



T.C.

ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ

MÜHENDİSLİK - MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2024-2025 Eğitim-Öğretim Yılı

152117123- ENGINEERING RESEARCH ON OPTIMIZATION ALGORITHMS (A)

TEZ RAPORU

Proje Başlığı

**Elektrikli Araçlarda Filo Yönetim Sistemleri için Araç Takip ve Performans İzleme Modüllerinin
Geliştirilmesi**

Projeyi Hazırlayanlar:

Berna ÇAKIR, 152120201041, Bilgisayar Mühendisliği

Şeyma COŞTUR, 152120201079, Bilgisayar Mühendisliği

Danışman:

Dr. Öğr. Üyesi Sinem BOZKURT KESER

| | |
|---|----|
| ŞEKİLLER DİZİNİ | 4 |
| TABLOLAR DİZİNİ | 5 |
| A. PLANLAMA | 7 |
| A.1.Özet(Abstract) ve Anahtar Kelimeler(Keywords) | 7 |
| A.2. Bilgi Gereksinim Belirleme, Problemin Tanımlanması | 8 |
| A.2.1. Amaç | 9 |
| A.2.2. Konu ve Kapsam..... | 10 |
| A.2.3. Literatür Özeti..... | 11 |
| A.3. Beklenen Fayda | 13 |
| A.3.1.Özgün Değer | 13 |
| A.3.2. Yayınlı Etki / Katma Değer..... | 14 |
| A.4. Yöntem..... | 14 |
| A.5. Araştırma Olanakları | 17 |
| A.6. Çalışma Takvimi..... | 17 |
| A.6.1. İş zaman Çizelgesi:..... | 17 |
| A.6.2. Kişi- İş Açıklaması..... | 17 |
| B.ANALİZ..... | 19 |
| B.1. Sistem Gereksinimlerini Ortaya Çıkarma Yöntem ve Teknikleri:..... | 19 |
| B.1.1. Yazılı Basılı Belge İnceleme | 19 |
| B.1.2. Yüz Yüze Görüşme | 19 |
| B.1.3. Anket | 19 |
| B.1.4. Gözlem | 25 |
| B.1.5. Prototip ve Hızlı Uygulama Tasarımı (Rapid Application Design – RAD)..... | 25 |
| B.1.6. Ortak Uygulama Tasarımı (Joint Application Design – JAD) | 25 |
| B.1.7. Veri Akış Şemaları | 26 |
| B.1.7.1 Kavramsal Veri Akış Şeması | 26 |
| B.1.7.2. Mantıksal Veri Akış Şeması | 26 |
| B.1.7.3 Fiziksel Veri Akış Şeması | 27 |
| B.1.8. Olay Tabloları, Durum Formları, İşlevsel Analiz Raporu, İş Akış Şeması | 27 |
| B.1.8.1. Olay Tabloları..... | 27 |
| B.1.8.2 Durum Formları | 28 |
| B.1.8.3. İşlevsel Analiz Raporu | 29 |
| B.2. Sistem Gereksinimler | 30 |
| B.2.1.İşlevsel Gereksinimler | 30 |
| B.2.2.Sistem ve Kullanıcı Arayüzleri ile ilgili Gereksinimler | 32 |
| B.2.3.Veriyle İlgili Gereksinimler:..... | 35 |
| B.2.4.Kullanıcılar ve İnsan Faktörü Gereksinimleri, Güvenlik Gereksinimleri: | 37 |
| B.2.5.Teknik ve Kaynak Gereksinimleri, Fiziksel Gereksinimler: | 37 |
| C.TASARIM | 38 |
| C.1. Sistem Tasarımı | 38 |
| C.2.Kullanıcı ve Sistem Ara yüzü Tasarımları | 40 |
| C.3.Test Tasarımı | 45 |
| C.3.1. Gereksinim analizlerinden teste yönelik hedeflerinin detaylandırılması | 45 |
| C.3.1.1. Araç Takip Modülü Test Hedefleri | 45 |
| C.3.1.2. Performans İzleme Modülü Test Hedefleri | 45 |
| C.3.2. Fonksiyonel Test Tasarımı | 46 |
| C.3.2.1. Birim (Unit) Testler:..... | 46 |
| C.3.2.2. Entegrasyon Testleri:..... | 48 |
| C.3.3. Performans Test Tasarımı | 50 |
| C.3.3.1. Araç Takip Modülü Test Senaryoları | 50 |
| C.3.3.2. Performans Modülü Test Senaryoları..... | 51 |

| | |
|---|----|
| C.4.Yazılım Tasarımı..... | 52 |
| C.4.1.Gereksinime Bağlı Tasarım Kalıpları Seçimi | 52 |
| C.4.2. UML Kullanarak Tasarımı Diyagramları Oluşturma..... | 52 |
| C.5.Veri Tabanı Tasarımı | 54 |
| C.6. Donanım Tasarımı | 54 |
| D.UYGULAMA..... | 54 |
| D.1. Geliştirilen Sistemin Sistem Tasarımlarını Karşılanmasıın Değerlendirilmesi..... | 54 |
| D.2. Kullanıcı ve Sistem Arayüzü Gerçeklemeleri..... | 55 |
| D.2.1 Araç Takip Modülü | 55 |
| D.2.1.1 Genel Yapı | 55 |
| D.2.1.2 Harita Alanı | 55 |
| D.2.1.3 Uyarı Paneli | 58 |
| D.2.1.4 Performans Paneli | 59 |
| D.2.2 Performans İzleme..... | 61 |
| D.2.2.1 Genel Yapı | 61 |
| D.2.2.2 Raporlar | 63 |
| D.3. Gerçeklenen Testler | 73 |
| D.3.1 Birim (Unit) Testleri | 73 |
| D.3.2 Entegrasyon Testleri..... | 76 |
| D.3.3 Performans Test Tasarımı | 78 |
| D.4. Yazılım/Veri Tabanı/Donanım Gerçeklemeleri | 82 |
| D.4.2 Araç Takip Modülü Yazılım/Veri Tabanı/Donanım Gerçeklemeleri | 82 |
| D.4.1.1 Enerji Tüketim Modeli | 82 |
| D.4.1.2 Menzil Tahmini..... | 84 |
| D.4.1.3 Kök Neden Analizi | 85 |
| D.4.1.4 Sumo ve FIWARE Entegrasyonu | 86 |
| D.4.2 Performans İzleme Modülü Yazılım/Veri Tabanı/Donanım Gerçeklemeleri | 87 |
| D.4.2.1 Docker ile Çevre Kurulumu..... | 87 |
| D.4.2.2 Dashboard ve Panel Oluşturma..... | 88 |
| D.4.2.3 iframe ile Panellerin Arayüze Gömülmesi..... | 89 |
| E. SONUÇ VE ÖNERİLER | 91 |
| REFERANSLAR | 93 |

ŞEKİLLER DİZİNİ

| | |
|--|----|
| Figure 1: Anket Sorusu 1 ve Sonuçları | 20 |
| Figure 2: Anket Sorusu 2 ve Sonuçları | 20 |
| Figure 3: Anket Sorusu 3 ve Sonuçları | 21 |
| Figure 4: Anket Sorusu 4 ve Sonuçları | 21 |
| Figure 5: Anket Sorusu 5 ve Sonuçları | 22 |
| Figure 6: Anket Sorusu 6 ve Sonuçları | 22 |
| Figure 7: Anket Sorusu 7 ve Sonuçları | 23 |
| Figure 8: Anket Sorusu 8 ve Sonuçları | 23 |
| Figure 9: Anket Sorusu 9 ve Sonuçları | 24 |
| Figure 10: Anket Sorusu 10 ve Sonuçları | 24 |
| Figure 11: Kavramsal Veri Akış Şeması | 26 |
| Figure 12: Mantıksal Veri Akış Şeması | 26 |
| Figure 13: Fiziksel Veri Akış Şeması | 27 |
| Figure 14: Araç Takip Modülü Sistem Tasarımı Şeması | 35 |
| Figure 15: Performans İzleme Modülü Sistem Tasarımı Şeması | 36 |
| Figure 16: Araç Takip modülü Arayüzü | 36 |
| Figure 17: Araç Takip Ekranı - Harita | 37 |
| Figure 18: Araç Takip Ekranı – Uyarı Paneli | 37 |
| Figure 19: Araç Takip Ekranı – Performans Paneli | 38 |
| Figure 20: Araç Takip Ekranı – Performans Paneli – Rota detay | 39 |
| Figure 21: Araç Takip Ekranı – Performans Paneli – Sipariş Detay | 39 |
| Figure 22: Performans İzleme Ekranı 1 | 40 |
| Figure 23: Performans İzleme Ekranı 2 | 40 |
| Figure 24: Performans İzleme Ekranı Raporlama | 41 |
| Figure 25: Test 1 | 42 |
| Figure 26: Test 2 | 42 |
| Figure 27: Test 3 | 43 |
| Figure 28: Test 4 | 43 |
| Figure 29: Test 5 | 44 |
| Figure 30: Test 6 | 44 |
| Figure 31: Test 7 | 45 |
| Figure 32: Test 8 | 45 |
| Figure 33: Use Case Diyagram | 48 |
| Figure 34: UML Diyagram | 49 |
| Figure 35: Araç Takip Genel Yapı | 51 |
| Figure 36: Sumo Harita | 51 |
| Figure 37: Araç Detay | 52 |
| Figure 38: sipariş detay | 52 |
| Figure 39: şarj istasyonu detay | 52 |
| Figure 40: Teslimat listesi | 53 |
| Figure 41: Harita Fiware | 53 |
| Figure 42: Uyarı paneli | 54 |

| | |
|---|----|
| Figure 43: Performans paneli | 55 |
| Figure 44: Performans detay | 56 |
| Figure 45: Sipariş panel | 56 |
| Figure 46: Sipariş detay | 57 |
| Figure 47: Performans izleme raporlar | 58 |
| Figure 48: Zaman aralığı seçme | 58 |
| Figure 49: Bitiş tarihi seçme | 58 |
| Figure 50: başlangıç tarihi seçme | 58 |
| Figure 51: Performans grafikler | 59 |
| Figure 52: Performans tablolar | 59 |
| Figure 53: Grafik bilgi | 59 |
| Figure 54: Şarj İstasyonu Rapor | 60 |
| Figure 55: Algoritma Performans Raporu | 61 |
| Figure 56: Karbon Kazancı Analiz Raporu | 61 |
| Figure 57: Müşteri Bazlı Rapor | 62 |
| Figure 58: Sürücü Performans Raporu | 63 |
| Figure 59: Sürücü Performans Raporu | 64 |
| Figure 60: Sistem Uyarı Raporu | 64 |
| Figure 61: Planlanan ve Gerçekleşen Rota Raporu | 65 |
| Figure 62: Sürücü Başarı Raporu | 66 |
| Figure 63: Sistem Performans Raporu | 66 |
| Figure 64: Genel Sipariş Raporu | 67 |
| Figure 65: Araç-Rota Performans Raporu | 68 |
| Figure 66: Filo Operasyon Performans Raporu | 68 |
| Figure 67: Araç Karşılaştırma Raporu | 69 |
| Figure 68: Gerçeklenen Test 1 | 69 |
| Figure 69: Test sonuç 1 | 70 |
| Figure 70: Gerçeklenen Test 2 | 70 |
| Figure 71: Gerçeklenen Test 3 | 71 |
| Figure 72: Gerçeklenen Test 4 | 71 |
| Figure 73: Test sonuç 2 | 72 |
| Figure 74: Gerçeklenen Test 5 | 72 |
| Figure 75: Gerçeklenen Test 6 | 73 |
| Figure 76: Gerçeklenen Test 7 | 74 |
| Figure 77: Test sonuç 3 | 74 |
| Figure 78: Gerçeklenen Test 8 | 74 |
| Figure 79: Gerçeklenen Test 9 | 75 |
| Figure 80: Gerçeklenen Test 10 | 75 |
| Figure 81: Test sonuç 4 | 76 |
| Figure 82: Gerçeklenen Test 11 | 76 |
| Figure 83: Gerçeklenen Test 12 | 77 |
| Figure 84: Test sonuç 5 | 77 |
| Figure 85: Model sonuçları | 79 |

| | |
|--|----|
| <i>Figure 86: model kullanım</i> | 79 |
| <i>Figure 87: menzil kullanım</i> | 80 |
| <i>Figure 88: Kök Neden Analizi</i> | 81 |
| <i>Figure 89: sumo kullanım</i> | 82 |
| <i>Figure 90: Sipariş Parse</i> | 82 |
| <i>Figure 91: docker</i> | 83 |
| <i>Figure 92: grafana</i> | 84 |
| <i>Figure 93: grafana dashboard</i> | 84 |
| <i>Figure 94: grafana raporlar</i> | 84 |
| <i>Figure 95: grafana grafik edit</i> | 85 |
| <i>Figure 96: grafana paylaşma</i> | 86 |
| <i>Figure 97: Grafana arayüze ekleme</i> | 86 |

TABLOLAR DİZİNİ

| | |
|---|----|
| <i>Tablo 1: Projede Kullanılacak Mevcut Makine - Teçhizat Listesi</i> | 16 |
| <i>Tablo 2: İş Zaman Çizelgesi</i> | 17 |
| <i>Tablo 3: Kişi – İş Açıklaması</i> | 17 |
| <i>Tablo 4: Olay Tabloları</i> | 27 |
| <i>Tablo 5: Durum Formları</i> | 28 |
| <i>Tablo 6: İşlevsel Analiz Raporu</i> | 29 |

A. PLANLAMA

A.1.Özet(Abstract) ve Anahtar Kelimeler(Keywords)

ÖZET

Son zamanlarda elektrikli araçların kullanımı, çevre dostu ve sürdürülebilir olmaları nedeniyle birçok filo yönetim sistemi tarafından tercih edilmektedir. Ancak bu araçların filo sistemlerine entegrasyonu, geleneksel yöntemlerden farklı olarak yeni izleme, analiz ve yönetim gereksinimlerini beraberinde getirmektedir. Elektrikli araçların etkin bir biçimde izlenmesi, enerji verimliliğinin sağlanması, menzil durumunun yönetilmesi ve performanslarının değerlendirilmesi bu alanda önem kazanmıştır. Bu kapsamda geliştirilen çalışmada, elektrikli araç filolarına yönelik bir yönetim sisteminde yer alan Araç Takip Modülü ve Performans İzleme Modülü tasarlanmış ve uygulanmıştır. Araç Takip Modülü; araçların anlık konum, hız, enerji tüketimi ve menzil verilerinin izlenmesini sağlayarak operasyonel süreçlerin etkinliğini artırmayı hedeflemektedir. Bu veriler doğrultusunda araçların rota durumu, şarj gereksinimi, aracın enerji tüketimini etkileyen faktörler ve teslimat süreçlerine ilişkin analizler gerçekleştirilebilmektedir. Performans İzleme Modülü ise, araçlardan ve sistemden elde edilen verilerin bütüncül bir şekilde analiz edilmesini sağlayarak, filo yönetimi performanslarının zamana bağlı olarak değerlendirilmesine imkân tanımaktadır. Geliştirilen bu modüller elektrikli araç filolarında yönetimsel verimliliği artırmak, enerji tüketimini optimize etmek ve sürdürülebilir ulaşım hedeflerine katkı sunmak amacıyla özgün bir yazılım çözümü sunmaktadır. Sunulan sistem, kullanıcı odaklı yapısı ve analiz yetenekleriyle benzer çalışmalardan ayışarak genişletilebilir, esnek ve veri temelli bir yaklaşım ortaya koymaktadır.

Anahtar Kelimeler: Elektrikli Araçlar, Filo Yönetimi, Araç Takibi, Performans İzleme, Enerji Tüketimi, Menzil Tahmini, Makine Öğrenmesi, Grafana

ABSTRACT

Recently, the use of electric vehicles has been increasingly preferred by many fleet management systems due to their environmentally friendly and sustainable nature. However, the integration of these vehicles into fleet systems brings new monitoring, analysis, and management requirements that differ from traditional methods. Effective monitoring of electric vehicles, ensuring energy efficiency, managing range status, and evaluating performance have become crucial in this field. In this context, a Vehicle Tracking Module and a Performance Monitoring Module have been designed and implemented within a management system tailored for electric vehicle fleets. The Vehicle Tracking Module aims to enhance the efficiency of operational processes by enabling the monitoring of real-time data such as vehicle location, speed, energy consumption, and range. Based on this data, analyses can be conducted regarding route status, charging needs, factors affecting energy consumption, and delivery processes. The Performance Monitoring Module allows for a comprehensive analysis of data obtained from vehicles and the system, enabling the evaluation of fleet management performance over time. These developed modules offer a unique software solution designed to improve management efficiency in electric vehicle fleets, optimize energy consumption, and contribute to sustainable transportation goals. The presented system distinguishes itself from similar studies with its user-centric structure and analytical capabilities, offering a scalable, flexible, and data-driven approach.

Keywords: Electric Vehicles, Fleet Management, Vehicle Tracking, Performance Monitoring, Energy Consumption, Range Estimation, Machine Learning, Grafana

A.2. Bilgi Gereksinim Belirleme, Problemin Tanımlanması

Günümüzde sürdürülebilir ve çevre dostu ulaşım çözümlerine yönelik artan ilgi, filo yönetim sistemlerinde elektrikli araç kullanımını yaygınlaştırmıştır. Ancak batarya kapasitesinin sınırlı olması, şarj istasyonlarının yeterince yaygınlaşmaması menzil ve şarj durumu (SoC, State of Charge) açısından bazı kısıtlar doğurmaktadır. Bu durum, özellikle yoğun dağıtım ve teslimat süreçlerinin yönetildiği filo sistemlerinde operasyonel planlamaları karmaşıklığa neden olmaktadır. Elektrikli araçlara sahip bir filonun etkin biçimde yönetilebilmesi için, araç durumu ve dağıtım süreci gibi operasyonel süreçlerin sürekli izlenmesini ve analiz edilmesini sağlayan bir filo yönetim yazılımına ihtiyaç duyulmaktadır. Bu tür bir yazılımın modüler yapıda tasarlanması, birçok farklı işlevin bağımsız şekilde yönetilmesine ve güncellenebilmesine olanak tanımaktadır. Bu proje kapsamında ise, elektrikli araç filo yönetiminin ilgili gereksinimleri karşılamak üzere Araç Takip Modülü ve Performans İzleme Modülü geliştirilmesi hedeflenmiştir. Bu iki modül, filo yönetimini kolaylaştırmak ve etkili bir kullanım sunmak amacıyla planlanmıştır. Araç Takip Modülü, araçların anlık olarak izlenmesine olanak tanımakta olup; enerji tüketimi, şarj durumu ve kalan menzil gibi verilerin belirli aralıklarla toplanmasını ve görselleştirilmesini sağlamaktadır. Bu verilerin anlık takibi filo yöntemlerinin etkinliğini artırmaktadır. Ayrıca Elektrikli araçların kullanımında şarj tüketim durumunun takip edilmesi oldukça kritik bir durumdur bunun sebeplerinden biri aracın gideceği menzilin şarj durumuna bağlı olmasıdır. Bu nedenle, projede bu faktörlerin şarj tüketimine olan etkilerinin belirlenmesi ve ağırlıklarının hesaplanması amacıyla makine öğrenmesi tabanlı bir analiz yöntemi uygulanmaktadır. Performans modülünde ise filo yönetiminden toplanan veriler kullanılarak analizler gerçekleştirilebilmektedir. Bu analizler, Grafana kullanılarak oluşturulan grafik ve tablolarla desteklenecek ve kullanıcı arayüzünde görselleştirilerek kullanıcıların erişimine sunulacaktır. Sonuç olarak, bu çalışma ile geliştirilen modüller; elektrikli araçların filo yönetimi sürecini destekleyen bir yazılım çözümü sunmayı amaçlamaktadır.

A.2.1. Amaç

Elektrikli araçlar, sürdürülebilirlik, enerji verimliliği ve düşük karbon salınızı gibi çevresel ve ekonomik avantajları nedeniyle son yıllarda kullanımı giderek artmaktadır. Bu araçların filo operasyonlarında kullanımı ise beraberinde çeşitli yönetimsel ihtiyaçlar doğurmaktadır. Sınırlı menzil, şarj durumunun kritik öneme sahip olması ve operasyonel süreçlerdeki zamanlama hassasiyeti, elektrikli araç filolarının etkin bir şekilde yönetilmesini zorlaştıran ana faktörlerdir. Bu çalışma, söz konusu yönetimsel ihtiyaçlara çözüm sağlamak amacıyla, bir elektrikli araç filosuna odaklanan Araç Takip Modülü ve Performans İzleme Modülünün geliştirilmesini amaçlamaktadır. Araç Takip Modülü, elektrikli araçlara ait verilerin anlık olarak izlenmesini sağlayarak; araçların operasyonel durumu doğrultusunda değerlendirmeler yapılmasına olanak tanır. Bu modül aracılığıyla araçların SoC (şarj durumu), menzil ve enerji tüketimi gibi veriler doğrultusunda; dağıtım süreçleri içerisinde teslimat yapılabilmesi ve enerji yönetiminin etkin şekilde gerçekleştirilmesi hedeflenmektedir. Ayrıca, enerjiye etki eden faktörlerin birbirleriyle oranlarının belirlenmesi ile sistemin öngörü kapasitesi artırmakta ve daha doğru planlamalar yapılmasına imkân sağlanmaktadır. Araç Takip Modülü kapsamında aynı zamanda, araçlardan gelen uyarı, hata ve bilgilendirme mesajları sistematik olarak sınıflandırmakta; sürücü, araç, rota, teslimat, performans ve sistem kaynaklı problemler kök neden analizi perspektifiyle değerlendirilerek kullanıcıya karar destek bilgileri sunulmaktadır. Bu yaklaşım, sistemin erken müdahale ihtiyalini yükseltmektedir ve olası operasyonel engellerin önlenmesine katkı sağlamaktadır. Performans İzleme Modülü ise, filo sisteminden elde edilen tüm verilerin zaman bazlı olarak analiz edilmesine imkan sunmaktadır. Farklı zaman dilimlerine (günlük, haftalık, aylık, yıllık) göre verilerin analizi

yapılarak; araç, sürücü, rota, şarj, şarj istasyonu, müşteri ve sipariş performanslarının veriye dayalı ve tarafsız olarak değerlendirilmesi mümkün hale gelmektedir. Bu analizler, Grafana kullanılarak grafik ve tablolarla görselleştirilmekte ve kullanıcı arayüzü üzerinden erişilebilir şekilde sunulmaktadır. Bu sayede sistemin avantajlı ve dezavantajlı yönleri detaylı olarak belirlenebilmekte, iyileştirme yapılacak alanlar tespit edilmekte ve ileri dönemli stratejiler oluşturulabilmektedir. Bu kapsamda geliştirilen modüller, elektrikli araç filolarının daha verimli, sürdürülebilir ve veri odaklı bir şekilde yönetilmesine katkı sağlamayı hedeflemektedir.

A.2.2. Konu ve Kapsam

Günümüzde sürdürülebilir ve temiz ulaşım çözümlerine yönelik artan ilgi sonucunda filo yönetiminde elektrikli araçların kullanımı yaygınlaşmaktadır. Fakat batarya kapasitesinin sınırlı olması, şarj istasyonlarının yeterince yaygınlaşmaması hem menzil hem şarj durumu(SoC) açısından bazı kısıtlamalarla karşılaşmasına sebep olabilmektedir. Bu durum elektrikli araçlardaki filo yönetimini zorlaştırmaktadır. Bir elektrikli araç filo dağıtımında; filonun yönetimi, etkin bir dağıtım ve durumu analizleyebilmek için bir filo yönetim yazılımı şarttır. Yazılımın farklı modüllerden oluşması filo yönetim sisteminde bakım ve geliştirme süreçleri kolaylaştırılmış esnek bir arayüzün olmasını sağlamaktadır. Bu çalışma proje geliştirilen sistem aşağıdaki iki ana modülten oluşmaktadır.

Araç takip modülü:

Bu modül, filoya ait elektrikli araçların anlık olarak izlenmesini ve ilgili verilerin analiz edilmesini sağlamaktadır. Modül, aşağıdaki işlevleri kapsamaktadır

- Araçlara ait anlık konum, hız, şarj durumu, enerji tüketimi ve kalan menzil gibi verilerin izlenmesi,
- Araçtan elde edilen verilerin belirli periyotlarla kaydedilerek izlenebilirliğin sağlanması,
- Dağıtım sürecine ilişkin rota durumu ve sipariş akışının anlık olarak takip edilmesi,
- Araçların enerji tüketim durumuna bağlı olarak menzilin tahmin edilmesi,
- Menzile etki eden faktörlerin (örneğin yol eğimi, hız, yük, çevresel koşullar) belirlenmesi ve bu faktörlerin ağırlıklarının makine öğrenmesi yöntemleri ile analiz edilmesi,
- Sistemde meydana gelen durumlara yönelik uyarı, hata ve bilgilendirme mesajlarının kullanıcıya iletilmesi,
- Bu mesajların kök neden analizi yaklaşımıyla değerlendirilerek yöneticilerin olası aksaklıklara hızlı ve etkili müdahale edilmesine olanak sağlanması.

Performans İzleme Modülü:

- Bu modül, filo sisteme ait tüm verilerin belirli zaman aralıklarında analiz edilmesini ve geçmişe yönelik performans değerlendirmelerinin yapılmasını sağlamaktadır. Modül, aşağıdaki işlevleri kapsamaktadır.
- Araç, sürücü, müşteri, sistem, şarj istasyonu ve sipariş bilgileri gibi toplanan verilerin performans raporlarına dönüştürülmesi,
- Günlük, haftalık, aylık ve yıllık zaman dilimlerine göre verilerin istatistiksel olarak analiz edilmesi,
- Kullanıcıların istenilen gün ve saat aralıklarına göre detaylı filtreleme yaparak verileri analiz edebilmesi,
- Filo operasyonlarında kullanılan sipariş sayısı, menzil, ortalama hız, şarj istasyonu kullanımı sürücü gibi metrikler üzerinden performans eğilimlerinin değerlendirilmesi
- Elde edilen performans sonuçları doğrultusunda sistemsel aksyonların alınmasına imkân tanınması

- Uzun vadeli stratejik kararlar için geçmiş verilerin kullanılarak karar destek süreçlerine katkı sunulması.

Bu bağlamda geliştirilecek sistem, elektrikli araçların oluşturduğu filolarda karşılaşılan zorlukların üstesinden gelinmesini ve filo yönetiminin daha verimli hale getirilmesini amaçlamaktadır.

A.2.3. Literatür Özeti

Literatürde, filo yönetim sistemlerinde elektrikli araç takibi ve performans izleme ile ilgili birçok çalışma yer almaktadır. Elde edilen araştırma sonuçları bu bölümde sunulmaktadır.

Meenambika ve ark. [1], "Fleet Guard" adında ileri teknolojiye sahip bir filo yönetimi ve araç izleme projesini dart dilinde Flutter kullanarak geliştirmiştir. Çalışmada, gerçek zamanlı takip için GPS ve IoT tabanlı sensörler kullanılarak araçların anlık konumu, hızı ve rota bilgileri anlık alınmaktadır. Alınan bu veriler merkezi bir veritabanına kaydedilmektedir. Ayrıca raporlama modülü ile araç bakım ve raporlama da projeye dahil edilmiştir. Bu çalışmada yenilikçi özellikler olarak gerçek zamanlı basıncı izleme, öngörücü bakım ve sürücü davranış analizi de sunmaktadır.

Husak ve ark. [2] gerçek zamanlı araç takibi için akıllı bir bilgi sistemi üzerinde çalışmışlardır ve bu çalışmanın file yönetimi ile araç izleme alanında önemli bir katkı sağlayacağını düşünmüştür. Anlık olarak araçtan gelen veriler üzerinde çalışmışlar ve gelişmiş veri işleme sunmuşlardır. verileri filtreleme yöntemleri kullanarak hatalı verileri elimine etmişlerdir. Ayrıca sistemde, her aracın taşınan yük kapasitesi ve toplam mesafe bilgileri analiz edilmektedir. Tahmine dayalı bakım konusuna da araştırmalarında yer vermişlerdir.

Martins ve ark. [3], elektrikli araçlardaki şarj işlemleri zorlukları ve akıllı izleme sistemlerini ele almışlardır. Elektrikli araç şarj istasyonlarının verimli bir şekilde yönetilmesi ve optimizasyonu için kullanılan gelişmiş algoritmalar ve teknolojilere yer vermişlerdir. Ayrıca projede araçlar anlık izlemede potansiyel sorunlarda veya anomal durumlarda anlık uyarı vermektedirler. IoT sensörlerini, uç bilişimi ve bulut hizmetlerini kullanarak gerçek zamanlı izleme, öngörücü bakım ve kullanıcı davranışının duyarlı analizini yapmaktadır.

Reddy ve ark. [4], çalışmalarında filo performansını sürekli izlemek ve analiz etmeyi amaçlamışlardır. Bunun için IoT sensörleri, GPS ve Büyük Veri analitiği kullanan çözümler sunarak var olan sorunları ele almışlardır. Sistemlerinde Gömülü GPS modülleri ve veri analitiği kullanarak araçların gerçek zamanlı, konum, hız ve yakıt seviyelerini izleme ve sürüş davranışlarını değerlendirme üzerine çalışmışlardır. Bulut altyapısı ile veri toplama, depolama, işleme ve bu verilere kolay erişip ölçeklenebilirliğini sağlamışlardır. Sistemde ek özellikler olarak öngörücü araç bakımı, rota sapmaları gibi kısımlar için uyarı sistemleri içermektedir.

Kauffmann ve ark. [5], filo yönetimi performans izleme sistemleri tasarım ve uygulanabilirliği ile ilgili bir araştırma yapmışlardır. Araştırmaları filo yönetimi sistemlerinin verimliliğini artırmak için sistemin performans izlemesinin önemini ele almaktadır. Bunun için gerçek zamanlı takibinde performans için gerekli parametreleri araçların yakıt verimliliği, kullanım süreleri, bakım gereksinimleri, kilometre performansı ve araç duruş süreleri olarak tanımlamışlardır. Bu performans verilerinin toplanması ve analizi için GPS, IoT sensörleri ve veri analitiği teknolojilerini kullanmışlardır. Özette araştırmaları, filo yöneticilerinin araç performansının optimizasyonu, bakım süreçleri iyileştirmeleri ve verimlilik sağlamak için detaylı bir izleme sisteminin ihtiyacına katkı sağlamıştır.

Shukla ve ark. [6], GPS tabanlı takip ve navigasyon sistemlerinin, GSM/GPRS gibi iletişim standartları, ek sensörler ve ana kontrolörlerle entegre edildiğinde filo yönetimi için nasıl verimli bir çözüm sunmaktadır. Makale araç takip sistemlerinin tasarımını ve kullanımı ile ilgili detayları da içermektedir. Çalışmada takip süreçleri aktif gerçek zamanlı ve pasif takip olarak iki ana kategoride ele alınır. Pasif takip için araç konum ve durum bilgileri sürekli izlenir kaydedilir ve sonra yazılım uygulamasında gösterir. Aktif takip için ise anlık olarak veriler GSM/GPRS üzerinden bir iletişim ağı aracılığı ile takip edilir. Bu iki sistemin filo yönetiminde farklı ihtiyaçlara yönelik uygulanabilirliği de çalışmada yer almaktadır.

Huang ve ark.[7] , elektrikli araçlarda menzil tahmini için Destek Vektör Regresyonu (SVR, Support Vector Regression) modelini kullanmışlardır. Bu modeli oluştururken Elektrikli araç (EV) verilerinde yer alan sürüs menzili, motor sıcaklığı ve toplam voltaj gibi değişkenler kullanılmıştır. EV menzil tahmini için özellik şarj durumu (SoC) etki eden faktörleri seçen özellik çıkarmada Random Forest kullanılmıştır. Gerçek sürüs verilerini birleştirerek destek vektör regresyonuna dayalı EV'lerin sürüs menzili tahmin modeli oluşturulmuştur.

Hasib ve ark. [8], EV tahmini için farklı çoklu doğrusal regresyon (MLR, Multiple Linear Regression), Destek Vektör Regresyonu (SVR, Support Vector Regression), Poinomsal Regresyon (PR, Polynomial Regression), Random Forest Regression ve Simple Linear Regression makine öğrenimi modellerini Elektrikli araçların menziline etki eden faktör tahmininde kullanmışlardır. Özellikle çoklu doğrusal regresyon modelinin diğer modellerden daha başarılı olduğu, ancak daha fazla optimizasyona ihtiyaç duyduğu vurgulamışlardır.

Witvoet ve ark.[9], elektrikli araçlarda SoC(State of Charge) ve tahmini menzil için AutoML araçlarından AutoGluon kullanarak bir model geliştirmiştir. Bu model, batarya durumu, rota verileri ve akım-voltaj değerleri kullanmıştır. batarya SOC ve sağlık durumu (SOH) tahminlerinde Yinelemeli Sinir Ağı (RNN, Recurrent Neural Networks), Uzun-Kısa Vadeli Bellek (LSTM, Long Short-Term Memory), Transfer Öğrenme (TL, Transfer Learning) modellerini kullanmıştır. Makine öğrenimi framework'leri olarak TensorFlow, PyTorch ve AutoGluon karşılaştırılmıştır. AutoGluon, kullanıcı dostu olması ve model otomasyonunu sağlama nedeniyle tercih edilmiştir.

Huang ve ark.[10], çalışmalarında istatistiksel, tahmine dayalı ve nedensel analizler kullanarak faktörlerin enerji tüketimi üzerindeki etkisi detaylı incelemiştir. İstatistiksel yöntem olarak Spearman korelasyon analizini, tahmine dayalı yöntem için XGBoost + SHAP(SHapley Additive ExPlanations) ve nedensel için de DML (Double/Debiased Machine Learning) + BIB(Bootstrap-of-little-bags) yöntemlerini kullanmıştır. Çalışmalarının sonucunda enerji tüketimine üzerinde en büyük etkisi olan faktör %37.75 mesafe olarak bulunmuştur. Hız faktörü için de en iyi aralığı 20-50km/s olarak hesaplamışlardır.

Mediouni ve ark.[11], elektrikli araçlarda enerji tüketimine etki eden faktörleri incelemiştir. Enerji tüketimini tahmin etmek için analitik, istatistiksel ve makine öğrenimi tabanlı üç modelden bahsetmiştir. Faktörleri içsel ve dışsal olarak ikiye ayırmışlardır. İçsel faktörler için araç teknolojisi faktörleri ve sürücü davranışları faktörleri almışlardır. Dışsal faktörler için de iklim koşulları, yol topografyası ve Yol Koşullarını ele almışlardır. Ayrıca makalelerinde rejeneratif frenleme ile enerji geri kazanımı üzerinde durmuşlar ve frenleme esnasında bataryaya geri dönen enerjinin, yavaşlama seviyesine bağlı olarak değiştğini incelemiştir. Testleri sonucunda da elektrikli araçların frenleme verimliliğinin %50-60 seviyelerinde olduğunu bulmuşturlar.

A.3. Beklenen Fayda

A.3.1. Özgün Değer

Elektrikli araçlara özgü batarya kapasitesi, istasyon altyapısı, menzil kaygısı gibi kısıtlar geleneksel filo yönetim sistemlerinin elektrikli araçların ihtiyaçlarını karşılamakta yetersiz kaldığı alanlardır. Bu nedenle projede savunulan görüş, elektrikli araçların gereksinimlerine uygun, anlık veri işleyebilen, esnek, modüler çözümlerin geliştirilmesi gerektiğini yönündedir. Bu projede yer alan menzil tahmini, enerji tüketimi, enerji tüketimine etki eden faktörler, sürücü, rota, araç gibi değişkenlerin kapsamlı şekilde izlenmesi; filonun verimini ve sürdürülebilirliğini artıracağı gibi bu alandaki yazılımlara da katkı sağlayacaktır. Özellikle enerji tüketimine etki eden faktörlerin yüzdeslerinin makina öğrenmesi ile belirli aralıklarla anlık olarak izlenebilmesi ve görselleştirilmesi mevcut sistemlerde nadir kullanılan bir yaklaşımdır. Uyarı, hata, bilgilendirme mesajlarının gerçek zamanlı olarak alınması kritik durumlara karşı hızlı aksiyon alınmasını sağlayacaktır ve operasyonel kesintileri önüne geçecektir. Verilerin görselleştirilmesi ve analizlenmesi kullanıcılar için gelecek stratejileri belirlenmesini kolaylaşacaktır. Kullanıcılar, gerekli grafik ve tabloları oluşturarak verileri görselleştirebilir ve analiz edebilmektedir; bu da gelecek stratejilerin belirlenmesini kolaylaştırmaktadır. Bu çalışma tamamlandığında, elektrikli araçlara özgü filo yönetim yazılımlarının geliştirilmesine yönelik literatürdeki sınırlı örnekleri genişleterek fayda sağlayacaktır.

A.3.2. Yaygın Etki / Katma Değer

Projenin gerçekleştirilmesi, elektrikli araçlardan oluşan filo yönetiminde karşılaşılan araç takip ve performans izleme problemlerine birçok yönden katkı sağlayacaktır. Özellikle menzil tahmini, enerji tüketimi faktörleri, rota verilerinin analiz edilmesi sayesinde operasyonel verimlilik artacaktır; bu durum lojistik süreçlerde enerji maliyetlerini azaltacak ve işletmelerin operasyonel giderlerini düşürerek piyasada avantaj elde etmelerini sağlayacaktır. Projenin bilimsel katma değeri ise, beş farklı modülün birlikte çalıştığı, enerji tüketimine etki eden faktörlerin makina öğrenmesi ile değerlendirildiği modüller, izlenebilirliği yüksek ve veriye odaklı bir yazılım sistemi geliştirilmesinden kaynaklanmaktadır. Bu sistem; enerji verimliliği ve sürdürülebilir filo yönetim hedefleri kapsamında literatürde referans oluşturabilecek bir model niteliğindedir. Geliştirilen sistem hem filo yöneticilerine hemde bu alanda çalışan kişilere katkı sağlayacaktır.

A.4. Yöntem

Bu proje kapsamında geliştirilecek yazılım araç takip ve performans izleme modüllerinin de bir parçası olduğu bir filo yönetim sistemidir. Her bir modülün geliştirme süreci belirli parametrelere dayalı olarak planlanmış ve uygulanacak yöntemler aşağıda adım adım açıklanmıştır:

Araç Takip Modülü:

Adım 1: Gerçek Zamanlı Araç Verilerinin Toplanması

Elektrikli araçlardan, CAN bus ağı, GPS, sıcaklık, eğim ve batarya yönetim sisteminden gelen veriler toplanır ve ön işleme tabi tutulur. Daha sonra bu veriler, haberleşme protokollerini aracılığıyla FIWARE platformuna aktarılır. MQTT protokolü, kritik verilerin düşük gecikme ile FIWARE ortamına iletilmesini sağlar.

