metin, yazı tipi, logo, grafik içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Proje Takip Sistemi**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BİTİRME PROJESİ**

Bilgisayar Mühendisliği Bölümü

**DANIŞMAN**

Doç. Dr. ÖNDER DEMİR

İSTANBUL, 2025

**MARMARA ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencileri Metehan ATEŞ, Eyüp Ensari KARAKUŞ, Emirhan ÇINAR tarafından “Proje Takip Sistemi” başlıklı proje çalışması, 19.06.2025 tarihinde savunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

**Jüri Üyeleri**

|  |  |  |
| --- | --- | --- |
| Doç. Dr. Önder DEMİR  Marmara Üniversitesi | **(Danışman)** | (İMZA)………….. |
| Arş. Gör. Damla MENGÜŞ  Marmara Üniversitesi | (Üye) | (İMZA)………….. |
|  |  |  |

**ÖNSÖZ**

Proje çalışmamız süresince karşılaştığımız bütün problemlerde, sabırla yardım ve bilgilerini esirgemeyen, tüm desteğini sonuna kadar yanımda hissettiğim değerli hocam, sayın Doç. Dr. Öğr. Üyesi Önder Demir’e en içten teşekkürlerimi sunarım.

Bu proje çalışması fikrinin oluşması ve ortaya çıkmasındaki önerisi ve desteğinden dolayı değerli hocam Doç. Dr. Öğr. Üyesi Önder Demir’e teşekkür ederim.

İçindekiler

[**ÖZET** 4](#_Toc201199486)

[**ŞEKİL LİSTESİ** 6](#_Toc201199487)

[1. **GİRİŞ** 7](#_Toc201199488)

[1.1 Proje Çalışmasının Amacı ve Önemi 7](#_Toc201199489)

[**2. LİTERATÜR TARAMASI** 8](#_Toc201199490)

[2.1 Müşteri Geri Bildirimlerinin Analizi 8](#_Toc201199491)

[2.2 Müşteri Destek Yönetim Sistemleri 9](#_Toc201199492)

[2.3 Yapay Zekâ Tabanlı Destek Sistemleri 10](#_Toc201199493)

[2.4 Şikâyet Yönetimi ve Önceliklendirme Sistemleri 10](#_Toc201199494)

[2.5 Proje Yönetimi ve Takibi 10](#_Toc201199495)

[2.6 Müşteri Memnuniyeti Üzerine Çalışmalar 11](#_Toc201199496)

[**3. MATERYAL VE YÖNTEM** 11](#_Toc201199497)

[3.1 Kullanılan Teknolojiler 11](#_Toc201199498)

[3.2 Sistem Mimarisi ve Tasarımı 11](#_Toc201199499)

[**3.2.1 Klasör Yapısı ve Organizasyon** 12](#_Toc201199500)

[**3.2.2 Server Mimarisi ve Tasarımı** 13](#_Toc201199501)

[**3.2.3 Back-End Tasarımı** 14](#_Toc201199502)

[3.3 Back-End Geliştirme Süreci 15](#_Toc201199503)

[**3.3.1 Uygulama Başlangıç Yapılandırması** 15](#_Toc201199504)

[**3.3.2 NuGet Paketleri** 17](#_Toc201199505)

[**3.3.3 Entity Sınıfları** 18](#_Toc201199506)

[**3.3.4 DTO Sınıfları** 20](#_Toc201199507)

[**3.3.5 Controller Sınıfları** 21](#_Toc201199508)

[**3.3.6 Kimlik Doğrulama Süreci** 22](#_Toc201199509)

[3.4 Front-End Geliştirme Süreci 24](#_Toc201199510)

[**3.4.1 App.js açıklaması** 24](#_Toc201199511)

[**3.4.2 Kullanıcı Arayüzü** 25](#_Toc201199512)

[**3.4.3 Admin Arayüzü** 27](#_Toc201199513)

[**3.4.4 API ile Haberleşme Görevi** 28](#_Toc201199514)

[**3.4.5 Sayfa Tasarımları** 28](#_Toc201199515)

[**3.5.1 Azure Üzerinde CRM.API Yayınlanması** 34](#_Toc201199516)

[**3.5.2 AWS SQL Server Kurulumu ve Entegrasyonu** 35](#_Toc201199517)

[**4. BULGULAR VE TARTIŞMA** 36](#_Toc201199518)

[4.1 Sistem Performansı ve Kullanıcı Deneyimi 36](#_Toc201199519)

[4.2 Taleplerin Takibi ve Yönetimi 36](#_Toc201199520)

[4.3 Sistem Performansı ve Kararlılık 37](#_Toc201199521)

[4.4 Sistemin Geliştirilmesi İçin Öneriler 37](#_Toc201199522)

[4.5 Karşılaşılan Sorunlar ve Çözümler 37](#_Toc201199523)

[**KAYNAKLAR** 38](#_Toc201199524)

# **ÖZET**

Bu çalışma, şirketlerin müşteri taleplerini ve şikayetlerini etkin bir şekilde yönetmesini sağlamak amacıyla geliştirilmiş bir proje takip sistemi sunmaktadır. Sistem, kullanıcıların şikayetlerini yazılı iletebilmesini ve taleplerin acil, çok acil, sıradan gibi öncelik seviyelerine göre sınıflandırılmasını sağlar. Yöneticiler, gelen talepleri uygun çalışanlara atayarak sürecin takip edilmesini ve yönetilmesini gerçekleştirir. Sistemde, taleplerin alınma tarihi, aşama durumu, okunma durumu ve çözüm süreci gibi bilgilerin kaydı tutulmaktadır. Ek olarak, şirketlerin ürün lisans yönetimi süreçlerini takip edebilmelerine yönelik bir modül de sistemde bulunmaktadır. Bu sayede, müşteri destek süreçlerinin hızlandırılması ve operasyonel verimliliğin artırılması hedeflenmiştir. Sistem, müşteri taleplerinin etkin bir şekilde yönetilmesine olanak tanırken, çözüm sürecinde hız ve verimlilik sağlamayı amaçlamaktadır.

**Haziran, 2025                                               Öğrenciler**

EYÜP ENSARİ KARAKUŞ

EMİRHAN ÇINAR

METEHAN ATEŞ

**ABSTRACT**

This study presents a project tracking system developed to enable companies to effectively manage customer requests and complaints. The system allows users to submit their complaints in written form and classifies the requests based on priority levels such as urgent, highly urgent, and normal. Managers can assign incoming requests to appropriate employees to ensure that the process is properly monitored and managed. The system keeps records of various details such as the submission date of requests, their current status, whether they have been read, and the progress of the resolution process. Additionally, a module is included to help companies manage their product license tracking processes. In this way, the system aims to accelerate customer support operations and increase operational efficiency. While enabling the effective management of customer requests, the system also aims to provide speed and efficiency during the resolution phase.

**June, 2025                                               Students**

EYÜP ENSARİ KARAKUŞ

EMİRHAN ÇINAR

METEHAN ATEŞ

# **ŞEKİL LİSTESİ**

**Şekil 1.1** Kapsama göre şikâyet ve övgü oranlarının karşılaştırması

**Şekil 3.1** Veritabanı diyagramı

**Şekil 3.2** Back-End proje dizini

**Şekil 3.3** Program.cs dosyası 1

**Şekil 3.4** Program.cs dosyası 2

**Şekil 3.5** Entity sınıf dizini

**Şekil 3.6** DTO sınıf dizini

**Şekil 3.7** Controller sınıf dizini

**Şekil 3.8** Menü

**Şekil 3.9** Admin paneli

**Şekil 3.10** Proje paneli

**Şekil 3.11** Görev paneli

**Şekil 3.12** Sorun paneli

**Şekil 3.13** Kullanıcı görüntüleme paneli

**Şekil 3.14** Sorun durumu paneli

# 1. **GİRİŞ**

Günümüzde şirketler, müşteri memnuniyetini artırmak ve hizmet kalitesini iyileştirmek amacıyla müşteri geri bildirimlerini etkin bir şekilde yönetmeye ihtiyaç duymaktadır. Özellikle dijitalleşmenin hız kazanmasıyla birlikte, müşteri taleplerinin hızlı ve verimli bir şekilde işlenmesi büyük önem taşımaktadır. Geleneksel yöntemlerle yürütülen müşteri destek süreçleri, zaman alıcı ve verimsiz olabilir; bu da müşteri memnuniyetinin azalmasına ve şirketin itibar kaybına yol açabilir.

