YAPAY ZEKÂ DESTEKLİ YEMEK ÖNERİ VE TARİF DEFTERİ PLATFORMU-OleAI

Öğrenci Adı Soyadı: Onur Aydı

Öğrenci Numarası: 221201011

Danışman Adı: Prof. Dr. Taner Çevik

Bölüm: Bilgisayar Mühendisliği

Dönem: Güz Dönemi-2025

Teslim Tarihi: 18.01.2026

GitHub: https://github.com/onuraydi/OleAIDocuments

(NOT: GitHub Reposunda proje ile ilgili tasarım ve tabloların hepsini, daha yüksek çözünürlükte bulabilirsiniz)

1.Giriş:

OleAI, kullanıcıların sahip oldukları gıda malzemeleri, beslenme tercihleri ve geçmiş etkileşimleri doğrultusunda yapay zekâ destekli yemek önerileri sunmayı amaçlayan web ve mobil tabanlı bir yazılım sistemidir. Projenin temel hedefi; kullanıcı odaklı, ölçeklenebilir ve sürdürülebilir bir mimari yapıya sahip, gerçek dünya senaryolarında kullanılabilir nitelikte bir öneri sistemi geliştirmektir.

Proje kapsamında sistem gereksinimleri detaylı bir şekilde analiz edilmiş; fonksiyonel ve fonksiyonel olmayan gereksinimler kullanıcı hikâyeleri ile ilişkilendirilerek tanımlanmıştır. Elde edilen gereksinimler doğrultusunda yüksek seviyeli sistem mimarisi, veri tabanı modeli, iş mantığı yapısı ve bileşenler arası entegrasyon planı tasarlanmıştır. Sistem; web ve mobil istemciler, merkezi bir backend servisi ve yapay zekâ tabanlı öneri modülünden oluşan katmanlı bir mimari ile geliştirilmiştir.

Backend katmanı .NET platformu üzerinde Clean Architecture prensipleri doğrultusunda yapılandırılmış; RESTful API yaklaşımı ile istemcilerden bağımsız bir servis altyapısı oluşturulmuştur. Veri yönetimi, ilişkisel veri tabanı yapısına uygun olarak SQL tabanlı bir sistem üzerinden gerçekleştirilmiştir. Yapay zekâ öneri sistemi ise kullanıcı etkileşimlerini ve tarif içeriklerini analiz ederek dinamik ve kişiselleştirilmiş öneriler üretmektedir.

Sistem tasarımı sürecinde UML diyagramları (Use Case, Class, Sequence, ER ve Component Diagramları) kullanılarak mimari kararlar görselleştirilmiş; veri akış diyagramları ile sistemin işleyişi netleştirilmiştir. Ayrıca entegrasyon planı, farklı bileşenlerin birbiriyle nasıl haberleştiğini açık şekilde ortaya koymaktadır.

Sonuç olarak OleAI projesi; yazılım mühendisliği prensiplerine uygun, modüler, genişletilebilir ve bakım yapılabilir bir yapı sunmakta olup, yapay zekâ destekli öneri sistemlerinin web ve mobil platformlarda nasıl entegre edilebileceğini bütüncül bir yaklaşımla ele almaktadır. Proje, analizden tasarıma kadar tüm aşamaları kapsayan kapsamlı bir yazılım geliştirme sürecinin somut bir çıktısı niteliğindedir.

2. Gereksinim Analizi:

2.1 Proje Tanımı:

Bu proje, kullanıcıların mutfaklarında bulunan mevcut malzemeleri, bireysel beslenme tercihlerini ve geçmiş kullanım alışkanlıklarını dikkate alarak kişiselleştirilmiş yemek önerileri sunmayı amaçlayan, yapay zekâ destekli bir mobil ve web uygulamasının geliştirilmesini kapsamaktadır. Sistem, günlük yaşamda sıkça karşılaşılan “ne pişirileceği” problemine kullanıcı odaklı ve veri temelli bir çözüm sunmayı hedeflemektedir. Aynı zamanda gıda israfının azaltılması, zaman yönetiminin iyileştirilmesi ve sağlıklı beslenme alışkanlıklarının desteklenmesi proje kapsamında ele alınan temel motivasyonlar arasındadır.

Geliştirilecek yazılım çözümü, modern yazılım mühendisliği prensipleri doğrultusunda; ölçeklenebilir, sürdürülebilir ve bakım yapılabilir bir mimari yapıya sahip olacak şekilde tasarlanmaktadır. Proje yalnızca işlevsel gereksinimleri karşılamayı değil, aynı zamanda yazılım kalitesi, güvenlik, performans ve uzun vadeli genişletilebilirlik gibi niteliksel kriterleri de ön planda tutmaktadır. Bu yaklaşım, sistemin gerçek dünya senaryolarında etkin ve güvenilir bir şekilde kullanılabilmesini amaçlamaktadır.

Sistem mimarisi; kullanıcı etkileşimini sağlayan mobil ve web istemciler (client), bu istemcilerle haberleşen bir sunucu tarafı (backend) ve sunucu üzerinde çalışan yapay zekâ destekli öneri sistemi bileşenlerinden oluşmaktadır. Bu çok katmanlı ve modüler yapı sayesinde, sistem bileşenleri birbirinden bağımsız olarak geliştirilebilmekte ve ilerleyen süreçlerde yeni özelliklerin eklenmesi veya mevcut bileşenlerin iyileştirilmesi

mümkün hâle gelmektedir. Yapay zekâ bileşeni, kullanıcıdan elde edilen verileri analiz ederek zaman içerisinde daha isabetli ve kişiselleştirilmiş öneriler üretmeyi hedeflemektedir.

2.2 Hedef Kitle Analizi:

OleAI, günlük yaşamın yoğunluğunda yemek planlama ve mutfak yönetimi süreçlerinde karar verme zorluğu yaşayan kullanıcı gruplarına yönelik olarak tasarlanmıştır. Hedef kitle analizi, sistemin işlevsel ve kullanıcı deneyimi gereksinimlerinin doğru şekilde belirlenebilmesi amacıyla gerçekleştirilmiştir. Bu analiz kapsamında, potansiyel kullanıcıların davranışları, ihtiyaçları ve beklentileri dikkate alınarak sistemin yapısı şekillendirilmektedir.

1-) Günlük yaşam temposu içerisinde pratik ve hızlı çözümler arayan yoğun bireyler: Bu kullanıcılar genellikle yoğun iş hayatına sahip, ne yemek yapacağına karar vermekte zorlanan, karar verme süreçlerine zaman harcamak istemeyen kullanıcılardır. Bu kullanıcılar yemek planlama sürecini daha verimli hâle getirecek akıllı bir sisteme ihtiyaç duymaktadır. Bu doğrultuda, sistemin kullanıcı ara yüzü bu karar verme sürecini daha da zorlaştırmayacak şekilde sade ve kolay kullanılabilir şekilde tasarlanacaktır.

2-) Beslenme tercihleri veya kısıtları bulunan bireyler: Vejetaryen, vegan veya belirli bir gıda hassasiyetine sahip kullanıcılar, kendilerine uygun tariflere ulaşmakta zorlanabilirler. OleAI, kullanıcıların beslenme tercihlerini ve kısıtlarını dikkate alarak kişiselleştirilmiş öneriler sunmayı amaçlamakta; bu sayede kullanıcıların sistemle olan etkileşimlerini artırmayı ve uzun vadeli kullanım sağlamayı hedeflemektedir.

3-) Gıda israfını azaltma konusunda bilinçli bireyler: Evde bulunan malzemelerin etkin şekilde değerlendirilmesini sağlayan öneri mekanizması, bu kullanıcı grubunun motivasyonlarıyla doğrudan örtüşmektedir. Sistem, malzeme bazlı öneriler sunarak hem ekonomik hem de çevresel açıdan sürdürülebilir bir kullanım deneyimi sağlamayı amaçlamaktadır.

Son olarak, hedef kitle içerisinde yer alan kullanıcıların teknik bilgi seviyeleri farklılık gösterebilmektedir. Bu nedenle sistem, teknik yeterliliği sınırlı kullanıcıların dahi rahatlıkla kullanabileceği bir yapı sunarken, daha deneyimli kullanıcılar için gelişmiş filtreleme ve kişiselleştirme seçenekleri de sağlamayı hedeflemektedir. Bu yaklaşım, OleAI'ın geniş bir kullanıcı kitlesine hitap edebilmesini ve farklı kullanım senaryolarında etkin bir şekilde kullanılabilmesini mümkün kılmaktadır.

2.3 Paydaşlar ve Kullanıcı Profilleri:

Bu bölümde, OleAI sisteminin geliştirilmesi, işletilmesi ve kullanımı sürecinde rol alan paydaşlar ile sistemle doğrudan etkileşime giren kullanıcı profilleri tanımlanmaktadır. Paydaşların ve kullanıcı profillerinin doğru şekilde belirlenmesi, gereksinimlerin netleştirilmesi ve sistemin farklı beklentilere uygun olarak tasarlanabilmesi açısından kritik öneme sahiptir.

2.3.1 Paydaşlar:

OleAI projesi kapsamında tanımlanan başlıca paydaşlar aşağıda sunulmuştur:

1-) Son Kullanıcılar: Sistemi mobil veya web platformları üzerinden aktif olarak kullanan, yemek önerileri alan, tarifleri ekleyebilen, diğer tariflerle etkileşimde bulunabilen ve diğer birçok işlevi yerine getiren bireylerdir. Projenin temel hedef kitlesini oluşturmaktadırlar.

2-) Ürün Sahibi (Product Owner): Sistem gereksinimlerinin belirlenmesinden, önceliklendirilmesinden ve ürün vizyonunun korunmasından sorumlu paydaştır. Kullanıcı ihtiyaçları ile teknik gereksinimler arasında denge kurulmasını sağlar.

3-) Geliştirici/Geliştirme Ekibi: Mobil uygulama, web arayüzü, backend servisleri ve yapay zekâ bileşenlerinin tasarım ve geliştirilmesinden sorumludur. Sistem mimarisi, kod kalitesi ve teknik sürdürülebilirlik bu paydaş grubunun sorumluluğundadır. Sistem ilk etapta tek kişi tarafından geliştirilecek olsa da ilerleyen aşamalarda bir geliştirici ekibi kurulacaktır.

4-) Sistem Yöneticisi (Admin): Sistem üzerinde içerik, kullanıcı ve genel yapılandırma işlemlerini yöneten paydaştır. Uygulamanın sağlıklı ve güvenli bir şekilde çalışmasını sağlar.

2.3.2 Kullanıcı Profilleri:

OleAI, sisteminde yer alan kullanıcılar, sistemle olan etkileşim biçimlerine göre farklı profiller altında incelenmektedir.

1-) Standart Kullanıcı: Sistemin temel işlevlerini kullanan, mevcut mutfak malzemelerini sisteme tanımlayan ve önerilen tarifleri görüntüleyen kullanıcı grubudur. Bu kullanıcı profili için öncelikli beklenti, hızlı ve pratik öneriler sunulmasıdır.

2-) İçerik Üreten Kullanıcı: Kendi tariflerini sisteme ekleyebilen ve diğer kullanıcılarla paylaşabilen kullanıcı grubudur. Bu kullanıcılar, sistemin sosyal ve topluluk odaklı yönünü güçlendirmektedir.

3-) Kısıtlı Beslenme Tercihine Sahip Kullanıcı: Vejetaryen, vegan veya belirli besinlere karşı hassasiyeti bulunan kullanıcılardan oluşmaktadır. Bu kullanıcı profili için sistemin kişiselleştirme ve filtreleme yetenekleri ön plana çıkmaktadır.

4-) Yönetici Kullanıcı (Admin): Kullanıcı hesaplarını, içerikleri ve sistem yapılandırmalarını yöneten yetkili kullanıcıdır. Sistem güvenliği ve içerik denetimi açısından kritik bir role sahiptir.

2.4 Proje Kapsamı (Scope Definition):

Bu bölümde, OleAI projesi kapsamında geliştirilecek işlevler, sistem bileşenleri ve özellikler ile proje kapsamı dışında bırakılan unsurlar açık ve net bir şekilde tanımlanmaktadır. Proje kapsamının doğru belirlenmesi, gereksinimlerin yönetilebilir hale getirilmesi, zaman ve kaynak planlamasının sağlıklı yapılabilmesi açısından büyük önem taşımaktadır.

2.4.1 Kapsam Dahilinde Olan İşlevler:

OleAI projesi kapsamında geliştirilecek başlıca işlevler aşağıda sıralanmıştır:

1-) Kullanıcı Hesap Yönetimi: Kullanıcıların sisteme kaydolabilmesi, kimlik doğrulama işlemlerini gerçekleştirebilmesi ve kendi profillerini yönetebilmesi.

2-) Mutfak Envanteri (Dolabım) Yönetimi: Kullanıcıların evlerinde bulunan gıda malzemelerini sisteme ekleyebilmesi, güncelleyebilmesi ve bu verilerin öneri mekanizmasında kullanılabilmesi. Bir tarif yapıldığında ilgili malzemeleri stoktan düşme ve biten malzemeleri alışveriş listesine ekleme.

3-) Yapay Zekâ Destekli Yemek Öneri Sistemi: Kullanıcıların mevcut malzemeleri, beslenme tercihleri ve geçmiş etkileşimleri dikkate alınarak kişiselleştirilmiş yemek önerilerinin sunulması.

4-) Tarif Görüntüleme ve Yönetimi: Sistem içerisinde yer alan tariflerin görüntülenmesi, filtrelenmesi ve detaylı şekilde incelenmesi.

5-) Tarif Ekleme ve Paylaşma: Kullanıcıların kendi tariflerini sisteme ekleyebilmesi ve bu tarifleri isteğe bağlı olarak diğer kullanıcıların görebileceği şekilde paylaşabilmesi.

6-) Beslenme Tercihi ve Kısıt Yönetimi: Kullanıcıların vejetaryen, vegan veya belirli alerjenlere yönelik tercihlerini sisteme tanımlayabilmesi ve önerilerin bu doğrultuda şekillendirilmesi.

7-) Temel Sosyal Etkileşim Özellikleri: Tarifler üzerinde beğenme, beğenmeme, kaydetme görüntüleme gibi etkileşimlerin sağlanması.

2.4.1 Kapsam Dışı Olan İşlevler:

Proje sürecinde odak kaybını önlemek ve sistemin temel hedeflerine yoğunlaşabilmek amacıyla aşağıdaki işlevler bilinçli olarak proje kapsamı dışında tutulmuştur:

1-) Gerçek Zamanlı Market ve Tedarikçi Entegrasyonları: Market stokları, fiyat karşılaştırmaları veya marketlerde otomatik sepet oluşturma gibi entegrasyonlar kapsam dışındadır.

2-) Donanım veya IoT Entegrasyonları: Akıllı buzdolabı, barkod okuyucu veya sensör tabanlı donanım entegrasyonları proje kapsamına dâhil değildir.

3-) Profesyonel Diyetisyen veya Sağlık Hizmetleri: Kişisel sağlık danışmanlığı, profesyonel diyetisyen hizmetleri veya tıbbi öneriler sunulmamaktadır.

Sonuç olarak, OleAI projesinin kapsamı; kullanıcıların mutfak yönetimi ve yemek planlama süreçlerini destekleyecek temel ve katma değerli işlevlere odaklanacak şekilde sınırlandırılmıştır. Bu yaklaşım, sistemin hem teknik hem de işlevsel açıdan tutarlı, yönetilebilir ve sürdürülebilir bir yapıda geliştirilmesini amaçlamaktadır.

2.5 Kullanıcı Hikayeleri (User Stories):

Bu bölümde, OleAI sisteminin işlevsel kapsamı; kullanıcıların sistemle olan etkileşimlerini, beklentilerini ve olası kullanım senaryolarını yansıtacak şekilde yapılandırılmış kullanıcı hikâyeleri aracılığıyla tanımlanmaktadır. Kullanıcı hikâyeleri, yalnızca sistemin ne yapacağını değil; aynı zamanda hangi koşullarda, hangi öncelikle ve hangi iş değeriyle gerçekleştireceğini ortaya koymaktadır.

Tanımlanan hikâyeler, ürün geliştirme sürecinde gereksinim analizine, kullanım senaryolarına (use case) ve test süreçlerine doğrudan girdi sağlayacak şekilde yapılandırılmıştır. Her hikâye; rol, öncelik, bağımlılık, amaç, iş değeri ve kabul kriterleri bileşenlerini içermektedir. Gerekli görülen durumlarda alternatif ve istisnai akışlar da tanımlanmıştır.