Adım 2: Anlık İzleme

Adım 2.1: Gerçek Araç

FIWARE platformuna iletilen anlık veriler için sistem içerisinde bir MQTT listener (dinleyici) oluşturulacaktır. Bu listener, MQTT broker üzerinden gelen verileri abone olduğu konulardan (topics) anlık olarak alacaktır. Araçlardan gelen bu veriler, belirli aralıklarla güncellenerek sistem içerisinde aktarılacaktır. Listener, gelen verileri JSON formatında parse edecek ve bu verileri uygulama haritasında gösterimi yapılacaktır.

Adım 2.2: Sumo

Rota Optimizasyon Modülünden gelen planlanan senaryo SUMO'ya iletecektir. SUMO simülasyonu gerçekleştirilecek ve araçların simülasyonu haritada gösterilecektir.

Adım 3: Hata , Uyarı ve Bilgilendirme Mesajları Entegrasyonu

Araçlardan gelen verilere göre kritik durumlara sebep olabilecek veriler için hata, uyarı ve bilgilendirme mesajları gösterilecektir. Hata, uyarı ve bilgilendirme mesajlarındaki neden-sonuç ilişkisi kök neden analizi yaklaşımı ile gösterilecektir. Nedenler ise sürücü, araç, rota, performans, teslimat ve sistem kaynaklı olarak sınıflandırılacaktır.

Adım 4: Performans İzleme Paneli

Gerçek zamanlı veriler anlık olarak ve yüksek frekansta aktığı için, veriler belirli segmentlere ayrılır. Segmentasyon işlemi, aracın belirli bir konumdan başka bir konuma gerçekleştirdiği yolculukları tanımlamak amacıyla yapılır. Segmentlere ayrılan veriler üzerinde daha sonra analiz ve doğrulama (validation) işlemleri gerçekleştirilir. Bu analiz sürecinde, değişkenler arasındaki ilişkiler incelenir ve hem değişkenler arası korelasyonlar hem de hedef değişken olan enerji tüketimi ile olan etkileşimleri analiz edilir. Bu amaçla korelasyon haritaları oluşturularak hangi faktörlerin enerji tüketimini ne ölçüde etkilediği görselleştirilir. Ardından makine öğrenimi modeli ile aracın bulunduğu ana kadar tükettiği enerji miktarına göre kalan menzil ve son 5 saniyede araca etki eden faktörlerin etki yüzdeleri bulunur.

Adım 4.1: Veri Toplama ve İşleme

Bu adımda, makine öğrenimi modellerinde kullanılacak veriler gerçek zamanlı araç verilerinden toplanır. İlk olarak, bu veriler üzerinde veri temizleme (eksik, hatalı veya uç değerlerin ayıklanması) ve veri dönüştürme (gerekli formatlara çevirme, birim dönüşümleri vb.) işlemleri uygulanır.

Adım 4.2: Segmentasyon Belirlenmesi

Gerçek zamanlı veriler anlık olarak ve yüksek frekansta aktığı için, veriler belirli segmentlere ayrılır. Segmentasyon işlemi, aracın belirli bir konumdan başka bir konuma gerçekleştirdiği yolculukları tanımlamak amacıyla yapılır.

Adım 4.3: Enerji Tüketimi Faktör Analizi

Segmentlere ayrılan veriler üzerinde daha sonra analiz ve doğrulama (validation) işlemleri gerçekleştirilir. Bu analiz sürecinde, değişkenler arasındaki ilişkiler incelenir ve hem değişkenler arası hem de hedef değişken olan enerji tüketimi ile olan etkileşimleri analiz edilir. Bu amaçla faktör analizleri yapılarak hangi faktörlerin enerji tüketimini ne ölçüde etkilediği görselleştirilir.

Adım 4.4: Modelleme

Makine öğrenimi modeli ile aracın bulunduğu ana kadar tükettiği enerji miktarına göre kalan menzil ve son 5 saniyede araca etki eden faktörlerin etki yüzdeleri bulunur.

Adım 4.5: Entegrasyon ve Görselleştirme

Bu modelin çıktıları, araç takip modülündeki performans panelinde anlık olarak gösterilecektir. Dinamik bir şekilde beş saniyede bir son 100 metredeki enerji tüketim dağılımını pie chart şeklinde göstermesi gerekmektedir. Rotalar bazında araçların anlık olarak bulundukları rota üzerinde, son 100 metredeki enerji tüketimini etkileyen faktörler ve tahmini kalan menzil makine öğrenmesi yöntemleri ile analiz edilerek performans izleme panelinde sunulacaktır. Sipariş bazında ise o ona kadar yolda olan, teslim edilen ve iptal edilen siparişlerin grafiği görselleştirilecektir.

Performans İzleme Modülü:

Adım 1: Veri Depolama

MariaDB üzerinde sürücü, araç, rota, sipariş, şarj istasyonları gibi tüm yapısal veriler detaylı şekilde depolanacaktır. Günlük, haftalık, aylık ve yıllık bazda analiz yapılabilmesi için bu veriler zaman dilimlerine göre sorulabilecek şekilde saklanacaktır.

Adım 2: Grafana ile Görselleştirme

Grafana entegrasyonu ile, MariaDB veritabanına bağlanarak çeşitli sorgular aracılığıyla grafik ve tabloların yer aldığı dashboardlar oluşturulmuştur. Bu dashboardlar; menzil, enerji tüketimi, teslimat süresi ve durumu, sürücü performansı, şarj istasyonu kullanımı, araç bakım durumu, sipariş verimliliği, rota uyumluluğu ve karbon emisyonu gibi kritik performans göstergelerini içeren grafik ve tablolardan oluşmaktadır. Kullanıcılar bu grafikler aracılığıyla filo performansını detaylı şekilde takip edebilecek, farklı zaman aralıkları için analizler yapabilecek ve verileri görsel olarak kolayca değerlendirebileceklerdir.

Adım 3: Entegrasyon

Grafana dashboardları, iframe yöntemiyle React tabanlı kullanıcı arayüzüne entegre edilmiştir. Bu sayede kullanıcılar, herhangi bir ek işlem yapmadan sistem arayüzü üzerinden grafik ve tablo panellerine doğrudan erişebilmektedirler. Arayüzde tarih ve zaman aralıkları seçimi yapılabilecek alanlar bulunmaktadır; böylece

kullanıcılar günlük, haftalık, aylık veya özel tarih-saat aralıklarında detaylı filtreleme yaparak verileri inceleyebilmektedir.

Kullanılan Teknolojiler

- Python: Veri işleme, makine öğrenmesi ve sistem arayüzünün geliştirilmesinde kullanılacaktır.
- Pandas & NumPy: Veri analizi ve matematiksel işlemler için temel kütüphanelerdir.
- Grafana: Performans verilerinin zaman serisi bazlı görselleştirilmesinde kullanılacaktır.
- Sumo: Araç Simülasyonunda kullanılacaktır.
- React: Web tabanlı kullanıcı arayüzünün geliştirilmesinde kullanılacaktır.
- Node.js: Arka uç sunucu işlemleri ve API yönetimi için kullanılacaktır.
- MongoDB: Ortak çalışma veritabanı için bazı rota ve araç verileri tutulmaktadır.
- MariaDB: Yapısal filo verilerinin saklanması ve Grafana ile entegrasyonunda kullanılacaktır.
- FIWARE(MQTT): Araçlardan gelen sensör verilerinin iletimi için kullanılacaktır. Bu veri iletiminde MQTT (Message Queuing Telemetry Transport) protokolü kullanılacaktır.

A.5. Araştırma Olanakları

Tablo 1: Projede Kullanılacak Mevcut Makine - Teçhizat Listesi

| Projede Kullanılacak Mevcut Makine – Teçhizat Listesi (*) | |
|---|--|
| Adı/Modeli | Projede Kullanım Amacı |
| MONSTER ABRA A5 V20.3 | Kod yazma, Web tasarımı, Model eğitimi |
| Lenovo v15 IIL 82C5 i7 | Kod yazma, Web tasarımı, Model eğitimi |

A.6. Çalışma Takvimi

A.6.1. İş zaman Çizelgesi:

Tablo 2: İş Zaman Çizelgesi

| İş Paketi Ad/Tanım | Haftalar | | | | | | | | | | | | | | |
|--|----------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| PLANLAMA | | | | | | | | | | | | | | | |
| Projenin planlanması. | | | | | | | | | | | | | | | |
| Literatür taraması. | | | | | | | | | | | | | | | |
| ANALİZ | | | | | | | | | | | | | | | |
| Sistem gereksinimlerini belirlenmesi. | | | | | | | | | | | | | | | |
| Kullanılacak teknolojilerin araştırılması | | | | | | | | | | | | | | | |
| Teknolojilerin teorik bilgi edinme süreçlerinin tamamlanması | | | | | | | | | | | | | | | |
| TASARIM | | | | | | | | | | | | | | | |
| Uygulama prototipi yazılması | | | | | | | | | | | | | | | |
| Sistem tasarımının tamamlanması | | | | | | | | | | | | | | | |
| Arayüz geliştirme ve implementasyon süreçleri | | | | | | | | | | | | | | | |
| Gerekli modüllerin kodlama süreçleri | | | | | | | | | | | | | | | |
| Test tasarımlarının tamamlanarak uygulamaya entegre edilmesi | | | | | | | | | | | | | | | |
| UYGULAMA | | | | | | | | | | | | | | | |
| Gerekli modüllerin kodlama ve birim test süreçleri | | | | | | | | | | | | | | | |
| Tüm modüllerin birleştirilerek monolitik hale getirilmesi | | | | | | | | | | | | | | | |
| Dış servis entegrasyonlarının tamamlanması | | | | | | | | | | | | | | | |
| Entegrasyon ve kabul testlerinin tamamlanması | | | | | | | | | | | | | | | |
| RAPORLAMA/TOPLANTI SÜREÇLERİ | | | | | | | | | | | | | | | |

A.6.2. Kişi- İş Açıklaması

Tablo 3: Kişi – İş Açıklaması

| İş Paketi | Çalışacak Kişi | İş Paketi Açıklaması |
|---------------------------------------|-----------------------------|--|
| Problem tanımlaması | Berna ÇAKIR Şeyma COŞTUR | Problemin belirlenmesi ve tanımlanması. |
| Literatür araştırması | Berna ÇAKIR Şeyma COŞTUR | Literatür araştırılarak probleme ilgili yapılan çalışmaların incelenmesi ve analiz yapılması |
| Sistem gereksinimlerinin belirlenmesi | Berna ÇAKIR Şeyma COŞTUR | Programın hangi dilde ve hangi geliştirme ortamında yazılabileceğine karar verilmesi. |

| | | |
|--|-----------------------------|--|
| Kullanılacak ortamların araştırılması | Berna ÇAKIR Şeyma COŞTUR | Programın hangi dilde ve hangi ortamda yazılacağına karar verilmesi |
| Prototip çalışmasının yapılması | Berna ÇAKIR | Programın nasıl çalışacağı ile ilgili bir ön çalışma yapılması |
| Anket ve analiz çalışması | Şeyma COŞTUR | Proje konusunda anket çalışması yapılması |
| Ara rapor ve sunumun hazırlanması | Berna ÇAKIR Şeyma COŞTUR | Projeye ait rapor ve sunum hazırlama |
| Arayüz tasarıımı | Berna ÇAKIR | Tasarlanan uygulama doğrultusunda arayüz tasarımı yapılması |
| Sistem tasarıımı | Şeyma COŞTUR | Sistemde kullanılacak teknolojilerin kendi arasındaki bağlantılarının tasarlanması |
| Araç takip modülünün geliştirme ve test süreçleri | Berna ÇAKIR | Araç takip modülünün geliştirilmesi ve test edilmesi |
| Performans izleme modülünün geliştirme ve test süreçleri | Şeyma COŞTUR | Performans izleme modülünün geliştirilmesi ve test edilmesi |
| Uygulama modül birleştirme ve entegrasyon süreçleri | Berna ÇAKIR Şeyma COŞTUR | Modüllerin entegrasyonu yapılarak uygulamanın test edilmesi. |

| | | |
|-------------------------------------|-----------------------------|--|
| Final rapor ve sunumun hazırlanması | Berna ÇAKIR Şeyma COŞTUR | Proje bitiminde final raporu ve sunum hazırlanması |
|-------------------------------------|-----------------------------|--|

B.ANALİZ

B.1. Sistem Gereksinimlerini Ortaya Çıkarma Yöntem ve Teknikleri:

B.1.1. Yazılı Basılı Belge İnceleme

Danışman hocamız ile problem ve problemin nasıl çözümlenebileceğine karar verdikten sonra elektrikli araçlarda yönetim uygulamaları, araç takibi, performans izleme, enerji tüketimi ve menzil tahmini üzerine literatür taraması yapılmıştır. Literatür taramaları sonucunda bu alanlardaki çalışmalar incelenmiş, kullanılan yöntemler ve elde edilen zorluklar analiz edilmiştir. Bu inceleme sürecinde projede kullanılacak tekniklerin, teknolojilerin ve gereksinimlerin belirlenmesinde büyük katkı sağlamıştır.

B.1.2. Yüz Yüze Görüşme

Projenin ilk aşamasından son aşamasına kadar izlenecek yolun belirlenmesi proje danışmanı Dr. Öğr. Üyesi Sinem BOZKURT KESER ve Prof. Dr. AHMET YAZICI ile haftada iki gün olacak şekilde düzenli görüşmeler yapılmıştır. Projedeki ilerlemeler görüşmeler doğrultusunda gerçekleştirilmiştir.

B.1.3. Anket

Anket yapılmasının temel sebebi, elektrikli araç yönetim sistemlerindeki araç takip ve performans izleme süreçleri odağında kullanıcıların bekłentilerini ortaya koyarak gereksinimleri ve sistem tasarımını geliştirebilmektir. Bu noktada 10 adet soru hazırlanmış ve farklı yaş grubu ve meslekten insanlara yöneltilmiştir.

1. Daha önce aşağıdaki sistemlerden herhangi birini kullandınız mı?

103 yanıt

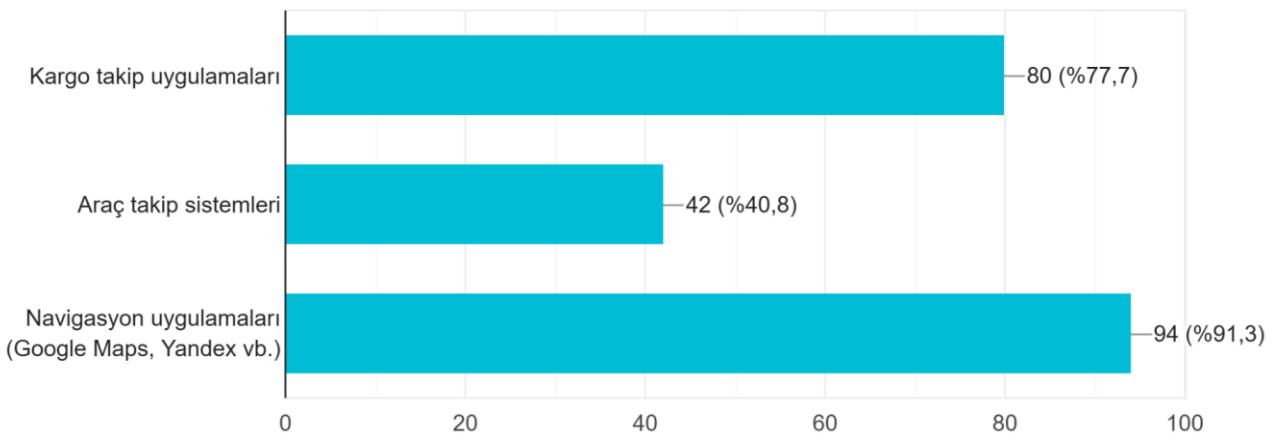


Figure 1: Anket Sorusu 1 ve Sonuçları

Katılımcıların %40'luk bir kısmı araç takip sistemlerini daha önce kullandığı belirtmiştir.

2. Daha önce teslimat ve kargo sürecinde görev aldınız mı?

103 yanıt

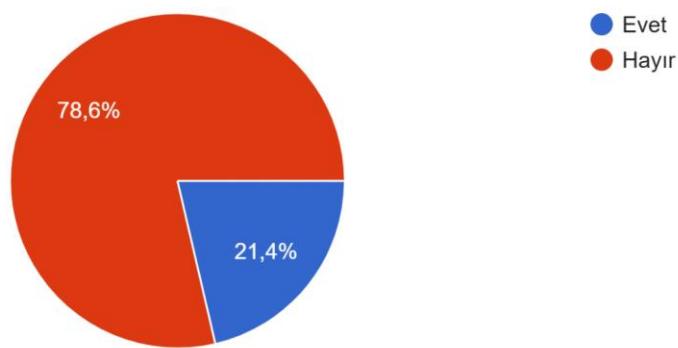


Figure 2: Anket Sorusu 2 ve Sonuçları

Katılımcıların %21'inin teslimat ve kargo sürecine dahil olduğu görülmüştür.

3. Elektrikli araçların kargo, su dağıtımı, lojistik gibi alanlarda yaygınlaşması hakkında ne düşünüyorsunuz?

103 yanıt

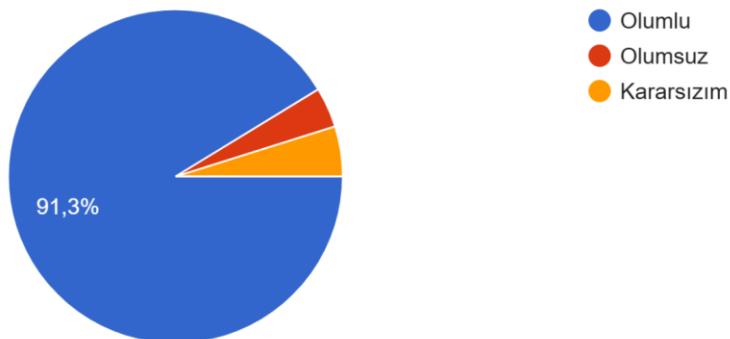


Figure 3: Anket Sorusu 3 ve Sonuçları

Katılımcıların çok büyük bir kısmı EV lojistik alanında yaygınlAŞmasına olumlu bakmaktadır.

4. Elektrikli aracın nerede olduğunu anlık olarak harita üzerinden takip edebilmek sizin için ne kadar önemli?

103 yanıt

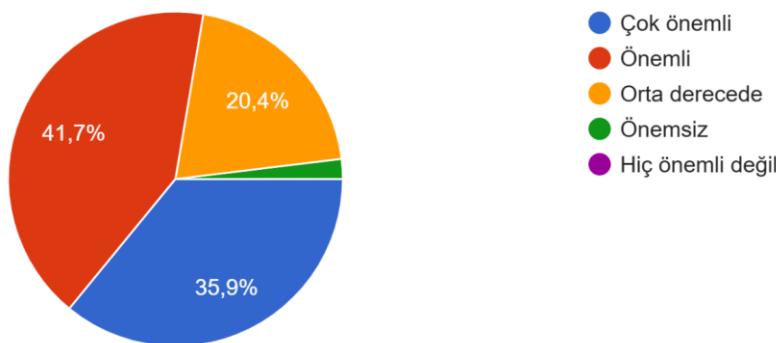


Figure 4: Anket Sorusu 4 ve Sonuçları

Katılımcıların %77,6'lık kısmı elektrikli aracın harita üzerinden anlık takibini önemli veya çok önemli bulmaktadır.

5. Sizce bir filo yönetimi uygulamasında en önemli performans ölçütü nedir?

103 yanıt

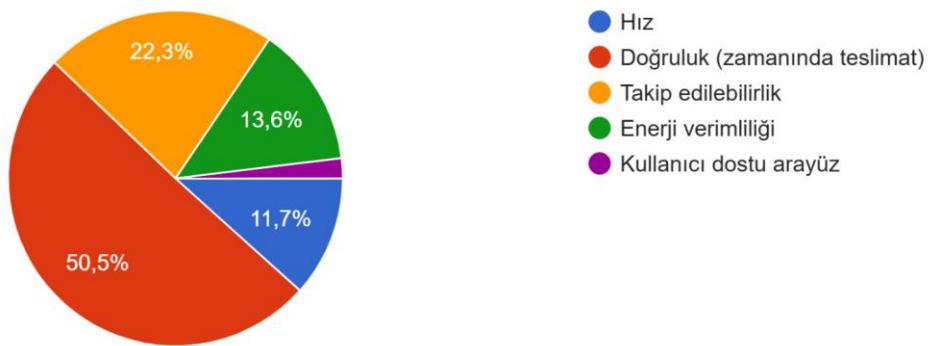


Figure 5: Anket Sorusu 5 ve Sonuçları

Katılımcıların yarısı doğruluk parametresini en önemli performans ölçütü seçmişlerdir.

6. Elektrikli Araç Teslimat süreci boyunca sürücülerin aşağıdaki bilgilerin hangilerini görmesi gerektiğini düşünüyorsunuz? (Birden fazla seçebilirsiniz)

103 yanıt

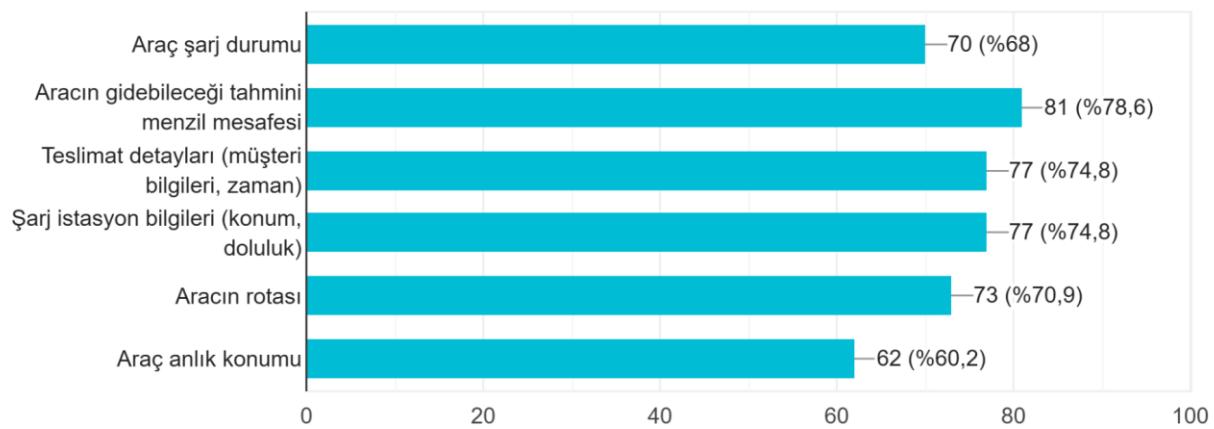


Figure 6: Anket Sorusu 6 ve Sonuçları

Kullanıcıların en çok ihtiyaç duyduğu bilgiler; menzil mesafesi, teslimat detayları ve şarj istasyonu bilgileridir.

7. Elektrikli Araç Teslimat süreci boyunca müşterilerin aşağıdaki bilgilerin hangilerini görmesi gerektiğini düşünüyorsunuz? (Birden fazla seçebilirsiniz)

103 yanıt

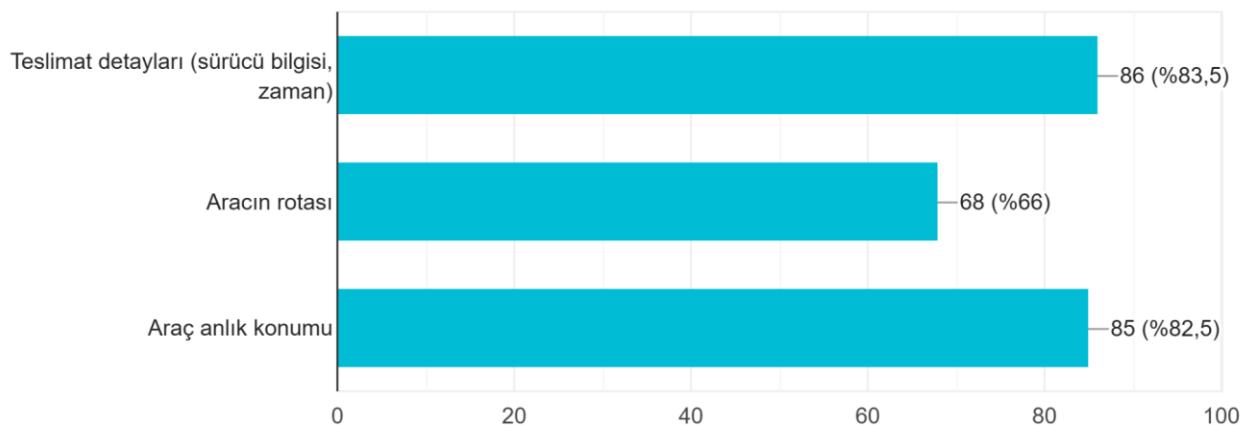


Figure 7: Anket Sorusu 7 ve Sonuçları

Kullanıcıların en çok teslimat detayları (%83,5) ve araç anlık konumu (%82,5) bilgilerini görmeyi gereklili bulmaktadır.

8. Elektrikli Araç Teslimat süreci boyunca sürücülerin şarja ve menzile etki eden faktörleri görmeleri gerektiğini düşünüyor musunuz?

103 yanıt

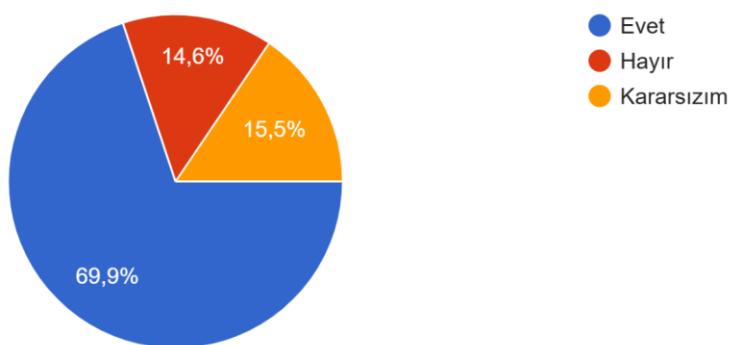


Figure 8: Anket Sorusu 8 ve Sonuçları

Katılımcıların büyük çoğunluğu sürücülerin şarja ve menzile etki eden faktörleri görmesi gerektiğini düşünüyor.

9. Elektrikli Araç Filo Yönetimi Uygulamasında raporlamanın önemli ve gerekli olduğunu düşünüyor musunuz?

103 yanıt

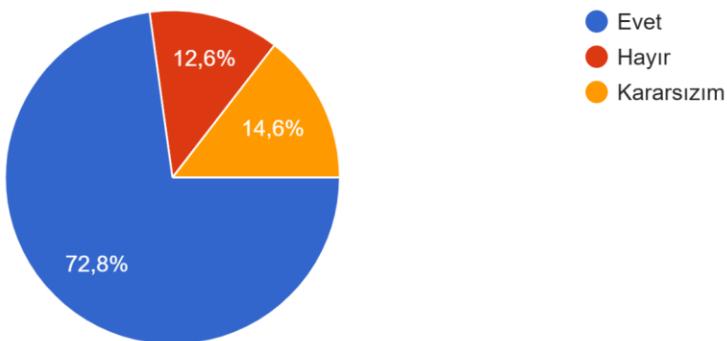


Figure 9: Anket Sorusu 9 ve Sonuçları

Katılımcılar büyük oranda raporlamanın elektrikli araç filo yönetiminde önemli ve gerekli olduğunu düşünmektedir.

10. Elektrikli Araç Filo Yönetimi Uygulamasında hangi raporların önemli olduğunu düşünüyorsunuz?
(Birden fazla seçebilirsiniz)

103 yanıt

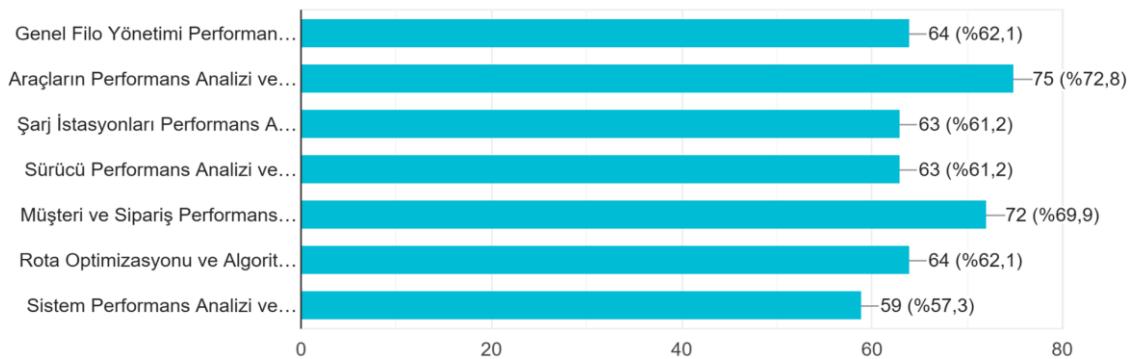


Figure 10: Anket Sorusu 10 ve Sonuçları

Kullanıcılar araç performans analizi ve raporlaması, müşteri ve sipariş performans analizi ve raporlamasının en önemli raporlar olduğunu düşünmektedir.

B.1.4. Gözlem

Yapılan anketler göz önüne alındığında katılımcılar elektrikli araçların dağıtım, kargo ve lojistik alanlarında kullanılmasına olumlu yaklaşmaktadır. Kullanıcılar aracın harita üzerinde takip edilmesini önemli bulunmakla birlikte, en önemli performans ölçütünün ise doğruluk olduğunu düşünmektedir ayrıca sürücülerin tahmini menzil, teslimat detayları, şarj istasyon bilgileri, rota bilgilerini , şarj ve menzile etki eden faktörleri yüksek oranda görmesini gerektiğini düşünmektedir. Raporlamanın ise önemli ve gereklili olduğu düşünmekle birlikte araç performans analizi ve raporlaması, müşteri ve sipariş performans analizi ve raporlamasının önemli olduğunu düşünmektedir.

B.1.5. Prototip ve Hızlı Uygulama Tasarımı (Rapid Application Design – RAD)

Prototip ve Hızlı Uygulama Tasarımı (RAD), projenin detaylı anlaşılabilmesi için yapılması planlanmaktadır. Araç izleme sayfasının prototipi Figma'da yapılmış ve danışmanlara sunulmuştur. İlgili geri dönüşlere göre asıl proje için düzeltmeler planlanmaktadır.

Araç Takip Prototip: <https://www.figma.com/proto/lPyNf0ATs2pNt1cmPNj3KE/Vehicle-Tracking?node-id=120-4233&t=inAi746eWFuyHzNI-1>

B.1.6. Ortak Uygulama Tasarımı (Joint Application Design – JAD)

Oluşturulan prototipten sonra proje danışmanı, yürütücüüsü ve çalışanlar sistemde ilgili tasarımları ortak olarak ele alacak ve proje tasarlanacaktır.

B.1.7. Veri Akış Şemaları

B.1.7.1 Kavramsal Veri Akış Şeması

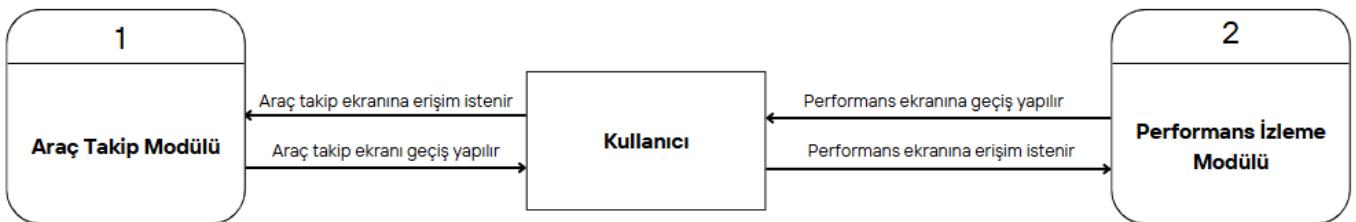


Figure 11: Kavramsal Veri Akış Şeması

B.1.7.2. Mantıksal Veri Akış Şeması

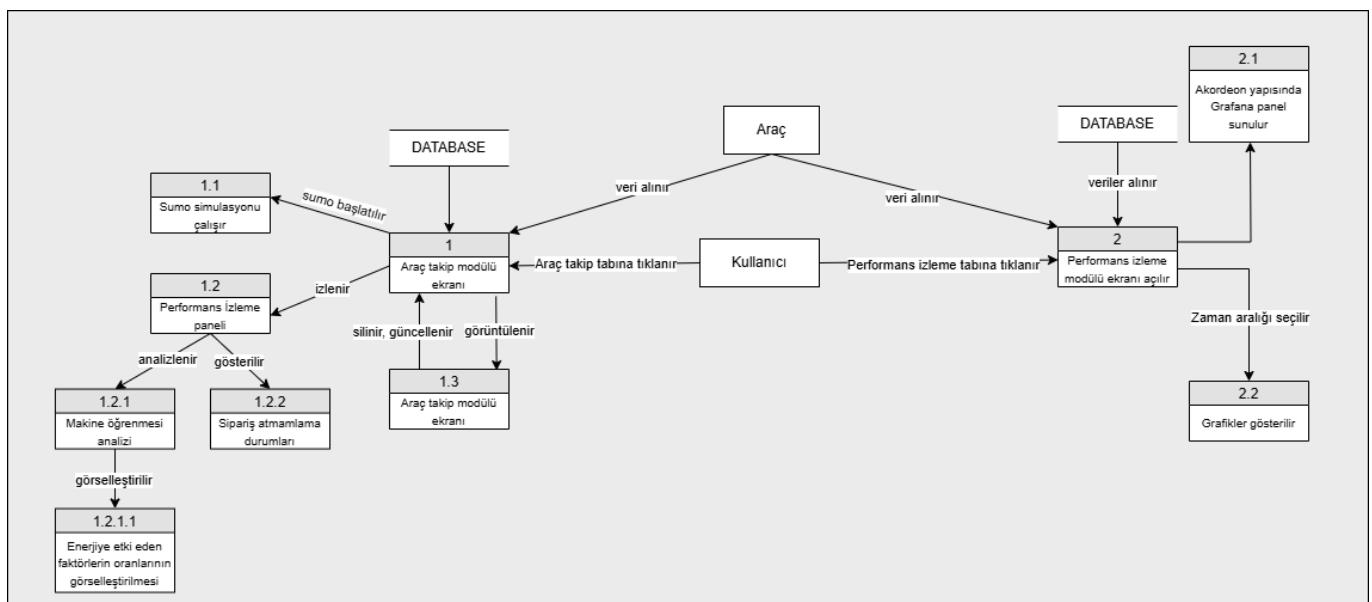


Figure 12: Mantıksal Veri Akış Şeması

B.1.7.3 Fiziksel Veri Akış Şeması

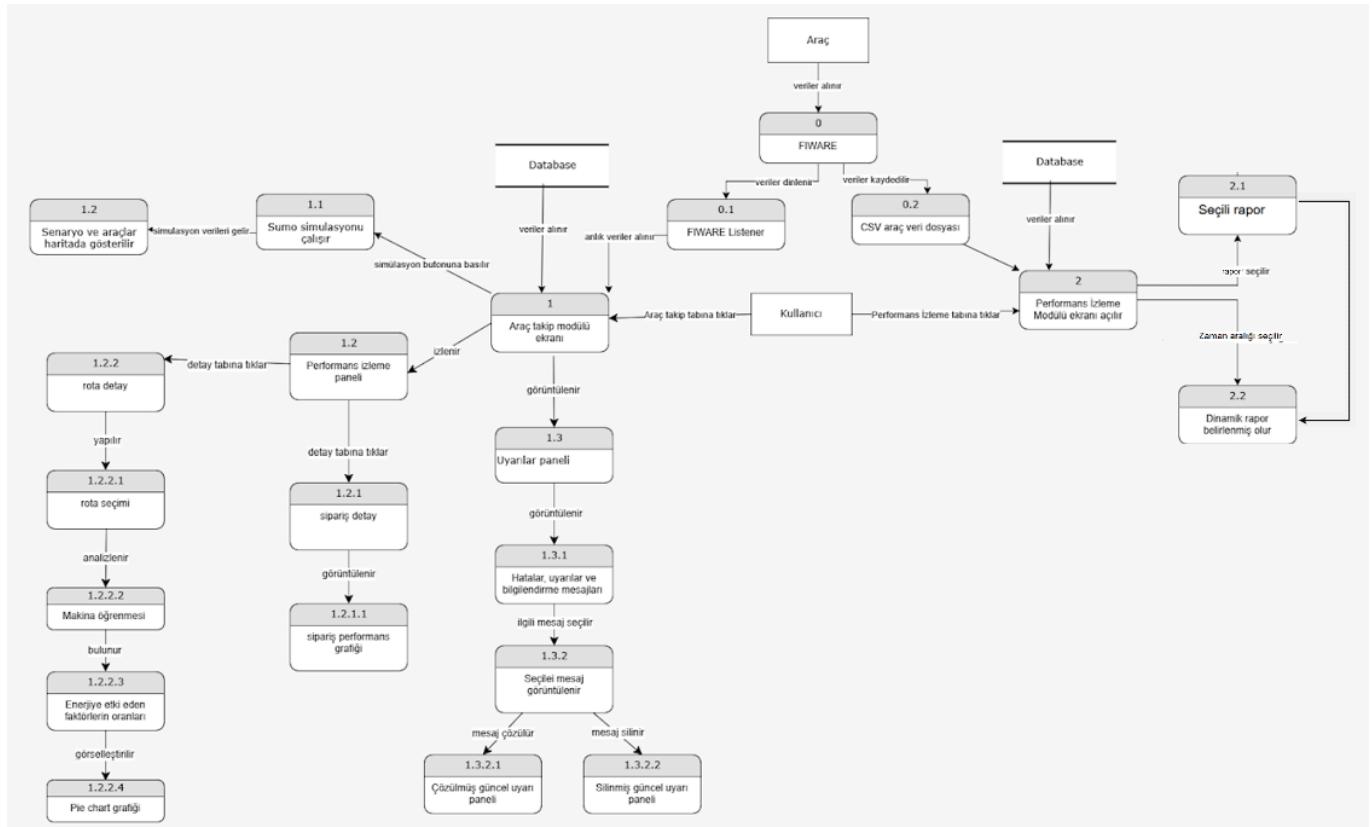


Figure 13: Fiziksel Veri Akış Şeması

B.1.8. Olay Tabloları, Durum Formları, İşlevsel Analiz Raporu, İş Akış Şeması

B.1.8.1. Olay Tabloları

Tablo 4: Olay Tabloları

| Olay | İstemci | Tetikleyici | İşlem | Yanıt | Hedef |
|---|-----------|---------------|--|----------|-----------------|
| Araç Bilgilerinin Alınması | Kullanıcı | Takip Talebi | Anlık araç verilerinin alınması | Uygulama | İzleme Modülü |
| Konum Güncellemesinin Yapılması | Sistem | Konum Verisi | Konum bilgisinin harita üzerinden güncellenmesi | Uygulama | Harita |
| Uyarı, Hata , Bilgilendirme Mesajlarının Gönderilmesi | Sistem | Kritik Durum | Uyarı, hata , bilgilendirme mesajlarının oluşturulması | Uygulama | Bildirim Paneli |
| Model Verisinin Gönderilmesi | Sistem | Eğitim Verisi | Enerji tüketim verisinin modele iletilmesi | Uygulama | Makine Modeli |

| | | | | | |
|---|-----------|--------------------|--|----------|-------------------|
| Performans Verisinin Kaydedilmesi | Sistem | Zamanlayıcı | Araçlardan gelen verilerin toplanması | Uygulama | Veritabanı |
| Performans Verisinin Görselleştirilmesi | Kullanıcı | Görüntüleme Talebi | Verilerin grafiksel olarak sunulması | Uygulama | Performans Modülü |
| Görselleştirilmiş Verilerle Etkileşim | Kullanıcı | Tarih Filtreleme | Tarih filtrelemesi ile güncellenmiş grafiklerin görüntülenmesi | Uygulama | Raporlama Modülü |

B.1.8.2 Durum Formları

Tablo 5: Durum Formları

| Durum İsmi: Araç konum bilgisinin güncellenmesi | İşlem No: 1 | | |
|--|--------------------|-------------------------|--------------------|
| Tanım: Araçtan alınan konum bilgisi sisteme işlenir ve izleme ekranında güncel konumu gösterilir. | | | |
| Tetikleyici: Konum değişikliği | | | |
| Tetikleyici Türü: GPS verisi | | | |
| Girdi İsmi | Kaynak | Çıktı | Hedef |
| Konum bilgisi | Araç'ın GPS | Güncellenen araç konumu | Araç İzleme Modülü |

B.1.8.3. İşlevsel Analiz Raporu

Tablo 6: İşlevsel Analiz Raporu

Proje Kodu

Proje Adı: Elektrikli Araçlarda Filo Yönetim Sistemleri için Araç Takip ve Performans İzleme Modüllerinin Geliştirilmesi

Hazırlayanlar: Şeyma COŞTUR, Berna ÇAKIR

Başlangıç Tarihi: 16.09.2024

Son Değiştirme Tarihi: 26.03.2025

Versiyon: 0.1

İlgili Kişi: Dr. Öğr. Üyesi SİNEM BOZKURT KESER

Görevi: Proje Danışmanı

Mevcut Sistemlere Etkisi: Elektrikli araçların anlık takibi ve performans verilerinin analizi ile filo yönetim sistemlerinde verimlilik ve sürdürülebilirlik sağlayacaktır.

