Araç Kiralama Otomasyonu Proje Raporu

[İbrahim Kızılarslan 231307045, Mehmet Enes Dinç 231307043, Cihat Karataş 231307078][28.Grup] [Bilişim Sistemleri Mühendisliği] [Kocaeli Üniversitesi]
[mehmetenes2004.dinc@gmail.com]
[cihatkaratas32@gmail.com] [kizilarslanibrahim8@gmail.com]

Akademisyen: Dr. Öğretim Üyesi Samet Diri

Özetçe

Bu rapor, geliştirilen Araç Kiralama Otomasyonu projesinin detaylarını sunmaktadır. Proje, araç kiralama süreçlerini dijitalleştirerek verimliliği artırmayı, kullanıcı deneyimini iyileştirmeyi ve operasyonel yönetimi kolaylaştırmayı amaçlamaktadır. Rapor kapsamında problemin tanımı, yapılan araştırmalar, sistemin akış şeması, yazılım mimarisi, kullanılan veri tabanı yapısı (ER diyagramı ile birlikte) ve projenin genel yapısı ele alınmıştır. Proje, modern teknolojiler ve en iyi yazılım geliştirme pratikleri kullanılarak hayata geçirilmiştir.

Anahtar Kelimeler: Araç Kiralama, Otomasyon, .NET Core, Yazılım Mimarisi, Veri Tabanı Tasarımı, Web Uygulaması.

1. Problem Tanımı

Geleneksel araç kiralama süreçleri, genellikle manuel operasyonlara dayanmakta, bu da zaman kaybına, operasyonel hatalara, müşteri memnuniyetsizliğine ve kaynakların verimsiz kullanılmasına yol açabilmektedir. Müşterilerin araçları kolayca arayıp rezerve edebileceği, şirketlerin ise araç filolarını, rezervasyonları ve müşteri bilgilerini etkin bir şekilde yönetebileceği merkezi ve otomatik bir platforma ihtiyaç duyulmaktadır. Bu proje, bu ihtiyaçlara cevap vermek üzere, kullanıcı dostu bir arayüze sahip, güvenli ve ölçeklenebilir bir araç kiralama otomasyon sistemi geliştirmeyi hedeflemektedir. Temel problem, araç kiralama işlemlerindeki verimsizliği ortadan kaldırarak, hem son kullanıcılar hem de kiralama şirketleri için süreci basitleştirmek ve iyileştirmektir.

2. Yapılan Araştırmalar

Proje geliştirme sürecinde zaman zaman teknik sorunlarla karşılaşılmış ve bu sorunlara çözüm bulmak amacıyla çeşitli kaynaklardan araştırmalar yapılmıştır. Özellikle ASP.NET Core altyapısında servis katmanlarının oluşturulması, veritabanı ilişkilerinin doğru modellenmesi ve Entity Framework Core kullanımı gibi konularda çevrim içi kaynaklardan yararlanılmıştır.

Karşılaşılan bazı örnek problemler ve çözümleri şunlardır:

- **Veritabanı İlişkileri ve Foreign Key Hataları:**
  Entity Framework ile tablolar arasında ilişki kurulurken bazı foreign key hatalarıyla karşılaşıldı. Bu hataların çözümünde Microsoft'un resmi dökümantasyonu ve Stack Overflow üzerindeki örnekler incelendi.
- **DTO Kullanımı:**
  Katmanlı mimaride veri transferini kolaylaştırmak için DTO yapısı kullanıldı. Bu yapıların doğru şekilde modellenmesi ve controller–service katmanlarında kullanımı hakkında çeşitli blog yazıları ve GitHub örnek projelerinden faydalanıldı.