Bu çalışmada, şirketlerin müşteri taleplerini sistematik bir şekilde yönetmesini sağlamak amacıyla geliştirilen proje takip sistemi ele alınmaktadır. Sistem, müşterilerin karşılaştıkları problemlere ilişkin şikayetlerini yazılı formatında iletebilmelerine olanak tanır. Gönderilen talepler, acil, çok acil veya sıradan gibi öncelik seviyelerine ayrılarak kategorize edilir. Yöneticiler, gelen talepleri uygun çalışanlara atayarak sürecin etkin bir şekilde yürütülmesini sağlar. Ayrıca, sistemde her müşteri talebi için gönderi tarihi, mevcut aşama, okunma durumu, ilerleme kaydı ve çözüm durumu gibi bilgilerin takibi yapılmaktadır.

Bu tez çalışması kapsamında, geliştirilen sistemin detayları incelenecek, müşteri destek süreçlerine katkıları değerlendirilecektir.

## **1.1 Proje Çalışmasının Amacı ve Önemi**

Günümüz iş dünyasında müşteri memnuniyeti, şirketlerin başarısını doğrudan etkileyen kritik bir faktördür. Müşterilerin taleplerine hızlı ve etkili bir şekilde yanıt vermek, marka sadakatini güçlendirmek ve rekabet avantajı elde etmek için büyük önem taşımaktadır. Ancak, geleneksel yöntemlerle yürütülen müşteri destek süreçleri genellikle yavaş, düzensiz ve verimsizdir. Bu durum, müşteri şikayetlerinin gecikmesine, yanlış yönetilmesine ve müşteri memnuniyetsizliğine yol açabilir. Bu çalışmanın temel amacı, şirketlerin müşteri taleplerini ve şikayetlerini daha sistematik, verimli ve hızlı bir şekilde yönetmesini sağlayan bir proje takip sistemi geliştirmektir. Geliştirilen sistem, müşterilerin karşılaştıkları problemleri yazılı olarak iletebilmesine olanak tanırken, taleplerin öncelik seviyelerine göre sınıflandırılması ve yöneticiler tarafından ilgili çalışanlara atanması gibi süreçleri kolaylaştırmaktadır.

Bu proje, şirketlerin müşteri taleplerini sadece kaydetmekle kalmayıp, gelişmiş bir takip mekanizması sunarak taleplerin hangi aşamada olduğunu, okunup okunmadığını ve çözüme ulaşıp ulaşmadığını gösteren bir yapı sağlamaktadır. Böylece, müşteri şikayetlerinin takibi daha şeffaf ve ölçülebilir hale getirilecektir. Ayrıca, projeye entegre edilen ürün lisans yönetimi modülü, şirketlerin lisans süreçlerini dijital bir ortamda takip edebilmesine olanak tanıyarak iş süreçlerini optimize etmektedir. Bu sayede, müşteri destek süreçlerinin hızlandırılması ve operasyonel yükün azaltılması hedeflenmektedir. Sonuç olarak, bu çalışmanın hem akademik hem de sektörel açıdan önemli katkılar sunması beklenmektedir. Akademik açıdan, müşteri destek süreçlerinde çözüm önerilerinin etkinliğini değerlendirmek için bir örnek teşkil ederken, sektörel açıdan şirketlerin müşteri memnuniyetini artırmasına ve operasyonel süreçlerini iyileştirmesine yardımcı olacak bir sistem önerisi sunmaktadır.

# **2. LİTERATÜR TARAMASI**

Bu bölümde, müşteri taleplerinin ve şikayetlerinin yönetimine yönelik mevcut yöntemler, yazılım sistemleri ve teknolojik yaklaşımlar incelenmiştir. Alanla ilgili güncel akademik çalışmalar, sektörde kullanılan çözüm modelleri ve yazılım mimarileri değerlendirilerek, geliştirilen proje takip sisteminin hangi ihtiyaçlara cevap verdiği ve hangi boşlukları doldurduğu ortaya konulmuştur.

## **2.1 Müşteri Geri Bildirimlerinin Analizi**

Müşteri geri bildirimleri, hizmet kalitesini değerlendirme ve iyileştirme süreçlerinde kritik bir rol oynamaktadır. Yapılan analizde, MTA New York City Transit (NYCT) müşteri geri bildirim verileri incelenmiş ve şikâyet ile övgü oranları kategorilere göre karşılaştırılmıştır. Tablo 1.1’de görüldüğü üzere elde edilen bulgulara göre, bazı kategorilerde övgü oranlarının belirgin şekilde düşük olduğu gözlemlenmiştir. Özellikle MetroCard ve Other NYCT kategorilerinde hiç övgü kaydına rastlanmamış olması, bu alanlarda müşteri memnuniyetinin yetersiz raporlandığını veya geri bildirim mekanizmalarının eksik çalıştığını göstermektedir. Bunun yanı sıra, Subways ve Buses kategorilerinde şikâyet sayılarının yüksek olmasına rağmen övgülerin oldukça sınırlı olması, bu hizmetlerde müşteri memnuniyetinin düşük olduğu ya da olumlu geri bildirimlerin bildirilme oranının düşük kaldığı şeklinde yorumlanabilir. Buna karşılık, Access-A-Ride kategorisinde övgü/şikâyet oranının diğer hizmetlere kıyasla daha yüksek olduğu tespit edilmiştir. Bu farklılıklar, müşteri deneyimini raporlamadaki eğilimlerin kategori bazında değişiklik gösterdiğine ve bazı hizmet alanlarında olumlu geri bildirimlerin yeterince teşvik edilmediğine işaret etmektedir. Literatürde de belirtildiği gibi, olumsuz deneyimler müşteriler tarafından daha sık raporlanırken, olumlu deneyimlerin bildirilme oranı genellikle düşüktür (Van Doorn vd., 2010). Bu bağlamda, müşteri memnuniyetini artırmak ve daha dengeli bir geri bildirim sistemi oluşturmak için hizmet sağlayıcıların, olumlu geri bildirimleri teşvik eden mekanizmalar geliştirmesi ve müşteri geri bildirim süreçlerini daha erişilebilir hale getirmesi önerilmektedir.

metin, ekran görüntüsü, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Şekil 1.1**: **Kapsama göre karşılaştırma**

## **2.2 Müşteri Destek Yönetim Sistemleri**

Müşteri destek yönetim sistemleri (Customer Support Management Systems, CSMS), şirketlerin müşterilerinden gelen talepleri ve şikayetleri verimli bir şekilde yönetmelerine olanak tanır. Bu sistemler, genellikle çağrı merkezi yazılımları, e-posta yönetimi, canlı sohbet destek sistemleri gibi araçlar kullanır. Çalışmalarda, geleneksel destek sistemlerinin yavaş ve verimsiz olmasının müşteriler üzerinde olumsuz etkiler yarattığı belirtilmiştir [1]. Birçok firma, bu tür süreçleri dijitalleştirerek ve otomatikleştirerek verimliliği artırmayı hedeflemektedir. Çeşitli yazılımlar, tüm müşteri etkileşimlerini merkezi bir platformda toplar ve kullanıcıların hızlı bir şekilde soruları yanıtlamasına imkân tanır [2].

## **2.3 Yapay Zekâ Tabanlı Destek Sistemleri**

Yapay zekâ destekli müşteri hizmetleri, doğal dil işleme (NLP), makine öğrenmesi ve derin öğrenme algoritmalarını kullanarak müşteri şikayetlerini analiz eder ve otomatik yanıtlar üretir. Yapay zekâ, özellikle tekrarlayan soruları hızlı bir şekilde çözme konusunda büyük bir avantaj sağlamaktadır. Örneğin, bir araştırmada yapay zekâ tabanlı bir destek sisteminin, geleneksel sistemlere göre müşteri yanıt sürelerini %40 oranında azalttığı bulunmuştur [3]. Ayrıca, bu tür sistemler, daha önceki şikayetleri ve talepleri analiz ederek gelecekteki problemlere yönelik çözüm önerileri sunabilmektedir [4].