2.5.1 Temel Kullanıcı Hikayeleri:

ID	Hikaye	Rol	Öncelik	Bağımlılık	Amaç & İş Değeri	Kabul Kriterleri
US-01	Kullanıcı Kayıt ve Kimlik Doğrulama	Son Kullanıcı	High	Yok	**Amaç:** Sisteme güvenli erişim **Değer:** Kişiselleştirilmiş ve güvenli deneyim	• Kayıt ve giriş yapılabilmeli • Güvenli kimlik doğrulama • Yetkisiz erişim engeli
US-02	Mutfak Envanteri (Dolabım) Yönetimi	Son Kullanıcı	High	US-01	**Amaç:** Malzemeleri tanımlamak **Değer:** Önerileri kişiselleştirmek	• Malzeme ekleme/silme/güncellem • Malzemelerin öneride kullanımı

US-03	Yapay Zekâ Destekli Yemek Önerisi	Son Kullanıcı	High	US-02	**Amaç:** Uygun tarif önerileri almak **Değer:** Karar süresini azaltmak, israfı önlemek	• Mevcut malzemeleri dikkate alma • Eksik malzemeleri belirtme • Tercihlere uygunluk

2.5.2 AI ve Kişiselleştirme Odaklı Senaryolar:

ID	Hikaye	Rol	Öncelik	Bağımlılık	Amaç & İş Değeri	Kabul Kriterleri
US-04	Beslenme Tercihlerinin Tanımlanması	Son Kullanıcı	High	US-01	**Amaç:** Tercihleri sisteme tanımlamak **Değer:** Uygunsuz içeriklerin elenmesi	• Tercihleri güncelleyebilme • Önerilerin tercihlere uyması
US-05	Kullanıcı Davranışına Göre Öneri Kalitesi	Son Kullanıcı	Medium	US-03	**Amaç:** Etkileşime göre isabetli öneri **Değer:** Uzun vadede kişiselleştirme	• Beğeni/etkileşim analizi • Sonuçların zamanla iyileşmesi
US-06	Yetersiz Veride Alternatif Öneri	Son Kullanıcı	Medium	US-02, US-03	**Amaç:** Eksik veride bile öneri almak **Değer:** Kullanıcıyı sistemde tutmak	• Veri yetersizliği tespiti • Alternatif/en yakın tarif sunumu

2.5.3 Sosyal ve İçerik Odaklı Senaryolar:

ID	Hikaye	Rol	Öncelik	Bağımlılık	Amaç & İş Değeri	Kabul Kriterleri
US-07	Tarif Görüntüleme ve Filtreleme	Son Kullanıcı	Medium	US-01	**Amaç:** Hızlı ve düzenli erişim **Değer:** Kullanıcı deneyimi iyileştirme	• Tarif filtreleme • Detayların açık sunumu
US-08	Tarif Ekleme ve Paylaşma	İçerik Üreten	Medium	US-01	**Amaç:** İçeriği sisteme kazandırmak **Değer:** İçeriğin zenginleştirilmesi	• Tarif ekleyebilme • Herkese açık/özel paylaşım

2.5.4 Diğer Senaryolar:

ID	Hikaye	Rol	Öncelik	Bağımlılık	Amaç & İş Değeri	Kabul Kriterleri
US-09	Uygun Tarif Bulunamaması Durumu	Son Kullanıcı	Low	US-03	**Amaç:** Öneri yoksa bilgilendirilmek **Değer:** Kullanıcı memnuniyeti	• Açık bilgilendirme • Alternatif aksiyonlar
US-10	Kullanıcı ve İçerik Yönetimi	Admin	Medium	US-01	**Amaç:** Sistem düzeni ve güvenliği **Değer:** Platform sürdürülebilirliği	• Kullanıcı/içerik yönetimi • Gerektiğinde müdahale

Bu kullanıcı hikâyeleri; OleAI sisteminin temel işlevlerini, yapay zekâ destekli karar mekanizmalarını ve istisnai durumları kapsayacak şekilde bütüncül bir bakış açısıyla ele alınmıştır. Tanımlanan senaryolar, bir sonraki aşamada gerçekleştirilecek olan Fonksiyonel ve Fonksiyonel Olmayan Gereksinim Analizi için doğrudan referans niteliği taşımaktadır.

2.6 Gereksinim Analizi:

Bu bölümde OleAI sisteminin fonksiyonel ve fonksiyonel olmayan gereksinimleri sistematik bir şekilde tanımlanmaktadır. Gereksinim analizi, sistemin hangi işlevleri yerine getireceğini ve bu işlevleri hangi kalite kriterleri altında sunacağını açıkça ortaya koymayı amaçlamaktadır. Tanımlanan gereksinimler, önceki aşamalarda oluşturulan kullanıcı hikâyeleri ile doğrudan ilişkilendirilmiş olup, tasarım, geliştirme ve test süreçleri için temel bir referans niteliği taşımaktadır.

2.6.1 Fonksiyonel Gereksinimler:

Fonksiyonel gereksinimler, OleAI sisteminin kullanıcılar ve yöneticiler için sunması gereken işlevleri tanımlamaktadır.

ID	Başlığı	Açıklama	Hikâye
FR-01	Kullanıcı Kayıt ve Kimlik Doğrulama	Sistem, kullanıcıların güvenli bir şekilde kaydolmasına ve kimlik doğrulama işlemlerini gerçekleştirmesine olanak sağlamalıdır.	US-01
FR-02	Kullanıcı Profil Yönetimi	Sistem, kullanıcıların profil bilgilerini görüntüleyebilmesine ve güncelleyebilmesine imkân tanımalıdır.	US-01, US-04
FR-03	Mutfak Envanteri (Dolabım) Yönetimi	Sistem, kullanıcıların sahip oldukları gıda malzemelerini ekleyebilmesine, güncelleyebilmesine ve silebilmesine olanak sağlamalıdır.	US-02
FR-04	Yapay Zekâ Destekli Yemek Önerisi	Sistem, kullanıcının tanımladığı malzemeleri ve tercihlerini dikkate alarak yemek önerileri üretmelidir.	US-03
FR-05	Eksik Malzeme Bilgilendirmesi	Sistem, önerilen tariflerde eksik olan malzemeleri kullanıcıya açık ve anlaşılır bir şekilde göstermelidir.	US-03
FR-06	Beslenme Tercihi ve Kısıt Filtreleme	Sistem, kullanıcı tarafından tanımlanan beslenme tercihlerini ve kısıtlarını öneri sürecine dâhil etmelidir.	US-04
FR-07	Kullanıcı Etkileşimlerinin Analizi	Sistem, kullanıcıların tarifler üzerindeki etkileşimlerini (görüntüleme, beğeni vb.) analiz edebilmelidir.	US-05
FR-08	Alternatif Öneri Üretimi	Sistem, yeterli veri bulunmadığında (yetersiz veri durumu) alternatif veya en yakın tarif önerilerini sunabilmelidir.	US-06
FR-09	Tarif Görüntüleme ve Filtreleme	Sistem, tariflerin kullanıcı tarafından belirlenen kriterlere göre görüntülenmesine ve filtrelenmesine olanak sağlamalıdır.	US-07
FR-10	Tarif Ekleme ve Paylaşma	Sistem, kullanıcıların kendi tariflerini ekleyebilmesine ve bu tarifleri herkese açık veya özel olarak paylaşabilmesine imkân tanımalıdır.	US-08
FR-11	Uygun Tarif Bulunamaması Durumu	Sistem, uygun tarif üretilemediği durumlarda kullanıcıyı bilgilendirmeli ve alternatif aksiyonlar sunmalıdır.	US-09
FR-12	Yönetici (Admin) İşlevleri	Sistem, yetkili yöneticilerin kullanıcıları ve içerikleri yönetebilmesine olanak sağlamalıdır.	US-10

2.6.2 Fonksiyonel Olmayan Gereksinimler:

Fonksiyonel olmayan gereksinimler, sistemin nasıl çalışması gerektiğini tanımlamakta olup yazılım kalitesi, güvenlik ve sürdürülebilirlik açısından kritik öneme sahiptir.

ID	Kategori	Gereksinim Detayları ve Kriterler
NFR-01	**Performans**	• Sistem, kullanıcı taleplerine makul süreler içerisinde yanıt vermelidir. • Yapay zekâ öneri işlemleri, kullanıcı deneyimini olumsuz etkilemeyecek şekilde optimize edilmelidir.
NFR-02	**Güvenlik**	• Kullanıcı verileri güvenli şekilde saklanmalıdır (Encryption). • Kimlik doğrulama ve yetkilendirme mekanizmaları (OAuth, JWT vb.) uygulanmalıdır. • Yetkisiz erişimler engellenmelidir.
NFR-03	**Ölçeklenebilirlik**	• Sistem, artan kullanıcı ve veri hacmine (High Load) uyum sağlayabilecek şekilde tasarlanmalıdır. • Mimari yapı, yeni özelliklerin eklenmesine (Extensibility) olanak tanımalıdır.
NFR-04	**Kullanılabilirlik**	• Kullanıcı arayüzü sade, anlaşılır ve kullanıcı dostu (User-friendly) olmalıdır. • Teknik bilgisi sınırlı kullanıcılar için dahi öğrenme eğrisi düşük olmalıdır.
NFR-05	**Bakım Yapılabilirlik**	• Sistem modüler bir mimariye (Microservices veya Modular Monolith) sahip olmalıdır. • Kod yapısı okunabilir, standartlara uygun ve sürdürülebilir olmalıdır.
NFR-06	**Güvenilirlik**	• Sistem beklenmeyen durumlarda kararlı davranış sergilemelidir (Fault Tolerance). • Hata durumlarında kullanıcı nazikçe bilgilendirilmelidir.
NFR-07	**Taşınabilirlik**	• Sistem, mobil (iOS/Android) ve web platformlarında tutarlı (Responsive/Cross-platform) bir deneyim sunmalıdır.

2.6.3 Gereksinim İzlenebilirlik Matrisi:

Bu bölümde, OleAI sistemi için tanımlanan fonksiyonel ve fonksiyonel olmayan gereksinimlerin; kullanıcı hikâyeleri ve sistem bileşenleri ile olan ilişkisi gereksinim izlenebilirlik matrisi aracılığıyla sunulmaktadır. Bu yapı, gereksinimlerin tasarım, geliştirme ve test süreçleri boyunca takip edilebilirliğini sağlamaktadır.

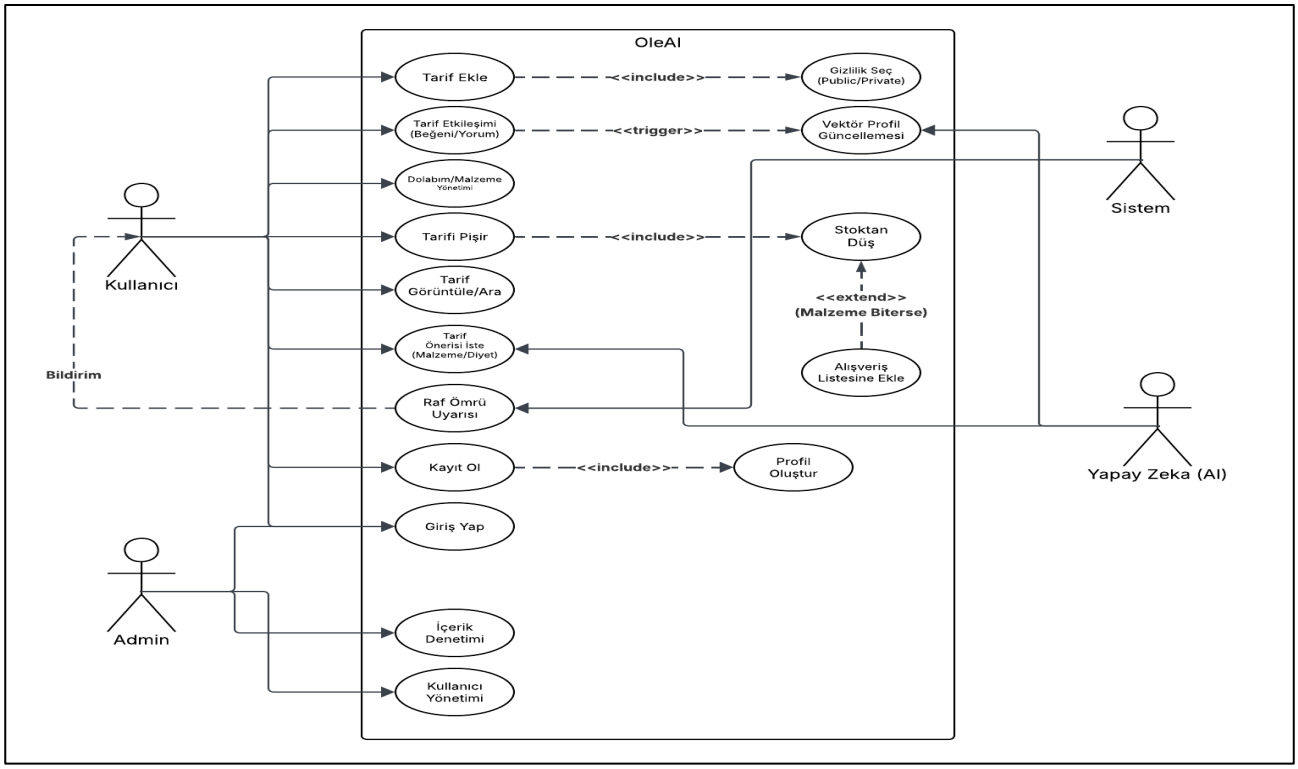
ID	Gereksinim Tanımı	Tür	İlgili US	Sistem Bileşeni	Öncelik	Doğrulama Yöntemi
FR-01	Kullanıcı kayıt ve kimlik doğrulama işlemleri	Fonksiyonel	US-01	Backend	Yüksek	Senaryo Testi
FR-02	Kullanıcı profil bilgilerinin görüntülenmesi/güncellenmesi	Fonksiyonel	US-01, US-04	Backend	Orta	Fonksiyonel Test
FR-03	Mutfak envanterinin (Dolabım) yönetilmesi	Fonksiyonel	US-02	Backend	Yüksek	Senaryo Testi
FR-04	Yapay zekâ destekli yemek önerilerinin üretilmesi	Fonksiyonel	US-03	AI / Backend	Yüksek	Model Çıktı Analizi
FR-05	Eksik malzeme bilgisinin kullanıcıya sunulması	Fonksiyonel	US-03	Backend	Orta	Senaryo Testi

FR-06	Beslenme tercihleri ve kısıtların sürece dâhil edilmesi	Fonksiyonel	US-04	AI / Backend	Yüksek	Fonksiyonel Test
FR-07	Kullanıcı etkileşimlerinin analiz edilmesi	Fonksiyonel	US-05	AI / Backend	Orta	Veri Analizi
FR-08	Yetersiz veri durumunda alternatif önerilerin sunulması	Fonksiyonel	US-06	AI	Orta	Senaryo Testi
FR-09	Tariflerin görüntülenmesi ve filtrelenmesi	Fonksiyonel	US-07	Backend / Frontend	Yüksek	Kullanıcı Testi
FR-10	Kullanıcıların tarif ekleyebilmesi ve paylaşabilmesi	Fonksiyonel	US-08	Backend	Orta	Fonksiyonel Test
FR-11	Uygun tarif bulunamadığında kullanıcı bilgilendirilmesi	Fonksiyonel	US-09	Backend	Orta	Senaryo Testi
FR-12	Yönetici (admin) işlemlerinin gerçekleştirilmesi	Fonksiyonel	US-10	Backend	Düşük	Yetki Testi
NFR-01	Sistem performansının kabul edilebilir yanıtta olması	Fonk. Olmayan	–	Backend / AI	Yüksek	Performans Testi
NFR-02	Kullanıcı verilerinin güvenliğinin sağlanması	Fonk. Olmayan	–	Backend	Yüksek	Güvenlik Testi
NFR-03	Sistemin ölçeklenebilir mimariye sahip olması	Fonk. Olmayan	–	Sistem Geneli	Orta	Mimari İnceleme
NFR-04	Sistemin kullanıcı dostu ve erişilebilir olması	Fonk. Olmayan	–	Frontend	Orta	Kullanılabilirlik Testi
NFR-05	Sistemin bakım yapılabilir ve modüler olması	Fonk. Olmayan	–	Sistem Geneli	Orta	Kod İncelemesi
NFR-06	Sistemin güvenilir ve hataya dayanıklı çalışması	Fonk. Olmayan	–	Backend	Orta	Hata Senaryosu
NFR-07	Web ve mobil platformlarda tutarlı çalışması	Fonk. Olmayan	–	Frontend	Orta	Platform Testi

Bu bölümde tanımlanan fonksiyonel ve fonksiyonel olmayan gereksinimler, OleAI sisteminin kapsamını, sınırlarını ve kalite beklentilerini açık bir şekilde ortaya koymaktadır. Gereksinimlerin kullanıcı hikâyeleri ile ilişkilendirilmiş olması, sistem tasarımı ve test süreçlerinde izlenebilirlik sağlamaktadır. Bu yapı, projenin bir sonraki aşamalarında gerçekleştirilecek olan risk analizi, süreç akışları ve use case çalışmalarına sağlam bir temel oluşturmaktadır.

2.7 Use Case'ler:

Bu bölümde OleAI sisteminin fonksiyonel gereksinimleri, Use Case UML diyagramı ile modellenmiştir. Diyagramda, sistem ile etkileşime giren aktörler ve bu aktörlerin gerçekleştirebildiği temel kullanım senaryoları gösterilmektedir.



2.8 Risk Analizi ve Önleme Stratejileri:

Bu bölümde OleAI'a projesinde ortaya çıkabilecek riskler, ortaya çıkma olasılıkları, etkileri, risk seviyeleri ve önleme stratejileri bulunmaktadır.

ID	Risk Tanımı	Risk Türü	Olasılık	Etki	Risk Seviyesi	Önleme / Azaltma Stratejisi
R-01	Yapay zekâ modelinin yetersiz veya hatalı öneriler üretmesi	Teknik	Orta	Yüksek	**Yüksek**	Modelin iteratif olarak iyileştirilmesi, alternatif öneri mekanizması geliştirilmesi.
R-02	Kullanıcıdan alınan verilerin eksik veya hatalı olması	Operasyonel	Yüksek	Orta	**Yüksek**	Veri doğrulama kontrolleri, kullanıcı yönlendirmeleri.
R-03	Sistem performansının yoğun kullanımda düşmesi	Teknik	Orta	Yüksek	**Yüksek**	Ölçeklenebilir mimari, performans testleri.
R-04	Kullanıcı verilerinin güvenliği ile ilgili açıklar	Güvenlik	Düşük	Çok Yüksek	**Yüksek**	Güçlü kimlik doğrulama, erişim kontrolü, güvenli veri saklama.
R-05	Kullanıcıların sistemi terk etmesi (düşük benimseme)	Kullanıcı Deneyimi	Orta	Orta	**Orta**	Kullanıcı dostu arayüz, kişiselleştirilmiş öneriler.
R-06	Proje kapsamının kontrolsüz şekilde genişlemesi	Yönetimsel	Orta	Orta	**Orta**	Net kapsam tanımı, gereksinimlerin önceliklendirilmesi.

Bu tablo kapsamında belirlenen riskler, proje süresince düzenli olarak gözden geçirilmekte ve gerekli önlemler alınmaktadır. Risklerin erken aşamada tanımlanması, projenin zamanında, güvenli ve sürdürülebilir bir şekilde tamamlanmasına katkı sağlamaktadır.

3. Sistem Tasarımı:

Bu bölüm OleAI'a ait bazı tasarımları içermektedir.

