyonu
6.3 Performans ve Yük Altında Yanıt Süresi
6.4 Güvenlik ve Yetkilendirme Zorlukları

7. Sonuç ve Tartışma

7.1 Projenin Genel Değerlendirmesi
7.2 Güçlü Yönler
7.3 Zayıf Yönler
7.4 Geliştirme Önerileri
7.5 Geleceğe Yönelik Perspektif

8. Risk Analizi

9. Örnek Fonksiyonlar ve Kod Parçacıkları

9.1 Rezervasyon Servisi
9.2 Sipariş Servisi
9.3 Menü Servisi
9.4 Yorum Servisi
9.5 Bildirim Servisi

10. Kaynakça

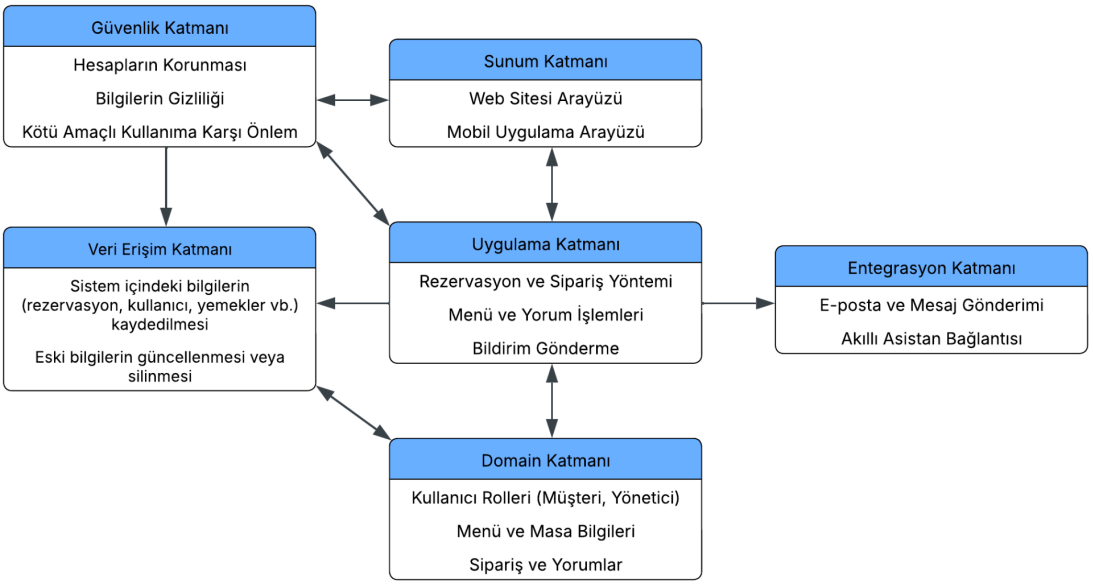
1. Detaylı Yazılım Mimarisi Tasarımı

1.1 Mimari Yaklaşımın Tanıtımı

Dijital Restoran Rezervasyon ve Menü Sistemi'nde hem kullanıcı deneyimini sorunsuz hale getirecek şekilde hızlı yanıt verebilmesi, hem de yönetim tarafında güvenli, esnek ve sürdürülebilir bir yapıda olması hedeflenmiştir. Bu bağlamda, proje geliştirme sürecinde katmanlı mimari (layered architecture) yapısı tercih edilmiştir.

Katmanlı mimari, uygulamanın farklı sorumluluklara sahip bileşenlerini mantıksal olarak ayırarak, karmaşıklığı yönetilebilir seviyeye indirger. Her bir katman yalnızca kendi görevinden sorumlu olacak şekilde yapılandırılmıştır ve bu katmanlar arasında sıkı bir yönlendirilmiş veri akışı mevcuttur. Bu yapı sayesinde:

- Geliştirme ekipleri paralel olarak farklı katmanlar üzerinde çalışabilir,
- Yeni özelliklerin eklenmesi mevcut yapıyı bozmadan gerçekleştirilebilir,
- Birim testleri izole olarak yürütülebilir,
- Güvenlik açıkları daha kontrollü şekilde sınırlandırılabilir.



1.2 Katmanların Ayrıntılı Açıklamaları

1.2.1 Sunum Katmanı (Presentation Layer)

Bu katman, son kullanıcı ile sistem arasında doğrudan iletişimin sağlandığı bölümdür. Hem masaüstü tarayıcılar hem de mobil cihazlar üzerinden erişilebilen kullanıcı arayüzlerini içerir. Kullanıcı tarafından gerçekleştirilen tüm işlemler (örneğin rezervasyon oluşturma, menü görüntüleme, yorum yapma) bu katmanda başlatılır ve alt katmanlara iletilir.

Başlıca Sorumluluklar:

- Giriş/kayıt ekranları
- Restoran ve menü listeleme sayfaları
- Sipariş ve rezervasyon formları
- Geri bildirim alanları (yorum ve puanlama)

Kullanılan Teknolojiler:

- React.js veya Vue.js (SPA yaklaşımı ile hızlı yükleme ve gezinme)
- HTML5, CSS3, JavaScript
- Responsive tasarım (Bootstrap veya TailwindCSS)

1.2.2 Uygulama Katmanı (Application Layer)

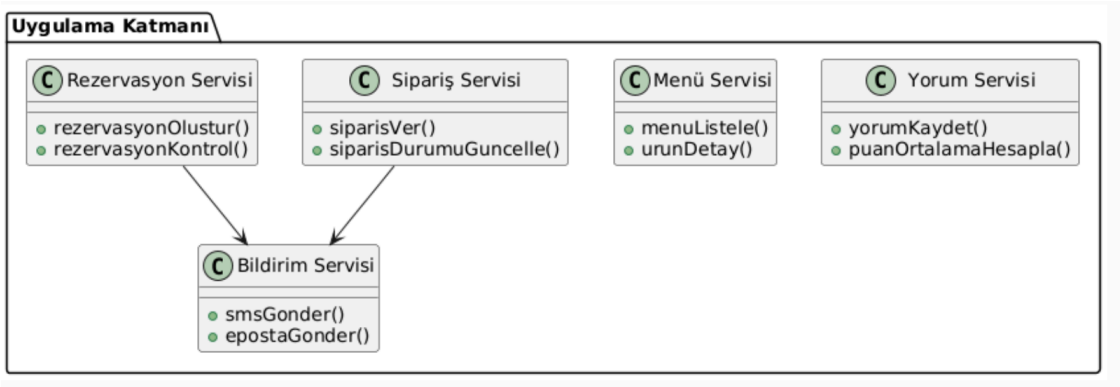
Uygulama katmanı, sistemin iş kurallarını barındıran merkezi yöneticidir. Sunum katmanından gelen tüm işlem istekleri burada işlenir ve uygun domain modellerine ya da veri erişim katmanına yönlendirilir. Bu katman aynı zamanda sistem içi işlemlerin sırasını, geçerlilik durumlarını ve hata yönetimini de üstlenir.

İşlev Örnekleri:

- Rezervasyon çakışmalarının tespiti
- Sipariş geçerlilik kontrolleri
- Kullanıcılara özel önerilerin sunulması (akıllı asistan mantığı)
- Yorumların filtrelenmesi ve sıralanması

Kullanılan Teknolojiler:

- .NET Core / Spring Boot servis mimarisi
- REST API mimarisi üzerinden HTTP üzerinden JSON veri alışverişi
- Arka planda çalışan servisler: Bildirim, zamanlayıcı (rezervasyon hatırlatma gibi)



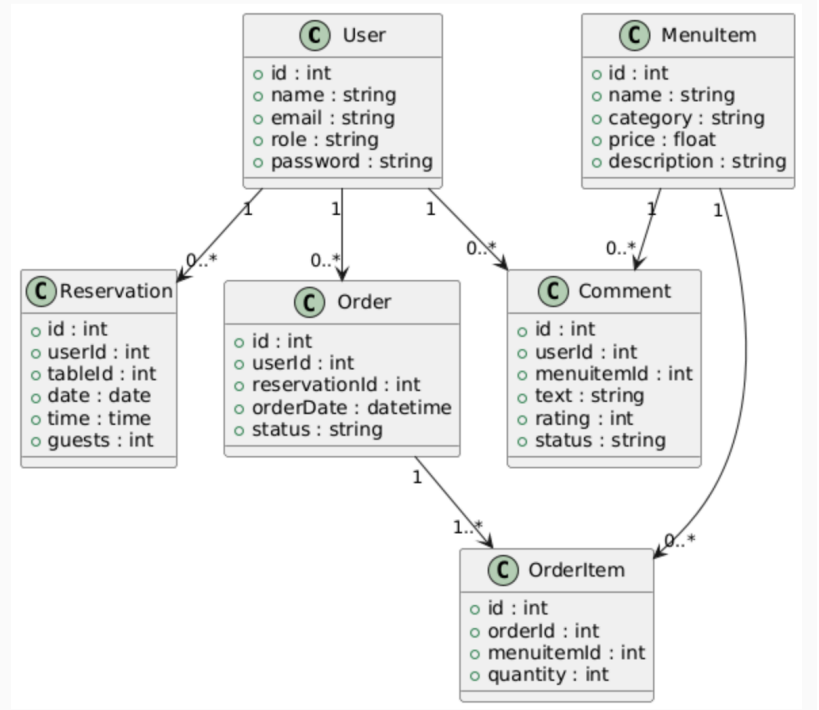
1.2.3 Domain Katmanı (Domain Layer)

Domain katmanı, uygulamanın iş alanına ait kavramların (Entity ve Value Object'ler) tanımlandığı, sistemin "ne yaptığı" ile ilgilenen merkezidir. Rezervasyon, Sipariş, Menü, Yorum gibi temel kavramlar bu katmanda modellenir. Her bir domain nesnesi, kendi iş kurallarını (örneğin minimum rezervasyon süresi, yorum karakter sınırı) de içerebilir.

Nesne Örnekleri:

- User: Rol bilgisi (Müşteri, Yönetici), iletişim detayları
- Reservation: Tarih, saat, masa bilgisi
- Order: Sipariş durumu, ödeme şekli

- Comment: Metin, puan, onay durumu



1.2.4 Veri Erişim Katmanı (Data Access Layer)

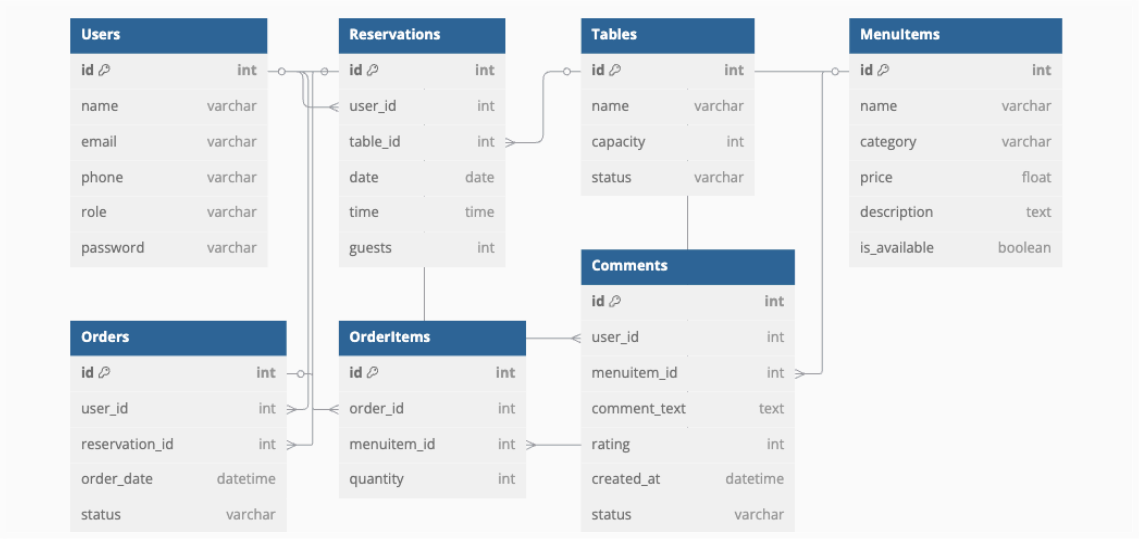
Veri Erişim Katmanı, sistemin veritabanı ile olan tüm iletişimini yöneten bölümdür. Uygulama katmanından gelen istekler doğrultusunda SQL sorguları çalıştırılır, veriler çekilir veya güncellenir. ORM teknolojisi ile Entity sınıfları ile veritabanı arasında eşleme yapılır.