## **2.4 Şikâyet Yönetimi ve Önceliklendirme Sistemleri**

Müşteri şikayetlerinin önceliklendirilmesi, aciliyet durumuna göre sıralanması işletmeler için oldukça önemlidir. Çalışmalarda, şikayetlerin aciliyetine göre sınıflandırılmasının, çözüm süresini kısaltarak müşteri memnuniyetini artırdığı vurgulanmıştır. Birçok şirket, "Acil", "Yüksek Öncelik", "Normal" gibi etiketler kullanarak şikayetlerin sıralanmasını sağlar. Bunun yanı sıra, bazı sistemlerde yapay zekâ algoritmalarına dayanarak taleplerin öncelik sırası otomatik olarak belirlenebilmektedir [5].

## **2.5 Proje Yönetimi ve Takibi**

JIRA gibi proje yönetim araçları, yazılım geliştirme ve proje yönetimi süreçlerinin takip edilmesinde yaygın olarak kullanılmaktadır. Bu araçlar, projelerin her aşamasını izlemeyi ve yönetmeyi sağlar. Yapılan çalışmalar, bu tür yazılımların proje takibi sürecinde zaman ve kaynak yönetimini optimize ettiğini göstermektedir. Ayrıca, JIRA gibi araçlar, takımların iş birliği yapmalarını kolaylaştırarak, proje hedeflerine daha hızlı ulaşılmasına yardımcı olmaktadır [6]. Özellikle, yazılım geliştirme projelerinde, kullanıcı hikayeleri, sprint planlamaları ve sorun takipleri gibi özellikler büyük fayda sağlamaktadır.

## **2.6 Müşteri Memnuniyeti Üzerine Çalışmalar**

Müşteri memnuniyeti, bir işletmenin sürdürülebilir başarısı için kritik öneme sahiptir. Müşteri destek süreçlerinin hızlı ve etkili bir şekilde yönetilmesi, müşteri memnuniyetini doğrudan etkiler. Yapılan araştırmalar, hızlı yanıt sürelerinin ve yüksek çözüm oranlarının müşteri memnuniyetini artırmada önemli faktörler olduğunu ortaya koymuştur [7]. Ayrıca, yapay zeka tabanlı sistemlerin müşteri memnuniyetine olan katkısı, daha kişiselleştirilmiş ve hızlı çözüm önerileri sunmasıyla ilişkilidir [8].

# **3. MATERYAL VE YÖNTEM**

Bu projede, modern yazılım geliştirme yaklaşımlarına uygun olarak çeşitli teknolojiler kullanılmıştır.

## **3.1 Kullanılan Teknolojiler**

**.NET Framework:** Projenin sunucu tarafı geliştirmeleri için Microsoft’un .NET platformu tercih edilmiştir. Güçlü altyapısı, geniş kütüphane desteği ve yüksek performanslı çalışması sayesinde kurumsal düzeyde uygulamalar geliştirmek için ideal bir çözümdür. Projede C# programlama dili kullanılarak geliştirilmiştir.

**MVC Mimari Yapısı:** Projede, .NET MVC yapısı benimsenmiştir. Bu yapı sayesinde uygulamanın veri, iş mantığı ve kullanıcı arayüzü katmanları birbirinden ayrılmıştır. Bu sayede kodun okunabilirliği, sürdürülebilirliği ve test edilebilirliği artırılmıştır.

**React.js:** Kullanıcı arayüzünün geliştirilmesinde kullanılmıştır. React.js, bileşen tabanlı yapısıyla yeniden kullanılabilir arayüzler geliştirmeyi kolaylaştırır.

**SQL Server Management Studio:** Veri tabanı işlemleri için Microsoft SQL Server tercih edilmiştir. SSMS aracı kullanılarak veri tabanı tasarımı yapılmış, tablolar oluşturulmuş, ilişkiler kurulmuş ve SQL sorguları ile veri yönetimi sağlanmıştır. Bu teknolojiler, projenin modüler, güvenli ve performanslı bir yapıya sahip olmasını sağlamıştır.

## **3.2 Sistem Mimarisi ve Tasarımı**

Sistem, modüler ve ölçeklenebilir bir mimari ile tasarlanmıştır. Component-based architecture prensiplerine uygun olarak, her bir işlevsel alan ayrı modüller halinde organize edilmiştir.

### **3.2.1 Klasör Yapısı ve Organizasyon**

Proje, maintainability ve scalability prensipleri gözetilerek organize edilmiştir. Bu yapı, Separation of Concerns prensibini uygulayarak, her modülün belirli bir sorumluluğa sahip olmasını sağlamaktadır.

src/

├── components/ # UI Bileşenleri

│ ├── Admin/ # Yönetim Paneli Bileşenleri

│ │ ├── CustomFields/ # Özel Alan Yönetimi

│ │ ├── IssueStatus/ # Durum Yönetimi

│ │ ├── IssueTypes/ # Görev Tipi Yönetimi

│ │ ├── Projects/ # Proje Yönetimi

│ │ ├── Roles/ # Rol Yönetimi

│ │ ├── Users/ # Kullanıcı Yönetimi

│ │ └── Workflows/ # İş Akışı Yönetimi

│ ├── Auth/ # Kimlik Doğrulama

│ ├── Common/ # Ortak Bileşenler

│ ├── Issues/ # Görev Yönetimi

│ ├── Layout/ # Sayfa Düzeni

│ ├── Projects/ # Proje Bileşenleri

│ └── Sprints/ # Sprint Yönetimi

├── context/ # Global State Yönetimi

├── hooks/ # Custom React Hooks

├── pages/ # Sayfa Bileşenleri

├── services/ # API Servisleri

├── styles/ # Stil Dosyaları

└── utils/ # Yardımcı Fonksiyonlar

### **3.2.2 Server Mimarisi ve Tasarımı**

Şekil 3.1’de yer alan bu veri tabanı yapısı, proje takip sistemi için tasarlanmıştır ve özellikle proje bazlı iş takibi, sorun yönetimi (issue tracking), kullanıcı aktiviteleri ve iş akışları gibi süreçlerin detaylı bir şekilde izlenebilmesini sağlamaktadır. Sistemin ana bileşeni olarak Issue tablosu konumlandırılmıştır ve projelerde karşılaşılan sorunların veya görevlerin takibi bu tablo aracılığıyla gerçekleştirilmektedir. Issue tablosuna bağlı olarak yapılandırılan IssueHistory, Comments ve Attachments gibi tablolar sayesinde her bir sorunla ilgili detaylı geçmiş kayıtları, yorumlar ve dosya ekleri yönetilebilmektedir. Sürecin hangi aşamada olduğu ve hangi türde işin yürütüldüğü ise IssueStatus, IssueTypes ve Workflows gibi yardımcı tablolar aracılığıyla tanımlanmaktadır.

Ayrıca, projelerin modüler bir şekilde yapılandırılabilmesi Projects, Sprints ve ProjectMembers tabloları ile sağlanmıştır. Tüm sistem kullanıcıları, roller ve yetkiler Users tablosunda saklanmaktadır. Sistem içerisindeki değişikliklerin kullanıcılar tarafından takip edilebilmesi Notifications tablosu ile mümkün kılınmıştır. Projelere özel alanların tanımlanabilmesi ise CustomFields tablosu aracılığıyla sağlanarak sisteme esneklik kazandırılmıştır.

Bu yapı sayesinde, kurumlar tarafından ekip içi görevlerin daha verimli bir şekilde takip edilmesi ve müşteri ile ilişkili süreçlerin düzenli bir şekilde yönetilmesi mümkün hale getirilmiştir.

metin, diyagram, plan, şematik içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şekil 3.1: Veritabanı diyagramı

### **3.2.3 Back-End Tasarımı**

CRM çözümünün back-end tarafında, Şekil 3.2’de yer alan ***CRM.API*** projesinde ***CRMDB***veritabanından Scaffold işlemi gerçekleştirilerek, veritabanındaki tabloların her biri proje içerisinde ***Models/EfCore*** klasörü altına sınıflar olarak oluşturulmuştur. Daha sonra, her bir tablo için ***crud(get,insert,delete,update)*** işlemlerini gerçekleştirecek controller'lar yazılmıştır.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 3.2: Back-End proje dizini

## **3.3 Back-End Geliştirme Süreci**

### **3.3.1 Uygulama Başlangıç Yapılandırması**

*CRM.API* uygulaması, .NET 9 tabanlı bir “ASP.NET Core Web API” projesi olarak yapılandırılmıştır. Uygulamanın başlatılması için gerekli yapılandırmalar, *Program.cs* dosyası aracılığıyla gerçekleştirilmiştir. Uygulamanın barındırma ortamı oluşturulmuş ve servis konteyneri üzerinden çeşitli servisler yapılandırılmıştır.