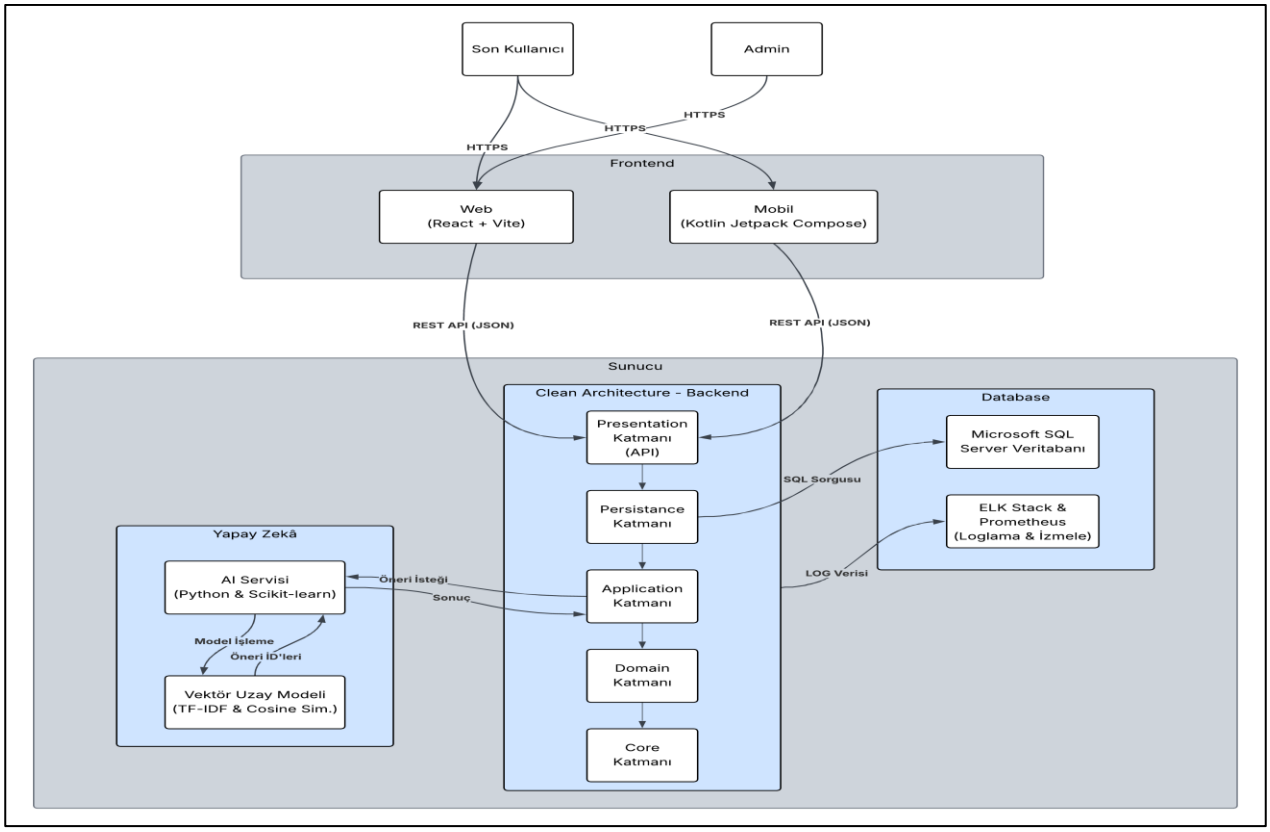
3.1 Mimari Tasarım:

Bu bölümde, OleAI sisteminin yüksek seviyeli mimari tasarımı sunulmaktadır. Analiz aşamasında belirlenen gereksinimler doğrultusunda sistemin genel yapısı, bileşenleri ve bu bileşenler arasındaki ilişkiler tanımlanmıştır.

Şekilde de gösterildiği üzere OleAI sistemi; web istemcisi, mobil istemci, backend API katmanı, yapay zekâ öneri modülü ve veri tabanı bileşenlerinden oluşmaktadır. Web ve mobil istemciler, kullanıcı etkileşimlerini yönetmekte ve backend sistem ile RESTful API'ler aracılığıyla JSON formatında haberleşmektedir.

Sistemin merkezinde yer alan Backend katmanı; iş mantığını, güvenlik protokollerini ve veri yönetimini Clean Architecture prensipleriyle üstlenerek istemcilerin veri tabanına doğrudan erişimini engeller. Yapay zekâ destekli öneri sistemi, backend altyapısına entegre edilmiş bağımsız bir bileşen olarak tasarlanmıştır. Bu yapı, öneri modülünün sistemin diğer bileşenlerinden izole edilerek geliştirilmesine ve ölçeklenmesine olanak sağlamaktadır.

Sistem mimarisi, modülerlik ve bakım yapılabilirlik ilkeleri gözetilerek tasarlanmış olup, ilerleyen aşamalarda yeni istemcilerin veya servislerin eklenmesini destekleyecek esnek bir yapı sunmaktadır.



3.2 Frontend Tasarımları:

////////////// BU KISIM DAHA SONRA DOLDURULACAK

3.3 Backend Tasarımı:

Bu bölümde, OleAI sisteminin backend katmanında yer alan iş mantığı bileşenleri ve bu bileşenlerin sorumlulukları tanımlanmaktadır.

3.3.1 İş Mantığı (Business Logic) Tasarımı:

İş mantığı; kullanıcı etkileşimlerinden bağımsız olarak sistemin temel kurallarını, veri işleme süreçlerini ve uygulama akışlarını yönetmektedir. Bu yaklaşım, sistemin sürdürülebilir, test edilebilir ve genişletilebilir bir yapıda geliştirilmesini amaçlamaktadır. İş mantığı, projenin application katmanında yönetilecektir.

1-) Kullanıcı ve Kimlik Yönetimi (User Management): Bu bileşen, kullanıcıların sisteme kaydolma, kimlik doğrulama ve yetkilendirme süreçlerini yönetmektedir. Kullanıcıya ait temel bilgiler, rol ve yetkiler bu katman aracılığıyla kontrol edilmektedir. Güvenlik gereksinimleri doğrultusunda, kimlik doğrulama işlemleri merkezi olarak ele alınmakta ve sistemin diğer bileşenleri bu mekanizma üzerinden yetkilendirilmektedir.

2-) Profil ve Tercih Yönetimi (Profile & Preference Management): Bu bileşen, kullanıcıların beslenme tercihleri, kısıtları ve kişisel ayarlarının yönetilmesinden sorumludur. Tanımlanan tercihler, yapay zekâ destekli öneri sistemine girdi olarak sağlanmakta ve önerilerin kişiselleştirilmesinde aktif rol oynamaktadır.

3-) Mutfak Envanteri Yönetimi (Kitchen Inventory Management): Mutfak envanteri bileşeni, kullanıcıların sahip oldukları malzemelerin eklenmesi, güncellenmesi ve silinmesi işlemlerini gerçekleştirmektedir. Bu bilgiler, öneri sürecinin temel veri kaynaklarından biri olup sistemin doğru ve anlamlı sonuçlar üretmesi açısından kritik öneme sahiptir.

4-) Tarif Yönetimi (Recipe Management): Bu bileşen, sistemde yer alan tariflerin oluşturulması, görüntülenmesi, filtrelenmesi ve paylaşılması süreçlerini kapsamaktadır. Tariflere ait içerik, malzeme listeleri ve besin değerleri gibi bilgiler bu domain altında yönetilmektedir.

5-) Yapay Zekâ Destekli Öneri Yönetimi (Recommendation Management): Öneri yönetimi bileşeni, kullanıcıdan elde edilen envanter, tercih ve etkileşim verilerini kullanarak yapay zekâ tabanlı yemek önerilerinin üretilmesini sağlamaktadır. Backend katmanı, bu süreçte yapay zekâ modülü ile kontrollü bir şekilde haberleşmekte ve üretilen sonuçları kullanıcıya sunulacak formata dönüştürmektedir.

6-) Kullanıcı Etkileşimlerinin Analizi (User Interaction Analysis): Bu bileşen, kullanıcıların tarifler üzerindeki etkileşimlerini (görüntüleme, beğeni, geri bildirim vb.) analiz etmektedir. Toplanan bu veriler, öneri sisteminin zamanla daha isabetli sonuçlar üretmesini sağlamak amacıyla değerlendirilmekte ve modele geri besleme olarak sunulmaktadır.

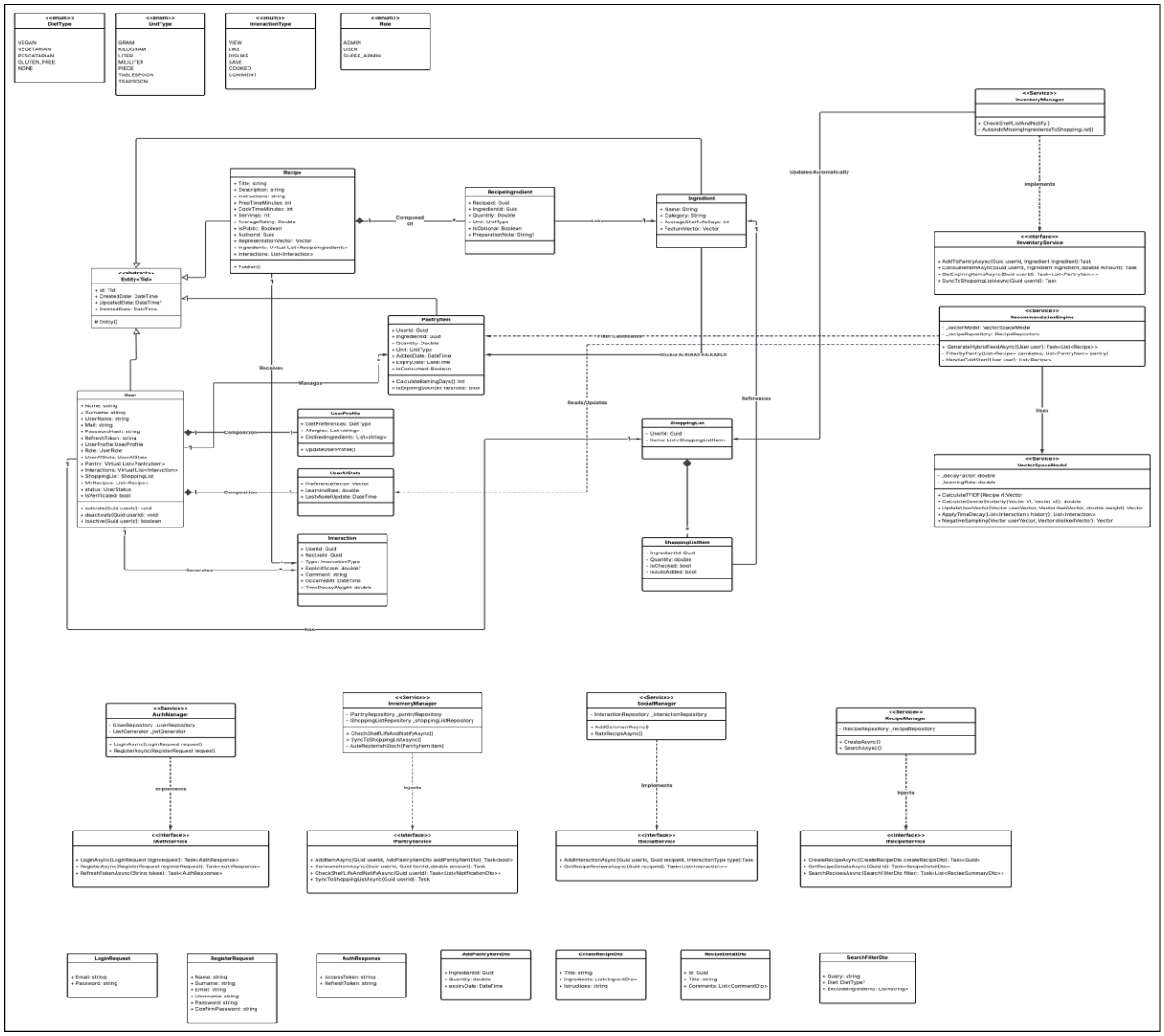
7-) Yönetici İşlevleri (Admin Management): Yönetici bileşeni, yetkili kullanıcıların sistem üzerinde içerik ve kullanıcı yönetimi işlemlerini gerçekleştirmesine olanak tanımaktadır. Bu kapsamda, tariflerin denetlenmesi, kullanıcıların kontrol edilmesi ve sistem bütünlüğünün korunmasına yönelik işlemler yürütülmektedir.

3.3.2 UML Class Diyagramı:

Bu bölümde OleAI sisteminin iş mantığı ve nesne yapısı, UML Class Diyagramı ile modellenmiştir. Diyagram, sistemin domain ve application katmanlarında yer alan temel sınıfları, bu sınıfların sahip olduğu alanları (attributes), davranışsal metotları (methods) ve aralarındaki ilişkileri göstermektedir.

Class diyagramında entity sınıfları, yalnızca veri taşıyan yapılar olarak değil; sistemin iş kurallarını temsil eden davranışsal metotları da içerecek şekilde tasarlanmıştır. Bu yaklaşım, iş mantığının merkezi ve tutarlı bir şekilde yönetilmesini sağlamaktadır.

Sınıflar arasındaki ilişkiler, sistemdeki gerçek iş akışlarını yansıtacak biçimde association, aggregation ve composition ilişkileri kullanılarak modellenmiştir. Diyagram, veri erişim detaylarından bağımsız olarak, sistemin kavramsal ve mantıksal yapısını ortaya koymaktadır.



3.3.3 Sequence Diyagramları:

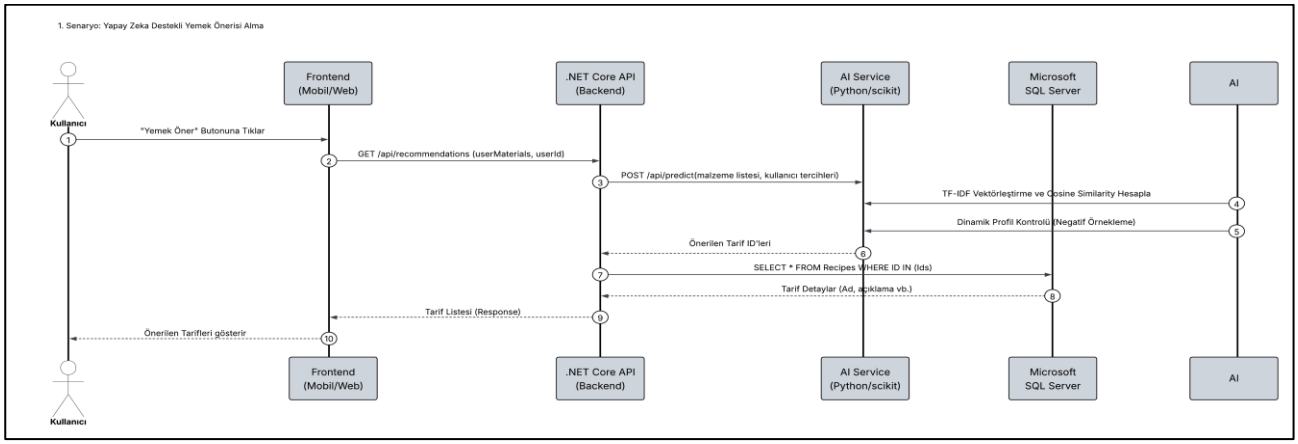
Bu bölümde OleAI sisteminde gerçekleşen temel iş akışları, UML Sequence Diyagramları kullanılarak modellenmiştir. Sequence diyagramları; sistem bileşenleri arasındaki mesajlaşmayı, metod çağrı sırasını ve süreç içerisindeki etkileşimleri zamansal olarak göstermeyi amaçlamaktadır. Bu diyagramlar sayesinde, sistemin dinamik davranışı ve bileşenler arası iletişim açık bir şekilde ortaya konulmaktadır.

1-) Yapay Zekâ Destekli Öneri Alma:

Aşağıdaki sequence diyagramında, kullanıcının sahip olduğu malzemeler ve tercihleri doğrultusunda yapay zekâ destekli yemek önerilerinin oluşturulma süreci adım adım gösterilmektedir. Süreç, kullanıcının istemci uygulama üzerinden öneri talebinde bulunmasıyla başlamakta ve önerilen tariflerin kullanıcıya sunulmasıyla sonlanmaktadır.

Bu akışta, istemci uygulama backend API'lerine istek gönderir. Backend katmanı, kullanıcıya ait profil ve envanter bilgilerini değerlendirerek yapay zekâ öneri modülünü tetikler. Yapay zekâ modülü, ilgili verileri analiz ederek uygun tarifleri belirler ve sonuçları backend katmanına iletir. Backend, elde edilen sonuçları işleyerek istemciye geri döndürür.

Bu yapı, iş mantığının merkezi olarak backend katmanında yönetilmesini ve yapay zekâ bileşeninin sistemden bağımsız, modüler bir servis olarak çalışmasını sağlamaktadır.



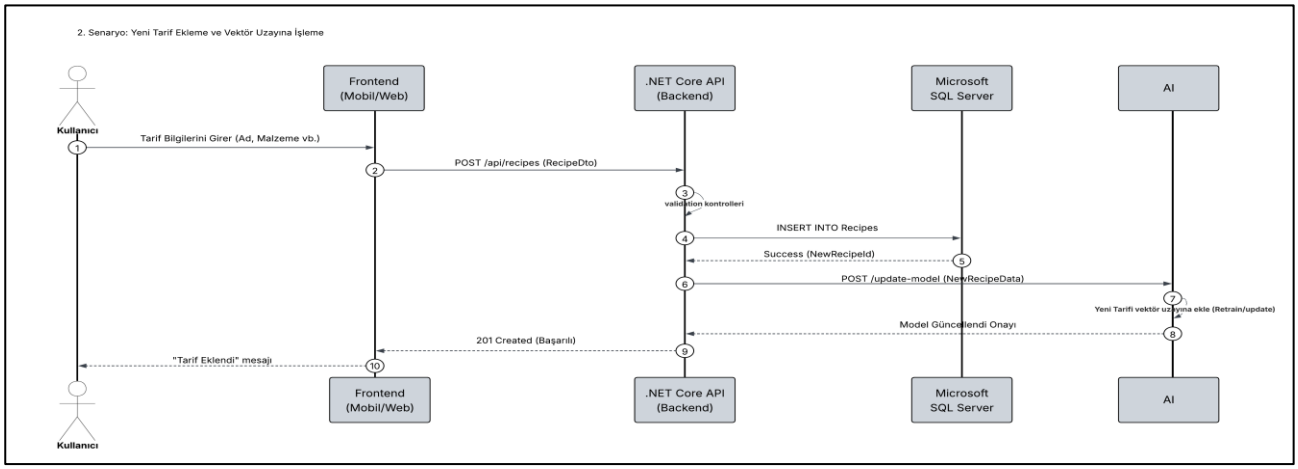
2-) Yeni Tarif Ekleme ve Vektör Uzayına İşleme:

Bu sequence diyagramı, kullanıcı tarafından sisteme eklenen yeni bir tarifin, yapay zekâ öneri sistemine entegre edilme sürecini göstermektedir. Süreç, kullanıcının istemci uygulama üzerinden yeni tarif bilgilerini girmesiyle başlamakta ve tarifin vektör uzayında temsil edilerek öneri sistemine dâhil edilmesiyle tamamlanmaktadır.