- **Bağımlılık Enjeksiyonu:**
  ASP.NET Core içerisinde servislerin ve repository'lerin DI ile nasıl tanımlanacağı konusunda Microsoft Docs üzerinde yer alan örnek kodlar incelendi.
- **Tasarım ve Kullanıcı Deneyimi:**
  Projenin arayüzünde kullanıcı dostu bir yapı oluşturmak için mevcut hazır bir **HTML/CSS/JS template** kullanılmıştır.

Teknoloji Seçimi:
Backend: .NET Core ve C# tercih edilmiştir. .NET Core'un platform bağımsızlığı, yüksek performansı, modüler yapısı ve geniş topluluk desteği bu kararda etkili olmuştur. ASP.NET Core Web API, RESTful servislerin geliştirilmesi için kullanılmıştır.

Frontend: Uygulamanın frontend (kullanıcı arayüzü) kısmı, herhangi bir modern JavaScript frameworkü (örneğin React, Angular veya Vue.js) kullanılmadan, klasik yöntemlerle geliştirilmiştir. Arayüz, HTML, CSS ve JavaScript dilleri kullanılarak oluşturulmuş ve Razor tabanlı .cshtml dosyaları ile ASP.NET Core yapısına entegre edilmiştir.

Veri Erişimi: Entity Framework Core (EF Core), ORM aracı olarak kullanılmıştır. Veritabanı işlemlerini kolaylaştırması, LINQ desteği ve migration özellikleri sayesinde geliştirme sürecini hızlandırmıştır.

Güvenlik: Kullanıcı kimlik doğrulama ve yetkilendirme işlemleri için JSON Web Tokens (JWT) kullanılmıştır.

Bağımlılık Yönetimi (DI): Projede, bağımlılıkların loosely coupled (gevşek bağlı) olması amacıyla ASP.NET Core'un yerleşik **Dependency Injection** yapısı kullanılmıştır. Gerekli servis ve repository sınıfları Program.cs dosyasında IServiceCollection üzerinden sisteme tanıtılmış ve constructor injection ile kullanılacak sınıflara enjekte edilmiştir. Bu yapı test edilebilirliği ve modülerliği artırmıştır.

Doğrulama (Validation): FluentValidation, model doğrulamalarını daha okunabilir ve yönetilebilir bir şekilde yapmak için kullanılmıştır.

Mimari Yaklaşımlar: Bu projede, **bağımlılıkların yönünü tersine çevirerek** sürdürülebilir, test edilebilir ve genişletilebilir bir yapı oluşturmak amacıyla **Onion Architecture (Soğan Mimarisi)** tercih edilmiştir.

Onion Architecture, katmanlar arası bağımlılıkları merkezden dışa doğru yönlendirerek, **iş mantığını (business logic)** merkeze alır ve dış katmanları (UI, veri erişimi vb.) bu mantığa bağımlı hale getirir. Bu sayede uygulama, altyapı katmanlarında yapılacak değişikliklerden etkilenmeden kolayca geliştirilebilir.

Kullanıcı Deneyimi (UX) ve Arayüz Tasarımı (UI): Kullanıcıların sistemi kolayca anlayıp kullanabilmesi için modern UX/UI prensipleri araştırılmış ve Bootstrap gibi kütüphanelerden faydalanılmıştır.

API Tasarımı: Swagger (OpenAPI) kullanılarak API dokümantasyonunun otomatik oluşturulması ve test edilebilirliğinin artırılması sağlanmıştır.

3. Akış Şeması
Sistemin temel kullanıcı ve yönetici akışları aşağıdaki gibi özetlenebilir:
Kullanıcı Akışı:
i. Sisteme Kayıt Olma / Giriş Yapma.
ii. Araçları Listeleme (Filtreleme: tarih, lokasyon, araç tipi vb.).
iii. Araç Detaylarını Görüntüleme (Özellikler, fiyat, uygunluk durumu).
iv. Araç Rezervasyonu Yapma (Kiralama ve iade tarihlerini seçme).
v. Ödeme Bilgilerini Girme ve Ödeme Yapma.
vi. Rezervasyon Onayını Alma.
vii. Rezervasyonlarını Görüntüleme / İptal Etme.
Yönetici (Admin) Akışı:
i. Sisteme Giriş Yapma.
ii. Araç Yönetimi (Ekleme, Güncelleme, Silme, Durumunu Değiştirme).
iii. Müşteri Yönetimi (Listeleme, Bilgilerini Görüntüleme).
iv. Rezervasyon Yönetimi (Listeleme, Onaylama, İptal Etme).
v. Marka/Model Yönetimi.
vi. Raporlama (Kiralama istatistikleri vb.).