Kırtasiye ve Sarf Malzemesi İhtiyacı: Herhangi bir kırtasiye ve sarf malzemesi ihtiyacı bulunmamaktadır.

Donanım: Herhangi bir donanım ihtiyacı bulunmamaktadır.

Maliyetler: Lisanslama açısından bir maliyet bulunmamaktadır.

B.2. Sistem Gereksinimleri

B.2.1. İşlevsel Gereksinimler

Araç Takip Modülü İşlevsel Gereksinimler:

- Araçlara ait bilgiler opsyonel olarak haritada araca tıklandığında görülebilmelidir
- Araçların anlık enerji tüketim değerleri gösterilmelidir

- Elektrikli araçların batarya seviyeleri sürekli izlenmelidir.
- Araçların dinamik olarak hareket etmelidir.
- Seçilen aracın ve sürücünün teslimat listesi ve anlık konum bilgisinin gösterilmesi veritabanından çekilerek gösterilecektir.
- Seçili rotada ilgili siparişlerin güncel durumuna erişilebilecektir.
- Mevcut şarj seviyesi rota tamamlanabilirliğine göre değerlendirilmelidir.
- Şarj istasyonlarının konumlarının ve doluluk bilgilerinin harita üzerinde gösterimi yapılmalıdır.
- Şarj durumundaki kritik seviyelerde müdahale gerektiren bildirimler oluşturulmalıdır.
- Anlık enerji tüketim değerleri görüntülenebilmelidir.
- Rota seçimine göre enerji tüketimlerinin faktör analizi grafik ile makina öğrenmesi yapılarak görselleştirilecektir.
- Siparişlerin durumları tamamlanan, iptal edilen grafik ile görselleştirilmelidir.
- Tahmini enerji tüketimi, tahmini kalan menzil ve tahmini şarj miktari hesaplanmalı ve gösterilmelidir.
- Sistem hata ve uyarıları üç seviyede sınıflandırılmalıdır.: Uyarı, Hata, Bilgi.
- Hata, uyarı ve bilgilendirmeler renk kodları ve ikonlar ile gösterilmelidir.
- Kök neden analizi ile hata ve uyarılar ilişkilendirilmeli ve neden-sonuç bağlantıları görselleştirilebilmelidir.
- Sürücü , rota, araç, performans, teslimat ve sistem kaynaklı hatalar belirlenmeli, izlenmelidir.
- Uyarılar kullanıcı tarafından çözüldü olarak işaretlenebilmelidir.
- Araçların ve siparişlerin durumu listelenebilmelidir.
- Haritada siparişe tıklandığında sipariş detayları görüntülenmelidir
- Araca rotasına tıklandığında teslimat listesi görüntülenebilmelidir.

Performans İzleme Modülü İşlevsel Gereksinimler:

- Modülde bütün rapor ve grafikler için gerekli veriler belirli aralıklarla (günlük, haftalık, aylık, yıllık) görüntülenebilmelidir.
- Kullanıcılar istedikleri özel tarih ve saat aralıklarını seçerek rapor ve grafikleri bufiltrelere göre görüntüleyebilmektedir.
- Tüm filo yönetim sistemindeki toplam veriler ile genel filo yönetim raporu oluşturulmalıdır
- Fosil yakıtlı dağıtıma göre EV ile oluşan karbon kazanım analizi raporu alınabilmelidir. ? (Karbon Kazanım Analizi Raporu)
- Araç Performans Raporu, Araç analizlerini grafik ile gösterilebilmeli ve raporlanmalıdır. (Araç Performans Raporu)
- Araçlar genel performans düzeyine göre "İyi", "Orta" ve "Kötü" olarak sıralanmalıdır.
- Hız, enerji tüketimi ve menzil gibi ölçütlerde göre sıralama yapılabilmelidir.
- Araçlara atanan(planlanan) ve araçların gerçekleştirdiği rotalar incelenerek her rota için doğruluk oranları hesaplanmalı ve grafiklerle gösterilmelidir.
- Kullanılan araçlar için performans karşılaştırması yapılabilmelidir. (Araç Karşılaştırma Raporu)
- Karşılaştırma kriterleri: enerji tüketimi, menzil, ortalama hız, toplam taşınan yükür.

- Araçların son bakım tarihleri ve yaklaşan bakım zamanları incelenmeli ve araç bakım raporu oluşturulmalıdır. (Araç Bakım Raporu)
- Şarj istasyonlarının günlük, haftalık ve aylık kullanım sıklıkları analiz edilmelidir.
- Şarj işlemleri için harcanan ortalama ve toplam süreler raporlanmalıdır. (Şarj İstasyonlarının Kullanım Raporu)
- Şarj istasyonlarının hangi zaman aralıklarında en yoğun şekilde kullanıldığı belirlenmelidir.
- Gün içerisindeki yoğun saatler tespit edilerek şarj istasyonlarının kapasite planlaması yapılmalıdır.
- Arıza kayıtları tutulmalı ve ortalama arıza süresi ile birlikte bakım gereksinimleri raporlanmalıdır.
- Şarj süreleri analiz edilerek araç başına ortalama şarj süresi hesaplanmalı ve sistem performansı değerlendirilmelidir.
- Sürücülerin performans verileri raporlanmalı ve grafikler ile gösterilmelidir. (Sürücü Performans Raporu)
- Sürücüler için siparişlerinde gecikme oranları ve başarılı teslimat durumları analiz edilmeli ve sürücü başarı raporu oluşturulmalıdır.
- Sürücü Başarı Raporunda Gecikme oranları, toplam başarılı teslimat sayısı ve performans ortalamaları ölçülmelidir. Performans seviyeleri: "Çok İyi" (>95%), "İyi" (80-95%), "Orta" (60-80%), "Düşük" (<60%)
- Sürücüler arasında ölçülebilir verilere dayalı karşılaştırma yapılmalıdır.
- Müşteri bazlı sipariş analizleri yapılabilmeli, belirli müşterilerin sipariş sıklıkları ve toplam sipariş sayıları raporlanabilmelidir. (Müşteri Bazlı Sipariş Raporu)
- Siparişler raporlanmalı ve grafiklerle gösterilmelidir. (Genel Sipariş Raporu)
- Rota optimizasyon algoritmalarının çözüm süresi ve doğruluk oranı analiz edilerek raporlanmalıdır. (Algoritma Performans Raporu)
- Algoritma performans ölçütleri: Çözüm süreleri (ms cinsinden), doğruluk oranı (planlanan vs. gerçekleşen rota farkı % cinsinden) ve kullanılan algoritmalar: Simulated Annealing, OR-Tools vb.
- Sistem performansı, sunucu tepki süresi (ms), anlık CPU ve bellek kullanımı gibi kriterler üzerinden izlenmeli ve raporlanmalıdır. (Sistem Performans Raporu)
- Genel sistem logları raporlanmalı ve bu loglar grafiksel olarak analiz edilebilmelidir. (Sistem Uyarı(Log) Raporu)

B.2.2.Sistem ve Kullanıcı Arayüzleri ile ilgili Gereksinimler

Araç Takip Modülü Sistem ve Kullanıcı Arayüzleri ile İlgili Gereksinimler

- Sipariş durumu takibi için siparişlerin durumu, renk kodları ve ikonlarla net bir şekilde gösterilmelidir.
 - Requested (Talep Edildi) – turuncu renk ve ikon
 - On the way (Yolda) – mavi renk ve ikon
 - Delivered (Teslim Edildi) – yeşil renk ve ikon
 - Cancelled (İptal Edildi) – kırmızı renk ve ikon
- Kullanıcılar, sipariş durumlarını görsel olarak kolayca ayırt edebilmelidir.
- Sipariş ve teslimat konumları, Leaflet haritası üzerinde anlık olarak gösterilmelidir.
- Harita, araç konumlarını ve sipariş durumlarını gerçek zamanlı olarak yansıtmalıdır.
- Araç hareketi haritada görüntülenebilmelidir
- Kritik olaylar (düşük şarj, geciken teslimat, rota sapması) için anlık bildirimler ekranда olmalıdır.

- Bildirim renk kodları ile yapmalıdır
- Uyarılar yanıp sönen ikonlarla dikkat çeken bir renkte olmalıdır.
- Sipariş detaylarına tıklayarak teslimat zamanı, ürün tipi, alıcı bilgisi ve iptal durumu doğru şekilde görüntülenebilmelidir.
- Bir araca tıklandığında, aracın şoförüne ait teslimat listesi ve teslim edilen/sıradaki siparişlerin harita üzerindeki konumu doğru şekilde gösterilmelidir.
- Kullanıcı arayüzü, kolay anlaşılır ve sezgisel bir tasarıma sahip olmalıdır.
- Renk kodları, ikonlar ve görseller, bilgilerin hızlı ve doğru algılanmasını sağlayacak şekilde tasarlmalıdır.
- Harita üzerindeki simgeler (araçlar, sipariş noktaları vb.), durumlarına göre değişen ikonlarla gösterilmelidir.
- Zaman bazlı animasyonlar veya efektlerle, sipariş durumu ve araç hareketleri belirgin hale getirilmelidir.
- Masaüstü uyumlu bir arayüz tasarımı sağlanmalıdır.

Performans İzleme Modülü Sistem ve Kullanıcı Arayüzleri ile ilgili Gereksinimler

- Sistem ve Kullanıcı Arayüzleri ile ilgili Gereksinimler
- Filo operasyonları performans raporu kısmında:
 - Araçların harcanan şarj değerleri, toplam kat edilen mesafeleri sütun grafik ile görselleştirilecektir.
 - Araçların ortalama hızları ve eğimleri çizgi grafik olarak görselleştirilecektir.
- Genel sipariş raporu kısmında:
 - Siparişlerin dağılımları pasta grafik ile görselleştirilecektir.
 - Siparişlerin günlük sayıları ve ortalama servis süreleri sütun grafik ile görselleştirilecektir.
- Araç bakım, karşılaştırma, performans ve karbon analizi kısmında:
 - Araçların bakım zamanları, türleri, maliyetleri, gecikmeleri ve karbon analizleri tablo ile görselleştirilecektir.
 - Araçların rota bazında enerji kullanımları, hızları, eğimleri, yükleri kıyaslanacak ve performans kategorileri renklendirilerek görselleştirilecektir.
 - Araçların ortalama hız, enerji tüketimi, taşınan yük, batarya durumu sütun grafiği ile görselleştirilecektir.
- Rota karşılaştırması, optimizasyon ve algoritmalar kısmında:
 - Planlanan ve gerçekleşen rotaların ortalama süresi ile mesafesi karşılaştırılacak, tamamlanma sürelerinin yüzdeleri renklendirilerek görselleştirilecektir.
 - Kullanılan algoritmaların oranları pasta grafik ile görselleştirilecektir.
 - Planlana algoritmaların seçim nedenleri renklendirilerek tablo formatında analiz edilecektir.
 - Rotaların planlanan ve gerçekleşen mesafeleri ile algoritma doğruluk oranları tablo formatında karşılaştırılarak görselleştirilecektir.
- Şarj istasyonu kısmında:
 - Şarj istasyonu analizlerinde kullanım sıklığı pasta grafikle gösterilecektir.
 - Şarj istasyonlarının kullanım süreleri ve saatleri sütun grafik ile görselleştirilecektir.
- Sürücü performansı ve başarısı kısmında:
 - Sürücülerin toplam teslimat sayısı, toplam çalışma süresi ve ortalama hız yığın sütun grafiği ile görselleştirilecektir.

- Sürücülerin gecikme oranları, performans skorları ve teslimat sayıları sütun grafik ile görselleştirilecektir.
 - Sürücülerin sayısı, toplam teslimat sayıları, toplam çalışma saatleri ve toplam kat edilen mesafeleri stat şeklinde görselleştirilecektir.
- Müşteri odaklı sipariş raporu kısmında:
 - Müşterin sipariş sayıları, geri bildirim puanları ve sipariş bölgeleri sütun grafik olarak görselleştirilecektir.
 - Müşteri yorumları ve skorları tablo yapısında görselleştirilecektir.
- Sistem performansı izleme ve uyarıları kısmında:
 - Sunucu tepki süresi, CPU ve bellek kullanımı çizgi grafiklerle izlenebilecektir.
 - Sistem logları türlerine ve önem derecelerine göre renk kodlaması ile sunularak hata analizini kolaylaşdıracaktır.

B.2.3.Veriyle İlgili Gereksinimler:

Araç Takip Modülü Veriyle İlgili Gereksinimler

- Araçlara ait bilgiler (araç ID, batarya seviyesi, enerji tüketimi, mevcut konum vb.) sistem veritabanında tutulmalı ve harita arayüzü ile senkronize edilmelidir.
- Araçların anlık enerji tüketim verileri sensörlerden alınarak veritabanına aktarılması ve harita üzerinden gerçek zamanlı gösterim için kullanılmalıdır.
- Batarya seviyeleri düzenli aralıklarla güncellenenerek izlenebilir olmalı ve bu veriler tahmini menzil ve şarj ihtiyacı hesaplamalarında kullanılmalıdır.
- Teslimat listesi, siparişlerin durumu, alıcı bilgileri, tahmini teslimat zamanı gibi bilgiler veritabanında her araç ve sürücü ile ilişkilendirilmiş şekilde saklanmalıdır.
- Sipariş durumu (Talep Edildi, Yolda, Teslim Edildi, İptal Edildi) verileri sistem tarafından gerçek zamanlı olarak güncellenmeli ve tarih/saat damgası ile kaydedilmelidir.
- Şarj istasyonlarına ait bilgiler (konum, doluluk oranı, ortalama şarj süresi) sistemde ayrı bir tablo yapısıyla tutulmalıdır.
- Sistem içi bildirimlerin (düşük batarya, rota sapması, geciken teslimat vb.) tipleri, önem seviyesi (Uyarı, Hata, Bilgi), zaman damgası ve çözülme durumu ile birlikte loglanmalıdır.
- Tüm hata, uyarı ve bilgi mesajları sistem loglarına yazılmalı; mesaj tipi, nedeni, kok neden (araç, sürücü, sipariş vs.) ve zaman bilgisi kaydedilmelidir..
- Geçmiş rota, enerji tüketimi ve teslimat detayları arşivlenerek sistemde zaman bazlı analizler için erişilebilir olmalıdır.

Performans İzleme Modülü Veriyle İlgili Gereksinimler

- Filo genel verileri (toplam menzil, toplam görev, iptal oranı, enerji tüketimi, ortalama teslimat süresi, araç ve müşteri sayısı) toplanmalı ve analiz edilebilir formatta saklanmalıdır.

- Her araca ait performans verileri (ortalama hız, tüketilen enerji, menzil, taşınan yük, batarya durumu, görev tamamlama oranı vb.) detaylı biçimde tutulmalı ve tarih damgalı olarak zaman serisi şeklinde işlenmelidir.
- Sürücü performans verileri (toplam teslimat, çalışma süresi, gecikme oranı, başarılı teslimat sayısı) kaydedilmelidir.
- Müşterilerin teslimat istekleri, yorumları ve puanları tutulmalı geri bildirim olarak geliştirme yapabilmek için kullanılmalıdır.
- Araç bakım bilgileri (son bakım tarihi, bakım periyodu, yaklaşan bakım zamanı) zaman bazlı kontrol ve raporlama için sistemde ayrı tabloda tutulmalıdır.
- Şarj istasyonlarına ilişkin günlük/haftalık kullanım sıklığı, ortalama şarj süresi ve yoğun saatler ayrı ayrı işlenmeli ve analizler için etiketlenmelidir.
- Algoritma performansı için kullanılan yöntem, çözüm süresi, doğruluk oranı gibi metrikler simülasyon sonrası veritabanına kaydedilmeli ve karşılaştırma raporlarında kullanılmalıdır.
- Sistem performansına ilişkin metrikler (CPU kullanımı, bellek kullanımı, sunucu tepki süresi) düzenli aralıklarla izlenerek saklanmalıdır.
- Uygulamaların log verileri detaylı biçimde tutulmalı, sebepleri ilişkilendirilmelidir.
- Tüm geçmiş veriler arşivlenerek performans için kullanılabilir şekilde tutulmalıdır.

B.2.4.Kullanıcılar ve İnsan Faktörü Gereksinimleri, Güvenlik Gereksinimleri:

- Sistem, filo yöneticileri olan kullanıcı profiline göre tasarlanmalıdır. Arayüzler, bu kullanıcıların ihtiyaçlarına ve görevlerine uygun şekilde özelleştirilmelidir.
- Filo yöneticileri, uygulama üzerindeki tüm modüllere ve ekranlara sınırsız erişim yetkisine sahip olmalıdır.
- Güvenlik kontrolü için Kritik işlemler (örneğin: uyarı silme, sistem ayarlarını değiştirme) yalnızca filo yöneticileri tarafından gerçekleştirilmelidir. İlgili ekran kodları içerisinde kontrolü yapılmalıdır.

B.2.5.Teknik ve Kaynak Gereksinimleri, Fizikal Gereksinimler:

- **Python (3.11.9)** dili, sistem arayüzü geliştirme, veri işleme ve makine öğrenmesi süreçlerinde kullanılacaktır.
- **Pandas** ve **NumPy** kütüphaneleri, veri analizi ve matematiksel işlemler için temel bileşenler olarak görev yapacaktır.
- **Grafana**, zaman serisi performans verilerinin izlenmesi ve görselleştirilmesi amacıyla kullanılacaktır.
- **SUMO (Simulation of Urban MObility)**, araç simülasyonlarının gerçekleştirilmesinde kullanılacaktır.
- **FIWARE platformu**, araçlardan gelen sensör verilerinin toplanmasında kullanılacaktır.
- Veriler, **MQTT (Message Queuing Telemetry Transport)** protokolü ile iletilecektir.
- **React**, kullanıcı arayüzünün modern ve etkileşimli olarak tasarılanmasında kullanılacaktır.
- **Node.js (18.20.2)**, sistemin backend tarafında kullanılacaktır.
- **NPM (10.2.4)**, Node.js paket yönetimi ve bağımlılılıkların kurulumu için kullanılacaktır.
- **MongoDB**, yapılandırılmamış verilerin (loglar, sensör çıktıları) saklanması sırasında kullanılacaktır.
- **MariaDB (11.3.2)**, ilişkisel veritabanı yönetimi için tercih edilecektir.
- **HeidiSQL (12.3.0)**, MariaDB veritabanına grafik arayüz üzerinden erişim ve yönetim için kullanılacaktır.
- **Figma**, kullanıcı arayüz tasarımlarının prototiplenmesi ve paylaşımı için kullanılacaktır.

- **Visual Studio Code 2022**, yazılım geliştirme ortamı olarak kullanılacaktır.
- Sunucu tarafında performans izleme ve MQTT verisi alımı için sürekli çalışan, düşük gecikmeli bir sunucu altyapısı gereklidir.

C.TASARIM

C.1. Sistem Tasarımı

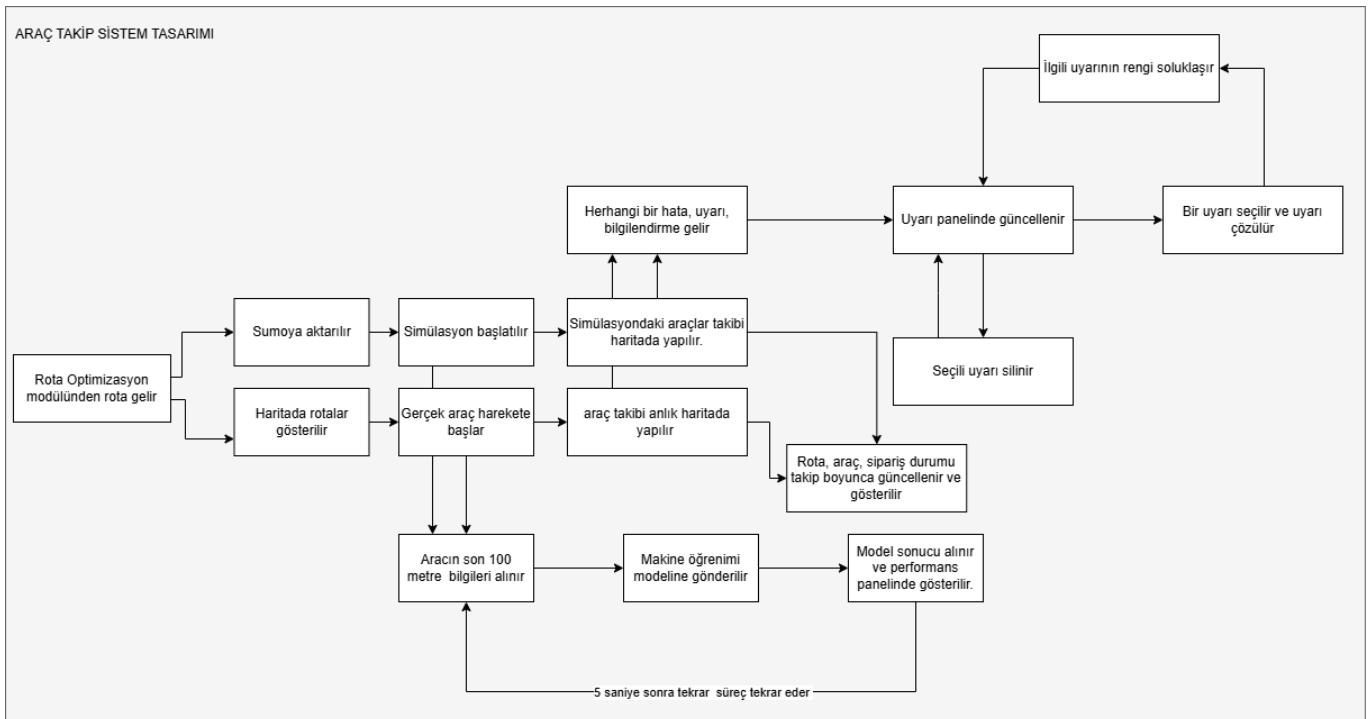


Figure 14: Araç Takip Modülü Sistem Tasarımı Şeması

Sistem tasarımı Araç Takip ve Performans Modülü için modüller bazda ayrı olarak ele alınır. Araç Takip Sistem Tasarımı, elektrikli araç filosunun hem SUMO simülasyon ortamında hem de gerçek aracın anlık olarak izlenmesini ele almaktadır. Sistem, rota optimizasyon modülünden gelen rotaların alınması ile başlar. Bu rotalar iki ayrı şekilde ilerler. İlk akışta SUMO'ya rotaların aktarılması sonucu simülasyon başlatılır ve sumodan gelen araç verilerinin takibi yapılır. Diğer ise gerçek araç için rotaları haritaya gönderir. Ardından aracın ilerlemesi ile haritadan takibi yapılır. Araçlardan belirli periyotlarla, özellikle aracın son 100 metrelük sürüşüne ait hız, enerji tüketimi gibi veriler alınır. Bu veriler, makine öğrenimi modeline gönderilerek aracın menzil tahmini yapılır ve enerji tüketimine etki eden faktörlerin yüzde analizi yapılır. Elde edilen sonuçlar, performans panelinde grafiksel olarak gösterilir. Bu analizler, hem operasyonel verimlilik hem de enerji performansı açısından önemlidir. Tüm bu süreç yaklaşık her 5 saniyede bir tekrar edilerek sistemde dinamik hale getirilir. Araç takip sisteme herhangi bir hata, uyarı veya bilgilendirme mesajı meydana gelirse bu mesajlar uyarı paneline ilettilir ve panel üzerinde güncellenir. Mesajlardan seçilen mesaj çözülürse çözüldüğünü kullanıcının anlayacağı şekilde belirtilir. Ayrıca uyarılar panelden silinebilir. Bu da sisteme basit bir hata çözüm akışı oluşturur.

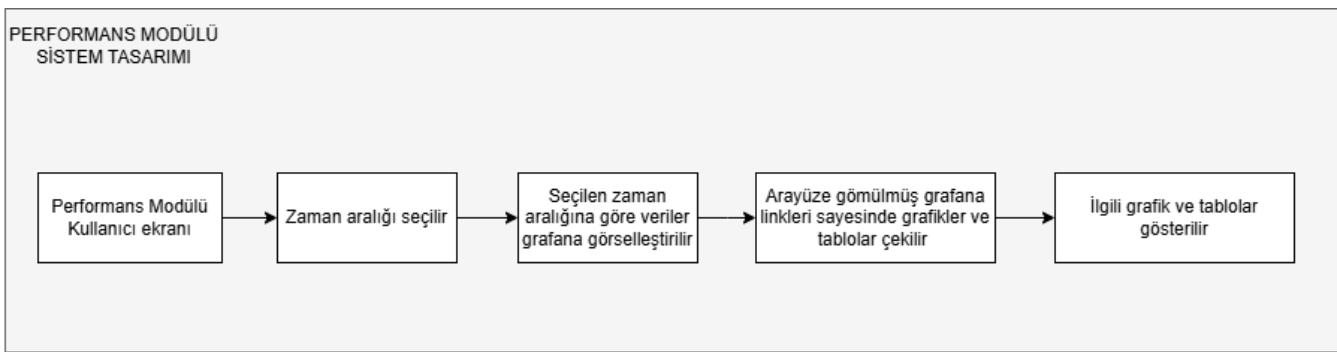


Figure 15: Performans İzleme Modülü Sistem Tasarımı Şeması

Performans izleme modülü, react arayüzünde akordeon yapısında düzenlenen on dört farklı rapordan oluşmaktadır. Her bir akordiyon başlığında, grafikler, tablolar ve istatistik panellerinden oluşan grid tabanlı paneller yer almaktadır. Bu paneller, Grafana üzerinde geliştirilmiştir ve grafiklerin, tabloların oluşturulması belirlenen SQL sorguları ile MariaDB veritabanı kullanılarak yapılandırılmaktadır. Veriler, belirlenen SQL sorguları ile alınmakta ve grafikler ile tablolar dinamik olarak oluşturulmaktadır. Panellerin açıklamaları, renk düzenlemeleri ve başlıklarları doğrudan Grafana arayüzü üzerinden yapılandırılmıştır. Modülde sunulan veriler; araç, sürücü, rota, müşteri, şarj istasyonları, sistem performansına ilişkin bilgileri kapsamaktadır. Veri değişimleri renk geçişleri ile kullanıcının dikkatini çekecek ve önemli noktaları belirtecek şekilde geliştirilmektedir. Görselleştirmeler sayesinde kullanıcılar, filoya ait genel durumu hızlı ve etkili bir şekilde analiz edebilmektedir. Modül içerisinde kullanıcının yoğun olarak kullanacağı hazır zaman aralıkları (bugün, bu hafta, bu ay, geçen ay ve bu yıl) bulunmakla birlikte modül içerisinde kullanıcıya özel tarih ve saat seçimi yapabileceği filtreleme alanları bulunmaktadır. Bu yapı sayesinde hem uzun hem de kısa dönemli performans izlemeleri gerçekleştirilebilecektir.

C.2.Kullanıcı ve Sistem Ara yüzü Tasarımları

Araç Takip Modülü Arayüz Tasarımı:

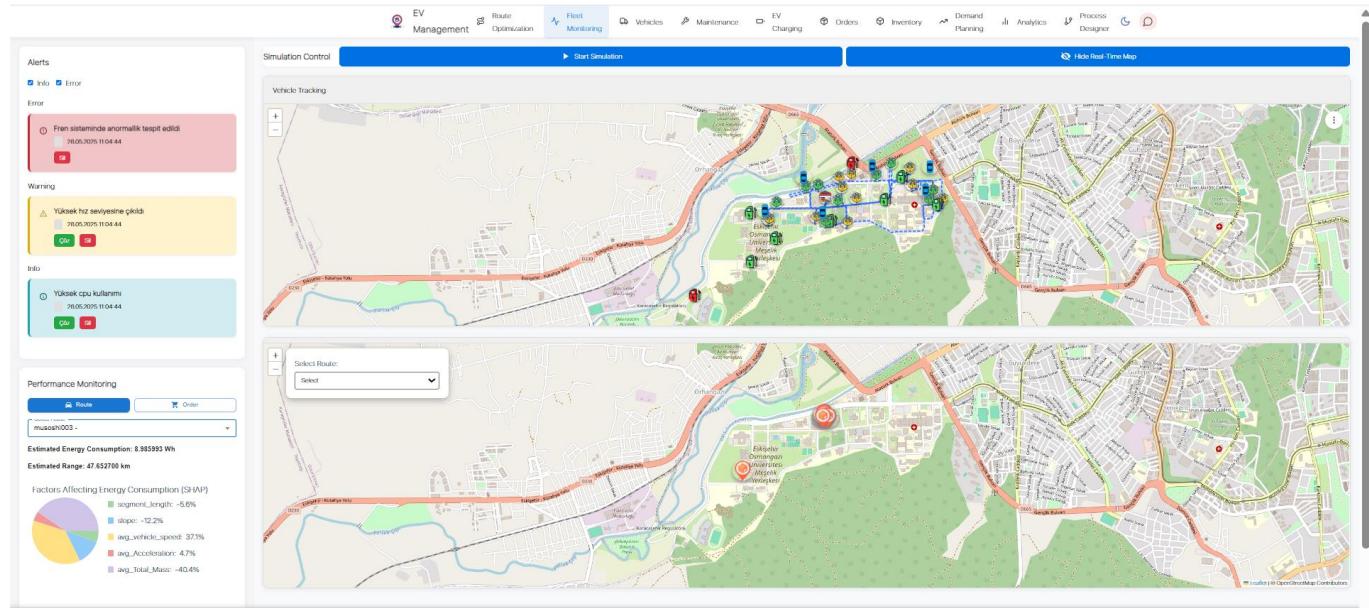


Figure 16: Araç Takip modülü Arayüzü

Şekilde görüldüğü gibi uygulamada araç takip ekranı bulunmaktadır. Bu ekran temel olarak haritalar, uyarı paneli, performans paneli olmak üzere üç bölümden oluşur. Her bölüm kendi içerisinde ele alarak arayüzleri tanıtlacaktır.

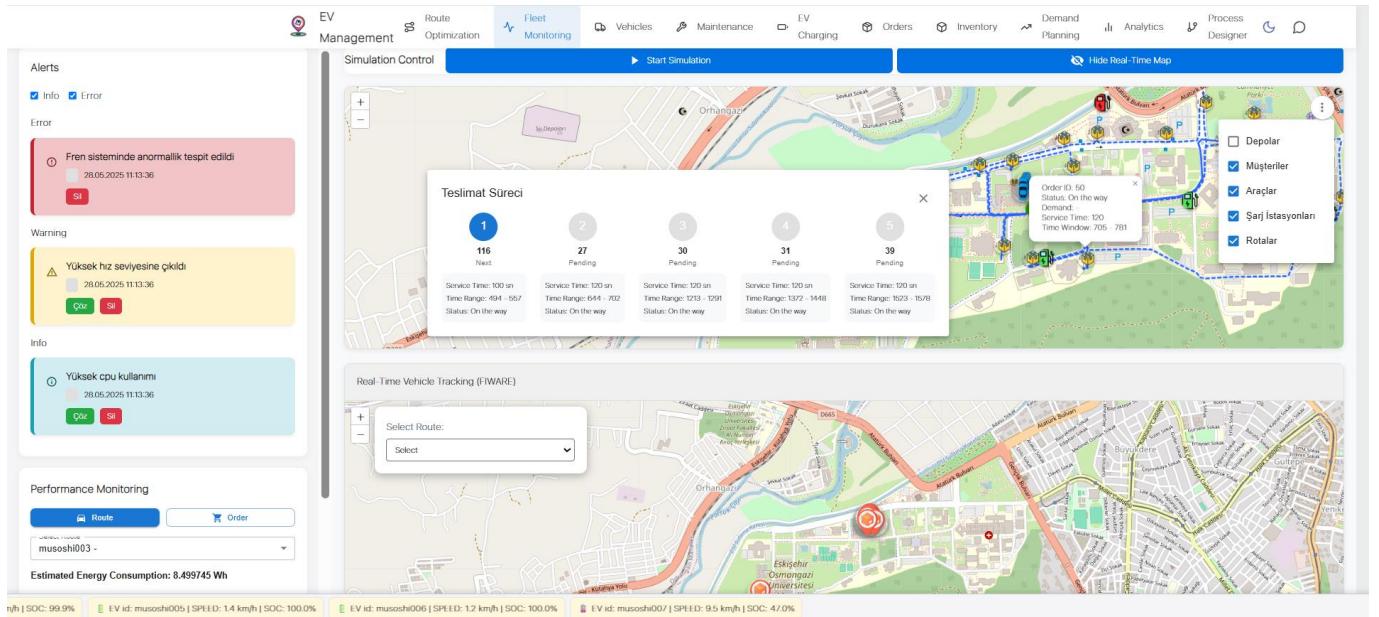


Figure 17: Araç Takip Ekranı - Harita

Haritalar, araçların hareketini anlık veya sumo simülasyonunu başlatıldığından görüntülenen bölümdür. İki harita bulunmaktadır. İlk harita sumo simülasyonunun izlendiği haritadır. Bu haritanın sağ üst köşesinde bir layers butonu bulunmaktadır. O kısımda işaretlenecek depolar, araçlar, rotalar, siparişler, şarj istasyonları gibi haritada görmek istenecek markerların görünürügü ayarlanabilecektir. Ayrıca haritadaki araç, sipariş, şarj istasyonlarının makerları üzerine tıklandığında ilgili bilgiler pop-up olarak gösterilecektir. Rotaya tıklandığında o rotaya ait siparişleri, sipariş durumlarını ve aracın o anda ilerlediği sipariş gösterilecektir. Diğer harita ise gerçek aracın hareketini görüntülediği FIWARE haritasıdır. Bu haritada gerçek aracın takibi yapılır..

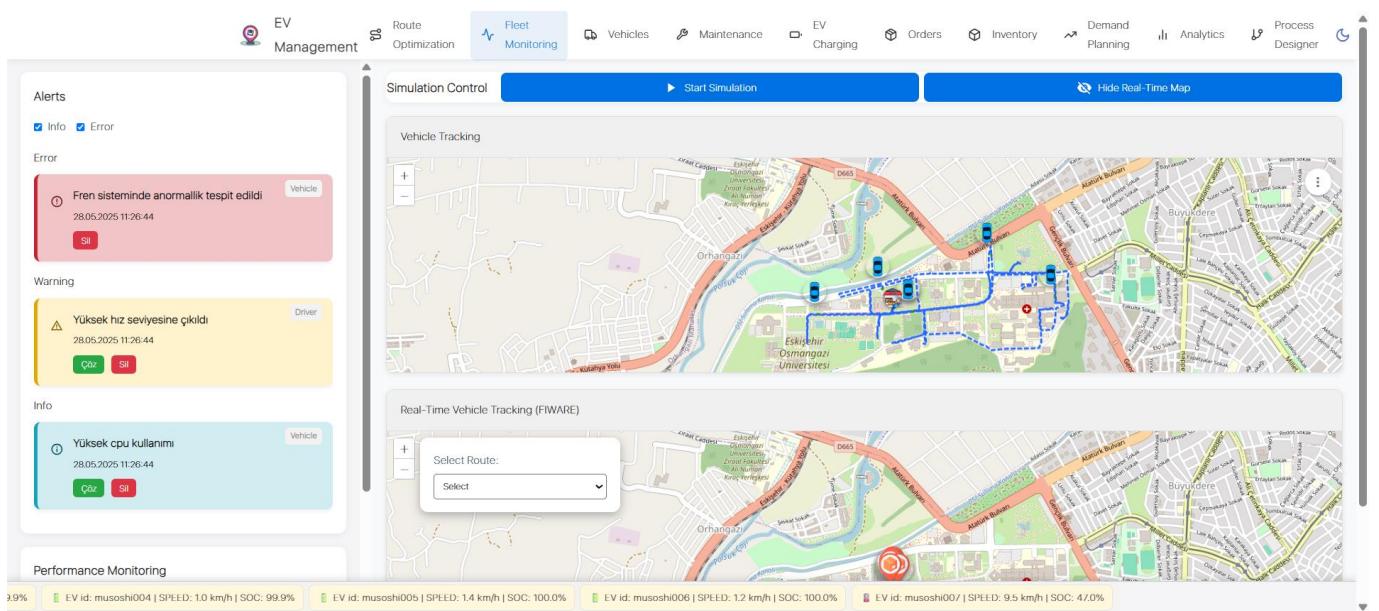


Figure 18: Araç Takip Ekranı – Uyarı Paneli

Uyarı panelinde ise mesajların hata, uyarı ve bilgilendirme olarak gruplu bir şekilde gösterilmesi sağlanacaktır. Her mesajın sahip olduğu kök nedeni mesajın bir kısmında belirtilecektir. Her mesajı silme veya çözme butonları eklenecektir ve buna göre gösterimi yapılacaktır. Ayrıca uyarı panelinde hata mesajları daima gösterilirken, uyarı ve bilgilendirmeler isteğe bağlı olarak gösterilebilecek bir yapıda oluşturulacaktır. Bunun içinde panel içerisinde checkboxlar eklenecektir.