Kullanıcı tarafından girilen tarif bilgileri, istemci uygulama aracılığıyla backend API katmanına iletilir. Backend katmanı, gelen verilerin doğrulama ve iş kurallarına uygunluk kontrollerini gerçekleştirdikten sonra tarif bilgilerini kalıcı veri deposuna kaydeder. Kaydetme işleminin ardından, ilgili tarif verileri yapay zekâ servis katmanına iletilir.

Yapay zekâ modülü, tarif içeriğini ön işleme adımlarından geçirerek metinsel verileri sayısal vektör temsiline dönüştürür. Bu aşamada, tarif ve malzeme bilgileri belirlenen özellik çıkarım (feature extraction) yöntemleri kullanılarak vektör uzayında konumlandırılır. Elde edilen vektör temsili, mevcut tarif vektörleri ile birlikte öneri modeline entegre edilir.

Bu yaklaşım sayesinde, sisteme sonradan eklenen tarifler, ek bir yeniden eğitim sürecine gerek duyulmadan öneri mekanizmasına dinamik olarak dâhil edilebilmekte ve sistemin zaman içerisinde daha zengin ve güncel öneriler sunması sağlanmaktadır.



3-) Dolabım Sekmesinin Yönetimi ve Akıllı Alışveriş Listesi:

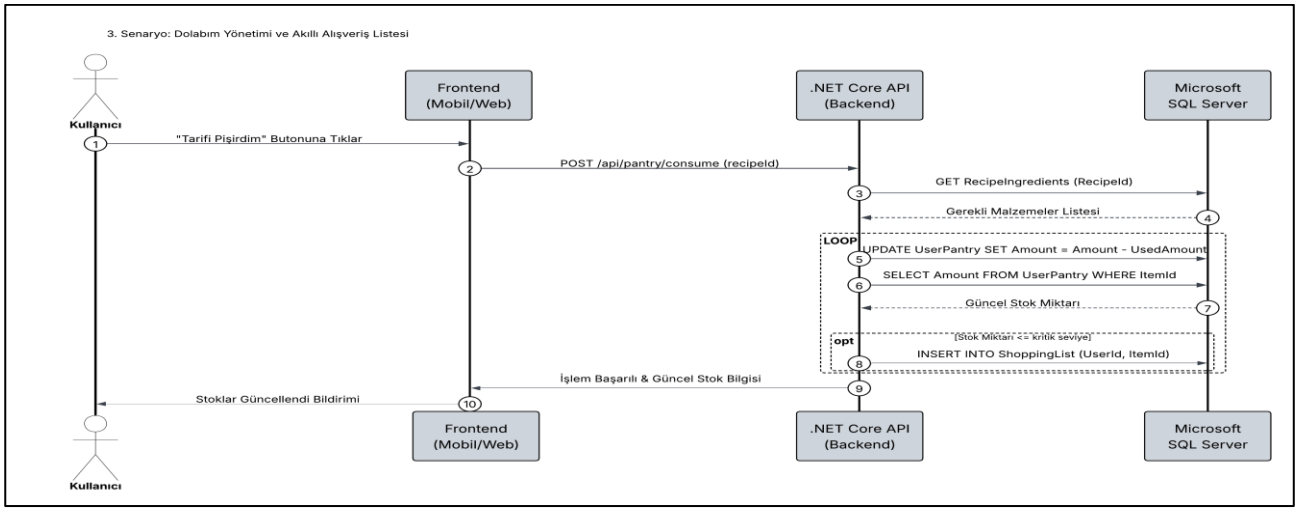
Bu sequence diyagramı, kullanıcının mutfak envanterini (“Dolabım”) yönetmesi ve bu envanter verilerine dayalı olarak akıllı alışveriş listesinin oluşturulması sürecini bütüncül bir iş akışı olarak göstermektedir. Süreç,

kullanıcının sahip olduğu gıda malzemelerini sisteme eklemesi, güncellemesi veya silmesi ile başlamakta; tarif gereksinimleri doğrultusunda eksik malzemelerin belirlenmesi ve alışveriş listesinin oluşturulmasıyla tamamlanmaktadır.

Kullanıcı, istemci uygulama üzerinden mutfak envanterine yeni malzeme ekleme, mevcut malzemelerin miktar veya son kullanma tarihi bilgilerini güncelleme ya da envanterden malzeme silme işlemlerini gerçekleştirir. Bu işlemler backend API katmanına iletilir ve tanımlı iş kuralları kapsamında doğrulama kontrollerinden geçirilir. Doğrulanan envanter bilgileri veri tabanına kaydedilerek sistem genelinde güncel hâle getirilir.

Akıllı alışveriş listesi oluşturma aşamasında, backend katmanı kullanıcı tarafından seçilen tariflerin gerektirdiği malzemeleri, kullanıcının mevcut envanteri ile karşılaştırır. Bu karşılaştırma sonucunda eksik olan malzemeler ve gerekli miktarlar belirlenir. Aynı malzemeye ait tekrar eden ihtiyaçlar birleştirilerek optimize edilmiş bir alışveriş listesi oluşturulur.

Oluşturulan alışveriş listesi istemci uygulama üzerinden kullanıcıya sunulur. Kullanıcı, liste üzerinde düzenleme yapabilir ve alışveriş tamamlandıktan sonra ilgili malzemeleri doğrudan mutfak envanterine ekleyerek sistemin güncel kalmasını sağlayabilir. Bu bütünleşik yapı, kullanıcı deneyimini artırırken envanter yönetimi ile öneri ve planlama süreçleri arasında tutarlı bir veri akışı sağlamaktadır.

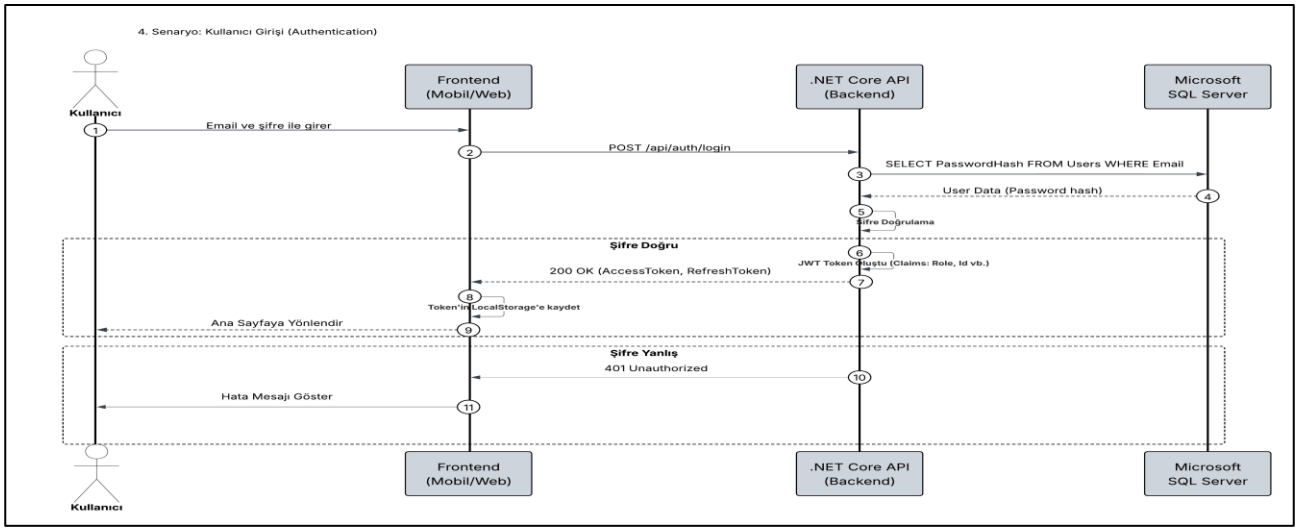


4-) Kullanıcı Girişi (Authentication):

Bu sequence diyagramı, kullanıcının sisteme giriş yapma sürecini ve kimlik doğrulama mekanizmasını göstermektedir. Süreç, kullanıcının istemci uygulama üzerinden giriş bilgilerini göndermesiyle başlamakta ve yetkilendirilmiş bir oturumun oluşturulmasıyla tamamlanmaktadır.

İstemci uygulama tarafından iletilen kullanıcı kimlik bilgileri backend API katmanında doğrulanmaktadır. Kimlik doğrulama işlemi sırasında kullanıcı bilgileri kontrol edilmekte ve gerekli güvenlik doğrulamaları uygulanmaktadır. Doğrulamanın başarılı olması durumunda, kullanıcı için yetkilendirme bilgileri oluşturulmakta ve istemci uygulamaya iletilmektedir.

Bu yapı, web ve mobil istemcilerin ortak bir kimlik doğrulama altyapısı kullanmasını sağlayarak güvenli ve tutarlı bir erişim mekanizması sunmaktadır.



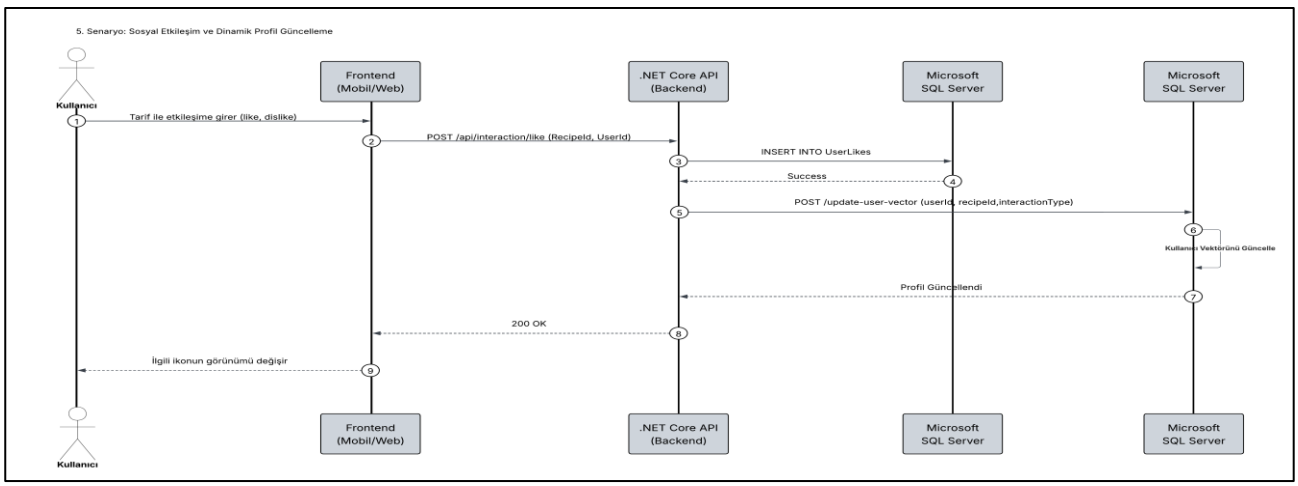
5-) Sosyal Etkileşim ve Dinamik Profil Güncelleme:

Bu sequence diyagramı, kullanıcıların tarifler üzerindeki sosyal etkileşimlerinin (beğenme, beğenmeme, görüntüleme gibi) sistem tarafından nasıl işlendiğini ve bu etkileşimlerin kullanıcı profiline dinamik olarak nasıl yansıtıldığını göstermektedir. Süreç, kullanıcının bir tarif ile etkileşime girmesiyle başlamakta ve kullanıcı profilinin güncellenmesiyle devam etmektedir.

Kullanıcının gerçekleştirdiği sosyal etkileşimler, istemci uygulama aracılığıyla backend API katmanına iletilir. Backend katmanı, bu etkileşimleri ilgili tarif ve kullanıcı ile ilişkilendirerek veri tabanına kaydeder. Kaydedilen etkileşim verileri, kullanıcının davranışsal tercihlerini temsil eden dinamik profil bilgilerinin güncellenmesinde kullanılmaktadır.

Dinamik profil güncelleme sürecinde, kullanıcının zaman içerisinde gerçekleştirdiği etkileşimler analiz edilerek tercih vektörleri güncellenir. Bu güncellenmiş profil bilgileri, yapay zekâ destekli öneri sistemine girdi olarak sağlanır. Böylece sistem, kullanıcının yalnızca statik tercihlerine değil, zamanla değişen davranışlarına da uyum sağlayarak daha isabetli ve kişiselleştirilmiş öneriler sunabilmektedir.

Bu yaklaşım, sistemin kullanıcı alışkanlıklarına uyum sağlamasını sağlayan sürekli öğrenen bir yapı sunmakta ve öneri kalitesini zaman içerisinde artırmaktadır.



3.4 Veri Tabanı Tasarımı:

Bu bölümde OleAI sisteminin kalıcı veri yönetimi için kullanılan veri tabanı yapısı tasarlanmıştır. Veri tabanı tasarımı, sistemin fonksiyonel gereksinimleri, iş mantığı katmanı ve UML Class Diyagramı dikkate alınarak gerçekleştirilmiştir. Tasarım sürecinde veri bütünlüğü, tutarlılık, performans ve genişletilebilirlik kriterleri ön planda tutulmuştur.

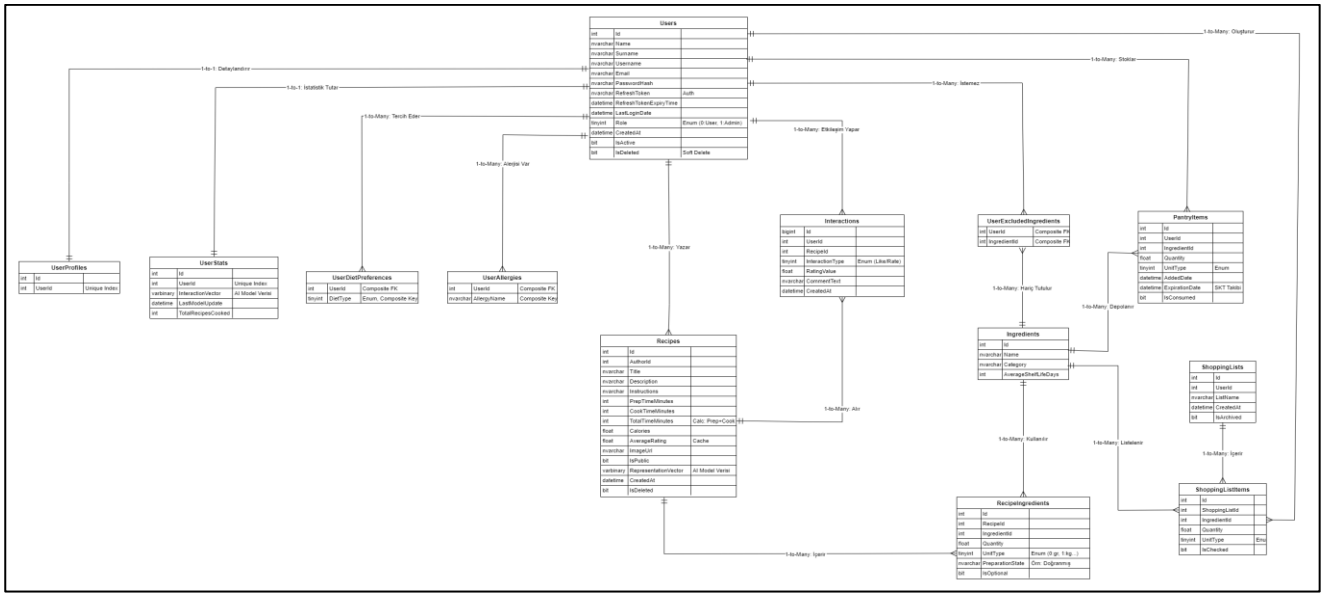
3.4.1 Veri Modeli ve ER Diyagramı:

OleAI sisteminde kullanılan veri tabanı modeli, ilişkisel veri tabanı yaklaşımı temel alınarak tasarlanmıştır. Sistem içerisinde kullanıcılar, tarifler, mutfak envanteri, sosyal etkileşimler ve yapay zekâ öneri süreçlerine ait veriler arasında güçlü ilişkiler bulunmaktadır. Bu nedenle, veri modelinin açık, tutarlı ve normalize edilmiş bir yapıda olması hedeflenmiştir.

Aşağıda yer alan ER (Entity-Relationship) diyagramı, sistemde bulunan temel varlıkları (entity), bu varlıklara ait öznitelikleri (attributes) ve aralarındaki ilişkileri göstermektedir. Diyagram, sistemin veri yapısının kavramsal düzeyde anlaşılmasını sağlamaktadır.

ER diyagramı tasarlanırken, daha önce oluşturulan UML Class Diyagramı referans alınmış ve iş mantığı katmanındaki entity'ler ile veri tabanı tabloları arasında doğrudan bir eşleşme sağlanmıştır. Bu yaklaşım, uygulama katmanı ile veri katmanı arasında tutarlılık oluşturmakta ve bakım yapılabilirliği artırmaktadır.

Veri modelinde, her tablo için birincil anahtarlar (Primary Key) tanımlanmış, tablolar arasındaki ilişkiler yabancı anahtarlar (Foreign Key) aracılığıyla kurulmuştur. Bu sayede veri bütünlüğü korunmakta ve ilişkisel sorguların güvenli bir şekilde gerçekleştirilmesi sağlanmaktadır.



3.5 Entegrasyon Planı:

OleAI sisteminde yer alan bileşenlerin birbiriyle tutarlı ve platformdan bağımsız şekilde haberleşebilmesi amacıyla katmanlı ve servis odaklı bir entegrasyon yaklaşımı benimsenmiştir. Entegrasyon planı; **Front-End ↔ Back-End** ve **Back-End ↔ Yapay Zekâ Servisi** olmak üzere iki temel eksen üzerinde kurgulanmıştır.

3.5.1 Frontend ve Backend Entegrasyonu:

Kullanıcı arayüzü ile sunucu tarafı arasındaki iletişim, RESTful API mimarisi temel alınarak gerçekleştirilmiştir. Frontend katmanı, sistemde sunulan tüm işlevlere HTTP protokolü üzerinden erişmekte olup, veri alışverişi JSON (JavaScript Object Notation) formatı kullanılarak sağlanmaktadır.

Bu yaklaşım sayesinde:

- Web ve mobil platformlar aynı API uç noktalarını kullanabilmekte,
- Platform bağımsızlık sağlanmakta,
- İstemci tarafı teknolojilerinden bağımsız bir yapı elde edilmektedir.