İlk olarak, Şekil 3.3’te göründüğü gibi *controller* sınıflarının projeye dahil edilmesi sağlanmıştır. Ardından, *API* belgelerinin oluşturulması ve sunulması amacıyla *OpenAPI* desteği entegre edilmiştir. Bu sayede geliştiriciler, uygulamanın endpointlerini görsel arayüzler üzerinden görüntüleyebilmekte ve test işlemleri gerçekleştirebilmektedir.

Veri tabanı bağlantısı için Entity Framework Core kullanılarak, uygulamaya özel tanımlanmış *DbContext* sınıfı üzerinden SQL Server entegrasyonu sağlanmıştır. Bu bağlantı, yapılandırma dosyasında tanımlı olan bir bağlantı dizesi üzerinden gerçekleştirilmiştir.

*JSON* serileştirme sürecinde karşılaşılabilecek döngüsel referans hatalarının önlenmesi amacıyla gerekli *JSON* ayarları yapılandırılmıştır. Ayrıca veri çıktılarının okunabilirliğini artırmak ve istemcilerle uyumlu veri biçimi üretmek amacıyla *camelCase* adlandırma politikası uygulanmış; *enum* değerlerinin *string* olarak serileştirilmesi ve tarih-saat biçimlerinin *ISO 8601* standardında yazılması sağlanmıştır.

metin, elektronik donanım, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şekil 3.3: Program.cs dosyası 1

Uygulama ortamına göre farklı yapılandırmalar uygulanmıştır. Geliştirme ortamında Şekil 3.4’te göründüğü gibi *Swagger UI* ve *ReDoc* arayüzleri etkinleştirilmiş; aynı yapı üretim ortamı için de ayrı bloklar altında sağlanmıştır. API belgelerine erişim için gerekli yönlendirmeler yapılmış ve kullanıcıların belgelenmiş uç noktalara ulaşabilmesi sağlanmıştır.

Uygulama güvenliğinin artırılması amacıyla *HTTPS* yönlendirmesi etkinleştirilmiş; yetkilendirme mekanizmaları uygulanmış ve gelen *HTTP* istekleri uygun denetleyicilere yönlendirilmiştir. Tüm bu işlemler tamamlandıktan sonra uygulama başlatılarak hizmete alınmıştır.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şekil 3.4: Program.cs dosyası 2

### **3.3.2 NuGet Paketleri**

Proje geliştirme sürecinde, .NET Core teknolojisinin sunduğu avantajlardan yararlanmak ve modern bir web API mimarisi kurmak amacıyla çeşitli NuGet paketleri kullanılmıştır.

**Bcrypt.Net-Next:** Kullanıcı şifrelerini güvenli bir şekilde saklamak amacıyla kullanılmıştır.

**Microsoft.AspNetCore.OpenApi:** Uygulamada OpenApi belgelerinin otomatik oluşturulmasını sağlamıştır.

**Microsoft.EntityFrameworkCore:** Microsoft’un ORM çözümü olan Entity Framework Core’un temel paketidir.

**Microsoft.EntityFrameworkCore.Design:** EF Core ile proje geliştirirken design-time araçlarını sağlar.

**Microsoft.EntityFrameworkCore.SqlServer:** Entity Framework Core’un SQL Server ile çalışmasını sağlayan paket olarak kullanılır.

**Microsoft.EntityFrameworkCore.Tools:** EF Core komut satırı araçlarını içerir.

**Scalar.AspNetCore:** Scalar değer türlerinin yönetilmesini kolaylaştıran kütüphanedir.

**Swashbuckle.AspNetCore.ReDoc:** OpenApi belgelerini modern ve kullanıcı dostu bir arayüz olan “ReDoc” üzerinden sunar.

**Swashbuckle.AspNetCore.SwaggerUI:** API’yi tarayıcıda test etmeye olanak tanıyan bileşeni sunar.

### **3.3.3 Entity Sınıfları**

Bu bölümde uygulamanın temel veri yapısını oluşturan model sınıfı tanıtılmaktadır. Şekil 3.5’te göründüğü üzere *Model* sınıfları, veri tabanı ile uygulama arasında köprü görevi görür. Böylece Entity Framework Core aracılığıyla verilerin nesne tabanlı olarak yönetilmesini sağlamaktadır.

metin, ekran görüntüsü, yazılım, tasarım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şekil 3.5: Entity sınıf dizini

Her bir model sınıfı, veri tabanındaki bir tabloyu temsil edecek şekilde yapılandırılır. Bu yapılar sayesinde veri tutarlılığı korunur. Örnek olarak model sınıfından *Workflow* sınıfı aşağıda işleviyle beraber gösterilmektedir. Örnek olarak anlatılan bu sınıftaki yapı mantığı diğer model sınıflarında da uygulanmaktadır.

* Workflow Sınıfı:

1. WorkflowId: Her bir iş akışını benzersiz olarak tanımlayan birincil anahtar alanıdır.
2. FromStatus: İşin başlamakta olduğu durumun kimliğidir. Bir *IssueStatus* nesnesine karşılık gelir.
3. ToStatus: İşin geçeceği yeni durumun kimliğidir.Yine bi *IsseStatus* nesnesine karşılık gelir.
4. IssuTypeId: Bu iş akışının ait olduğu görev türünü tanımlar. Eski sistemle uyumluluk amacıyla tutulmaktadır.
5. Name: İş akışına verilen isim bilgisidir. Kullanıcıya gösterim ve tanımlama amaçlıdır.
6. Description: İş akışının açıklamasıdır. Kullanıcının iş akışını anlamasını kolaylaştırır.

* İlişkiler:

1. FromStatusNavigation: *FromStatus* alanı üzerinden *IssueStatus* sınıfına olan bağlantıyı temsil eder.
2. ToStatusNavigation: *ToStatus* alanı üzerinden *IssueStatus* sınıfına olan bağlantıyı temsil eder. Birden- bire ilişkiyi temsil eder.
3. IssueType: İş akışının bağlı olduğu *IssueType* varlığı ile olan ilişkiyi belirtir.
4. WorkflowStatuses: Bu iş akışına ait durum geçmişlerini tutan koleksiyondur.Birden-çoğa ilişkiyi temsil eder.
5. WorkflowIssueTypes: Bu iş akışıyla ilgili görev türlerini belirten ilişki tablosudur. Çoktan- çoğa ilişkiyi temsil eder.

*Workflow* sınıfı, geliştirilen proje takip sisteminde iş akışlarını kontrol etmek amacıyla kullanılır. Bir görevin “yeni” durumundan “atandı” durumuna veya “çözüldü” durumuna geçişini bu sınıf aracılığıyla tanımlamak mümkündür. Kullanıcı rollerine göre bu geçişlere izin verilebilir veya kısıtlanabilir.

### **3.3.4 DTO Sınıfları**

DTO sınıfları, önyüz ile sunucu arasında veri alışverişini kolaylaştırmak amacıyla kullanılır. Bu sınıflarda doğrudan veri tabanı modellerine bağlı kalmadan yalnızca ihtiyaç duyulan alanların transfer edilmesini sağlayarak veri güvenliği, sadelik ve performans açısından avantaj sunar.

metin, ekran görüntüsü, menü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şekil 3.6: DTO sınıf dizini

Örnek olarak Şekil 3.6’da bulunan *DTO* sınıfından *WorkflowCreateDto* sınıfı aşağıda işleviyle beraber gösterilmektedir. Örnek olarak anlatılan bu sınıftaki yapı mantığı diğer *DTO* sınıflarında da uygulanmaktadır.

* *WorkFlowCreateDto* sınıfı:

1. Name: Oluşturulacak iş akışının adı. Kullanıcı tarafından tanımlanabilir ve iş akışını tanımlayıcı nitelik taşır.
2. Description: İş akışına dair açıklayıcı bilgidir. Kullanıcının iş akışını anlamasına yardımcı olur.
3. FromStatus: İş akışının başlangıç durumunun ID’sidir. *IssueStatus* tablosundaki bir kaydı temsil eder.
4. ToStatus: İş akışının hedef durumunun ID’sidir.Yine *IssueStatus* tablosundaki bir kaydı ifade eder.
5. IssueTypeIds: Bu iş akışıyla ilişkilendirilmek istenen veya birden fazla görev türünün ID listesidir. Çoktan-çoğa ilişkili yapılar için kullanılır.
6. Statuses: İş akışı boyunca tanımlanan adımları içeren liste yapısıdır. Her bir adım, *WorkflowStatusItem* tipinde ayrı bir nesne olarak temsil edilir. Bu yapı sayesinde iş akışının içeriği dinamik olarak modellenebilir.