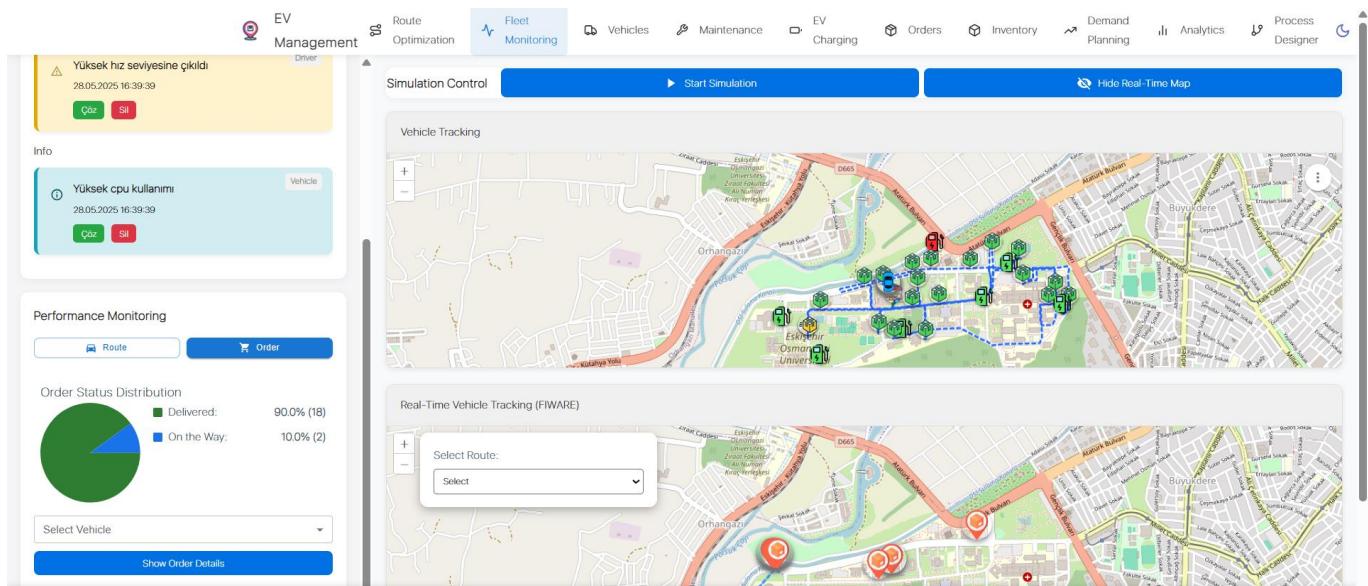


Figure 19: Araç Takip Ekranı – Performans Paneli

Araç takip sayfasının sol alt köşesinde anlık performans paneli bulunacaktır. Bu panelde rota ve sipariş alanları için butonlar yer alır. Rota için alt kısmında tahmini menzil ve enerji tüketimi model sonuçları bulunur. Ayrıca rota için enerjiye etki eden faktör pie chart da bu kısma eklenmesi planlanmaktadır. Sipariş için sipariş durumları ve araç görev tamamlama yüzdeleri bulunacaktır.

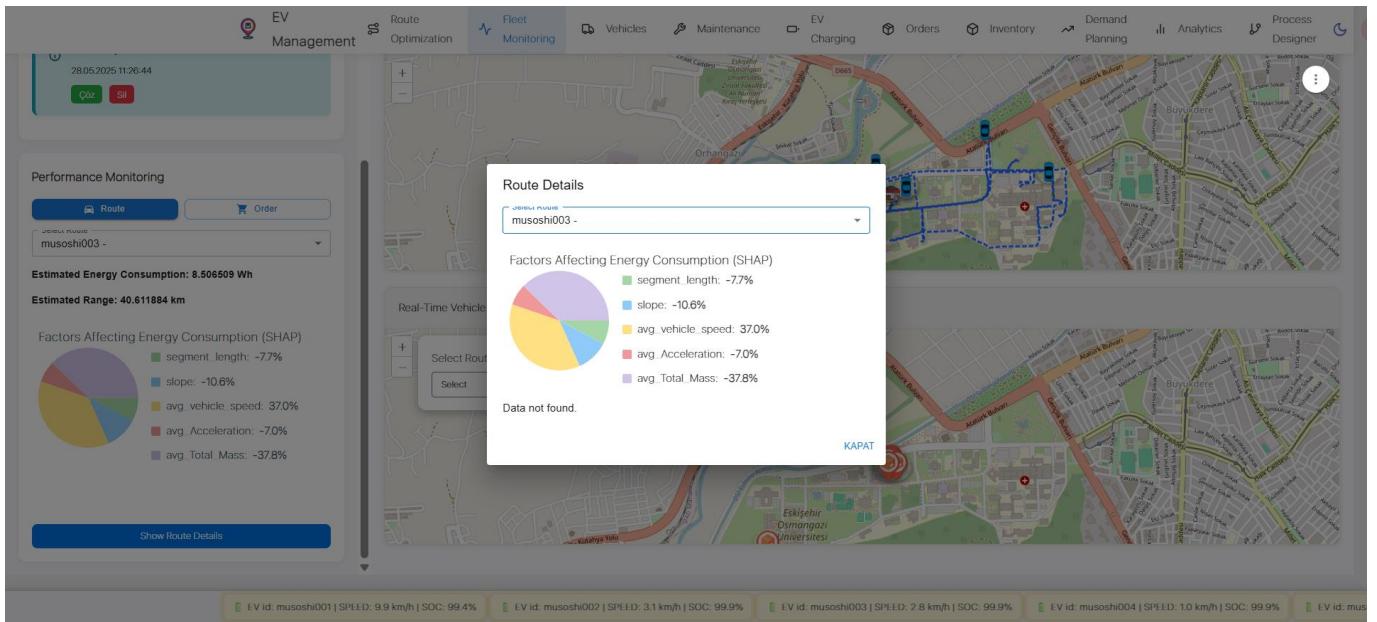


Figure 20: Araç Takip Ekranı – Performans Paneli – Rota detay

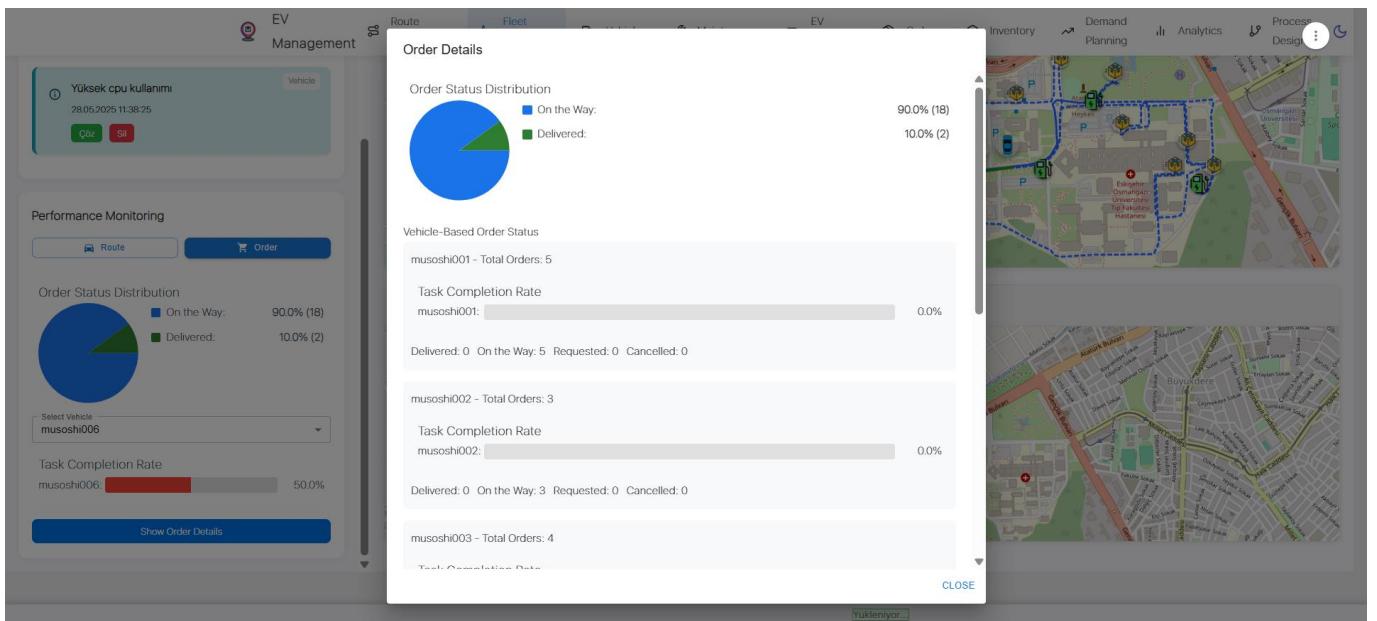


Figure 21: Araç Takip Ekranı – Performans Paneli – Sipariş Detay

Performans panelindeki detay butonlarına tıklandığında rota ve sipariş için detay bilgi pop-up ekranı olacaktır. Rota için bu ekranda hangi rotaya ait bilgileri görmek istiyorsak onun seçimini yapabileceğimiz bir box olacaktır. Buna ek olarak seçili rota için rotanın son 100 metrende harcadığı enerjiye etki eden faktörlerin yüzde dağılımı pie chart grafiği, tahmini menzil bilgisi gibi detaylar bulunacaktır. Sipariş için ise bütün siparişlerin durum pie chartı ve bütün araçların görev tamamlama bar chartları bulunacaktır.

Performans İzleme Modülü Arayüzü Tasarımı:

The screenshot shows the 'Performance Monitoring' module. At the top, there's a navigation bar with icons for EV Management, Route Optimization, Fleet Monitoring, Vehicles, Maintenance, EV Charging, Orders, Inventory, Demand Planning, Analytics, and Process Designer. Below the navigation bar is a toolbar with buttons for YESTERDAY, TODAY, LAST 7 DAYS, LAST MONTH, THIS MONTH, and THIS YEAR. There are also date input fields for Start Date and End Date, and an APPLY button. The main content area is titled 'Vehicle Performance Report' and lists various reports: Charging Station Usage Report, Fleet Operation Performance Report, Vehicle Maintenance Report, Carbon Gain Analysis Report, System Warning Log Report, Vehicle Comparison Report, Planned vs Actual Route Report, Algorithm Performance Report, General Order Report, System Performance Report, Driver Performance Report, Driver Success Report, and Customer Based Order Report. Each report has a dropdown arrow next to it.

Figure 22: Performans İzleme Ekranı 1

Performans izleme ekranı 2 kısımdan oluşmaktadır. Bunlar ondört raporun bulunduğu akordiyon yapısı ve grafiklerin farklı zaman dilimleri içerisinde görselleştirilmesini sağlayan tarih filtreleme alanıdır.

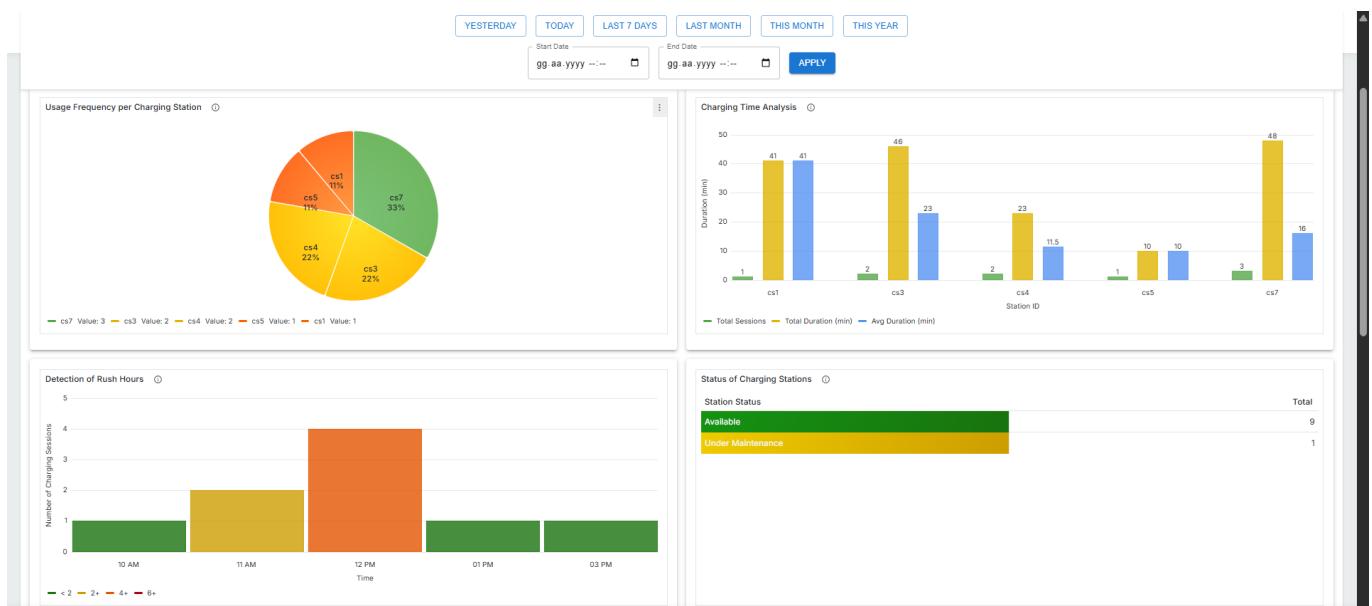


Figure 23: Performans İzleme Ekranı 2

Tarih filtreleme alanında, kullanıcının sık kullanacağı tarih aralıkları hazır olarak verilmektedir. Bunlar: dün, bugün, son 7 gün, geçen ay, bu ay ve bu yıl olarak belirlenmiştir. Ayrıca kullanıcı istediği tarih aralığını ve saatini takvim üzerinden seçebilmektedir.



Figure 24: Performans İzleme Ekranı Raporlama

Akordiyon yapısı içerisinde elektrikli araçlarda filo yönetim sistemi için gerekli olan on dört adet rapor ve rapor içeriği için üretilmiş yaklaşık elli adet tablo, grafik ve stat bulunmaktadır. Grafikler ve tablolar performans verilerinin izlenebilirliğini artırmak amacıyla değerlere göre renk geçişleri kullanılarak fark edilebilirlik artırılmıştır.

C.3.Test Tasarımı

C.3.1. Gereksinim analizlerinden teste yönelik hedeflerinin detaylandırılması

Yazılım testleri, uygulama için belirlenen gereksinimlerin karşılanabilmesi ve karşılanmadığı durumlar sonucu oluşabilecek herhangi bir sorunu belirlemek için süreç içerisinde yapılandırılır. Test tasarımını uygulamada bulunan araç takip ve performans izleme modüllerinin istenilen şekilde gerçekleştirildiğinin kontrolü ve yazılımın kullanım amacına uygun olması için önemlidir. Testlerin genel hedefleri filo yönetimi uygulaması içerisinde bulunan araç takip ve performans izleme modüllerinin kullanımı, performansı ve doğruluğunun takip edebilmektir. Projenin modüller bazında hedef metrikleri de aşağıda sunulmaktadır.

C.3.1.1. Araç Takip Modülü Test Hedefleri

- Araca, istasyonlara ve rotalara ait verilerin haritada doğru gösterilebilmesinin kontrolü yapılacaktır.
- Araca ait anlık verilerin doğru bir şekilde okunabildiğinin kontrolü yapılacaktır.
- SUMO simülasyonundan gelen araç verilerinin doğru bir şekilde işlenip kaydedildiğinin kontrolü yapılacaktır.
- Veritabanına kayıtlı anlık araç verilerinin veritabanından doğru bir şekilde çekilebilmesinin kontrolü yapılacaktır.
- Araç takip ekranındaki uyarı panelindeki verilerin doğru bir şekilde veritabanından okuma/güncelleme işlemlerinin test edilebilmesi yapılacaktır.

- Araç takip ekranındaki performans paneli için kullanılan makine öğrenimi model sonucunun kontrolü yapılmalı ve analizi gösterimi doğru olmalıdır.

C.3.1.2. Performans İzleme Modülü Test Hedefleri

- Performans izleme arayüzündeki tarih filtreleme fonksiyonun kullanıcı etkileşimine uygun çalıştığı ve panellere doğru zaman aralıklarını ettiği test edilecektir.
- Bugün, yarın, son 7 gün gibi butonlara tıklanarak grafiklerin doğru yüklendiği kontrol edilecektir.
- Performans verilerini gösteren gömülü Grafana panellerinin doğru gösterim yaptığı test edilecektir.
- Her rapora ait veri panellerinin Accordion yapısı içinde doğru şekilde gruplandırıldığı, genişletme ve daraltma işlemlerinin düzgün çalıştığı test edilecektir.
-

C.3.2. Fonksiyonel Test Tasarımı

Fonksiyonel test tasarımlı, üzerinde çalışılan araç takip ve performans izleme modüllerinin işlevsel gereksinimlerinin beklenen şekilde çalıştığını doğrulamak amacıyla hazırlanmıştır.

C.3.2.1. Birim (Unit) Testler:

Araç Takip Modülü Birim Testleri:

1. Bu birim testi ile araç izleme ekranı yüklenliğinde, o sayfada araçların takibini yaptığı haritanın doğru bir şekilde render edilip edilmediğinin kontrolü yapılır.

```
test('Harita container render ediliyor', () => {
  render(<FleetMonitoringMap orders={[]} chargingStations={[]} /);
  expect(screen.getByTestId('fleet-map-container')).toBeInTheDocument();
});
```

Figure 25: Test 1

2. Alert oluşturulduğunda bir alertin ekranga görünüp görünmediğinin kontrol testi yapılır

```
test('Tek bir alert ekranda görünüyor', () => {
  render(<FM_AlertPanel type="Error" message="Hata!" />);
  expect(screen.getByText(/Hata!/)}.toBeInTheDocument());
});
```

Figure 26: Test 2

Performans Modülü Birim Testler:

1. Performans izleme ekranında tarih filtreleme ve tarih seçim butonlarının doğru zaman aralığı üretmesi için birim test bulunmaktadır.

```

test('Tarih butonları doğru zaman aralığı üretiyor', () => {
  render(<Performans />);
  const todayBtn = screen.getByRole('button', { name: /today/i });
  fireEvent.click(todayBtn);

  // Burada state değişimini test etmek için, iframe src'sinde from ve to değerlerini kontrol edeceğiz
  const iframes = screen.getAllByTitle(/panel/i);
  expect(iframes.length).toBeGreaterThan(0);

  // Bugünün başlangıcı ve bitisi
  const todayStart = dayjs().startOf('day').valueOf();
  const todayEnd = dayjs().endOf('day').valueOf();

  // Her iframe'in src'sinde from ve to parametreleri olmalı
  iframes.forEach((iframe) => {
    expect(iframe.src).toContain(`from=${todayStart}`);
    expect(iframe.src).toContain(`to=${todayEnd}`);
  });
});
});

```

Figure 27: Test 3

2. Iframe panellerine from ve to değerlerinin doğru aktarılması birim test ile kontrol edilmiştir.

```

test('Iframe panellerine from ve to değerleri doğru aktarıyor', () => {
  render(<Performans />);
  // Tarih alanlarını doldur
  const startInput = screen.getByLabelText(/start date/i);
  const endInput = screen.getByLabelText(/end date/i);
  const applyBtn = screen.getByRole('button', { name: /apply/i });

  // Örnek tarih aralığı
  fireEvent.change(startInput, { target: { value: '2024-05-01T00:00' } });
  fireEvent.change(endInput, { target: { value: '2024-05-10T23:59' } });
  fireEvent.click(applyBtn);

  const from = dayjs('2024-05-01T00:00').valueOf();
  const to = dayjs('2024-05-10T23:59').valueOf();

  // Her iframe'in src'sinde from ve to parametreleri olmalı
  const iframes = screen.getAllByTitle(/panel/i);
  iframes.forEach((iframe) => {
    expect(iframe.src).toContain(`from=${from}`);
    expect(iframe.src).toContain(`to=${to}`);
  });
});

```

Figure 28: Test 4

3. Accordion yapısının açılıp kapanma işlevi için kullanıcı etkileşimine dayalı birim test yapılmıştır.

```
test('Accordion açılıp kapanabiliyor', () => {
  render(<Performans />);
  // ilk accordion başlığını bul
  const accordionSummary = screen.getByText(/vehicle performance report/i);
  // Accordion detayları ilk başta açık olmalı
  expect(accordionSummary.closest('.Mui-expanded')).toBeTruthy();

  // Kapatmak için tıkla
  fireEvent.click(accordionSummary);
  // Accordion detayları kapanmalı
  expect(accordionSummary.closest('.Mui-expanded')).toBeFalsy();
});

// Tekrar açmak için tıkla
fireEvent.click(accordionSummary);
expect(accordionSummary.closest('.Mui-expanded')).toBeTruthy();
});
```

Figure 29: Test 5

C.3.2.2. Entegrasyon Testleri:

Projede birçok entegrasyon bulunmaktadır. Entegrasyon testleri ile, proje içerisinde bulunan farklı bileşen ve yazılımların birbirleri ile etkileşimleri test edilmektedir.

1. Araç takip izleme modülü için kullanılan makine öğrenme modelinin, araçtan gelen verileri doğru bir şekilde işleyip araç takip sayfasındaki performans panelinde kalan menzil ve şarja etki eden faktörler gibi bilgilerin sunulup sunulmadığı bu entegrasyon testi ile kontrol edilir.

```
13  test('ML API: Araç verisi gönderildiğinde kalan menzil ve SHAP faktörleri dönüyor', async () => {
14    const input = {
15      avg_vehicle_speed: 35,
16      segment_length: 500,
17      avg_Acceleration: 0.2,
18      avg_Total_Mass: 1800,
19      slope: 1.5,
20      soc: 80,
21      remaining_energy: 12000,
22      nonlinear_energy: 150,
23      segment_count: 10
24    };
25
26    const response = await fetch('http://localhost:5002/predict', {
27      method: 'POST',
28      headers: { 'Content-Type': 'application/json' },
29      body: JSON.stringify(input)
30    });
31
32    expect(response.status).toBe(200);
33    const data = await response.json();
34
35    expect(typeof data.prediction).toBe('number');
36    expect(typeof data.remaining_range).toBe('number');
37    expect(Array.isArray(data.shap_values)).toBe(true);
38    expect(data.shap_values.length).toBeGreaterThan(0);
39  });
40
41  // 2. MongoDB ve MariaDB Entegrasyonu
42  describe('Veritabanı entegrasyonu', () => {
43    let pool;
    ...
```

Figure 30: Test 6

2. Bu entegrasyon test kodu MariaDB entegrasyonlarını kontrol eder. Örnek araç verisinin veri tabanına doğru kaydedilip kaydedilmmediği test edilir.

```
41 // 2. MongoDB ve MariaDB Entegrasyonu
42 describe('Veritabanı entegrasyonu', () => {
43   let pool;
44   beforeAll(async () => {
45     // MongoDB bağlantısı
46     await mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true, useUnifiedTopology: true });
47     // MariaDB bağlantısı
48     pool = mariadb.createPool({
49       host: process.env.DB_HOST || 'localhost',
50       port: process.env.DB_PORT || 3306,
51       user: process.env.DB_USER || 'root',
52       password: process.env.DB_PASSWORD || 'password',
53       database: process.env.DB_NAME || 'vehicle_tracking'
54     });
55   });
56   afterAll(async () => {
57     await mongoose.disconnect();
58     await pool.end();
59   });
60   test('MariaDB: Araç verisi eklenip okunabiliyor', async () => {
61     const conn = await pool.getConnection();
62     await conn.query("INSERT INTO vehicle_tracking_fiware (vehicle_id, latitude, longitude, speed, timestamp) VALUES ('test_vehicle', 39.75, 30.48, 50, new Date())");
63     const rows = await conn.query("SELECT * FROM vehicle_tracking_fiware WHERE vehicle_id = ?", ['test_vehicle']);
64     expect(rows.length).toBeGreaterThan(0);
65     await conn.query("DELETE FROM vehicle_tracking_fiware WHERE vehicle_id = ?", ['test_vehicle']);
66     conn.release();
67   });
68 });
69 });
70 
```

Figure 31: Test 7

3. Sumodan verileri bir python server ile alınması planlandı. Python Traci kütüphanesi kullanılarak alınan araç verilerinin doğru bir şekilde alındığının kontrolü için bir entegrasyon testi bulunmaktadır.

```
71 // 3. SUMO Python Server Entegrasyonu
72 test('SUMO serverdan araç verisi alınabiliyor', async () => {
73   const response = await fetch('http://localhost:5002/sumo-data');
74   expect(response.status).toBe(200);
75   const data = await response.json();
76   expect(data).toHaveProperty('vehicle_id');
77   expect(data).toHaveProperty('latitude');
78   expect(data).toHaveProperty('longitude');
79   expect(data).toHaveProperty('speed');
80 });
81 });
```

Figure 32: Test 8

C.3.3. Performans Test Tasarımı

Projede performans testleri ile projenin işlevsel gereksinimlerinin performans kontrolü için kullanılır. Sistemin ve modüllerin yoğun veri yükü, sistem kaynak kullanımı gibi durumlarla karşılaşlığında olası sorunları test etmek için yapılacaktır. Bu testler sayesinde projenin olabilecek en optimize şekilde işlemesi ve ileride gerçekleşebilecek olası hataların önüne geçebilmek için önemlidir.

C.3.3.1. Araç Takip Modülü Test Senaryoları

1. API Veri Doğruluk Testi

Amaç: Araç konumlarının, sistem API'si üzerinden eksiksiz ve doğru şekilde okunabilir olduğunu doğrulamaktır.

Yöntem: API'den üzerinden veri çekilmiş, dönen cevabın bir dizi (array) formatında olduğu ve en az bir araç verisi içerdiği kontrol edilmiştir. Ayrıca her bir araç nesnesinde bilgi alanlarının eksiksiz bulunup bulunmadığı test edilir.

Beklenen Sonuç: API'den başarılı şekilde veri dönmesi ve her araç kaydında gerekli tüm alanlar eksiksiz olması beklenmektedir.

2. Gecikme (Latency) Testi

Amaç: Sistemden dönen araç konumu verilerinin gerçek zamanlı olup olmadığı belirlenmesidir.

Yöntem: API'den endpoint üzerinden elde edilen araç verilerinin her biri içinson güncelleme alanındaki zaman damgası alınır ve bu bilginin sistem saatine göre 10 dakikadan eski olup olmadığı kontrol edilir.

Beklenen Sonuç: Yapılan kontrol sonucunda tüm araçların konum verileri 10 dakika içerisinde güncellenmiş olarak görülmeli beklenmektedir.

3. Yanıt Süresi (Performans) Testi

Amaç: Araç konum verilerinin sistem API'si üzerinden ne kadar sürede elde edilebildiğini ölçmektedir.

Yöntem: Test başında zaman kaydedilir ve, ardından aynı API endpoint'ine istek atılmış ve cevabın alınmasıyla birlikte süre tekrar ölçülecek toplam yanıt süresi hesaplanır. Beklenen maksimum süre 2 saniye olarak belirlenir.

Beklenen sonuç: Sonucun süresi <2 saniyeden az olması beklenmektedir.

C.3.3.2. Performans Modülü Test Senaryoları

1. Tarih Filtreleme Testi

Amaç: Seçilen tarih aralığına göre iframe panellerinin veri güncellemesi yapıp yapmadığı kontrol edilmektedir.

Yöntem: Hızlı tarih butonları ve manuel tarih giriş alanı kullanılarak filtreleme yapılır.

Beklenen sonuç: URL parametrelerinde doğru timestamp değerleri yer almalıdır.

2. Hızlı Tarih Seçimi Testi

Amaç: "Today", "Yesterday" gibi butonların doğru zaman aralığı üretip üretmediği test edilmektedir.

Yöntem: Her butona tıklanarak onSelect fonksiyonunun tetiklenmesi ve iframe'lerin doğru zaman parametresi ile çalışması gözlemlenir.

Beklenen sonuç: Panellerde gösterilen veriler seçilen butona göre doğru şekilde filtrelenmiş olmalıdır.

C.4.Yazılım Tasarımı

C.4.1.Gereksinime Bağlı Tasarım Kalıpları Seçimi

1. Gözlemci (Observer) Tasarım Kalıbı

Proje içerisinde araçlardan alınan veriler değişir . Bu değişiklikleri dinleyen harita, uyarı sistemi veri tabanı gibi birçok birleşen vardır. Aracın, konum hız enerji bilgileri anlık olarak değişip güncellendikçe bağlı olan Harita, log, performans paneli bilgilendirilir. Bu tasarım kalıbı ile sistemin modülerliği ve esnekliği artar ve ileride eklenebilecek yeni bileşenler için kolaylık sağlar.

2. Singleton Tasarım Kalıbı

Projede MongoDB ve MariaDB kullanılıyor. Bu veritabanı bağlantılarının daha kontrollü olmasını sağlamak için singleton tasarım kalıbı kullanılır. Bununla uygulama boyunca tek bir veritabanı nesnesi oluşturulur ve paylaşılır. Bu sayede sistemin veritabanı bağlantı yönetimi sağlanır

C.4.2. UML Kullanarak Tasarımı Diyagramları Oluşturma

Kullanım durumu diyagramları, ile sistemdeki aktörlerin gerçekleştirdiği etkileşimler ele alınmıştır. Aktör olarak filo yöneticisi bulunmaktadır. Araç Takip ve Performans izleme modülleri için gerçekleştirilecek durumlar ve aktörlerin o durumlar ile etkileşimleri için kullanım durumu diyagramı çizilmiştir

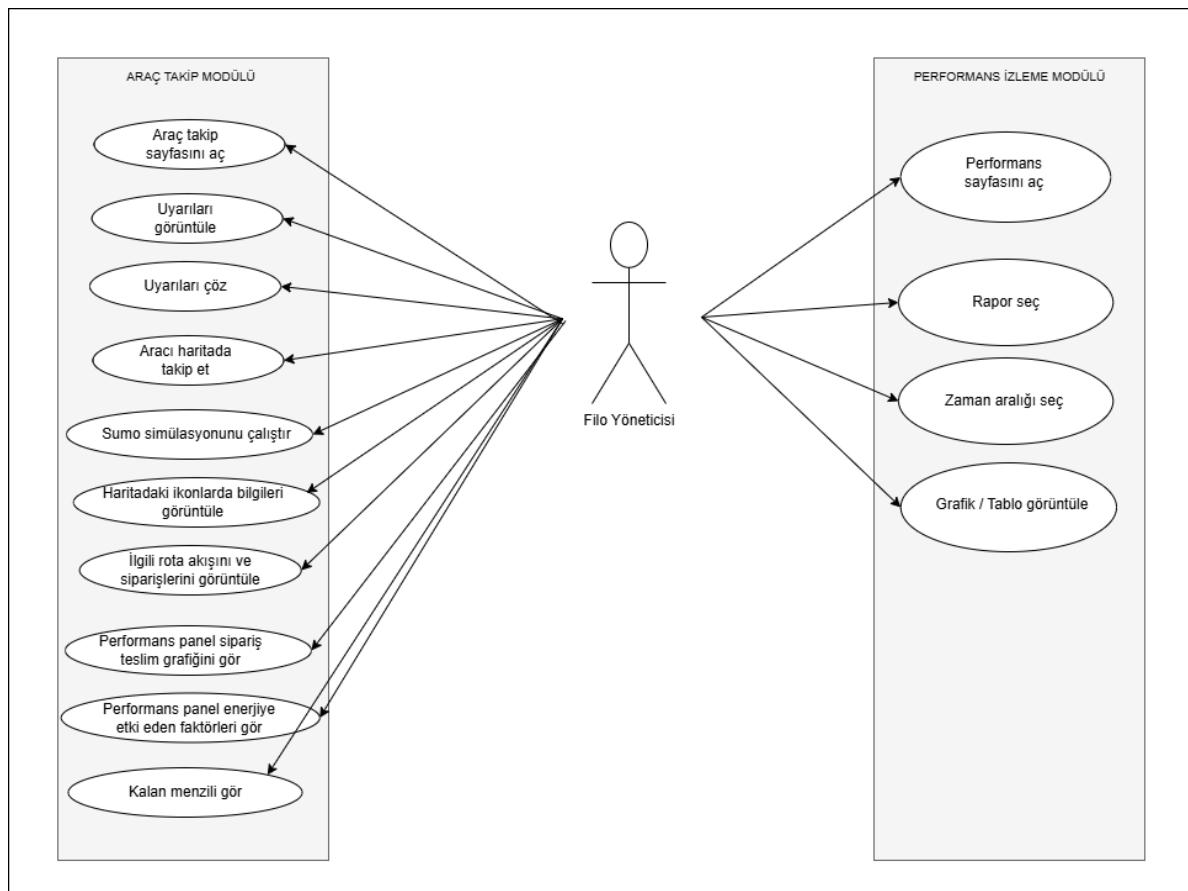


Figure 33: Use Case Diagram

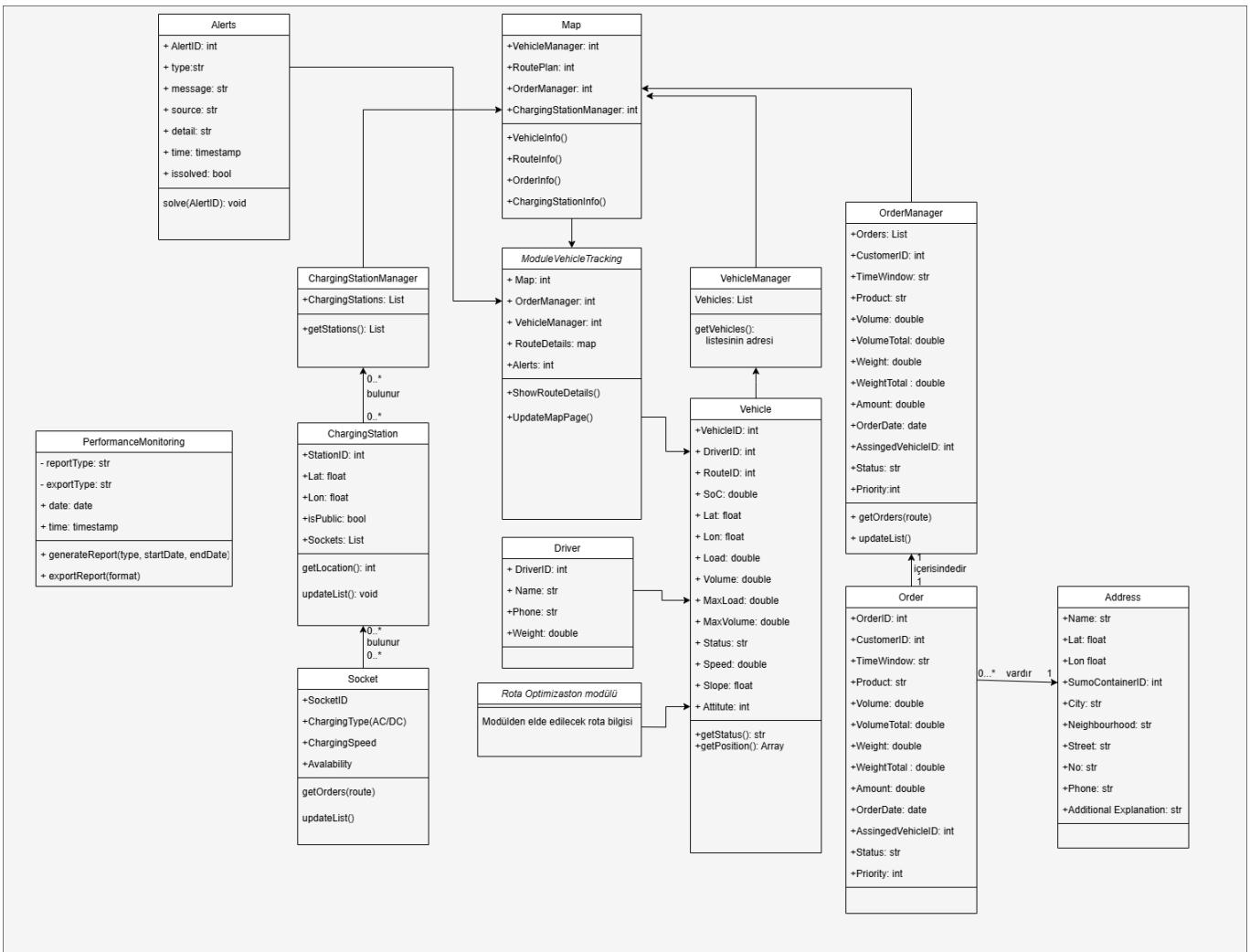


Figure 34: UML Diagram

Bu UML diyagramı, elektrikli araçlar için geliştirilen araç Takip ve Performans izleme ekranlarının temel bileşenlerini ve aralarındaki ilişkileri göstermektedir. Sistemin merkezinde yer alan ModuleVehicleTracking sınıfı, farklı yöneticileri (VehicleManager, RoutePlan, OrderManager, RouteDetails) ve olayları (Alerts) bir araya getiren ana modül olarak konumlandırılmıştır. Aynı şekilde Map sınıfı da farklı bileşenlerin kullanıldığı ve haritada kullanılabilen yapıları tutan bir sınıfır. VehicleManager, filodaki araçları yöneten bir modüldür ve her araç nesnesi, Vehicle sınıfı ile temsil edilmiştir. Araçlar, konum (Lat, Lon), hız, eğim, yük bilgisi, batarya durumu (SoC) ve atanmış şoför (Driver) gibi detaylı özelliklere sahiptir. OrderManager, sistemdeki tüm siparişleri yönetir. Her sipariş, Order sınıfı ile temsil edilir ve müşteri bilgisi, zaman aralığı (TimeWindow), ürün detayları, miktar, hacim, ağırlık, öncelik ve atanmış araç ID'si gibi birçok alanı içerir. Siparişler, Address sınıfı ile tanımlanan bir teslimat adresiyle ilişkilidir. Bu adresler, konum bilgisi ve detaylı açıklamalar içerir. ChargingStationManager modülü, sistemdeki tüm şarj istasyonlarını yöneten bir yapıdır ve her bir istasyon ChargingStation sınıfı ile modellenmiştir. Her istasyon, konum, halka açık olup olmadığı gibi özelliklerin yanında, Socket nesneleri içerir. Socket'ler, AC/DC tipi, şarj hızı ve uygunluk bilgisi ile tanımlanmıştır. Sistem, gerçek zamanlı uyarılar (Alerts) modülü ile donatılmıştır. Bu modül, sistemde oluşan olaylara dair mesajları, kaynakları, zaman damgasını ve çözülme durumunu içerir. Bu uyarılar gerekiğinde çözülmek üzere işaretlenebilir. Ayrıca sistemde, performans analizlerinin yapılabilmesi için PerformanceMonitoring modülü yer almaktadır. Bu modül, rapor türü, tarih aralığı ve dışa aktarım formatına göre performans verilerini dışa aktarma yeteneğine sahiptir. Genel olarak bu UML yapısı, bir elektrikli araç filosunun

tüm yönleriyle dijital ortamda izlenmesini, siparişlerin ve rotaların yönetilmesini, şarj altyapısının kontrolünü ve sistem performansının sürekli olarak izlenmesini sağlayan modüler ve genişletilebilir bir mimariyi temsil etmektedir.