Kimlik doğrulama gerektiren isteklerde, kullanıcı oturum bilgileri güvenli şekilde taşınmakta ve yetkilendirme kontrolleri backend katmanında gerçekleştirilmektedir.

3.5.2 Backend ve Yapay Zekâ Entegrasyonu:

OleAI sisteminde yapay zekâ destekli tarif öneri mekanizması, backend katmanı üzerinden yönetilmektedir. Kullanıcıdan alınan tercihler, mutfak envanteri bilgileri ve geçmiş etkileşimler, öncelikle sunucu tarafında işlenmekte; ardından ilgili veriler yapay zekâ servislerine aktarılmaktadır.

Bu entegrasyon sürecinde:

- Yapay zekâ servisleri backend katmanından izole edilmiştir,
- Veri alışverişi REST tabanlı servis çağrıları ile sağlanmıştır,
- Genişletilebilir ve değiştirilebilir bir mimari hedeflenmiştir.

Bu yapı, ilerleyen aşamalarda farklı yapay zekâ modellerinin veya servislerinin sisteme entegre edilebilmesine olanak tanımaktadır.

3.5.3 API İletişim Yapısı:

Sistem genelinde API iletişimi aşağıdaki prensiplere göre tasarlanmıştır:

- REST standartlarına uygun uç noktalar
- Anlamlı HTTP durum kodlarının kullanımı
- JSON tabanlı veri transferi
- Platformdan bağımsız erişim

Bu yaklaşım, OleAI sisteminin hem bakımını hem de ileride yapılacak entegrasyonları kolaylaştırmaktadır.

3.6 Veri Akış Diyagramı (DFD):

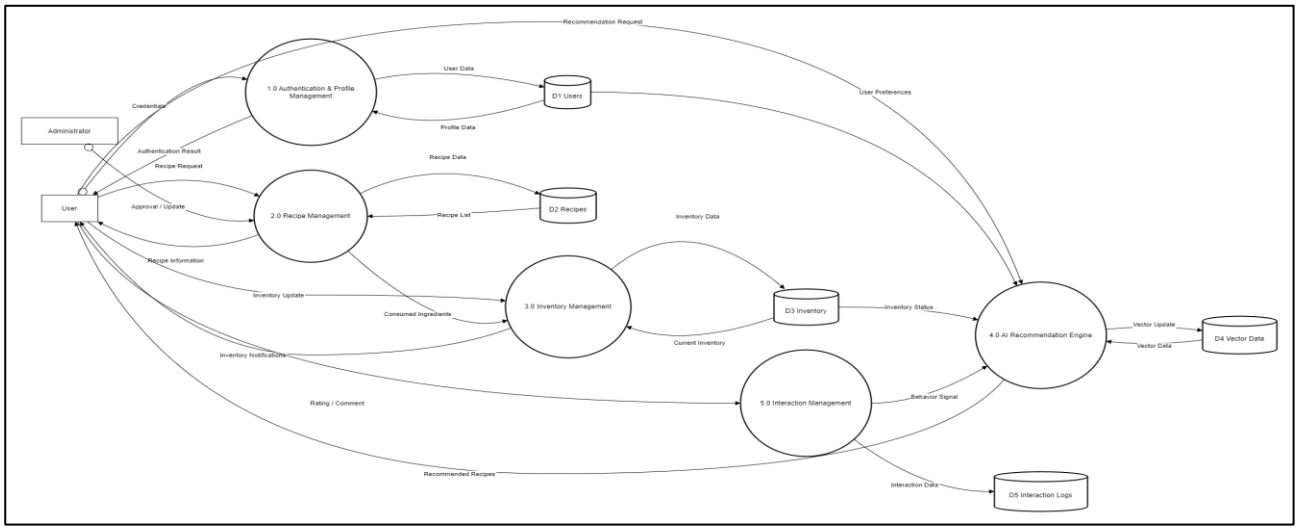
OleAI sisteminin temel veri giriş–çıkışlarını ve ana iş süreçlerini görselleştirmek amacıyla Veri Akış Diyagramı (Data Flow Diagram – DFD) oluşturulmuştur. Bu diyagram, sistemde yer alan aktörler ile ana süreçler arasındaki veri hareketini yüksek seviyede göstermektedir.

Hazırlanan DFD, Seviye-0 (Context Diagram) yaklaşımına uygun olarak tasarlanmıştır. Diyagramda aşağıdaki temel bileşenler yer almaktadır:

- Aktörler: Kullanıcı, Yönetici

- Ana Süreçler: Kullanıcı yönetimi, tarif öneri süreci, mutfak envanteri yönetimi, alışveriş listesi oluşturma
- Veri Depoları: Kullanıcı verileri, tarif verileri, malzeme ve envanter bilgileri
- Veri Akışları: Kullanıcı girdileri, sistem çıktıları ve öneri sonuçları

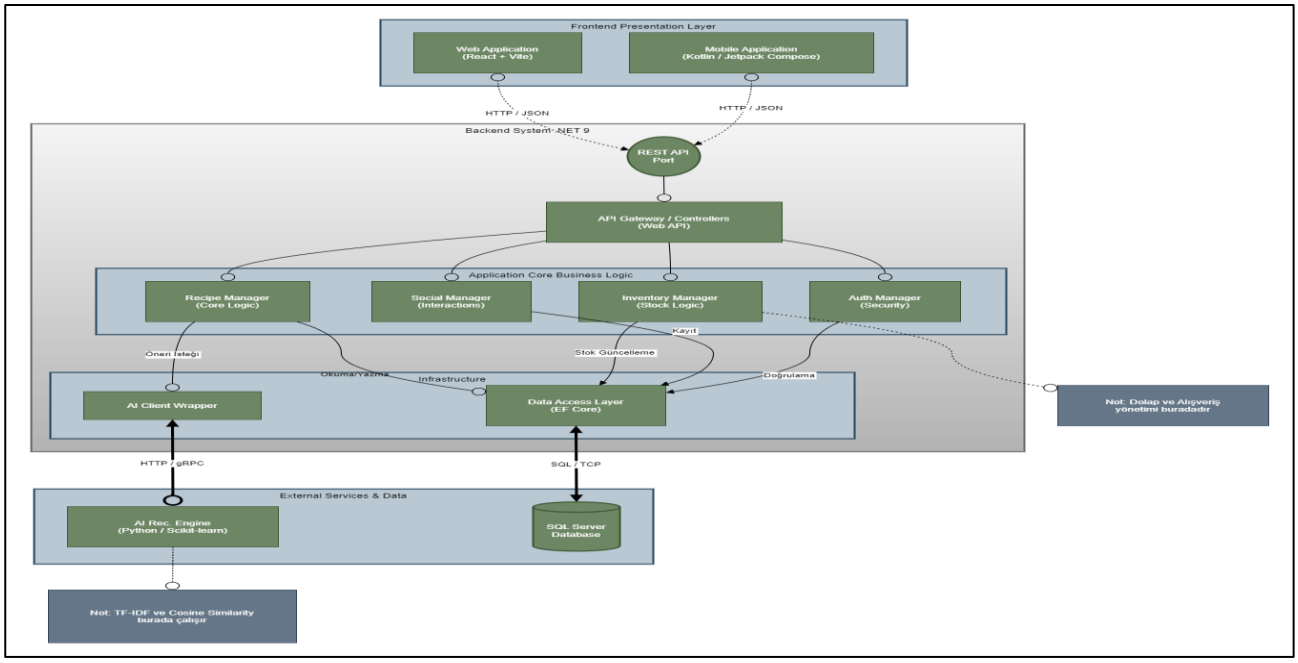
Bu diyagram sayesinde, sistemin genel işleyişi ve veri hareketleri net bir şekilde anlaşılabilir hale gelmiştir.



3.7 Bileşen Diyagramı (Component Diagram):

OleAI sisteminin yazılım bileşenleri arasındaki bağımlılıkları ve etkileşimleri yüksek seviyede göstermek amacıyla Bileşen Diyagramı (Component Diagram) hazırlanmıştır. Bu diyagram, sistemin modüler yapısını ve katmanlar arası sorumluluk dağılımını açık bir şekilde ortaya koymaktadır.

Bileşen diyagramı; istemci uygulamalar, sunucu tarafı servisler, yapay zekâ bileşenleri ve veri erişim katmanları arasındaki ilişkileri görselleştirmektedir. Bu sayede sistemin hangi parçalarının hangi sorumlulukları üstlendiği ve bileşenler arası bağımlılıkların nasıl yönetildiği net biçimde anlaşılabilmektedir.



4. Test Planı:

Bu bölüm, OleAI projesi için hazırlanan Test Planı Belgesini temsil etmektedir. Test sürecinin kapsamı, stratejisi, kullanılan araçlar, test senaryoları, test case'ler ve hata yönetimi yaklaşımları bu başlık altında sunulmaktadır.

4.1 Test Stratejisi:

Bu bölümde, OleAI sisteminin kalite güvencesini sağlamak amacıyla uygulanacak test stratejisi tanımlanmaktadır. Test planı, sistemin hem **tasarım gereksinimlerine uygunluğunu (Verification)** hem de **kullanıcı ihtiyaçlarını karşılayıp karşılamadığını (Validation)** doğrulamayı hedeflemektedir. Uygulama geliştirme sürecine geçilmeden önce belirlenen bu strateji, test faaliyetlerinin sistematik, izlenebilir ve tekrarlanabilir şekilde yürütülmesini amaçlar.

4.1.1 Verification ve Validation Yaklaşımı:

Test süreci, iki temel kalite güvencesi yaklaşımı olan **verification** ve **validation** çerçevesinde ele alınmıştır.

1-) Verification (Doğrulama): Bu aşamada, geliştirilen yazılım bileşenlerinin tasarım dokümanları, mimari kararlar ve gereksinim tanımları ile uyumlu olup olmadığı kontrol edilir. Verification faaliyetleri; unit testler ve integration testler aracılığıyla, özellikle iş mantığı (business logic), servis katmanı ve veri erişim katmanının doğruluğunu ölçmeyi amaçlar.

2-) Validation (Geçerleme): Validation sürecinde ise sistemin, gerçek kullanıcı ihtiyaçlarını ve kullanım senaryolarını karşılayıp karşılamadığı değerlendirilir. System test ve acceptance test aşamalarında, kullanıcı perspektifinden bakılarak sistemin beklenen çıktıları üretip üretmediği test edilir.

Bu iki yaklaşım birlikte ele alınarak hem teknik doğruluk hem de kullanıcı memnuniyeti hedeflenmektedir.

4.1.2 Test Kapsamı:

Bu proje kapsamında gerçekleştirilecek test faaliyetleri aşağıdaki bileşenleri kapsayacaktır:

- Backend katmanındaki iş kuralları ve servis metotları
- RESTful API uç noktaları
- Veritabanı işlemleri (CRUD operasyonları)
- Kullanıcı kimlik doğrulama ve yetkilendirme süreçleri
- Temel kullanıcı akışları (örneğin giriş yapma, veri oluşturma, veri görüntüleme)

4.1.3 Başarı Kriterleri:

Test sürecinin başarılı sayılabilmesi için aşağıdaki kriterlerin sağlanması hedeflenmektedir:

- Kritik fonksiyonlara ait test senaryolarının %100 başarıyla geçmesi
- Yüksek öncelikli (critical ve high) hataların giderilmiş olması
- Gereksinim matrisi ile test senaryoları arasında izlenebilirliğin sağlanması

4.2 Test Türleri:

OleAI projesinde test faaliyetleri; sistemin çok katmanlı mimarisi (web, mobil, backend ve yapay zekâ modülü) dikkate alınarak planlanmıştır. Test süreci, yazılım geliştirme yaşam döngüsü (SDLC) ile uyumlu olacak şekilde,

artan karmaşıklık seviyelerinde uygulanacaktır. Her test türü, sistemin belirli bir bileşenini veya işlevini doğrulamayı hedeflemektedir.

4.2.1 Unit Test (Birim Testler):

Amaç: Unit testlerin amacı, OleAI sistemini oluşturan en küçük test edilebilir bileşenlerin (metotlar ve sınıflar) doğru çalışıp çalışmadığını doğrulamaktır. Bu testler özellikle iş kurallarının ve algoritmik işlemlerin doğruluğunu güvence altına almayı hedefler.

Kapsam:

- Backend tarafında:
  - Tarif yönetimi iş kuralları
  - Kullanıcı profili ve tercih yönetimi
  - Malzeme (Dolabım) modülündeki stok düşme ve alışveriş listesi mantığı
  - JWT tabanlı kimlik doğrulama işlemleri
- Yapay zekâ modülünde:
  - TF-IDF vektör oluşturma fonksiyonları
  - Kosinüs benzerliği hesaplamaları
  - Negatif örnekleme ve zaman sönümlemesi fonksiyonları

Yaklaşımlar:

- Unit testler, altyapı bağımlılıklarından (veri tabanı, API çağrıları) izole edilerek yazılacaktır.
- Mock nesneler kullanılarak yalnızca test edilen fonksiyonun davranışı ölçülecektir.
- Her test, tek bir işlevi doğrulayacak şekilde tasarlanacaktır.

Kullanılacak Araçlar:

- Backend (.NET): xUnit / NUnit
- Mocklama: Moq
- Yapay zekâ modülü (Python): PyTest

4.2.2 Integration Test (Entegrasyon Test):

Amaç: Integration testlerin amacı, OleAI sisteminde yer alan farklı katmanların ve modüllerin birbiriyle doğru şekilde etkileşim kurduğunu doğrulamaktır. Bu testler, veri akışının ve entegrasyonların tutarlılığını kontrol eder.

Kapsam:

- Backend API'leri ile veritabanı arasındaki etkileşimler
- Kullanıcı işlemleri (tarif ekleme, yorum yapma, beğenme)
- Kimlik doğrulama ve yetkilendirme süreçlerinin API seviyesinde doğrulanması
- Backend ile yapay zekâ öneri servisleri arasındaki veri alışverişi

Yaklaşım:

- Testler, gerçek konfigürasyona yakın bir test ortamında gerçekleştirilecektir.
- API uç noktalarına HTTP istekleri(POST, GET vb.) gönderilerek yanıtlar doğrulanacaktır.
- Test veritabanı, üretimdeki ortamdan tamamen izole olacaktır.

Kullanılacak Araçlar:

- Postman / Swagger
- Test amaçlı SQL Server instance

4.2.3 System Test (Sistem Testi):

Amaç: System testlerin amacı, OleAI sisteminin tüm bileşenleriyle birlikte uçtan uca (end-to-end) doğru çalıştığını doğrulamaktır. Bu testler, sistemin fonksiyonel gereksinimleri bir bütün olarak karşılayıp karşılamadığını ölçer.

Kapsam:

- Web ve mobil uygulama üzerinden kullanıcı giriş süreçleri
- Tarif arama ve yapay zekâ destekli öneri alma akışı
- "Dolabım" modülü üzerinden malzeme ekleme, eksiltme ve alışveriş listesi oluşturma
- Sosyal etkileşimler (yorum, beğeni, favori ekleme) vb.

Yaklaşım:

- Gerçek kullanıcı senaryoları temel alınacaktır.
- Hem pozitif hem de negatif senaryolar test edilecektir.
- Sistem, Docker container'ları üzerinde ayağa kaldırılarak test edilecektir.

Kullanılacak Araçlar:

- Postman ve Swagger (API bazlı sistem testleri)
- Selenium (Web UI testleri – React)
- Manuel testler (Mobil uygulama – Jetpack Compose)

4.2.4 Acceptance Test (Kabul Testi):

Amaç: Acceptance testlerin amacı, OleAI sisteminin gereksinim analizinde tanımlanan kullanıcı ihtiyaçlarını ve use case'leri eksiksiz şekilde karşılayıp karşılamadığını doğrulamaktır. Bu testler, sistemin teslim edilebilirliğini belirleyen son aşamadır.

Kapsam:

- Kişiselleştirilmiş tarif önerilerinin doğruluğu
- Malzeme bazlı öneri mekanizmasının çalışması
- Kullanıcıların tarif ekleyebilmesi ve topluluk etkileşimi
- Gıda israfını azaltmaya yönelik "Dolabım" ve raf ömrü bildirimleri vb.

Yaklaşım:

- Test senaryoları, gereksinim matrisi ve use case'ler ile birebir eşleştirilecektir.
- Kullanıcı bakış açısıyla değerlendirme yapılacaktır.
- Kritik acceptance testlerin tamamı başarılı olmadan sistem kabul edilmeyecektir.

Başarı Kriterleri:

- Tüm kritik acceptance testlerin başarıyla geçmesi
- Yüksek öncelikli hata bulunmaması

4.3 Test Ortamı ve Araçları:

OleAI projesinde test faaliyetlerinin güvenilir, tekrarlanabilir ve üretim ortamına yakın koşullarda gerçekleştirilebilmesi amacıyla ayrı bir test ortamı tanımlanmıştır. Test ortamı; backend, frontend (web ve mobil), yapay zekâ modülü ve veri tabanı bileşenlerini kapsayacak şekilde yapılandırılmıştır. Tüm testler, üretim ortamından izole edilmiş test verileri üzerinde yürütülecektir.

Test sürecinde kullanılan araçlar; proje mimarisi, kullanılan teknolojiler ve sektörde yaygın kabul gören test pratikleri dikkate alınarak seçilmiştir. Bu araçlar hem manuel hem de otomatik testlerin etkin bir şekilde yürütülmesini desteklemektedir.

4.3.1 Test Ortamının Yapılandırılması:

Test ortamı aşağıdaki temel prensiplere göre yapılandırılacaktır:

- Test ortamı, üretim ortamından tamamen izole olacaktır.
- Veritabanı, yalnızca test amaçlı oluşturulmuş örnek (dummy) veriler içerecektir.
- Backend, frontend ve yapay zekâ servisleri Docker container'ları üzerinde çalıştırılacaktır.
- Test ortamı, CI/CD sürecine entegre edilebilir olacak şekilde tasarlanacaktır.

Bu yaklaşım sayesinde, test sonuçlarının tutarlılığı ve güvenilirliği sağlanacaktır.