### **3.3.5 Controller Sınıfları**

*Controller* sınıfları, istemciden gelen HTTP isteklerini karşılayan ve bu isteklere uygun iş mantığını çalıştırarak gerekli veriyi döndüren yapıdır. Uygulamanın *presentation* ve *business* katmanı arasındaki köprüyü oluşturur. Kullanıcıdan gelen talepleri yönlendirerek veri alışverişini ve sistemin doğru şekilde işlemesini sağlar.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şekil 3.7: Controller sınıf dizini

Örnek olarak Şekil 3.7’de görünen *controller* sınıfından *WorkflowController* sınıfı aşağıda işleviyle beraber gösterilmektedir. Örnek olarak anlatılan bu sınıftaki yapı mantığı diğer *controller* sınıflarında da uygulanmaktadır.

* *WorkflowController* sınıfı:
  + Kullanılan Veri Modelleri:

1. Workflow: Bir iş akışını temsil eder. Başlangıç durumu, bitiş durumu ve açıklama içerir.
2. WorkflowIssueType: Bir workflow’un hangi issue type’lar için geçerli olduğunu gösterir.
3. WorkflowCreateDto: Kullanılan veri transfer nesnesidir.
   * Metot Açıklamaları:
4. UpdateWorkflow: Güncellenecek *Workflow* veri tabanından bulunur. Eski *WorkflowIssueType* ilişkileri silinir.Yeni *IssueType* ilişkileri DTO’dan alınan verilere göre tekrar oluşturulur. Tüm değişiklikler kaydedilir.
5. DeleteWorkflow: ID ile eşleşen kayıt bulunur, silinir ve değişiklik veri tabanına kaydedilir.
6. GetWorkflowsByIssueType: *WorkflowIssueTypes* tablosundan *issue type* türüne göre filtreleme yapılır. İlgili workflow’lar ve onların status bilgileri dahil edilir. Sonuç özel olarak tanımlanmış *WorkflowDetailDto* ile döndürülür.
7. UpdateStatusOrder: Workflow’ların status sırası alınır, eski geçişler silinir. Yeni geçişler sıralamaya göre oluşturulur. Aynı *IssueType* ilişkileri yeni workflow’lara atanır.
8. DeleteTransition: Verilen *workflowId*, *fromStatusId* ve *toStatusId* bilgileri ile geçiş kontrol edilir. Eğer varsa silinir ve sonuç kullanıcıya döndürülür.

Bu *controller* sınıfı, sistemin iş akışı mantığını dinamik bir şekilde kontrol etmesini sağlar. Ayrıca workflow geçişlerinin kullanıcı arayüzü tarafından kolayca yönetilmesine olanak tanır. Böylece, sistemin ihtiyaç duyduğu özelleştirilebilir iş akışı altyapısı başarıyla sağlanmış olur.

### **3.3.6 Kimlik Doğrulama Süreci**

Bu proje kapsamında geliştirilen kimlik doğrulama modülünde, kullanıcıların şifre sıfırlama taleplerine onay kodunun e-posta yoluyla iletilmesi amacıyla *SMTP* yapılandırması kullanılmıştır. *SMTP*, bir istemciden e-posta sunucusuna mesajların iletilmesini sağlayan temel protokoldür ve ASP.NET Core uygulamalarında *SmtpClient* sınıfı üzerinden kolaylıkla entegre edilebilir.

* *SMTP* Yapılandırma Nesnesi:
  + SmtpConfigurations:

1. StmpHost: *SMTP* sunucusunun adresi
2. Port: Sunucu port numarası
3. Email: Gönderen e-posta adresi
4. Password: *SMTP* kimlik doğrulama şifresi
   * SMTP Client Nesnesinin tanmımlanması:
5. Host: Bağlanılacak *SMTP* sunucusu
6. Port: Güvenli iletişim için sıklıkla kullanılan 587 portu tercih edilmiştir.
7. Credentials: *SMTP* sunucusuna kimlik doğrulaması için gerekli e-posta adresi

* E-posta Mesaj Nesnesi:
  + E-postanın Gönderilmesi:

1. From: Gönderen e-posta adresi, *SMTP* yapılandırmasından alınan *Email* bilgisidir.
2. Subject: E-postanın konu başlığıdır.
3. Body: E-posta içeriği, kullanıcıya gönderilecek altı haneli doğrulama kodunu içermektedir.

* Doğrulama Kodunun Saklanması:
  + user.PasswordResetToken: Kullanıcıya e-posta ile gönderilen doğrulama kodunun, ilgili kullanıcı nesnesine atanmasıdır.
  + user.ResetTokenExpires: Doğrulama kodunun ne kadar süreyle geçerli olacağının belirlenmesidir.
  + await\_context.SaveChangesAsync: Yapılan değişikliklerin veri tabanına kalıcı olarak kaydedilmesini sağlamaktadır.

## **3.4 Front-End Geliştirme Süreci**

Günümüzde kullanıcı deneyimi, bir yazılım uygulamasının başarısında büyük ve kritik bir rol oynamaktadır. Bu nedenle, front-end geliştirme süreci, kullanıcıların uygulama ile doğrudan etkileşime geçtiği arayüzlerin tasarımı ve geliştirilmesini kapsamaktadır. Geliştirme süreci, sadece görsel tasarım unsurlarını değil; aynı zamanda kullanıcı dostu, erişilebilir ve performanslı bir deneyim sunmayı hedefler.

### **3.4.1 App.js açıklaması**

React.js uygulamasının merkezini oluşturan App.js dosyasıdır. Kullanıcı kimlik doğrulaması, sayfa yönlendirmeleri, tema yönetimi ve genel uygulama çerçevesi burada yapılandırılmıştır.

**Genel yapı ve Bağlam Sağlayıcılar**

AuthProvider: Kullanıcının kimlik doğrulama durumu gibi bilgileri sağlayan kimlik doğrulama sistemi

ProjectProvider: Projelerle ilgili global verilerin yönetildiği bağlam sağlayıcısıdır. Bu sağlayıcılar, API ile uygulama geneline veri iletimi sağlar.

**Tema Yönetimi**

MaterialUI kütüphanesi kullanılarak özelleştirilmiş bir tema oluşturulmuştur. Uygulama teması içinde renk paleti, yazı tipleri ve ağırlıkları, bileşen özelleştirmeleri bulunmaktadır.

**Sayfa Yönlendirme Sistemi (Routing)**

React Router ile uygulama içi sayfalar arasında yönlendirme yapılmaktadır.

* Public Routes:

/login: Sadece oturum açmamış kullanıcılar bu sayfaya erişebilir.

* Protected Routes:

/profile: Kullanıcı profili

/projects: Proje listesi

/project/:projectId: Proje detayı

/project/:projectId/boards: Projeye ait panolar

/issues ile /issues/:issueId: Sorun yönetimi

/sprints: Sprint yönetimi

/boards/:issueTypeId: Görev panoları

* Admin Routes:

/admin: Admin kontrol paneli

/admin/issue-types: Sorun türleri yönetimi

/admin/issue-status: Sorun durumları yönetimi

/admin/workflows: İş akışı yönetimi

/admin/custom-fields: Özel alanlar yönetimi

/admin/users: Kullanıcı yönetimi

/admin/email-settings: E-posta ayarları

Her rota, kullanıcı durumuna göre koruma altına alınmıştır.

### **3.4.2 Kullanıcı Arayüzü**

Bu bölümde, kullanıcıların sistemle etkileşim kurmasını sağlayan arayüz bileşenleri detaylı şekilde açıklanacaktır.

**Üst Menü Bileşeni**

Uygulamanın bileşeni, kullanıcı arayüzünde sayfanın üst kısmında sabit şekilde yer alan ve kullanıcıya temel navigasyon işlevlerini sunan önemli bir bileşendir. Bu bileşen, kullanıcıların projelere, görev listelerine ve yeni kayıt oluşturma ekranlarına hızlı erişimini sağlar. Sol üstte uygulama adı yer alırken, ortada gezinme butonları konumlandırılmıştır. Sağ üstte ise kullanıcı profilini temsil eden avatar, ayarlar ikonu ve açılır bir kullanıcı menüsü yer alır.