C.5.Veri Tabanı Tasarımı

Bu proje kapsamında veri tabanı tasarımları bulunmamaktadır.

C.6. Donanım Tasarımı

Bu proje kapsamında donanım tasarımları bulunmamaktadır.

D. UYGULAMA

D.1. Geliştirilen Sistemin Sistem Tasarımlarını Karşılanması Değerlendirilmesi

Geliştirilen araç takip ve performans izleme sistemi, belirlenen işlevsel, sistem ve kullanıcı arayüzü ile veri gereksinimlerini eksiksiz karşılayacak şekilde tasarlanarak uygulanmıştır.

Araç Takip Modülü, sistemde bulunan araçların konum, batarya seviyesi, anlık enerji tüketimi gibi kritik bilgilerini gerçek zamanlı olarak veritabanından çekerek kullanıcıya harita arayüzü üzerinden sunmaktadır. Kullanıcı arayüzü, renk kodları ve ikonlar aracılığıyla sipariş durumlarını açıkça ayırt edebilmeyi sağlamakta, araçların dinamik hareketi, teslimat listeleri ve sipariş detayları sorunsuz şekilde görüntülenebilmektedir. Ayrıca kritik durumlar için uyarılar ve bildirimler gerçek zamanlı olarak kullanıcılara iletmektedir. Bu özellikler, sistemin gereksinimlerde belirtilen işlevsel ve arayüz gereksinimlerine uygun olarak çalıştığını göstermektedir.

Performans İzleme Modülü ise geniş veri setlerini zaman bazlı filtreleme seçenekleriyle analiz ederek, filo genel performansı, sürücü başarıları, araç bakım durumları ve şarj istasyonu kullanıcıları gibi kritik metriklerin raporlanmasılığını sağlamaktadır. Grafiksel görselleştirmeler (sütun grafik, çizgi grafik, pasta grafik) kullanıcıların verileri hızlı ve doğru şekilde anlamalarına yardımcı olmakta, ayrıca rota optimizasyonu ve algoritma performans raporları sistemin optimizasyon çalışmalarına ışık tutmaktadır. Sistem performans izleme ve uyarı mekanizmaları da sunucu tepki süresi, CPU ve bellek kullanımı gibi metriklerle takip edilerek, sistemin sürekli izlenebilirliği sağlanmıştır.

Genel olarak, geliştirilen sistem tasarımı, belirlenen tüm gereksinimleri kapsamlı biçimde karşılamakta ve dinamik bir filo yönetim platformu olarak kullanıcıya yüksek kullanılabilirlik ve performans sunmaktadır. Tasarımda kullanılan harita entegrasyonu, dinamik görselleştirmeler, bildirim mekanizmaları ve kapsamlı raporlama yapısı, sistemin işlevsellliğini ve kullanıcı deneyimini artırmaktadır. Bu doğrultuda, sistem öngörülen amaçlar doğrultusunda sorunsuz çalışmakta ve gelecekte ihtiyaç duyulacak genişletme veya entegrasyonlara da uygun altyapı sunmaktadır.

D.2. Kullanıcı ve Sistem Arayüzü Gerçeklemeleri

D.2.1 Araç Takip Modülü

D.2.1.1 Genel Yapı

Araç Takip ekranı, elektrikli araç filosunun hem gerçek zamanlı olarak hem de SUMO simülasyon ortamında izlenmesini sağlayan, kullanımı kolay bir arayüz sunmaktadır. Bu ekran 3 kısımdan oluşmaktadır. Bunlar Filo yönetimindeki uyarıların yer aldığı alert paneli, araç enerji menzil ve sipariş durumunun izlendiği Performans izleme paneli, ve haritaların bulunduğu alandır.

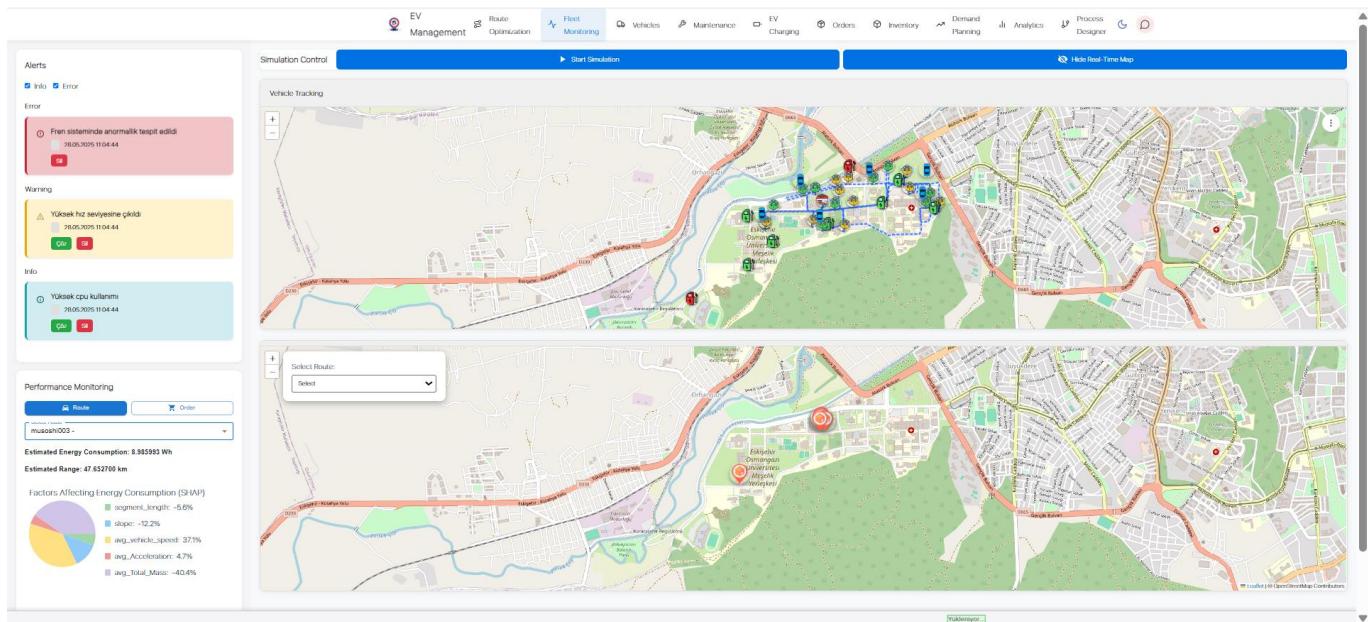


Figure 35: Araç Takip Genel Yapı

D.2.1.2 Harita Alanı

Harita bileşeni, iki ayrı harita ile sunulmaktadır.

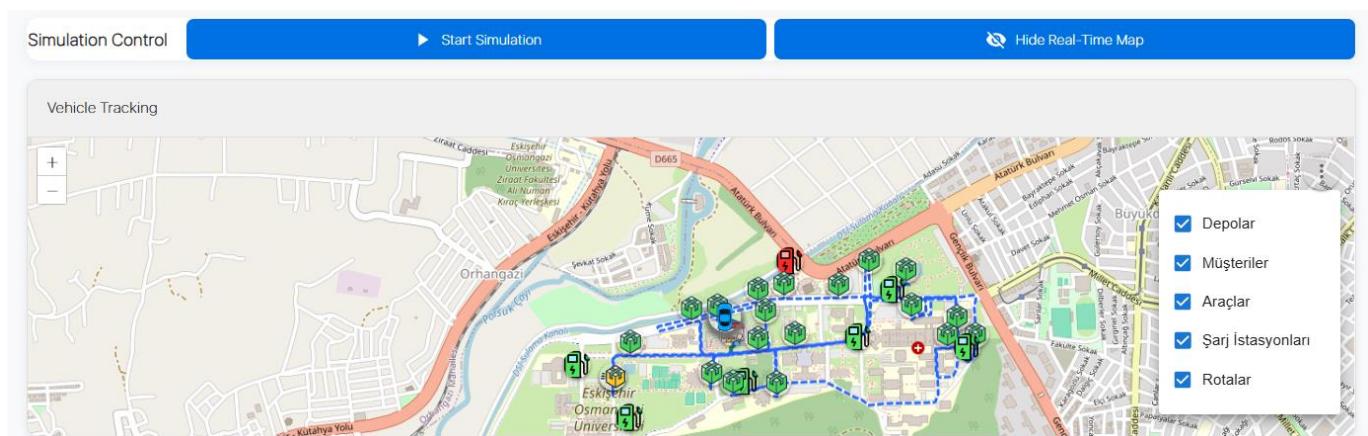


Figure 36: Sumo Harita

İlk harita, SUMO simülasyonunda yer alan araçların anlık konumlarını ve hareketlerini göstermektedir. Bu harita, sistem geliştiricilerin simülasyon temelli analizler yapmasına olanak sağlar. Haritanın sağ üst köşesinde yer alan

katman seçme (layer) butonu sayesinde kullanıcı, görmek istediği nesneleri seçebilir. Bu seçeneklerde depolar, müşteriler, araçlar, şarj istasyonları ve rotalar bulunmaktadır. Default olarak hepsi işaretli olarak gelmektedir. Bu layer sistemi ile harita üzerindeki bilgi yoğunluğu, kullanıcının ihtiyacına göre ayarlanabilir.

Harita üzerindeki markerlara(icons) tıklanılarak detaylı bilgilere ulaşılabilmektedir.

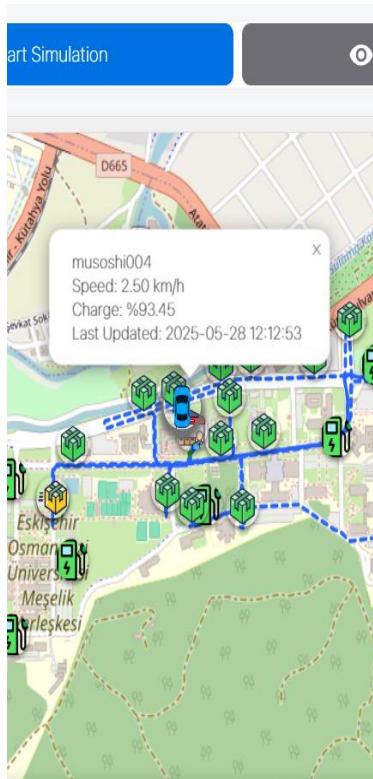


Figure 37: Araç Detay

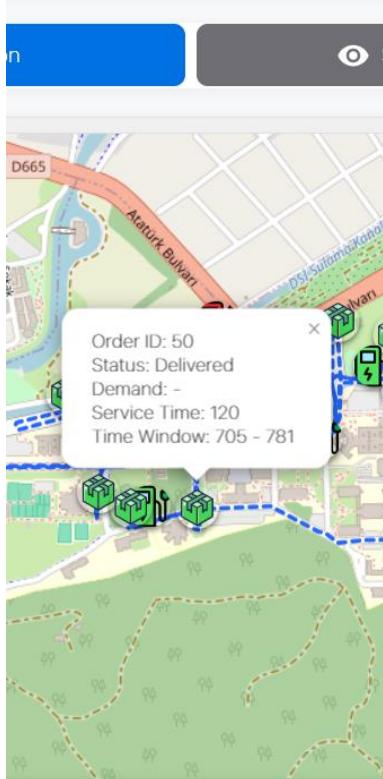


Figure 38: sipariş detay

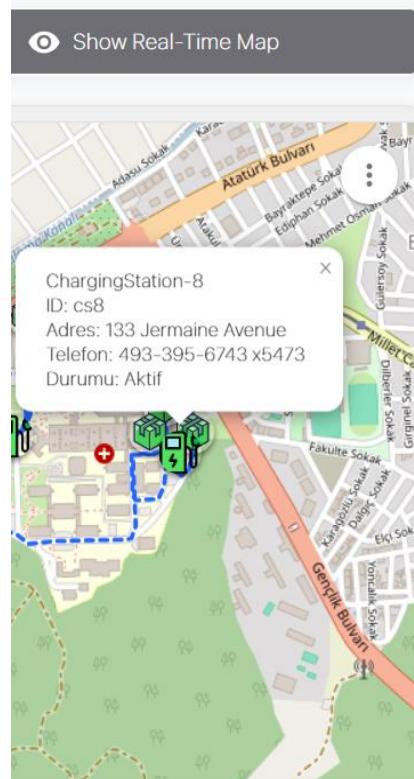


Figure 39: şarj istasyonu detay

Araç ikonu popup kısmında araç adı, hızı, şarj durumu ve son güncellenme durumu görülmektedir. Müşteri ikonuna tıklandığında karşımıza çıkan popup'da sipariş ID, sipariş durumu, talep, servis zamanı ve siparişin zaman penceresi bilgileri bulunmaktadır. Şarj istasyonu ikon popup kısmında ise istasyon adı, ID'si, adresi, telefon numarası ve istasyon durumu bilgileri görüntülenebilmektedir. Her marker için durumuna göre renk kodlarıyla gösterilmektedir. Şarj istasyonu aktif ise yeşil pasif ise kırmızı ikon ile gösterilir. Müşteri teslimatlar yolda ise turuncu, teslim edildi ise yeşil, iptal ise kırmızı görülmektedir.

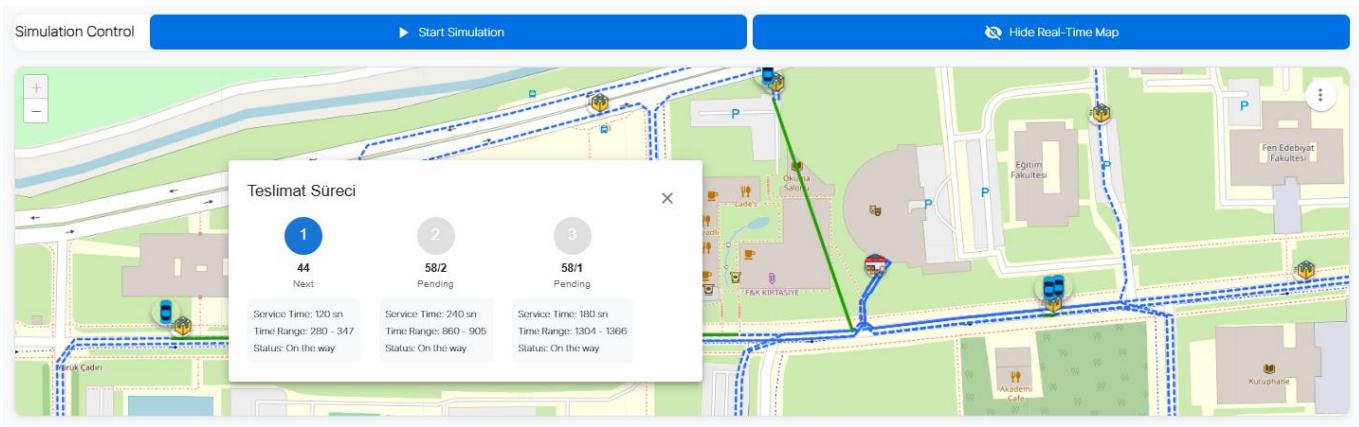


Figure 40: Teslimat listesi

Haritada araç ilerlerken rotaya tıklandığında o araca ve rotasına ait siparişlerin listesi, güncel durumları ve araç hangi siparişi teslim ediyor bilgisi gösterilecek bir popup panel çıkar. Bu panelde siparişlerin detaylarını ve tamamlanma durumlarını görüntüleyebiliriz. Bu kısım uygulama kullanıcılarının sipariş takip sürecini kolaylaştırır.

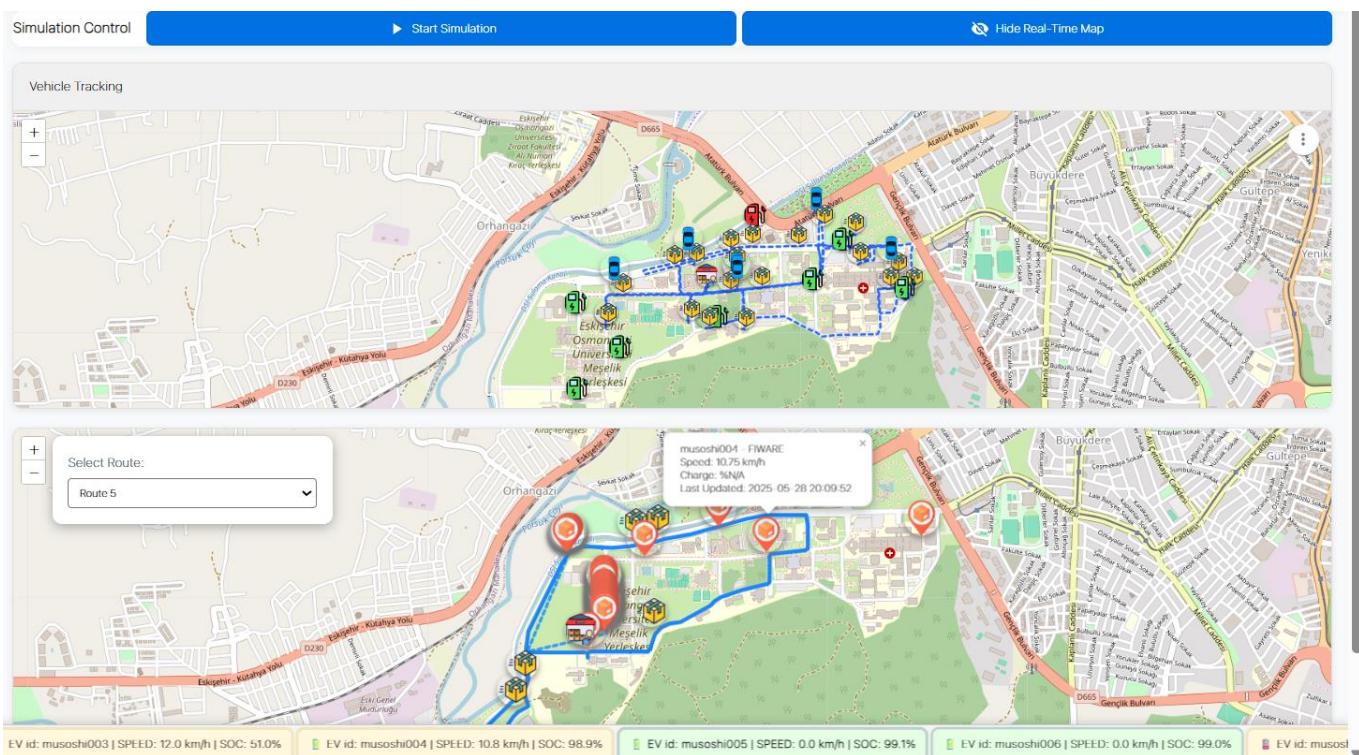


Figure 41: Harita Fiware

İkinci harita ise gerçek araçların FIWARE üzerinden alınan anlık verilerine dayanmaktadır. Bu harita, canlı filo takibi için kullanılır ve araçların gerçek zamanlı olarak konumlarının takip edilmesini sağlar. Burada araca tıklandığında yine araç adı, hızı, şarj durumu ve son güncellenme durumu görülmektedir. FIWARE haritasının sol üstünde bir selection box bulunmakta burada aracın gideceği rotayı seçerek haritaya yansıtılabilir. Bu sayede gerçek aracın ilerleyişi görüntülenir. Bu harita sayfada bulunan bir buton ile kullanılmaması durumunda ekranda

saklanır ve bu şekilde kullanım kolaylığı sağlanır. Bu harita ile sistem yalnızca simülasyonla sınırlı kalmaz, gerçek operasyonları da destekler.

D.2.1.3 Uyarı Paneli

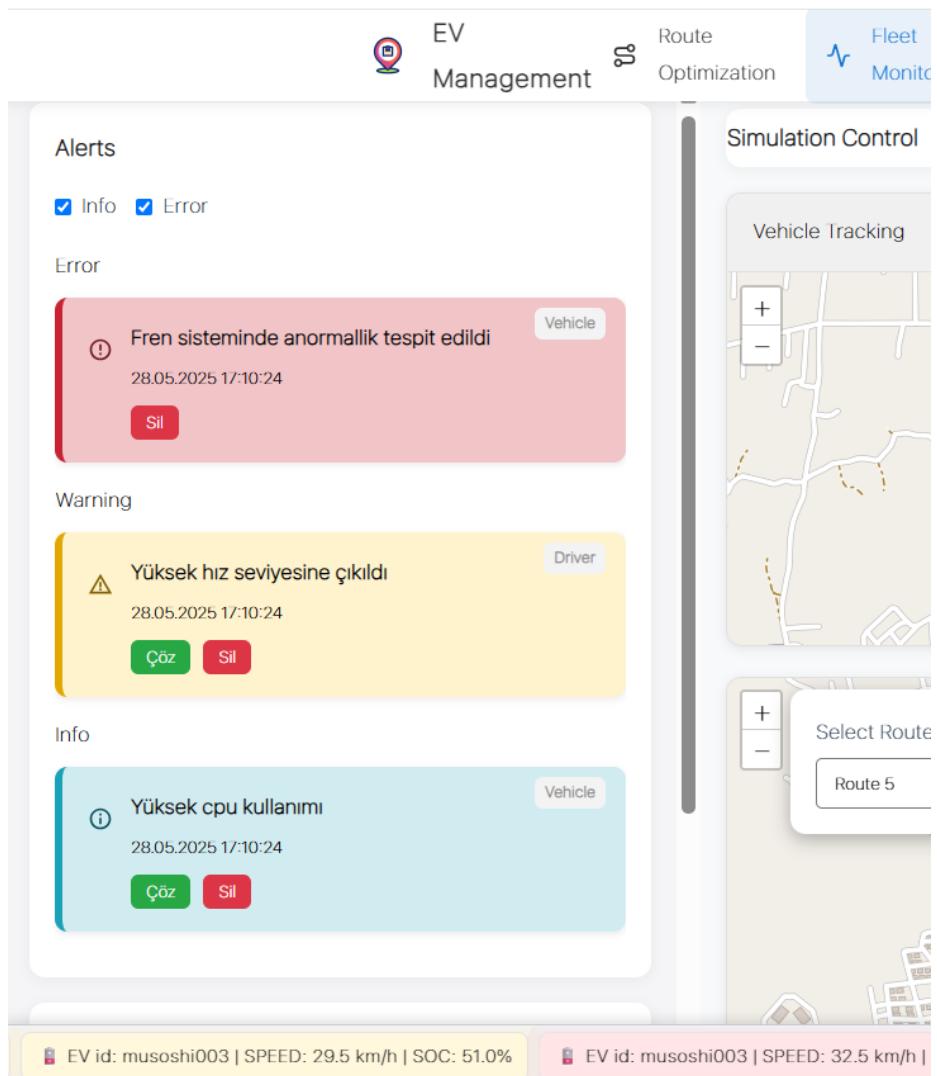


Figure 42: Uyarı paneli

Uyarı paneli, sistemde oluşabilecek hata, uyarı ve bilgi mesajlarının kullanıcıya kategorize edilmiş şekilde sunulmasını sağlar. Hata mesajları sistem tarafından kritik olarak kabul edilir ve daima gösterilirken, uyarı ve bilgi mesajları isteğe bağlı olarak görüntülenebilir. Kullanıcının bu mesajları filtreleyebilmesi için panel içerisinde checkbox filtre seçenekleri sunulmuştur.

Her mesaj bileşeni içinde mesajın türü (Hata, Uyarı, Bilgi), oluşma zamanı, sağ üstte kök nedeni ve çözüm butonları (Sil, Çöz) yer almaktadır. Bu sayede kullanıcı, sistemde oluşan problemleri takip edebilir ve çözüm sürecine müdahale edebilir. Mesajlar renk ve ikon kodları ile ayırt edilmiştir: kırmızı (hata), sarı (uyarı), mavi (bilgi). Kritik mesajlarda dikkat çekiciliği artırmak için yanıp sönen ikonlar da kullanılmaktadır. Çözülen mesajlar panelde gösterilmeye devam etmektedir ancak çöz butonu ve ikon yanıp sönmeye durumu olmamaktadır.

D.2.1.4 Performans Paneli

Performans paneli, sistemdeki araçlara ait enerji, menzil ve sipariş durum bilgilerini özetler. Ekranın sol alt köşesinde konumlandırılan bu panelde, kullanıcı rota ve sipariş detaylarına erişebileceğii iki ayrı butona sahiptir.

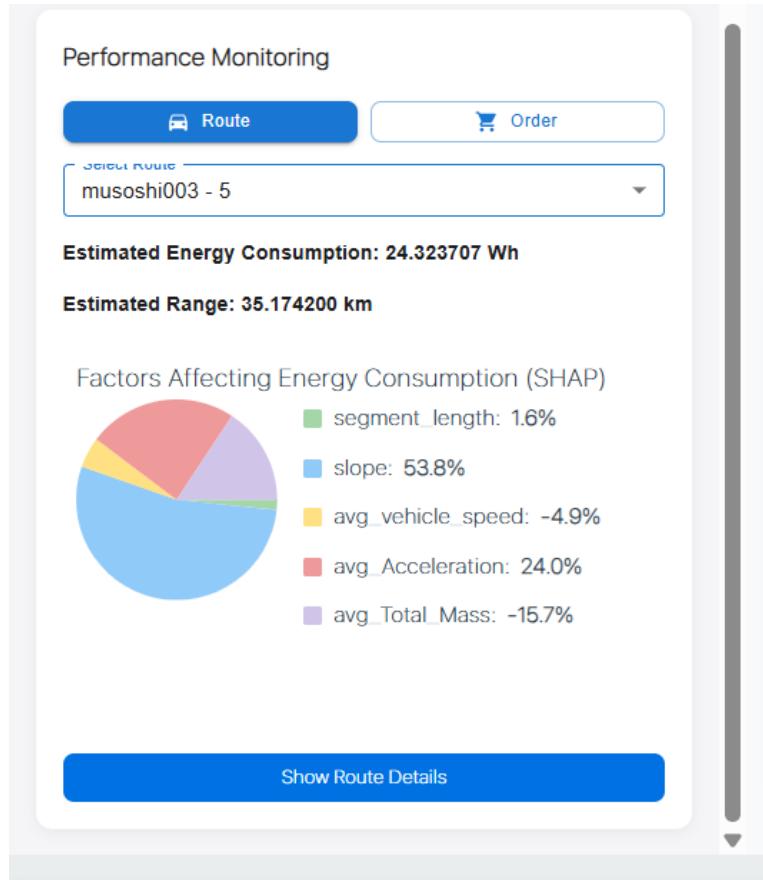


Figure 43: Performans paneli

Rota seçildiğinde, ilgili rotaya ait tahmini enerji tüketimi, kalan menzil ve ML modelinden alınan analiz sonuçları gösterilir. Ayrıca, enerji tüketimine etki eden faktörlerin yüzde dağılımı bir pie chart grafik ile sunulur. Bu grafik, sistemin enerji tüketim davranışını analiz etme açısından oldukça önemlidir. Bu ekranın araç rota seçme kısmı bulunur. Bu kısımda haritalardaki mevcut araçlar ve rotaları listelenir. Seçilen rota ile ilgili olarak enerji tüketim, menzil ve faktör analizi gösterilir. Bu analiz araçların son 100 meteelik verilerini analizlediği için her 10 saniyede bir güncellenen sonuçları gösterir.

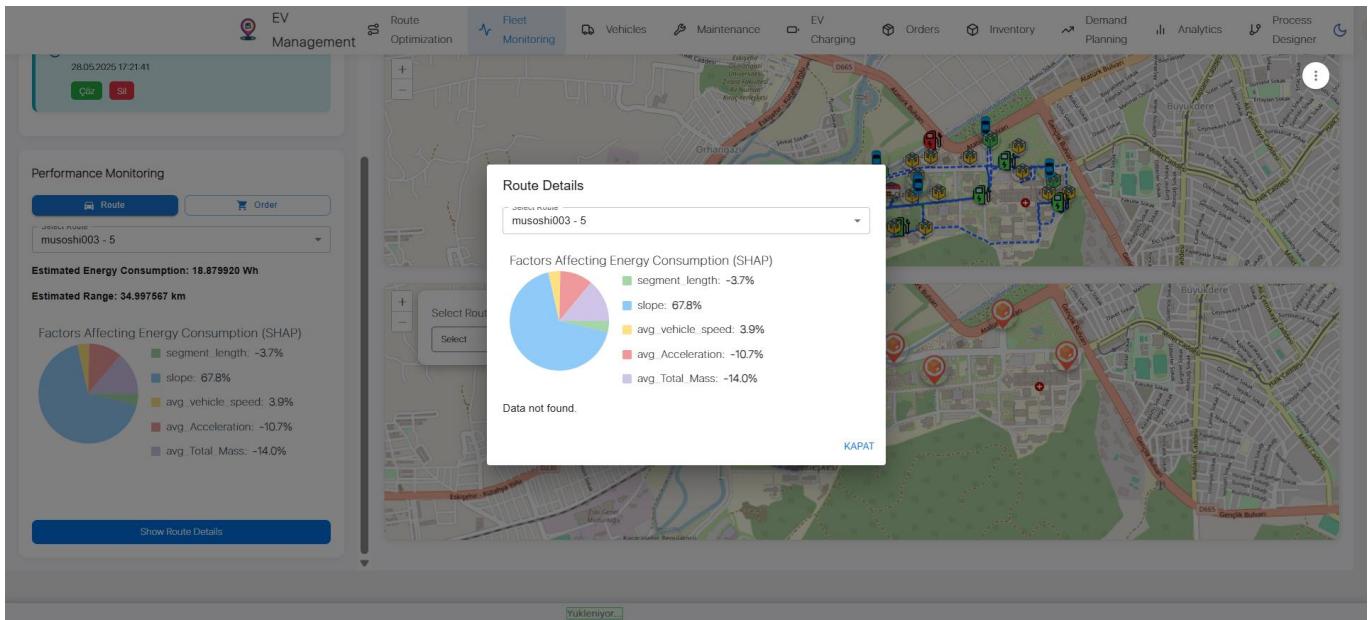


Figure 44: Performans detay

Rota kısmının detay butonuna tıklanıldığında burada panelde görüntülenen faktör grafiği popup olarak ekranın ortasında görüntülenir. Burada araç seçimi ve incelemeleri için yine bir alan sunulmuştur.

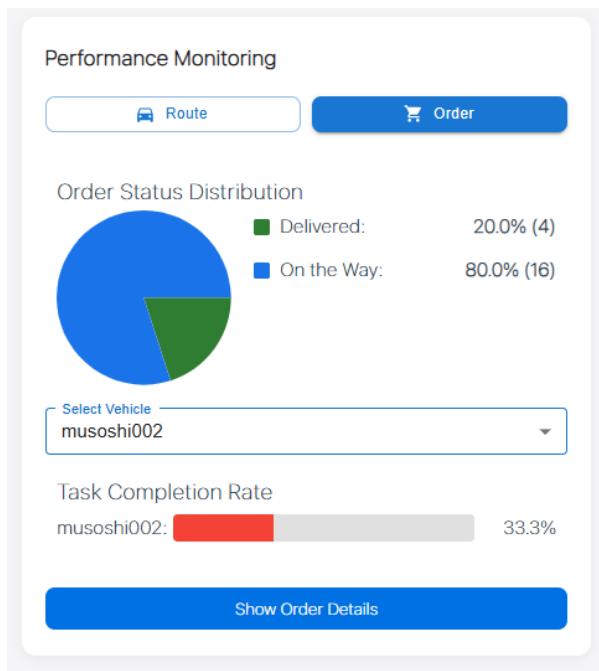


Figure 45: Sipariş panel

Sipariş seçildiğinde ise sistemdeki tüm siparişlerin durumu bir pie chart ile görselleştirilir. Sipariş durumları şu şekilde kodlanmıştır:

- Requested (Turuncu)
- On the Way (Mavi)
- Delivered (Yeşil)
- Cancelled (Kırmızı)

İlgili koda ait durumlar var ise panelde miktar ve yüzdeleri ile gösterilmektedir. Ayrıca pie chart ile de sunulmaktadır. Araçların siparişlerini tamamlama, görev tamamlama durumları da burada yer almaktadır. Seçilen aracın görev tamamlama yüzdesi gösterilmektedir.

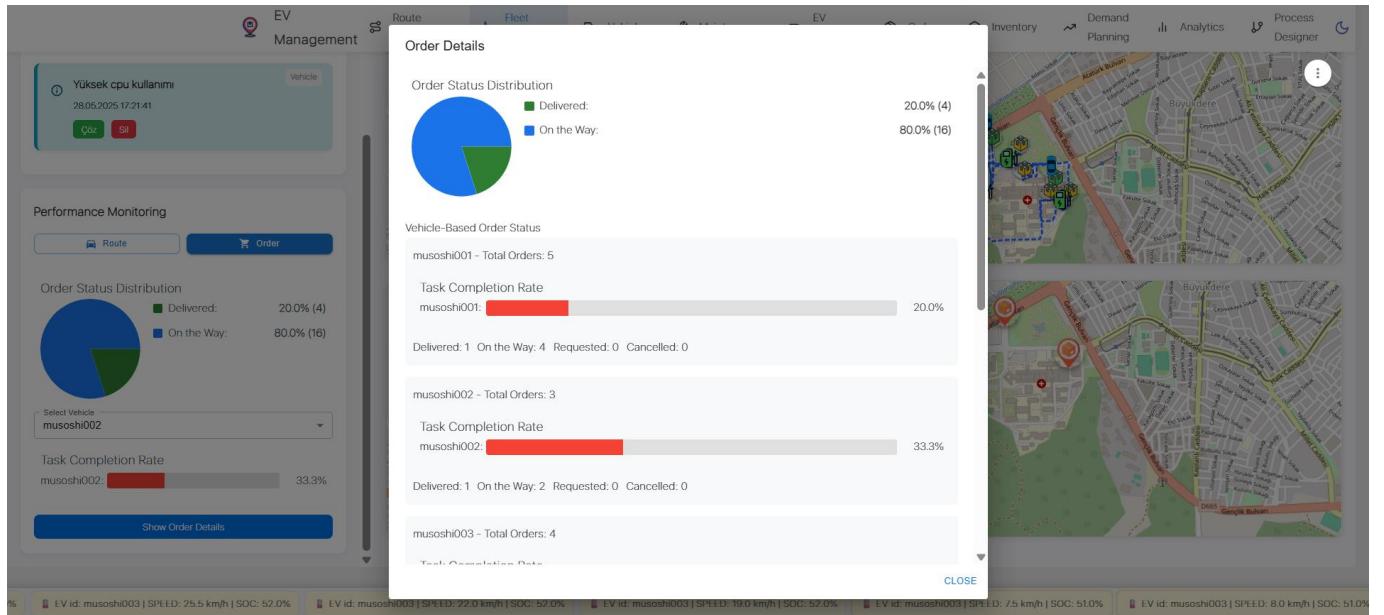


Figure 46: Sipariş detay

Burada sipariş detay butonuna tıklandığında sunodaki siparişlerin güncel durumları ile ilgili bir popup açılmaktadır. Yine pie chart burada da bulunmaktadır fakat bu kez tüm araçların görev tamamlama oranları bar chart ile gösterilmektedir. Her araç için ayrı ayrı siparişlerinin durum sayıları da burada yer almaktadır. Bu detay kısmı ile yöneticiler, filonun genel performansını hızlıca analiz edebilmektedir.

D.2.2 Performans İzleme

D.2.2.1 Genel Yapı

Performans izleme ekranı 2 kısımdan oluşmaktadır. Bunlar ondört raporun bulunduğu akordiyon yapısı ve grafiklerin farklı zaman dilimleri içerisinde görselleştirilmesini sağlayan tarih filtreleme alanıdır. Akordiyon yapısı içerisinde elektrikli araçlarda filo yönetim sistemi için gerekli olan on dört adet rapor ve rapor içeriği için üretilmiş

yaklaşık elli adet tablo, grafik ve stat bulunmaktadır. Akordiyon yapısı karmaşıklığı azaltmak ve daha kompakt bir yapı kurmak amacıyla kullanılmaktadır.

The screenshot shows a navigation bar at the top with links for EV Management, Route Optimization, Fleet Monitoring, Vehicles, Maintenance, EV Charging, Orders, Inventory, Demand Planning, Analytics, and Process Designer. Below the navigation bar is a date range selector with buttons for YESTERDAY, TODAY, LAST 7 DAYS, LAST MONTH, THIS MONTH, and THIS YEAR. A detailed list of reports follows, each with a collapse arrow:

- Vehicle Performance Report
- Charging Station Usage Report
- Fleet Operation Performance Report
- Vehicle Maintenance Report
- Carbon Gain Analysis Report
- System Warning Log Report
- Vehicle Comparison Report
- Planned vs Actual Route Report
- Algorithm Performance Report
- General Order Report
- System Performance Report
- Driver Performance Report
- Driver Success Report
- Customer Based Order Report

Figure 47: Performans izleme raporlar

Tarih filtreleme alanında, kullanıcının sık kullanacağı tarih aralıkları hazır olarak verilmektedir. Bunlar: gün, bugün, son 7 gün, geçen ay, bu ay ve bu yıl olarak belirlenmiştir.

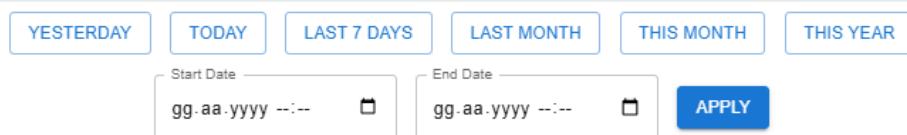


Figure 48: Zaman aralığı seçme

Ayrıca istenen tarih aralığı ve saatı takvim üzerinden seçilebilmektedir. Start Date ve End Date seçildikten sonra Apply butonuna basmak panellerin url yapısını güncelleyerek grafiklerin güncellenmesini sağlar.

The screenshot shows a date range selector with buttons for YESTERDAY, TODAY, LAST 7 DAYS, LAST MONTH, THIS MONTH, and THIS YEAR. Below these are two date input fields with placeholder 'gg.aa.yyyy --::--' and a calendar icon. The calendar for May 2025 is displayed, showing days from Monday to Sunday. The current date is highlighted in blue. To the right of the calendar is an 'APPLY' button.

Figure 50: başlangıç tarihi seçme

Figure 49: Bitiş tarihi seçme

Grafikler ve tablolar performans verilerinin izlenebilirliğini artırmak amacıyla verilerdeki değer değişikliklerine göre her grafik ve tablo için, içerdikleri verilere uygun mantıklı renk geçişleri kullanılarak fark edilebilirlik artırılmıştır.