Veri Tabloları:

- Users, Reservations, Tables, MenuItems, Orders, OrderItems, Comments

Teknolojiler:

- Entity Framework Core
- PostgreSQL (yüksek veri tutarlılığı ve performans avantajı nedeniyle)
- LINQ sorguları ile dinamik filtreleme

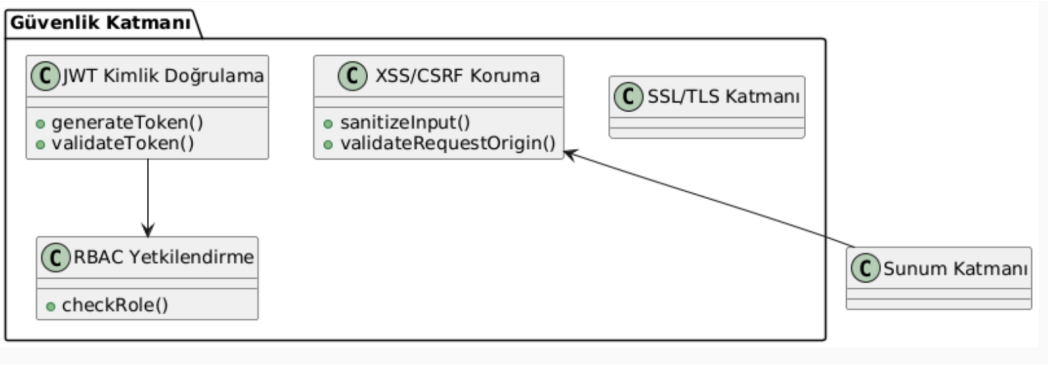


1.2.5 Güvenlik Katmanı (Security Layer)

Kullanıcı verilerinin gizliliği ve sistemin kötü amaçlı kullanıma karşı korunması için bağımsız bir güvenlik katmanı tasarlanmıştır. Tüm oturum yönetimi, kimlik doğrulama ve yetkilendirme işlemleri burada gerçekleştirilir.

Özellikler:

- JWT (JSON Web Token) ile oturum bazlı kimlik doğrulama
- Rol tabanlı erişim kontrolü (RBAC)
- HTTPS / TLS şifreleme protokolü
- 15 dakikalık pasiflik sonrası oturum sonlandırma
- CSRF ve XSS saldırılarına karşı içerik filtreleme
- AES-256 ile şifrelenmiş kullanıcı verisi

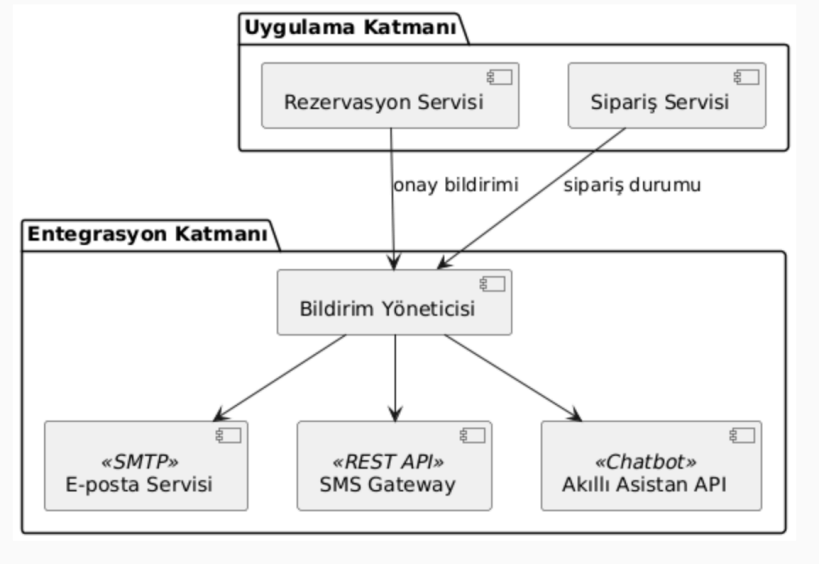


1.2.6 Entegrasyon Katmanı (Integration Layer)

Sistem dışı kaynaklarla etkileşim, entegrasyon katmanı üzerinden gerçekleştirilir. E-posta gönderimi, SMS hatırlatmaları ve yapay zeka destekli akıllı asistanla iletişim gibi işlemler bu katmanda konumlanır.

Servisler:

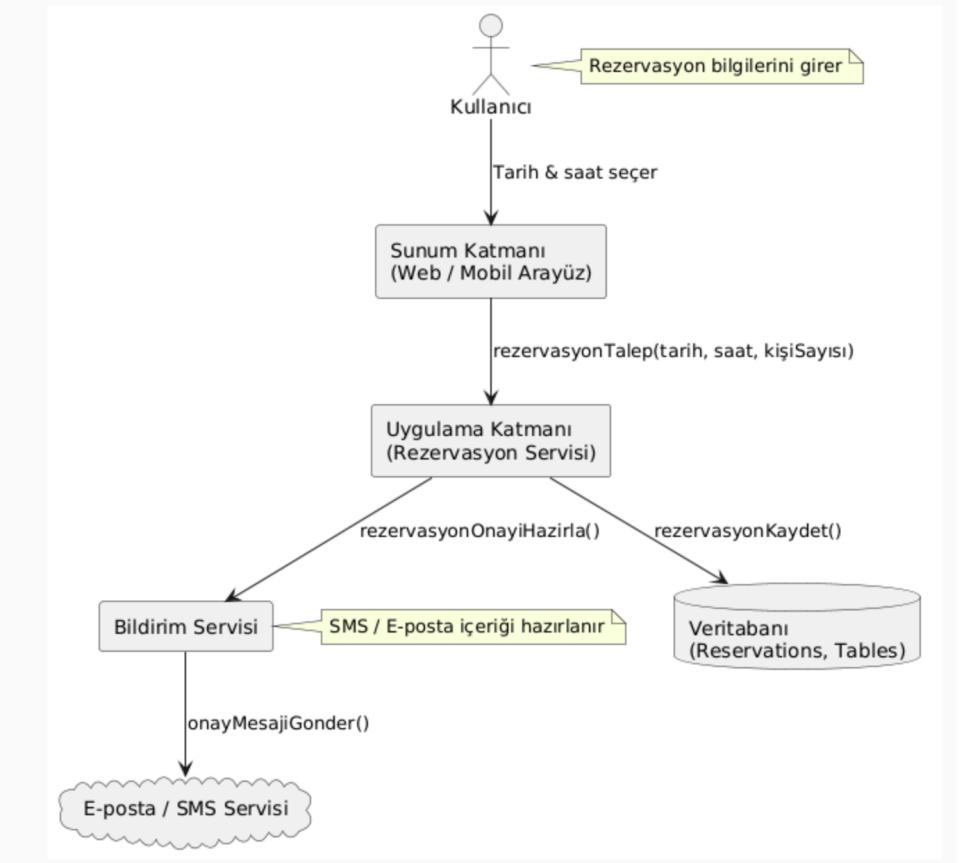
- SMTP veya üçüncü parti servislerle e-posta gönderimi (SendGrid, Gmail API)
- Firebase Push Notification
- Akıllı asistan ile RESTful API üzerinden veri alışverişi
- JSON formatında dış servislerle haberleşme



1.3 Katmanlar Arası Etkileşim

Sistemin genel veri akışı, aşağıdaki şekilde ilerler:

1. Kullanıcı, Sunum Katmanı üzerinden bir işlem başlatır (örneğin rezervasyon).
2. Bu işlem, Uygulama Katmanı üzerinden iş mantığına göre değerlendirilir.
3. Gerekli doğrulamalar sonrası, Domain Katmanı ilgili nesneleri oluşturur.
4. Veri saklanacaksa, Veri Erişim Katmanı kullanılarak kalıcı hale getirilir.
5. Eşzamanlı olarak, Entegrasyon Katmanı üzerinden kullanıcıya bildirim gider.
6. Tüm süreç boyunca Güvenlik Katmanı aktif olarak işlem denetimi yapar.



2. Kullanım Senaryoları ve Akış Diyagramları

2.1 Senaryo 1: Kullanıcının Rezervasyon Yapması

Senaryo Adı: Rezervasyon Oluşturma

Amaç: Kullanıcının bir restoranda uygun tarih ve saate rezervasyon yapabilmesi

Aktörler:

- Birincil: Kullanıcı (Müşteri)
- İkincil: Sistem, Rezervasyon Servisi, Bildirim Servisi

Ön Koşullar:

- Kullanıcı sisteme giriş yapmış olmalıdır.
- Kullanıcı, rezervasyon yapılabilecek bir restoran seçmiş olmalıdır.

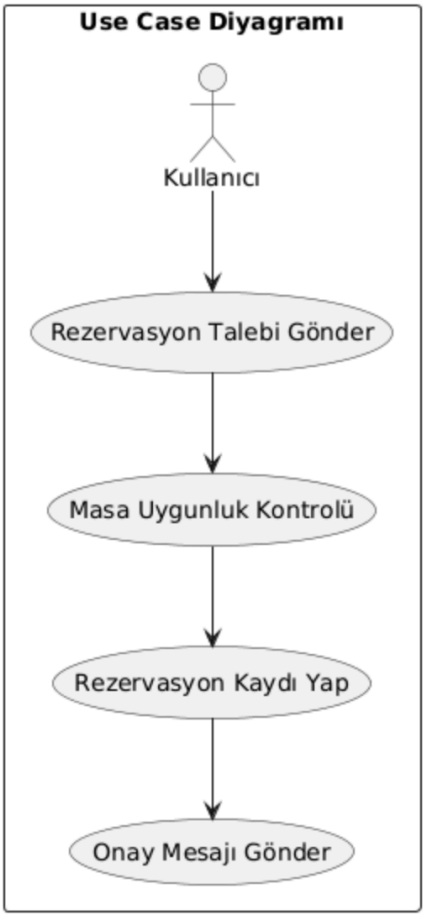
Temel Akış:

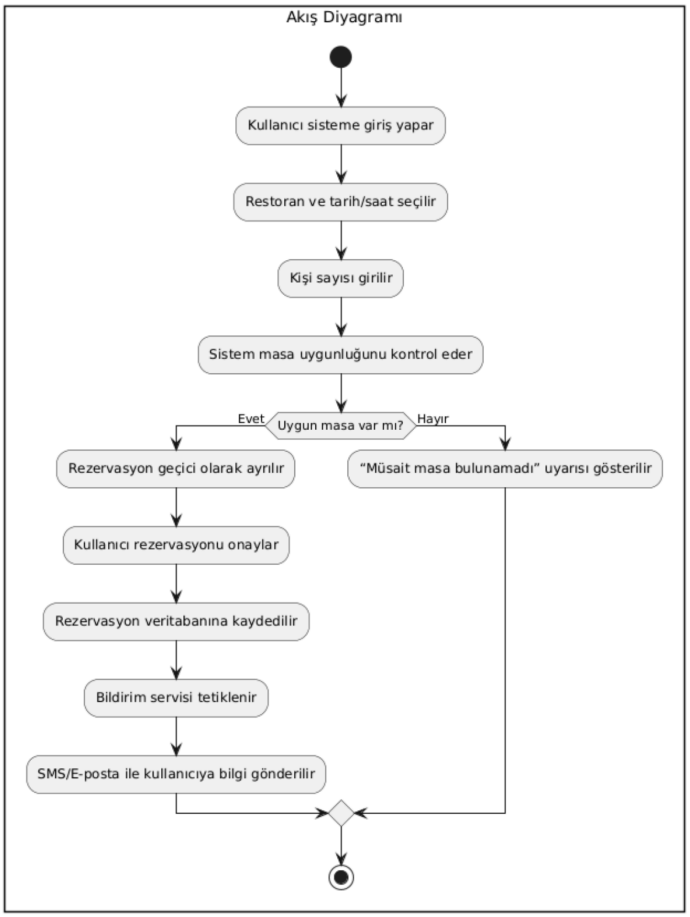
1. Kullanıcı ana sayfadan restoranı seçer.
2. Sistem, seçilen restoranın masa ve takvim bilgilerini yükler.
3. Kullanıcı tarih, saat ve kişi sayısı belirler.
4. Sistem, belirtilen tarihte masa uygunluğu olup olmadığını kontrol eder.
5. Eğer uygun masa varsa, sistem rezervasyonu geçici olarak rezerve eder.
6. Kullanıcı “Rezervasyonu Tamamla” butonuna basar.
7. Rezervasyon, veritabanına kaydedilir ve durumu "Bekliyor" olur.
8. Bildirim Servisi devreye girer ve kullanıcıya SMS/e-posta ile onay mesajı gönderir.

Alternatif Akışlar:

- 4A. Eğer uygun masa yoksa, sistem “Boş Masa Bulunamadı” mesajı gösterir.
- 6A. Kullanıcı işlemi yarıda bırakırsa, rezervasyon kaydı oluşturulmaz.

Sonuç: Kullanıcının belirlediği tarihte bir masa rezerve edilir ve sistem tarafından onay bildirimleri gönderilir.