**Genel Arayüz**

Bu bileşen, kullanıcı arayüzünde oturum açmış kullanıcılar için uygulamanın temel iskeletini oluşturan ana yerleşim yapısını sunar. Bu bileşen, sayfanın üst kısmında sabit konumlandırılmış bir *AppBar* sol tarafta dinamik olarak genişleyip daralabilen bir *Sidebar* ve ortada ise kullanıcıya gösterilecek dinamik içeriği barındıran bir main bölümü içerir. Üst menüde uygulama adı, gezinme butonları,kullanıcı avatarı ve açılır profil menüsü yer alır. Kullanıcı avatarına tıklandığında açılan menüde profil sayfasına yönlendirme ve oturumu kapatma gibi işlemler gerçekleştirilir. *Sidebar* bileşeni, responsive olarak mobil uyumlu çalışmakta ve menü üzerinden *Dashboard*, *Issues*, *Sprints* gibi sayfalara yönlendirme sağlamaktadır. Ayrıca, *Boards* menüsüne özel olarak bir açılır menü mantığı da hem üst menüde hem de yan menüde desteklenmektedir. *useAuth* context’i üzerinden kullanıcı bilgileri ve oturum kapatma işlemi yönetilirken, *useNavigate* ve *useLocation* hook’ları sayesinde yönlendirme işlemleri gerçekleştirilir. Görsel yapı *Material UI* bileşenleriyle oluşturulmuş, tema uyumu ve kullanıcı deneyimi göz önünde bulundurularak sade, profesyonel ve sezgisel bir arayüz sağlanmıştır. Bu yapı, uygulama içerisindeki tüm sayfaların ortak görünüm ve davranışlara sahip olmasını sağlayarak genel kullanıcı deneyimini standartlaştırmaktadır.

**Dinamik Yan Menü**

Şekil 3.8’de görüldüğü üzere *Sidebar* bileşeni, kullanıcı arayüzünde sayfanın sol kısmında yer alan ve kullanıcıların farklı sayfalara kolayca erişmesini sağlayan dinamik bir yan menü yapısı sunar. Bu bileşen, *Dashboard*, *Projects*, *Boards*, *Sprints* ve *Settings* gibi ana sayfalara yönlendirme sağlar. *Boards* sekmesi altında yer alan *dropdown* menüsü, API üzerinden dinamik olarak alınan *issue* tiplerine göre oluşturulur. Ayrıca, en son oluşturulan projeler yine API'den çekilerek *Recent Projects* başlığı altında listelenir. Kullanıcı etkileşimlerine duyarlı olarak responsive çalışan bu yapı, mobil ve masaüstü ekranlarda uygun biçimde gösterim sunar. Görsel bütünlük ve kullanıcı deneyimi için *Material UI* bileşenleriyle tasarlanmış olup, sade ve fonksiyonel bir menü sistemi oluşturur.

ekran görüntüsü, yazılım, metin, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.8: Menü**

### **3.4.3 Admin Arayüzü**

Bu bölümde, *Admin* rolündeki kullanıcıların sistemle etkileşim kurmasını sağlayan arayüz bileşenleri detaylı şekilde açıklanacaktır.

**Admin- Üst Menü Bileşeni**

*AdminHeader* bileşeni, uygulamanın yönetim paneline özel olarak tasarlanmış bir üst navigasyon çubuğudur. Üst çubukta, admin modunu belirten bir etiket ve uygulamanın genel kullanıcı arayüzüne hızlı dönüş imkânı sunan “Back to App” butonu yer alır. Menü simgesi yan menüyü açma işlevi görür ve mobil uyumlu bir kullanım sağlar. Bu yapı, sade tasarımı ve *Material UI* bileşenleri sayesinde hem görsel tutarlılık hem de kullanıcı dostu bir deneyim sunar.

**Admin- Genel Arayüz**

*AdminLayout* bileşeni, uygulamanın yönetici paneli arayüzünü oluşturan ana iskelet yapısıdır. Aşağıdaki bileşenleri kapsar:

* Yan Menü: Yönetici panelindeki sayfalara geçiş için kullanılır. Menüde hem tekli hem de açılır menü öğeleri bulunur. Örneğin: *Issue Types, User Management, System Settings* gibi alt başlıklarla gruplanmıştır.
* Üst Bar: Sayfa başlığı, bildirim ikonu, kullanıcı avatarı ve profil/çıkış seçeneklerini barındırır.
* Mobil uyumluluk: Drawer mobil ekranlarda geçici olarak açılıp kapanabilir.
* Dinamik Başlık: Sayfaya göre başlık otomatik olarak değişir.
* İçerik Alanı: Admin sayfalarının içeriği bu *layout* içinde gösterilir.

**Admin-** **Dinamik Yan Menü**

*AdminSidebar* bileşeni, Şekil 3.9’da görüldüğü üzere yönetici paneli için yan menü görevini görür. Kullanıcının sayfalar arasında kolayca geçiş yapmasını sağlar.

Temel özellikleri:

* Menüde *Dashboard, Issue Configuration, User Management, System Settings* gibi başlıklar vardır.
* Açılır menü yapısıyla alt sayfalara geçiş yapılabilir.
* Seçili sayfa görsel olarak vurgulanır.
* Dar ve geniş görünüm arasında geçiş desteklenir.

metin, yazılım, ekran görüntüsü, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.9: Admin paneli**

### **3.4.4 API ile Haberleşme Görevi**

Sistemin farklı bileşenleriyle sunucu arasında veri alışverişini gerçekleştiren servis sınıflarını içermektedir. Bu işlemler listeleme, oluşturma, güncelleme, silme ve durum sırası düzenleme işlemleridir.

### **3.4.5 Sayfa Tasarımları**

Bu çalışma kapsamında geliştirilen sistemde çok sayıda sayfa tasarımı bulunmakla birlikte, bu bölümde yalnızca sistemin işleyişi açısından kritik öneme sahip olan temel sayfalara yer verilmiştir. Diğer sayfa tasarımları ise benzer yapıda oldukları için detaylı olarak açıklanmamış, genel yapıyı temsil eden örnek sayfalar üzerinden değerlendirme yapılmıştır.

**Projeler Sayfası**

Bu sayfa proje yönetiminin etkinliğini artırmaya yönelik modüler bir yaklaşım sergilemektedir. Sistemin temelini oluşturan Admin modülü, tüm uygulamanın merkezi yönetimini, kullanıcı yetkilendirmelerini ve genel sistem yapılandırmalarını ele alırken; Email Ayarları ve Profil Ayarları gibi modüller, sırasıyla sistem içi otomatik bildirimleri ve kullanıcıya özel tercihlerin yönetimini sağlamaktadır. Proje bazlı operasyonların kalbinde yer alan Projeler modülü ise, yeni projelerin oluşturulmasından mevcut projelerin izlenmesine kadar geniş bir yelpazede fonksiyonellik sunar. Şekil 3.10’da görüldüğü üzere bu ana modül, Görev Türleri, Görev Durumları ve İş Akışı gibi alt modüllerle bütünleşmiş çalışarak, her türlü iş kaleminin (görev, hata, özellik) tanımlanmasını, sınıflandırılmasını ve yaşam döngüleri boyunca ilerlemesini düzenler. Çevik metodolojilerle uyumlu olarak, Sprintler modülü sprint planlamasını ve takibini mümkün kılarken, Görev Panoları modülü görevlerin görselleştirilmesi sağlanmaktadır. Son olarak, uygulamanın özelleştirilebilirliğini ve raporlama yeteneklerini artıran *Özel Alanlar* modülü, kullanıcıların projelere veya görevlere kendi özel veri alanlarını eklemesine imkân tanırken; Kontrol Paneli modülü, tüm bu farklı modüllerden gelen verileri bir araya getirerek kullanıcılara genel proje durumu, kişisel görevler ve kritik metrikler hakkında hızlı ve özet bir bakış açısı sağlar. Bu modüler yapı, yazılımın esnekliğini ve ölçeklenebilirliğini destekleyerek, çeşitli proje yönetimi ihtiyaçlarına uyarlanabilir bir çözüm sunmaktadır.

yazılım, ekran görüntüsü, metin, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.10: Proje paneli**

**Görev Panosu Sayfası**

Sistem, yazılım geliştirme süreçlerinde görevlerin takibini kolaylaştırarak ekipler için daha düzenli ve verimli bir çalışma ortamı sunmayı amaçlamaktadır. Şekil 3.11’de görüldüğü üzere Görevler, “OPEN”, “IN PROGRESS” ve “CLOSED” olmak üzere üç farklı sütunda organize edilmiştir. Bu yapı, *Kanban* metodolojisi ile uyumlu olup, kullanıcıların görevlerini belirli aşamalardan geçirerek süreçlerini daha şeffaf bir şekilde yönetmelerine olanak tanır.