4. Yazılım Mimarisi
Proje, yönetilebilirliği, test edilebilirliği ve genişletilebilirliği artırmak amacıyla katmanlı bir mimari üzerine inşa edilmiştir. README.md dosyasında belirtilen klasör yapısı (Core, Infrastructure, Presentation, Frontends) bu yaklaşımı desteklemektedir.

Core Katmanı: Projenin temel iş mantığını, varlıklarını (entities), arayüzlerini (interfaces) ve iş kurallarını içerir. Diğer katmanlardan bağımsızdır ve projenin kalbini oluşturur. Domain Driven Design (DDD) prensiplerine uygun olarak tasarlanmıştır.
Application: Uygulama servisleri, DTO'lar (Data Transfer Objects), AutoMapper konfigürasyonları, FluentValidation kuralları.

Domain: Varlıklar (Entities), temel iş kuralları, soyut repositori arayüzleri.

Infrastructure Katmanı : Core katmanında tanımlanan arayüzlerin somut uygulamalarını içerir. Dış bağımlılıklarla (veritabanı, dosya sistemi, harici servisler vb.) ilgilenir.

Persistence: Entity Framework Core ile veritabanı erişim kodları (DbContext, somut repositoriler, migration'lar).

Presentation Katmanı (Backend API): ASP.NET Core Web API projesini içerir. İstemcilerden gelen istekleri karşılar, Core katmanındaki uygulama servislerini kullanarak işlemleri gerçekleştirir ve sonuçları istemciye döndürür. Kimlik doğrulama (JWT), yetkilendirme ve Swagger API dokümantasyonu bu katmanda yer alır.

Frontends Katmanı (Frontend UI): Kullanıcı arayüzü, temel olarak:

- **HTML** ile sayfa yapısı oluşturulmuş,
- **CSS** ile stil ve tasarımlar düzenlenmiş,
- **JavaScript** kullanılarak sayfa dinamikliği sağlanmış ve API istekleri yapılmıştır.

API ile frontend arasında iletişim, JavaScript üzerinden yapılan fetch/XMLHttpRequest istekleri ile sağlanmıştır. Sayfa yenilenmeden veri alışverişi yapılmasına olanak tanıyan bu yapı sayesinde, kullanıcı deneyimi geliştirilmiştir. ASP.NET Core'un Razor altyapısı sayesinde, bazı statik içerikler de sunucu tarafında oluşturulmuştur.

Bu yapı, küçük ve orta ölçekli projelerde modern frameworklere ihtiyaç duymadan işlevsel çözümler üretmeye olanak sağlamaktadır.

5. Veri Tabanı Diyagramı

Araç kiralama otomasyonu için tasarlanan ilişkisel veri tabanı yapısı aşağıdaki temel varlıkları ve ilişkileri içermektedir.