Figure 52: Performans tablolar

Figure 51: Performans grafikler

Her panelde, ilgili grafiğin amacını açıklayan bir bilgi alanı bulunmaktadır. Bu açıklamalar, kullanıcının grafik üzerindeki verileri daha iyi anlamasını ve yorumlamasını kolaylaştırmaktadır.

The table displays the next maintenance schedule for various vehicles. It includes columns for vehicle ID, maintenance type, scheduled date, and days remaining until the next maintenance. The maintenance types are color-coded: blue for periodic maintenance and purple for tire and suspension checks.

| Next Maintenance | | Scheduled Date | Days Remaining |
|--|---------------------------|----------------|----------------|
| Displays the upcoming maintenance schedule for all vehicles. Color-coded by urgency: Red for within 7 days, Yellow for within 30 days, Green otherwise. | | | |
| veh_2 | Periodic maintenance | 2025-09-28 | 123 |
| veh_3 | Periodic maintenance | 2025-10-12 | 137 |
| veh_4 | Periodic maintenance | 2025-10-19 | 144 |
| veh_0 | Tire and suspension check | 2025-10-26 | 151 |
| veh_1 | Tire and suspension check | 2026-04-04 | 311 |
| veh_2 | Tire and suspension check | 2026-04-11 | 318 |
| veh_3 | Tire and suspension check | 2026-04-18 | 325 |
| veh_4 | Tire and suspension check | 2026-04-25 | 332 |
| | | 2026-05-02 | 339 |

Figure 53: Grafik bilgi

D.2.2.2 Raporlar

Şarj İstasyonları Kullanım Raporu

Bu rapor şarj istasyonlarının filo yönetimindeki kullanımını görselleştirmek ve analizlemek amacıyla kullanılmaktadır. Şarj İstasyonlarının Kullanım Sıklığı, Şarj Süresi Analizi, Yoğun Saatlerin Tespiti, Şarj İstasyonlarının Durumu ve Bakımda Olan Şarj İstasyonları olmak üzere 5 adet panel bulunmaktadır.

- Şarj İstasyonların Kullanım Sıklığı Grafiği: Bu pasta grafiği şarj istasyonlarının kullanım sıklığını, kullanım sayısına göre görselleştirmektedir. İstasyonların yoğunluğunu ölçmeye ve sık kullanılan şarj istasyonlarını belirlemeye yardımcı olur.
- Şarj Süresi Analizi Grafiği: Kullanılan şarj istasyonlarının kaç defa kullanıldığını ve oturum sürelerinin ortalama, min ve max şarj sürelerini dakika cinsinden görselleştirmektedir. Böylece şarj istasyonunda

harcanan süreyi ölçülebilmektedir. Her istasyonun aktifliğini ve verimli kullanıldığını değerlendirmeyi sağlamaktadır.

- Yoğun Saatlerin Tespiti Grafiği: Şarj istasyonlarında gerçekleşen oturumların saatlik bazda görselleştirilmesine olanak sağlar. Dinamik renk eşikleri ile yoğun şarj saatlerini vurgulamayı kolaylaştırır.
- Şarj İstasyonlarının Durumu Grafiği: Tüm şarj istasyonlarının çalışma durumunu gösterir. Aktif ve bakım altında olan şarj istasyonlarının sayısını belirtir.
- Bakımda Olan Şarj İstasyonları Grafiği: Bakımda olan şarj istasyonlarını kimlikleri, adları ve bakım zamanları ile listeler.

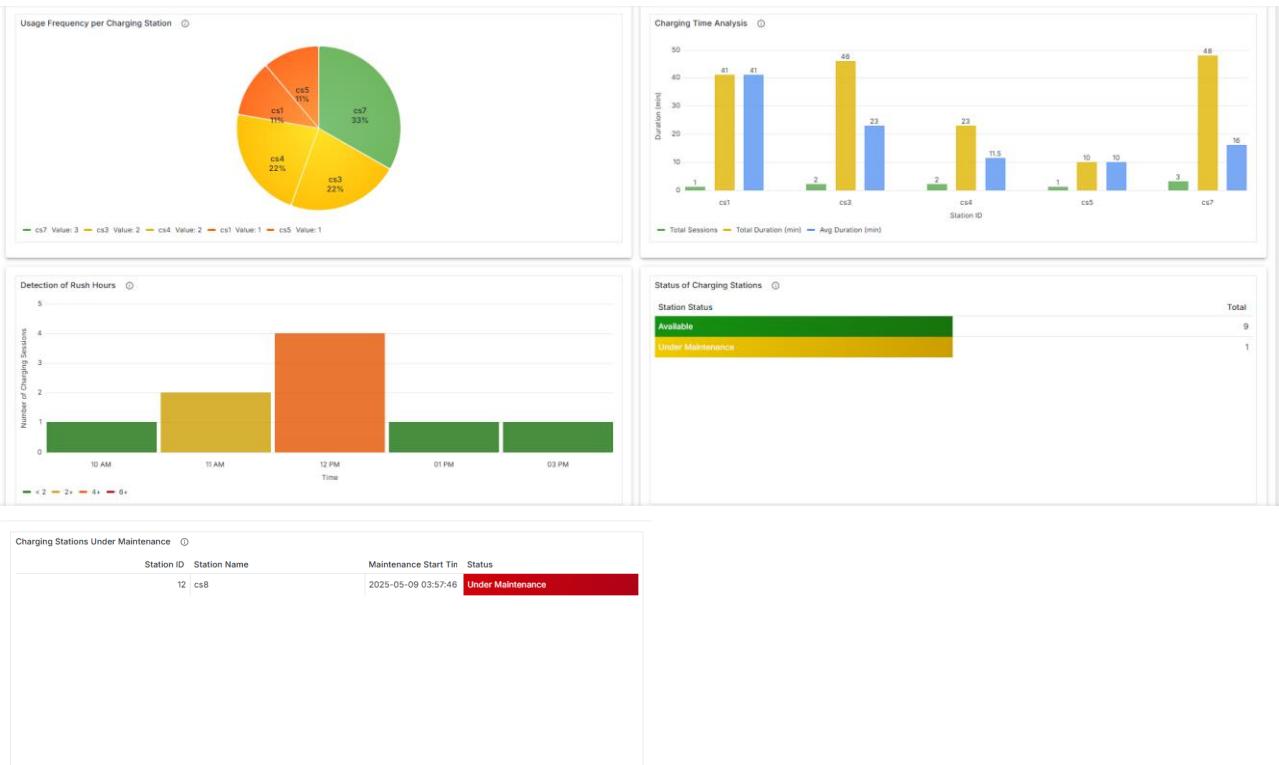


Figure 54: Şarj İstasyonu Rapor

Algoritma Performans Raporu

Bu rapor filo yönetim sistemlerindeki rota oluşturmak amacıyla kullanılan algoritmaları görselleştirmek ve analizlemek amacıyla oluşturulmuştur. Optimizasyon Algoritmalarının Dağılımı, Rota Optimizasyon Özeti ve Optimizasyon Algoritmasına Göre Rota Uygulama Doğruluğu olmak üzere 3 panel bulunmaktadır.

- Optimizasyon Algoritmalarının Dağılımı Grafiği: Bu pasta grafiğinde seçilen zaman aralığındaki kullanılan algoritmaların dağılımı görselleştirilmektedir..
- Rota Optimizasyon Özeti Grafiği: Amaç foksiyonu olan minimum süre, minimum mesafe ve minimum enerji amaçları doğrultusunda optimize edilen rotaları göstermektedir. Her rota için optimizasyon yöntemini, algoritmayı toplam rota mesafesini ve tahmini enerji tüketimini göstermektedir.
- Optimizasyon Algoritmasına Göre Rota Uygulama Doğruluğu (%) Grafiği: Bu tablonun amacı rota için planlanan mesafe ile gerçek kat edilen mesafeyi oranlayarak doğruluk oranını renk kodlaması ile göstermektedir.

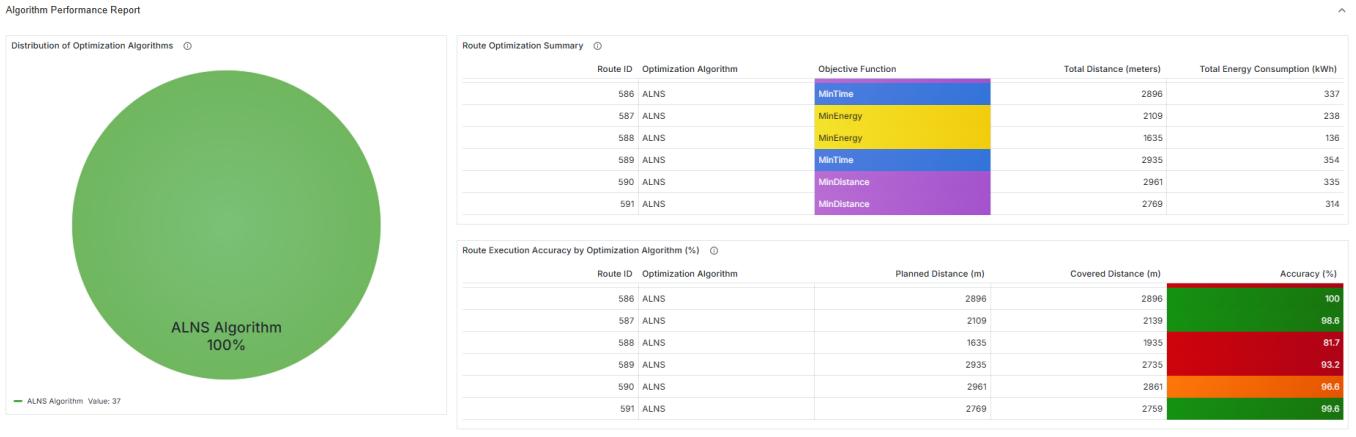


Figure 55: Algoritma Performans Raporu

Karbon Kazancı Analiz Raporu

Bu rapor filo yönetim sistemlerinde elektrikli araç kullanımının, rota bazlı karbon salınımı üzerindeki etkisini analiz etmek amacıyla oluşturulmuştur. Bu rapor Rota Başına Karbon Tasarrufu tablosundan oluşmaktadır.

- Rota Başına Karbon Tasarrufu Tablosu:** Bu tablonun amacı, fosil yakıtlı araçlar ile elektrikli araçların karbon emisyonlarını karşılaştırarak her rota için tahmini karbon tasarrufunu göstermektir. Hesaplamalar Türkiye'nin enerji karışımına göre belirlenen emisyon katsayıları kullanılarak yapılmaktadır. Emisyon katsayıları fosil yakıtlı araçlar için 0.1722 kg/km, elektrikli araçlar için ise 0.0891 kg/km olarak alınmaktadır. "Karbon Tasarrufu (kg)" sütunu, planlanan mesafe üzerinden elde edilen emisyon farkını ifade etmektedir. Karbon emisyonunun kazanımları üç renk aralığında renklendirerek daha etkin bir görselleştirme yapılmaktadır.

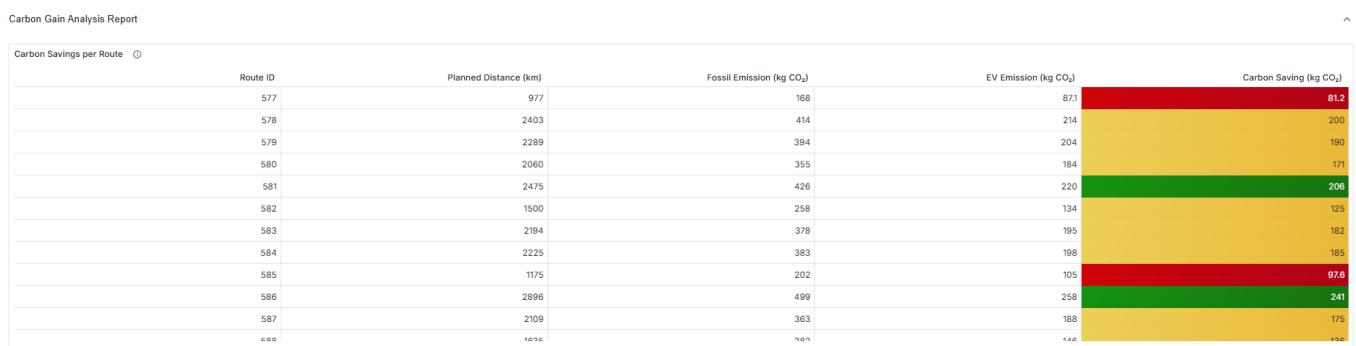


Figure 56: Karbon Kazancı Analiz Raporu

Müşteri Bazlı Rapor

Bu raporun amacı, müşterilerin bölgesel dağılımlarını, siparişlerini, yorumlarını ve skorlarını detaylı bir şekilde analiz etmektir. Analizler sistemin geliştirilmesine olanak sağlayarak hizmet performansını artırmaya yardımcı olabilmektedir. Rapor Sipariş Sıklığı, Müşteri Yorumları, Bölgeye Göre Sipariş Sayısı, Müşteri Bazında Toplam Sipariş Sayısı ve Müşteri Bazlı Ortalama Puan Sorgulaması olmak üzere beş panelden oluşmaktadır.

- Sipariş Sıklığı Grafiği:** Bu çizgi grafiği her müşterinin verdiği sipariş sayısını takip etmek amacıyla kullanılmaktadır. Aktif müşterileri ve yoğun sipariş zamanlarının belirlenmesine yardımcı olmaktadır.

- Müşteri Yorumları Tablosu:** Bu tablo müşterilerin geri bildirimlerinden oluşmaktadır. Müşterinin yorumlarını ve puanlarını tablo formatında görselleştir bu sayede memnuniyet düzeyini yansıtır.
- Bölgeye göre sipariş sayısı Grafiği:** Bu sütun grafiği seçilen zaman aralığındaki bölgelere göre gruplanmış toplam sipariş sayısını görselleştir. Böylece bölgelerin sipariş hacimlerini analiz edilmesine olanak sağlar.
- Müşteri Bazında Toplam Sipariş Sayısı Grafiği:** Bu sütun grafiği seçilen zaman aralığındaki müşterinin vermiş olduğu toplam sipariş sayısını göstermektedir. Bu grafik en çok sipariş veren müşterilerin tespit edilmesine olanak sağlamakla birlikte müşteri devamlılığının takibinde önemli bir rol oynamaktadır.
- Müşteri Bazlı Puan Sorulaması Grafiği:** Bu sütun grafiği seçilen zaman aralığındaki müşterinin vermiş olduğu memnuniyet puanlarını göstermektedir. Bu sayede müşterilerin geri bildirimleri etkin bir şekilde görselleştirilmektedir.

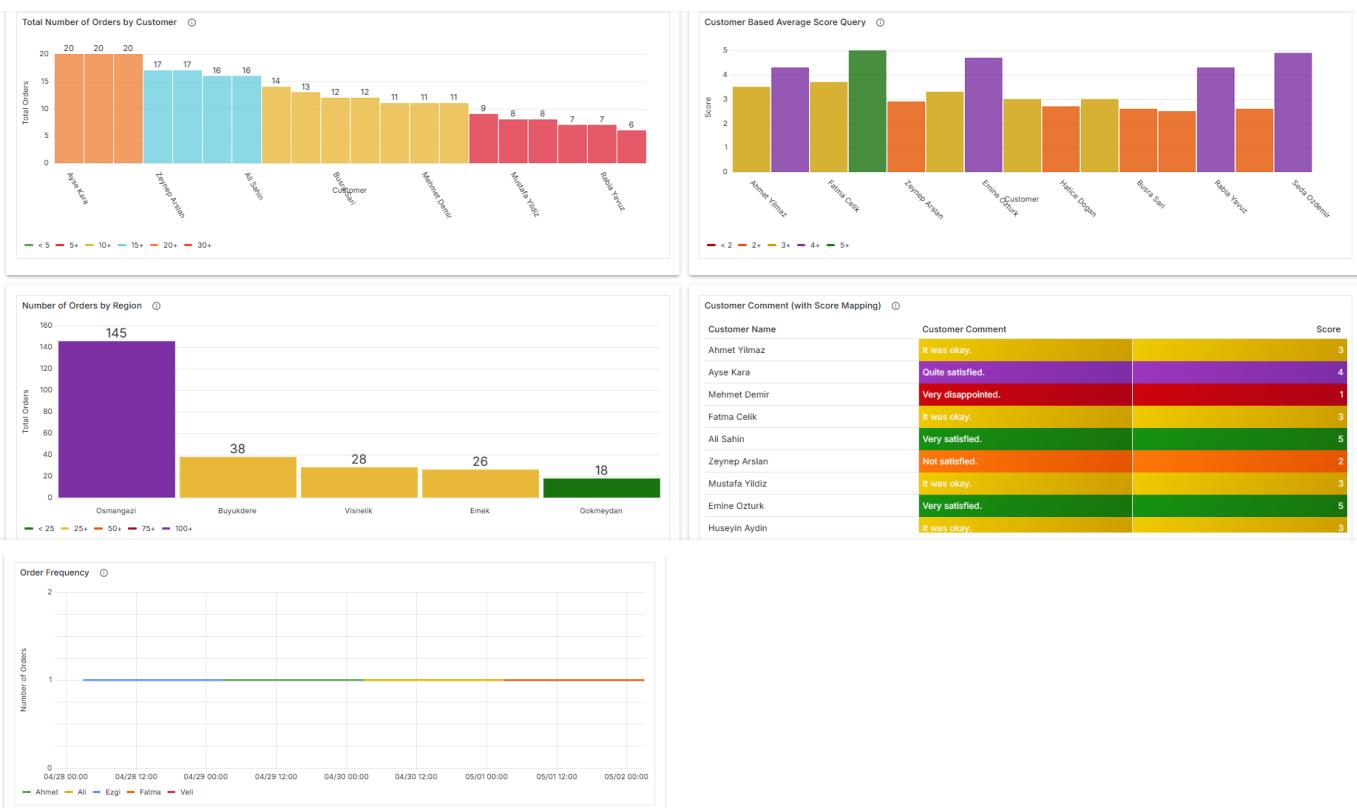


Figure 57: Müşteri Bazlı Rapor

Sürücü Performans Raporu

Bu rapor, sürücülerin operasyonel performanslarının değerlendirilmesini amaçlamaktadır. Teslimat sayısı, kat edilen mesafe ve enerji tüketimi gibi temel metrikler üzerinden sürücülere ait veriler renk geçişleri ile analiz edilebilmektedir. Ayrıca sürücü performans skoru ile sürücülerin performans takibi sağlanabilmektedir.

- Toplam Sürücü Sayısı istatistiği Grafiği:** Seçilen zaman aralığında sistemde kaydedilmiş toplam benzersiz sürücü sayısını göstermektedir.
- Toplam Teslimat Sayısı istatistiği Grafiği:** Tüm sürücüler tarafından seçilen zaman dilimi içinde tamamlanan toplam teslimat sayısını göstermektedir.
- Toplam Enerji Tüketimi (kWh) istatistiği Grafiği:** Seçilen zaman aralığında tüm sürücüler tarafından kaydedilen toplam enerji tüketimini kWh cinsinden göstermektedir.

- Toplam Çalışma Süresi (saat) istatistiği Grafiği: Tüm sürücüler için kayıt altına alınan toplam çalışma süresini göstermektedir.
- Toplam Kat Edilen Mesafe (km) istatistiği Grafiği: Seçilen zaman aralığında tüm sürücüler tarafından kat edilen toplam mesafeyi kilometre (km) cinsinden göstermektedir.
- Sürücü Bazlı Performans Skoru Grafiği: Bu sütun grafiği seçilen zaman aralığında her sürücünün 100 üzerinden performans skorunu göstermektedir. Bu grafik, sürücülerin bireysel performanslarını karşılaştırılmalı olarak görselleştirilmesini sağlamaktadır.
- Sürücü Ortalama Hızları (km/s) Grafiği: Bu sütun grafiği her sürücünün seçilen zaman aralığında kaydedilen ortalama hızını (km/saat) gösterir. Grafikte kullanılan renk geçişleri, sürücü davranışlarının görsel olarak ayırt edilebilirliğini artırarak analiz sürecini daha etkin hale getirmektedir.

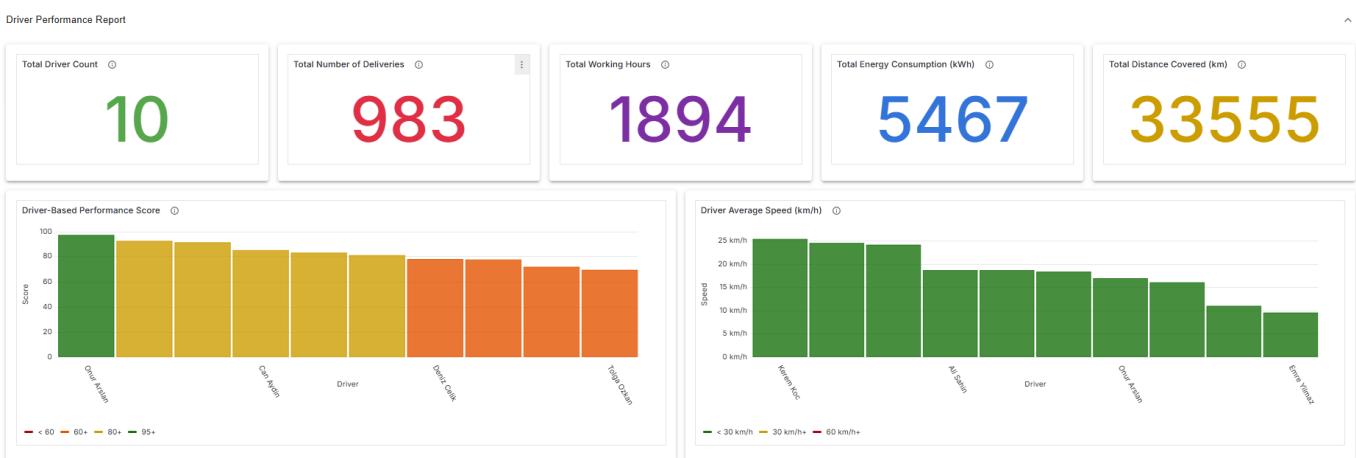


Figure 58: Sürücü Performans Raporu

Araç Bakım Raporu

Bu rapor, filo yönetimindeki elektrikli araçların bakım süreçlerinin verimli bir şekilde takip edilmesi amacıyla oluşturulmuştur. Araçların bakım durumu hakkında sistematik bilgi sağlayarak süreklişılık sağlar ve olası kritik durumların önlenmesinde etkilidir. Bu rapor; Sonraki Bakım, Son Bakım Tarihi, Yaklaşan Bakımlar – 1 Ay ve Gecikmiş Bakımlar olmak üzere dört tablodan oluşmaktadır.

- **Sonraki Bakım Grafiği:** Tüm araçların bakım tarihlerini göstermektedir. Bakımların aciliyet durumlarına göre renk kodlaması bulunmaktadır. Böylece bakım tarihleri ciddi durumlara olanak vermeyecek şekilde takip edilebilmektedir.
- **Son Bakım Tarihi Grafiği:** Her aracın bakım geçmişi bu tabloda görselleştirilir.
- **Yaklaşan Bakımlar 1-Ay Grafiği:** Bu tablo filodaki araçlar arasında bir aylık süre içinde bakım zamanı yaklaşanların takibinin etkin bir şekilde yapmasına olanak sağlamaktadır, böylece maliyetlerin planlanması imkan sunmaktadır.
- **Gecikmiş Bakımlar Grafiği:** Bu tabloda bakım zamanı gecikmiş araçlar listelenmektedir. Araçların gecikme süreleri bakım bilgileri ile birlikte tutularak öncelik sıralaması oluşturulmasına olanak sağlamaktadır.

| Next Maintenance | | | | | Last Maintenance Date | | | | |
|------------------|---------------------------|----------------|----------------|--|-----------------------|---------------------------|-----------------------|--|--|
| Vehicle ID | Maintenance Type | Scheduled Date | Days Remaining | | Vehicle ID | Maintenance Type | Last Maintenance Date | | |
| veh_1 | Periodic maintenance | 2025-06-25 | 28 | | veh_4 | Periodic maintenance | 2025-04-29 | | |
| veh_0 | Periodic maintenance | 2025-09-28 | 123 | | veh_3 | Tire and suspension check | 2025-04-25 | | |
| veh_2 | Periodic maintenance | 2025-10-12 | 137 | | veh_3 | Periodic maintenance | 2025-04-22 | | |
| veh_3 | Periodic maintenance | 2025-10-19 | 144 | | veh_2 | Tire and suspension check | 2025-04-18 | | |
| veh_4 | Periodic maintenance | 2025-10-26 | 151 | | veh_1 | Periodic maintenance | 2025-04-15 | | |
| veh_0 | Tire and suspension check | 2026-04-04 | 311 | | veh_1 | Tire and suspension check | 2025-04-11 | | |
| veh_1 | Tire and suspension check | 2026-04-11 | 318 | | veh_0 | Tire and suspension check | 2025-04-04 | | |
| veh_2 | Tire and suspension check | 2026-04-18 | 325 | | veh_0 | Periodic maintenance | 2025-04-01 | | |
| veh_3 | Tire and suspension check | 2026-04-25 | 332 | | veh_4 | Tire and suspension check | 2025-03-23 | | |

| Upcoming Maintenance - 1 Month | | | | | Overdue Maintenance | | | | | |
|--------------------------------|----------------------|-----------------------|-----------------------|----------------|---------------------|---------------------------|-----------------------|-----------------------|----------------|--------------|
| Vehicle ID | Maintenance Type | Last Maintenance Date | Next Maintenance Date | Estimated Cost | Vehicle ID | Maintenance Type | Last Maintenance Date | Next Maintenance Date | Estimated Cost | Days Overdue |
| veh_1 | Periodic maintenance | 2024-04-08 | 2025-06-25 | 796 | veh_4 | Tire and suspension check | 2024-10-18 | 2025-04-16 | 1350 | 42 |

Figure 59: Sürücü Performans Raporu

Sistem Uyarı Raporu

Bu rapor, sistemde oluşan hata kayıtlarının türlerini, kaynaklarını, seviyelerini analiz ederek sistem davranışındaki anormal durumların tespit edilmesini ve izlenmesini sağlamaktadır. Böylece müdahale süreçlerinin etkin bir şekilde yürütülmesini sağlamaktadır.

- **Toplam Log Sayısı Grafiği:** Seçilen zaman aralığındaki toplam log sayısının günlük olarak görselleştirilmesini sağlar. Logların gün içindeki sayılarına göre sütunlar renklendirilmektedir. Böylece anomali artışlarını tespit etmek kolaylaşmaktadır.
- **Log Türlerinin Dağılımı Grafiği:** Bu sütun grafiğinde sistemde meydana gelen hata seviyelerinin dağılımı renk geçişleri ile kullanıcıya etkili ve açık bir şekilde iletilmektedir.
- **Toplam Log Grafiği:** Bu istatistik panelinde seçilen aralıktaki sistemde üretilen toplam log sayısı görselleştirilmektedir.
- **Şiddetine Göre Saatlik Log Dağılımı Grafiği:** Bu tabloda seçilen zaman aralığındaki saat bazında oluşan logların, şiddet dereceleri ve kaynakları farklı koşullar için renklendirilerek açık bir şekilde görselleştirilmektedir.

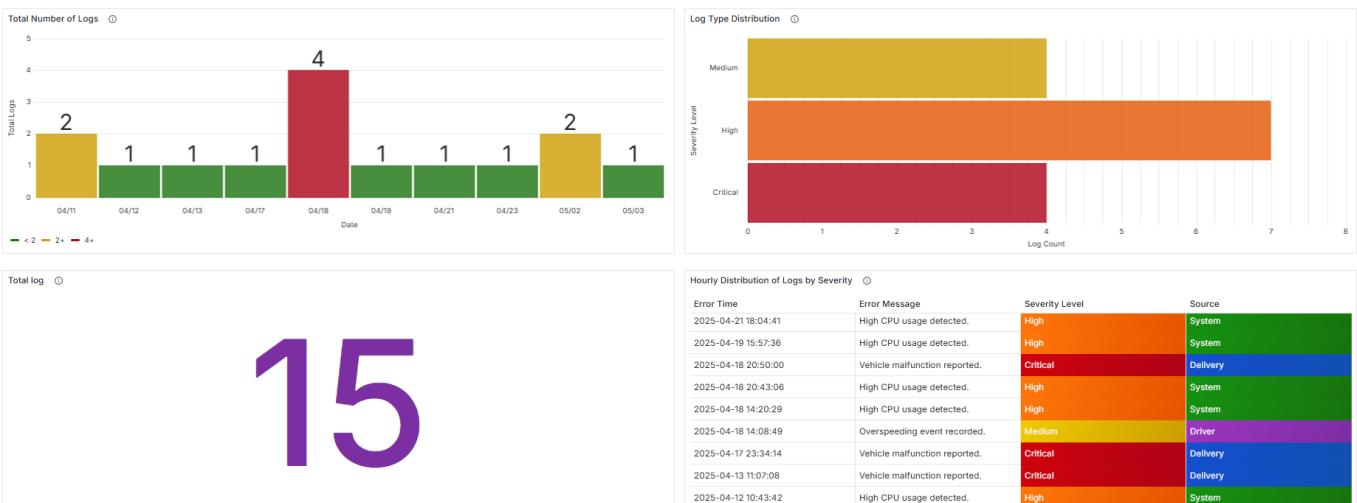


Figure 60: Sistem Uyarı Raporu

Planlanan ve Gerçekleşen Rota Raporu

Bu rapor, planlanan ve gerçekleşen rotaların arasındaki farkların analiz edilmesini, süreçlerin doğruluğunu ve verimliliğini ölçmeyi amaçlamaktadır.

- Ortalama Zaman ve Mesafe Bilgisi Grafiği: Planlanan ve tamamlanan rotaların rotayı tamamlama süreleri ve kat ettikleri mesafe karşılaştırılmaktadır. Rotalardaki sapmaların ve rota doğruluğunun takip edilmesi açısından önemli bir tablodur.
- Planlanan ve Tamamlanan Rota Sayıları Grafiği: Bu grafik seçili zaman aralığında planlanan ve gerçekleşen rota sayılarını karşılaştırmaktadır. Rotalama sürecinin başarı düzeyinin gözlemlenmesine imkân tanımaktadır.

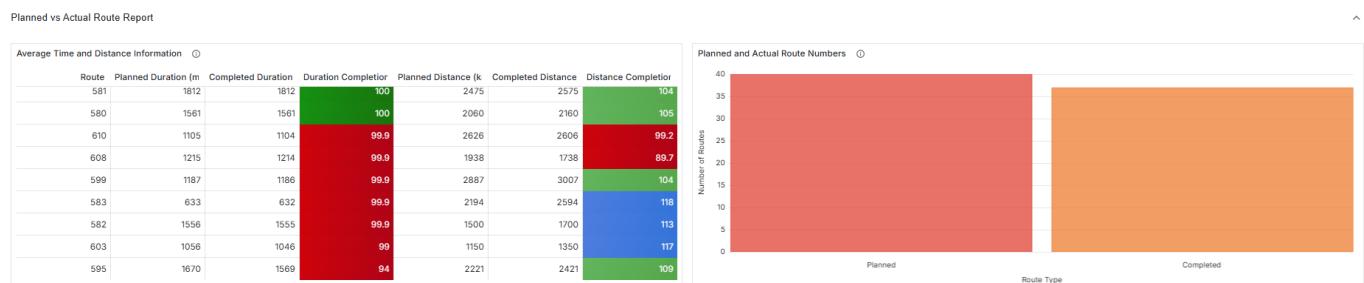


Figure 61: Planlanan ve Gerçekleşen Rota Raporu

Sürücü Başarı Raporu

Bu rapor, filo yönetimindeki her sürücülerin teslimat performanslarının hem bireysel hemde diğer sürücüler ile detaylı ve objektif bir şekilde değerlendirilmesini sağlar. Böylece performans iyileştirme süreçlerinin desteklenmesine fayda sağlamaktadır.

- Toplam Teslimat Sayısı Grafiği: Her sürücünün gerçekleştirdiği teslimat sayısını görselleştirmektedir. Sürücü performanslarının sütun grafik olarak renk geçişleri ile gözlemlenmesine olanak sağlar.
- Zamanında Teslimatlar Grafiği: Sürücülerin planlanan zaman penceresinde tamamladıkları teslimat sayısını gösteriri, sürücünün performansının izlenmesinde oldukça kritiktir.
- Performans Skoru Grafiği: Her sürücünün performans skorlarını görselleştirmektedir, sürücülerin karşılaştırılmasını ve değerlendirilmesini kolaylaştırmaktadır.
- Teslimat Gecikme Oranı Grafiği: Her sürücünün geç teslim ettiği sipariş sayısının toplam sipariş sayısına oranını vermektedir.

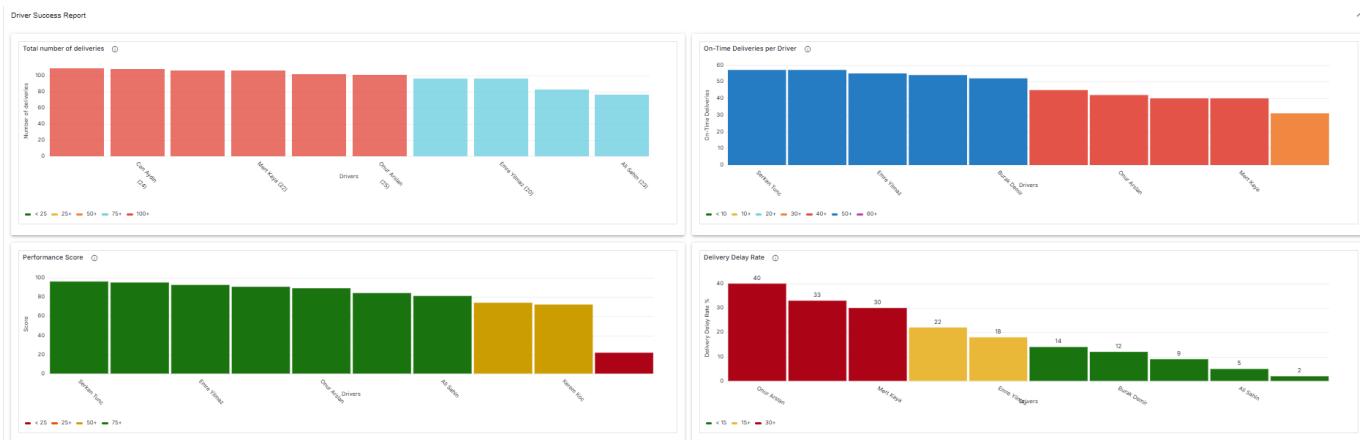


Figure 62: Sürücü Başarı Raporu

Sistem Performans Raporu

Bu rapor, sistem performansında kritik faktörler olan sunucu yanıt süresi, bellek kullanımı ve CPU kullanımını detaylı olarak takip ederek, olası sorunların tespit edilmesini amaçlar.

- Sunucu Yanıt Süresi (ms) Grafiği:** Çizgi grafik şeklinde görselleştirilmektedir, sunucu yanıt sürelerinin görselleştirilmesini sağlar böylece ağ sorunları veya aşırı yüklenmeye dair ipucu verebilmektedir.
- Bellek Kullanımı Grafiği:** Sistem belleğinin performansını izlemek ve aşırı yük durumlarını tespit edebilmek amacıyla RAM kullanım yüzdesini görselleştirmektedir.
- CPU Kullanımı Grafiği:** Seçilen zaman dilimine göre CPU kullanım değerlerini görselleştirir. Kullanım oranları renk kodları ile açık bir şekilde tablo formatında sunulmuştur.

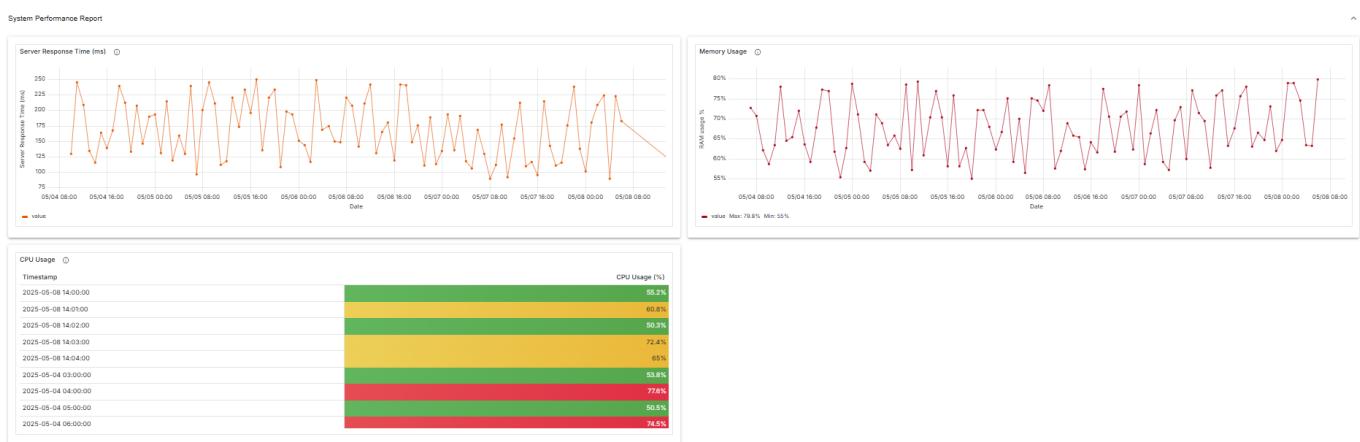


Figure 63: Sistem Performans Raporu

Genel Sipariş Raporu

Bu rapor, filo yönetim sipariş sürecindeki performansın izlenmesini ve değerlendirilmesini renk geçişleri ile anlaşılır bir şekilde görselleştirmeyi amaçlamaktadır. Sipariş durum dağılımı, teslimat süreleri değerlendirilerek sipariş süreçlerinin optimize edilmesini ve hizmet kalitesinin artırılmasına yardımcı olmaktadır.

- Sipariş Durumu Dağılımı Grafiği:** Seçilen zaman aralığındaki sipariş durumlarının dağılımını gösterir. Pasta grafiğini oluşturan değerler Teslim Edilen, İptal Edilen, Devam Eden ve Beklemede olan sipariş durumları ile

bunların sayıları ve yüzdeleri olarak belirlenmiştir. Bu grafiğin temel amacı sipariş verimliliğinin etkin bir şekilde takip edilmesidir.

- **Toplam Sipariş Sayısı Grafiği:** Seçilen zaman aralığında verilen toplam sipariş sayısını gösterir. Sipariş sürecini takip edilmesinde önemlidir.
- **İptal Edilen Sipariş Sayısı Grafiği:** Seçilen zaman aralığında iptal edilen sipariş sayısını gösterir. Operasyonel sorunların tespit edilmesinde ve müşteri memnuniyeti açısından önemlidir.
- **Teslim Edilen Sipariş Sayısı (İstatistik Paneli);** seçilen zaman aralığında teslim edilen sipariş sayısını gösterir. Teslimat performansını takip edebilmek açısından önemlidir.
- **Ortalama Teslim Süresi (Dakika) Grafiği:** Seçilen zaman aralığında teslim edilen siparişlerin ortalama teslim süresini dakika cinsinden görselleştirmektedir.
- **Günlük Teslim Süresi Karşılaştırması Grafiği:** Seçilen zaman aralığında teslim edilen siparişlerin ortalama, max ve min teslim süresini dakika cinsinden görselleştirmektedir.
- **Günlük Sipariş Sayısı Grafiği:** Seçilen zaman aralığında her gün verilen sipariş sayısını renk geçişleri ile görselleştirmektedir.