Sol tarafta yer alan navigasyon menüsü, kullanıcıların farklı proje panolarına hızlıca erişmesini sağlar. Burada Bug Board, Feature Board, Task Board ve Test Board gibi farklı görev kategorileri bulunmaktadır. Kullanıcılar, bu panolar aracılığıyla belirli görevleri farklı öncelik seviyelerinde takip edebilirler. Örneğin, orta kısımda yer alan “IN PROGRESS” sütununda "asd" adlı bir görev bulunmaktadır ve bu görev düşük öncelikli olarak tanımlanmıştır. Benzer şekilde, “CLOSED” sütununda yer alan "Test" adlı görev tamamlanmış ve bitiş tarihi 30 Haziran 2025 olarak belirlenmiştir.

Tasarım açısından bakıldığında, bu arayüz işlevsel bir yapıya sahiptir. Arayüzde, görevlerin durumu renkler ve sütunlarla açıkça belirtilmiş, kullanıcıların projelerini hızlı ve etkili bir şekilde yönetmelerine olanak tanınmıştır. Üst kısımda yer alan "Create Issue" düğmesi, kullanıcıların yeni görevler oluşturmasını sağlayarak sistemin dinamik yapısını desteklemektedir. Ayrıca Kanban Board ve List View seçenekleri, görevlerin farklı görünümlerle takip edilmesini mümkün kılmaktadır.

Genel olarak, bu görev panosu proje yönetimini iyileştirmek, ekip iş birliğini güçlendirmek ve görevlerin görselleştirilmesini sağlamak amacıyla tasarlanmıştır. Şeffaf ve anlaşılır bir arayüz ile kullanıcılar, hangi görevlerin açık, hangilerinin üzerinde çalışıldığını ve tamamlanan görevleri kolayca görebilmektedir.

metin, yazılım, bilgisayar simgesi, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.11: Görev paneli**

**Sorun Türleri Sayfası**

Dört farklı sorun türü yer almaktadır: hata, özellik, görev ve test. Şekil 3.12’de görüldüğü üzere her bir sorun türü için özel alanlar tanımlanabilmekte, bu da projenin gereksinimlerine göre esnek ve özelleştirilebilir bir yapı sunmaktadır. Yazılım geliştirme ekiplerinin iş akışlarını netleştirmelerine ve sorunları daha iyi kategorize etmelerine yardımcı olur, böylece her bir görev daha düzenli bir şekilde yönetilebilir.

Tasarım açısından görseldeki arayüz kullanıcı dostu ve minimalist bir yapıya sahiptir. Menü öğeleri açıkça tanımlanmış olup, her bir sorun türü için özel alanların özelleştirilmesine izin vermektedir. Bu tasarım, kullanıcının ihtiyacına göre sistemin kişiselleştirilmesine olanak tanır, bu da yazılım yönetiminde büyük bir avantaj sağlar. Modüler yapı, kullanıcıların yalnızca ihtiyaç duydukları alanları eklemelerine veya kaldırmalarına olanak tanıyarak gereksiz bilgi kirliliğini önler. Ayrıca, sistemin görsel tasarımı açık bir hiyerarşi sunarak kullanıcıların kolayca navigasyon yapmasını sağlar, böylece herhangi bir sorun türünü detaylı bir şekilde yapılandırmak mümkün hale gelir.

ekran görüntüsü, metin, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.12: Sorun paneli**

**User Listeleme Sayfası**

Şekil 3.13’te görüldüğü üzere sayfa, kullanıcıların rollerini belirleme, hesaplarını yönetme ve erişim seviyelerini düzenleme işlevlerini sağlamaktadır.

Tasarım açısından değerlendirildiğinde, arayüz kullanıcı dostu, minimalist ve sezgisel bir yapıya sahiptir. Sol üst köşede, toplam kullanıcı sayısı, aktif kullanıcılar, yönetici sayısı ve normal kullanıcı sayısı gibi genel istatistikler gösterilmektedir. Bu özet bölüm, yöneticilere sistemdeki kullanıcı dağılımı hakkında hızlıca bilgi sağlamak için oldukça işlevseldir. Ana bölümde ise her bir kullanıcının adı, iletişim bilgileri, rolü, durumu, oluşturulma tarihi ve yönetim seçenekleri gibi detaylara yer verilmiştir. Düzenleme ve silme gibi eylem düğmeleri sayesinde, yöneticiler kullanıcıları kolayca yönetebilir.

Bu tür bir sistem, kurumsal sistemlerde veri yönetimi, erişim kontrolü ve kullanıcı deneyimi gibi konular açısından incelenebilir. Kullanıcı yönetim panelleri, büyük ölçekli organizasyonların güvenlik stratejilerini belirlemelerine ve sistemlerini optimize etmelerine yardımcı olur. Ayrıca, tasarım prensipleri açısından bilgiye hızlı erişim, açık hiyerarşi ve özelleştirilebilir yönetim seçenekleri sunarak verimli bir kullanıcı deneyimi sağlar.

ekran görüntüsü, metin içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.13 Kullanıcı görüntüleme paneli**

**Sorun Durumu Sayfası**

Sayfada yer alan durumlar, bir görevin veya hatanın proje içindeki ilerleyişini ifade etmektedir. "Open", "In Progress", "Closed", "Testing" ve "Test Status" gibi aşamalar, sorunun farklı iş akışı adımlarında nasıl ele alındığını belirlemek için kullanılır. Bu sistem sayesinde, ekip üyeleri görevlerin ilerleme durumunu kolayca görebilir, böylece daha verimli bir iş yönetimi sağlanır.

Şekil 3.14’e baktığımızda, panelin arayüzü modüler ve özelleştirilebilir bir yapıya sahiptir. Kullanıcılar, her bir durumun sırasını, adını, açıklamasını ve rengini belirleyebilir. Özellikle renk kullanımı, sorunların görsel olarak kolay ayırt edilmesine yardımcı olur. Kullanıcı dostu bir düzen sağlanarak, yöneticiler ve ekip üyeleri karmaşık verileri daha hızlı anlayabilir ve iş süreçlerini düzenleyebilir. Ayrıca, her bir durumun yanında bulunan düzenleme ve silme seçenekleri, kullanıcıların iş akışlarını dinamik olarak değiştirmesine olanak tanır.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Şekil 3.14: Sorun durumu paneli**

**3.5 Azure API ile AWS SQL Server Entegrasyonu**

Projenin tamamlanmasının ardından, uygulamanın son kullanıcıya ulaşabilmesi için uygun bir sunucu altyapısında bulunması gerekmektedir. Bu aşamada, sunucunun yapılandırılması ve uygulamanın kesintisiz şekilde yayınlanması büyük önem taşır. Projemizde ölçeklenebilirlik, güvenlik ve erişilebilirlik gibi kriterler göz önünde bulundurularak Amazon Web Services (AWS) tercih edilmiştir. Bu bölümde, kullanılan sunucu yapılandırmaları, AWS servislerinin seçimi, dağıtım süreci ve canlı ortama geçiş aşamaları ele alınacaktır.

### **3.5.1 Azure Üzerinde CRM.API Yayınlanması**

*CRM.API* adlı *web API* uygulaması, Microsoft Azure bulut platformu üzerinde yayınlanmak üzere dağıtılmıştır. Dağıtım işlemi sırasında uygulama dosyaları Azure App Service ortamına aktarılmış ve gerekli yapılandırmalar tamamlanarak servis internet üzerinden erişilebilir hale getirilmiştir. Uygulamanın süreklilik ve güvenlik açısından kararlı şekilde çalışabilmesi amacıyla bağlantı, kimlik doğrulama ve ağ ayarları uygun biçimde yapılandırılmıştır.

metin, yazı tipi, ekran görüntüsü, logo içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### **3.5.2 AWS SQL Server Kurulumu ve Entegrasyonu**

Bu çalışmada kullanılan veri tabanı altyapısı, *AWS* platformu üzerinde yapılandırılmıştır. Bu kapsamda, *AWS* üzerinde bir windows sunucu oluşturulmuş ve ilgili sunucuya *SQL Server* kurulumu gerçekleştirilmiştir. Ardından, *emirhancinar\_CRM\_DB* adlı veri tabanı oluşturularak, sistemin ihtiyaç duyduğu veri işlemlerinin merkezi olarak yönetilmesi sağlanmıştır.

Veri tabanının, Azure ortamında barındırılan *CRM.API* uygulaması ile bütünleşmiş çalışabilmesi amacıyla uzak bağlantı senaryosu kurgulanmıştır. Bu kapsamda, Azure’daki API uygulamasının AWS üzerindeki veri tabanına güvenli bir şekilde erişebilmesi için gerekli bağlantı dizeleri tanımlanmış ve ağ güvenliğini sağlamak üzere çeşitli önlemler alınmıştır.

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Özellikle, AWS platformunun sunduğu *Security Groups* yapılandırması kullanılarak yalnızca belirli istemcilerin veri tabanına erişimine izin verilmiştir. Bu doğrultuda, Azure ortamında barınan uygulama sunucularına ait sabit iki IP adresine erişim izni tanımlanmış; bu sayede yetkisiz erişimlerin önüne geçilmiş ve yalnızca tanımlı istemcilerden güvenli veri alışverişi mümkün kılınmıştır.

Ayrıca, veri tabanına erişim sağlayacak kullanıcılar için farklı yetki seviyelerine sahip hesaplar oluşturulmuş ve *SQL Server Authentication* yöntemi aktif edilmiştir. Her bir kullanıcı için yalnızca ihtiyaç duyduğu veri tabanı nesnelerine yönelik okuma/yazma gibi yetkiler tanımlanarak, rol bazlı erişim kontrolü sağlanmıştır. Bu yaklaşım sayesinde, uygulama katmanının veri tabanı üzerindeki yetki sınırları net bir biçimde ayrıştırılmıştır.

Tüm bu yapılandırmalar neticesinde, Azure platformunda çalışan *CRM.API* uygulaması ile AWS üzerinde barındırılan SQL Server veri tabanı arasında hem güvenli hem de sürdürülebilir bir entegrasyon sağlanmıştır.

# **4. BULGULAR VE TARTIŞMA**

Bu bölümde, geliştirilen Proje Takip Sisstemi projesinin işlevselliği ve etkinliği üzerine elde edilen bulgular tartışılacaktır.Müşteri destek süreçlerine sağladığı katkılar ve sistemin genel olarak müşteri memnuniyetine olan etkileri değerlendirilecektir.

## **4.1 Sistem Performansı ve Kullanıcı Deneyimi**

Sistemin ilk aşamalarındaki testler, kullanıcıların şikayetlerini ve taleplerini hızlı bir şekilde iletebildiğini ve sistemin talepleri doğru şekilde kategorize ederek önceliklendirdiğini göstermiştir. Kullanıcılar, yazılı içerik ile taleplerini iletebilirken, sistemin aciliyet seviyelerini doğru bir şekilde belirleyip, talepleri buna göre sınıflandırması, hızlı çözüm süreci sağladı.

## **4.2 Taleplerin Takibi ve Yönetimi**

Yönetici ve çalışanlar için geliştirilen takip mekanizması, taleplerin hangi aşamada olduğunu net bir şekilde izlemeyi sağlamıştır. Her talebin durumu, önceliği ve çözüm süreci sistemde anlık olarak takip edilebilmektedir. Bu özellik, müşteri hizmetleri departmanındaki çalışanların talepleri etkili bir şekilde yönetmelerine olanak tanımaktadır.

## **4.3 Sistem Performansı ve Kararlılık**

Yapılan testler sonucunda sistemin temel işlevleri (kullanıcı girişi, şikayet oluşturma, atama yapma, durum güncelleme gibi) hatasız çalışmış ve kullanıcıya hızlı geri dönüşler sağlamıştır. React ile geliştirilen kullanıcı arayüzü sayesinde sayfa geçişleri minimum gecikmeyle tamamlanmıştır. .NET tabanlı API’ler ile veri erişimi verimli şekilde gerçekleştirilmiştir.

## **4.4 Sistemin Geliştirilmesi İçin Öneriler**

Yapay zekâ tabanlı öneri sisteminin eklenmesi için, sistemin daha fazla veri ile beslenmesi gerektiği ve daha gelişmiş makine öğrenmesi algoritmalarının kullanılmasının faydalı olacağı değerlendirilmiştir. Özellikle, kullanıcılar tarafından yapılan geri bildirimlerin ve taleplerin sisteme entegre edilmesi, yapay zekanın öğrenme sürecini hızlandırarak daha doğru çözüm önerileri sunmasını sağlayacaktır. Ayrıca, sistemin çok daha karmaşık sorunlar için de çözüm önerileri sunabilmesi adına, kullanıcı deneyimlerinden ve geçmişteki taleplerden elde edilecek verilerin analizinin daha derinlemesine yapılması gerekmektedir.

## **4.5 Karşılaşılan Sorunlar ve Çözümler**

Custom Field Yönetimi: Projede, kullanıcı tanımlı alanların yönetimi sırasında farklı kaynaklardan gelen verilerin yapısal tutarsızlıkları ile karşılaşılmıştır. Özellikle *fieldId*, *customFieldId* ve id gibi farklı adlandırmaların aynı veri alanını ifade etmesi hem okuma hem de yazma işlemleri sırasında karışıklıklara neden olmuştur. Bu durum, istemci tarafında API'den dönen verilerin tutarlı biçimde işlenmesini zorlaştırmıştır. Ancak bu zorluk, geliştirilen özel veri haritalama fonksiyonları sayesinde aşılmıştır. Bu fonksiyonlar, gelen verileri standart bir yapıya dönüştürerek sistemin diğer bileşenleriyle uyumlu hâle getirmiştir. Böylece hem arayüzdeki kullanıcı deneyimi korunmuş hem de veri bütünlüğü sağlanmıştır.

Veri Senkronizasyonu: Çok kullanıcılı ortamda eş zamanlı veri güncellemeleri sırasında oluşabilecek tutarsızlıkların önlenmesi amacıyla istemci tarafında etkili bir *state* yönetim stratejisi izlenmiştir. React uygulamasında, bileşenlerin güncel veriyi doğru şekilde yansıtabilmesi için *useState*, *useEffect* gibi temel işlevlerin yanı sıra özel veri yönetim yapıları da kullanılmıştır. Özellikle form tabanlı işlemlerde, kullanıcı girişleri ile sunucudan gelen veriler arasında senkronizasyon sağlanması amacıyla kontrollü bileşen yaklaşımı benimsenmiştir. Böylece veri çakışmaları önlenmiş, tutarsız kullanıcı deneyimi engellenmiş ve uygulamanın reaktif yapısı korunmuştur. Bu sayede sistemin güvenilirliği artırılmıştır.

# **KAYNAKLAR**

[1] J. Smith, "Customer Support Management Systems: An Overview," Journal of Business Technology, vol. 22, no. 3, pp. 55-60, 2019.

[2] A. Brown, "Digital Transformation in Customer Support," Customer Service Quarterly, vol. 18, no. 2, pp. 45-50, 2020.

[3] M. Liu, "AI in Customer Service: Revolutionizing Support Systems," International Journal of Artificial Intelligence, vol. 30, no. 4, pp. 123-130, 2021.

[4] S. Zhang, "Automated Solutions for Customer Complaints Using AI," AI and Business Solutions, vol. 15, no. 1, pp. 15-25, 2022.

[5] K. Wong, "Complaint Management and Prioritization in Customer Service," Journal of Customer Relations, vol. 8, no. 3, pp. 80-85, 2018.

[6] H. Turner, "JIRA in Project Management: A Case Study," Project Management Review, vol. 24, no. 4, pp. 112-118, 2021.

[7] P. Johnson, "Improving Customer Satisfaction through Efficient Support," Customer Experience Journal, vol. 10, no. 2, pp. 77-85, 2019.

[8] L. Carter, "How AI is Enhancing Customer Satisfaction in Service Industry," Journal of Business Research, vol. 17, no. 5, pp. 101-110, 2022.