1. Ana Varlık Tabloları

Authors (Yazarlar)

- AuthorID (PK, int, auto-increment)
- Name (nvarchar(100), not null)
- ImageUrl (nvarchar(500), nullable)
- Description (nvarchar(max), nullable)

Brands (Markalar)

- BrandID (PK, int, auto-increment)
- Name (nvarchar(100), not null)

Cars (Arabalar)

- CarID (PK, int, auto-increment)
- BrandID (FK, int, → Brands.BrandID)
- Model (nvarchar(100), not null)
- CoverImageUrl (nvarchar(500), nullable)
- Km (int, not null)
- Transmission (nvarchar(50), not null)
- Seat (tinyint, not null)
- Luggage (tinyint, not null)
- Fuel (nvarchar(50), not null)
- BigImageUrl (nvarchar(500), nullable)

İlişki: Cars.BrandID → Brands.BrandID (Many-to-One)

Features (Özellikler)

• FeaturesID (PK, int, auto-increment)
• Name (nvarchar(100), not null)

Pricings (Fiyat Kategorileri)

• PricingID (PK, int, auto-increment)
• Name (nvarchar(100), not null)

Locations (Lokasyonlar)

• LocationID (PK, int, auto-increment)
• Name (nvarchar(255), not null)

Customers (Müşteriler)

• CustomerID (PK, int, auto-increment)
• CustomerName (nvarchar(100), not null)
• CustomerSurname (nvarchar(100), not null)
• CustomerMail (nvarchar(255), not null)

2. Blog Sistemi Tabloları

Blogs (Blog Yazıları)

• BlogID (PK, int, auto-increment)
• Title (nvarchar(255), not null)
• AuthorID (FK, int, → Authors.AuthorID)
• CoverImageUrl (nvarchar(500), nullable)
• CreatedDate (datetime, not null)
• Description (nvarchar(max), nullable)

İlişki: Blogs.AuthorID → Authors.AuthorID (Many-to-One)

Comments (Blog Yorumları)

• CommentID (PK, int, auto-increment)
• Name (nvarchar(100), not null)
• Description (nvarchar(max), not null)
• Email (nvarchar(255), not null)
• CreatedDate (datetime, not null)
• BlogID (FK, int, → Blogs.BlogID)

İlişki: Comments.BlogID → Blogs.BlogID (Many-to-One)

TagClouds (Etiketler)

• TagCloudID (PK, int, auto-increment)
• Title (nvarchar(100), not null)
• BlogID (FK, int, → Blogs.BlogID)

İlişki: TagClouds.BlogID → Blogs.BlogID (Many-to-One)

3. Araç Detay Tabloları

CarDescriptions (Araç Açıklamaları)

• CarDescriptionID (PK, int, auto-increment)
• CarID (FK, int, → Cars.CarID)
• Details (nvarchar(max), nullable)

İlişki: CarDescriptions.CarID → Cars.CarID (One-to-One)

CarFeatures (Araç Özellikleri)

• CarFeaturesID (PK, int, auto-increment)
• CarID (FK, int, → Cars.CarID)
• FeaturesID (FK, int, → Features.FeaturesID)
• Available (bit, not null)

İlişkiler:

- CarFeatures.CarID → Cars.CarID (Many-to-One)
- CarFeatures.FeaturesID → Features.FeaturesID (Many-to-One)

CarPricings

• CarPricingID (PK, int, auto-increment)
• CarID (FK, int, → Cars.CarID)
• PricingID (FK, int, → Pricings.PricingID)
• Amount (decimal(18,2), not null)

İlişkiler:

- CarPricings.CarID → Cars.CarID (Many-to-One)
- CarPricings.PricingID → Pricings.PricingID (Many-to-One)

4. Kiralama Sistemi Tabloları

RentCars (Kiralık Araçlar)

• RentCarID (PK, int, auto-increment)
• LocationID (FK, int, → Locations.LocationID)
• CarID (FK, int, → Cars.CarID)
• Available (bit, not null)

İlişkiler:

- RentCars.LocationID → Locations.LocationID (Many-to-One)
- RentCars.CarID → Cars.CarID (Many-to-One)

RentCarProcesses (Kiralama İşlemleri)