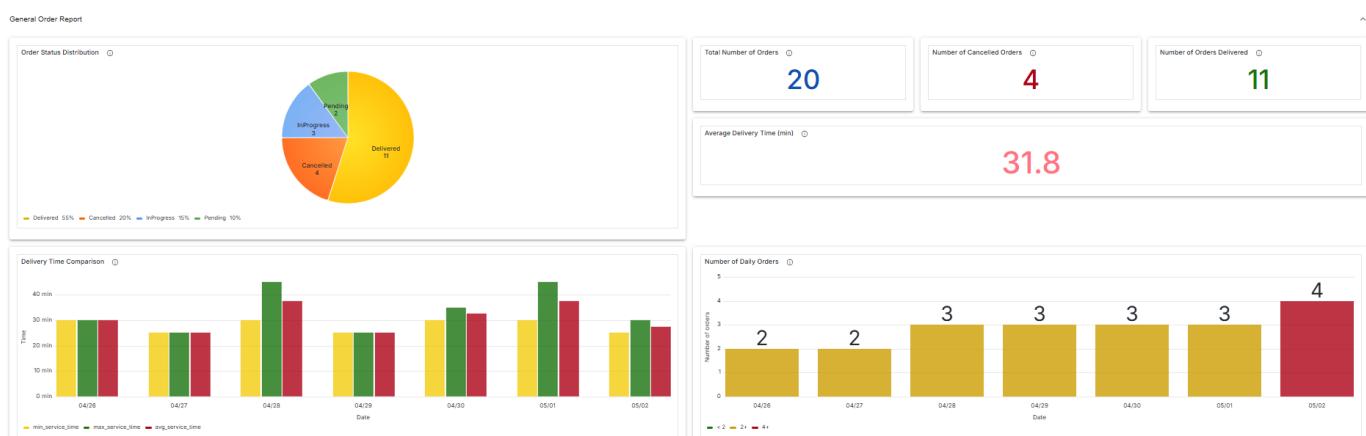


Figure 64: Genel Sipariş Raporu

Araç-Rota Performans Raporu

Bu raporda, filo yönetimindeki araçların rota bazında performansları çok boyutlu olarak değerlendirilmektedir. Araçların hız profilleri, kat ettiğleri mesafeler, karşılaştıkları yol eğimleri, taşıma kapasitesi kullanımı ve batarya doluluk oranları gibi kritik metrikleri analiz edilerek operasyonel iyileştirmeler için karar destek sağlamayı hedefler.

- **Ortalama Hız Grafiği:** Her rota için seçilen zaman içerisindeki ortalama araç hızlarını göstermektedir. Anlık dalgalanmalar ve hız değişimleri gözlemlenebilir.
- **Kat Edilen Yol Grafiği;** Rota bazında araçların kat ettiği toplam mesafeyi göstermektedir.
- **Yol Eğimi Grafiği;** belirlenen yüzde aralıklarına göre rota içindeki eğim profilini görselleştirir.
- **Başlangıç Yükü Grafiği,** Rota bazında operasyonların başlangıcında araçlara ne kadar yük yüklenliğinin değerlendirilmesini sağlar.
- **Ortalama Batarya (SOC) Durumu Grafiği;** Rota bazında araçların ortalama batarya durumlarını göstermektedir.



Figure 65: Araç-Rota Performans Raporu

Filo Operasyon Performans Raporu

Bu raporun amacı; filoda yer alan araçların belirli zaman aralığındaki operasyonel verimliliğini analiz etmektir. Bu analizler temel performans göstergeleri üzerinden kapsamlı bir değerlendirme sunmaktadır. Bu rapor sayesinde araçların kullanım durumları ve performansları değerlendirilip karşılaştırıla bilmektedir.

- **Toplam Tüketilen Enerji Grafiği:** Belirlenen tarih aralığında her bir aracın tükettiği toplam enerji miktarını gösterir. Bu grafik, enerji verimliliği ve operasyon yoğunluğu hakkında bilgi sunar.
- **Araç Bazlı Ortalama Hız Grafiği:** Her aracın belirli bir zaman aralığındaki ortalama hız değerlerini görselleştirir. Araçlar arasında hız performansı karşılaştırması yapılmasına olanak tanır ve hız eğilimlerini tespit etmeyi kolaylaştırır.
- **Araç Bazlı Ortalama Eğim Grafiği:** Her araç için saatlik bazda ortalama yol eğimi (%) verilerini göstermektedir. Bu grafik, farklı rotaların eğim zorluklarını karşılaştırmaya olanak sağlamaktadır.
- **Toplam Kat Edilen Mesafe Grafiği:** Her aracın ilgili zaman aralığında toplam kaç kilometre yol kat ettiğini göstermektedir.
- **Aktif Araç Sayısı Grafiği:** Seçilen zaman aralığında benzersiz araç sayısını gösterir. Filonun sahip olduğu araç sayısına dair özet sunar.

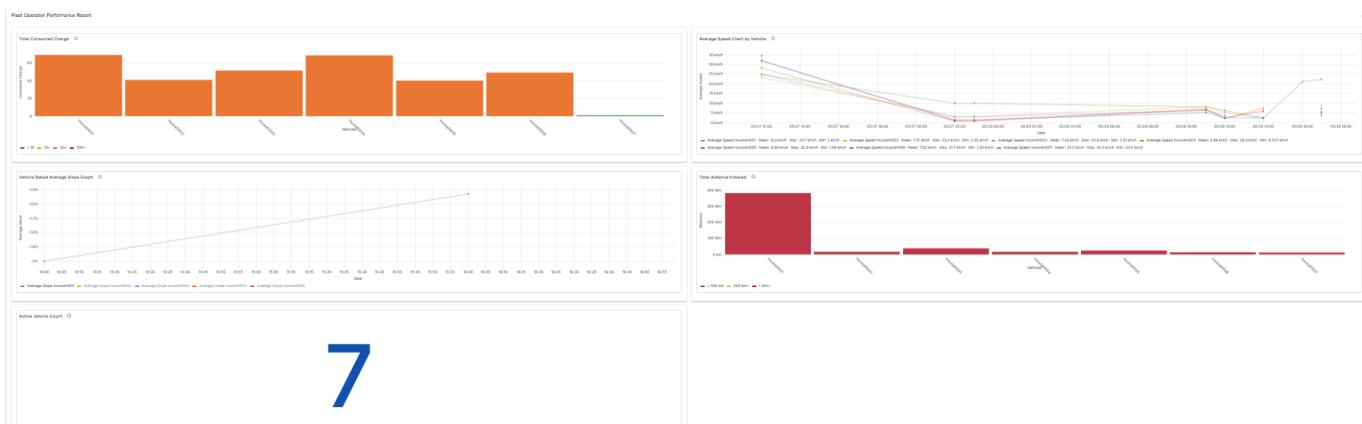


Figure 66: Filo Operasyon Performans Raporu

Araç Karşılaştırma Raporu

Bu rapor, filo içindeki araçların performanslarını karşılaştırmalı olarak analiz etmeyi amaçlamaktadır. Araçlar, ortalama hız, menzil ve batarya durumu gibi metriklere göre sınıflandırılır. Böylece filo verimliliği iyileştirilerek verimlilik arttırılabilir.

- Araç Performans Sınıflandırması Grafiği:** Bu tablo, filo içerisindeki araçların performanslarını karşılaştırmalı olarak analiz etmeyi amaçlamaktadır. Ortalama hız, menzil ve batarya doluluk oranı (SoC) gibi operasyonel metriklere göre araçlar sınıflandırılarak "İyi", "Orta" veya "Zayıf" kategorilerine ayrılmaktadır.
- Araç Performans Karşılaştırması Grafiği:** Bu tabloda araçların ortalama enerji tüketimi, katedilen mesafe ve ortalama hız gibi performans metrikleri karşılaştırmalı biçimde görselleştirilmektedir.

The screenshot displays two tables from a 'Vehicle Comparison Report' interface. The first table, 'Vehicle Performance Classification', lists vehicle IDs, hour intervals, average speeds, average ranges, average SoCs, and performance categories (Poor, Average). The second table, 'Vehicle Performance Comparison', lists vehicle IDs, hour intervals, average energy used, average ranges, average speeds, initial vehicle loads, total distances, and total energy used. Both tables include a color-coded legend for performance categories.

| Vehicle Performance Classification | | | | | |
|------------------------------------|---------------------|----------------------|--------------------|-----------------|----------------------|
| Vehicle ID | Hour Interval | Average Speed (km/h) | Average Range (km) | Average SoC (%) | Performance Category |
| musoshi001 | 2025-05-28 14:00:00 | 2 | 81 | 81.5 | Poor |
| musoshi001 | 2025-05-28 16:00:00 | 20.9 | 81 | 79.2 | Average |
| musoshi001 | 2025-05-28 17:00:00 | 22.3 | 81 | 76.5 | Average |
| musoshi003 | 2025-05-28 19:00:00 | 19.6 | 76.4 | 3 | Poor |

| Vehicle Performance Comparison | | | | | | |
|--------------------------------|---------------------|-------------------------|--------------------|----------------------|---------------------------|---------------------|
| Vehicle ID | Hour Interval | Average Energy Used (%) | Average Range (km) | Average Speed (km/h) | Initial Vehicle Load (kg) | Total Distance (km) |
| musoshi003 | 2025-05-28 19:00:00 | 3 | 78.4 | 19.6 | 878 | 24444 |

Figure 67: Araç Karşılaştırma Raporu

D.3. Gerçeklenen Testler

D.3.1 Birim (Unit) Testleri

Araç Takip Birim Testleri:

```

1 import React from 'react';
2 import '@testing-library/jest-dom';
3 import { render, screen } from '@testing-library/react';
4 import FleetMonitoringMap from '../components/FleetMonitoring/FleetMonitoringMap';
5 import FM_AlertPanel from '../components/FleetMonitoring/FM_AlertPanel';
6
7 beforeEach(() => {
8   global.fetch = jest.fn(() =>
9     Promise.resolve({
10       json: () => Promise.resolve([]),
11     })
12   );
13 });
14
15 test('Harita container'da render ediliyor', () => {
16   render(<FleetMonitoringMap orders={[]} chargingStations={[]} />);
17   expect(screen.getByTestId('fleet-map-container')).toBeInTheDocument();
18 });
19
20 test('Tek bir alert ekranında görünüyor', () => {
21   render(<FM_AlertPanel type="Error" message="Hata!" />);
22   expect(screen.getByText(/Hata!/)}.toBeInTheDocument();
23 });

```



Figure 68: Gerçeklenen Test 1

Araç takip modülü için iki adet birim testi yapılmıştır. İlk test Araç takip ekranı üzerindeki harita için oluşturulan FleetMonitoringMap içindir. FleetMonitoringMap bileşeninin başarıyla DOM'a yerleşip yerleşmediği test edilmiştir. data-testid="fleet-map-container" değeri ile belirlenen container'ın sayfada olup olmadığı kontrol edilmiştir. İkinci testimiz ise FM_AlertPanel bileşeni olan uyarıların görüntülenmesi kısmı için yapılmıştır. FM_AlertPanel bileşeni hata tipi (type="Error") ve "Hata!" mesajı ile render edilmiştir. Ekranda "Hata!" yazısının görünüp görünmediği test edilmiştir.

```
PASS frontend/src/_tests__tracking_birim_tests.test.js (9.02 s)
  ✓ Harita containerı render ediliyor (417 ms)
  ✓ Tek bir alert ekranda görünüyor (17 ms)

  Test Suites: 1 passed, 1 total
  Tests:       2 passed, 2 total
  Snapshots:   0 total
  Time:        10.799 s
  Ran all test suites.

PS C:\app>
```

Figure 69: Test sonuç 1

Test sonuçlarına göre, iki birim test de sorunsuz şekilde geçmiştir. Tüm testlerin çalıştırılması toplamda 10.799 saniye sürmüştür. Bu sonuç verileri, testlerin başarılı ve istikrarlı şekilde çalıştığını göstermektedir.

Performans İzleme Birim Testleri:

```
frontend > src > _tests_ > ⚡ performance_birim_tests.test.js > ⚡ test('Accordion açılıp kapanabiliyor') callback
  1 import React from 'react';
  2 import '@testing-library/jest-dom';
  3 import { render, screen, fireEvent } from '@testing-library/react';
  4 import dayjs from 'dayjs';
  5 import Performans from '../pages/Performance';

  6
  7 describe('Performans Ekranı', () => {
  8   test('Tarih butonları doğru zaman aralığı üretiyor', () => {
  9     render(<Performans />);
 10     const todayBtn = screen.getByRole('button', { name: /today/i });
 11     fireEvent.click(todayBtn);

 12     // Burada state değişimini test etmek için, iframe src'sinde from ve to parametreleri olmalı
 13     const iframes = screen.getAllByTitle(/panel/i);
 14     expect(iframes.length).toBeGreaterThan(0);

 15     // Bugünün başlangıcı ve bitisi
 16     const todayStart = dayjs().startOf('day').valueOf();
 17     const todayEnd = dayjs().endOf('day').valueOf();

 18     // Her iframe'in src'sinde from ve to parametreleri olmalı
 19     iframes.forEach((iframe) => {
 20       expect(iframe.src).toContain(`from=${todayStart}`);
 21       expect(iframe.src).toContain(`to=${todayEnd}`);
 22     });
 23   });
 24 });
 25 });
 26 });
 27 });

~
```



Figure 70: Gerçeklenen Test 2

Performans İzleme modülü için üç adet birim test yapılmıştır. İlk olarak "Today" butonuna basıldığında iframe bileşenlerinin src parametreleri içerisinde from ve to zaman değerlerinin günün başlangıç ve bitiş zamanına göre güncellenip güncellenmediği test edilmiştir. Sonunda iframe kaynaklarında ilgili zaman damgalarının doğru şekilde yer aldığı görülmüş.

```
frontend > src > __tests__ > ⚙️ performance_birim_tests.test.js > ⚡ test('Accordion açılıp kapanabiliyor') callback
  7  describe('Performans Ekranı', () => {
26    });
27  });
28
29 test('Iframe panellerine from ve to değerleri doğru aktarılıyor', () => {
30   render(<Performans />);
31   // Tarih alanlarını doldur
32   const startInput = screen.getByLabelText(/start date/i);
33   const endInput = screen.getByLabelText(/end date/i);
34   const applyBtn = screen.getByRole('button', { name: /apply/i });
35
36   // Örnek tarih aralığı
37   fireEvent.change(startInput, { target: { value: '2024-05-01T00:00' } });
38   fireEvent.change(endInput, { target: { value: '2024-05-10T23:59' } });
39   fireEvent.click(applyBtn);
40
41   const from = dayjs('2024-05-01T00:00').valueOf();
42   const to = dayjs('2024-05-10T23:59').valueOf();
43
44   // Her iframe'in src'sinde from ve to parametreleri olmalı
45   const iframes = screen.getAllByTitle(/panel/i);
46   iframes.forEach((iframe) => {
47     expect(iframe.src).toContain(`from=${from}`);
48     expect(iframe.src).toContain(`to=${to}`);
49   });
50 });
51
52 test('Accordion açılıp kapanabiliyor', () => {
53   render(<Performans />);
54   // İlk accordion başlığını bul
55   const accordionSummary = screen.getByText(/vehicle performance report/i);
56
57   expect(accordionSummary.closest('.Mui-expanded')).toBeTruthy();
58
59   // Kapatmak için tıkla
60   fireEvent.click(accordionSummary);
61   // Accordion detayları kapanmalı
62   expect(accordionSummary.closest('.Mui-expanded')).toBeFalsy();
63
64   // Tekrar açmak için tıkla
65   fireEvent.click(accordionSummary);
66   expect(accordionSummary.closest('.Mui-expanded')).toBeTruthy();
67 });

```



Figure 71: Gerçeklenen Test 3

İkinci testte, kullanıcıların manuel olarak bir tarih aralığı seçip "Apply" butonuna tıkladıklarında, bu tarih aralığına ait zaman damgalarının iframe bileşenlerine doğru bir şekilde aktarıldığı test edilmiştir. Belirlenen örnek tarih aralığında (1 Mayıs 2024 - 10 Mayıs 2024) yapılan kontrol, iframe URL'lerinde from ve to parametrelerinin bu zaman dilimini içerdigini göstermiştir. Bu doğrulamayla birlikte test geçmiştir.

```
frontend > src > __tests__ > ⚙️ performance_birim_tests.test.js > ⚡ test('Accordion açılıp kapanabiliyor') callback
  29 test('Iframe panellerine from ve to değerleri doğru aktarılıyor', () => {
  46   iframes.forEach((iframe) => {
  48     expect(iframe.src).toContain(`to=${to}`);
  49   });
  50 });
  51
  52 test('Accordion açılıp kapanabiliyor', () => {
  53   render(<Performans />);
  54   // İlk accordion başlığını bul
  55   const accordionSummary = screen.getByText(/vehicle performance report/i);
  56   // Accordion detayları ilk başta açık olmalı
  57   expect(accordionSummary.closest('.Mui-expanded')).toBeTruthy();
  58
  59   // Kapatmak için tıkla
  60   fireEvent.click(accordionSummary);
  61   // Accordion detayları kapanmalı
  62   expect(accordionSummary.closest('.Mui-expanded')).toBeFalsy();
  63
  64   // Tekrar açmak için tıkla
  65   fireEvent.click(accordionSummary);
  66   expect(accordionSummary.closest('.Mui-expanded')).toBeTruthy();
  67 });

```



Figure 72: Gerçeklenen Test 4

Üçüncü ve son testte ise kullanıcı etkileşimiyle performans sayfasındaki accordion bileşeninin açılıp kapanıldığı kontrol edilmiştir. Accordion başlığına yapılan tıklamalar sonucunda, içerik bölümünün önce kapandığı sonra tekrar açılabildiği başarıyla gözlemlenmiştir. Bu sayede kullanıcı arayüzünde etkileşimli bileşenlerin beklenildiği gibi çalıştığı doğrulanmıştır.

```
PASS frontend/src/_tests_/performance_birim_tests.test.js (11.146 s)
  ✓ Iframe panellerine from ve to değerleri doğru aktarılıyor (2145 ms)
  ✓ Accordion açılıp kapanabiliyor (1087 ms)
    Performans Ekranı
      ✓ Tarih butonları doğru zaman aralığı üretiyor (2767 ms)

    Test Suites: 1 passed, 1 total
    Tests:       3 passed, 3 total
    Snapshots:  0 total
    Time:        12.56 s
    Ran all test suites.
PS C:\app> █
```

Figure 73: Test sonuç 2

Yapılan bu üç testin tamamı başarıyla geçmiştir. Tek bir test dosyasında (performance_birim_tests.test.js) çalıştırılan bu testler toplamda 12.56 saniyede tamamlanmış ve herhangi bir hata gözlemlenmemiştir.

D.3.2 Entegrasyon Testleri

```
13  test('ML API: Araç verisi gönderildiğinde kalan menzil ve SHAP faktörleri dönüyor', async () => {
14    const input = {
15      avg_vehicle_speed: 35,
16      segment_length: 500,
17      avg_Acceleration: 0.2,
18      avg_Total_Mass: 1800,
19      slope: 1.5,
20      soc: 80,
21      remaining_energy: 12000,
22      nonlinear_energy: 150,
23      segment_count: 10
24    };
25
26    const response = await fetch('http://localhost:5002/predict', {
27      method: 'POST',
28      headers: { 'Content-Type': 'application/json' },
29      body: JSON.stringify(input)
30    });
31
32    expect(response.status).toBe(200);
33    const data = await response.json();
34
35    expect(typeof data.prediction).toBe('number');
36    expect(typeof data.remaining_range).toBe('number');
37    expect(Array.isArray(data.shap_values)).toBe(true);
38    expect(data.shap_values.length).toBeGreaterThan(0);
39  });
40
41 // 2. MongoDB ve MariaDB Entegrasyonu
42 describe('Veritabanı entegrasyonu', () => {
43   let pool;
44   ...
```

Figure 74: Gerçeklenen Test 5

İlk entegrasyon testinde, makine öğrenmesi (ML) API'si ile sistemin doğru iletişim kurup kuramadığı test edilmiştir. Test kapsamında, ortalama hız, segment uzunluğu, ortalama ivmelenme, eğim, SOC (State of Charge), kalan enerji gibi çeşitli sürüs ve batarya parametrelerini içeren bir veri kümesi hazırlanarak <http://localhost:5002/predict> adresine POST isteğiyle gönderilmiştir. API'den dönen yanıtın HTTP durum kodu 200 (başarılı) olduğu kontrol edilmiş, ayrıca prediction, remaining_range ve shap_values gibi alanların doğru veri tipinde ve dolu olarak geldiği doğrulanmıştır. Bu testin başarıyla geçmesi, ML servisinin beklediği şekilde çalıştığını ve sistemle uyumlu veri akışı sağladığını göstermektedir.

```

41 // 2. MongoDB ve MariaDB Entegrasyonu
42 describe('Veritabanı entegrasyonu', () => {
43     let pool;
44     beforeAll(async () => {
45         // MongoDB bağlantısı
46         await mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true, useUnifiedTopology: true });
47         // MariaDB bağlantısı
48         pool = mariadb.createPool({
49             host: process.env.DB_HOST || 'localhost',
50             port: process.env.DB_PORT || 3306,
51             user: process.env.DB_USER || 'root',
52             password: process.env.DB_PASSWORD || 'password',
53             database: process.env.DB_NAME || 'vehicle_tracking'
54         });
55     });
56     afterAll(async () => {
57         await mongoose.disconnect();
58         await pool.end();
59     });
60     test('MariaDB: Araç verisi eklenip okunabiliyor', async () => {
61         const conn = await pool.getConnection();
62         await conn.query("INSERT INTO vehicle_tracking_fiware (vehicle_id, latitude, longitude, speed, timestamp)
63         | ['test_vehicle', 39.75, 30.48, 50, new Date()]);
64         const rows = await conn.query("SELECT * FROM vehicle_tracking_fiware WHERE vehicle_id = ?",
65         expect(rows.length).toBeGreaterThan(0);
66         await conn.query("DELETE FROM vehicle_tracking_fiware WHERE vehicle_id = ?",
67         ['test_vehicle']);
68         conn.release();
69     });
70 });

```

Figure 75: Gerçeklenen Test 6

İkinci test grubunda, sistemin hem MongoDB hem de MariaDB veritabanlarına sorunsuz erişebildiği ve bu veritabanlarıyla temel CRUD işlemlerini gerçekleştirebildiği test edilmiştir. beforeAll bloğunda her iki veritabanına bağlantılar gerçekleştirilmiş, afterAll kısmında ise bağlantılar düzgün bir şekilde kapatılmıştır. MariaDB testinde örnek bir araç verisi vehicle_tracking_fiware tablosuna eklenmiş, ardından bu veri okunmuş ve varlığı doğrulanmıştır. Son olarak test verisi silinerek veritabanı temiz tutulmuştur. Bu test, MariaDB bağlantısının, veri yazma ve okuma işlemlerinin başarılı olduğunu göstermektedir. MongoDB ile ilgili işlemler için ortam hazır hale getirilmiş olsa da bu örnekte sadece bağlantı testi yapılmıştır.

```

71 // 3. SUMO Python Server Entegrasyonu
72 test('SUMO serverdan araç verisi alınabiliyor', async () => {
73   const response = await fetch('http://localhost:5002/sumo-data');
74   expect(response.status).toBe(200);
75   const data = await response.json();
76   expect(data).toHaveProperty('vehicle_id');
77   expect(data).toHaveProperty('latitude');
78   expect(data).toHaveProperty('longitude');
79   expect(data).toHaveProperty('speed');
80 });
81 });

```

Figure 76: Gerçeklenen Test 7

Üçüncü ve son teste, SUMO simülasyonu tarafından sağlanan Python tabanlı bir API servisine erişim ve veri alma kabiliyeti test edilmiştir. <http://localhost:5002/sumo-data> adresine GET isteği gönderilmiş, dönen yanıtın HTTP durum kodu 200 olması beklenmiştir. Ayrıca, gelen JSON verisinde vehicle_id, latitude, longitude ve speed gibi temel alanların yer alıp olmadığı kontrol edilmiştir. Testin başarılı geçmesi, SUMO tabanlı simülasyon sisteminin aktif olduğunu ve frontend veya diğer bileşenler tarafından kullanılabilir olduğunu doğrulamaktadır.

```

PASS frontend/src/__tests__/entegrasyontests.test.js (6.545 s)
  ✓ SUMO serverdan araç verisi alınabiliyor (1615 ms)
  ✓ MariaDB: Araç verisi eklenip okunabiliyor (1064 ms)
  ✓ ML API: Araç verisi gönderildiğinde kalan menzil ve SHAP faktörleri dönüyor (1025 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:  0 total
Time:        7.221 s
Ran all test suites.
PS C:\app>

```

Figure 77: Test sonuç 3

Toplamda 3 testin tamamı başarıyla geçmiş ve tüm sistem bileşenlerinin doğru ve entegre şekilde çalıştığı onaylanmıştır. Test dosyası 6.545 saniyede tamamlanmıştır.

D.3.3 Performans Test Tasarımı

Araç Takip Modülü Performans Testleri:

```

test('Son 50 araç konumu okunabiliyor ve alanlar eksiksiz', async () => {
  const response = await fetch('http://localhost:3001/api/direct-query/vehicle-locations');
  expect(response.status).toBe(200);
  const data = await response.json();
  expect(Array.isArray(data)).toBe(true);
  expect(data.length).toBeGreaterThan(0);

  // Her kayıttı gerekli alanlar var mı?
  data.forEach(vehicle => [
    expect(vehicle).toHaveProperty('vehicle_id');
    expect(vehicle).toHaveProperty('latitude');
    expect(vehicle).toHaveProperty('longitude');
    expect(vehicle).toHaveProperty('speed');
    expect(vehicle).toHaveProperty('last_updated');
  ]);
});

```

Figure 78: Gerçeklenen Test 8

Araç konumlarının API üzerinden okunabilirliğini ve performansını ölçmek amacıyla üç farklı test senaryosu uygulanmıştır. İlk testte, /api/direct-query/vehicle-locations endpoint'inden dönen verinin bir dizi (array) olup olmadığı ve içinde en az bir araç verisi içerip içermediği kontrol edilmiştir. Her bir araç nesnesinin vehicle_id, latitude, longitude, speed ve last_updated gibi tüm gerekli alanlara sahip olması beklenmiştir. Yapılan kontroller sonucunda tüm veriler eksiksiz şekilde geldiği için bu test başarıyla geçmiştir.

```
test('Canlı araç konumları okunabiliyor ve güncel', async () => {
  const response = await fetch('http://localhost:3001/api/direct-query/vehicle-locations');
  expect(response.status).toBe(200);
  const data = await response.json();
  expect(Array.isArray(data)).toBe(true);

  // Her kayıtta gerekli alanlar var mı ve son güncelleme çok eski mi?
  const now = Date.now();
  data.forEach(vehicle => {
    expect(vehicle).toHaveProperty('vehicle_id');
    expect(vehicle).toHaveProperty('latitude');
    expect(vehicle).toHaveProperty('longitude');
    expect(vehicle).toHaveProperty('speed');
    expect(vehicle).toHaveProperty('last_updated');
    // Son güncelleme 10 dakikadan eski olmamalı (örnek)
    const updated = new Date(vehicle.last_updated).getTime();
    expect(now - updated).toBeLessThan(10 * 60 * 1000);
  });
});
```

Figure 79: Gerçeklenen Test 9

İkinci testte, gelen araç konumlarının sadece yapısal doğruluğu değil aynı zamanda güncelliliği de kontrol edilmiştir. Yine aynı endpoint üzerinden gelen veriler incelenmiş ve her bir aracın last_updated alanının zaman damgası alınarak, bu bilginin 10 dakikadan eski olup olmadığı denetlenmiştir. Bu, sistemin canlı takip özelliğini güvenilir biçimde sürdürübilmesi için önemli bir kriterdir. Tüm kayıtlar güncel olduğu için bu test de başarıyla tamamlanmıştır.

```
test('Son 50 araç konumu yanıtı <2s içinde geliyor', async () => {
  const start = Date.now();
  const response = await fetch('http://localhost:3001/api/direct-query/vehicle-locations');
  expect(response.status).toBe(200);
  const data = await response.json();
  const end = Date.now();
  expect(end - start).toBeLessThan(2000);
  expect(Array.isArray(data)).toBe(true);
});
```

Figure 80: Gerçeklenen Test 10

Üçüncü ve son testte ise sistemin performansına odaklanılmıştır. API'nin yanıt süresi ölçüleerek, veri setinin 2 saniyeden kısa sürede döndürülp döndürülmediği kontrol edilmiştir. Test başlangıcı ve bitiş arasındaki süre hesaplanarak yapılan bu performans testi de 136 milisaniye gibi oldukça kısa bir sürede tamamlanmış ve başarılı bulunmuştur.

```
PASS performance-tests/performance.test.js
Araç Konumları Okuma Testleri
Araç Konumları Okuma Testleri
✓ Son 50 araç konumu okunabiliyor ve alanlar eksiksiz (187 ms)
✓ Canlı araç konumları okunabiliyor ve güncel (151 ms)
✓ Son 50 araç konumu yanıtı <2s içinde geliyor (136 ms)

Test Suites: 1 passed, 1 total
Tests:      3 passed, 3 total
Snapshots:  0 total
Time:       3.372 s, estimated 5 s
Ran all test suites.
PS C:\ann> □
```

Figure 81: Test sonuç 4

Toplamda 3 testin tamamı başarıyla geçmiştir. performance.test.js dosyasında yer alan bu testler 3.372 saniyede çalışmış ve sistemin hem doğruluk hem güncelilik hem de performans açısından yeterli seviyede olduğunu göstermiştir.

Performans İzleme Modülü Performans Testleri:

```
frontend > src > __tests__ > 🚩 performance_monitoring.test.js > ...
1 import React from 'react';
2 import '@testing-library/jest-dom';
3 import { render, screen, fireEvent } from '@testing-library/react';
4 import dayjs from 'dayjs';
5 import Performance from './pages/Performance';
6
7 test('Tarih filtreleme ile iframe URL parametreleri doğru güncelleniyor'
8   render(<Performance />);
9   // Tarih inputlarını bul
10  const startInput = screen.getByLabelText(/start date/i);
11  const endInput = screen.getByLabelText(/end date/i);
12  const applyBtn = screen.getByRole('button', { name: /apply/i });
13
14  // Tarihleri değiştir
15  fireEvent.change(startInput, { target: { value: '2024-05-01T00:00' } })
16  fireEvent.change(endInput, { target: { value: '2024-05-10T23:59' } });
17  fireEvent.click(applyBtn);
18
19  const from = dayjs('2024-05-01T00:00').valueOf();
20  const to = dayjs('2024-05-10T23:59').valueOf();
21
22  // Tüm iframe'lerin src'sinde from ve to parametreleri olmalı
23  const iframes = screen.getAllByTitle(/panel/i);
24  iframes.forEach((iframe) => {
25    expect(iframe.src).toContain(`from=${from}`);
26    expect(iframe.src).toContain(`to=${to}`);
27  });
28});
```



Figure 82: Gerçeklenen Test 11

İlk testte, kullanıcıların manuel olarak tarih aralığı seçerek performans verilerini filtreleme özelliği test edilmiştir. React Testing Library kullanılarak Performance bileşeni render edilmiştir. Ardından, "start date" ve "end date" etiketli tarih giriş alanları bulunmuş, bu alanlara sırasıyla "2024-05-01T00:00" ve "2024-05-10T23:59" değerleri girilmiştir. Bu girişlerin ardından "Apply" adlı butona tıklanarak tarih aralığı uygulanmıştır. Testin devamında, sayfadaki tüm iframe bileşenlerinin src URL'leri kontrol edilmiştir. Her bir iframe'in src parametresinde from ve to zaman damgalarının, girilen tarih aralığına uygun olarak doğru şekilde yer aldığı doğrulanmıştır. Sonuç olarak, tüm iframe'ler doğru tarih aralığını içeren URL'lerle güncellendiği için bu test başarıyla geçmiştir. Test süresi yaklaşık 2455 milisaniye sürmüştür.

```


30  test('Hızlı tarih butonları doğru zaman aralığı üretiyor', () => {
31    render(<Performance />);
32    const todayBtn = screen.getByRole('button', { name: /today/i });
33    fireEvent.click(todayBtn);
34
35    const todayStart = dayjs().startOf('day').valueOf();
36    const todayEnd = dayjs().endOf('day').valueOf();
37
38    const iframes = screen.getAllByTitle(/panel/i);
39    iframes.forEach((iframe) => {
40      expect(iframe.src).toContain(`from=${todayStart}`);
41      expect(iframe.src).toContain(`to=${todayEnd}`);
42    });
43  });


```

Figure 83: Gerçeklenen Test 12

İkinci testte ise, kullanıcıların tarih seçmeden yalnızca "Today" (Bugün) butonuna basarak hızlı şekilde bugünün verilerini filtreleme özelliği test edilmiştir. Yine Performance bileşeni render edilerek test ortamına alınmıştır. Ardından, "Today" adlı buton bulunarak kullanıcı tıklaması simüle edilmiştir. Bu işlem sonucunda iframe bileşenlerinin src URL'leri tekrar kontrol edilmiş, her bir iframe'in adresinde bugünün başlangıç ve bitiş zaman damgalarının (from ve to) doğru şekilde yer aldığı doğrulanmıştır. Bu test, arayüzdeki hızlı tarih filtreleme özelliğinin doğru ve etkili çalıştığını göstermiştir. Test başarıyla tamamlanmış olup, yaklaşık 1518 milisaniyede sonuçlanmıştır.

```

PASS frontend/src/_tests_/performance_monitoring.test.js (7.634 s)
  ✓ Tarih filtreleme ile iframe URL parametreleri doğru güncelleniyor (2455 ms)
  ✓ Hızlı tarih butonları doğru zaman aralığı üretiyor (1518 ms)
  ✓ Tarih filtreleme ile iframe URL parametreleri doğru güncelleniyor (2455 ms)
  ✓ Hızlı tarih butonları doğru zaman aralığı üretiyor (1518 ms)
  ✓ Hızlı tarih butonları doğru zaman aralığı üretiyor (1518 ms)

```

```

Test Suites: 1 passed, 1 total
Tests:      2 passed, 2 total
Snapshots:  0 total
Time:       8.693 s
Ran all test suites.
PS C:\app>

```

Figure 84: Test sonuç 5

Her iki test sonucunda, performans izleme ekranının hem kullanıcı etkileşimlerine duyarlı hem de URL parametrelerini doğru biçimde oluşturup iframe'leri senkronize edebildiği görülmüştür. Bu da sistemin hem doğruluk hem de kullanılabilirlik açısından yeterli seviyede olduğunu göstermektedir.

D.4. Yazılım/Veri Tabanı/Donanım Gerçeklemeleri

D.4.2 Araç Takip Modülü Yazılım/Veri Tabanı/Donanım Gerçeklemeleri

D.4.1.1 Enerji Tüketim Modeli

Modeli eğitmeden önce ilk olarak araçtan alınan veriler üzerinde temizleme düzenleme ve segmentleme işlemleri yaptık. Veriler zaman sırasına göre alınır ve Haversine formülü kullanılarak son konumdan geriye doğru gidilerek ardışık noktalar arasındaki mesafeler toplanır ve toplamda 100 metreyi aşan bir segment oluşturulur. Veriler bu şekilde 100 metrelük segmentler halinde kaydedilir. Bu segmentlerde ortalama hız, ivme ve yük gibi değerler bu veri kümesi üzerinden hesaplanır. Ayrıca segmentin başlangıç ve bitiş noktaları arasındaki yükseklik farkına göre eğim değeri belirlenir. Verileri düzenleme işlemi bu şekilde tamamlanır.

Segmentleme işleminin ardından segment bazlı enerji tüketimi tahmini için çeşitli regresyon algoritmaları kullanılarak kapsamlı bir model karşılaştırması yapılmıştır. Amaç, belirli yol segmentleri için elektrikli araçların toplam enerji tüketimini en iyi şekilde tahmin edebilecek modeli belirlemektir.

Modellemeye dahil edilen değişkenler:

- segment_length: Segmentin uzunluğu (metre cinsinden)
- slope: Yol eğimi (derece)
- avg_vehicle_speed: Ortalama araç hızı (m/s)
- avg_Acceleration: Ortalama ivmelenme (m/s^2)
- avg_Total_Mass: Araç kütlesi (kg)

Hedef değişken (target):

- Total_Energy_Consumption: Segmentte tüketilen toplam enerji (Wh)

Veriler eğitim ve test seti olmak üzere %80 / %20 oranında bölünmüştür, ardından çeşitli regresyon algoritmaları uygulanmıştır. Aşağıdaki 23 farklı regresyon modeli değerlendirilmiştir:

- **Lineer Modeller:** Linear Regression, Ridge, Lasso, ElasticNet, Bayesian Ridge
- **Karar Ağaçları ve Topluluk Yöntemleri:** Decision Tree, Random Forest, Extra Trees, Gradient Boosting, AdaBoost
- **Modern Yükseltme Algoritmaları:** XGBoost, LightGBM, CatBoost
- **Destek Vektör Makineleri ve KNN:** SVR, KNN
- **Polinom Tabanlı Model:** Polynomial Regression (degree=3)
- **Robust Modeller:** Huber, Theil-Sen, PassiveAggressive
- **Diğerleri:** Gaussian Process, Neural Network (MLP), Stacking, Voting

Her model aşağıdaki üç temel regresyon metriği ile değerlendirilmiştir:

- R² Score (Açıklanan varyans)
- MAE (Ortalama mutlak hata)
- RMSE (Kök ortalama kare hata)

| | Model | R^2 Score | MAE | RMSE |
|----|----------------------------------|-----------|-----------|-----------|
| 13 | CatBoost | 0.938127 | 2.109285 | 2.750254 |
| 5 | Random Forest | 0.924065 | 2.232641 | 3.046794 |
| 6 | Extra Trees | 0.917217 | 2.464114 | 3.181197 |
| 22 | Voting Regressor | 0.912888 | 2.403383 | 3.263317 |
| 21 | Stacking Regressor | 0.910614 | 2.552748 | 3.305641 |
| 7 | Gradient Boosting | 0.900941 | 2.796861 | 3.479904 |
| 12 | LightGBM | 0.884988 | 2.860647 | 3.749672 |
| 11 | XGBoost | 0.870633 | 2.924588 | 3.976787 |
| 14 | Polynomial Regression (Degree 3) | 0.866813 | 3.124242 | 4.035086 |
| 15 | Huber Regressor | 0.830965 | 3.727469 | 4.545790 |
| 0 | Linear Regression | 0.830396 | 3.831923 | 4.553434 |
| 1 | Ridge Regression | 0.830239 | 3.836466 | 4.555545 |
| 2 | Lasso Regression | 0.829802 | 3.851372 | 4.561402 |
| 18 | Bayesian Ridge | 0.829554 | 3.854004 | 4.564720 |
| 3 | ElasticNet | 0.828178 | 3.883844 | 4.583107 |
| 16 | Theil-Sen Regressor | 0.821022 | 3.738937 | 4.677574 |
| 8 | AdaBoost | 0.812122 | 3.870517 | 4.792472 |
| 4 | Decision Tree | 0.795821 | 3.784560 | 4.996045 |
| 9 | Support Vector Regressor (SVR) | 0.775151 | 4.217463 | 5.242836 |
| 10 | K-Nearest Neighbors (KNN) | 0.522800 | 6.098976 | 7.637857 |
| 19 | Gaussian Process | -0.798274 | 10.910514 | 14.826858 |
| 17 | Passive Aggressive Regressor | -1.038743 | 12.218539 | 15.787103 |
| 20 | Neural Network (MLP) | -1.455416 | 14.786066 | 17.325418 |

Figure 85: Model sonuçları

En iyi performansı gösteren model CatBoost olmuştur. CatBoost yüksek doğruluk $R^2 = 0.938$ ile veri varyansının büyük kısmını açıklayabilmektedir ve MAE ve RMSE gibi hata metriklerinde de en düşük değerlere ulaşır. Bu nedenle bu model seçilmiştir.