2.2 Senaryo 2: Kullanıcının Sipariş Vermesi

Senaryo Adı: Online Sipariş Süreci

Amaç: Kullanıcının menüden ürün seçerek online sipariş oluşturması

Aktörler:

- Birincil: Kullanıcı
- İkincil: Sipariş Servisi, Veritabanı, Bildirim Servisi

Ön Koşullar:

- Kullanıcı giriş yapmış olmalıdır.
- Siparişe uygun menü öğeleri sistemde kayıtlı ve erişilebilir olmalıdır.

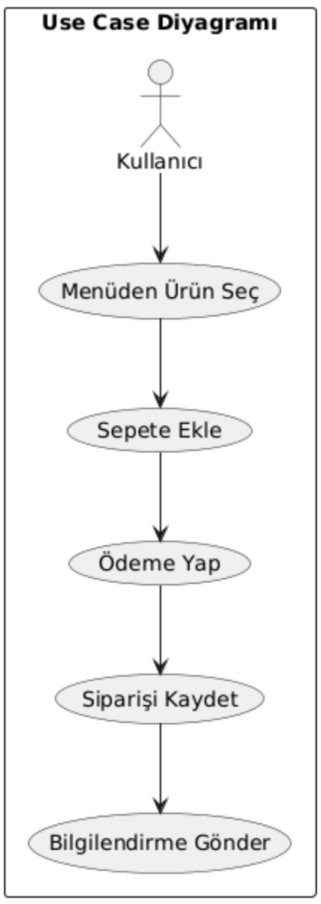
Temel Akış:

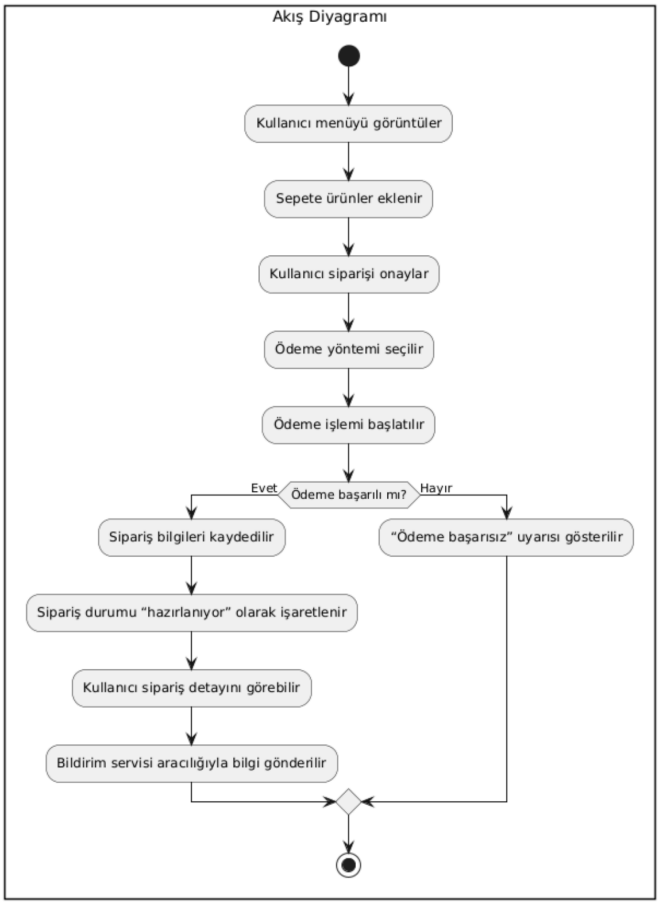
1. Kullanıcı menüyü görüntüler.
2. Ürünleri sepete ekler.
3. "Siparişi Tamamla" butonuna tıklar.
4. Kullanıcı, ödeme yöntemi seçer ve onay verir.
5. Sistem ödeme bilgilerini doğrular.
6. Sipariş "hazırlanıyor" durumu ile veritabanına kaydedilir.
7. Kullanıcının sipariş geçmişine işlenir.
8. Bildirim Servisi üzerinden sistem, sipariş özeti ve durumu hakkında kullanıcıya mesaj gönderir.

Alternatif Akışlar:

- 5A. Ödeme başarısız olursa, sistem hata bildirimi gösterir ve sipariş kaydedilmez.
- 4A. Kullanıcı işlemden vazgeçerse sistem herhangi bir kayıt oluşturmaz.

Sonuç: Sipariş oluşturulur, sistemde kaydedilir ve kullanıcı bilgilendirilir.





2.3 Senaryo 3: Kullanıcının Ürün Yorumlaması

Senaryo Adı: Ürün Değerlendirme
Amaç: Kullanıcının daha önce sipariş ettiği bir ürün için yorum ve puan bırakması

Aktörler:

- Birincil: Kullanıcı
- İkincil: Yorum Servisi, Yönetici

Ön Koşullar:

- Kullanıcı ilgili ürünü daha önce sipariş etmiş olmalıdır.
- Yorum yapılacak ürün sistemde hâlâ mevcut olmalıdır.

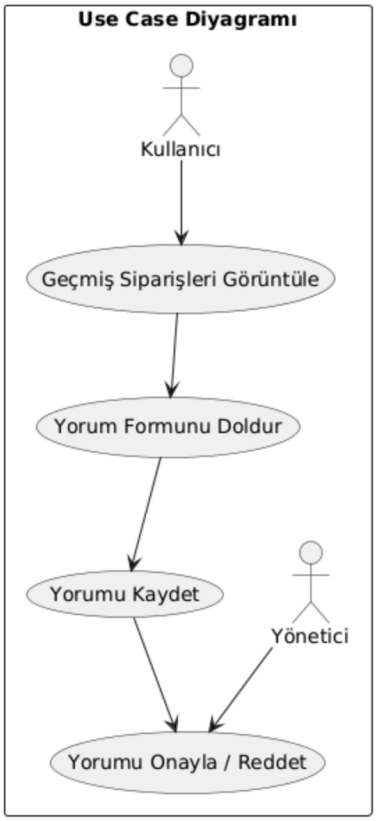
Temel Akış:

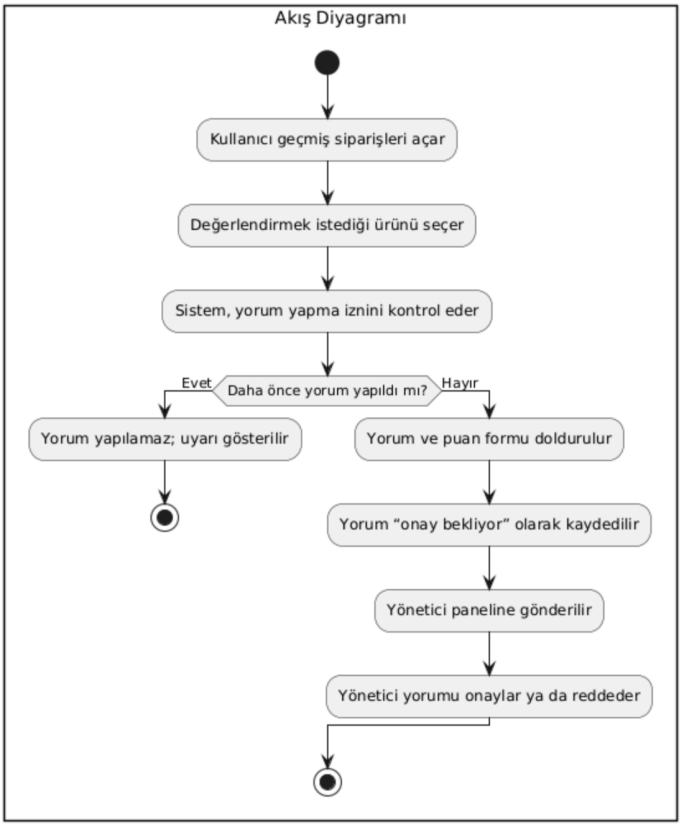
1. Kullanıcı, "Geçmiş Siparişler" ekranına erişir.
2. Sipariş içinden bir ürün seçer.
3. Sistem, daha önce yorum yapılmışsa uyarı gösterir.
4. Kullanıcı yorum metni ve puan girerek formu gönderir.
5. Yorum, "onay bekliyor" durumu ile veritabanına kaydedilir.
6. Yönetici panelinde bu yorum görünür.
7. Yönetici yorumları inceleyerek onaylar veya reddeder.

Alternatif Akışlar:

- 3A. Kullanıcı daha önce aynı ürüne yorum yaptıysa sistem yeni yorum izni vermez.
- 7A. Yönetici uygunsuz içerik tespit ederse yorumu siler.

Sonuç: Yorum başarıyla oluşturulur ve moderasyon sürecinden sonra yayımlanır.





2.4 Senaryo 4: Yöneticinin Menü Güncellemesi

Senaryo Adı: Menü Güncelleme
Amaç: Yöneticinin restoran menüsündeki ürünleri sistem üzerinden düzenlemesi, eklemesi veya silmesi

Aktörler:

- Birincil: Yönetici
- İkincil: Menü Servisi, Sistem Arayüzü, Veritabanı

Ön Koşullar:

- Yönetici sisteme giriş yapmış olmalıdır
- Menüde işlem yapılacak ürün veya kategori sistemde tanımlı olmalıdır

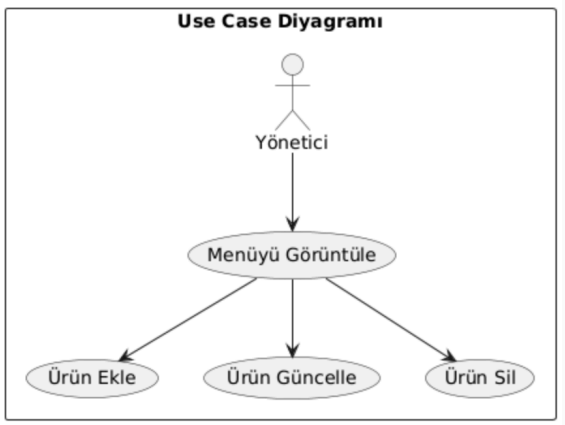
Temel Akış:

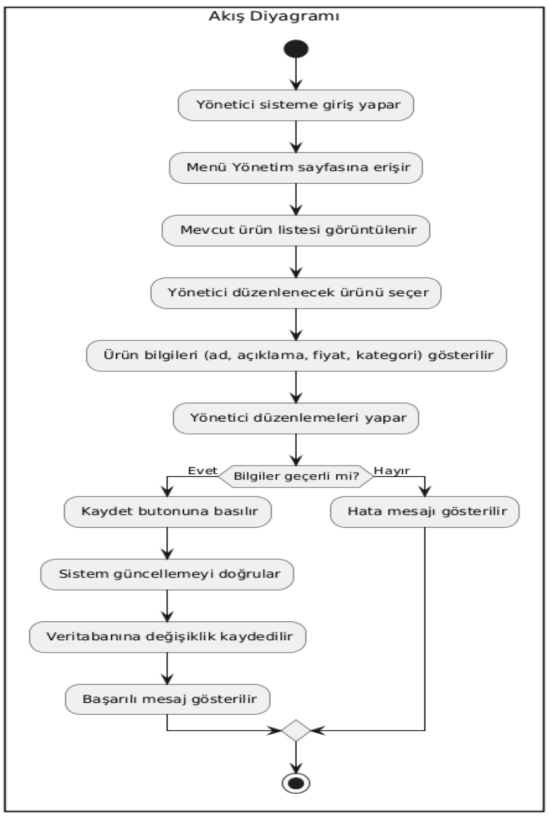
1. Yönetici sistemde "Menü Yönetimi" bölümünü açar.
2. Mevcut menü listesi ekranda gösterilir.
3. Yönetici bir ürünü seçerek bilgilerini düzenler veya yeni ürün ekler.
4. Kaydet butonuna tıklayarak değişiklikleri sisteme gönderir.
5. Sistem verileri kontrol eder ve güncellemeyi veritabanına kaydeder.
6. Güncelleme başarılıysa sistem bilgi mesajı gösterir.

Alternatif Akışlar:

- 3A. Yönetici işlemi iptal ederse hiçbir değişiklik yapılmaz.
- 5A. Ürün bilgileri eksikse sistem hata mesajı gösterir.

Sonuç: Menüdeki ürün başarıyla güncellenmiş, eklenmiş veya silinmiş olur. Sistem arayüzü yeni verilerle anlık olarak yenilenir.





3. Tasarım Desenleri

3.1 Singleton Tasarım Deseni

3.1.1 Tanımı ve Amacı

Singleton tasarım deseni, bir sınıfın yalnızca bir örneğinin (instance) oluşturulmasını ve bu örneğe tüm sistem boyunca global bir erişim sağlanmasını garanti altına alır. Genellikle sistemde merkezi kontrol noktaları için tercih edilir. Yapılandırma yöneticisi, oturum yöneticisi, log servisi gibi bileşenlerde kullanılarak, tutarlılık, kaynak optimizasyonu ve kontrol kolaylığı sağlar.

Bu projede Singleton deseni; aynı anda birden fazla bildirim, veritabanı bağlantısı veya konfigürasyon nesnesi oluşmasını önleyerek performans ve güvenilirlik açısından avantaj sağlamaktadır. Ayrıca, yazılımın sürdürülebilirliğini artırmakta ve genişletilebilir bir yapı sunmaktadır.

3.1.2 Projedeki Kullanım Alanları

3.1.2.1 Bildirim Yöneticisi (NotificationManager)

3.1.2.1.1 E-posta ve SMS bildirimlerini göndermek için kullanılır.
3.1.2.1.2 Her rezervasyon, iptal, sipariş gibi olayda kullanıcıya bildirim gitmesi gerekir.
3.1.2.1.3 Bildirimlerin karışmaması, tutarlı biçimde gönderilmesi için tek merkezden kontrol gereklidir.

Kullanım Senaryosu: Kullanıcı rezervasyon yaptığında veya iptal ettiğinde sistem bu sınıfı çağırır. Bildirim formatı, dili ve içeriği burada yönetilir.

3.1.2.2 Veritabanı Bağlantı Yöneticisi (DatabaseConnection)

3.1.2.2.1 Tüm sistem bileşenleri veritabanı ile çalışır ama her biri ayrı ayrı bağlantı oluşturursa kaynak israfı olur.
3.1.2.2.2 Tek bir bağlantı yöneticisi örneğiyle işlemler kontrol edilir.

Kullanım Senaryosu: Rezervasyon ekleme, sipariş güncelleme veya yorum silme işlemleri bu tek örnek üzerinden yapılır.

3.1.2.3 Loglama Yöneticisi (Logger)

3.1.2.3.1 Sistem hatalarını, kullanıcı etkinliklerini, güvenlik uyarılarını tek bir yere kaydeder.
3.1.2.3.2 Farklı bileşenler bu sınıfı çağırarak log kaydı oluşturur ama sınıfın örneği tektir.

Kullanım Senaryosu: Kullanıcı girişi, hata oluşumu ya da yetkisiz erişim denemeleri bu log dosyasına yazılır.

3.1.2.4 Ayar Yöneticisi (ConfigurationManager)

3.1.2.4.1 Sistem ayarları (API anahtarı, SMTP ayarları, sistem dili, tema seçimi vb.) tek bir yerden okunur.
3.1.2.4.2 Bu bilgiler tüm sistem tarafından kullanılabilir ama merkezi olarak yönetilmelidir.
Kullanım Senaryosu: Sistem teması, çoklu dil ayarları ya da mesajlaşma sunucusu konfigürasyonları bu sınıf üzerinden kontrol edilir.

3.1.2.5 Oturum Yöneticisi (SessionManager)

3.1.2.5.1 Kullanıcıların oturum bilgileri (login durumu, rol bilgisi, oturum süresi) burada saklanır.

3.1.2.5.2 Tek bir oturum yöneticisi ile tüm oturumlar takip edilir.

Kullanım Senaryosu: Kullanıcı sisteme giriş yaptığında bu sınıfa oturum açılır, çıkış yaptığında buradan sonlandırılır.

3.1.2.6 Yapay Zeka Tabanlı Chatbot Motoru (ChatbotEngine)

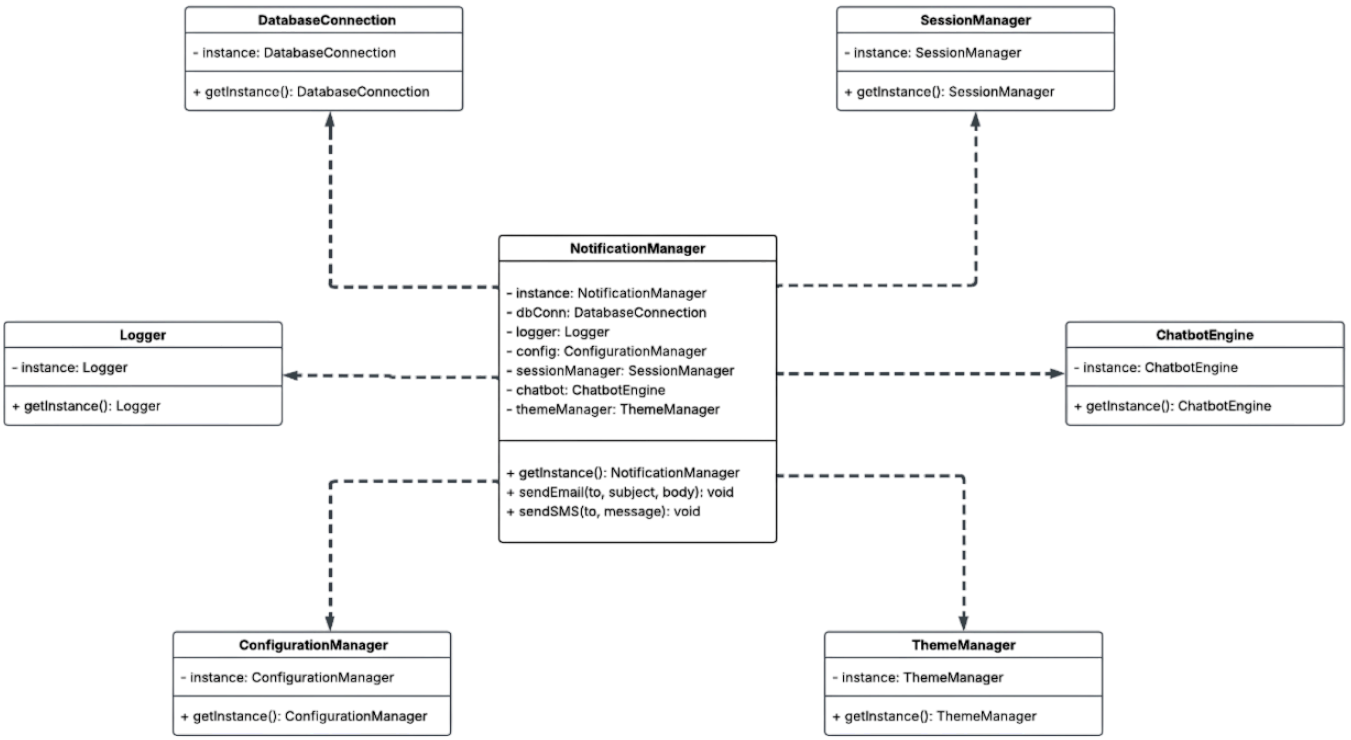
3.1.2.6.1 AI destekli akıllı asistan modülü sistemin her yerinde çağrılabilir ama tek motor örneği üzerinden çalışmalıdır.

Kullanım Senaryosu: Kullanıcı chatbot'tan menü sorduğunda veya rezervasyon sorguladığında hep aynı AI motoru kullanılır.

3.1.2.7 Tema Yöneticisi (ThemeManager)

3.1.2.7.1 Açık/Koyu tema seçimi gibi kullanıcı arabirimi ayarları tek bir yapıdan yönetilmelidir.

Kullanım Senaryosu: Kullanıcı uygulama temasını değiştirdiğinde, sistem tüm bileşenlere bu ayarı iletir.



3.2 Observer Tasarım Deseni

3.2.1 Tanımı ve Amacı

Observer (Gözlemci) Tasarım Deseni, bir nesnede (Subject) meydana gelen değişikliklerin, o nesneye bağlı olan diğer nesnelere (Observer'lara) otomatik olarak bildirilmesini sağlayan bir yapıdır. Yayıncı (Subject) bir olay gerçekleştirdiğinde, bu olayı izleyen tüm gözlemciler (Observer) anında haberdar edilir. Bu sayede bileşenler birbirine sıkı bağlı olmadan haberleşebilir ve sistem daha esnek, sürdürülebilir ve genişletilebilir hale gelir.

Projemiz kapsamında Observer deseni, kullanıcıya yönelik anlık bilgilendirmelerin etkin biçimde yönetilmesi amacıyla kullanılmıştır. Sistemde aşağıdaki olaylar gerçekleştiğinde, ilgili kullanıcıların çeşitli kanallardan bilgilendirilmesi hedeflenmiştir:

- Rezervasyon onayı gönderilmesi,
- Sipariş durumundaki değişikliklerin bildirilmesi,
- Kullanıcı yorumlarına gelen yanıtların iletilmesi

Bu tür olaylar gerçekleştiğinde, sistemde birden fazla işlem aynı anda tetiklenmelidir (örneğin e-posta gönderimi, SMS bilgilendirmesi, log kaydı vb.). Observer deseni bu ihtiyacı karşılayarak, sistem bileşenlerinin birbirini doğrudan tanımadan iletişim kurmasını ve olaylara eşzamanlı tepki vermesini sağlar.

Bu bağlamda:

ReservationSystem, OrderSystem, CommentSystem gibi bileşenler Subject rolünü üstlenir,
EmailService, SMSService, Logger gibi bileşenler ise Observer olarak bu sistemlere abone olur.
Bu yapı sayesinde sistemde meydana gelen her önemli değişiklik, bağlı tüm servisler tarafından eşzamanlı olarak algılanır ve gerekli aksiyonlar otomatik şekilde alınır.

3.2.2 Projedeki Kullanım Alanları

3.2.2.1 *Bildirim Sistemi*

3.2.2.1.1 ReservationSystem, OrderSystem gibi sınıflar bir Subject olarak davranır.

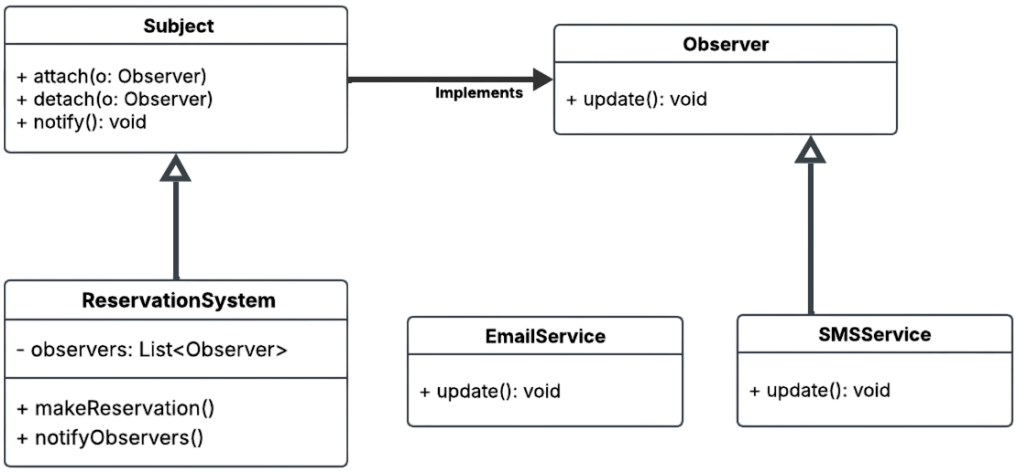
3.2.2.1.2 EmailService, SMSService, Logger gibi sınıflar Observer olarak kaydolur.

3.2.2.2 *Yorum Takip Sistemi*

3.2.2.2.1 Kullanıcının yaptığı yorumlara gelen yanıtlar sistem tarafından izlenir. Eğer biri kullanıcı yorumuna cevap verirse, sistem ilgili kullanıcıya e-posta/SMS gönderir.

3.2.2.3 *UI Bileşen Güncellemeleri (Gelişmiş UI'de)*

3.2.2.3.1 Kullanıcı arayüzü, veri güncellendiğinde otomatik olarak yenilenebilir.



3.3 Factory (Fabrika) Tasarım Deseni

3.3.1 Tanımı ve Amacı

Factory (Fabrika) Tasarım Deseni, nesne oluşturma sürecini soyutlayarak, istemcinin hangi sınıfın örneğini oluşturduğunu bilmesine gerek kalmadan, doğru nesnenin oluşturulmasını sağlar. Yani, istemci doğrudan new anahtar kelimesiyle nesne yaratmak yerine, bir factory metodu aracılığıyla ihtiyacı olan nesneyi elde eder. Bu yapı sayesinde sistem daha esnek, yeniden kullanılabilir ve genişletilebilir hale gelir.

Bu proje kapsamında; kullanıcı türleri, menü elemanları ve ödeme yöntemleri gibi farklı türlerde nesneler oluşturulmaktadır. Bu tür durumlarda Factory deseni kullanılarak, nesnelerin merkezi bir üretici sınıf aracılığıyla yaratılması sağlanmıştır.

Factory deseninin projedeki uygulama alanları şunlardır:

- *Kullanıcı türleri (Admin, Müşteri):* Sisteme giriş yapan kişinin rolüne göre ilgili kullanıcı nesnesi oluşturulur.
- *Menü elemanları (Yemek, İçecek, Tatlı):* Sipariş verilen ürüne göre uygun MenuItem alt sınıfı üretilir.
- *Ödeme yöntemleri (Kredi Kartı, Nakit, QR Kod):* Kullanıcının seçtiği ödeme türüne uygun nesne, factory yöntemi ile oluşturulur.