• RentCarProcessID (PK, int, auto-increment)
• CarID (FK, int, → Cars.CarID)
• PickUpLocation (FK, int, → Locations.LocationID)
• DropOffLocation (FK, int, → Locations.LocationID)
• PickUpDate (date, not null)
• DropOffDate (date, not null)
• PickUpTime (time, not null)
• DropOffTime (time, not null)
• CustomerID (FK, int, → Customers.CustomerID)
• PickUpDescription (nvarchar(max), nullable)
• DropOffDescription (nvarchar(max), nullable)
• TotalPrice (decimal(18,2), not null)

İlişkiler:

- RentCarProcesses.CarID → Cars.CarID (Many-to-One)
- RentCarProcesses.PickUpLocation → Locations.LocationID (Many-to-One)
- RentCarProcesses.DropOffLocation → Locations.LocationID (Many-to-One)
- RentCarProcesses.CustomerID → Customers.CustomerID (Many-to-One)

Reservations (Rezervasyonlar)

• ReservationID (PK, int, auto-increment)
• Name (nvarchar(100), not null)
• Surname (nvarchar(100), not null)
• Phone (nvarchar(50), not null)
• PickUpLocationID (FK, int, → Locations.LocationID)
• DropOffLocationID (FK, int, → Locations.LocationID)
• Email (nvarchar(255), not null)
• CarID (FK, int, → Cars.CarID)
• Age (int, not null)
• DriverLicenseYear (int, not null)
• Description (nvarchar(max), nullable)
• Status (nvarchar(50), not null)

İlişkiler:

- Reservations.PickUpLocationID → Locations.LocationID (Many-to-One)
- Reservations.DropOffLocationID → Locations.LocationID (Many-to-One)
- Reservations.CarID → Cars.CarID (Many-to-One)

Reviews (Değerlendirmeler)

• ReviewID (PK, int, auto-increment)
• CustomerName (nvarchar(100), not null)
• CustomerImage (nvarchar(255), nullable)
• Comment (nvarchar(max), not null)
• RatingValue (int, not null)
• ReviewDate (datetime, not null)
• CarID (FK, int, → Cars.CarID)

İlişki: Reviews.CarID → Cars.CarID (Many-to-One)

5. İçerik Yönetimi Tabloları

Abouts (Hakkımızda)

• AboutID (PK, int, auto-increment)
• Title (nvarchar(255), not null)
• Description (nvarchar(max), nullable)
• ImageUrl (nvarchar(500), nullable)

Banners (Bannerlar)

• BannerId (PK, int, auto-increment)
• Title (nvarchar(255), not null)
• Description (nvarchar(max), nullable)
• VideoDescription (nvarchar(max), nullable)
• VideoUrl (nvarchar(500), nullable)

Services (Hizmetler)

• ServiceID (PK, int, auto-increment)
• Title (nvarchar(200), not null)
• Description (nvarchar(max), not null)
• IconUrl (nvarchar(255), nullable)

Testimonials (Referanslar)

• TestimonialID (PK, int, auto-increment)
• Name (nvarchar(100), not null)
• Title (nvarchar(100), not null)
• ImageUrl (nvarchar(255), nullable)
• Comment (nvarchar(max), not null)

6. İletişim ve Sosyal Medya Tabloları

Contacts (İletişim)

• ContactID (PK, int, auto-increment)
• Name (nvarchar(100), not null)
• Email (nvarchar(255), not null)
• Subject (nvarchar(255), not null)
• Message (nvarchar(max), not null)
• SendDate (datetime, not null)

FooterAddresses (Alt Adresler)

• FooterAddressID (PK, int, auto-increment)
• Phone (nvarchar(50), nullable)
• Email (nvarchar(255), nullable)
• Description (nvarchar(max), nullable)
• Address (nvarchar(500), nullable)

SocialMedias

• SocialMediaID (PK, int, auto-increment)
• Name (nvarchar(100), not null)
• Url (nvarchar(255), not null)
• Icon (nvarchar(255), not null)