4.3.2 Kullanılan Test Araçları:

Aşağıdaki tabloda OleAI projesinde kullanılacak test araçları ve kullanım amaçları özetlenmiştir:

Katman	Araç / Teknoloji	Kullanım Amacı
Backend Test	xUnit / NUnit	.NET tabanlı servislerin birim (unit) ve entegrasyon testleri.
Mocklama	Moq	Bağımlılıkların izole edilmesi ve sahte nesneler oluşturulması.
API Test	Postman / Swagger	REST API uç noktalarının manuel ve otomatik testleri.
Veritabanı	SQL Server (Test DB)	CRUD işlemleri ve veri bütünlüğü testleri.
Yapay Zekâ Test	PyTest	TF-IDF, kosinüs benzerliği ve dinamik profil algoritmalarının test edilmesi.
Web UI Test	Selenium	React tabanlı web arayüzünün uçtan uca (E2E) testleri.
Mobil Test	Manuel Testler	Jetpack Compose tabanlı mobil arayüzün fonksiyonel testleri.
Containerization	Docker / Docker Compose	Test ortamının standart ve taşınabilir hâle getirilmesi.
CI/CD	GitHub Actions	Testlerin otomatik çalıştırılması ve raporlanması.
Kod Kalitesi	SonarQube	Test kapsamı, kod kalitesi ve teknik borç analizi.

4.3.3 Test Verisi Yönetimi:

Test sürecinde kullanılacak veriler aşağıdaki prensiplere uygun olarak yönetilecektir:

- Gerçek kullanıcı verileri test ortamında kullanılmayacaktır.
- Tüm test verileri anonim ve sentetik olarak oluşturulacaktır.
- Yapay zekâ modülünde kullanılan tarif ve malzeme verileri, öneri algoritmalarını temsil edecek örnek veri setlerinden oluşacaktır.
- Testler tamamlandıktan sonra test verileri silinecek veya sıfırlanacaktır.

4.3.4 Otomasyon ve CI/CD Entegrasyonu:

OleAI projesinde test süreçlerinin sürdürülebilirliğini artırmak amacıyla otomasyon yaklaşımı benimsenmiştir. Bu kapsamda:

- Her kod değişikliği sonrasında unit ve integration testler otomatik olarak çalıştırılacaktır.
- Test başarısızlıkları, CI/CD hattı üzerinden raporlanacaktır.

- Kritik testlerin başarısız olması durumunda, ilgili sürümün ilerlemesi engellenecektir.

Bu yaklaşım sayesinde, yazılım kalitesi sürekli olarak izlenecek ve hatalar erken aşamada tespit edilecektir.

4.4 Test Senaryoları:

Bu bölümde, OleAI sistemi için tanımlanan temel use case'ler doğrultusunda oluşturulan test senaryoları sunulmaktadır. Test senaryoları; sistemin fonksiyonel gereksinimlerini karşılayıp karşılamadığını doğrulamak amacıyla hazırlanmış olup, her use case için hem pozitif hem de negatif senaryolar içermektedir.

Test senaryoları, kullanıcı bakış açısı esas alınarak oluşturulmuş ve gereksinim analizinde tanımlanan fonksiyonlar ile birebir eşleştirilmiştir.

4.4.1 Kullanıcı Kayıt ve Giriş Test Senaryoları:

Use Case: Kullanıcı sisteme kaydolur ve giriş yapar.

Senaryo Türü	Senaryo Detayları
Pozitif Senaryolar	• Kullanıcı geçerli e-posta ve şifre bilgileri ile kaydolur, e-posta adresini doğrular. • Kayıtlı kullanıcı doğru kimlik bilgileriyle sisteme başarılı şekilde giriş yapar. • JWT token başarıyla oluşturulur ve istemciye iletilir.
Negatif Senaryolar	• Kullanıcı, sistemde kayıtlı olmayan bir e-posta ile giriş yapmaya çalışır. • Kullanıcı yanlış şifre girer. • Zorunlu alanlar (e-posta, şifre) boş bırakılır. • Süresi dolmuş veya geçersiz JWT token ile korumalı endpoint'e erişilmeye çalışılır.

4.4.2 Tarif Ekleme ve Yönetimi Test Senaryoları:

Use Case: Kullanıcı sisteme tarif ekler ve tariflerini yönetir.

Senaryo Türü	Senaryo Detayları
Pozitif Senaryolar	• Kullanıcı yeni bir tarifi başarıyla ekler. • Tarif public veya private olarak işaretlenebilir. • Kullanıcı kendi tariflerini listeleyebilir ve düzenleyebilir.
Negatif Senaryolar	• Zorunlu alanları (tarif adı, malzeme listesi) eksik olan tarif eklenmeye çalışılır. • Yetkisiz bir kullanıcı, başka bir kullanıcıya ait private tarifi düzenlemeye çalışır. • Geçersiz veri formatı ile tarif ekleme isteği gönderilir.

4.4.3 Yapay Zekâ Destekli Tarif Öneri Test Senaryoları

Use Case: Kullanıcıya kişisel tercihlerine ve mevcut malzemelere göre tarif önerilir.

Senaryo Türü	Senaryo Detayları
Pozitif Senaryolar	• Kullanıcının diyet ve beslenme tercihleri dikkate alınarak öneri sunulur. • Kullanıcının "Dolabım" modülündeki malzemelere uygun tarifler önerilir. • Öneri süresi belirlenen performans sınırları içinde tamamlanır.

Negatif Senaryolar	• Kullanıcının dolabında hiç malzeme bulunmaması. • Kullanıcı profil verilerinin eksik olması (soğuk başlangıç durumu). • Yapay zekâ servisinden geçersiz veya boş yanıt alınması.

4.4.4 “Dolabım” Modülü ve Malzeme Yönetimi Test Senaryoları

Use Case: Kullanıcı malzemelerini yönetir ve stok takibi yapar.

Senaryo Türü	Senaryo Detayları
Pozitif Senaryolar	• Kullanıcı dolabına yeni malzeme ekler. • Bir tarif yapıldığında, ilgili malzemelerin miktarı otomatik olarak düşer. • Tükenen malzemeler alışveriş listesine otomatik olarak eklenir.
Negatif Senaryolar	• Negatif miktarda malzeme eklenmeye çalışılır. • Mevcut olmayan bir malzeme güncellenmeye çalışılır. • Yetkisiz erişim ile başka kullanıcıya ait dolap verilerine ulaşılmaya çalışılır.

4.4.5 Sosyal Etkileşim Test Senaryoları:

Use Case: Kullanıcılar tarifler üzerinde etkileşim kurar.

Senaryo Türü	Senaryo Detayları
Pozitif Senaryolar	• Kullanıcı bir tarife yorum yapar. • Kullanıcı bir tarifi beğenir, beğenmez veya favorilere ekler. • Yorumlar ilgili tarif altında doğru şekilde listelenir.
Negatif Senaryolar	• Boş veya çok uzun yorum gönderilmeye çalışılır. • Yetkisiz kullanıcı yorum yapmaya çalışır. • Silinmiş bir tarife etkileşim yapılmaya çalışılır.

4.4.6 Performans ve Güvenlik Test Senaryoları:

Use Case: Sistem performans ve güvenlik gereksinimlerini karşılar.

Senaryo Türü	Senaryo Detayları
Pozitif Senaryolar	• Aynı anda çok sayıda kullanıcının öneri alabilmesi. • Şifrelerin güvenli şekilde hash’lenerek saklanması. • Yetkilendirme kontrollerinin doğru çalışması.
Negatif Senaryolar	• Yetkisiz erişim denemeleri. • Aşırı istek (rate limit) durumları. • Hatalı veya manipüle edilmiş istek payload’ları.

4.4.7 Yapay Zekâ Modeli Doğrulama Test Senaryoları:

Use Case: Yapay zekâ modelinin doğruluğu ve tutarlılığı.

Senaryo Türü	Senaryo Detayları
Doğrulama	• Kullanıcının beğeni geçmişine göre önerilerin zamanla kişiselleşmesi. • Beğenilmeyen tarif türlerinin önerilerden uzaklaştırılması. • Yeni kullanıcılar (cold-start) için genel ve dengeli önerilerin sunulması. • Kullanıcı malzeme listesi değiştiğinde öneri sonuçlarının güncellenmesi.

4.4.8 Sınır Durumları (Edge Case) Test Senaryoları

Use Case: Sistemin uç koşullardaki davranışı.

Senaryo Türü	Senaryo Detayları
Sınır Durumları	• Kullanıcının dolabında yalnızca tek bir malzeme bulunması. • Çok büyük veri setleri ile tarif önerisi alınması. • Aynı tarife kısa sürede tekrar tekrar etkileşim yapılması. • Uzun süre kullanılmayan kullanıcı profilleri.

4.4.9 Yetkilendirme ve Rol Bazlı Test Senaryoları

Use Case: Rol ve yetki kontrolleri.

Senaryo Türü	Senaryo Detayları
Yetkilendirme	• Kullanıcının yalnızca kendi private tariflerini düzenleyebilmesi. • Public tariflerin tüm kullanıcılar tarafından görüntülenebilmesi. • Token süresi dolduğunda erişimin engellenmesi. • Yetkisiz kullanıcıların korumalı servis uç noktalarına erişememesi.

4.5 Test Case Tablosu

Bu bölümde, 4.4 Test Senaryoları başlığı altında tanımlanan use case'lere karşılık gelen test case'ler sunulmaktadır. Test case'ler; sistemin fonksiyonel ve fonksiyonel olmayan gereksinimlerini doğrulamak amacıyla hazırlanmıştır.

Her bir test case; girdi (input), beklenen çıktı (expected output), gerçekleşen çıktı (actual output) ve test durumu (status) bilgilerini içerecek şekilde tanımlanmıştır. Gerçekleşen çıktı ve durum alanları, uygulama ve test aşamasında doldurulacaktır.

Bu bölüme örnek olması adına sadece belirli test case tabloları eklenmiştir. Diğer test case tablolarına gitHub reposu üzerinden ulaşılabilir.

4.5.1 Yapay Zekâ Destekli Tarif Önerisi Test Case'leri:

ID	Test Adı	Girdi (Input)	Beklenen Çıktı	Gerçekleşen Çıktı	Durum
TC-12	Malzemeye göre öneri	Dolap verisi (Mevcut malzemeler)	Malzemelere uygun tarif listesi	—	—
TC-13	Diyet tercihine göre öneri	Kullanıcı tercihi (Örn: Vejetaryen)	Tercihe uyumlu (et içermeyen) tarifler	—	—

TC-14	Cold-start kullanıcı	Profil veya geçmiş veri yok	Popüler/Genel tarif önerileri	—	—
TC-15	Boş malzeme listesi	Malzeme girişi yok	Kullanıcıyı yönlendiren mesaj	—	—
TC-16	AI servisi boş cevap	AI servisinden Null/Timeout yanıtı	Sistem çökmemeli, kontrollü hata mesajı	—	—

4.5.2 Performans ve Güvenlik Test Case'leri:

ID	Test Adı	Girdi (Input)	Beklenen Çıktı	Gerçekleşen Çıktı	Durum
TC-26	Eşzamanlı kullanıcı yükü	100 eşzamanlı istek (Concurrent Request)	Sistem yanıt verir, çökme veya timeout olmaz	—	—
TC-27	Yük altında AI öneri	Aynı anda çoklu öneri isteği	Yanıt süresi kabul edilebilir seviyede kalır (Örn: <3sn)	—	—
TC-28	Şifre güvenliği	Veritabanı kontrolü (Plain password girişi)	Şifre hash'lenerek (Argon2/BCrypt) saklanır	—	—
TC-29	Yetkisiz endpoint erişimi	Token olmadan korumalı kaynağa istek	401 Unauthorized hatası döner	—	—
TC-30	Rate limit ihlali	Belirlenen sürede çok sık istek	İstekler sınırlandırılır (429 Too Many Requests)	—	—
TC-31	Manipüle edilmiş payload	Bozuk veya zararlı JSON verisi	İstek reddedilir (400 Bad Request)	—	—

4.5.3 Yapay Zekâ Modeli Doğrulama Test Case'leri:

ID	Test Adı	Girdi (Input)	Beklenen Çıktı	Gerçekleşen Çıktı	Durum
TC-32	Kullanıcı öğrenme etkisi	Tekrarlı beğeni geçmişi	Öneriler kişiselleşir ve kullanıcının zevkine yaklaşır	—	—
TC-33	Negatif geri bildirim etkisi	Beğenilmeyen tarifler	Benzer tariflerin önerilme sıklığı azalır	—	—
TC-34	Cold-start önerisi	Yeni kullanıcı (Geçmiş veri yok)	Dengeli, popüler ve genel öneriler sunulur	—	—
TC-35	Malzeme değişim etkisi	Güncellenmiş dolap envanteri	Öneriler yeni malzemelere göre anında güncellenir	—	—
TC-36	Profil tercihi etkisi	Diyet tercihi (Örn: Glütensiz)	Sadece tercihe uyumlu tarifler listelenir	—	—
TC-37	Zaman etkisi (time decay)	Eski etkileşimler vs. Yeni etkileşimler	Güncel tercihler daha yüksek ağırlıkla değerlendirilir	—	—

4.6 Hata Yönetimi ve Bug Tracking Planı:

Bu bölümde, OleAI projesinde test süreci sırasında tespit edilen hataların (bug) nasıl raporlanacağı, sınıflandırılacağı, takip edileceği ve çözüme ulaştırılacağı açıklanmaktadır. Amaç; yazılım kalitesini artırmak, hataların tekrar oluşmasını önlemek ve geliştirme sürecinde şeffaf bir hata yönetimi sağlamaktır.

4.6.1 Hata Yönetimi Yaklaşımı:

OleAI projesinde hata yönetimi süreci aşağıdaki adımlardan oluşmaktadır:

1. **Hatanın Tespiti:** Unit test, integration test, sistem testleri veya manuel testler sırasında hata tespit edilir.
2. **Hata Kaydı (Bug Report):** Tespit edilen hata, sistem üzerinde kayıt altına alınır.
3. **Önceliklendirme ve Sınıflandırma:** Hatalar, etki alanına ve ciddiyetine göre sınıflandırılır.
4. **Düzeltme (Fix):** Geliştirici, hatayı giderir ve gerekli kod güncellemelerini yapar.
5. **Doğrulama (Re-test):** Hata giderildikten sonra ilgili test case tekrar çalıştırılır.
6. **Kapanış (Close):** Hata çözüldüyse kapatılır, aksi durumda yeniden açılır.

Bu döngü sayesinde, hataların kontrolsüz şekilde birikmesi önlenir ve kalite sürekli olarak izlenir.

4.6.2 Hata Türleri ve Sınıflandırma:

Tespit edilen hatalar aşağıdaki kategorilere ayrılacaktır:

- **Fonksiyonel Hatalar:** Sistem fonksiyonlarının gereksinimlere uygun çalışmaması
- **Güvenlik Hataları:** Yetkisiz erişim, token doğrulama sorunları, veri güvenliği açıkları
- **Performans Hataları:** Yük altında yavaşlama, timeout, gecikme problemleri
- **Yapay Zekâ Hataları:** Anlamsız, tutarsız veya kullanıcı profiline aykırı öneriler
- **UI/UX Hataları:** Arayüzde hatalı veya beklenmeyen kullanıcı davranışları

4.6.3 Hata Öncelik Seviyeleri:

Hatalar, sistem üzerindeki etkilerine göre aşağıdaki öncelik seviyelerine ayrılacaktır:

Öncelik	Açıklama
Kritik (Critical)	Sistem çalışamaz durumda (Showstopper).
Yüksek (High)	Ana fonksiyonlar etkileniyor, iş akışı bozuluyor.
Orta (Medium)	Hata var ancak alternatif çözüm (Workaround) mevcut.
Düşük (Low)	Kullanıcı deneyimini sınırlı etkiler (Kozmetik hatalar vb.).

Bu önceliklendirme sayesinde, kritik hatalar öncelikli olarak ele alınacaktır.

4.6.4 Bug Tracking Aracı ve Kullanım

OleAI projesinde hata takibi için **GitHub Issues** kullanılacaktır. GitHub Issues, proje deposu ile entegre çalışması ve şeffaflık sağlaması nedeniyle tercih edilmiştir.

Her bir bug kaydı aşağıdaki bilgileri içerecektir:

- Bug ID
- Hata başlığı
- Hata açıklaması
- Tekrarlanabilirlik adımları
- Beklenen davranış
- Gerçekleşen davranış
- Öncelik seviyesi
- İlgili test case ID
- Durum (Open, In Progress, Resolved, Closed)

4.6.5 Hata–Test Case İlişkilendirmesi

Tespit edilen her hata, ilgili **test case ID** ile ilişkilendirilecektir. Bu yaklaşım sayesinde:

- Hatanın hangi senaryoda ortaya çıktığı netleşir
- Aynı hatanın tekrar oluşması önlenir
- Test–geliştirme süreci izlenebilir hâle gelir

Bu yapı, yazılım test süreçlerinde kullanılan **traceability (izlenebilirlik)** prensibini desteklemektedir.

5. DevOps Entegrasyonu:

5.1 DevOps Yaklaşımı ve Projedeki Rolü:

DevOps, yazılım geliştirme (Development) ve operasyon (Operations) süreçlerini entegre ederek; yazılımın daha hızlı, güvenilir ve sürdürülebilir bir şekilde geliştirilmesini amaçlayan modern bir yaklaşımdır. Bu yaklaşım, sürekli entegrasyon (CI), sürekli dağıtım (CD), otomasyon ve izlenebilirlik gibi prensipleri temel alır.

OleAI projesinde DevOps yaklaşımının benimsenmesinin temel amacı; geliştirilen yazılımın yalnızca çalışır olması değil, aynı zamanda tekrar üretilebilir, ölçeklenebilir ve bakımı kolay bir yapıya sahip olmasını sağlamaktır. Geliştirme sürecinde yapılan her kod değişikliğinin kontrollü bir şekilde test edilmesi, paketlenmesi ve dağıtılması hedeflenmektedir.

Güz dönemi kapsamında DevOps süreci, tasarım ve planlama seviyesinde ele alınacaktır. Bu aşamada; CI/CD pipeline yapısı tanımlanacak, container tabanlı mimari planlanacak ve otomasyon, güvenlik, monitoring gibi süreçlerin sistem mimarisi içerisindeki yeri belirlenecektir. Bahar döneminde ise bu plan doğrultusunda DevOps bileşenleri uygulamaya alınacak ve gerçek bir CI/CD süreci hayata geçirilecektir.

Bu yaklaşım sayesinde OleAI projesi; manuel işlemlere bağımlılığı azaltan, hata oranını düşüren ve profesyonel yazılım geliştirme pratiklerine uygun bir yapıya kavuşacaktır.

5.2 DevOps Stratejisi (CI/CD Yaklaşımı):

OleAI projesinde DevOps stratejisi; yazılım geliştirme sürecinin başından dağıtıma kadar olan tüm aşamaları kapsayacak şekilde Sürekli Entegrasyon (CI) ve Sürekli Dağıtım (CD) prensipleri temel alınarak tasarlanmıştır. Bu stratejinin temel amacı, geliştirilen kodun her değişiklikte otomatik olarak doğrulanması, kalite kontrollerinden geçirilmesi ve dağıtıma hazır hâle getirilmesidir.

Sürekli Entegrasyon (CI) sürecinde, geliştiriciler tarafından GitHub reposuna gönderilen her kod değişikliği otomatik olarak bir pipeline'ı tetikleyecektir. Bu aşamada; uygulama derlenecek, birim testleri çalıştırılacak ve

kod kalitesi analizleri gerçekleştirilecektir. Kod kalitesi ve güvenlik analizleri için SonarQube kullanılması planlanmaktadır. SonarQube sayesinde kod içerisinde yer alan potansiyel hatalar, güvenlik açıkları ve standart dışı yapılar erken aşamada tespit edilebilecektir.

Sürekli Dağıtım (CD) süreci ise CI aşamasını başarıyla geçen sürümlerin çalıştırılabilir hâle getirilmesini ve ilgili ortamlara otomatik olarak dağıtılmasını kapsar. Testleri ve kalite kontrollerini geçen uygulama bileşenleri Docker container'ları olarak paketlenecek; bu container'lar test veya canlı ortamda çalışan sunuculara dağıtılacaktır.

Uygulamanın çalışacağı altyapı için VPS (Virtual Private Server) tabanlı bir sunucu ortamı planlanmaktadır. Bu sunucu üzerinde container'lar izole bir şekilde çalıştırılacak; böylece uygulama bileşenleri arasında bağımlılık problemleri ve ortam farklılıkları en aza indirilecektir. İlerleyen aşamalarda, container yönetimi ve ölçeklenebilirlik ihtiyaçları doğrultusunda Kubernetes altyapısına geçiş planlanmaktadır.

Güz dönemi kapsamında bu CI/CD stratejisi tasarım ve mimari düzeyde ele alınacak; pipeline adımları, kullanılacak araçlar ve ortam yapısı detaylı şekilde planlanacaktır. Bahar döneminde ise bu plan doğrultusunda gerçek bir CI/CD pipeline'ı kurulacak ve aktif olarak uygulanacaktır.

5.3 CI/CD Pipeline Akışı:

OleAI projesi için tasarlanan CI/CD pipeline akışı, yazılımın geliştirme sürecinden dağıtımına kadar olan adımların otomatik ve kontrollü bir şekilde yönetilmesini hedeflemektedir. Bu akış sayesinde kod değişiklikleri sistematik olarak test edilmekte, kalite kontrollerinden geçirilmekte ve dağıtıma hazır hâle getirilmektedir.

Planlanan CI/CD pipeline süreci aşağıdaki aşamalardan oluşmaktadır:

Aşama	Açıklama	Kullanılan Araç / Teknoloji
Commit	Kod değişikliklerinin GitHub reposuna gönderilmesi.	Git / GitHub
CI Tetikleme	Commit işlemi sonrası pipeline'ın otomatik başlatılması.	GitHub Actions
Build	Uygulamanın derlenmesi ve bağımlılıkların kontrol edilmesi.	.NET Build Tools
Test	Birim testlerin (Unit Tests) otomatik olarak çalıştırılması.	xUnit
Kod Kalitesi & Güvenlik	Kodun kalite ve güvenlik açısından statik analizi.	SonarQube
Containerization	Uygulamanın Docker image hâline getirilmesi.	Docker
Deploy	Image'ın sunucu ortamına dağıtılması ve çalıştırılması.	VPS / Docker Runtime
Monitoring & Logging	Uygulamanın çalışma durumunun ve logların izlenmesi.	Prometheus, ELK Stack

Bu yapı ile birlikte, yazılımın her sürümü aynı standartlara göre değerlendirilmekte ve dağıtım süreci insan hatasına kapalı hâle getirilmektedir. CI/CD pipeline akışı, bir sonraki aşamada hazırlanacak olan DevOps entegrasyon diyagramının temelini oluşturmaktadır.

5.4 Containerization ve Orchestration Planı:

OleAI projesinde, yazılım bileşenlerinin dağıtım, yönetim ve ölçeklenebilirlik süreçlerini standartlaştırmak amacıyla **container tabanlı bir mimari** benimsenmiştir. Bu yaklaşım, modern ve kurumsal yazılım projelerinde yaygın olarak tercih edilen; uygulamaların tutarlı, taşınabilir ve sürdürülebilir bir şekilde çalışmasını sağlayan temel DevOps pratiklerinden biridir.

Sistem; backend servisleri, frontend uygulaması ve yapay zekâ modülleri olmak üzere birbirinden bağımsız çalışan bileşenlerden oluşmaktadır. Bu bileşenlerin her biri **Docker container**'ları içerisinde izole edilerek çalıştırılacak şekilde tasarlanmıştır. Containerization sayesinde her servis kendi bağımlılıkları, çalışma ortamı ve konfigürasyonları ile birlikte paketlenmekte; bu durum hem servisler arası bağımlılıkları azaltmakta hem de bakım ve güncelleme süreçlerini kolaylaştırmaktadır.

Container tabanlı yapı, uygulamanın farklı ortamlar (test, staging, production) arasında tutarlı şekilde çalışmasını garanti altına alırken, sürüm geçişlerinin kontrollü ve geri alınabilir (rollback) biçimde yapılmasına da olanak tanımaktadır. Böylece dağıtım süreçleri manuel işlemlerden arındırılmış, tekrarlanabilir ve izlenebilir bir hâle getirilmektedir.

Dağıtım altyapısı olarak, container'ların merkezi bir sunucu altyapısı (VPS veya bulut tabanlı ortam) üzerinde çalıştırılması planlanmaktadır. Bu altyapı; servislerin merkezi olarak yönetilmesini, izlenmesini ve gerektiğinde hızlı müdahale edilmesini mümkün kılmaktadır. Ayrıca container tabanlı dağıtım modeli, servislerin bağımsız olarak güncellenebilmesine ve sistemin tamamını etkilemeden yeni sürümlerin devreye alınmasına imkân sağlamaktadır.

Sistemin ölçeklenebilirliği ve operasyonel yönetiminin daha etkin hâle getirilmesi amacıyla **Kubernetes** tabanlı bir orchestration katmanı tasarlanmıştır. Kubernetes; container'ların yaşam döngüsü yönetimi, servis keşfi, yük dengeleme ve otomatik ölçeklendirme gibi kritik operasyonel sorumlulukları üstlenerek sistemin dinamik ihtiyaçlara uyum sağlamasını mümkün kılacaktır. Bu yapı sayesinde OleAI, artan kullanıcı yükü ve servis taleplerine karşı esnek ve dayanıklı bir mimariye sahip olacaktır.

Güz dönemi kapsamında containerization ve orchestration yaklaşımı **mimari tasarım ve planlama düzeyinde** ele alınacak; servislerin container yapısı, dağıtım senaryoları ve ölçeklenme stratejileri detaylı şekilde tanımlanacaktır. Bahar döneminde ise bu mimari doğrultusunda Docker ve Kubernetes altyapısı uygulamaya alınarak sistemin operasyonel hâle getirilmesi hedeflenmektedir.

5.5 Otomasyon Planı (Build – Test – Deploy):

OleAI projesinde yazılım geliştirme ve dağıtım süreçlerinin güvenilir, izlenebilir ve sürdürülebilir bir şekilde yönetilebilmesi amacıyla uçtan uca otomasyon yaklaşımı benimsenmiştir. Bu kapsamda Continuous Integration (CI) ve Continuous Deployment (CD) prensipleri doğrultusunda tasarlanan pipeline, yalnızca derleme ve test süreçlerini değil, aynı zamanda kod kalitesi ve güvenlik kontrollerini de içerecek şekilde yapılandırılmıştır.

5.5.1 Build Otomasyonu:

Kod deposuna yapılan her commit işlemi sonrasında CI pipeline otomatik olarak tetiklenecektir. Bu aşamada uygulamanın kaynak kodu derlenerek bağımlılık kontrolleri gerçekleştirilecek ve olası derleme hataları erken safhada tespit edilecektir. Build sürecinin otomatikleştirilmesi, farklı geliştiriciler tarafından yapılan değişikliklerin tek bir standart üzerinden doğrulanmasını sağlamaktadır.

5.5.2 Test Otomasyonu:

Build aşamasının başarıyla tamamlanmasının ardından, uygulamaya ait birim testler otomatik olarak çalıştırılmaktadır. Bu testler, yazılımın temel işlevlerinin beklenen şekilde çalıştığını doğrulamayı amaçlamaktadır. Test sonuçları pipeline kapsamında raporlanmakta ve başarısız testler durumunda dağıtım süreci otomatik olarak durdurulmaktadır.

5.5.3 Kod Kalitesi ve Güvenlik Analizi (SonarQube):

Test aşamasının ardından, kod tabanı SonarQube üzerinden statik kod analizi sürecine tabi tutulmaktadır. Bu analiz kapsamında kod tekrarları, karmaşıklık seviyeleri, potansiyel hatalar ve güvenlik açıkları otomatik olarak tespit edilmektedir. SonarQube tarafından tanımlanan Quality Gate kriterlerinin sağlanamaması durumunda pipeline durdurulmakta ve ilgili sürümün dağıtıma ilerlemesine izin verilmemektedir. Bu yaklaşım, kod kalitesinin sürekli ve ölçülebilir bir şekilde korunmasını sağlamaktadır.

5.5.4 Deploy Otomasyonu

Kod kalitesi ve güvenlik kontrollerinden başarıyla geçen sürümler, otomatik olarak Docker image hâline getirilmekte ve belirlenen dağıtım ortamına aktarılmaktadır. Deploy süreci, manuel müdahaleye ihtiyaç duyulmadan gerçekleştirilecek şekilde tasarlanmıştır. Bu sayede sürüm geçişleri kontrollü, izlenebilir ve geri alınabilir (rollback) bir yapıda yönestilmektedir.

5.5.5 Monitoring ve Logging

Otomasyon sürecinin tamamlayıcı bir parçası olarak, uygulamanın çalışma durumu, performansı ve hata kayıtları merkezi bir yapı üzerinden izlenmektedir. Monitoring ve logging mekanizmaları sayesinde sistem davranışları sürekli olarak gözlemlenebilmekte ve olası problemlere hızlı bir şekilde müdahale edilebilmektedir.

Bu otomasyon planı ile OleAI projesinde yazılım geliştirme süreci yalnızca hızlandırılmakla kalmamakta; aynı zamanda kalite, güvenlik ve operasyonel sürdürülebilirlik açısından kurumsal standartlara uygun bir yapıya kavuşturulmaktadır.

5.6 Güvenlik Entegrasyonu (DevSecOps):

OleAI projesinde güvenlik, yazılım geliştirme sürecinin ayrı bir aşaması olarak değil; tüm yazılım yaşam döngüsüne entegre edilmiş sürekli bir süreç olarak ele alınmıştır. Bu doğrultuda projede DevSecOps yaklaşımı benimsenmiş ve güvenlik kontrolleri, Continuous Integration ve Continuous Deployment (CI/CD) süreçlerinin ayrılmaz bir parçası hâline getirilmiştir. Amaç, güvenlik risklerini dağıtım sonrasına bırakmak yerine, erken aşamalarda tespit ederek etkisini en aza indirmektir.

CI sürecinin önemli bir bileşeni olarak, kod tabanı her değişiklik sonrasında statik kod analizi sürecine tabi tutulmaktadır. Bu analizler, SonarQube platformu üzerinden gerçekleştirilmekte; potansiyel güvenlik açıkları, hatalı kod kullanımları, kod tekrarları ve bakım maliyetini artırabilecek yapılar otomatik olarak tespit edilmektedir. Analiz sonuçları doğrultusunda tanımlanan Quality Gate kuralları, kodun belirli güvenlik ve kalite eşiklerini karşılamasını zorunlu kılmaktadır. Bu eşiklerin sağlanamaması durumunda CI pipeline otomatik olarak durdurulmakta ve ilgili sürümün dağıtıma ilerlemesine izin verilmemektedir.

Bunun yanı sıra, projede kullanılan üçüncü parti kütüphaneler ve bağımlılıklar da güvenlik perspektifiyle ele alınmaktadır. Bağımlılık tarama (dependency scanning) mekanizmaları sayesinde, bilinen güvenlik açıklarına (CVE) sahip kütüphaneler tespit edilmekte ve bu kütüphanelerin güncellenmesi veya alternatifleriyle

değiştirilmesi sağlanmaktadır. Bu yaklaşım, yazılımın yalnızca kendi kodu açısından değil, ekosistem bağımlılıkları açısından da güvenli olmasını hedeflemektedir.

Container tabanlı mimari, güvenlik entegrasyonunun bir diğer önemli bileşenini oluşturmaktadır. Uygulama servislerinin izole Docker container'ları içerisinde çalıştırılması, olası güvenlik ihlallerinin sistemin tamamına yayılma riskini azaltmaktadır. Ayrıca container image'larının yalnızca güvenlik kontrollerinden geçmiş ve doğrulanmış sürümlerden oluşturulması, dağıtım ortamında daha güvenli bir çalışma yapısı sağlamaktadır.

Bu kapsamlı DevSecOps yaklaşımı sayesinde OleAI projesinde güvenlik; ölçülebilir, otomatik ve sürdürülebilir bir yapıya kavuşturulmuştur. Güvenlik risklerinin erken aşamada tespit edilmesi, yazılım kalitesinin artırılmasını sağlarken, operasyonel süreçlerde oluşabilecek güvenlik kaynaklı kesintilerin de önüne geçmektedir.

5.7 CI/CD Akış Tablosu:

OleAI projesi kapsamında tasarlanan CI/CD sürecinin daha anlaşılır ve izlenebilir hâle getirilmesi amacıyla, pipeline adımları tablo formatında modellenmiştir. Bu tablo, yazılım geliştirme sürecinde gerçekleştirilen her otomatik adımı, bu adımların girdilerini ve üretilen çıktıları açık bir şekilde göstermektedir.

Hazırlanan CI/CD akış tablosu, sistemin uçtan uca otomasyon yapısını somutlaştırmakta ve DevOps entegrasyonunun nasıl işletildiğini net bir biçimde ortaya koymaktadır.

No	Pipeline Adımı	Girdi (Input)	Gerçekleştirilen İşlem	Çıktı
1	**Kod Commit**	Geliştirici tarafından yapılan kod değişiklikleri	Kodun GitHub reposuna gönderilmesi (Push).	Güncellenmiş kaynak kod
2	**CI Tetikleme**	Yeni commit	Otomatik CI pipeline başlatılması (Trigger).	Aktif pipeline süreci
3	**Build**	Kaynak kod	Uygulamanın derlenmesi ve bağımlılıkların kontrol edilmesi.	Başarılı / Başarısız build sonucu
4	**Test**	Derlenmiş uygulama	Birim (Unit) testlerin otomatik olarak çalıştırılması.	Test raporları
5	**Kod Kalitesi Analizi**	Testten geçen kod	SonarQube üzerinden statik kod analizi ve Quality Gate kontrolü.	Kalite değerlendirme sonucu
6	**Güvenlik Kontrolleri**	Kod ve bağımlılıklar	Güvenlik açıkları ve riskli bağımlılıkların analizi (Vulnerability Scan).	Güvenlik onayı
7	**Containerization**	Onaylı kod	Docker image oluşturulması.	Versiyonlanmış Docker image
8	**Deploy**	Docker image	Image'ın hedef sunucu ortamına dağıtılması.	Canlıya alınmış uygulama
9	**Monitoring & Logging**	Çalışan uygulama	Performans, log ve hata takibi (Prometheus/ELK).	İzleme ve log kayıtları

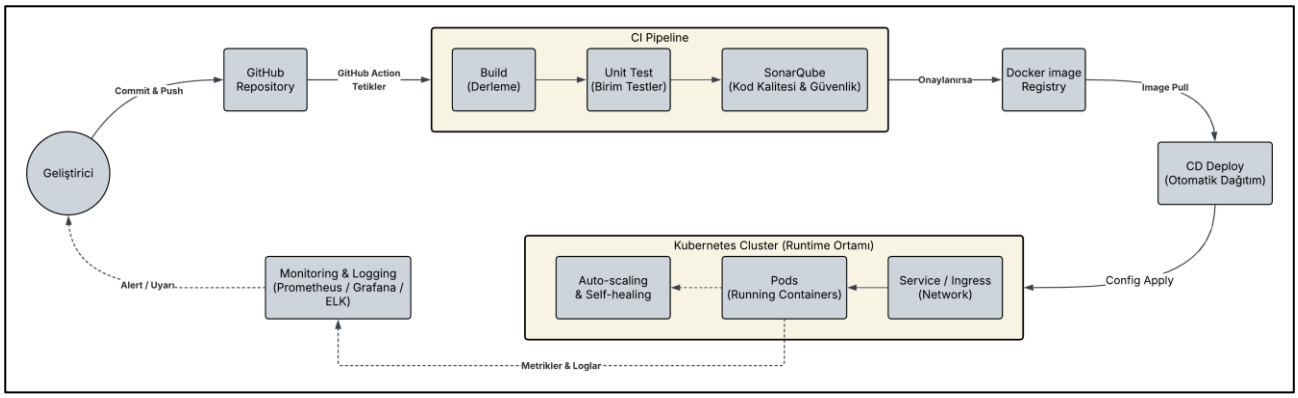
Bu tablo sayesinde CI/CD sürecinin her aşaması açık bir şekilde tanımlanmakta ve pipeline'ın hangi koşullarda ilerlediği veya durdurulduğu net olarak görülebilmektedir. Özellikle kod kalitesi ve güvenlik adımlarının deploy öncesinde konumlandırılması, sistemin kontrollü ve güvenilir bir şekilde işletildiğini göstermektedir.

5.8 DevOps Pipeline Diyagramı:

OleAI projesinde uygulanan DevOps yaklaşımının uçtan uca görünür hâle getirilmesi amacıyla, yazılım geliştirme sürecinden canlı ortamda çalışmasına kadar olan tüm aşamaları kapsayan bir DevOps entegrasyon diyagramı hazırlanmıştır. Bu diyagram, CI/CD pipeline, kod kalitesi ve güvenlik kontrolleri, containerization ve orchestration katmanları arasındaki ilişkiyi bütüncül bir bakış açısıyla göstermektedir.

Hazırlanan diyagramda, geliştiricinin yaptığı kod değişikliklerinin Git tabanlı versiyon kontrol sistemi üzerinden başlatılması; bu değişikliklerin otomatik olarak CI/CD pipeline'a aktarılması ve burada derleme, test, kod kalitesi ve güvenlik analizlerinden geçirilmesi açık bir şekilde modellenmiştir. Kalite ve güvenlik kontrollerini başarıyla geçen sürümlerin Docker container'ları hâline getirilerek dağıtım ortamına aktarılması ve sonrasında orchestration katmanı tarafından yönetilmesi diyagramın temel akışını oluşturmaktadır.

Bu yapı sayesinde, yazılımın geliştirme süreci ile operasyonel süreçleri arasında kesintisiz ve izlenebilir bir entegrasyon sağlanmakta; manuel müdahaleler en aza indirilerek kalite, güvenlik ve sürdürülebilirlik hedefleri desteklenmektedir.



Bu diyagram, OleAI projesinde uygulanan DevOps yaklaşımının geliştirme, dağıtım ve operasyon aşamalarını kapsayan bütüncül mimarisini göstermektedir.

5.9 DevOps Araç Seçim Raporu:

OleAI projesinde DevOps entegrasyonu kapsamında kullanılan araçlar, sektör standartları, proje gereksinimleri ve sürdürülebilirlik kriterleri göz önünde bulundurularak seçilmiştir. Araç seçiminde temel amaç; yazılım geliştirme, dağıtım ve operasyon süreçlerini otomatik, izlenebilir ve ölçeklenebilir bir yapı içerisinde yönetebilmektir.

5.9.1 Versiyon Kontrol Sistemi – GitHub

GitHub, projenin kaynak kod yönetimi için tercih edilmiştir. Geliştiriciler arası iş birliğini desteklemesi, branch ve pull request yapıları ile kontrollü geliştirme imkânı sunması ve CI/CD araçları ile doğal entegrasyonu sayesinde merkezi bir geliştirme platformu olarak konumlandırılmıştır.

5.9.2 CI/CD Aracı – GitHub Actions

CI/CD süreçlerinin otomasyonu için GitHub Actions kullanılması planlanmıştır. GitHub Actions; repository ile doğrudan entegre çalışabilmesi, pipeline tanımlarının kod olarak (YAML) yönetilebilmesi ve esnek workflow yapısı sayesinde CI/CD süreçlerinin kolayca yapılandırılmasına olanak tanımaktadır.

5.9.3 Kod Kalitesi ve Güvenlik – SonarQube

Kod kalitesi ve güvenlik analizleri için SonarQube tercih edilmiştir. Statik kod analizi, quality gate mekanizması ve güvenlik odaklı kural setleri sayesinde, yazılımın belirli kalite ve güvenlik standartlarını sürekli olarak sağlaması hedeflenmektedir. SonarQube, DevSecOps yaklaşımının temel bileşenlerinden biri olarak pipeline'a entegre edilmiştir.

5.9.4 Containerization – Docker

Uygulama bileşenlerinin izole, taşınabilir ve tutarlı bir şekilde çalıştırılabilmesi amacıyla Docker kullanılmıştır. Docker, uygulamanın farklı ortamlarda aynı şekilde çalışmasını garanti altına alırken, dağıtım süreçlerini de standartlaştırmaktadır.

5.9.5 Orchestration – Kubernetes

Container'ların çalışma zamanı yönetimi, ölçeklenebilirlik ve yüksek erişilebilirlik gereksinimleri doğrultusunda Kubernetes orchestration katmanı olarak planlanmıştır. Kubernetes; container yaşam döngüsü yönetimi, yük dengeleme ve otomatik ölçeklendirme gibi operasyonel sorumlulukları üstlenerek sistemin dinamik ihtiyaçlara uyum sağlamasını mümkün kılmaktadır.

5.9.6 Monitoring ve Logging Araçları

Uygulamanın performansını ve çalışma durumunu izlemek amacıyla monitoring ve logging araçları kullanılacaktır. Bu araçlar sayesinde sistem davranışları sürekli olarak gözlemlenebilecek, hata ve performans problemlerine hızlı müdahale edilebilecektir.

Bu araç seti ile OleAI projesinde DevOps süreçleri; modern, ölçeklenebilir ve kurumsal yazılım projeleriyle uyumlu bir yapıya kavuşturulmuştur.

6.Proje Yönetimi:

6.1 Proje Yönetim Yaklaşımı:

OleAI projesinde yazılım geliştirme süreci, akademik takvim ve proje kapsamı göz önünde bulundurularak fazlara ayrılmış bir proje yönetim yaklaşımı ile ele alınmıştır. Proje, güz ve bahar dönemlerini kapsayacak şekilde planlanmış; her dönem için hedeflenen çıktılar ve sorumluluklar net olarak tanımlanmıştır.

Proje yönetiminde, yazılım yaşam döngüsünün temel aşamaları olan analiz, tasarım, geliştirme, test ve raporlama adımları birer milestone olarak ele alınmıştır. Bu yaklaşım sayesinde proje süreci hem zaman açısından kontrol edilebilir hâle getirilmiş hem de ilerleme durumu düzenli olarak takip edilebilir bir yapıya kavuşturulmuştur.

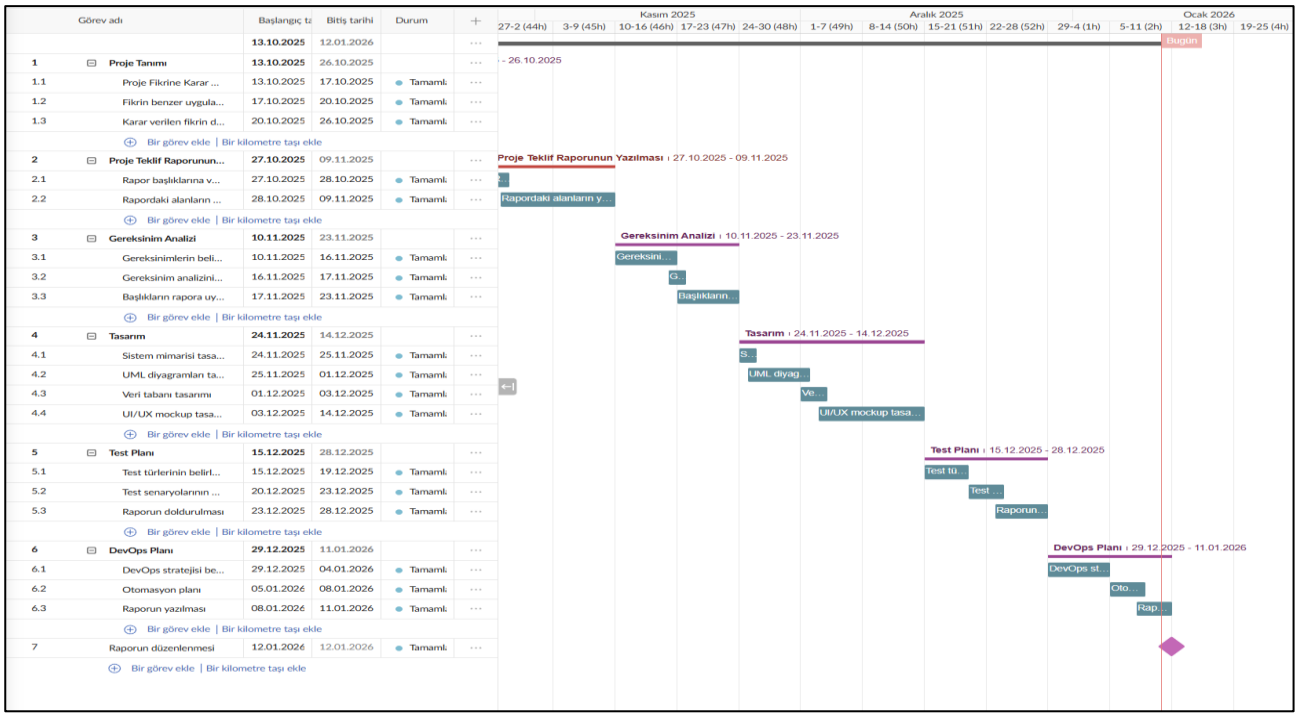
Yönetim sürecinde, teknik geliştirme faaliyetleri ile dokümantasyon ve raporlama çalışmaları paralel olarak yürütülmüş; her aşamanın çıktısı bir sonraki aşama için girdi oluşturacak şekilde planlanmıştır. Bu yapı, proje sürecinin sistematik ve sürdürülebilir bir şekilde ilerlemesini sağlamıştır.

6.2 Zaman Planlaması ve Görev Yönetimi:

OleAI projesinde zaman planlaması ve görev yönetimi, proje sürecinin kontrollü ve izlenebilir bir şekilde ilerlemesini sağlamak amacıyla yapılandırılmıştır. Proje, akademik takvim doğrultusunda aşamalara ayrılmış; her aşama için hedeflenen çıktılar ve zaman aralıkları önceden belirlenmiştir.

Zaman planlamasında, projenin analiz, tasarım, geliştirme, test ve raporlama gibi temel faaliyetleri birer milestone olarak ele alınmıştır. Bu milestone'lar, proje sürecindeki ilerlemenin takip edilebilmesi ve gecikmelerin erken aşamada tespit edilebilmesi açısından referans noktaları olarak kullanılmıştır.

Görev yönetimi kapsamında, her aşamada gerçekleştirilecek faaliyetler açık bir şekilde tanımlanmış ve bu görevler proje zaman çizelgesine entegre edilmiştir. Bu yaklaşım sayesinde proje süreci planlı bir şekilde yürütülmüş ve ilerleme durumu düzenli olarak izlenebilmiştir.



Hazırlanan Gantt Chart, projenin analizden teslim aşamasına kadar olan tüm sürecini zaman bazlı olarak göstermektedir. Çalışma, faz bazlı olarak planlanmış ve her ana aşama bir milestone ile sonlandırılmıştır. Bu yapı sayesinde proje ilerleyişinin belirlenen zaman çizelgesine uygun şekilde takip edilmesi hedeflenmiştir.

Analiz, tasarım, geliştirme ve test aşamaları sonunda belirlenen milestone'lar sayesinde projenin planlanan zaman çizelgesine uygun ilerleyip ilerlemediği izlenmiştir.

6.3 Görev Dağılımı ve Versiyon Kontrolü:

Bu projede görev dağılımı ve çalışma süreci, proje faaliyetlerinin planlı, izlenebilir ve sürdürülebilir bir şekilde yürütülmesini sağlamak amacıyla yapılandırılmıştır. Proje tek geliştirici tarafından yürütülmesine rağmen, görevler faz bazlı olarak ayrılmış ve her aşama için sorumluluklar açık şekilde tanımlanmıştır. Bu yaklaşım sayesinde proje sürecinde yapılacak çalışmaların kapsamı netleştirilmiş ve ilerleme durumu sistematik olarak takip edilebilmiştir.

6.3.1 Görev Dağılımı:

Görev dağılımı oluşturulurken, proje yaşam döngüsü analiz, tasarım, geliştirme, test, DevOps planlama ve raporlama aşamaları dikkate alınmıştır. Her bir aşamada gerçekleştirilecek faaliyetler belirlenmiş ve bu faaliyetlerin sorumluluğu proje geliştiricisine atanmıştır. Tek geliştiriciyle yürütülen projelerde bu tür bir yapılandırma, işlerin kontrolsüz ilerlemesini önlemek ve sürecin disiplinli bir şekilde yönetilmesini sağlamak açısından önem taşımaktadır.

Aşağıda, proje kapsamındaki temel görevler ve sorumluluk dağılımı özetlenmiştir.

Aşama	Görev	Sorumlu
Analiz	Gereksinimlerin analiz edilmesi ve sistem kapsamının belirlenmesi	Proje Geliştiricisi
Analiz	Fonksiyonel ve fonksiyonel olmayan gereksinimlerin tanımlanması	Proje Geliştiricisi
Tasarım	Sistem mimarisinin ve katmanlı yapının tasarlanması	Proje Geliştiricisi
Tasarım	UML diyagramlarının (Use Case, Class, Sequence) hazırlanması	Proje Geliştiricisi
Tasarım	Veritabanı yapısının ve veri modelinin tasarlanması	Proje Geliştiricisi
Tasarım	UI/UX mockuplarının tasarımı	Proje Geliştiricisi
Planlama	Zaman planlamasının oluşturulması ve Gantt Chart hazırlanması	Proje Geliştiricisi
Test	Test planının ve test senaryolarının oluşturulması	Proje Geliştiricisi
Test	Test süreçlerinin dokümante edilmesi	Proje Geliştiricisi
DevOps	DevOps stratejisinin ve CI/CD sürecinin planlanması	Proje Geliştiricisi
DevOps	Otomasyon, izleme ve güvenlik yaklaşımlarının planlanması	Proje Geliştiricisi
Raporlama	Teknik raporların hazırlanması	Proje Geliştiricisi
Raporlama	Proje dokümantasyonunun sonlandırılması ve teslim	Proje Geliştiricisi

Bu yapı sayesinde, her aşamada yapılacak çalışmalar önceden belirlenmiş ve proje süreci planlı bir şekilde yürütülmüştür.

6.3.2 Versiyon Kontrolü ve Repo Yönetimi:

Proje sürecinde versiyon kontrolü, dokümantasyon ve tasarım çıktılarının düzenli bir şekilde yönetilmesi amacıyla kullanılmıştır. Kod geliştirme aşamasına henüz geçilmemiş olsa da, proje kapsamında üretilen raporlar, diyagramlar ve planlama dokümanları bir Git tabanlı versiyon kontrol sistemi üzerinden saklanmıştır. Bu yaklaşım, proje sürecinde yapılan değişikliklerin izlenebilmesini ve önceki sürümlere gerektiğinde geri dönülebilmesini sağlamıştır.

Versiyon kontrolü için GitHub platformu tercih edilmiştir. Repository yapısı, proje dokümanlarının merkezi bir noktada toplanmasını ve ilerleyen aşamalarda geliştirilecek yazılım bileşenlerinin aynı yapı içerisinde yönetilebilmesini mümkün kılacak şekilde planlanmıştır. Böylece, geliştirme aşamasında gerçekleştirilecek olan faaliyetler için hazır ve sürdürülebilir bir altyapı oluşturulmuştur.

Versiyon kontrol sisteminin kullanılmasıyla birlikte, proje sürecinde yapılan güncellemelerin kayıt altına alınması, çalışma düzeninin korunması ve proje yönetiminin daha kontrollü bir şekilde yürütülmesi hedeflenmiştir.

7. Sonuç ve Referanslar:

Sonuç:

OleAl projesinin Güz dönemi çalışmaları kapsamında, gıda israfını önlemeyi ve kullanıcıların karar verme süreçlerini hızlandırmayı hedefleyen yapay zekâ destekli bir yemek öneri platformunun analiz ve tasarım aşamaları başarıyla tamamlanmıştır. Yazılım Geliştirme Yaşam Döngüsü (SDLC) prensiplerine sadık kalınarak yürütülen bu süreçte, projenin kavramsal çerçevesinden teknik mimarisine kadar tüm detaylar netleştirilmiştir.

Bu dönemde elde edilen temel çıktılar şunlardır:

- **Gereksinim Analizi:** Kullanıcı ihtiyaçları ve sistem gereksinimleri, detaylı "User Story" ve "Use Case" çalışmalarıyla belirlenmiş; projenin kapsamı ve hedefleri somutlaştırılmıştır.
- **Mimari Tasarım:** Sistemin sürdürülebilirliği ve ölçeklenebilirliği gözetilerek Clean Architecture prensiplerine dayalı, çok katmanlı bir backend yapısı tasarlanmıştır. Web (React) ve mobil (Kotlin) istemcilerin, merkezi bir API üzerinden yapay zekâ servisleri ile nasıl haberleşeceği entegrasyon planlarıyla ortaya konulmuştur.
- **Yapay Zekâ Entegrasyonu:** Kullanıcı alışkanlıklarını ve malzeme envanterini (TF-IDF ve Kosine Benzerliği algoritmalarıyla) analiz edecek öneri motorunun matematiksel modeli ve sistem içindeki konumu belirlenmiştir.
- **Veri Yönetimi:** İlişkisel veri tabanı tasarımı (ER Diyagramları) tamamlanarak, veri tutarlılığını sağlayacak altyapı hazırlanmıştır.

Yapılan bu detaylı tasarım çalışmaları, projenin Bahar döneminde gerçekleştirilecek olan implementasyon (kodlama) ve test aşamaları için sağlam bir temel oluşturmuştur. Sonuç olarak OleAl, sadece teknik bir yazılım projesi olarak değil; aynı zamanda modern yazılım mühendisliği standartlarına uygun, genişletilebilir ve sosyal fayda odaklı bir çözüm olarak hayata geçmeye hazırdır.

Referanslar:

1. **Martin, R. C.** (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall. (Projenin backend mimarisindeki temel prensipler için).
2. **Sommerville, I.** (2015). *Software Engineering* (10th Edition). Pearson. (SDLC ve gereksinim analizi metodolojileri için).
3. **Microsoft Documentation.** (2025). .*NET Application Architecture: Microservices and Containers*. Erişim: https://learn.microsoft.com/en-us/dotnet/architecture/ (Backend teknolojileri ve mimari desenler için).
4. **Aggarwal, C. C.** (2016). *Recommender Systems: The Textbook*. Springer. (Yapay zekâ öneri algoritmaları, TF-IDF ve benzerlik hesaplamaları için teorik altyapı).
5. **Scikit-learn Developers.** (2025). *User Guide: Feature Extraction & Text Data*. Erişim: https://scikit-learn.org/stable/modules/feature_extraction.html (Python tabanlı yapay zekâ modellemesi için).
6. **Google Developers.** (2025). *Android Jetpack Compose Documentation*. Erişim: https://developer.android.com/jetpack/compose (Mobil uygulama tasarım standartları için).
7. **React Documentation.** (2025). *React: The Library for Web and Native User Interfaces*. Erişim: https://react.dev/ (Web arayüzü teknolojisi için).
8. **Google.** (2025). *Gemini (AI Model)*. Erişim: https://deepmind.google/technologies/gemini/ (Proje dokümantasyonu oluşturma, UML diyagramlarının kurgulanması gibi süreçlerde yapay zekâ asistanı olarak kullanılmıştır).