Bu yaklaşım sayesinde:

- Kod tekrarları azaltılmış,
- Yeni türlerin sisteme eklenmesi kolaylaştırılmış,
- -if-else veya switch bloklarının karmaşasından kaçınılmıştır.

Factory deseni, projenin hem genişletilebilirliğini hem de bakım kolaylığını önemli ölçüde artırmaktadır.

3.3.2 Projedeki Kullanım Alanları

3.3.2.1 Menü Elemanı Üretimi

3.3.2.1.1 Sistem kullanıcıya ait siparişin içeriğine göre bir MenuItem üretir.

3.3.2.1.2 Factory sınıfı, kullanıcı girişine göre Food, Drink, Dessert nesnelerinden uygun olanı döner.

3.3.2.2 Kullanıcı Rolü Oluşturma

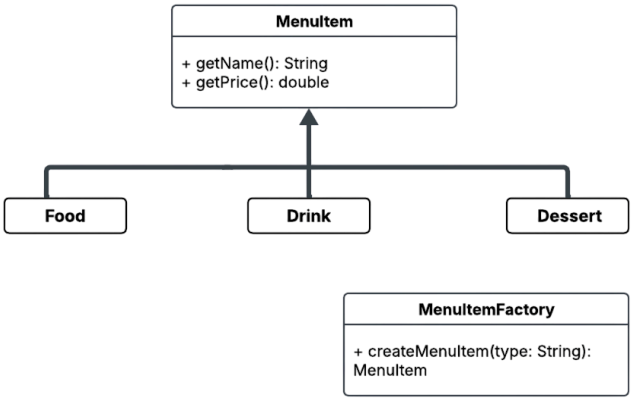
3.3.2.2.1 Giriş yapan kullanıcının rolüne göre (Admin veya Customer) sistem doğru sınıfın nesnesini oluşturur.

3.3.2.2.2 Böylece rol bazlı işlemler soyutlanmış olur.

3.3.2.3 Ödeme Yöntemleri

3.3.2.3.1 Kullanıcı ödeme ekranında bir yöntem seçtiğinde, sistem o tipe uygun ödeme sınıfını üretir (örneğin CreditCardPayment, CashPayment).

3.3.2.3.2 Tüm ödeme sınıfları aynı arayüzü uygular ve sistem bu soyut yapı üzerinden işlem yapar.



3.4 Strategy (Strateji) Tasarım Deseni

3.4.1 Tanımı ve Amacı

Strategy Design Pattern, birbirinin yerine kullanılabilen algoritmaların veya işlemlerin bir arayüz (interface) üzerinden tanımlanmasını sağlayan davranışsal bir tasarım desenidir. Bu desen sayesinde bir işlemin nasıl yapılacağına dair stratejiler, birbirinden bağımsız olarak geliştirilebilir ve çalıştırma zamanında (runtime) kolayca değiştirilebilir.

Strategy deseni şu avantajları sunar:

- Algoritmaların birbirinden ayrılmasını sağlar,
- Kod tekrarını önler,
- Yeni stratejilerin sisteme entegre edilmesini kolaylaştırır,
- Yazılımın açık/kapalı prensibine (Open/Closed Principle) uygun geliştirilmesini destekler.

Bu projede, kullanıcılar restoran üzerinden farklı ödeme yöntemleri ile işlem gerçekleştirebilmektedir: kredi kartı, nakit, QR Kod / Dijital Cüzdan.

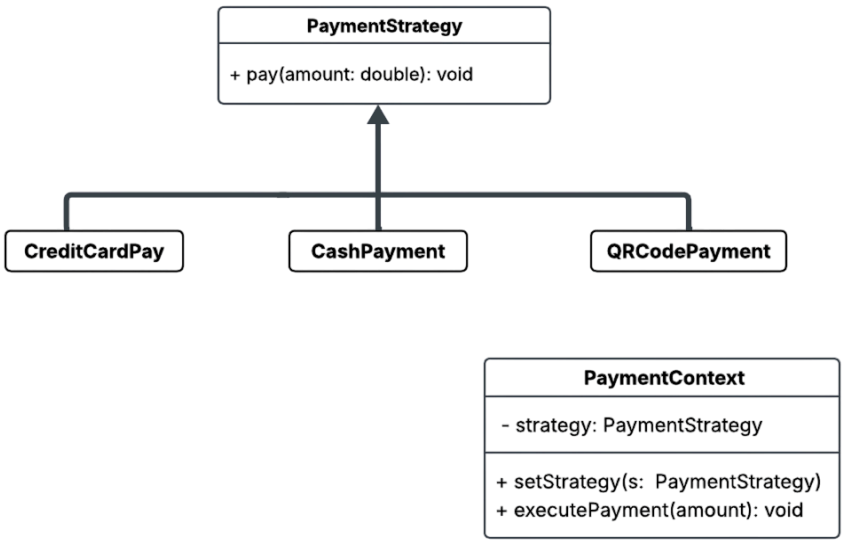
Bu ödeme türlerinin her biri, farklı işlem adımlarına, doğrulama kontrollerine ve akışlara sahiptir. Bu nedenle her ödeme türü, ortak bir PaymentStrategy arayüzü üzerinden tanımlanmış ve Strategy deseni kullanılarak uygulanmıştır. Kullanıcının ödeme tercihi, çalışma zamanında uygun stratejinin seçilmesini sağlar ve böylece sistem esnek bir şekilde genişletilebilir hale gelir.

3.4.2 Projedeki Kullanım Alanları

3.4.2.1 Ödeme Stratejileri

3.4.2.1.1 Kullanıcı ödeme sayfasında bir yöntem seçtiğinde (CreditCardPayment, CashPayment, QRCodePayment), sistem bu stratejiyi kullanarak ödemeyi gerçekleştirir.

3.4.2.1.2 Tüm ödeme sınıfları aynı PaymentStrategy arayüzünü uygular.



3.5 MVC (Model-View-Controller)

3.5.1 Tanımı ve Amacı

Kullanıcı arayüzü, iş mantığı ve veri erişim işlemlerini birbirinden ayırarak sürdürülebilir, test edilebilir ve genişletilebilir bir sistem mimarisi oluşturmak.

Restoran rezervasyon ve sipariş sistemi; kullanıcı işlemleri, menü yönetimi, ödeme süreçleri gibi birden çok işlev barındırmaktadır. Bu yapıların birbirinden bağımsız olarak geliştirilebilmesi ve sistemin modüler şekilde yönetilebilmesi için MVC mimarisi tercih edilmiştir.

3.5.2 Projedeki Kullanım Alanları

Dijital restoran sisteminiz, kullanıcıların rezervasyon yapabildiği, menüleri görüntüleyebildiği ve ödeme işlemleri gerçekleştirebildiği çok katmanlı bir yapıya sahip. Bu nedenle MVC deseni, projenize mimari bir temel sağlamak için oldukça uygundur.

3.5.2.1 Model:

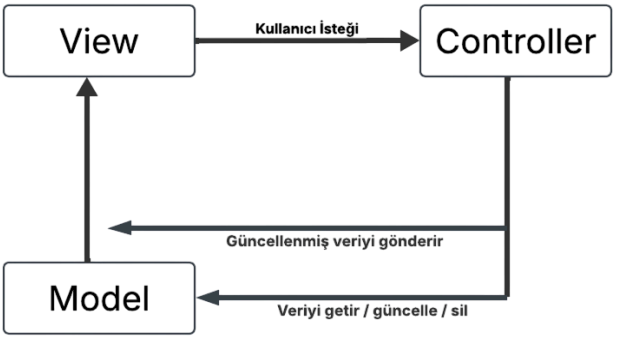
User, Reservation, MenuItem, Order, Payment gibi sınıflar model katmanında yer alır. Bu sınıflar veritabanı ile doğrudan ilişkilidir.

3.5.2.2 View:

Kullanıcının menüyü görüntülediği sayfa, rezervasyon formu, ödeme ekranı gibi arayüzler.Web arayüzü veya mobil uygulama bileşenleri View katmanını oluşturur.

3.5.2.3 Controller:

ReservationController, UserController, OrderController gibi sınıflar kullanıcıdan gelen istekleri alır. Bu istekleri işler, Model ile etkileşime girer ve uygun View'ı çağırır.



4. Veri Modeli ve Veritabanı Yapısı

4.1 Tablolar

4.1.1 Users Tablosu

4.1.1.1 id : Birincil anahtar, her kullanıcıya benzersiz kimlik atar

4.1.1.2 name : Kullanıcının tam adı

4.1.1.3 email : Kullanıcının e-posta adresi (benzersiz)

4.1.1.4 phone : Telefon numarası

4.1.1.5 role : Kullanıcı rolü (customer, admin)

4.1.1.6 password : Hash'lenmiş şifre (Güvenlik nedeniyle düz metin olarak değil, algoritma sayesinde hash değerleriyle saklanır)

4.1.2 Reservations Tablosu

4.1.2.1 id : Rezervasyon kaydının birincil anahtarı

4.1.2.2 user_id : Rezervasyonu yapan kullanıcıyı işaret eden yabancı anahtar(user_id, Users tablosundaki id alanına işaret eder)
4.1.2.3 table_id : Rezervasyonun ilişkilendirildiği masa (Tables tablosuna bağlıdır).
4.1.2.4 date : Rezervasyon tarihi
4.1.2.5 time : Rezervasyon saati
4.1.2.6 guests : Misafir sayısı
4.1.2.7 note : İsteğe bağlı kullanıcı notu veya özel talepler

4.1.3 Tables Tablosu

4.1.3.1 id : Masa kaydının birincil anahtarı
4.1.3.2 name : Masanın tanımlayıcı adı/numarası
4.1.3.3 capacity : Maksimum kişi sayısı
4.1.3.4 status :Masanın anlık durumu (available / reserved / out_of_service)

4.1.4 MenuItems Tablosu

4.1.4.1 id : Menü öğesinin birincil anahtarı
4.1.4.2 name : Ürün adı
4.1.4.3 category : Ürün kategorisi(başlangıç / ana yemek / tatlı / içecek)
4.1.4.4 price : Ürün fiyatı
4.1.4.5 description : Ürün hakkında kısa açıklama(malzeme/içerik)
4.1.4.6 is_available : Ürünün stokta olup olmadığını gösteren değer (boolean)

4.1.5 Orders Tablosu

4.1.5.1 id : Siparişin birincil anahtarı.
4.1.5.2 user_id : Siparişi veren kullanıcıyı işaret eder (Users.id).
4.1.5.3 reservation_id : (Opsiyonel) Siparişin ait olduğu rezervasyon varsa işaret eder (Reservations.id).
4.1.5.4 order_date : Siparişin verildiği zaman.
4.1.5.5 status : Sipariş durumu (pending / preparing / delivered).

4.1.6 OrderItems Tablosu

4.1.6.1 id : Sipariş öğesinin birincil anahtarı.
4.1.6.2 order_id : İlgili siparişi işaret eder (Orders.id).
4.1.6.3 menuitem_id : Sipariş edilen ürün (MenuItems.id).
4.1.6.4 quantity : Ürün miktarı.

4.1.7 Comments Tablosu

4.1.7.1 id : Yorum kaydının birincil anahtarıdır.
4.1.7.2 user_id : Yorumu yapan kullanıcıyı işaret eden yabancı anahtar (Users.id).
4.1.7.3 menuitem_id : Yorum yapılan menü öğesini işaret eden yabancı anahtar (MenuItems.id).
4.1.7.4 comment_text : Kullanıcının yazdığı yorum metni.
4.1.7.5 rating : Kullanıcının verdiği puan (1 ile 5 arasında).
4.1.7.6 created_at : Yorumun sisteme kaydedildiği tarih ve saat bilgisi.
4.1.7.7 status : Yorumun onay durumu (pending / approved / rejected).

4.2 İlişkiler

4.2.1 Users → Reservations (1:N): Bir kullanıcı birden fazla rezervasyon yapabilir ama her rezervasyon bir kullanıcıya aittir.Bu ilişki, Reservations tablosundaki user_id foreign key alanı ile sağlanır.

4.2.2 Tables → Reservations (1:N): Her masa farklı zaman dilimlerinde birden çok rezervasyonla çalışabilir.Bu ilişki table_id ile kurulur.

4.2.3 Users → Orders (1:N): Bir kullanıcı birden fazla sipariş verebilir. Orders.user_id ile sağlanır.

4.2.4 Orders → OrderItems (1:N): Bir sipariş birden fazla ürün içerebilir.

4.2.5 MenuItems → OrderItems (1:N): Bir ürün, farklı siparişlerde tekrar tekrar seçilebilir. Her OrderItem, bir MenuItem'a karşılık gelir.

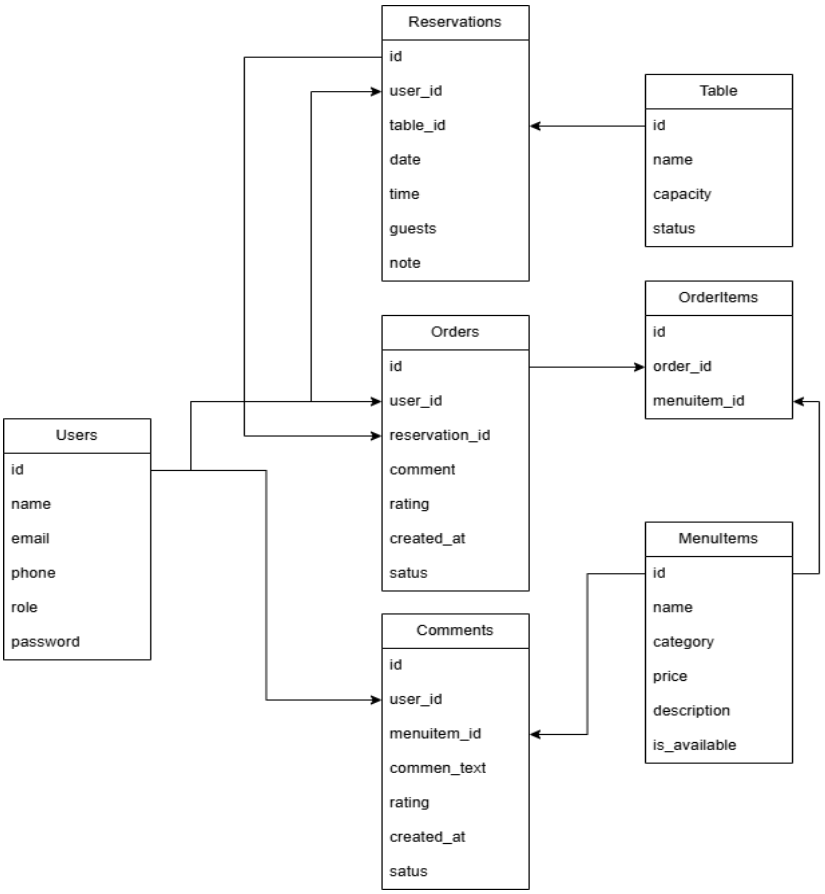
4.2.6 Reservations → Orders (Opsiyonel 1:N): Bir rezervasyona ait siparişler olabilir, ancak sipariş oluşturmak zorunlu değildir.

4.2.7 MenuItems şu anda Reservations veya Users ile doğrudan ilişkili değildir, ancak OrderItems tablosu aracılığıyla dolaylı olarak sipariş sistemiyle entegredir.

4.2.8 Users → Comments (1:N): Bir kullanıcı birçok yorum yapabilir. Comments.user_id ile sağlanır.

4.2.9 MenuItems → Comments (1:N): Her menü öğesi birçok yorum alabilir. Comments.menuitem_id ile sağlanır.

Veritabanı yapısına ilişkin ilişkiler, Şekilde gösterilen ER diyagramında görselleştirilmiştir.



5. Test Planı

5.1 Amaç ve Kapsam

5.1.1 Amaç

Bu bölümün amacı; Dijital Restoran Rezervasyon ve Menü Sistemi'nin tüm bileşenlerinin gereksinimlere uygun, güvenilir ve performans beklentilerini karşılar şekilde çalışıp çalışmadığını doğrulamaktır.

5.1.2 Kapsam

5.1.2.1 Web ve mobil taraf (responsive) kullanıcı arayüzleri

5.1.2.2 *Backend API'leri:*

Kullanıcı yönetimi (kayıt/giriş/rol bazlı yetki)

Rezervasyon (oluşturma, çakışma kontrolü, onay/iptal)

Menü görüntüleme ve sipariş

Yorum–puan sistemi (yemek ve restoran)

Bildirim servisi entegrasyonu (e-posta, SMS, uygulama içi)

5.1.2.3 Veritabanı (SQLite'dan MySQL/PostgreSQL'e geçiş senaryoları, veri bütünlüğü)

5.1.2.4 Güvenlik katmanları: JWT, RBAC, CSRF/XSS korumaları

5.1.2.5 Performans ve yük testleri

5.1.2.6 Kullanılabilirlik (usability) ve erişilebilirlik (accessibility) kriterleri

5.2 Test Yaklaşımı ve Strateji

Bu bölümde sistemin doğru, eksiksiz ve güvenli çalışmasını sağlamak üzere tanımlanan test türleri ile genel strateji yer almaktadır.

5.2.1 Test Türleri

5.2.1.1 Fonksiyonel Testler

5.2.1.1.1 Birim (unit) testleri: Her iş mantığı katmanındaki metodun doğru çıktıyı üretip üretmediği

5.2.1.1.2 Entegrasyon testleri: Backend bileşenlerinin (ör. rezervasyon servisi ile bildirim servisi) birlikte doğru çalışıp çalışmadığı

5.2.1.1.3 Sistem (end-to-end) testleri: Uçtan uca senaryoların (örn. "Müşteri rezervasyon yap → Yönetici onay → Müşteri bildirim al → Müşteri rezervasyon detayını görüntüle") doğrulanması

5.2.1.2 Performans ve Yük Testleri

5.2.1.2.1 Yük testi: Eş zamanlı kullanıcı senaryoları (rezervasyon, menü listeleme, sipariş) ile yanıt sürelerinin ölçülmesi

5.2.1.2.2 Stres testi: Maksimum kullanım (ör. 1.000 eşzamanlı aktif kullanıcı) üzerindeki koşullarda sistemin dayanıklılık sınırlarının tespit edilmesi

5.2.1.3 Güvenlik Testleri

5.2.1.3.1 Penetrasyon testi: SQL injection, XSS, CSRF, broken authentication, JWT kırma denemeleri gibi yaygın zafiyetlerin taranması

5.2.1.3.2 Yetkilendirme/RBAC testleri: "Müşteri" rolünün "Yönetici" rolüne ait ekranlara erişip erişemediğinin sınanması

5.2.1.3.3 Şifre güvenliği testi: SHA-256 hash'in doğruluğu, parola sıfırlama akışında token süre sonu kontrollerinin incelenmesi

5.2.1.4 Kullanılabilirlik (Usability) ve Erişilebilirlik Testleri

5.2.1.4.1 Kullanıcı kabul testleri (UAT): 5–7 gerçek kullanıcıyla senkron test oturumları

5.2.1.4.2 Erişilebilirlik kriterleri: WCAG 2.1 AA düzeyine uygunluk (klavye navigasyonu, kontrast oranı, ARIA etiketleri)

5.2.1.5 Regresyon Testleri

5.2.1.5.1 Her yeni sürümde daha önce başarılı olan test senaryolarının kontrol edilmesi, no-regression sağlanması.

5.3 Test Ortamı ve Kullanılacak Araçlar

Aşağıda hem donanım/yazılım konfigürasyonu hem de tercih edilen test araçları detaylandırılmıştır.

5.3.1 Test Ortamı

5.3.1.1 Donanım

Minimum: 8 GB RAM, 4 çekirdekli CPU, SSD

Önerilen: 16 GB RAM, 8 çekirdekli CPU, NVMe SSD

5.3.1.2 Yazılım

İşletim Sistemi: Ubuntu 22.04 LTS (both local & staging)

Backend: Python 3.10 (Django 4.x, FastAPI 0.95.x)

Veritabanı:

- Geliştirme/Test: SQLite (yerel)
- Staging/Üretim benzeri: MySQL 8.0 veya PostgreSQL 15

Frontend: React 18.x (veya Vue 3.x)

5.3.1.3 Network Konfigürasyonu

Staging ortamı, HTTPS destekli "staging.ornek.com" alt alan adı

İç test ağında (VPN/Docker ağı) izole test senaryoları

5.4 Kullanılacak Araçlar ve Çerçeveler

Test Kategorisi	Araç / Çerçeve	Açıklama
Birim/Entegrasyon Testleri	pytest + pytest-django + pytest-asyncio	Django ve FastAPI birim/entegrasyon testleri; factory_boy/faker ile test verisi üretimi
Frontend Testleri	Jest + React Testing Library + ESLint, Prettier	UI bileşenlerinin birim testleri; kod kalitesi ve stil kontrolleri
Uçtan Uca (E2E) Testleri	Playwright (veya Selenium) / Cypress	Çoklu tarayıcı (Chromium, Firefox, WebKit) ve mobil emülasyon; responsive senaryolar
API Testleri	Postman + Newman / pytest + httpx	Postman koleksiyonlarıyla manuel/otomatik API testleri; pytest-httpx ile entegre test otomasyonu
Performans & Yük Testleri	Apache JMeter / Locust	Eşzamanlı kullanıcı senaryoları; ramp-up ve test süresi tanımları

Güvenlik Testleri	OWASP ZAP / Burp Suite CE / Snyk, Dependabot	Otomatik ve manuel güvenlik taramaları; bağımlılık güvenlik taramaları
Veritabanı Doğrulama	SQLite Browser / MySQL Workbench / pgAdmin	Test veritabanı tabloları, kayıt ve bütünlük kontrolleri
Hata/Kayıt Yönetimi	GitHub Issues / Jira / Sentry	Test sırasında tespit edilen hata ve istisna loglarının takibi ve raporlanması

5.5 Test Süreçleri ve Zamanlama

Bu aşamada hem hazırlık süreci hem de test aktiviteleri sıralanmıştır

5.5.1 Test Hazırlık Aşamaları

Gereksinimlerin Netleşmesi

Her SRS maddesine karşılık “acceptance criteria” (kabul kriterleri) tanımlanacak.

Test Ortamının Kurulumu

Docker Compose ile yerel geliştirme ve test ortamı (backend, veritabanı, frontend).

CI/CD pipeline’da (GitHub Actions/GitLab CI) staging’e otomatik dağıtım.

Test Verilerinin Hazırlanması

factory_boy/faker ile:

1. 50 kullanıcı
2. 20 restoran, her birine 10’ar menü öğesi
3. 100 rastgele rezervasyon
4. 200 yapay yorum

5.5.2 Test Aşamaları ve Zaman Çizelgesi

Aşama	Açıklama	Sorumlu	Zamanlama
Birim Testleri	Kod komit’lerinde her işlev için pytest ile birim testi	Geliştirici	Sürekli
Entegrasyon Testleri	Veritabanı, bildirim servisi, cache entegrasyonlarının testi	QA Ekibi	2. Hafta
E2E Testleri	Playwright/Cypress ile uçtan uca senaryolar; Android/iOS emülasyon	QA Ekibi	3–4. Hafta
Performans Testleri	Locust/JMeter ile orta (200 kullanıcı) ve yoğun (1.000 kullanıcı) senaryoları	QA Ekibi	4. Hafta
Güvenlik Testleri	OWASP ZAP ve Burp Suite ile otomatik/manuel testler	Güvenlik Uzmanı	4. Hafta
Kullanılabilirlik Testleri	Beta kullanıcı oturumları, WCAG 2.1 AA erişilebilirlik kontrolleri	UX Ekibi	5. Hafta
Regresyon Testleri	CI/CD pipeline’da birim+entegrasyon+E2E testlerin otomatik çalışması	QA Ekibi	Sürekli
Test Kapanış Raporu	Tüm testler tamamlandıktan sonra özet rapor hazırlanması	QA Lider	6. Hafta

5.6 Test Senaryoları ve Örnek Matrisler

5.6.1 Test Senaryoları

ID	Başlık	Önkoşul	Adımlar	Beklenen Sonuç	Önem Düzeyi
TS-01	Kullanıcı Kayıt ve Giriş	Yok	1. “Kayıt Ol” sayfasına git 2. Ad, soyad, e-posta, telefon, şifre gir 3. “Kaydol” tıkla	Yeni kullanıcı oluşturulur, onay e-postası gönderilir	Yüksek
TS-02	Rezervasyon Yapma (Başarılı Akış)	Kullanıcı giriş yapmış	1. Ana sayfada restoran ara 2. Restoran seç 3. Geçerli tarih/saat ve masa seç4. “Rezervasyon Yap” tıkla	Rezervasyon “Bekliyor” durumunda veritabanına kaydedilir; onay e-postası gönderilir	Yüksek
TS-03	Rezervasyon Çakışma Kontrolü	Aynı tarih/saat ve masada mevcut rezervasyon	1. Aynı tarih/saat/masa için rezervasyon isteği gönder 2. “Rezervasyon Yap” tıkla	“Seçilen zamanda uygun masa bulunamadı” hatası gösterilir	Orta
TS-04	Menüden Ürün Sepete Ekleme ve Sipariş Gönder	Kullanıcı giriş yapmış	1. Menü ekranına git 2. Bir ürün seç ve “Sepete Ekle” tıkla 3. Sepet > “Sipariş Gönder” tıkla	Sipariş “Hazırlanıyor” durumunda kayıt edilir; kullanıcıya onay iletilir	Yüksek
TS-05	Yorum Yapma ve Moderasyon	Kullanıcı en az bir sipariş vermiş	1. Profil > “Geçmiş Siparişlerim” sayfasına git 2. İlgili yemeğin “Yorum Yap” butonuna tıkla 3. Yorum metni ve 1–5 yıldız girerek “Gönder” tıkla	Yorum “Onay Bekliyor” durumunda kaydedilir; yönetici panelinde onaya düşer	Orta
TS-06	Yönetici Menü Güncelleme	Yönetici olarak giriş yapılmış	1. Yönetici paneline git 2. “Menü Yönetimi” bölümüne tıkla 3. Yeni yemek için ad, açıklama, fiyat, kategori gir4. “Kaydet” tıkla	Yeni menü öğesi veritabanına eklenir; menü listesinde görüntülenir	Orta
TS-07	Performans: 500 Eşzamanlı Kullanıcı Rezervasyon	Veritabanı ve uygulama hazır	1. Locust ile 500 kullanıcıyı rezervasyon endpoint’ine aynı anda yönlendir2. Ramp-up: 5 dakika, test süresi: 30 dakika	Ortalama yanıt süresi < 2 saniye; %95’lik yanıt dilimi < 3 saniye; CPU/RAM < 80 %	Yüksek
TS-08	Güvenlik: SQL Injection Denemesi	Hazır test veritabanı	1. Rezervasyon endpoint’ine date='2025-01-01'; DROP TABLE users; -- gibi kötü niyetli payload gönder	Veritabanı tabloları korunur; istek reddedilir, hata loglanır	Yüksek

5.6.2 Test Veri Matrisi

Test Türü	Araç	Sorumlu	Test Verisi Açıklaması	Test Ortamı
Birim Testleri	pytest + factory_boy	Geliştirici	– 10 kullanıcı modeli– 5 restoran modeli– 20 sipariş modeli	Lokal Docker
Entegrasyon Testleri	pytest + Mailhog/Mok SMS	QA Ekibi	– Gerçek e-posta servisi yerine Mailhog (test SMTP)– Mock edilmiş SMS servisi	Staging
Uçtan Uca Testleri	Playwright / Cypress	QA Ekibi	– Test kullanıcı hesapları (örn. user1@test.com, şifre: Test123!)– Gerçekçi senaryolar: kayıt, rezervasyon, sipariş	Staging
Performans Testleri	Locust / Apache JMeter	QA Ekibi	– 50 restoran, 200 menü öğesi, 1000 test kullanıcı	Staging Özel
Güvenlik Testleri	OWASP ZAP / Burp Suite	Güvenlik Uzmanı	– SQLi, XSS, CSRF payload'ları– JWT manipülasyon script'leri	Staging
Kullanılabilirlik Testleri	Manuel (usability lab)	UX Ekibi	– 5–7 katılımcı ile senkron test oturumları– Erişilebilirlik araçları (Lighthouse, WAVE)	Test Kafesi

5.7 Başarı Kriterleri ve Ölçütler

Bu alt bölüm, test sonuçlarının kabul edilebilirlik sınırlarını ve hedefleri tanımlar.

5.7.1 Fonksiyonel Başarı

Tüm birim testleri %100 geçmeli.
Entegrasyon testleri en az %95 başarı oranı ile tamamlanmalı.
E2E senaryolarının %100'ü hatasız tamamlanmalı.

5.7.2 Performans Başarı

500 eşzamanlı kullanıcı senaryosunda ortalama yanıt süresi < 2 saniye.
%95'lik yanıt dilimindeki (percentile) işlem süresi < 3 saniye.
Sunucu kaynak kullanımı (CPU, RAM) %80'in altında kalmalı.
1.000 eşzamanlı kullanıcı senaryosunda sistem kritik bir çökme (downtime) yaşamadan, makul bir gecikmeyle yanıt vermeye devam etmeli.

5.7.3 Güvenlik Başarı

OWASP ZAP tarafından tespit edilen "High" ve "Critical" seviyedeki açıklar giderilmiş olmalı.
Manuel penetrasyon testlerinde tespit edilen "High" öncelikli zafiyetler kapatılmış olmalı.
JWT kırma, rol eskalasyonu denemelerinde sistem yetkisiz erişime izin vermemeli.

5.7.4 Kullanılabilirlik Başarı

Beta kullanıcı anketlerinde en az %80 "kolay kullanılabilir" değerlendirmesi sağlanmalı.
WCAG 2.1 AA kriterleri karşılanmalı ve "AA uyumlu" raporu belgelendirilmiş olmalı.

5.7.5 Regresyon Başarı

CI/CD pipeline'daki tüm test setleri (birim, entegrasyon, E2E) her "merge" işleminde hatasız çalışmalı.
Daha önce çözülmüş hatalar (bugfix) geriye dönük olarak tekrarlanmamalı (no-regression).

6. Karşılaşılan Zorluklar ve Çözüm Yolları

6.1 Rezervasyon Çakışma ve Eşzamanlılık Sorunları

6.1.1 Karşılaşılan Zorluklar

6.1.1.1 Aynı anda birden fazla kullanıcının aynı masa ve saat aralığı için rezervasyon yapmaya çalışması, çakışmalara yol açtı.
6.1.1.2 SQLite temelli geliştirme ortamında, yüksek eşzamanlı isteklerde kilitlenmeler (database locking) ve yavaşlamalar gözlendi.

6.1.2 Üretilen Çözüm Önerileri

6.1.2.1 *Optimistic Locking ve "WHERE" Koşullu Güncelleme:* Veritabanı güncellemesi sırasında "VERSİYON" (ya da "updated_at" zaman damgası) kontrolü eklenerek, önceki sürümle tutarsızlık varsa kullanıcıya çakışma uyarısı gösterilmesi sağlandı.

6.1.2.2 *Transaction Bazlı Rezervasyon Akışı:* Django ORM üzerinde select_for_update() yöntemi ile satır kilitleme, FastAPI tarafında ise SQLAlchemy'ın with_for_update parametresi kullanılarak, rezervasyon anında satır seviyesinde kilitleme yapıldı. Böylece çakışma ihtimali minimize edildi.

6.1.2.3 *Cache Katmanı (Redis) ile Hızlı Önbellekleme:* Rezervasyon yapılacak masa ve saat bilgileri, kısa süreli (TTL = 30 saniye) Redis önbelleğinde tutulup, rezerve edilecek masa ilk aşamada "tutuldu" olarak işaretlendi. Bu sayede, kullanıcı rezervasyon formunu doldururken aynı masa başka kullanıcı tarafından seçilemedi.

6.2 Gerçek Zamanlı Masa Durumu ve Bildirim Entegrasyonu

6.2.1 Karşılaşılan Zorluklar

6.2.1.1 Masa durumu değişiklikleri (doluluk-boşaltım) anlık olarak tüm bağlı istemcilere (web ve mobil) bildirilmedi. Kullanıcı, rezervasyon ekranında eski bir durumu görüp "habersiz" rezervasyon talepleri oluşturdu.
6.2.1.2 E-posta ve SMS servis sağlayıcı entegrasyonunda, bazı durumlarda bildirim gecikmeleri ve başarısız teslimatlar raporlandı; bu da kullanıcı deneyimini olumsuz etkiledi.

6.2.2 Üretilen Çözüm Önerileri

6.2.2.1 *WebSocket Tabanlı Gerçek Zamanlı Güncellemeler:* Django Channels (veya FastAPI üzerinde WebSocket desteği) kullanılarak "Masa Durumu" değiştiğinde ilgili restoran sayfasını dinleyen tüm istemciler anında güncellendi. Bu sayede kullanıcılar, rezervasyon ekranında her zaman geçerli doluluk bilgisine ulaşabildi.

6.2.2.2 *Asenkron Görev Kuyruğu (Celery + Redis/RabbitMQ):* E-posta ve SMS gönderimleri, anlık (senkron) isteklerden ayrılarak Celery worker'ları üzerinden arka planda işlendi. Bu sayede API istek süreleri kısaldı ve bildirim başarısızlıkları—gerekli durumda—yeniden deneme (retry) yapacak şekilde konfigüre edildi.

6.2.2.3 *Sağlayıcı Çeşitliliği ve Fallback Mekanizmaları:* Temel SMS sağlayıcısının (örn. Twilio) yanı sıra ikincil bir sağlayıcı (örn. Nexmo) eklendi. Birincil sağlayıcı başarısız olduğunda otomatik geçiş yapılarak bildirim teslim oranı %99'un üzerine çıkarıldı.

6.3 Performans ve Yük Altında Yanıt Süresi

6.3.1 Karşılaşılan Zorluklar

6.3.1.1 Ortalama 200–300 eşzamanlı kullanıcı testlerinde, sayfa yükleme süreleri 3–5 saniyeleri buldu. Özellikle menü listeleme ve rezervasyon sorgularında gecikmeler tespit edildi.
6.3.1.2 SQLite kullanımı, çoklu paralel okuyucu-yazıcı işlemlerinde performans darboğazı oluşturdu.

6.3.2 Üretilen Çözüm Önerileri

6.3.2.1 *Veritabanı Geçişi ve İndeksleme:* Geliştirme sırasında SQLite kullanılan veritabanı, staging aşamasında MySQL'e (veya PostgreSQL'e) taşındı. Sorgu çakışan tablolar (Rezervasyon, Masa, Menü) üzerinde uygun kolonlara (örn. restaurant_id, date_time, table_id) B-Tree indeksi eklendi.

6.3.2.2 *Cache Kullanımı (Redis/Memcached):* Menu listeleme ve restoran filtreleme sonuçları, 60 saniye TTL ile Redis önbelleğinde saklanarak tekrar eden isteklerde veritabanına gitme ihtiyacı azaltıldı.

6.3.2.3 *Lazy Loading ve Sayfalama (Pagination):* Menü ekranında 1000'den fazla ürün varsa, ilk aşamada sadece 20 ürün listelenip kullanıcı aşağı kaydırdıkça ek yükler yapılması sağlandı (infinite scroll). Bu, hem frontend performansını hem de ağ yükünü azalttı.

6.3.2.4 *API Optimizasyonu ve SQL Profiling:* Django tarafında select_related() ve prefetch_related() kullanılarak ilişkili tablo sorguları optimize edildi. Ayrıca, ağır sorgular için EXPLAIN ANALYZE çıktıları incelenerek gereksiz JOIN'ler ve büyük veri taramalar azaltıldı.

6.4 Güvenlik ve Yetkilendirme Zorlukları

6.4.1 Karşılaşılan Zorluklar

6.4.1.1 Rol bazlı erişim (RBAC) iş kuralları bazen doğru şekilde uygulanmadı; örneğin, "Müşteri" rolü token manipülasyonu yaparak "Yönetici" paneline kısmen erişebildi.
6.4.1.2 CSRF ve XSS açıklarının bazı form alanlarında yeterince filtrelenmediği tespit edildi. Özellikle yorum formunda zararlı JavaScript kodu enjekte edilmeye çalışıldı.

6.4.2 Üretilen Çözüm Önerileri

6.4.2.1 *JWT Yenileme ve Expiration Control:* Erişim (access) token süresi 15 dakikaya düşürülüp, "refresh token" ile oturum yenileme akışı eklendi. Token içinde "role" bilgisinin yanı sıra "token_issuer_id" gibi alanlar eklenerek, her istekte tokenın orijinalliği kontrol edildi.

6.4.2.2 *Ortam Değişkenleri ve Güvenli Anahtar Yönetimi:* JWT imza anahtarları, parolalar ve API anahtarları .env dosyalarında şifrelenmiş biçimde tutulup, CI/CD pipeline'ında yalnızca deploy zamanı decrypt edildi.

6.4.2.3 *CSRF Koruması ve Input Sanitization:* Django'da CSRF middleware zorunlu hale getirildi. FastAPI uç noktalarında fastapi-csrf-protect entegrasyonu yapıldı. Tüm form girdileri (yorum, profil güncelleme vb.) için Bleach kütüphanesiyle beyaz liste (allowed tags/attributes) tanımlandı.

7. Sonuç ve Tartışma

7.1 Projenin Genel Değerlendirmesi

Dijital Restoran Rezervasyon ve Menü Sistemi projesi, modern yazılım mimarisi ve tasarım desenlerinin etkin biçimde uygulanmasıyla başarılı bir dijital ürün olarak ortaya çıkmıştır. Sistem, restoran işletmelerine tüm rezervasyon ve sipariş süreçlerini dijital ortamda yönetme imkanı sağlamakta; kullanıcılar açısından ise hızlı, güvenilir ve kolay erişilebilir bir platform olarak işlev görmektedir. Singleton, Observer, Factory ve Strategy gibi tasarım desenleri sistemin esnekliğini ve sürdürülebilirliğini artırırken, MVC mimarisi kodun modüler yapısını güçlendirmiştir. Önemli fonksiyonel gereksinimler başarıyla karşılanmış, performans ve güvenlik testleri sonuçları umut vericidir. Uygulamanın responsive yapısı ve kullanıcı deneyimi dikkate alınarak tasarlanması, farklı cihazlardan kullanım kolaylığı sağlamıştır.

7.2 Güçlü Yönler

7.2.1 *Tasarım Desenlerinin Etkin Kullanımı:* Singleton deseni ile kaynakların verimli kullanımı sağlanmış; Observer ve Factory desenleri ile bileşenler arası gevşek bağ kurulmuştur. Bu da sistemin bakım ve genişletilmesini kolaylaştırmıştır.

7.2.2 *Performans Optimizasyonları:* Veritabanı indeksleme, cache kullanımı ve lazy loading gibi teknikler sayesinde yüksek kullanıcı sayısında bile yanıt süreleri kabul edilebilir seviyelerde tutulmuştur. Ayrıca, 500 eş zamanlı kullanıcı testi ile sistem kararlılığı gösterilmiştir.

7.2.3 *Güvenlik Önlemleri:* JWT tabanlı erişim kontrolleri, CSRF koruması ve input sanitization gibi önlemlerle kötü niyetli saldırılara karşı dayanıklı bir yapı oluşturulmuştur.

7.2.4 *Modüler ve Katmanlı Mimari:* MVC yapısı projenin modülerliğini artırarak, farklı ekiplerin paralel çalışmasına ve sistemin ölçeklenebilirliğine olanak tanımıştır.

7.2.5 *Zengin Test Planı:* Birim, entegrasyon, performans ve güvenlik testleri ile yüksek kalite standartları yakalanmış, hata ve zafiyetlerin erken tespiti mümkün olmuştur.

7.3 Zayıf Yönler

7.3.1 *Veritabanı Performansı:* Geliştirme aşamasında kullanılan SQLite, yüksek eş zamanlılıklarda kilitlenme ve yavaşlama problemleri göstermiştir; bu da gerçek ortam performansı üzerine endişe yaratmıştır.

7.3.2 *Bildirim Sistemi Gecikmeleri:* E-posta ve SMS servislerinde zaman zaman gecikmeler ve teslimat başarısızlıkları yaşanmış, bu da kullanıcı memnuniyetini etkileyebilecek bir durumdur.

7.3.3 *Rol Bazlı Erişim Uygulaması:* Bazı durumlarda RBAC kontrollerinde zayıflıklar gözlemlenmiş, token manipülasyonuna karşı daha katı denetimler gerekliliği ortaya çıkmıştır.

7.3.4 *Gerçek Zamanlı Veri Güncellemeleri:* Masa durumu gibi kritik bilgilerde zaman zaman senkronizasyon sorunları ve eski verinin gösterilmesi tespit edilmiştir.

7.4 Geliştirme Önerileri

7.4.1 *Veritabanı Altyapısının Güçlendirilmesi:* Üretim ortamı için SQLite yerine MySQL veya PostgreSQL gibi güçlü ilişkisel veri tabanlarına geçiş yapılması önerilmektedir. Bu sayede eş zamanlı işlem performansı ve veri bütünlüğü artırılabilir.

7.4.2 *Bildirim Altyapısının İyileştirilmesi:* Daha fazla bildirici sağlayıcı entegrasyonu ve asenkron görev yönetimi ile teslimat garantisi ve gecikme oranlarının minimize edilmesi hedeflenmelidir.

7.4.3 *Güvenlik Katmanlarının Güçlendirilmesi:* RBAC ve JWT mekanizmalarının sıkılaştırılması ve düzenli güvenlik testleri ile yetkisiz erişim risklerinin azaltılması gerekmektedir.

7.4.4 *Gerçek Zamanlı Yenileme Mekanizmaları:* WebSocket ve daha etkin cache mekanizmaları kullanılarak anlık veri güncellemeleri ve kullanıcı arayüzü senkronizasyonunun geliştirilmesi faydalı olacaktır.

7.4.5 *Kullanıcı Deneyimi ve Erişilebilirlik:* WCAG 2.1 AA gibi erişilebilirlik standartlarının tam karşılanması ve kullanıcı geri bildirimlerine göre arayüz iyileştirmelerinin sürdürülmesi önemlidir.

7.5 Geleceğe Yönelik Perspektif

Proje, temel fonksiyonlarını başarıyla tamamlamış olmakla birlikte, geleceğe dönük olarak çeşitli yenilikçi teknolojilerin entegrasyonuna açıktır. Örneğin; yapay zeka destekli öneri sistemleri, chatbotların geliştirilmesi ve daha kapsamlı veri analitiği modülleri ile kullanıcı deneyimi zenginleştirilebilir. Bulut tabanlı servis kullanımı ile ölçeklenebilirlik artırılabilir, mikroservis mimarisi uygulamaları ile sistem parçaları daha bağımsız yönetilebilir hale getirilebilir. Sürekli entegrasyon ve dağıtım (CI/CD) süreçleri ile geliştirme hızı ve yazılım kalitesi yükseltilebilir. Ayrıca, güvenlik standartlarının global değişikliklere göre güncel tutulması kritik bir gerekliliktir.

8. Risk Analizi

Risk Tanımı	Olasılık	Etki	Açıklama	Çözüm / Önlem
Yüksek eş zamanlı kullanıcı trafiği sistemi yavaşlatabilir	Orta	Yüksek	Özellikle yoğun saatlerde rezervasyon ve sipariş işlemleri aynı anda yapılabilir	Performans testleri yapılmalı, veritabanı sorguları optimize edilmeli, gerekli yerlerde cacheleme kullanılmalı
Rezervasyon çakışması	Yüksek	Orta	Aynı masa için aynı saatte birden fazla rezervasyon alınabilir	Uygulama katmanında çakışma kontrolü yapılmalı, veritabanında benzersiz rezervasyon kısıtı tanımlanmalı
Yorumlarda spam veya uygunsuz içerik yer alması	Orta	Orta	Kullanıcılar reklam, küfür veya zararlı içerik içeren yorumlar yazabilir	Yorumlar önce moderatör onayına düşmeli, filtreleme (regex, keyword) yapılmalı
Veritabanı erişim hataları	Düşük	Yüksek	Sistem çalışırken bağlantı kopması, veri kaybı yaşanabilir	Günlük otomatik yedekleme yapılmalı, hata anında yeniden deneme (retry) mekanizması uygulanmalı
Kimlik doğrulama açığı	Düşük	Yüksek	Güvensiz istemcilerden token'lar çalınabilir	JWT'ler kısa süreli tutulmalı, HTTPS zorunlu olmalı, oturum süresi kısıtlı tanımlanmalı
Mobil cihazlarda arayüz bozulması	Orta	Düşük	Farklı çözünürlüklerde arayüz düzgün görünmeyebilir	Responsive tasarım testleri yapılmalı, min. ve max. ekran boyutları belirlenmeli
Entegrasyon servisleri çalışmazsa kullanıcı bilgilendirilemez	Orta	Orta	Onaylar ve hatırlatmalar kullanıcıya ulaşmayabilir	Alternatif kanal (uygulama içi bildirim), loglama ve hata takibi eklenmeli

9. Örnek Fonksiyonlar ve Kod Parçacıkları

9.1 Rezervasyon Servisi

9.1.1 *rezervasyonOlustur():* Kullanıcıdan alınan bilgilerle yeni bir rezervasyon oluşturur. Uygunluk kontrolü içerir.

```
public bool RezervasyonOlustur(int userId, int tableId, DateTime date, TimeSpan time, int kisiSayisi)
{
    if (!IsSlotAvailable(tableId, date, time)) return false;

    var rezervasyon = new Reservation
    {
        UserId = userId,
        TableId = tableId,
        Date = date,
        Time = time,
        Guests = kisiSayisi
    };
```

9.1.2 *rezervasyonKontrol():* Belirtilen tarih, saat ve masa için sistemde önceden kayıtlı bir rezervasyon olup olmadığını kontrol eder.

```
public bool IsSlotAvailable(int tableId, DateTime date, TimeSpan time)
{
    return !_context.Reservations.Any(r =>
        r.TableId == tableId &&
        r.Date == date &&
        r.Time == time);
}
```

9.2 Sipariş Servisi

9.2.1 *siparisVer():* Kullanıcının verdiği siparişi ve ilgili ürünleri sistemde kayıt altına alır.

```
public int SiparisVer(int userId, int rezervasyonId, List<int> menuItemIds)
{
    var order = new Order
    {
        UserId = userId,
        ReservationId = rezervasyonId,
        OrderDate = DateTime.UtcNow,
        Status = "hazırlanıyor"
    };
    _context.Orders.Add(order);
    _context.SaveChanges();

    foreach (var itemId in menuItemIds)
    {
        _context.OrderItems.Add(new OrderItem
        {
            OrderId = order.Id,
            MenuItemId = itemId,
            Quantity = 1
        });
    }

    _context.SaveChanges();
    return order.Id;
}
```

9.2.2 *siparisDurumuGuncelle():* Bir siparişin durumunu (“hazırlanıyor”, “hazır”, “teslim edildi” gibi) günceller.

```
public void SiparisDurumuGuncelle(int orderId, string yeniDurum)
{
    var siparis = _context.Orders.Find(orderId);
    if (siparis != null)
    {
        siparis.Status = yeniDurum;
        _context.SaveChanges();
    }
}
```

9.3 Menü Servisi

9.3.1 *menuListele():* Sistemde mevcut olan menü öğelerini (isteğe bağlı kategori filtresiyle) listeler.

```
public List<MenuItem> MenuListele(string? kategori = null)
{
    var query = _context.MenuItems.Where(m => m.IsAvailable);
    if (!string.IsNullOrEmpty(kategori))
        query = query.Where(m => m.Category == kategori);

    return query.OrderBy(m => m.Name).ToList();
}
```

9.3.2 *urunDetay():* Seçilen menü öğesinin detay bilgilerini döner.

```
public MenuItem? UrunDetay(int menuItemId)
{
    return _context.MenuItems.FirstOrDefault(m => m.Id == menuItemId && m.IsAvailable);
}
```

9.4 Yorum Servisi

9.4.1 *yorumKaydet():* Kullanıcının belirli bir ürün için yaptığı yorumu veritabanına "beklemede" durumu ile kaydeder.

```
public void YorumKaydet(int userId, int menuItemId, string yorumMetni, int puan)
{
    var yorum = new Comment
    {
        UserId = userId,
        MenuItemId = menuItemId,
        CommentText = yorumMetni,
        Rating = puan,
        CreatedAt = DateTime.UtcNow,
        Status = "beklemede"
    };

    _context.Comments.Add(yorum);
    _context.SaveChanges();
}
```

9.4.2 *puanOrtalamaHesapla():* Belirli bir ürün için yapılan yayımlanmış yorumların puan ortalamasını hesaplar.

```
public double PuanOrtalamaHesapla(int menuItemId)
{
    var yorumlar = _context.Comments
        .Where(c => c.MenuItemId == menuItemId && c.Status == "yayinda")
        .ToList();

    if (yorumlar.Count == 0) return 0;

    return yorumlar.Average(c => c.Rating);
}
```

9.5 Bildirim Servisi

9.5.1 *smsGonder():* Rezervasyon onayı, sipariş bildirimi gibi amaçlarla SMS gönderimini simüle eder.

```
public void SmsGonder(string telefon, string mesaj)
{
    Console.WriteLine($"[SMS] {telefon} numarasına mesaj gönderildi: {mesaj}");
}
```

9.5.2 *epostaGonder():* Rezervasyon onayı, sipariş bildirimi gibi amaçlarla e-posta gönderimini simüle eder.

```
public void EpostaGonder(string email, string konu, string icerik)
{
    Console.WriteLine($"E-Posta gönderildi -> {email} | Konu: {konu}\n{icerik}");
}
```

10. Kaynakça

- Yapay zeka uygulaması (OpenAI GPT-3 / https://openai.com/gpt-3)
- Çevrimiçi diyagram ve akış şeması yazılımı (Lucidchart / https://www.lucidchart.com)
- UML çizim aracı (PlantUML / https://plantuml.com)
- Veritabanı şeması çizimi için araç (dbdiagram.io / https://dbdiagram.io)
- JWT güvenlik modeli (ASP.NET) (Microsoft Docs / https://learn.microsoft.com/en-us/aspnet/core/security/authentication/jwt)
- Web erişilebilirlik standardı (W3C WCAG 2.1 / https://www.w3.org/TR/WCAG21)