Uygulama içerisinde model .pkl dosyası olarak eklenmiş ve bir FlashAPI ile kullanılmıştır.

```
20 # Geçmiş segment verisi yükleniyor
21 historical_df = pd.read_csv("data/data.csv") > segment
22
23
24 @app.route('/predict', methods=['POST'])
25 def predict():
26     data = request.json
27     print("Received data:", data)
28
29     try:
30         # Eksik özellik kontrolü
31         missing_features = [feature for feature in feature_names if feature not in data]
32         if missing_features:
33             return jsonify({"error": f"Missing features: {missing_features}"}), 400
34
35         # Model girdisi oluşturuluyor
36         features = [data[feature] for feature in feature_names]
37         input_array = np.array(features).reshape(1, -1)
38
39         # Tahmin (Wh cinsinden segment tüketimi)
40         prediction = model.predict(input_array)[0]
41
42         # SHAP değerleri
43         shap_values = explainer(input_array)
44         shap_contribs = shap_values.values[0]
45
46         # Segment benzerliği üzerinden ortalama tüketim (Wh/km)
47         segment_length = float(data["segment_length"])
48         slope = float(data["slope"])
49         ...
```

Figure 86: model kullanım

Burada kullanılmak üzere aracın anlık FIWARE ile alınan verilerini kaydettiğimiz bir veri tabanından endpoint ile alınıp o veriler üzerinde segmentleme yapılır ve segment sonucu burdaki modele gönderilir. Model bir enerji tüketim tahmini oluşturur. Enerji tüketime etki eden faktörler için de aynı zamanda bu tahminin neye göre olduğunu da analiz etmek için SHAP (SHapley Additive exPlanations) yöntemi kullanılır. SHAP, modelin karar mekanizmasını şeffaf hale getiren bir yöntem sunmaktadır. Özellikle enerji tüketimi için faktörlerin tahminindeki etkisi sayısal olarak ortaya konulması gereklidir ve bu SHAP ile yapılır. Eğim (slope), hız (avg_vehicle_speed), yük(load) gibi değişkenlerin enerji tüketimini artırıcı ya da azaltıcı etkileri doğrudan görüntülenerek etkilerin yüzdelik değerleri pie chart ile arayüzde kullanıcıya sunulur.

D.4.1.2 Menzil Tahmini

```

flash-app > 🛡 app.py > 🚀 predict
25 def predict():
26
27     # Segment benzerliği üzerinden ortalama tüketim (Wh/km)
28     segment_length = float(data["segment_length"])
29     slope = float(data["slope"])
30     avg_speed = float(data["avg_vehicle_speed"])
31     avg_acceleration = float(data["avg_Acceleration"])
32     avg_mass = float(data["avg_Total_Mass"])
33
34     remaining_energy = data.get("remaining_energy")
35     nonlinear_energy = data.get("nonlinear_energy", 0)
36     segment_count = data.get("segment_count", 0)
37
38     # Sabit ilk menzil için kullanılıyor
39     if segment_count == 0:
40         segment_count = len(historical_df) # "data/data.csv" dosyasındaki segment sayısı
41         historical_df["km_energy_consumption"] = historical_df["Total_Energy_Consumption"] / (historical_df["Distance"] * segment_length)
42         nonlinear_energy = historical_df["km_energy_consumption"].mean() # Wh/km
43     else:
44         # Yeni segment tüketimi ile Enonlinear güncellemesi
45         segment_count += 1
46         nonlinear_energy = ((nonlinear_energy * (segment_count - 1)) + prediction) / segment_count
47
48     remaining_energy = remaining_energy - prediction
49     remaining_range = remaining_energy / nonlinear_energy
50
51     response = {
52         "prediction": prediction,
53     }
54
55     return response
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```

Figure 87: menzil kullanım

Uygulamanın menzil tahmini bileşeni hesaplayara, elektrikli araçlar için önemli bir özelliği barındırır. Bu sistem, aracın kalan batarya kapasitesine dayanarak gelecekte ne kadar daha yol alabileceğini tahmin eder. Menzil hesaplamasında iki temel veri noktası kullanılır: kalan enerji miktarı (Wh) ve kilometre başına ortalama enerji tüketimi (Wh/km). Kalan enerji değeri, her segment sonunda tahmin edilen enerji tüketimi modelden elde edilerek mevcut değerden düşürülür. Bu işlem, batarya kapasitesinin gerçek zamanlı olarak güncellenmesini sağlar.

Kalan menzil bilgisine ek olarak uygulama, tüketim ortalamasını belirlemek için önceki segmentlerdeki enerji tüketimlerini dikkate alır. Eğer sistemin başında bu değer mevcut değilse, geçmiş verilerin tutulduğu data.csv dosyasına dayanarak ortalama birim tüketim değeri hesaplanır. Bu geçmiş veriler her bir segmentin enerji tüketimini segment uzunluğuna bölgerek Wh/km cinsinden normalize eder. Eğer sistem çalışmaya başladıysa ve canlı segment verileri gelmeye devam ediyorsa, bu ortalama her yeni segment tüketimi de eklenecek tekrar bulunur ve güncellenir. Böylece, zamanla daha gerçekçi ve o araca özgü bir tüketim profili oluşturulmuş olur.

Menzil ise bu güncellenmiş ortalama ile kalan enerji değeri oranlanarak hesaplanır:

$$\text{remaining_range} = \text{remaining_energy} / \text{avg_consumption_wh_per_km}.$$

Bu yöntem sayesinde sistem yalnızca başlangıçtaki sabit varsayımlara değil, gerçek sürüş koşullarına ve canlı veriye dayalı olarak menzil tahmini yapar.

D.4.1.3 Kök Neden Analizi

Uyarı panelinde gösterilecek uyarılar, hata ve bilgilendirmeler için analizler yapılmıştır. Hata analizi olarak literatürde temel olarak 3 hata bulunması durumu için hata, uyarı ve bilgilendirme mesajları olarak gruplandırılmıştır. Ve her bir mesajın kök nedeni için temel olarak 6 neden belirlenmiştir. Her ana neden, sistemdeki farklı bileşenleri temsil eder ve bu bileşenlerin altında, uyarı üretimine yol açabilecek alt nedenler bulunmaktadır. Bu nedenler Ishikawa diyagramı(Balık kılıçlığı diyagramı) ile sunulmuştur.

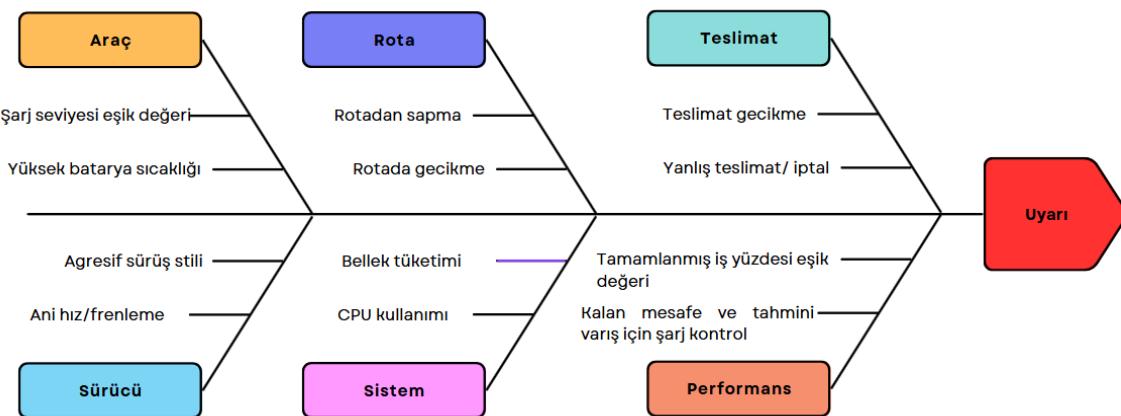


Figure 88: Kök Neden Analizi

Araç: Şarj seviyesi eşik değeri, yüksek batarya sıcaklığı gibi araçtan gelen hatalar için oluşturulan kök nedendir.

Rota: Rotadan sapma, rotada gecikme durumları için oluşturulan kök nedendir

Sürücü: Agresif sürüş stili, ani hız/frenleme gibi durumlar için oluşturulan kök nedendir

Sistem: Bellek tüketimi, CPU kullanımı gibi durumlar için oluşturulan kök nedendir

Teslimat: Teslimat gecikme, yanlış teslimat gibi durumlar için oluşturulan kök nedendir

Performans: Tamamlanmış iş yüzdesi eşik değeri, kalan mesafe ve tahmini varış için şarj kontrolü gibi durumlar için oluşturulan kök nedendir

Bu uyarılar filo yönetiminin bütününe kapsamaktadır ve uygulamaya statik gösterim olarak veritabanında bulunan verilerin gösterilmesi ile kullanılmaktadır.

D.4.1.4 Sumo ve FIWARE Entegrasyonu

```
sumo > sumo_func.py > ...
80     def sumo_close():
81         traci.close()
82         print("SUMO simulation closed successfully.")
83     except Exception as e:
84         print(f"Error closing SUMO: {str(e)}")
85
86
87
88
89
90
91
92
93
94     def sumo():
95         # SUMO Konfigürasyon Dosyası
96         SUMOCFG_PATH = 'ESOGU_SUMO/buyukdere_v5.sumocfg'
97         try:
98             # SUMO komutunu başlat
99             sumo_binary = os.path.join(os.getenv("SUMO_HOME"), "bin", "sumo-gui")
100            print(f"Starting SUMO with binary: {sumo_binary} and config: {SUMOCFG_PATH}")
101            traci.start([sumo_binary, "-c", SUMOCFG_PATH, "--start", "--quit-on-end", "--no-step-log", "--step-length", "0.5"])
102            print("SUMO started successfully!")
103        except Exception as e:
104            print(f"Error starting SUMO: {e}")
105            raise
106
107
108        db_connection = MySQLdb.connect(
109            host="127.0.0.1", # Changed from Docker config to standard localhost
110            user="root",
111            password="123456",
112            database="fleetmanagementdb",
113            connect_timeout=20
```



Figure 89: sumo kullanım

SUMO, şehir içi trafik ve araç hareketlerini simüle eden açık kaynaklı bir platformdur. Sistemimizde SUMO, sanal araçların ve siparişlerin senaryosunun hareket ettirilmesi ve bu verilerin backend'e aktarılması için kullanılır. Python traci kütüphanesi ile rota optimizasyondan gelmesi planlanan SUMO simülasyonu başlatılır. Her simülasyon adımında, SUMO'daki tüm araçların konum, hız, enerji, batarya gibi verileri alınır. Bu veriler, Python tarafında MQTT veya HTTP ile backend'e(MariaDB) kaydedilir. Anlık haritada aracın ve araç verilerinin gösterimi backenden alınan veriler ile sağlanır.

```
frontend > src > utils > parseRoutexml.js > parseRouteXmlFile > parsed > routes.map() callback > rawVehicleId
1  export async function parseRouteXmlFile(filePath = "parse-routes") {
2      try {
3          console.log("Fetching XML file from:", filePath);
4          const response = await fetch(filePath);
5          if (!response.ok) {
6              throw new Error(`Failed to fetch file: ${response.status}`);
7          }
8          const text = await response.text();
9          console.log(` Fetched XML content:`, text.slice(0, 200), "..."); // ilk 200 karakteri logla
10
11         // DOMParser ile XML'i parse et
12         const parser = new DOMParser();
13         const xmlDoc = parser.parseFromString(text, "application/xml");
14
15         // Rotayı çözümle
16         const routes = Array.from(xmlDoc.querySelectorAll("Route4Plan > Solution > Routes > Route"));
17         console.log(` Found routes:`, routes.length);
18
19         const parsed = routes.map((route) => {
20             const rawVehicleId = route.getAttribute("VehicleId");
21             const vehicleId = rawVehicleId; // Orijinal VehicleId'yi kullan
22             const routeId = route.getAttribute("RouteId");
23
24             const nodes = Array.from(route.querySelectorAll("Nodes > Node"));
25             console.log(` Vehicle ${vehicleId} has ${nodes.length} stops.`);
26
27             const waypoints = [];
28             const stops = [];
```



Figure 90: Sipariş Parse

Sumo başlat butonuna tıklandığında simülasyon siparişlerinin tutulduğu xml dosyası parse edilir ve siparişler veritabanına kaydedilir. Araç sumoda ilerleyip ilgili siparişe geldiğinde sipariş durumu güncellenir ve bu şekilde sipariş takibi yapılır.

Sisteme entegre edilen FIWARE platformu ise gerçek zamanlı veri akışını yönetmek amacıyla kullanılır. Bu kapsamında bir MQTT listener (dinleyici) oluşturularak, araçlardan gelen veriler MQTT broker üzerinden anlık olarak alınır. Listener, bu verileri JSON formatında parse eder ve sistemde ilgili harita bileşenlerine entegre eder. Böylece hem simülasyon verileri hem de gerçek zamanlı saha verileri, aynı arayüzde eş zamanlı olarak takip edilebilir hale gelir. Bu yapı sayesinde SUMO tabanlı sanal test ortamı ile FIWARE destekli canlı veri akışı birleştirilerek hibrit bir izleme ve analiz platformu oluşturulmuştur.

D.4.2 Performans İzleme Modülü Yazılım/Veri Tabanı/Donanım Gerçeklemeleri

Bu bölümde, Performans İzleme Modülü'nün teknik altyapısı ve veri görselleştirme süreci detaylandırılmaktadır. Sistem, konteyner tabanlı bir mimari ile Docker, Grafana ve iframe entegrasyonu kullanılarak geliştirilmiş; kullanıcıların etkileşimli grafik panellere erişebileceği dinamik bir yapı oluşturulmuştur.

D.4.2.1 Docker ile Çevre Kurulumu

Performans izleme modülünün altyapısı, Docker konteyner teknolojisi kullanılarak oluşturulmuştur. Servislerin tutarlı ve hızlı bir şekilde ayağa kaldırılabilmesi amacıyla [docker-compose.yml](#) yapılandırma dosyasından yararlanılmıştır. Bu yapı sayesinde sistemin temel bileşenlerinden biri olan Grafana, 3004 portu üzerinden erişilebilir olacak şekilde konumlandırılmıştır.

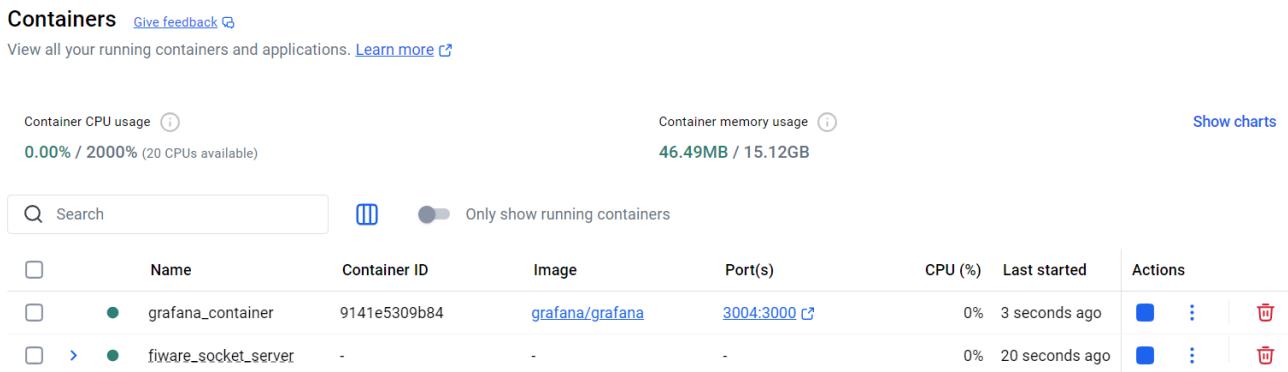


Figure 91: docker

Docker konteyneri başarıyla çalıştırıldıktan sonra, veritabanı Grafana'ya "Data Source" olarak tanıtılmıştır. Bu sayede panellerde yer alacak grafiklerin veri ihtiyaçları, tanımlanan bu kaynak aracılığıyla karşılanmış ve sistemin görselleştirme altyapısı etkin bir şekilde devreye alınmıştır.

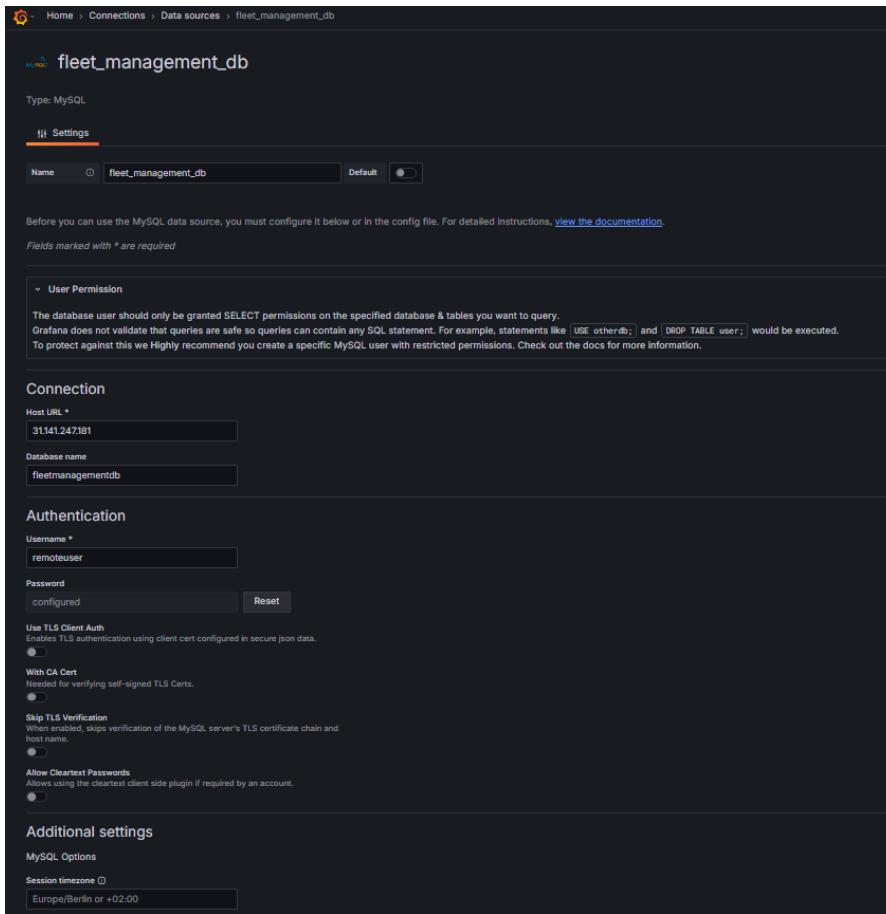


Figure 92: grafana

D.4.2.2 Dashboard ve Panel Oluşturma

Sistemde sunulması planlanan performans raporlarının her biri için Grafana üzerinde toplam 14 adet dashboard oluşturulmuştur. Her bir dashboard, ilgili raporun içerik ve veri yapısına uygun olarak tasarlanmıştır. Bu dashboard'lar içerisinde sütun grafik, çizgi grafik, pasta grafik ve tablo gibi farklı veri gösterme panelleri kullanılmıştır. Panellerde gösterilecek veriler, önceden tanımlanmış veri kaynaklarından sorgular aracılığıyla çekilerek dinamik biçimde sunulmuştur. Böylece her rapor için özel ve anlaşılır görselleştirmeler sağlanarak kullanıcıların veriye dayalı analiz yapmaları kolaylaştırılmıştır.

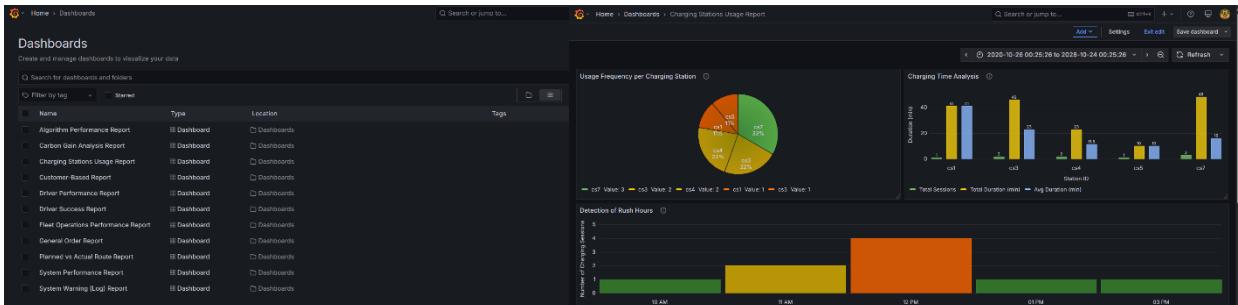


Figure 94: grafana raporlar

Figure 93: grafana dashboard

Her panel için, zaman damgalı performans verilerini çekmek amacıyla SQL veya uygun diğer sorgu dilleri kullanılarak sorgular tanımlanmıştır. Bu sorgular aracılığıyla elde edilen veriler, hız, enerji tüketimi, karbon kazanımı, bakım süreleri ve log sayısı gibi çeşitli metrikler temelinde grafiklere dönüştürülmüştür. Grafiklerin görselleştirilmesi sırasında renk geçişleri, eksen etiketleri ve açıklama metinleri gibi ayarlar, Grafana'nın web arayüzü üzerinden detaylı biçimde yapılandırılmıştır.

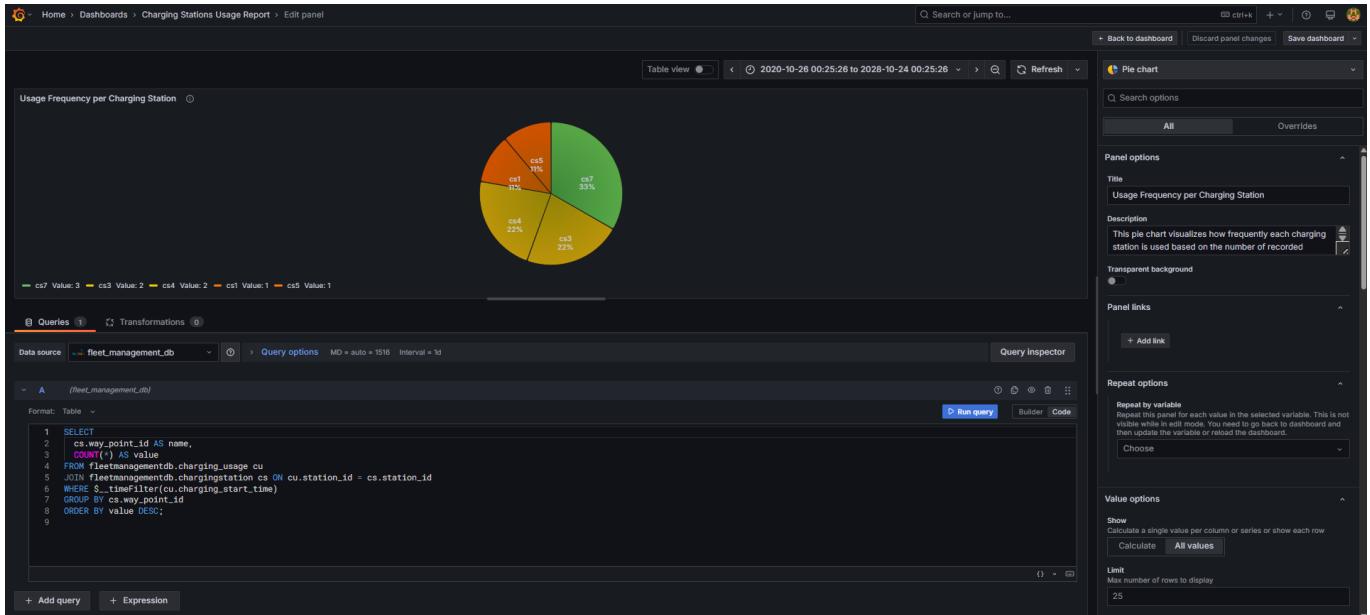


Figure 95: grafana grafik edit

D.4.2.3 iframe ile Panellerin Arayüze Gömülmesi

Grafana'dan elde edilen dashboard veya panellerin iframe kodları, "Share" menüsündeki "Embed" seçeneği kullanılarak alınmaktadır. Bu iframe kodları, uygulama arayüzüne gömülü olarak canlı ve etkileşimli veri görselleştirmeleri sağlar. İframe ile gömülü paneller, kullanıcıların raporlara kolayca erişmesini ve arayüzde farklı bölümler altında görüntülenmesini mümkün kılar. Ayrıca, iframe URL'leri zaman filtreleme parametreleri içerir; kullanıcılar "Dün", "Son 7 Gün" veya "Bu Ay" gibi ön tanımlı zaman aralıklarını seçiklerinde, iframe içindeki from ve to parametreleri dinamik olarak güncellenir. Bu sayede görseller, seçilen zaman aralığına göre otomatik olarak yenilenerek doğru ve güncel verilerin gösterilmesini sağlar. Böylece, uygulama hem kullanıcı dostu hem de esnek bir veri izleme deneyimi sunar.

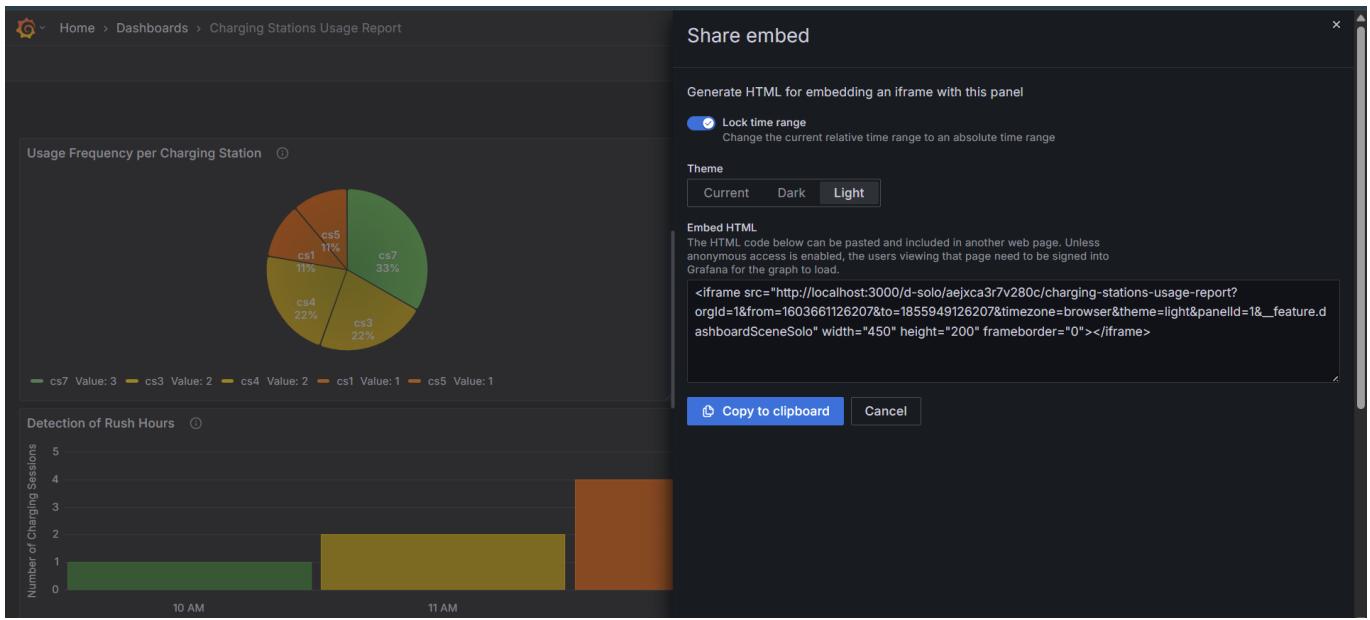


Figure 96: grafana paylaşma

Uygulama arayüzünde, raporlar akordiyon yapısı içinde organize edilmiştir. Her rapor başlığı, kullanıcıların ilgili veriye kolayca erişebilmesi için açılıp kapanabilir şekilde tasarlanmıştır. Açılan her akordiyon paneli içerisinde, ilgili dashboard veya panellere ait Grafana iframe'leri, düzenli bir grid yapısı kullanılarak yerleştirilmiştir. Bu sayede, birden fazla grafik veya tablo aynı anda düzenli ve estetik bir biçimde gösterilmekte, kullanıcılar ihtiyaç duyukları raporlara hızlıca ve rahatça erişebilmektedir.

```
<Accordion>
  <AccordionSummary expandIcon=<ExpandMoreIcon />>
    <Typography>Vehicle Maintenance Report</Typography>
  </AccordionSummary>
  <AccordionDetails>
    <Grid container spacing={2}>
      <Grid item xs={12} md={6}>
        <iframe
          src="http://localhost:3004/d-solo/deki0y109mlmoc/vehicle-maintenance-report?orgId=1&from=${from}&to=${to}&timezone=browser&theme=light&panelId=2&__feature.dashboardSceneSolo"
          width="100%"
          height="400"
          frameBorder="0"
          title="Vehicle Maintenance Panel 2"
        ></iframe>
      </Grid>
      <Grid item xs={12} md={6}>
        <iframe
          src="http://localhost:3004/d-solo/deki0y109mlmoc/vehicle-maintenance-report?orgId=1&from=${from}&to=${to}&timezone=browser&theme=light&panelId=1&__feature.dashboardSceneSolo"
          width="100%"
          height="400"
          frameBorder="0"
          title="Vehicle Maintenance Panel 1"
        ></iframe>
      </Grid>
      <Grid item xs={12} md={6}>
        <iframe
          src="http://localhost:3004/d-solo/deki0y109mlmoc/vehicle-maintenance-report?orgId=1&from=${from}&to=${to}&timezone=browser&theme=light&panelId=4&__feature.dashboardSceneSolo"
          width="100%"
          height="400"
          frameBorder="0"
          title="Vehicle Maintenance Panel 4"
        ></iframe>
      </Grid>
      <Grid item xs={12} md={6}>
        <iframe
          src="http://localhost:3004/d-solo/deki0y109mlmoc/vehicle-maintenance-report?orgId=1&from=${from}&to=${to}&timezone=browser&theme=light&panelId=3&__feature.dashboardSceneSolo"
          width="100%"
          height="400"
          frameBorder="0"
          title="Vehicle Maintenance Panel 3"
        ></iframe>
      </Grid>
    </Grid>
  </AccordionDetails>
</Accordion>
```

Figure 97: Grafana arayüze ekleme

E. SONUÇ VE ÖNERİLER

Sonuç olarak proje Filo yönetim sistemlerinde Araç Takip ve Performans İzleme modüllerinin geliştirilmesini kapsamaktadır. Araç Takip modülü içerisinde SUMO (Simulation of Urban MObility), şehir içi trafik ve araç hareketlerini modelleyerek, araç rotalarının ve teslimat noktalarının doğruluğunu test etme olanağı sağlamıştır. Python üzerinden başlatılan simülasyon senaryoları, araçların her adımda konum, hız, batarya ve enerji verilerini üretmiş; bu veriler MQTT veya HTTP protokollerile MariaDB veritabanına aktarılmıştır. Kullanıcı arayüzünde, bu veriler Leaflet tabanlı haritalar üzerinde anlık olarak görselleştirilmiştir. FIWARE platformu üzerinden gelen canlı araç verileri ise sistem içinde bir MQTT listener aracılığıyla alınmış, JSON formatında parse edilerek simülasyon verileriyle birlikte haritada sunulmuştur. Bu hibrit yaklaşım, hem sanal hem de gerçek araçları aynı sistem üzerinden izleme ve karşılaştırma olanağı tanımıştır. CatBoost algoritması ile geliştirilen makine öğrenmesi modeli ile yapılan enerji tüketimi ve geçmiş veri yaklaşımı ile kalan menzil tahmini sayesinde araç takibi detaylandırılmış, SHAP analizi ile bu enerji tahminine etki eden faktörler incelenmiştir. Ayrıca kök neden analizine dayalı olarak geliştirilen uyarı sistemi mimarisı, teslimat sürecinde yaşanabilecek gecikme, düşük batarya gibi problemler karşısında erken uyarıları gösterecek şekilde tasarlanmıştır.

Performans İzleme modülü kapsamında ise filo operasyonlarının derinlemesine analiz edilebilmesi amacıyla geliştirilmiştir. Toplamda 14 farklı rapor ve 50'den fazla grafik/tablo bileşeni ile operasyonel, çevresel ve sistemsel metrikleri kullanıcıya sunmaktadır. Akordiyon yapısında tasarılanan hiyerarşik panel sistemi, kullanıcıların ihtiyaç duydukları bilgilere sade bir arayüzle erişmesini sağlar. Tarih filtreleme özelliği, hazır zaman aralıklarının yanı sıra özel tarih-saat seçimi imkânı da sunarak, grafik ve tabloların dinamik şekilde güncellenmesini mümkün kılar. Renk geçişleri ve açıklayıcı bilgi metinleri, görsel farkındalık sağlayarak verilerin daha hızlı ve doğru yorumlanmasına katkıda bulunur. Modül içerisinde yer alan raporlama başlıklarını arasında şarj istasyonu kullanım analizleri, karbon tasarrufu hesaplamaları, sürücü ve araç performans karşılaştırmaları, kullanılan optimizasyon algoritmalarının etkinliği, sipariş, rota, sistem log ve araç bakım raporları yer almaktadır. Ayrıca zaman serisi tabanlı olarak sistem kaynak kullanımı da detaylı bir şekilde sunulmaktadır. Tüm bu veriler zaman damgali biçimde saklanmakta olup, geçmişe dönük eğilim analizlerinin yapılmasına olanak tanır. Özellikle karbon kazanımı, batarya doluluk durumu ve teslimat verimliliği gibi çevresel ve operasyonel göstergeler sistem içerisinde karşılaştırmalı olarak sunulmakta ve karar destek süreçlerine doğrudan katkı sağlamaktadır.

İlerleyen süreçlerde proje için bazı öneriler gerçekleştirilebilir

- **Gerçek Zamanlı Uyum Artırılmalı:** FIWARE'dan gelen verilerle simülasyonun daha senkron çalışması sağlanarak doğruluk artırılabilir ve FIWARE haritası için de sipariş kontrolü gibi diğer detaylar eklenebilir.
- **Enerji Tahmin Modeli Geliştirilmeli:** Hava durumu, yol tipi, yük gibi faktörler dahil edilerek modellerin doğruluğu yükseltilenbilir.
- **Kullanıcı Arayüzü Geliştirilmeli:** Kritik performans değişimlerini görsel olarak öne çıkarılan uyarı sistemleri geliştirilmelidir.

Bu proje, elektrikli araçların operasyonel etkinliğini artırmaya yönelik olarak simülasyon, tahmin, görselleştirme ve raporlama gibi birçok teknolojiyi entegre bir biçimde bir araya getirerek filo yönetim sistemleri için bir temel oluşturmaktadır. Sistem, yalnızca bugünün filo operasyonlarına değil, aynı zamanda gelecekteki akıllı şehir uygulamalarına da katkı sunacak niteliktedir.

REFERANSLAR

- [1] Meenambika, A., Karuppasamy, P., & Vithyaa, T. (2023). Fleet Guard: Advanced Fleet Management and Vehicle Monitoring System. *Journal of Electronic Design Technology*, 14(2), 18-24p.
- [2] Husak, V., Chyrun, L., Matseliukh, Y., Gozhyj, A., Nanivskyi, R., & Luchko, M. (2021). Intelligent Real-Time Vehicle Tracking Information System. In *MoMLET+ DS* (pp. 666-698).
- [3] Martins, J. A., & Rodrigues, J. M. (2025). Intelligent Monitoring Systems for Electric Vehicle Charging. *Applied Sciences*, 15(5), 2741.
- [4] Reddy, M. H., & Bhuvan, M. B. (2024). Real-Time vehicle tracking and fleet monitoring. *International Journal of Information Technology and Computer Engineering*, 12(4), 94-101.
- [5] Kauffmann, P., Williams, R., & Hildreth, J. (2013). *Fleet management performance monitoring* (No. FHWA/NC/2012-07). North Carolina. Dept. of Transportation. Research and Analysis Group.
- [6] Shukla, P., Swami, A. K., Kochhar, N., & Colaianni, F. Tracking Solutions for Fleet Management Systems. *Arrowtimes*.
- [7] Huang, J., Li, X., Zhou, T., Cai, B., He, J., & Hu, J. (2024, May). Prediction model of electric vehicle driving range based on support vector regression. In *2024 36th Chinese Control and Decision Conference (CCDC)* (pp. 875-880). IEEE.
- [8] Hasib, S. A., Saha, D. K., Islam, S., Tanvir, M., & Alam, M. S. (2021, November). Driving range prediction of electric vehicles: A machine learning approach. In *2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)* (pp. 1-6). IEEE.
- [9] Witvoet, K., Saad, S., Vidal, C., Ahmed, R., & Emadi, A. (2023, June). Electric Vehicle's Range and State of Charge Estimations using AutoML. In *2023 IEEE Transportation Electrification Conference & Expo (ITEC)* (pp. 1-6). IEEE.
- [10] Huang, H., Li, B., Wang, Y., Zhang, Z., & He, H. (2024). Analysis of factors influencing energy consumption of electric vehicles: Statistical, predictive, and causal perspectives. *Applied Energy*, 375, 124110.
- [11] Witvoet, K., Saad, S., Vidal, C., Ahmed, R., & Emadi, A. (2023, June). Electric Vehicle's Range and State of Charge Estimations using AutoML. In *2023 IEEE Transportation Electrification Conference & Expo (ITEC)* (pp. 1-6). IEEE.