7. Tablo İlişkileri Özeti

One-to-Many İlişkiler

- **Brands → Cars**: Bir marka birden fazla araca sahip olabilir
- **Authors → Blogs**: Bir yazar birden fazla blog yazabilir
- **Blogs → Comments**: Bir blog birden fazla yorum alabilir
- **Blogs → TagClouds**: Bir blog birden fazla etikete sahip olabilir
- **Cars → CarDescriptions**: Her araç bir açıklamaya sahip olabilir
- **Cars → Reviews**: Bir araç birden fazla değerlendirme alabilir
- **Cars → RentCarProcesses**: Bir araç birden fazla kiralamada kullanılabilir
- **Cars → Reservations**: Bir araç birden fazla rezervasyon alabilir
- **Customers → RentCarProcesses**: Bir müşteri birden fazla kiralama yapabilir
- **Locations → RentCarProcesses**: Bir lokasyon birden fazla kiralamada kullanılabilir
- **Locations → Reservations**: Bir lokasyon birden fazla rezervasyonda kullanılabilir

Many-to-Many İlişkiler

- **Cars ↔ Features** (CarFeatures tablosu ile): Araçlar birden fazla özelliğe sahip olabilir
- **Cars ↔ Pricings** (CarPricings tablosu ile): Araçlar birden fazla fiyat kategorisine sahip olabilir
- **Cars ↔ Locations** (RentCars tablosu ile): Araçlar birden fazla lokasyonda mevcut olabilir

6. Genel Yapı

Proje, modern bir araç kiralama otomasyon sistemi sunmak üzere geliştirilmiştir. Sistem, kullanıcıların web üzerinden araçları incelemesine, rezervasyon yapmasına ve ödemelerini gerçekleştirmesine olanak tanır. Yöneticiler için ise araç filosu yönetimi, müşteri takibi, rezervasyon ve ödeme işlemleri gibi kapsamlı bir yönetim paneli sunar.

Teknolojik Altyapı:
Backend: ASP.NET Core 8.0 Web API, C#
Frontend: Html,Css,Javascript Bootstrap
Veritabanı: İlişkisel bir veritabanı (SQL Server) ve Entity Framework Core
Kimlik Doğrulama & Yetkilendirme: JWT
API Dokümantasyonu: Swagger/OpenAPI
Bağımlılık Enjeksiyonu: ASP.NET Core'un yerleşik Dependency Injection (DI) kullanılmıştır

Nesne Eşleme: Projede katmanlar arası veri aktarımı için **DTO (Data Transfer Object)** sınıfları tanımlanmış ve bu sınıflar ile Entity (varlık) sınıfları arasında eşleme işlemleri gerçekleştirilmiştir.(Manuel Mapping)

Doğrulama: FluentValidation

Temel Fonksiyonlar:
Kullanıcı Kayıt ve Giriş İşlemleri
Detaylı Araç Arama ve Filtreleme
Online Rezervasyon ve Ödeme
Rezervasyon Yönetimi (Kullanıcı ve Yönetici)
Araç Filosu Yönetimi (Ekleme, Güncelleme, Durum Takibi)
Müşteri Yönetimi
Marka ve Model Yönetimi
Lokasyon Yönetimi

Sistem, modüler ve katmanlı mimarisi sayesinde gelecekteki ihtiyaçlara göre kolayca genişletilebilir ve bakımı yapılabilir bir yapıdadır.

7. Referanslar

[1] Microsoft, ".NET Documentation," [Online].
Available: https://docs.microsoft.com/dotnet/.[2] The Clean Code Blog, "The Clean Architecture," by Robert C. Martin, [Online]. https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.
[3] Microsoft, "Entity Framework Core Documentation," [Online].
https://docs.microsoft.com/ef/core/.
[4] J. Doe, Software Engineering Principles, New York, NY, USA: Academic Press, 2023.
[5] IEEE, "IEEE Citation Reference," [Online].
https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf.