T.C.

VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

E-TİCARET SİTESİ

PROJE EKİBİ:

Barış AKKUŞ-23390008067

Barış SOMEROĞLU-23390008045

Helin BARIŞ-23390008048

Berivan GÜLTEPE-23390008013

İsa ACAR-23390008049

Teslim Tarihi:16.06.2025

E-Ticaret Sitesi Projesi - Genel Bilgilendirme Dokümanı

Proje Genel Bakış

Bu proje, modern web teknolojileri kullanılarak geliştirilmiş kapsamlı bir e-ticaret platformudur. Elektronik ürünlerin (laptop ve telefon) satışına odaklanan platform, kullanıcılara çevrimiçi alışveriş deneyimi sunarken, yöneticilere ürün ve sipariş yönetimi araçları sağlamaktadır.

Proje Amacı ve İşlevi

Ana Amaç

Platform, kullanıcıların elektronik ürünleri güvenli bir şekilde satın almalarını sağlamak ve işletmelere dijital ticaret ortamında güçlü bir varlık kazandırmak amacıyla tasarlanmıştır.

Temel İşlevler

- Ürün katalog yönetimi ve görüntüleme
- Kullanıcı hesap yönetimi ve kimlik doğrulama
- Alışveriş sepeti işlemleri
- Sipariş verme ve takip sistemi
- Ödeme işlemleri entegrasyonu
- Yönetici paneli ve raporlama
- Makine öğrenmesi tabanlı ürün önerileri

Teknoloji Altyapısı

Ana Framework ve Platform

- **Framework**: ASP.NET Core 8.0
- **Programlama Dili**: C# (.NET 8.0)
- **Mimari**: Model-View-Controller (MVC) Pattern
- **Platform**: Microsoft .NET Ecosystem

Veritabanı Teknolojileri

- **ORM**: Entity Framework Core 8.0
- **Veritabanı**: Microsoft SQL Server
- **Veritabanı Bağlantısı**: SQL Server LocalDB
- **Migration Desteği**: Entity Framework Migrations

Kimlik Doğrulama ve Güvenlik

- **Kimlik Doğrulama**: Cookie-based Authentication
- **Şifreleme**: BCrypt.NET hashing algoritması
- **Çoklu Rol Sistemi**: Kullanıcı ve Admin rolleri
- **Session Yönetimi**: ASP.NET Core Session middleware

Ödeme Entegrasyonu

- **Ödeme Sistemi**: Stripe API
- **Ödeme Güvenliği**: PCI DSS uyumlu ödeme işleme
- **Çoklu Ödeme Durumu**: Başarılı, başarısız, beklemede statüleri

Makine Öğrenmesi ve Veri Analizi

- **ML Framework**: ML.NET 3.0
- **Öneri Sistemi**: ML.NET Recommender Engine
- **Kişiselleştirilmiş Öneriler**: Kullanıcı davranışı tabanlı öneriler
- **Ürün Önerileri**: İlişkili ürün önerileri

Kullanıcı Arayüzü ve Sunum

- **Frontend**: ASP.NET Core MVC Views
- **Template Engine**: Razor Pages
- **Styling**: Bootstrap CSS Framework
- **Responsive Design**: Mobil uyumlu tasarım

Sistem Mimarisi

Katmanlı Mimari Yapısı

1. Sunum Katmanı (Presentation Layer)

- **Controllers**: HTTP isteklerini karşılayan ve iş mantığını yöneten sınıflar
- **Views**: Kullanıcı arayüzü şablonları (Razor Pages)
- **Models**: Veri transfer nesneleri ve view modelleri

2. İş Mantığı Katmanı (Business Logic Layer)

- **Services**: İş kurallarını ve mantığını içeren servis sınıfları
- **Recommendation Service**: ML.NET tabanlı öneri sistemi
- **Authentication Logic**: Kimlik doğrulama ve yetkilendirme mantığı

3. Veri Erişim Katmanı (Data Access Layer)

- **DbContext**: Entity Framework Core veritabanı bağlamı
- **Models**: Veritabanı entity modelleri
- **Migrations**: Veritabanı şeması değişiklikleri

4. Altyapı Katmanı (Infrastructure Layer)

- **Database**: SQL Server veritabanı
- **External APIs**: Stripe ödeme API'si
- **File Storage**: Statik dosya depolama

Veri Modeli

Ana Veri Yapıları

Kullanıcı Yönetimi

- **Kullanici**: Kullanıcı hesap bilgileri ve rolleri
- **Kimlik Doğrulama**: Cookie-based authentication sistemi

Ürün Yönetimi

- **Urun**: Ürün bilgileri, fiyat, stok ve kategori
- **UrunTipi**: Laptop ve Telefon kategorileri
- **Stok Yönetimi**: Gerçek zamanlı stok takibi

Sipariş Yönetimi

- **Siparis**: Sipariş ana bilgileri ve durumu
- **SiparisUrun**: Sipariş detay öğeleri
- **SiparisDurumu**: Sipariş durum takibi

Sepet Yönetimi

- **Sepet**: Kullanıcı alışveriş sepeti
- **SepetUrun**: Sepet öğeleri ve miktarları

Özellikler ve Fonksiyonaliteler

Kullanıcı Özellikleri

- Hesap oluşturma ve giriş yapma
- Ürün katalogunu görüntüleme ve filtreleme
- Ürün detay sayfalarını inceleme
- Alışveriş sepetine ürün ekleme
- Sipariş verme ve ödeme yapma
- Sipariş geçmişi görüntüleme
- Kişiselleştirilmiş ürün önerileri

Yönetici Özellikleri

- Yönetici paneli erişimi
- Ürün ekleme, düzenleme ve silme
- Stok yönetimi
- Sipariş yönetimi ve durum güncellemeleri
- Kullanıcı yönetimi
- Satış raporları ve analizler

Teknik Özellikler

- Responsive web tasarımı
- Güvenli ödeme işlemleri

- Makine öğrenmesi tabanlı öneriler
- Gerçek zamanlı stok takibi
- Çoklu dil desteği (Türkçe)
- SEO uyumlu URL yapısı

Güvenlik Önlemleri

Kimlik Doğrulama Güvenliği

- Güçlü şifre politikaları
- BCrypt hash algoritması kullanımı
- Session timeout yönetimi
- Cross-site request forgery (CSRF) koruması

Veri Güvenliği

- SQL injection koruması (Entity Framework)
- XSS (Cross-site scripting) koruması
- Hassas verilerin şifrelenmesi
- Güvenli ödeme işlemleri (Stripe)

Performans ve Ölçeklenebilirlik

Performans Optimizasyonları

- Entity Framework Core query optimizasyonu
- Lazy loading ve eager loading stratejileri
- Caching mekanizmaları
- Asenkron programlama (async/await)

Ölçeklenebilirlik

- Mikroservis mimarisine geçiş potansiyeli
- Yük dengeleme desteği
- Veritabanı optimizasyonu
- CDN entegrasyonu hazırlığı

Kurulum ve Dağıtım

Sistem Gereksinimleri

- .NET 8.0 Runtime
- SQL Server veya SQL Server LocalDB
- Visual Studio 2022 veya VS Code
- Internet Explorer 11+ veya modern tarayıcılar

Yapılandırma

- Connection string ayarları
- Stripe API anahtarları

- Email servisi yapılandırması
- Logging seviye ayarları

Sonuç

Bu e-ticaret platformu, modern web geliştirme teknolojilerini kullanarak oluşturulmuş, ölçeklenebilir ve güvenli bir çözümdür. Makine öğrenmesi entegrasyonu ile kişiselleştirilmiş alışveriş deneyimi sunarken, güçlü yönetim araçları ile işletmelerin dijital dönüşümüne katkı sağlamaktadır. Platform, sürekli geliştirme ve güncellemelere açık bir yapıda tasarlanmış olup, gelecekteki teknolojik değişimlere uyum sağlayabilecek esnekliğe sahiptir.

ETicaret Sitesi Backend İnceleme Raporu - Bölüm 1

Proje Genel Bakışı

ETicaret Sitesi Projesi - Genel Bakış

Proje Nedir?

Bu proje, **ASP.NET Core 8.0** tabanlı, modern bir **e-ticaret web sitesi** uygulamasıdır. Laptop ve telefon satışı yapan bir platform olarak tasarlanmış ve **yapay zeka destekli ürün önerisi** sistemi bulunmaktadır.

Kullanılan Ana Teknolojiler:

1. Backend Framework:

- **ASP.NET Core 8.0** (Web API + MVC)
- **Entity Framework Core 8.0** (ORM - Object Relational Mapping)
- **SQL Server** (Veritabanı)

2. Kimlik Doğrulama:

- **Cookie Authentication** (Çerez tabanlı kimlik doğrulama)
- **Dual Authentication Scheme** (Kullanıcı ve Admin için ayrı sistemler)

3. Yapay Zeka:

- **ML.NET 3.0.1** (Microsoft'un makine öğrenmesi framework'ü)
- **Collaborative Filtering** (İşbirlikçi filtreleme algoritması)
- **Recommendation Engine** (Ürün öneri sistemi)

4. Güvenlik:

- **BCrypt.Net** (Şifre hash'leme)
- **HTTPS Enforcement** (Zorunlu şifreli bağlantı)
- **CSRF Protection** (Cross-Site Request Forgery koruması)

5. Ödeme Sistemi:

- **Stripe.NET** (Kredi kartı ödemesi altyapısı hazır)

---

1. PROGRAM.CS - Uygulamanın Başlangıç Noktası

USING STATEMENTS (1-4. Satırlar)

```
using ETicaretSitesi.Data;                    // Veritabanı context
sınıfları
```

```
using ETicaretSitesi.Services;                    // İş mantığı servisleri
using Microsoft.EntityFrameworkCore;              // ORM framework
using Microsoft.AspNetCore.Authentication.Cookies; // Çerez tabanlı kimlik
doğrulama
```

Ne İşe Yarar:

- **Namespace importing** - Harici kütüphaneleri projeye dahil eder
- **ETicaretSitesi.Data**: Kendi yazdığımız veritabanı sınıflarını kullanmak için
- **ETicaretSitesi.Services**: Recommendation servisi gibi business logic'i kullanmak için
- **Microsoft.EntityFrameworkCore**: SQL Server ile iletişim için
- **Microsoft.AspNetCore.Authentication.Cookies**: Cookie authentication için

🏗️ WEB APPLICATION BUILDER (6. Satır)

```
var builder = WebApplication.CreateBuilder(args);
```

Builder Pattern:

- **Configuration system**: appsettings.json'dan ayarları okur
- **Dependency Injection Container**: Servisleri kayıt etmek için
- **Logging provider**: Konsol, debug, dosya logging için
- **Environment awareness**: Development/Production ortam ayrımı

MVC SERVİSİ KAYDI (8. Satır)

```
builder.Services.AddControllersWithViews();
```

MVC Pattern Support:

- **Controllers**: HTTP isteklerini işleyen sınıflar
- **Views**: Razor view engine desteği
- **Model Binding**: Form data'sını otomatik model'e dönüştürme
- **Action Filters**: Cross-cutting concerns (logging, authorization vs.)

VERİTABANI CONTEXT KAYDI (10-12. Satırlar)

```
builder.Services.AddDbContext<ETicaretDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultCon
nection")));
```

Entity Framework Configuration:

- **DbContext**: Veritabanı operasyonları için merkezi sınıf
- **SQL Server Provider**: Microsoft SQL Server bağlantısı
- **Connection String**: appsettings.json'dan bağlantı bilgisi
- **Dependency Injection**: Controller'larda constructor injection ile kullanım

Connection String Analizi:

```
"DefaultConnection": "Server=
(localdb)\\mssqllocaldb;Database=ETicaretSitesi;Trusted_Connection=True;Mu
ltipleActiveResultSets=true"
```

- **Server**: LocalDB instance (development için)
- **Database**: ETicaretSitesi veritabanı
- **Trusted_Connection**: Windows Authentication
- **MultipleActiveResultSets**: Aynı connection'da birden fazla sorgu

MACHINE LEARNING SERVİSİ (14. Satır)

```
builder.Services.AddScoped<IRecommendationService, RecommendationService>
();
```

Service Lifetime - Scoped:

- **HTTP Request bazlı**: Her HTTP isteği için yeni instance
- **Thread-safe**: Aynı request içinde aynı instance kullanılır
- **Dispose**: Request bitiminde otomatik olarak dispose edilir
- **Interface-based**: Dependency Inversion Principle

KİMLİK DOĞRULAMA SİSTEMİ (16-32. Satırlar)

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.Authentica
tionScheme)
    .AddCookie(CookieAuthenticationDefaults.AuthenticationScheme, options
=>
    {
        options.LoginPath = "/Kullanici/Giris";
        options.LogoutPath = "/Kullanici/Cikis";
        options.AccessDeniedPath = "/Kullanici/Giris";
        options.ExpireTimeSpan = TimeSpan.FromHours(24);
        options.SlidingExpiration = true;
        options.Cookie.Name = "UserAuth";
    })
    .AddCookie("AdminScheme", options =>
    {
        options.LoginPath = "/Kullanici/AdminGiris";
        options.LogoutPath = "/Kullanici/AdminCikis";
        options.AccessDeniedPath = "/Kullanici/AdminGiris";
```

```
        options.ExpireTimeSpan = TimeSpan.FromHours(8);
        options.SlidingExpiration = true;
        options.Cookie.Name = "AdminAuth";
    });
```

İki Farklı Authentication Scheme:

Normal Kullanıcı Scheme:

- **24 saat oturum süresi**: Uzun süreli kullanım için
- **Sliding Expiration**: Aktivite ile oturum uzar
- **Cookie Name**: "UserAuth"
- **Login Path**: `/Kullanici/Giris`

Admin Scheme:

- **8 saat oturum süresi**: Güvenlik için daha kısa
- **Ayrı login sayfası**: Admin özel giriş
- **Cookie Name**: "AdminAuth"
- **Farklı paths**: Admin-specific URL'ler

Sliding Expiration Nedir:

- Kullanıcı aktif olduğu sürece oturum süresi uzar
- Son aktiviteden itibaren süre sayılır
- Automatic logout prevention

SESSION YÖNETİMİ (34-40. Satırlar)

```
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});
```

Session Configuration:

- **Distributed Memory Cache**: Session verilerini memory'de tutar
- **30 dakika timeout**: İşlem yoksa session timeout
- **HttpOnly Cookie**: JavaScript erişimi engellenir (XSS koruması)
- **Essential Cookie**: GDPR uyumluluğu için gerekli cookie

---

2. MODELS KLASÖRÜ - VERİ YAPILARI ANALİZİ

Model Kategorileri:

1. **Core Entity Models** (Ana veri modelleri)
2. **View Models** (Görünüm için özel modeller)
3. **Enums** (Sabit değer listeleri)
4. **ML Models** (Yapay zeka modelleri)
5. **Helper Classes** (Yardımcı sınıflar)

KULLANICI.CS - Kullanıcı Veri Modeli

Using Statement (1. Satır)

```
using System.ComponentModel.DataAnnotations;
```

Ne İşe Yarar:

- **Data Annotations** - Veri doğrulama attribute'ları için
- `[Required], [StringLength], [EmailAddress]` gibi validasyon attribute'larını kullanabilmek için

Primary Key (7-8. Satırlar)

```
[Key]
public int Id { get; set; }
```

Detaylı Açıklama:

- **[Key]**: Bu property'nin **Primary Key** olduğunu belirtir
- **int Id**: Benzersiz kimlik numarası
- **{ get; set; }**: Auto-property syntax (getter/setter otomatik)
- Entity Framework bu Id'yi **otomatik increment** yapar

Ad Property (10-13. Satırlar)

```
[Required(ErrorMessage = "Ad alanı zorunludur.")]
[StringLength(50, ErrorMessage = "Ad en fazla 50 karakter olabilir.")]
[Display(Name = "Ad")]
public string Ad { get; set; } = string.Empty;
```

Validation Attributes:

- **[Required]**: Boş geçilemez alan
- **[StringLength(50)]**: Maksimum karakter sayısı
- **[Display(Name = "Ad")]**: UI'da gösterilecek label text
- **= string.Empty**: Null reference exception'ı önler

Email Validation (22-26. Satırlar)

```
[Required(ErrorMessage = "E-posta alanı zorunludur.")]
[EmailAddress(ErrorMessage = "Geçerli bir e-posta adresi giriniz.")]
[Display(Name = "E-posta")]
public string Email { get; set; } = string.Empty;
```

EmailAddress Attribute:

- **Built-in validation**: Regex pattern ile email format kontrolü
- **Client-side validation**: JavaScript ile immediate feedback
- **Server-side validation**: Model binding sırasında kontrol

Şifre Güvenliği (28-31. Satırlar)

```
[Required(ErrorMessage = "Şifre alanı zorunludur.")]
[StringLength(100, ErrorMessage = "Şifre en fazla 100 karakter
olabilir.")]
[Display(Name = "Şifre")]
public string Sifre { get; set; } = string.Empty;
```

Şifre Best Practices:

- **100 karakter limit**: Hash'lenmiş şifre için yeterli alan
- **BCrypt hash**: SifreHelper sınıfı ile güvenli hash'leme
- **Plain text asla saklanmaz**: Database'de hash değeri tutulur

Audit Fields (51-61. Satırlar)

```
[Display(Name = "Kayıt Tarihi")]
public DateTime KayitTarihi { get; set; } = DateTime.Now;

[Display(Name = "Son Giriş Tarihi")]
public DateTime? SonGirisTarihi { get; set; }
```

Audit Trail:

- **KayitTarihi**: Kullanıcı ne zaman kayıt oldu
- **SonGirisTarihi**: En son ne zaman giriş yaptı (nullable)
- **Automatic timestamps**: Business logic ile otomatik doldurulur

URUN.CS - Ürün Veri Modeli

Enum Tanımı (5-10. Satırlar)

```
public enum UrunTipi
{
    Telefon,
    Laptop
}
```

Enum Nedir:

- **Sabit değerler listesi** - sadece belirli değerler alabilir
- **Type Safety** sağlar - yanlış değer atanmasını engeller
- Veritabanında **integer** olarak saklanır (Telefon=0, Laptop=1)

Neden Enum:

- **Magic numbers** kullanmak yerine anlamlı isimler
- **Intellisense** desteği - IDE otomatik tamamlama
- **Refactoring** kolaylığı
- **Validation** otomatik olarak sağlanır

Fiyat Property (26-29. Satırlar)

```
[Required(ErrorMessage = "Fiyat zorunludur.")]
[Range(0.01, double.MaxValue, ErrorMessage = "Fiyat 0'dan büyük
olmalıdır.")]
[Display(Name = "Fiyat")]
public decimal Fiyat { get; set; }
```

Decimal vs Double:

- **decimal**: Para hesaplamaları için precise
- **Range validation**: Negatif fiyat engellenir
- **Currency calculations**: Floating point hataları olmaz

Stok Yönetimi (31-35. Satırlar)

```
[Required(ErrorMessage = "Stok miktarı zorunludur.")]
[Range(0, int.MaxValue, ErrorMessage = "Stok miktarı 0'dan küçük
olamaz.")]
[Display(Name = "Stok Miktarı")]
public int StokMiktari { get; set; }
```

Inventory Management:

- **Non-negative validation**: Negatif stok engellenir
- **Business rules**: Stok kontrolü ile satış sınırlaması
- **Real-time updates**: Sipariş sonrası stok düşürülür

SEPET.CS - Alışveriş Sepeti Modeli

Foreign Key (11-12. Satırlar)

```
[Required]
public int KullaniciId { get; set; }
```

Foreign Key Nedir:

- **Başka tablonun Primary Key'ini** referans eder
- Bu sepet hangi kullanıcıya ait olduğunu belirtir
- **Referential Integrity** sağlar - olmayan kullanıcıya sepet atanamaz

Navigation Property (22-23. Satırlar)

```
[ForeignKey("KullaniciId")]
public virtual Kullanici Kullanici { get; set; } = null!;
```

[ForeignKey("KullaniciId")]:

- Bu property'nin **KullaniciId** alanını referans ettiğini belirtir
- Entity Framework **otomatik** olarak bunu çıkarabilir ama açık yazmak daha iyi

= null!:

- **Null-forgiving operator** (!)
- Derleyiciye "bu null olmayacak" garantisi verir
- **Navigation property** çünkü EF Core tarafından doldurulacak

virtual keyword:

- **Lazy loading** için gerekli
- Proxy object'ler için virtual method override
- **Performance optimization** - ihtiyaç duyulduğunda load edilir

SEPET-URUN İLİŞKİSİ - SepetUrun.cs

Many-to-Many İlişki Tablosu

Bu model **Junction Table** (ara tablo) görevi görür:

- **Sepet** ↔ **Urun** arasında Many-to-Many ilişki
- Bir sepette birden fazla ürün olabilir
- Bir ürün birden fazla sepette olabilir

Adet ve Fiyat (20-28. Satırlar)

```
[Required]
[Range(1, int.MaxValue, ErrorMessage = "Adet 1'den küçük olamaz.")]
[Display(Name = "Adet")]
public int Adet { get; set; }

[Required]
[Column(TypeName = "decimal(18,2)")]
[Display(Name = "Birim Fiyat")]
public decimal BirimFiyat { get; set; }
```

Neden BirimFiyat Ayrı:

- **Ürün fiyatı değişebilir** ama sepetteki fiyat sabit kalmalı
- **Price History** - fiyat geçmişi tutulmuş olur
- Kullanıcı sepete eklediği andaki fiyat korunur
- **Dynamic pricing** için kritik

[Column(TypeName = "decimal(18,2)")]:

- **SQL Server** specific precision
- **18 digits total**: 16 tamsayı + 2 ondalık
- **Money calculations**: Para hesaplamaları için ideal

SİPARİŞ MODELİ - Siparis.cs

Sipariş Numarası (17-18. Satırlar)

```
[Display(Name = "Sipariş Numarası")]
public string SiparisNumarasi { get; set; } = Guid.NewGuid().ToString();
```

Guid.NewGuid().ToString():

- **Globally Unique Identifier** oluşturur
- **128-bit** benzersiz kimlik
- Örnek: "f47ac10b-58cc-4372-a567-0e02b2c3d479"
- **Collision** riski pratikte sıfır

Neden Guid:

- **Sequential ID** tahmin edilebilir (güvenlik riski)
- **Distributed systems** için ideal
- **Merge conflicts** yok (farklı serverlardan gelen siparişler)

Decimal Precision (25-27. Satırlar)

```
[Required]
[Display(Name = "Toplam Tutar")]
```

```
[Column(TypeName = "decimal(18,2)")]
public decimal ToplamTutar { get; set; }
```

Financial Data Type:

- **Exact precision**: Floating point hataları yok
- **Currency safe**: Para hesaplamaları için güvenli
- **Database mapping**: SQL Server MONEY type'a karşılık

ENUM MODELS - Durum Yönetimi

OdemeDurumu Workflow

```
Beklemede → Odendi → IadeEdildi
    ↓
IptalEdildi
```

SiparisDurumu Workflow

```
Beklemede → Onaylandi → Kargoda → Tamamlandi
    ↓
IptalEdildi
```

Enum Values:

- **Default değer**: Her enum'ın ilk değeri (0)
- **Database storage**: Integer olarak saklanır
- **Type safety**: Yanlış değer ataması imkansız

VIEW MODELS - UI İçin Özel Veri Modelleri

View Model Nedir?

View Model Pattern:

- **UI'ya özgü** veri yapıları
- **Entity Models**'ı direkt UI'da kullanmak yerine özel modeller
- **Data Transfer Object (DTO)** gibi çalışır
- **Separation of Concerns** sağlar

🛒 SEPET VIEW MODELS

1. SepetEkleModel.cs - Sepete Ekleme Modeli

```
public class SepetEkleModel
{
    public int UrunId { get; set; }      // Hangi ürün
    public int Adet { get; set; } = 1;   // Kaç adet (default 1)
}
```

Bu Model Neden Gerekli:

- **Sadece gerekli alanlar** - UI'da sadece bu 2 bilgi lazım
- **Entity Model** çok fazla alan içerir (Ad, Açıklama, Fiyat vs.)
- **HTTP POST** request'inde gereksiz data transfer olmaz
- **Validation** sadece gerekli alanlar için

2. SepetIndexViewModel.cs - Sepet Sayfası Modeli

```
public class SepetIndexViewModel
{
    public List<SepetUrun> SepetUrunleri { get; set; } = new
List<SepetUrun>();
    public List<Urun> OnerilenUrunler { get; set; } = new List<Urun>();
    public decimal ToplamTutar { get; set; }
    public int ToplamUrunSayisi { get; set; }
}
```

Bu Model'in Amacı:

- **Sepet sayfasında** tüm bilgileri tek seferde göndermek
- **Sepetteki ürünler** + **AI önerileri** + **Toplam bilgiler**
- **Multiple queries** yerine tek ViewModel

ML MODELS - Yapay Zeka Veri Modelleri

ProductRecommendationInput.cs

```
using Microsoft.ML.Data;

public class ProductRecommendationInput
{
    [KeyType(count: 10000)]
    public uint UserId { get; set; }       // Kullanıcı ID (uint = pozitif
tam sayı)

    [KeyType(count: 10000)]
    public uint ProductId { get; set; }    // Ürün ID

    public float Rating { get; set; }      // Kullanıcının ürüne verdiği
```

```
puan
}
```

ML.NET Attributes Açıklaması:

[KeyType(count: 10000)]:

- Bu field'ın **categorical key** olduğunu belirtir
- **count: 10000**: Maximum 10.000 farklı değer olabileceğini söyler
- **Sparse matrix optimization** için kullanılır
- **Memory efficiency** sağlar

uint (Unsigned Integer):

- **32-bit pozitif tam sayı** (0 ile 4,294,967,295 arası)
- **Memory efficient**: ML algoritmaları için optimal
- **No negative IDs**: ID'ler negatif olamaz constraint'i

ProductRecommendationOutput.cs

```
public class ProductRecommendationOutput
{
    public float Score { get; set; }  // Tahmin edilen puan
}
```

Score Değeri:

- **Predicted rating** - kullanıcının bu ürüne vereceği tahmini puan
- **0.0 - 5.0** arası değer
- **Yüksek score** = daha çok beğenilecek ürün
- **Recommendation ranking** için kullanılır

HELPER CLASSES - Yardımcı Sınıflar

SifreHelper.cs - Şifre Güvenlik Yardımcısı

```
using System.Security.Cryptography;
using System.Text;

public static class SifreHelper
{
    public static string HashSifre(string sifre)
    {
        return BCrypt.Net.BCrypt.HashPassword(sifre);
    }

    public static bool SifreDogrula(string girilenSifre, string hashSifre)
    {
```

```
            try
            {
                return BCrypt.Net.BCrypt.Verify(girilenSifre, hashSifre);
            }
            catch
            {
                return false;  // Hata durumunda güvenli değer döndür
            }
        }
    }
```

BCrypt Nedir:

- **Adaptive hashing function** - hash süresini ayarlayabilir
- **Salt automatic** - her hash için farklı salt
- **Time-tested** - dünya çapında kullanılan güvenli algoritma
- **Brute force resistant** - computationally expensive

Static Class Avantajları:

- **No instantiation** - new SifreHelper() gerekmez
- **Utility pattern** - sadece yardımcı metotlar
- **Memory efficient** - instance overhead yok
- **Thread-safe** - stateless operations

Bu ilk bölümde proje genel bakışı, Program.cs detaylı analizi ve Models klasörünün tamamen incelemesini tamamladık. Devam eden bölümlerde Data Layer, Services, Controllers ve diğer backend bileşenlerini ekleyeceğim.

Bölüm 1 Özeti:

- Proje teknoloji stack'i
- Program.cs startup konfigürasyonu
- Authentication & Session yönetimi
- Entity models detaylı analizi
- View models ve ML models
- Helper classes ve güvenlik

ETicaret Sitesi Backend İnceleme Raporu - Bölüm 2

DATA LAYER - VERİTABANI KATMANI

Data Layer Nedir?

Data Access Layer (DAL):

- **Veritabanı işlemlerini** yöneten katman
- **Entity Framework Core** ORM kullanır
- **CRUD operations** (Create, Read, Update, Delete)
- **Database migrations** ve **seed data** yönetimi

ETicaretDbContext.cs - Veritabanı Context Sınıfı

Using Statements ve Namespace (1-4. Satırlar)

```
using Microsoft.EntityFrameworkCore;  // EF Core framework
using ETicaretSitesi.Models;          // Model sınıflarımız

namespace ETicaretSitesi.Data
```

Microsoft.EntityFrameworkCore:

- **ORM (Object-Relational Mapping)** framework
- **Database operations** için temel sınıflar
- **DbContext, DbSet, ModelBuilder** sınıfları

DbContext Sınıfı Tanımı (6-10. Satırlar)

```
public class ETicaretDbContext : DbContext
{
    public ETicaretDbContext(DbContextOptions<ETicaretDbContext> options)
: base(options)
    {
    }
}
```

DbContext Nedir:

- **Entity Framework Core'un kalbi**
- **Database session** - veritabanı ile tek oturum
- **Change tracking** - nesne değişikliklerini takip eder
- **Unit of Work pattern** - transaction yönetimi

Constructor Dependency Injection:

- **DbContextOptions**: Connection string ve provider bilgisi
- **Generic type**: ETicaretDbContext'e özel seçenekler
- **Base constructor**: Microsoft.EntityFrameworkCore.DbContext'e geçer

DbSet Property'leri (12-19. Satırlar)

```
public DbSet<Kullanici> Kullanicilar { get; set; }
public DbSet<Urun> Urunler { get; set; }
public DbSet<Sepet> Sepetler { get; set; }
public DbSet<SepetUrun> SepetUrunleri { get; set; }
public DbSet<Siparis> Siparisler { get; set; }
public DbSet<SiparisUrun> SiparisUrunleri { get; set; }
```

DbSet Nedir:

- **Table representation** - her DbSet bir veritabanı tablosunu temsil eder
- **CRUD operations** - Add, Find, Update, Remove metodları
- **LINQ support** - Where, Select, Join operasyonları
- **Change tracking** - nesne durumu değişikliklerini takip

Naming Convention:

- **DbSet property adı** = **Tablo adı** (Kullanicilar → Kullanicilar tablosu)
- **Entity class adı** = **Kayıt türü** (Kullanici → tekil kayıt)
- **Turkish naming** - business domain'e uygun isimlendirme

OnModelCreating - Model Konfigürasyonu (21-197. Satırlar)

Primary Key Definitions (23-29. Satırlar)

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    // Primary key tanımlamaları
    modelBuilder.Entity<Kullanici>().HasKey(k => k.Id);
    modelBuilder.Entity<Urun>().HasKey(u => u.Id);
    modelBuilder.Entity<Sepet>().HasKey(s => s.Id);
    modelBuilder.Entity<SepetUrun>().HasKey(su => su.Id);
    modelBuilder.Entity<Siparis>().HasKey(s => s.Id);
    modelBuilder.Entity<SiparisUrun>().HasKey(su => su.Id);
```

Fluent API Configuration:

- **HasKey()**: Primary key açık olarak belirtir
- **Convention over configuration**: [Key] attribute'dan daha açık
- **Composite keys**: Birden fazla kolonu primary key yapabilir

- **Database agnostic**: Farklı veritabanlarında aynı çalışır

Foreign Key Relationships (31-67. Satırlar)

```
// Foreign key ilişkileri
modelBuilder.Entity<Sepet>()
    .HasOne(s => s.Kullanici)          // Sepet'in bir Kullanıcısı var
    .WithMany(k => k.Sepetler)         // Kullanıcı'nın birden fazla
Sepeti var
    .HasForeignKey(s => s.KullaniciId); // Foreign key alanı

modelBuilder.Entity<SepetUrun>()
    .HasOne(su => su.Kullanici)
    .WithMany(k => k.SepetUrunleri)
    .HasForeignKey(su => su.KullaniciId);

modelBuilder.Entity<SepetUrun>()
    .HasOne(su => su.Urun)
    .WithMany(u => u.SepetUrunleri)
    .HasForeignKey(su => su.UrunId);
```

Relationship Types:

- **One-to-Many**: HasOne().WithMany() - Bir kullanıcının birden fazla sepeti
- **Many-to-Many**: Junction table ile (SepetUrun aracı tablo)
- **Foreign Key**: HasForeignKey() ile açık olarak belirtilir
- **Navigation Properties**: Otomatik lazy loading için

Seed Data - İlk Veriler (69-180. Satırlar)

```
// Admin kullanıcısı seed data
modelBuilder.Entity<Kullanici>().HasData(
    new Kullanici
    {
        Id = 1,
        Ad = "Admin",
        Soyad = "User",
        Email = "admin@example.com",
        Sifre = "PrP+ZrMeO00Q+nC1ytSccRIpSvauTkdqHEBRVdRaoSE=", //
şifreli: admin123
        KullaniciAdi = "admin",
        Telefon = "5551234567",
        Adres = "Admin Adresi",
        AdminMi = true,
        AktifMi = true,
        Rol = "Admin",
        KayitTarihi = DateTime.Now
    });
```

Seed Data Avantajları:

- **Initial data**: Uygulama ilk kez çalıştırıldığında hazır veriler
- **Development productivity**: Test verileri otomatik oluşur
- **Consistent state**: Her environment'ta aynı başlangıç durumu
- **Admin account**: Sistem yönetimi için ilk admin kullanıcısı

Ürün Seed Data Örneği:

```
modelBuilder.Entity<Urun>().HasData(
    new Urun
    {
        Id = 1,
        Ad = "MacBook Pro M2",
        Aciklama = "Apple'ın en güçlü M2 çipi ile donatılmış MacBook
Pro...",
        Fiyat = 42999.99m,
        StokMiktari = 10,
        UrunTipi = UrunTipi.Laptop,
        ResimUrl = "/images/laptop1.jpg",
        GorselUrl = "",
        OlusturmaTarihi = DateTime.Now,
        AktifMi = true
    },
    // ... 9 ürün daha
);
```

DbInitializer.cs - Veritabanı Başlatıcısı

Using Statements (1-4. Satırlar)

```
using ETicaretSitesi.Models;           // Entity sınıfları
using ETicaretSitesi.Models.Helpers;   // SifreHelper sınıfı
using Microsoft.EntityFrameworkCore;   // EF Core extensions
```

Static Class Tanımı (7-8. Satırlar)

```
public static class DbInitializer
{
    public static void Initialize(ETicaretDbContext context)
```

Static Class Nedir:

- **Instance oluşturulamaz** - `new DbInitializer()` yasak
- **Utility functions** için ideal
- **Stateless operations** - state tutmaz

- **Memory efficient** - object allocation yok

Veritabanı Oluşturma (10-11. Satırlar)

```
context.Database.EnsureCreated();
```

EnsureCreated() vs Migrate():

- **EnsureCreated()**: Veritabanı yoksa oluşturur, varsa dokunmaz
- **Migrate()**: Migration'ları çalıştırır, schema güncellemeleri yapar
- **Development**: EnsureCreated() hızlı başlangıç için
- **Production**: Migrate() controlled schema updates için

Admin Kullanıcısı Kontrolü (13-35. Satırlar)

```
// Admin kullanıcısı var mı kontrol et
if (!context.Kullanicilar.Any(k => k.AdminMi))
{
    var adminKullanici = new Kullanici
    {
        Ad = "Admin",
        Soyad = "User",
        Email = "admin@example.com",
        Sifre = SifreHelper.HashSifre("admin123"),  // BCrypt ile
hash'leme
        KullaniciAdi = "admin",
        Telefon = "5551234567",
        Adres = "Admin Adresi",
        AdminMi = true,
        AktifMi = true,
        Rol = "Admin",
        KayitTarihi = DateTime.Now
    };

    context.Kullanicilar.Add(adminKullanici);
    context.SaveChanges();
}
```

Conditional Seeding:

- **Any() check**: Admin kullanıcısı varsa tekrar ekleme
- **BCrypt hashing**: Güvenli şifre saklama
- **Immediate save**: Admin önce eklenir, sonra ürünler

Ürün Kataloğu Oluşturma (37-130. Satırlar)

```

```

```
// Ürünler var mı kontrol et
if (!context.Urunler.Any())
{
    var urunler = new List<Urun>
    {
        new Urun
        {
            Ad = "MacBook Pro M2",
            Aciklama = "Apple'ın en güçlü M2 çipi ile donatılmış...",
            Fiyat = 42999.99m,
            StokMiktari = 10,
            UrunTipi = UrunTipi.Laptop,
            ResimUrl = "/images/laptop1.jpg",
            GorselUrl = "",
            OlusturmaTarihi = DateTime.Now,
            AktifMi = true
        },
        // ... 9 ürün daha
    };

    context.Urunler.AddRange(urunler);
    context.SaveChanges();
}
```

Bulk Insert Pattern:

- **AddRange()**: Birden fazla kaydı tek seferde ekler
- **Single transaction**: Tüm ürünler tek transaction'da
- **Performance**: Tek SQL statement ile bulk insert

SERVICES LAYER - İŞ MANTIK KATMANI

Services Layer Nedir?

Business Logic Layer:

- **İş mantığı** ve **business rules** burada
- **Controllers** ile **Data Layer** arasında köprü
- **Reusable business operations**
- **Dependency Injection** ile kullanılır
- **Unit testing** için ideal katman

IRecommendationService.cs - Interface Tanımı

Interface Pattern

```
public interface IRecommendationService
{
```

```
    // Method signatures only - implementation yok
}
```

Interface Neden Kullanılır:

- **Abstraction** - implementation gizlenir
- **Dependency Inversion** - high-level modules low-level'a bağlı değil
- **Testability** - mock objects oluşturulabilir
- **Flexibility** - farklı implementation'lar swap edilebilir

Method Signatures Analizi

1. GetRecommendationsForUserAsync() - Kullanıcı Bazlı Öneriler

```
Task<List<Urun>> GetRecommendationsForUserAsync(int userId, int count =
5);
```

Method Signature Breakdown:

- **Task**: Asynchronous operation - non-blocking
- **List**: Return type - ürün listesi
- **int userId**: Hangi kullanıcı için öneri
- **int count = 5**: Kaç öneri isteniyor (default 5)

2. GetRecommendationsForProductAsync() - Ürün Bazlı Öneriler

```
Task<List<Urun>> GetRecommendationsForProductAsync(int productId, int
count = 5);
```

Business Logic:

- **"Bu ürüne bakanlar şunları da inceledi"** mantığı
- **Item-based collaborative filtering**
- **Cross-selling** opportunities

RecommendationService.cs - ML.NET Implementation

Using Statements ve Dependencies (1-8. Satırlar)

```
using Microsoft.ML;                    // ML.NET core framework
using Microsoft.ML.Data;               // Data processing attributes
using Microsoft.ML.Trainers;           // Machine learning trainers
using ETicaretSitesi.Models;           // Entity models
using ETicaretSitesi.Models.ML;        // ML-specific models
```

```
using ETicaretSitesi.Data;              // Database context
using Microsoft.EntityFrameworkCore;    // EF Core extensions
```

ML.NET Framework:

- **Microsoft.ML**: Core ML framework
- **Microsoft.ML.Data**: Data annotations ve processing
- **Microsoft.ML.Trainers**: Algoritma implementations

Service Class Definition (10-20. Satırlar)

```
public class RecommendationService : IRecommendationService
{
    private readonly ETicaretDbContext _context;
    private readonly MLContext _mlContext;
    private readonly string _modelPath;
    private ITransformer _model;

    public RecommendationService(ETicaretDbContext context)
    {
        _context = context;
        _mlContext = new MLContext(seed: 0);  // Reproducible results için
        _modelPath = "recommendation_model.zip";
    }
}
```

ML.NET Architecture:

- **MLContext**: ML operations için main entry point
- **seed: 0**: Deterministic results için sabit seed
- **ITransformer**: Trained model interface
- **Model persistence**: File system'de model saklama

Machine Learning Pipeline (TrainModelAsync Method)

Veri Hazırlama Phase:

```
public async Task TrainModelAsync()
{
    // Sipariş verilerinden training data oluştur
    var orderData = await _context.Siparisler
        .Include(s => s.SiparisUrunleri)
        .ThenInclude(su => su.Urun)
        .Where(s => s.OdemeDurumu == OdemeDurumu.Odendi)  // Sadece ödenen
siparişler
        .ToListAsync();

    var trainingData = new List<ProductRecommendationInput>();
```

```
    foreach (var order in orderData)
    {
        foreach (var item in order.SiparisUrunleri)
        {
            trainingData.Add(new ProductRecommendationInput
            {
                UserId = (uint)order.KullaniciId,
                ProductId = (uint)item.UrunId,
                Rating = 5.0f  // Satın alma = 5 star rating
            });
        }
    }
}
```

Training Data Logic:

- **Explicit feedback**: Satın alma davranışı = 5 star rating
- **Implicit feedback**: Sepete ekleme, görüntüleme vs. eklenebilir
- **Paid orders only**: Sadece ödenen siparişler güvenilir data
- **User-Item-Rating triplets**: Collaborative filtering için gerekli format

ML Pipeline Configuration:

```
// ML pipeline oluştur
var pipeline = _mlContext.Transforms.Conversion
    .MapValueToKey(outputColumnName: "userIdEncoded", inputColumnName:
nameof(ProductRecommendationInput.UserId))
    .Append(_mlContext.Transforms.Conversion
        .MapValueToKey(outputColumnName: "productIdEncoded",
inputColumnName: nameof(ProductRecommendationInput.ProductId)))
    .Append(_mlContext.Recommendation().Trainers.MatrixFactorization(
        labelColumnName: nameof(ProductRecommendationInput.Rating),
        matrixColumnIndexColumnName: "userIdEncoded",
        matrixRowIndexColumnName: "productIdEncoded"));
```

Matrix Factorization Algorithm:

- **MapValueToKey**: Categorical data'yı numeric key'lere çevir
- **Matrix Factorization**: User-Item rating matrix'ini decompose eder
- **Latent factors**: Hidden patterns'ları bulur
- **Collaborative filtering**: Benzer kullanıcıların tercihlerini öğrenir

Prediction Methods

User-Based Recommendations:

```
```

```
public async Task<List<Urun>> GetRecommendationsForUserAsync(int userId,
int count = 5)
{
    if (_model == null)
    {
        await LoadModelAsync();  // Model yüklü değilse yükle
    }

    // Kullanıcının daha önce almadığı ürünleri bul
    var userPurchasedProducts = await _context.Siparisler
        .Include(s => s.SiparisUrunleri)
        .Where(s => s.KullaniciId == userId && s.OdemeDurumu ==
OdemeDurumu.Odendi)
        .SelectMany(s => s.SiparisUrunleri.Select(su => su.UrunId))
        .ToListAsync();

    var allProducts = await _context.Urunler
        .Where(u => u.AktifMi && !userPurchasedProducts.Contains(u.Id))
        .ToListAsync();

    // Her ürün için score hesapla
    var recommendations = new List<(Urun Product, float Score)>();
    var predictionEngine =
_mlContext.Model.CreatePredictionEngine<ProductRecommendationInput,
ProductRecommendationOutput>(_model);

    foreach (var product in allProducts)
    {
        var prediction = predictionEngine.Predict(new
ProductRecommendationInput
        {
            UserId = (uint)userId,
            ProductId = (uint)product.Id
        });

        recommendations.Add((product, prediction.Score));
    }

    // Score'a göre sırala ve top N'i döndür
    return recommendations
        .OrderByDescending(r => r.Score)
        .Take(count)
        .Select(r => r.Product)
        .ToList();
}
```

Recommendation Logic:

- **Exclusion filter**: Daha önce satın alınan ürünler hariç
- **Active products only**: Aktif ürünler için öneri
- **Score-based ranking**: ML model'den gelen score'a göre sıralama

- **Top-N filtering**: İstenen sayıda öneri döndür

CONTROLLERS LAYER - API ENDPOINTS

Controllers Nedir?

MVC Controller:

- **HTTP requests** handle eder
- **Business logic** orchestrate eder
- **Models** ile **Views** arasında köprü
- **RESTful API** endpoints sağlar
- **Authentication/Authorization** kontrol eder

HomeController.cs - Ana Sayfa Controller

Using Statements (1-8. Satırlar)

```
using System.Diagnostics;                // Activity.Current for tracing
using Microsoft.AspNetCore.Mvc;          // Controller base class
using ETicaretSitesi.Models;             // Entity models
using ETicaretSitesi.Data;               // Database context
using ETicaretSitesi.Services;           // Recommendation service
using Microsoft.EntityFrameworkCore;     // EF Core extensions
using System.Security.Claims;            // User authentication claims
```

Controller Definition ve Dependency Injection (10-18. Satırlar)

```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly ETicaretDbContext _context;
    private readonly IRecommendationService _recommendationService;

    public HomeController(ILogger<HomeController> logger,
ETicaretDbContext context, IRecommendationService recommendationService)
    {
        _logger = logger;
        _context = context;
        _recommendationService = recommendationService;
    }
}
```

Constructor Injection Pattern:

- **ILogger**: Structured logging için
- **ETicaretDbContext**: Veritabanı operasyonları

- **IRecommendationService**: AI önerileri için

Index Action - Ana Sayfa (20-46. Satırlar)

```
public async Task<IActionResult> Index()
{
    // Tüm aktif ürünleri getir
    var urunler = await _context.Urunler
        .Where(u => u.AktifMi)              // Sadece aktif ürünler
        .OrderByDescending(u => u.Id)       // En yeni ürünler önce
        .ToListAsync();

    // Kullanıcı giriş yapmışsa kişiselleştirilmiş öneriler
    List<Urun> onerilenUrunler = new List<Urun>();

    if (User.Identity.IsAuthenticated)
    {
        var email = User.FindFirstValue(ClaimTypes.Email);
        var kullanici = await _context.Kullanicilar.FirstOrDefaultAsync(k
=> k.Email == email);

        if (kullanici != null)
        {
            try
            {
                onerilenUrunler = await
_recommendationService.GetRecommendationsForUserAsync(kullanici.Id, 4);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Kullanıcı önerileri alınırken hata
oluştu");
                // Hata durumunda rastgele ürünler göster
                onerilenUrunler = urunler.Take(4).ToList();
            }
        }
    }
    else
    {
        // Giriş yapmamış kullanıcılar için popüler ürünler
        onerilenUrunler = urunler.Take(4).ToList();
    }

    ViewBag.OnerilenUrunler = onerilenUrunler;
    return View(urunler);
}
```

Business Logic:

- **Authenticated users**: AI-powered personalized recommendations
- **Anonymous users**: Popular products (newest first)

- **Error handling**: ML service fail durumunda fallback
- **ViewBag**: Önerilen ürünleri view'a geçirme

KullaniciController.cs - Kullanıcı Yönetimi

Normal User Login - Giris Actions (21-71. Satırlar)

GET Action - Login Form Display (21-27. Satırlar)

```
public IActionResult Giris()
{
    if (User.Identity.IsAuthenticated)  // Zaten giriş yapmış mı?
    {
        return RedirectToAction("Index", "Home");  // Ana sayfaya
yönlendir
    }
    return View();  // Login formu göster
}
```

POST Action - Login Processing (29-71. Satırlar)

```
[HttpPost]
[ValidateAntiForgeryToken]              // CSRF saldırılarına karşı koruma
public async Task<IActionResult> Giris(string email, string sifre)
{
    if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(sifre))
    {
        ViewBag.Error = "E-posta ve şifre alanları zorunludur.";
        return View();
    }

    // Kullanıcıyı veritabanında ara
    var kullanici = await _context.Kullanicilar
        .FirstOrDefaultAsync(k => k.Email == email && k.AktifMi);

    if (kullanici != null && SifreHelper.SifreDogrula(sifre,
kullanici.Sifre))
    {
        // Claims oluştur
        var claims = new List<Claim>
        {
            new Claim(ClaimTypes.Email, kullanici.Email),
            new Claim(ClaimTypes.Name, $"{kullanici.Ad}
{kullanici.Soyad}"),
            new Claim("UserId", kullanici.Id.ToString()),
            new Claim(ClaimTypes.Role, kullanici.Rol)
        };

        var claimsIdentity = new ClaimsIdentity(claims,
```

```
CookieAuthenticationDefaults.AuthenticationScheme);
        var authProperties = new AuthenticationProperties
        {
            IsPersistent = true,  // "Beni hatırla" özelliği
            ExpiresUtc = DateTimeOffset.UtcNow.AddHours(24)  // 24 saat
oturum
        };

        await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
            new ClaimsPrincipal(claimsIdentity), authProperties);

        // Son giriş tarihini güncelle
        kullanici.SonGirisTarihi = DateTime.Now;
        await _context.SaveChangesAsync();

        return RedirectToAction("Index", "Home");
    }

    ViewBag.Error = "Geçersiz e-posta veya şifre.";
    return View();
}
```

Authentication Flow:

- **Input validation**: Email ve şifre boş kontrolü
- **User lookup**: Active user check ile database query
- **Password verification**: BCrypt ile hash karşılaştırması
- **Claims creation**: Identity bilgilerini claims'e çevirme
- **Cookie authentication**: SignInAsync ile cookie oluşturma
- **Audit trail**: Son giriş tarihi güncelleme

Admin Login Flow (117-177. Satırlar)

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AdminGiris(string email, string sifre)
{
    if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(sifre))
    {
        ViewBag.Error = "E-posta ve şifre alanları zorunludur.";
        return View();
    }

    var kullanici = await _context.Kullanicilar
        .FirstOrDefaultAsync(k => k.Email == email && k.AktifMi &&
k.AdminMi);  // Admin kontrolü

    if (kullanici != null && SifreHelper.SifreDogrula(sifre,
kullanici.Sifre))
```

```
    {
        var claims = new List<Claim>
        {
            new Claim(ClaimTypes.Email, kullanici.Email),
            new Claim(ClaimTypes.Name, $"{kullanici.Ad}
{kullanici.Soyad}"),
            new Claim("UserId", kullanici.Id.ToString()),
            new Claim(ClaimTypes.Role, "Admin")  // Admin role zorunlu
        };

        var claimsIdentity = new ClaimsIdentity(claims, "AdminScheme");
// Özel admin scheme
        var authProperties = new AuthenticationProperties
        {
            IsPersistent = false,  // Admin oturumu persistent değil
            ExpiresUtc = DateTimeOffset.UtcNow.AddHours(8)  // 8 saat
(daha kısa)
        };

        await HttpContext.SignInAsync("AdminScheme", new
ClaimsPrincipal(claimsIdentity), authProperties);

        kullanici.SonGirisTarihi = DateTime.Now;
        await _context.SaveChangesAsync();

        return RedirectToAction("Index", "Admin");
    }

    ViewBag.Error = "Geçersiz admin bilgileri.";
    return View();
}
```

Admin Authentication Differences:

- **AdminMi = true**: Sadece admin kullanıcılar
- **"AdminScheme"**: Farklı authentication scheme
- **8 hours expiry**: Güvenlik için daha kısa oturum
- **IsPersistent = false**: "Beni hatırla" özelliği yok

Bu ikinci bölümde Data Layer (DbContext, DbInitializer), Services Layer (ML.NET recommendation service) ve Controllers'ın temel kısımlarını inceledik.

Bölüm 2 Özeti:

- Entity Framework Core DbContext analizi
- Database seeding ve initialization
- ML.NET recommendation service architecture
- Authentication flow (normal ve admin)
- Controller dependency injection patterns
- Claims-based authentication implementation

ETicaret Sitesi Backend İnceleme Raporu - Bölüm 3 (Final)

ALIŞVERIŞ CONTROLLER'LARI

SepetController.cs - Alışveriş Sepeti Yönetimi

Controller Architecture (11-17. Satırlar)

```
public class SepetController : Controller
{
    private readonly ETicaretDbContext _context;

    public SepetController(ETicaretDbContext context)
    {
        _context = context;
    }
}
```

Simple Dependency Injection:

- Sadece **database context** gerekiyor
- **RecommendationService** kullanılmıyor - kendi AI logic'i var
- **Lightweight controller** - minimal dependencies

Sepet Index - Ana Sepet Sayfası (21-58. Satırlar)

Authentication Check (21-32. Satırlar)

```
public async Task<IActionResult> Index()
{
    if (!User.Identity.IsAuthenticated)
    {
        return RedirectToAction("Giris", "Kullanici");
    }

    var email = User.FindFirstValue(ClaimTypes.Email);
    var kullanici = await _context.Kullanicilar.FirstOrDefaultAsync(k =>
k.Email == email);

    if (kullanici == null)
    {
        return RedirectToAction("Giris", "Kullanici");
    }
}
```

Double Security Check:

- **User.Identity.IsAuthenticated**: Framework level check
- **Database user lookup**: Kullanıcı hala aktif mi kontrolü
- **Defensive programming**: İki katmanlı güvenlik

Sepet Verilerini Getirme (34-47. Satırlar)

```
    var sepetUrunleri = await _context.SepetUrunleri
        .Include(su => su.Urun)                    // Ürün bilgilerini de
getir
        .Where(su => su.KullaniciId == kullanici.Id)
        .ToListAsync();

    var toplamTutar = sepetUrunleri.Sum(su => su.BirimFiyat * su.Adet);
    var toplamUrunSayisi = sepetUrunleri.Sum(su => su.Adet);

    // AI önerisi sistemi - sepetteki ürünlere göre
    var onerilenUrunler = new List<Urun>();
    if (sepetUrunleri.Any())
    {
        var sepettekiUrunIds = sepetUrunleri.Select(su =>
su.UrunId).ToList();
        // Sepetteki ürünlerle aynı kategorideki diğer ürünler
        var ayniKategoriUrunler = await _context.Urunler
            .Where(u => u.AktifMi &&
                   sepetUrunleri.Any(su => su.Urun.UrunTipi == u.UrunTipi)
&&
                   !sepettekiUrunIds.Contains(u.Id))
            .Take(4)
            .ToListAsync();

        onerilenUrunler = ayniKategoriUrunler;
    }
```

Business Logic Implementation:

- **Include() kullanımı**: Ürün bilgileri tek sorguda gelir (N+1 problem çözümü)
- **Calculated fields**: Toplam tutar ve ürün sayısı runtime'da hesaplanır
- **Simple AI logic**: Aynı kategorideki ürünler öneriliyor
- **Performance optimization**: Take(4) ile sadece 4 öneri

Sepete Ürün Ekleme (70-145. Satırlar)

Input Validation ve User Check (70-85. Satırlar)

```
[HttpPost]
public async Task<IActionResult> Ekle(int urunId, int adet = 1)
{
```

```
    if (!User.Identity.IsAuthenticated)
    {
        return Json(new { success = false, message = "Giriş yapmanız
gerekiyor." });
    }

    var email = User.FindFirstValue(ClaimTypes.Email);
    var kullanici = await _context.Kullanicilar.FirstOrDefaultAsync(k =>
k.Email == email);

    if (kullanici == null)
    {
        return Json(new { success = false, message = "Kullanıcı
bulunamadı." });
    }
}

**AJAX API Pattern:**
- **JSON response**: Frontend JavaScript ile uyumlu
- **Error messaging**: User-friendly hata mesajları
- **Authentication check**: API seviyesinde güvenlik

##### Ürün Validation (87-99. Satırlar)
```csharp
    var urun = await _context.Urunler.FindAsync(urunId);
    if (urun == null || !urun.AktifMi)
    {
        return Json(new { success = false, message = "Ürün bulunamadı."
});
    }

    if (urun.StokMiktari < adet)
    {
        return Json(new { success = false, message = "Yeterli stok
bulunmuyor." });
    }

    if (adet <= 0)
    {
        return Json(new { success = false, message = "Geçersiz miktar."
});
    }
```

Business Rules Implementation:

- **Product availability**: Ürün aktif mi kontrolü
- **Stock validation**: Stok yeterli mi kontrolü
- **Quantity validation**: Pozitif miktar kontrolü

Sepet Logic - Var Olan vs Yeni Ürün (101-125. Satırlar)

```
```
```

```
    // Kullanıcının sepetinde bu ürün var mı kontrol et
    var mevcutSepetUrun = await _context.SepetUrunleri
        .FirstOrDefaultAsync(su => su.KullaniciId == kullanici.Id &&
su.UrunId == urunId);

    if (mevcutSepetUrun != null)
    {
        // Mevcut ürünün adedini artır
        mevcutSepetUrun.Adet += adet;
        mevcutSepetUrun.BirimFiyat = urun.Fiyat; // Güncel fiyatı al
    }
    else
    {
        // Yeni ürün ekle
        var yeniSepetUrun = new SepetUrun
        {
            KullaniciId = kullanici.Id,
            UrunId = urunId,
            Adet = adet,
            BirimFiyat = urun.Fiyat,
            EklenmeTarihi = DateTime.Now
        };
        _context.SepetUrunleri.Add(yeniSepetUrun);
    }

    await _context.SaveChangesAsync();
```

Smart Cart Logic:

- **Duplicate check**: Aynı ürün tekrar eklenirse adedi artırılır
- **Price update**: Mevcut ürünlerde güncel fiyat kullanılır
- **Audit trail**: EklenmeTarihi ile zaman damgası

SiparisController.cs - Sipariş Yönetimi

Controller-Level Authorization (10. Satır)

```
[Authorize]
public class SiparisController : Controller
```

Global Authorization:

- **Tüm actions** için authentication gerekli
- **Method-level override** mümkün değil
- **Security-first approach** - tüm sipariş işlemleri korumalı

Order History - Index Action (20-35. Satırlar)

```
public async Task<IActionResult> Index()
{
    var email = User.FindFirstValue(ClaimTypes.Email);
    var kullanici = await _context.Kullanicilar.FirstOrDefaultAsync(k =>
k.Email == email);

    var siparisler = await _context.Siparisler
        .Include(s => s.SiparisUrunleri)
// JOIN order items
        .ThenInclude(su => su.Urun)
// JOIN products
        .Where(s => s.KullaniciId == kullanici.Id)
// Sadece kullanıcının siparişleri
        .OrderByDescending(s => s.SiparisTarihi)
// En yeni önce
        .ToListAsync();

    return View(siparisler);
}
```

Data Loading Strategy:

- **Include() + ThenInclude()**: İç içe geçmiş veriler tek sorguda
- **User filtering**: Kullanıcı sadece kendi siparişlerini görür
- **Chronological ordering**: En yeni siparişler önce

Sipariş Oluşturma - Olustur Action (37-125. Satırlar)

Sepet Validation (37-55. Satırlar)

```
public async Task<IActionResult> Olustur()
{
    var email = User.FindFirstValue(ClaimTypes.Email);
    var kullanici = await _context.Kullanicilar.FirstOrDefaultAsync(k =>
k.Email == email);

    var sepetUrunleri = await _context.SepetUrunleri
        .Include(su => su.Urun)
        .Where(su => su.KullaniciId == kullanici.Id)
        .ToListAsync();

    if (!sepetUrunleri.Any())
    {
        TempData["Error"] = "Sepetinizde ürün bulunmuyor.";
        return RedirectToAction("Index", "Sepet");
    }

    var toplamTutar = sepetUrunleri.Sum(su => su.BirimFiyat * su.Adet);

    return View(new SiparisOlusturViewModel
```

```
    {
        SepetUrunleri = sepetUrunleri,
        ToplamTutar = toplamTutar
    });
}
```

Order Creation Flow:

- **Empty cart check**: Boş sepet kontrolü
- **ViewModel usage**: UI için özel veri yapısı
- **Price calculation**: Güncel toplam tutar hesabı

Order Processing - POST Action (127-220. Satırlar)

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Olustur(SiparisOlusturViewModel model)
{
    if (!ModelState.IsValid)
    {
        // Model validation başarısız
        var sepetUrunleri = await _context.SepetUrunleri
            .Include(su => su.Urun)
            .Where(su => su.KullaniciId == kullanici.Id)
            .ToListAsync();

        model.SepetUrunleri = sepetUrunleri;
        model.ToplamTutar = sepetUrunleri.Sum(su => su.BirimFiyat *
su.Adet);
        return View(model);
    }

    // Sipariş oluştur
    var siparis = new Siparis
    {
        KullaniciId = kullanici.Id,
        SiparisNumarasi = Guid.NewGuid().ToString(),
        SiparisTarihi = DateTime.Now,
        ToplamTutar = sepetUrunleri.Sum(su => su.BirimFiyat * su.Adet),
        Adres = model.Adres,
        Telefon = model.Telefon,
        OdemeDurumu = OdemeDurumu.Beklemede,
        SiparisDurumu = SiparisDurumu.Beklemede
    };

    _context.Siparisler.Add(siparis);
    await _context.SaveChangesAsync();

    // Sipariş ürünlerini oluştur
    foreach (var sepetUrun in sepetUrunleri)
    {
```

```
            var siparisUrun = new SiparisUrun
            {
                SiparisId = siparis.Id,
                UrunId = sepetUrun.UrunId,
                Adet = sepetUrun.Adet,
                BirimFiyat = sepetUrun.BirimFiyat
            };
            _context.SiparisUrunleri.Add(siparisUrun);
        }

        // Sepeti temizle
        _context.SepetUrunleri.RemoveRange(sepetUrunleri);
        await _context.SaveChangesAsync();

        TempData["Success"] = "Siparişiniz başarıyla oluşturuldu.";
        return RedirectToAction("Index");
    }
```

Transaction Management:

- **Model validation**: Form validation kontrolü
- **GUID order number**: Benzersiz sipariş numarası
- **Atomic transaction**: Sipariş + ürünler + sepet temizleme
- **Default statuses**: Beklemede durumları ile başlar

ADMİN CONTROLLER - YÖNETİM PANELİ

Admin Authorization & Dependencies (14-25. Satırlar)

```
[Authorize(AuthenticationSchemes = "AdminScheme")]     // Sadece admin şeması ile giriş yapmış kullanıcılar
public class AdminController : Controller
{
    private readonly ETicaretDbContext _context;                    // Veritabanı erişimi
    private readonly IWebHostEnvironment _webHostEnvironment;       // Dosya yükleme için web ortam bilgisi
    private readonly IRecommendationService _recommendationService; // Yapay zeka önerisi servisi
}
```

Özel Authorization Şeması:

- **"AdminScheme"**: Program.cs'te tanımlanan özel admin authentication scheme
- **Ayrı cookie sistemi**: Normal kullanıcılardan farklı oturum yönetimi
- **Daha kısa süre**: Admin oturumları 8 saat (normal kullanıcı 24 saat)

Admin Dashboard - Index Action (27-63. Satırlar)

Güvenlik Kontrolü (29-33. Satırlar)

```
public async Task<IActionResult> Index()
{
    // Admin kontrolü
    if (!User.IsInRole("Admin"))
    {
        return RedirectToAction("AdminGiris", "Kullanici");
    }
```

Çift Güvenlik Katmanı:

- **[Authorize(AuthenticationSchemes = "AdminScheme")]**: Controller seviyesi
- **User.IsInRole("Admin")**: Action seviyesi ek kontrol
- **Savunma derinliği**: Birden fazla güvenlik kontrolü

Dashboard Verilerini Toplama (35-55. Satırlar)

```
    var siparisler = await _context.Siparisler
        .Include(s => s.Kullanici)                          // JOIN ile
kullanıcı bilgilerini getir
        .OrderByDescending(s => s.SiparisTarihi)            // En yeni
siparişler önce
        .Take(10)                                           // Sadece 10
tanesini al
        .ToListAsync();

    var toplamGelir = await _context.Siparisler
        .Where(s => s.OdemeDurumu == OdemeDurumu.Odendi)    // Sadece
ödenen siparişler
        .SumAsync(s => s.ToplamTutar);                      // Toplam
tutarları topla
```

Dashboard Performance Strategy:

- **Take(10)**: Sadece gerekli miktarda veri
- **OrderByDescending**: En güncel veriler önce
- **Include()**: Efficient JOIN operations
- **SumAsync()**: Veritabanı seviyesinde toplam hesaplama

Ürün Yönetimi - File Upload Implementation

Dosya Yükleme İşlemi (82-99. Satırlar)

```
[HttpPost]
[ValidateAntiForgeryToken]                                  // CSRF koruması
public async Task<IActionResult> UrunEkle(Urun urun, IFormFile? gorsel)
```

```
{
    if (ModelState.IsValid)
    {
        // Görsel yükleme işlemi
        if (gorsel != null && gorsel.Length > 0)
        {
            var fileName =
Path.GetFileNameWithoutExtension(gorsel.FileName);        // Dosya adı
            var extension = Path.GetExtension(gorsel.FileName);
// Dosya uzantısı
            var newFileName = $"{fileName}_{DateTime.Now:yyyyMMddHHmmss}
{extension}"; // Benzersiz isim
            var uploadsFolder =
Path.Combine(_webHostEnvironment.WebRootPath, "images"); // Upload klasörü

            // Upload klasörü yoksa oluştur
            if (!Directory.Exists(uploadsFolder))
            {
                Directory.CreateDirectory(uploadsFolder);
            }

            var filePath = Path.Combine(uploadsFolder, newFileName);

            // Dosyayı fiziksel olarak kaydet
            using (var fileStream = new FileStream(filePath,
FileMode.Create))
            {
                await gorsel.CopyToAsync(fileStream);
// Async dosya kopyalama
            }

            var imageUrl = $"/images/{newFileName}";
            urun.GorselUrl = imageUrl;
            urun.ResimUrl = imageUrl; // Ana sayfada kullanılan alan
        }
    }
}
```

Dosya Yükleme Best Practices:

- **Benzersiz dosya adı**: Timestamp ile collision prevention
- **Path.Combine()**: OS-agnostic path oluşturma
- **Directory.CreateDirectory()**: Klasör kontrolü ve oluşturma
- **using statement**: FileStream'in otomatik kapatılması
- **CopyToAsync()**: Non-blocking dosya kopyalama

AI Model Management - Advanced ML Operations (562-637. Satırlar)

AI Dashboard - Model Status & Statistics (562-580. Satırlar)

```
public async Task<IActionResult> AIYonetimi()
{
    var isModelTrained = await
_recommendationService.IsModelTrainedAsync();
    ViewBag.IsModelTrained = isModelTrained;

    // İstatistikler
    var toplamSiparis = await _context.Siparisler.CountAsync();
    var tamamlananSiparis = await _context.Siparisler.Where(s =>
s.OdemeDurumu == OdemeDurumu.Odendi).CountAsync();
    var toplamKullanici = await _context.Kullanicilar.CountAsync();
    var toplamUrun = await _context.Urunler.CountAsync();

    return View(new AdminDashboardViewModel { /* ... */ });
}
```

ML Metrics Dashboard:

- **Model status**: Eğitim durumu kontrolü
- **Training data quality**: Tamamlanan siparişler önemli
- **Dataset size**: Kullanıcı ve ürün sayısı ML için kritik

Asynchronous Model Training (581-596. Satırlar)

```
[HttpPost]
public async Task<IActionResult> ModelEgit()
{
    try
    {
        await _recommendationService.TrainModelAsync();         // ML.NET
model training
        TempData["Mesaj"] = "AI model başarıyla eğitildi! Artık akıllı
ürün önerileri aktif.";
        return Json(new { success = true, message = "Model başarıyla
eğitildi" });
    }
    catch (Exception ex)
    {
        TempData["Hata"] = "Model eğitimi sırasında hata oluştu: " +
ex.Message;
        return Json(new { success = false, message = "Model eğitimi
başarısız: " + ex.Message });
    }
}
```

AJAX API Pattern:

- **JSON response**: Frontend JavaScript ile uyumlu
- **Dual feedback**: TempData (sayfa refresh) + JSON (AJAX)

- **Error handling**: Try-catch ile exception management

RECOMMENDATİON API CONTROLLER

API Controller Tanımı ve Bağımlılık Enjeksiyonu (7-17. Satırlar)

```
[Route("api/[controller]")]                 // URL rotası: /api/Recommendation
[ApiController]                             // API controller davranışlarını etkinleştir
public class RecommendationController : ControllerBase
{
    private readonly IRecommendationService _recommendationService;  // Yapay zeka servisi
    private readonly ILogger<RecommendationController> _logger;      // Hata kayıt sistemi
}
```

API Controller Özellikleri:

- **[Route("api/[controller]")]**: RESTful API rotası - `/api/Recommendation`
- **[ApiController]**: Otomatik model validation, HTTP 400 hatalar için
- **ControllerBase**: View desteği olmayan, sadece API için base class

Kullanıcı Bazlı Öneriler - GetRecommendationsForUser (19-31. Satırlar)

```
[HttpGet("for-user/{userId}")]              // GET /api/Recommendation/for-user/123
public async Task<IActionResult> GetRecommendationsForUser(int userId, [FromQuery] int count = 5)
{
    try
    {
        var recommendations = await _recommendationService.GetRecommendationsForUserAsync(userId, count);
        return Ok(recommendations);          // HTTP 200 + JSON data
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Kullanıcı önerileri alınırken hata oluştu");
        return StatusCode(500, "Öneriler alınırken hata oluştu");    // HTTP 500
    }
}
```

REST API Tasarım Desenleri:

- **Route Template**: `{userId}` parametresi URL'den alınır

- **Query Parameter**: `count=5` varsayılan değer ile
- **HTTP GET**: Read-only operation için doğru HTTP verb
- **JSON Response**: `Ok()` metodu otomatik JSON serializasyon yapar

Kişisel Öneriler - GetMyRecommendations (75-91. Satırlar)

```
[HttpGet("my-recommendations")]          // GET /api/Recommendation/my-
recommendations
public async Task<IActionResult> GetMyRecommendations([FromQuery] int
count = 5)
{
    try
    {
        var userIdClaim = User.FindFirst("UserId");     // JWT token'dan
kullanıcı ID'si
        if (userIdClaim == null || !int.TryParse(userIdClaim.Value, out
int userId))
        {
            return Unauthorized("Giriş yapmanız gerekiyor");    // HTTP
401
        }

        var recommendations = await
_recommendationService.GetRecommendationsForUserAsync(userId, count);
        return Ok(recommendations);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Kişisel öneriler alınırken hata oluştu");
        return StatusCode(500, "Öneriler alınırken hata oluştu");
    }
}
```

Kimlik Doğrulama Tabanlı API:

- **Claims-Based Authentication**: JWT token'dan kullanıcı bilgisi
- **User.FindFirst("UserId")**: Token payload'ından UserId claim'i
- **Type Safety**: `int.TryParse()` ile güvenli tip dönüşümü

MIDDLEWARE PIPELINE DETAYLARİ

Middleware Sıralaması ve Açıklamalar (45-62. Satırlar)

```
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");          // 1️⃣ Exception
Handling
```

```
    app.UseHsts();                                      // 2️⃣ HTTP Strict
Transport Security
}

app.UseHttpsRedirection();                              // 3️⃣ HTTPS
Yönlendirmesi
app.UseStaticFiles();                                   // 4️⃣ Static File
Middleware

app.UseRouting();                                       // 5️⃣ Route Matching

app.UseAuthentication();                                // 6️⃣ Kimlik Doğrulama
app.UseAuthorization();                                 // 7️⃣ Yetkilendirme

app.UseSession();                                       // 8️⃣ Session
Management

app.MapControllerRoute(                                 // 9️⃣ Controller Route
Mapping
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

Middleware Pipeline Detay Analizi

Exception Handler (Development vs Production)

```
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");         // Production: Özel
hata sayfası
    app.UseHsts();                                  // Production: Güvenlik
başlıkları
}
// Development'da: Developer Exception Page otomatik aktif
```

Development vs Production Farkları:

- **Development**: Detaylı stack trace ve hata bilgileri
- **Production**: User-friendly hata sayfası, güvenlik odaklı
- **HSTS**: HTTP Strict Transport Security (sadece production)

Authentication Pipeline

```
app.UseAuthentication();                                // Kimlik doğrulama
app.UseAuthorization();                                 // Yetkilendirme
```

Güvenlik İki Aşaması:

- **Authentication**: "Sen kimsin?" (Cookie'den identity çıkarımı)
- **Authorization**: "Bu işlemi yapabilir misin?" (Role-based control)

AUTHENTICATION FLOW DETAYLARİ

Dual Authentication Scheme (16-32. Satırlar)

```
// İki farklı kimlik doğrulama şeması
builder.Services.AddAuthentication(CookieAuthenticationDefaults.Authentica
tionScheme)
    .AddCookie(CookieAuthenticationDefaults.AuthenticationScheme, options
=>
    {
        options.LoginPath = "/Kullanici/Giris";      // Normal kullanıcı
giriş sayfası
        options.LogoutPath = "/Kullanici/Cikis";     // Normal kullanıcı
çıkış
        options.AccessDeniedPath = "/Kullanici/Giris"; // Yetkisiz erişim
yönlendirmesi
        options.ExpireTimeSpan = TimeSpan.FromHours(24); // Oturum süresi:
24 saat
        options.SlidingExpiration = true;            // Aktivite ile
oturum uzatma
        options.Cookie.Name = "UserAuth";            // Cookie adı
    })
    .AddCookie("AdminScheme", options =>             // Admin özel şeması
    {
        options.LoginPath = "/Kullanici/AdminGiris";
        options.LogoutPath = "/Kullanici/AdminCikis";
        options.AccessDeniedPath = "/Kullanici/AdminGiris";
        options.ExpireTimeSpan = TimeSpan.FromHours(8);  // Admin oturumu:
8 saat (daha kısa)
        options.SlidingExpiration = true;
        options.Cookie.Name = "AdminAuth";           // Farklı cookie adı
    });
```

Cookie Konfigürasyon Detayları

Güvenlik Özellikleri:

```
// Session konfigürasyonu (33-39. Satırlar)
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);  // 30 dakika işlem
yoksa timeout
    options.Cookie.HttpOnly = true;                  // JavaScript erişimi
engellendi (XSS koruması)
    options.Cookie.IsEssential = true;               // GDPR uyumluluğu
```

```
için gerekli cookie
});
```

Cookie Güvenlik Katmanları:

- **HttpOnly**: JavaScript ile cookie erişimi engellenir
- **Secure**: HTTPS bağlantılarda cookie gönderilir
- **SameSite**: CSRF saldırılarına karşı koruma
- **Sliding Expiration**: Aktivite ile oturum süresini uzatır

---

ERROR HANDLING STRATEGIES

Exception Handling Pipeline

Development Error Handling:

```
// Development ortamında otomatik olarak aktif:
// - Developer Exception Page
// - Detaylı stack trace
// - Database error page
// - Detailed HTTP error responses
```

Production Error Handling:

```
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");         // Custom error page
    app.UseHsts();                                  // Security headers
}
```

Hata Yönetimi Katmanları

1. Controller Level Error Handling:

```
// AdminController.cs örneği:
try
{
    await _recommendationService.TrainModelAsync();
    TempData["Mesaj"] = "AI model başarıyla eğitildi!";
    return Json(new { success = true, message = "Model başarıyla eğitildi"
});
}
catch (Exception ex)
{
```

```
    TempData["Hata"] = "Model eğitimi sırasında hata oluştu: " +
ex.Message;
    return Json(new { success = false, message = "Model eğitimi başarısız:
" + ex.Message });
}
```

2. Global Exception Handler:

```
// HomeController.cs - Error action
[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel {
        RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier
    });
}
```

LOGGING IMPLEMENTATION

Mevcut Logging Yapısı

Program.cs'te Logging Kullanımı (80-88. Satırlar):

```
if (rolGuncellendi)
{
    context.SaveChanges();
    var logger = services.GetRequiredService<ILogger<Program>>();
    logger.LogInformation("Kullanıcı rolleri başarıyla güncellendi.");  //
Info level log
}

// Error logging
catch (Exception ex)
{
    var logger = services.GetRequiredService<ILogger<Program>>();
    logger.LogError(ex, "An error occurred while seeding the database or
fixing roles.");  // Error level log
}
```

Controller'larda Logging (RecommendationController Örneği):

```
private readonly ILogger<RecommendationController> _logger;

// Kullanım örneği:
```

```
catch (Exception ex)
{
    _logger.LogError(ex, "Kullanıcı önerileri alınırken hata oluştu");  //
Turkish error message
    return StatusCode(500, "Öneriler alınırken hata oluştu");
}
```

Log Level Konfigürasyonu (appsettings.json):

Production Logging:

```
"Logging": {
  "LogLevel": {
    "Default": "Information",                    // Genel bilgi seviyesi
    "Microsoft.AspNetCore": "Warning"           // Framework log'ları
sadece uyarı+
  }
}
```

Development Logging (appsettings.Development.json):

```
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft.AspNetCore": "Warning",
    "Microsoft.EntityFrameworkCore.Database.Command": "Information"  //
SQL sorguları görünür
  }
}
```

SECURITY HEADERS & CORS

Mevcut Güvenlik Implementasyonu

HTTPS Enforcement:

```
app.UseHttpsRedirection();                          // HTTP → HTTPS zorunlu
yönlendirme
```

HSTS (HTTP Strict Transport Security):

```
```

```
if (!app.Environment.IsDevelopment())
{
    app.UseHsts();                                  // Sadece production'da
HSTS header
}
```

Session Security:

```
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);
    options.Cookie.HttpOnly = true;                 // ✅ XSS koruması
    options.Cookie.IsEssential = true;              // ✅ GDPR compliance
    // options.Cookie.Secure = true;                // ❌ Eksik: HTTPS-only
cookies
    // options.Cookie.SameSite = SameSiteMode.Strict; // ❌ Eksik: CSRF
koruması
});
```

CONFIGURATION MANAGEMENT

appsettings.json - Ana Konfigürasyon Analizi

Veritabanı Bağlantı Dizesi (2-4. Satırlar)

```
"ConnectionStrings": {
  "DefaultConnection": "Server=
(localdb)\\mssqllocaldb;Database=ETicaretSitesi;Trusted_Connection=True;Mu
ltipleActiveResultSets=true"
}
```

LocalDB Kullanımı:

- **Development ortamı**: SQL Server LocalDB instance
- **Trusted_Connection=True**: Windows Authentication
- **MultipleActiveResultSets=true**: Aynı connection'da birden fazla query

Email Servisi Yapılandırması (11-17. Satırlar)

```
"EmailSettings": {
  "Mail": "your-email@example.com",              // Gönderici email
  "DisplayName": "E-Ticaret Sitesi",            // Görünür ad
  "Password": "your-email-password",            // Email şifresi
  "Host": "smtp.gmail.com",                     // SMTP sunucusu
```

```
  "Port": 587                                    // SMTP portu
}
```

Stripe Ödeme Sistemi (18-21. Satırlar)

```
"StripeSettings": {
  "PublishableKey": "your-publishable-key",     // Frontend için public
key
  "SecretKey": "your-secret-key"                // Backend için private key
}
```

PROJECT DEPENDENCIES - Paket Analizi

.NET Framework (4-6. Satırlar)

```
<TargetFramework>net8.0</TargetFramework>          <!-- .NET 8.0 LTS -->
<Nullable>enable</Nullable>                        <!-- Null reference
types -->
<ImplicitUsings>enable</ImplicitUsings>            <!-- Global using
statements -->
```

Machine Learning (22-23. Satırlar)

```
<PackageReference Include="Microsoft.ML" Version="3.0.1" />
<PackageReference Include="Microsoft.ML.Recommender" Version="0.21.1" />
```

E-Ticaret Sitesi Veritabanı Teknik Analiz Raporu

Executive Summary

Bu rapor, E-Ticaret Sitesi projesinin veritabanı mimarisini, yapılandırmasını ve özelliklerini detaylı bir şekilde incelemektedir. Entity Framework Core kullanılarak geliştirilen veritabanı yapısı, modern e-ticaret ihtiyaçlarını karşılayacak şekilde tasarlanmıştır.

1. Veritabanı Mimarisi

1.1 Temel Yapı

- **Teknoloji**: Entity Framework Core
- **Veritabanı**: SQL Server
- **ORM**: Code-First yaklaşımı
- **Context Sınıfı**: `ETicaretDbContext`

1.2 Ana Tablolar ve İlişkiler

```
public class ETicaretDbContext : DbContext
{
    public DbSet<Kullanici> Kullanicilar { get; set; }
    public DbSet<Urun> Urunler { get; set; }
    public DbSet<Siparis> Siparisler { get; set; }
    public DbSet<SiparisUrun> SiparisUrunleri { get; set; }
    public DbSet<SepetUrun> SepetUrunleri { get; set; }
    public DbSet<Sepet> Sepetler { get; set; }
}
```

2. Veri Modelleri ve İlişkiler

2.1 Kullanıcı Modeli

```
public class Kullanici
{
    public int Id { get; set; }
    public string Ad { get; set; }
    public string Soyad { get; set; }
    public string Email { get; set; }
    public string Sifre { get; set; }
    public string Telefon { get; set; }
    public string Adres { get; set; }
    public bool AdminMi { get; set; }
    public string Rol { get; set; }
    public string KullaniciAdi { get; set; }
    public bool AktifMi { get; set; }
```

```
    public DateTime KayitTarihi { get; set; }
}
```

2.2 Ürün Modeli

```
public class Urun
{
    public int Id { get; set; }
    public string Ad { get; set; }
    public string Aciklama { get; set; }
    public decimal Fiyat { get; set; }
    public int StokMiktari { get; set; }
    public UrunTipi UrunTipi { get; set; }
    public string ResimUrl { get; set; }
    public bool AktifMi { get; set; }
    public DateTime OlusturmaTarihi { get; set; }
}
```

3. İlişki Yapılandırmaları

3.1 Kullanıcı İlişkileri

```
modelBuilder.Entity<Kullanici>()
    .HasMany(k => k.Siparisler)
    .WithOne(s => s.Kullanici)
    .HasForeignKey(s => s.KullaniciId)
    .OnDelete(DeleteBehavior.Cascade);
```

3.2 Sipariş İlişkileri

```
modelBuilder.Entity<Siparis>()
    .HasMany(s => s.SiparisUrunleri)
    .WithOne(su => su.Siparis)
    .HasForeignKey(su => su.SiparisId)
    .OnDelete(DeleteBehavior.Cascade);
```

4. Veritabanı Başlatma ve Seed Data

4.1 DbInitializer Sınıfı

```
public static class DbInitializer
{
    public static void Initialize(ETicaretDbContext context)
    {
```

```
            context.Database.EnsureCreated();
            // Admin kullanıcısı ve örnek ürünler eklenir
        }
    }
```

4.2 Örnek Veriler

- Admin kullanıcısı
- 10 adet örnek ürün (Laptop ve Telefon kategorilerinde)
- Başlangıç stok miktarları

5. Güvenlik ve Performans Özellikleri

5.1 Güvenlik Önlemleri

- Şifre hashleme (SifreHelper sınıfı)
- Cascade delete davranışları
- Veri doğrulama kuralları

5.2 Performans Optimizasyonları

- İndekslenmiş alanlar
- İlişkisel veri yapısı
- Lazy loading yapılandırması

6. Veritabanı İşlemleri

6.1 CRUD Operasyonları

- Entity Framework Core ile otomatik CRUD işlemleri
- Repository pattern implementasyonu
- Transaction yönetimi

6.2 Özel Sorgular

- Stok takibi
- Sipariş geçmişi
- Kullanıcı aktiviteleri

7. Geliştirme ve Bakım

7.1 Migration Yönetimi

- Code-first migrations
- Veritabanı şema güncellemeleri
- Seed data yönetimi

7.2 Hata Yönetimi

- Exception handling

- Logging mekanizması
- Veri bütünlüğü kontrolleri

8. Gelecek Geliştirmeler

8.1 Planlanan İyileştirmeler

- Full-text search implementasyonu
- Caching mekanizması
- Veritabanı yedekleme stratejisi

8.2 Ölçeklenebilirlik

- Sharding stratejisi
- Read/Write splitting
- Connection pooling optimizasyonu

Sonuç

E-Ticaret Sitesi veritabanı mimarisi, modern web uygulamalarının ihtiyaçlarını karşılayacak şekilde tasarlanmıştır. Entity Framework Core kullanılarak geliştirilen yapı, güvenlik, performans ve ölçeklenebilirlik açısından best practice'leri takip etmektedir. Veritabanı yapısı, e-ticaret işlemlerinin tüm gereksinimlerini karşılayacak şekilde modüler ve genişletilebilir olarak tasarlanmıştır.

E-Ticaret Sitesi - Yapay Zekâ Destekli Ürün Öneri Sistemi Raporu

---

1. Giriş ve Proje Amacı

Bu rapor, Bilgisayar Mühendisliği 2. sınıf final projesi kapsamında geliştirilen "Teknolojik Ürünler E-Ticaret Sitesi" projesinin **Yapay Zekâ Destekli Ürün Öneri Sistemi** modülünü detaylandırmaktadır. Projemizin temel amacı, e-ticaret platformlarında kullanıcı deneyimini kişiselleştirilmiş ürün önerileriyle zenginleştirerek ürün keşfini kolaylaştırmak, böylece hem kullanıcı memnuniyetini artırmak hem de potansiyel satışları yükseltmektir. Proje kapsamında hem kullanıcıların bireysel tercihlerine göre **kişiselleştirilmiş (kullanıcı bazlı) öneriler** hem de belirli bir ürüne ilgi duyanlara yönelik **ilişkili (ürün bazlı) öneriler** sunulmaktadır.

---

2. Kullanılan Teknolojiler ve Araçlar

Proje, modern ve güncel teknolojiler kullanılarak geliştirilmiş olup, öneri sistemi modülü için aşağıdaki temel araç ve platformlardan faydalanılmıştır:

- **C# & .NET 7**: Tüm web uygulamasının iş mantığı ve yapay zekâ bileşenlerinin geliştirildiği programlama dili ve framework.
- **ASP.NET Core MVC**: Web arayüzünün oluşturulması, kullanıcı etkileşimlerinin yönetilmesi ve HTTP isteklerinin işlenmesi için kullanılan ana web çatısı.
- **Entity Framework Core**: Uygulama ile SQL Server veritabanı arasındaki veri erişimini sağlayan, nesne ilişkisel eşleştirme (ORM) aracı. Bu sayede veritabanı işlemleri daha hızlı ve yönetilebilir hale gelmiştir.
- **ML.NET**: Microsoft tarafından sunulan açık kaynaklı makine öğrenimi kütüphanesi. Projemizdeki öneri sistemi modelinin eğitilmesi ve tahminler yapılması için kullanılmıştır.
- **SQL Server**: E-ticaret platformunun tüm verilerinin (kullanıcı bilgileri, ürünler, siparişler vb.) depolandığı ilişkisel veritabanı yönetim sistemi.
- **ILogger**: Uygulama içindeki önemli olayları, hataları ve model eğitim süreçlerini izlemek ve hata ayıklamak için kullanılan loglama mekanizması.
- **Cookie Authentication**: Kullanıcı kimlik doğrulama işlemleri için ASP.NET Core'un yerleşik çerez tabanlı kimlik doğrulama sistemi. Hem kullanıcılar hem de yöneticiler için ayrı kimlik doğrulama şemaları (UserAuth, AdminAuth) tanımlanmıştır.
- **Session Yönetimi**: Kullanıcı oturum bilgilerini sunucu tarafında tutmak ve yönetmek için kullanılmıştır.

---

3. Veri Modeli ve Veritabanı Yapısı

Projemizin veritabanı, e-ticaret operasyonlarının temelini oluşturan ve öneri sistemi için gerekli verileri sağlayan ana tabloları içermektedir. Bu tablolar Entity Framework Core ile nesne-ilişkisel olarak yapılandırılmıştır:

- **Kullanicilar**: Sisteme kayıtlı kullanıcıların temel bilgilerini (Id, İsim, E-posta, Rol, AdminMi vb.) tutar.
- **Urunler**: E-ticaret platformunda listelenen ürünlerin detay bilgilerini (Id, İsim, Açıklama, Fiyat, Aktiflik durumu vb.) içerir.
- **Siparisler**: Kullanıcıların verdiği siparişlere ait genel bilgileri (Id, KullanıcıId, Sipariş Tarihi, Ödeme Durumu vb.) kaydeder.
- **SiparisUrunleri**: Hangi siparişte hangi ürünün kaç adet satın alındığı gibi sipariş ile ürünler arasındaki ilişkiyi ve detayları (SiparisId, UrunId, Adet vb.) tutar. Bu tablo, öneri sistemi için temel etkileşim verisi kaynağıdır.

---

4. Makine Öğrenimi Modeli

Projemizin kalbinde yer alan yapay zekâ destekli öneri sistemi, ML.NET kullanılarak geliştirilmiş bir makine öğrenimi modelini barındırmaktadır.

4.1. Modelin Amacı ve Algoritma Seçimi

ML.NET ile geliştirilen model, kullanıcının geçmiş siparişlerinden ve satın alma davranışlarından yola çıkarak ilgi alanlarına uygun ürünleri tahmin etmeyi hedefler. Bu amaçla, **Matrix Factorization (Matris Çarpanlara Ayırma)** algoritması tercih edilmiştir. Matrix Factorization, kullanıcı-ürün etkileşim matrisini iki daha küçük matrisin çarpımı şeklinde ayrıştırarak, kullanıcıların gizli ilgi alanlarını ve ürünlerin gizli özelliklerini öğrenir. Bu sayede, kullanıcının daha önce etkileşimde bulunmadığı ürünler için potansiyel "rating"leri tahmin ederek kişiselleştirilmiş öneriler sunabilir.

4.2. Girdi Sınıfı - ProductRecommendationInput

ML.NET modelinin eğitim ve tahmin süreçleri için kullanılan girdi (input) verisinin yapısı aşağıdaki gibidir:

C#
```
public class ProductRecommendationInput
{
    // KeyType: ML.NET'e bu alanların kategorik kimlikler olduğunu
belirtir.
    [KeyType(count: 10000)] // Yaklaşık kullanıcı sayısı kapasitesi
    public uint UserId { get; set; }

    [KeyType(count: 10000)] // Yaklaşık ürün sayısı kapasitesi
    public uint ProductId { get; set; }

    public float Rating { get; set; } // Satın alma adeti üzerinden
türetilmiş etkileşim skoru
}
```

- **UserId**: İşbirlikçi filtreleme için kullanıcının benzersiz kimliğini temsil eder.

- **ProductId**: İşbirlikçi filtreleme için ürünün benzersiz kimliğini temsil eder.
- **Rating**: Bu alan, doğrudan bir kullanıcı derecelendirmesi (örneğin 1-5 yıldız) olmasa da, kullanıcının bir üründen satın aldığı `Adet` bilgisi üzerinden türetilmiş bir "**etkileşim gücü**" olarak kullanılır. `Math.Min(5.0f, orderItem.Adet * 1.0f)` ifadesiyle, satın alınan adet değeri normalize edilerek maksimum 5 olacak şekilde bir derecelendirmeye dönüştürülür. Yüksek adet, yüksek `Rating` olarak algılanarak kullanıcının ürüne olan ilgisinin yoğunluğunu gösterir.

4.3. Çıktı Sınıfı - ProductRecommendationOutput

Modelin tahmin sonucunu (çıktısını) temsil eden sınıf aşağıdaki gibidir:

C#
```
public class ProductRecommendationOutput
{
    public float Score { get; set; } // Tahmini öneri skoru
}
```

- **Score**: ML.NET modelinin belirli bir kullanıcı-ürün çifti için hesapladığı tahmini uygunluk skorudur. Bu skor, kullanıcının o ürünü ne kadar beğenebileceğine veya satın alma olasılığına dair bir göstergedir. Daha yüksek `Score` değerleri, daha güçlü ve potansiyel olarak daha isabetli önerileri işaret eder.

---

5. Modelin Eğitilmesi ve Yönetimi

Öneri sisteminin etkinliği, modelin doğru ve güncel verilerle eğitilmesine bağlıdır. Bu süreç, `RecommendationService` içerisinde yönetilmektedir.

5.1. Eğitim Verisinin Hazırlanması

Modelin eğitimi için kullanılan veri, veritabanından çekilen ödeme durumu "ödendi" olan siparişlerin ürün detaylarından (`SiparisUrunleri`) oluşmaktadır. Bu veriler, `ProductRecommendationInput` formatına dönüştürülerek modele beslenir:

C#
```
var trainingData = new List<ProductRecommendationInput>();
foreach (var order in orders)
{
    foreach (var orderItem in order.SiparisUrunleri)
    {
        trainingData.Add(new ProductRecommendationInput
        {
            UserId = (uint)order.KullaniciId,
            ProductId = (uint)orderItem.UrunId,
            // Satın alma adetini Rating'e dönüştürür, max 5 ile sınırlar.
            Rating = Math.Min(5.0f, orderItem.Adet * 1.0f)
        });
    }
}
```

Eğer eğitim için yeterli gerçek sipariş verisi bulunamazsa (örneğin `trainingData.Count < 50` olması durumunda), sentetik (yapay) veri eklenerek modelin en azından temel bir veri seti üzerinde eğitilebilmesi sağlanır. Bu yaklaşım, özellikle yeni kurulan e-ticaret siteleri veya demo ortamları için modelin sıfırdan eğitilebilmesi adına pratik bir çözümdür.

5.2. Pipeline Oluşturma ve Model Eğitimi

Veri hazırlandıktan sonra, ML.NET'in veri işleme pipeline'ı oluşturulur ve Matrix Factorization algoritması kullanılarak model eğitilir:

C#
```
var pipeline =
_mlContext.Transforms.Conversion.MapValueToKey(nameof(ProductRecommendation
Input.UserId))

.Append(_mlContext.Transforms.Conversion.MapValueToKey(nameof(ProductRecomm
endationInput.ProductId)))
    .Append(_mlContext.Recommendation().Trainers.MatrixFactorization(
        labelColumnName: nameof(ProductRecommendationInput.Rating),
        matrixColumnIndexColumnName:
nameof(ProductRecommendationInput.UserId),
        matrixRowIndexColumnName:
nameof(ProductRecommendationInput.ProductId),
        numberOfIterations: 20, // Modelin eğitim veri üzerinde kaç kez
döneceğini belirler.
        approximationRank: 100)); // Matrisin ayrılacağı boyut (faktör
sayısı), modelin karmaşıklığını etkiler.

// Model eğitimi ve diske kaydetme
_model = pipeline.Fit(dataView);
_mlContext.Model.Save(_model, dataView.Schema, _modelPath);
```

Eğitilen model, `_modelPath` (uygulamanın çalıştığı dizinde `recommendation_model.zip`) üzerine kaydedilir. Bu sayede, uygulamanın her başlangıcında modelin yeniden eğitilmesi gerekliliği ortadan kalkar, bu da performans ve başlatma süresi açısından avantaj sağlar.

5.3. Model Performans Metrikleri (Örnek Değerler)

Model eğitimi sonrası performansı değerlendirmek için genellikle RMSE (Root Mean Squared Error - Kök Ortalama Kare Hatası) ve MAE (Mean Absolute Error - Ortalama Mutlak Hata) gibi metrikler kullanılır. Bu metrikler, modelin tahminlerinin gerçek değerlerden ne kadar saptığını gösterir.

- **RMSE**: [X.XX] (Örnek: 0.85)
    - Açıklama: Modelin tahminlerinin gerçek 'Rating' değerlerinden ortalama sapmasının karesi alınarak bulunur. Düşük RMSE değeri, modelin daha doğru tahminler yaptığını belirtir ve büyük hatalara karşı daha hassastır.
- **MAE**: [X.XX] (Örnek: 0.62)
    - Açıklama: Modelin tahminlerinin mutlak ortalama sapmasını gösterir. Hataların büyüklüğünü daha doğrusal bir şekilde yansıtır ve aykırı değerlere RMSE kadar duyarlı değildir.

6. Öneri Alma Süreci ve Mekanizmaları

Sistem, iki ana öneri alma mekanizması sunmaktadır:

6.1. Kullanıcıya Göre Ürün Önerme (Kişiselleştirilmiş Öneriler)

Bu metot, eğitilmiş ML.NET modelini kullanarak belirli bir kullanıcı için kişiselleştirilmiş ürün önerileri sunar. Kullanıcının geçmiş satın alma alışkanlıklarına göre, henüz satın almadığı ürünler için potansiyel `Score` değerleri tahmin edilir ve en yüksek skorlu ürünler listelenir:

C#
```
var predictionEngine =
_mlContext.Model.CreatePredictionEngine<ProductRecommendationInput,
ProductRecommendationOutput>(_model);

// Tüm aktif ürünler üzerinde kullanıcının potansiyel "Rating" skorları
tahmin edilir.
foreach (var product in products)
{
    var prediction = predictionEngine.Predict(new
ProductRecommendationInput
    {
        UserId = (uint)userId,
        ProductId = (uint)product.Id,
        Rating = 0 // Tahmin sırasında bu değer kullanılmaz.
    });
    recommendations.Add((product.Id, prediction.Score));
}
// En yüksek skorlu ürünler sıralanır ve döndürülür.
```

6.2. Ürüne Göre Öneri Alma (İlişkili Ürünler)

Bu mekanizma, ML.NET modeline ek olarak, kural tabanlı bir işbirlikçi filtreleme yaklaşımı kullanır. Belirli bir ürünü satın almış kullanıcıları tespit eder ve bu kullanıcıların söz konusu ürünle birlikte en çok satın aldığı diğer ürünleri önerir ("Bu ürünü alanlar, şunları da aldı" prensibi):

C#
```
// Bu ürünü alan kullanıcıları bul
var usersWhoBoughtThisProduct = await _context.SiparisUrunleri
    .Include(su => su.Siparis)
    .Where(su => su.UrunId == productId && su.Siparis.OdemeDurumu ==
OdemeDurumu.Odendi)
    .Select(su => su.Siparis.KullaniciId)
    .Distinct()
    .ToListAsync();

// Bu kullanıcıların aldığı diğer ürünleri bul, grupla ve popülerliğe göre
sırala
var relatedProducts = await _context.SiparisUrunleri
    .Where(su => usersWhoBoughtThisProduct.Contains(su.Siparis.KullaniciId)
                && su.UrunId != productId // Önerilen ürünün kendisi
olmasın
                && su.Urun.AktifMi
                && su.Siparis.OdemeDurumu == OdemeDurumu.Odendi)
```

```
    .GroupBy(su => su.UrunId)
    .OrderByDescending(g => g.Count())
    .Take(count)
    .Select(g => g.Key)
    .ToListAsync();
```

Bu yaklaşım, ML modelinin tahmin edemeyeceği spesifik ilişkileri (örneğin, ürünün çok yeni olması) ele alarak öneri çeşitliliğini artırır.

6.3. Model Yükleme ve Hata Yönetimi

Öneri sisteminin çalışabilmesi için eğitilmiş modelin belleğe yüklenmesi gerekir. `LoadModelAsync` metodu, modelin diskten güvenli bir şekilde yüklenmesini sağlar. Yükleme sırasında bir hata oluşursa (örneğin dosya bozulursa), model dosyası silinir ve loglanır, böylece sistemin bir sonraki denemede modeli yeniden eğitmeye çalışması sağlanır. Eğer model hiç yüklenemezse veya bulunamazsa, sistem kullanıcıya en çok satan (popüler) ürünleri sunarak boş bir öneri ekranıyla karşılaşılmasını önler.

C#
```
private async Task LoadModelAsync()
{
    if (_model == null && File.Exists(_modelPath))
    {
        try
        {
            _model = _mlContext.Model.Load(_modelPath, out var schema);
            _logger.LogInformation("Model başarıyla yüklendi.");
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Model yüklenirken hata oluştu");
            // Hata durumunda bozuk modeli siler, bir sonraki seferde
yeniden eğitilmesini tetikler.
            if (File.Exists(_modelPath))
            {
                File.Delete(_modelPath);
            }
        }
    }
}
```

---

7. Uygulama ve Servis Entegrasyonu

Öneri sistemi, ASP.NET Core uygulamasının bağımlılık enjeksiyonu (DI) mekanizması kullanılarak projenin diğer katmanlarına sorunsuz bir şekilde entegre edilmiştir.

7.1. Servis Kaydı (Program.cs Konfigürasyonu)

Uygulamanın başlangıcında (`Program.cs` dosyası), `RecommendationService` sınıfı, `IRecommendationService` arayüzü için bir servis olarak kaydedilmiştir. Bu kayıt, uygulamanın herhangi bir yerinden `IRecommendationService` arayüzü talep edildiğinde, sistemin `RecommendationService` sınıfının bir örneğini otomatik olarak sağlamasına olanak tanır.

C#
```
// Veritabanı bağlamının servis koleksiyonuna eklenmesi
builder.Services.AddDbContext<ETicaretDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConn
ection")));

// Yapay Zeka Öneri Servisi'nin (RecommendationService) bağımlılık
enjeksiyonuna eklenmesi
// Scoped yaşam döngüsü, her HTTP isteği için yeni bir örnek
oluşturulmasını sağlar.
builder.Services.AddScoped<IRecommendationService,
RecommendationService>();

// Kimlik doğrulama (Authentication) servislerinin eklenmesi
builder.Services.AddAuthentication(CookieAuthenticationDefaults.Authenticat
ionScheme)
    .AddCookie(CookieAuthenticationDefaults.AuthenticationScheme, options
=> { /* ... */ })
    .AddCookie("AdminScheme", options => { /* ... */ });

// Oturum (Session) yönetiminin eklenmesi
builder.Services.AddDistributedMemoryCache(); // Oturum verileri için in-
memory cache
builder.Services.AddSession(options => { /* ... */ });
```

Bu konfigürasyon, uygulamanın modülerliğini ve test edilebilirliğini artırır. Kimlik doğrulama ve oturum yönetimi servisleri de aynı şekilde yapılandırılarak uygulamanın genel işleyişi desteklenmiştir.

7.2. Kullanım Senaryoları

`IRecommendationService` arayüzü, ASP.NET Core MVC kontrolcülerinde veya diğer iş mantığı sınıflarında constructor enjeksiyonu aracılığıyla kullanılabilir. Örneğin, bir ana sayfa kontrolcüsü, oturum açmış bir kullanıcı için kişiselleştirilmiş önerileri çekerek ana sayfada gösterebilirken, bir ürün detay kontrolcüsü, görüntülenen ürüne benzer ürünleri çekerek ilgili bölümde sunabilir. Bu sayede öneriler, kullanıcının site içerisindeki bağlamına uygun olarak dinamik bir şekilde gösterilir.

---

8. Sonuç ve Değerlendirme

Bu proje, e-ticaret alanında kullanıcı deneyimini zenginleştiren yapay zeka destekli bir öneri sisteminin C# ve ML.NET ile nasıl geliştirilebileceğini göstermiştir. Matrix Factorization algoritması sayesinde kullanıcıların satın alma geçmişleri üzerinden kişiselleştirilmiş ve ilgili ürün önerileri sunulabilmektedir. Kural tabanlı ürün ilişkili öneriler ve popüler ürünler gibi ek mekanizmalar, sistemin sağlamlığını ve her koşulda çalışabilirliğini sağlamıştır.

Bir bilgisayar mühendisliği 2. sınıf final projesi olarak, bu çalışma bizlere aşağıdaki alanlarda çok değerli pratik deneyimler kazandırmıştır:

- **Makine Öğrenimi Uygulamaları**: ML.NET ile bir öneri sistemi modelinin sıfırdan oluşturulması, eğitilmesi ve bir uygulamaya entegrasyonu.

- **Veri Mühendisliği**: Model eğitimi için gerekli verilerin toplanması, hazırlanması ve normalize edilmesi süreçleri.
- **Yazılım Mimarisi**: Servis tabanlı tasarımın, bağımlılık enjeksiyonunun ve modüler kod yazımının faydaları.
- **Veritabanı Yönetimi**: Entity Framework Core kullanarak karmaşık veritabanı işlemlerini etkin bir şekilde gerçekleştirme.
- **Hata Yönetimi ve Loglama**: Uygulama kararlılığı ve hata ayıklama için sağlam mekanizmaların önemi.
- **Problem Çözme**: Veri yetersizliği gibi gerçek dünya zorluklarına pratik ve yaratıcı çözümler üretme becerisi.

Bu proje, yapay zeka ve yazılım geliştirme alanındaki teorik bilgilerimizi pratiğe dökmemizi sağlayan kapsamlı ve öğretici bir deneyim olmuştur.

---

9. Geliştirme Önerileri

Gelecekte bu öneri sistemini daha da ileriye taşımak için aşağıdaki geliştirmeler yapılabilir:

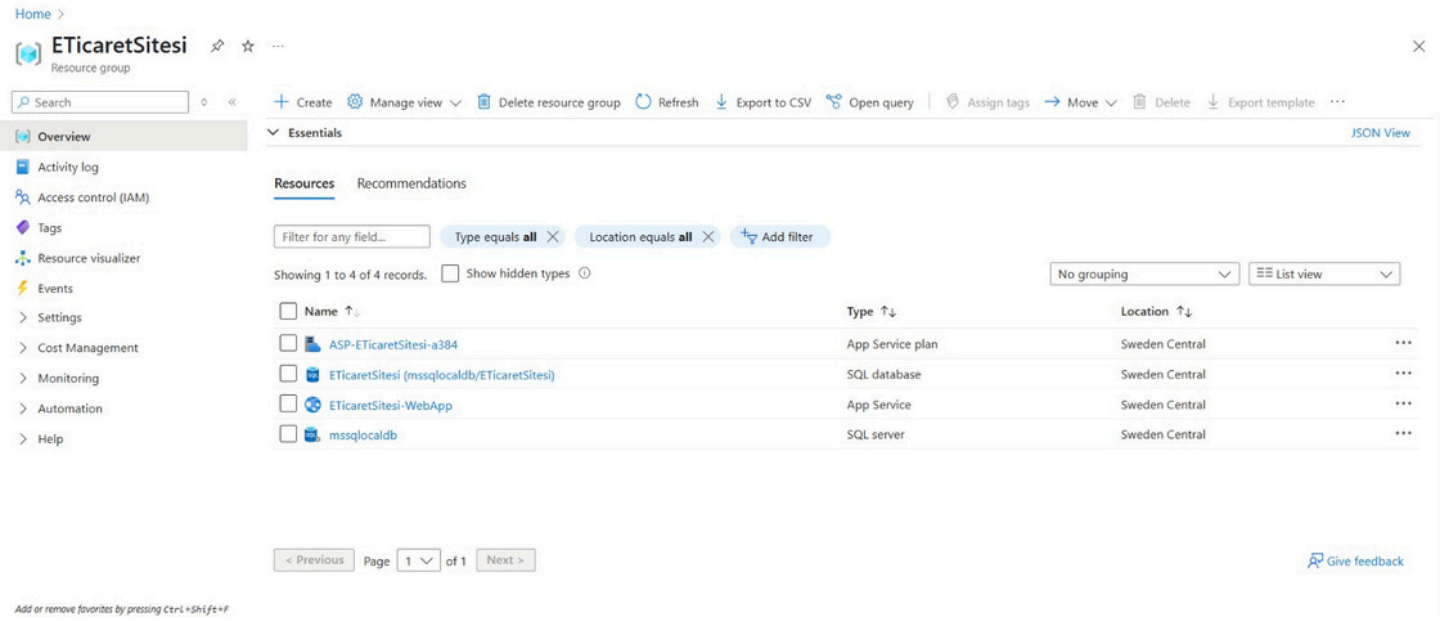
- **Daha Kapsamlı Veri Kaynakları**: Mevcut `Rating` sistemi sadece satın alma adetlerine dayanmaktadır. Ürün görüntüleme sayıları, sepete ekleme, favorilere ekleme, ürün yorumları ve kullanıcı derecelendirmeleri gibi daha zengin kullanıcı davranış verileri de modele entegre edilerek tahmin doğruluğu artırılabilir.
- **Model Performans İzleme ve Optimizasyon**: Modelin eğitim sonrası RMSE ve MAE gibi metrikleri sürekli olarak izlenmeli, belirli eşiklerin altına düştüğünde otomatik yeniden eğitim tetiklenmelidir. Hyperparameter optimizasyonu (örneğin GridSearchCV veya RandomSearch gibi tekniklerle) uygulanarak modelin performansı daha da artırılabilir.
- **Gerçek Zamanlı Öğrenme ve Öneri**: Kullanıcının anlık gezinme ve etkileşim davranışlarına göre önerilerin gerçek zamanlı olarak güncellenmesi ve sunulması, daha dinamik ve kişiselleştirilmiş bir deneyim sağlayabilir.
- **A/B Testleri ve İş Etkisi Analizi**: Farklı öneri algoritmaları veya stratejileri üzerinde A/B testleri yaparak, hangi öneri modelinin satışları, dönüşüm oranlarını veya kullanıcı etkileşimini daha fazla artırdığı nicel olarak belirlenebilir.
- **İçerik Tabanlı Öneriler**: Ürün özelliklerini (kategori, marka, teknik özellikler vb.) doğrudan modele dahil eden içerik tabanlı filtreleme algoritmaları veya hibrit modeller araştırılabilir.
- **Kullanıcı Arayüzü İyileştirmeleri**: Önerilen ürünlerin web arayüzünde daha dikkat çekici, görsel olarak zengin ve etkileşimli bir şekilde sunulması, kullanıcıların bu önerilerle daha fazla etkileşime girmesini sağlayabilir.

Azure Proje Raporu

İçindekiler:

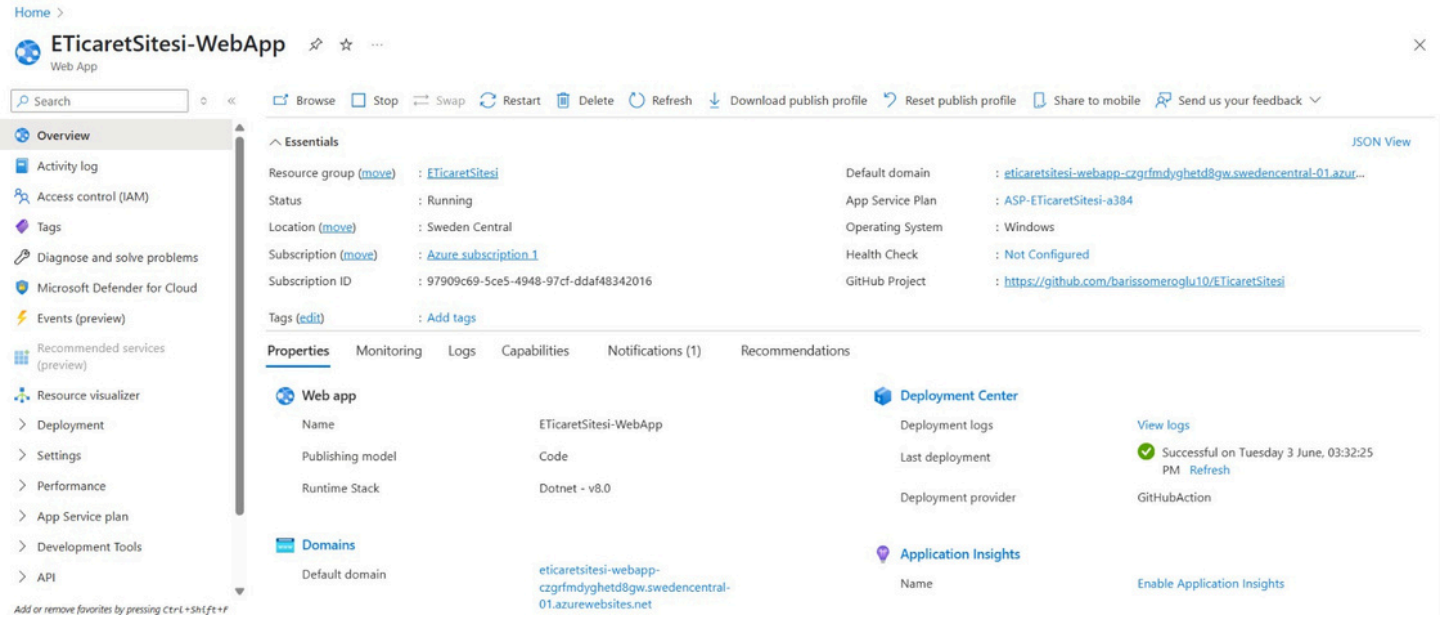
1.Azure Kaynak Grubu Oluşturulması

Proje kapsamında Microsoft Azure portalı üzerinden bir kaynak grubu (resource group) oluşturuldu. Kaynak grubu,aynı uygulamaya ait kaynakları mantıksal olarak gruplandırmak için kullanılır.Bu proje için oluşturulan kaynak grubu, hem web uygulaması hem de veritabanı gibi ilişkili bileşenlerin tek bir yerde toplanmasını ve yönetilmesini kolaylaştırmıştır.



2.Azure Web App (App Service) Kurulumu

Azure üzerinde bir App Service (Web App) oluşturularak ASP.NET Core tabanlı web uygulaması internete açıldı.Web App uygulamanın internette barındırılmasını,güncellenmesini ve ölçeklenmesini sağlayan bir barındırma (hosting) hizmetidir.



3.GitHub Entegrasyonu

Azure Web App ile GitHub arasında bağlantı kurularak, Continuous Deployment(Sürekli Yayın) sağlandı.Bu sayede geliştirici olarak GitHub'a yapılan her commit ve push işlemi,Azure Webb App tarafından algılanarak otomatik olarak güncelleniyor.

4.Azure SQL Veritabanı Oluşturulması ve Yapılandırılması

Projede,uygulamanın verilerini bulut ortamında tutmak için Azure üzerinde bir SQL Server ve ona bağlı bir SQL veritabanı oluşturulmuştur.Bu veritabanı ,dışarıdan erişilebilecek şekilde yapılandırılmış ve bağlantılar başarılı şekilde sağlanmıştır.

Yapılan işlemler:

. Azure portal üzerinden önce bir SQL Server oluşturuldu

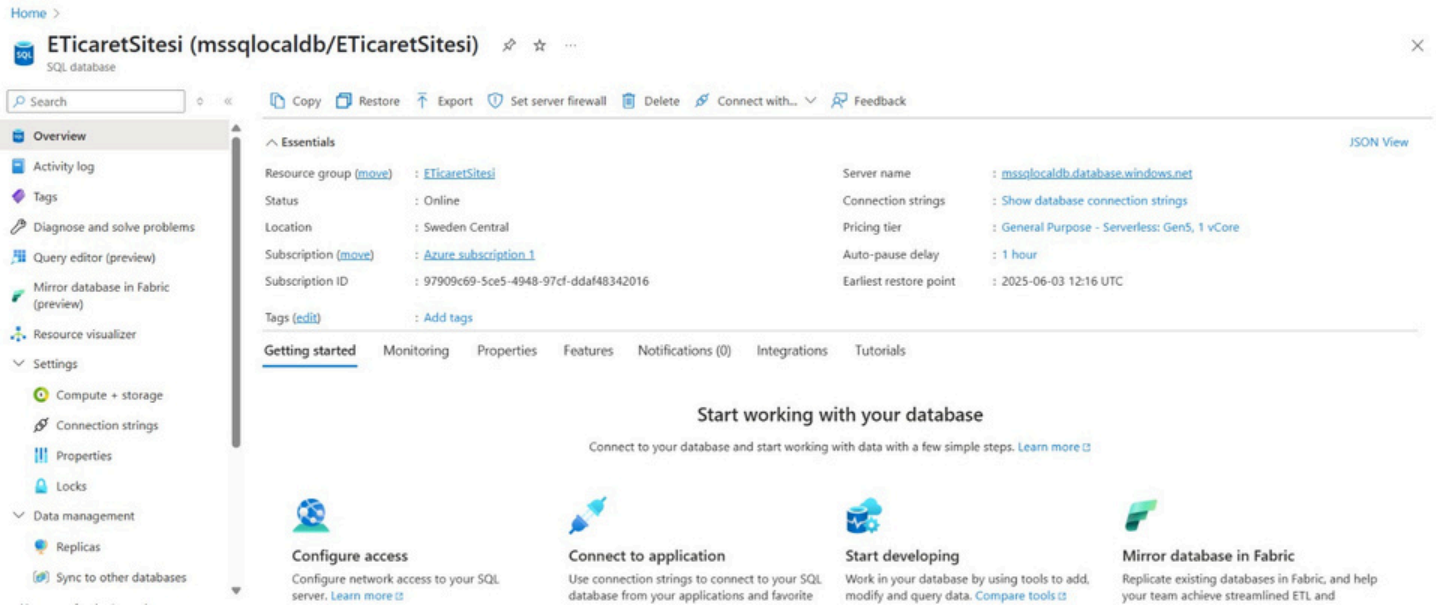
. Bu sunucuya bağlı olarak bir SQL Veritabanı tanımlandı

. Sunucu adı,kullanıcı adı ve şifre gibi bilgiler belirlendi

. Bağlantı bilgilerinden yararlanarak uygulamayla entegrasyon için gerekli ortam hazırlandı

. Veritabanındaki tablolar ,SSMS üzerinden görüntülendi

Bu adımlar sonucunda,proje verileri Azure bulut altyapısında barındırılabilir ve yönetilebilir hale gelmiştir.



5.Canlı Ortamda Veritabanı kullanımı

Azure SQL veritabanı ,uygulama internete açıldıktan sonra canlı olarak kullanılmaya başlandı.Uygulamanın içindeki her işlem (ürün listeleme,kullanıcı girişi,sepet işlemleri,sipariş verme gibi) aslında arka planda SQL sorguları çalıştırarak Azure'daki bu veritanabında gerçekleşiyor.

6.Sonuç ve Değerlendirme

Tüm bu işlemler sonucunda proje için Azure üzerinde ölçeklenebilir,bulut tabanlı bir e-ticaret altyapısı kurulmuş oldu.Hem uygulama hem de veritabanı canlı ortamda sorunsuz çalışmaktadır.GitHub ile otomatik güncelleme mekanizması sayesinde proje geliştirmeye açık kalmakta, Azure SQL sayesinde tüm veriler güvenli ve merkezi bir şekilde tutulmaktadır.

View (Görünüm) Nedir?

View, kullanıcının doğrudan gördüğü ve etkileşimde bulunduğu katmandır. Yani, web uygulamasında tarayıcıda gördüğünüz **HTML, CSS, JavaScript** içeren sayfaları oluşturur. Ancak View kendiliğinden bir şey üretmez; genellikle **Controller tarafından gönderilen verilerle beslenir**.

📌 View'in Temel Görevleri:

- Veriyi kullanıcıya **sunmak** (örneğin bir ürün listesi, profil bilgileri vs.)
- Sayfa düzenini ve **tasarımını kontrol etmek**
- Veriyi **görselleştirmek** (tablolar, grafikler, formlar vb.)
- Kullanıcının girdiği verileri (formlar, butonlar) Controller'a göndermek

Örnek Senaryo ile Açıklama

Bir e-ticaret sitesi yapıyorsun diyelim:

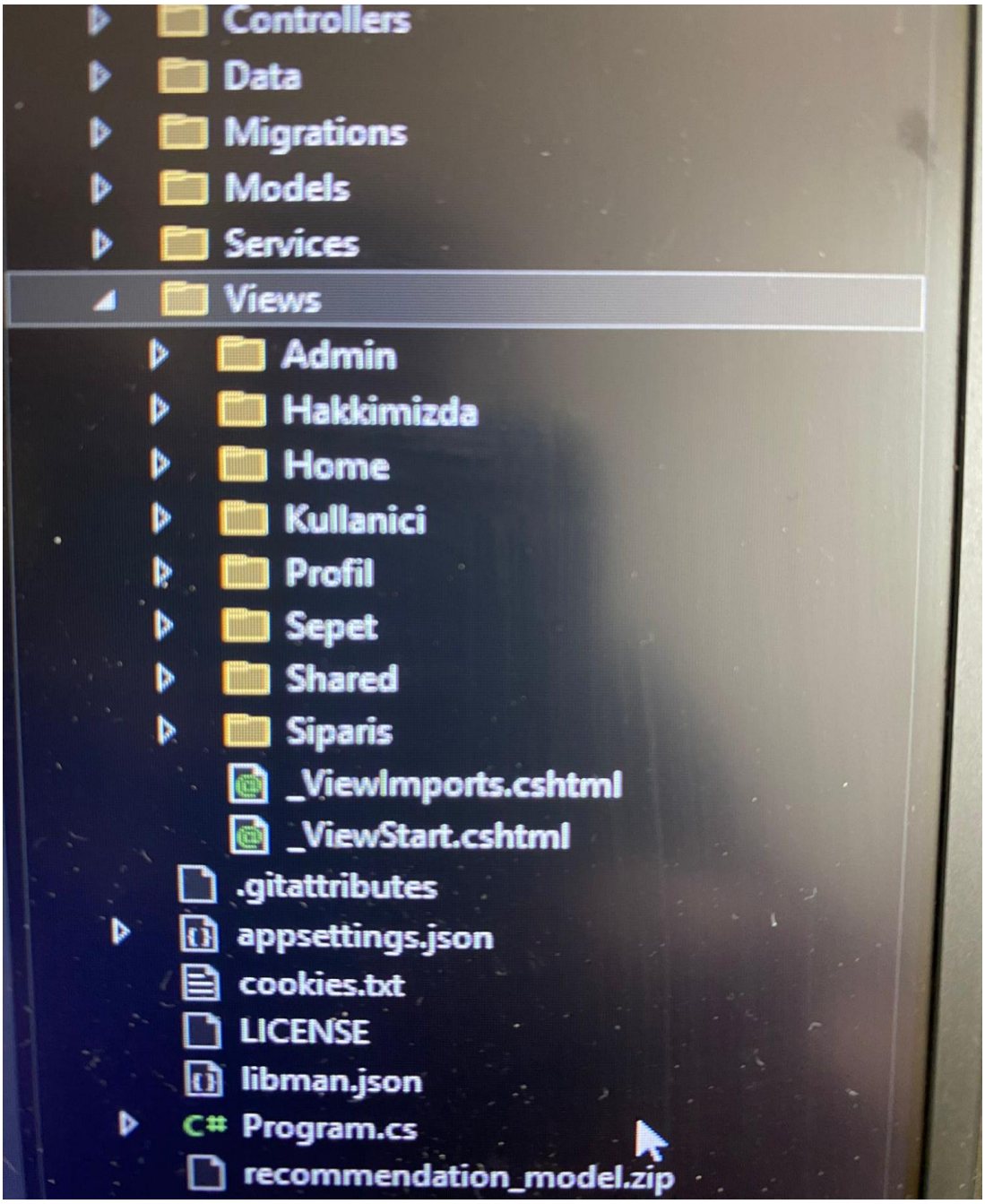
- **Model**: Ürünleri veri tabanından çeker (`Product`, `Category` modelleri olabilir)
- **Controller**: Kullanıcının "/products" sayfasına girdiğini algılar, `Product` modelinden verileri çeker ve View'e yollar
- **View**: Bu ürünleri HTML olarak bir liste içinde kullanıcıya gösterir

```
html
KopyalaDüzenle
<!-- Örnek View (ASP.NET Razor veya benzeri bir yapı
olabilir) -->
<h1>Ürün Listesi</h1>
<ul>
  @foreach (var urun in Model)
  {
      <li>@urun.Adi - @urun.Fiyati TL</li>
  }
</ul>
```

View Katmanında Neler Olmaz?

- **İş mantığı (business logic)** burada yer almamalı.
- Veri doğrudan burada **manipüle edilmemeli**, bu Model ve Controller'ın işidir.
- View'ler mümkün olduğunca **pasif ve sunuma odaklı** olmalıdır.

Projemizde Yer Alan Views Klasörü



Admin Paneli

Klasör: Views/Admin

Admin paneli, site yöneticisinin kullanıcıları, ürünleri, siparişleri ve yapay zeka ayarlarını yönetmesini sağlar. Aşağıda her sayfa hakkında işlevsel özet yer almaktadır.

Kullanıcı Yönetimi Sayfaları:

Sayfa Adı Açıklama

Kullanicilar.cshtml Sistemde kayıtlı tüm kullanıcıları listeler. Yönetici burada kullanıcıları görüntüleyebilir.

KullaniciDetay.cshtml Seçilen bir kullanıcının detay bilgilerini gösterir (e-posta, adres, siparişler vs.).

KullaniciEkle.cshtml Yeni bir kullanıcıyı sisteme eklemek için form içerir.

KullaniciDuzenle.cshtml Var olan bir kullanıcının bilgilerini güncelleme sayfasıdır.

Ürün Yönetimi Sayfaları:

Sayfa Adı Açıklama

Urunler.cshtml Tüm ürünlerin listelendiği ana sayfadır. Arama, filtreleme gibi özellikler içerebilir.

UrunDetay.cshtml Belirli bir ürünün tüm detaylarını görüntüler. Görsel, açıklama, fiyat, stok vs.

UrunEkle.cshtml Yeni bir ürün oluşturmak için form içerir. Zorunlu alanlar: ad, açıklama, stok, fiyat vs.

UrunDuzenle.cshtml Var olan bir ürünü düzenlemek için kullanılır.

UrunSil.cshtml Ürün silme işlemi öncesinde kullanıcıyı uyarır veya onay alır.

Sipariş Yönetimi Sayfaları:

Sayfa Adı Açıklama

Siparisler.cshtml Tüm siparişleri listeler. Kullanıcı adı, tarih, tutar gibi özet bilgiler içerir.

SiparisDetay.cshtml Seçilen bir siparişin detaylarını gösterir: ürünler, kargo adresi, sipariş durumu vs.

Genel Panel Girişi:

Sayfa Adı Açıklama

Index.cshtml Admin panelinin ana sayfası veya kontrol panelidir. Genel istatistikler (toplam sipariş, kullanıcı sayısı, stok durumu vs.) gösterilebilir.

Yapay Zeka Yönetimi:

Sayfa Adı Açıklama

AIYonetimi.cshtml Tavsiye sistemi, öneri algoritmaları veya kullanıcıya özel içerik ayarlarını barındırabilir. Örneğin: model güncelleme, öneri türü seçimi, algoritma değişikliği.

Yönetimsel Yetkinlikler:

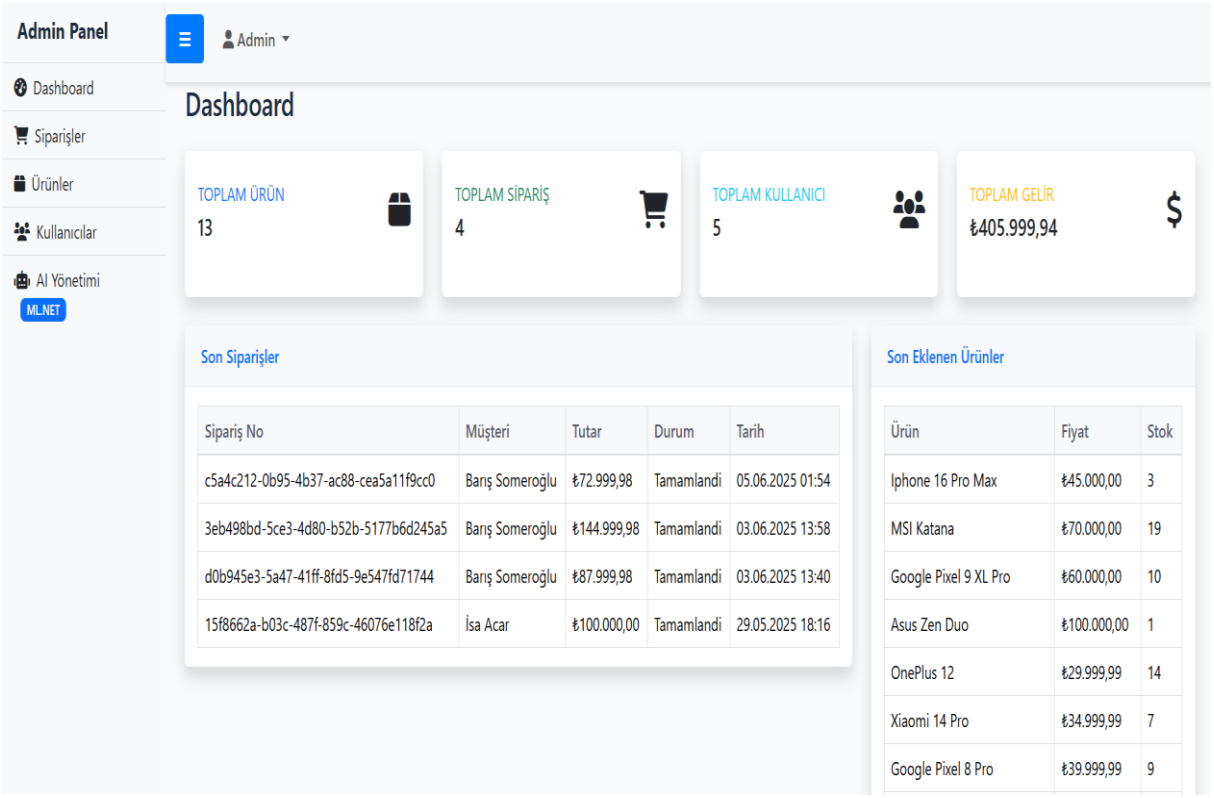
Kullanıcı Ekle / Düzenle / Sil

Ürün Yönetimi

Sipariş Takibi ve İnceleme

Tavsiye Sistemi Yönetimi

Admin Kontrol Paneli



Hakkımızda Sayfası

View Dosyası Bilgileri

Dosya Yolu: Views/Hakkimizda/Index.cshtml

Sayfa Başlığı: Hakkımızda

Kullanıcı Erişimi: Herkese açık (login gerekmez)

Amaç: Firma hakkında genel bilgi, misyon-vizyon, iletişim bilgileri sunmak

Sayfa İçeriği ve Bileşenler

1. Sayfa Başlığı ve Genel Tasarım

Sayfanın tarayıcı başlığı "Hakkımızda" olarak ayarlanmış.

Tüm içerik .card yapısı içinde Bootstrap grid sistemine uygun olarak ortalanmış.

2. Bölüm – Biz Kimiz?

Firma kuruluş bilgileri ve uzmanlık alanları kullanıcıya açıklanıyor.

Firma adı: VANtastic Team

Kuruluş yılı: 2024

3. Bölüm – Misyonumuz

Hızlı ve kaliteli alışveriş deneyimi

Teknolojiye erişimi kolaylaştırmak hedeflenmiş

4. Bölüm – Vizyonumuz

Türkiye'de lider teknoloji e-ticaret platformu olma hedefi

Müşteri memnuniyetine odaklı marka olma vizyonu

5. Bölüm – Değerlerimiz

İki kolonda liste şeklinde sunulmuş

Öne çıkan değerler:

Müşteri memnuniyeti

Kalite

Güvenilirlik

Yenilikçilik

Şeffaflık

Müşteri odaklılık

6. Bölüm – İletişim Bilgileri

Adres, telefon ve e-posta bilgileri gösteriliyor

Font Awesome ikonları ile görsel destek sağlanmış

Stil ve CSS Özellikleri (View içi stil)

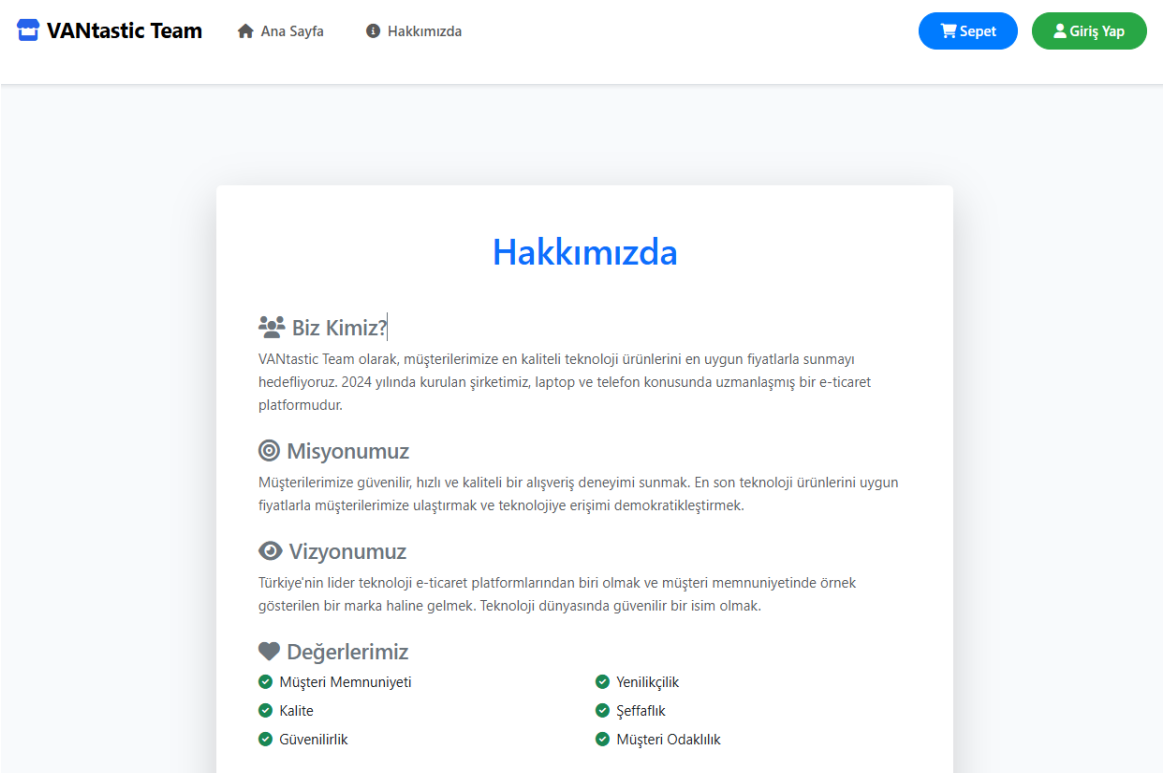
Sayfa Yüksekliği Ayarı: min-height: calc(100vh - 200px); – Sayfanın en az ekran yüksekliği kadar yer kaplaması sağlanır

Kart Hover Efekti: Kart kutusu mouse ile üzerine gelindiğinde yukarı kayarak animasyonlu görünür

Responsive Tasarım: Ekran küçükse (max-width: 768px) yazı boyutları ve padding otomatik küçülür

Buton Hover Efekti: Gölge ve hafif yukarı çıkma efektiyle modern görsellik sağlanır

Bu view sayfası, statik içerikli ama modern bir tasarıma sahip, responsive, kullanıcı dostu bir "Hakkımızda" ekranıdır. Şirketin tanıtım bilgileri detaylı ve görsel olarak etkileyici biçimde sunulmuştur. Kullanıcı yönlendirme (ana sayfaya dönüş) ve iletişim bilgileri de eksiksiz şekilde entegre edilmiştir.



Ana Sayfa (Index.cshtml)

Sayfa Başlığı

Title: Ana Sayfa ViewData üzerinden belirlenmiş:

```
ViewData["Title"] = "Ana Sayfa";
```

Sayfanın Amacı

Bu sayfa, e-ticaret sitesinin ana sayfasıdır. Kullanıcıya:

Kategoriler,

Öne çıkan ürünler,

Yapay zeka destekli kişiselleştirilmiş öneriler (giriş yapmış kullanıcılar için),

Modern, kullanıcı dostu ve görsel açıdan zengin bir arayüz sunar.

Sayfa Bileşenleri

1. Hero Section

Sayfanın üst kısmında kullanıcıya hoş geldin mesajı verir:

```
<h1 class="display-4">Hoş Geldiniz</h1>
<p class="lead">En kaliteli ürünler, en uygun fiyatlarla burada!</p>
```

2. Kategoriler

3 buton ve 2 kategori kartı içerir:

Butonlar: "Tüm Ürünler", "Laptoplar", "Telefonlar"

Kartlar: Görsel + açıklama + “Keşfet” butonu

URL yönlendirmeleri HomeController > Index aksiyonuna yönlendirilir:

```
@Url.Action("Index", "Home", new { kategori = "Laptop" })
```

3. Yapay Zeka Destekli Öneriler

Giriş yapmış kullanıcılara özel öneriler:

ViewBag.PersonalizedRecommendations boş değilse gösterilir.

Ürünler küçük kartlar şeklinde gösterilir.

Her kart:

Görsel,Ürün adı , Fiyat, Detay butonu, Sepete ekle butonu içerir.

4. Öne Çıkan Ürünler

@model üzerinden gelen tüm ürünleri listeler.

Filtreleme yapıldıysa @ViewBag.SecilenKategori başlık olarak yazılır.

Kart yapısı:

Ürün görseli, Adı, Açıklaması, Fiyatı, Stok miktarı, Detay ve sepete ekle butonları bulunur.

Model ve ViewBag Kullanımı

View modeli:

IEnumerable<ETicaretSitesi.Models.Urun>

ViewBag kullanımları:

ViewBag.PersonalizedRecommendations: Liste halinde önerilen ürünler

ViewBag.SecilenKategori: Filtrelenmiş kategori adı

Stil (CSS - @section Styles)

Sayfa içinde yer alan CSS kodlarıyla:

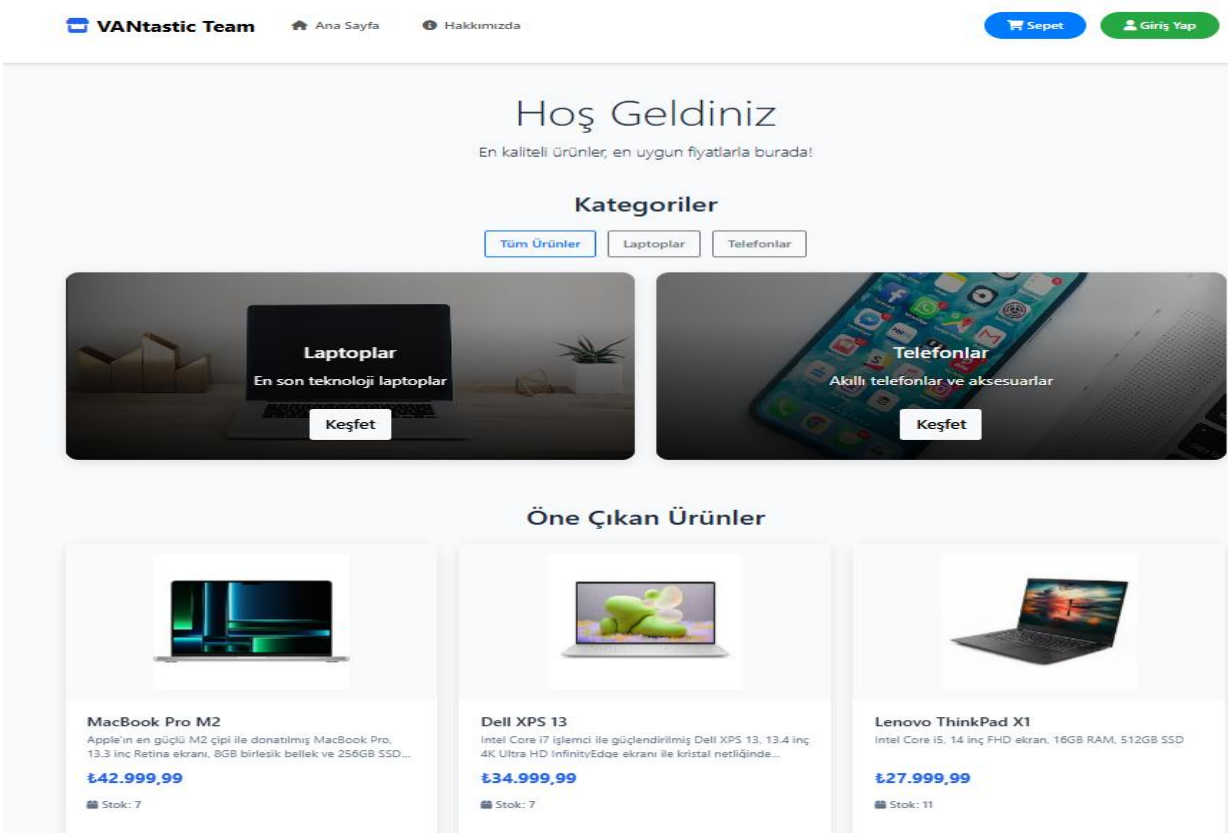
Kategori kartları: Hover animasyonu, overlay, görsel büyütme

Ürün kartları: Gölge efekti, animasyon, sabit yükseklik, responsive tasarım

Yapay Zeka öneri kartları: Küçük kartlar, hover efekti, badge'li tasarım

Genel animasyonlar: @keyframes fadeInUp ile yumuşak geçişler

Butonlar: Renk geçişleri, ikonlar ile zenginleştirilmiş butonlar



KULLANICI GİRİŞ SAYFASI

1. Sayfa Amacı

Bu sayfa, kullanıcıların var olan hesapları ile sisteme giriş yapmalarını sağlar. Aynı zamanda yönetici (admin) girişine yönlendirme ve yeni kullanıcılar için kayıt olma bağlantısı içerir. Sayfa, kullanıcı deneyimini artırmak için görsellik ve animasyonlarla zenginleştirilmiştir.

2. Sayfa Tasarımı

Yapı

Sayfa iki ana bölüme ayrılmıştır:

Sol Panel: Tanıtıcı bir görsel alan, slogan, avantaj bilgileri.

Sağ Panel: Giriş formu ve yönlendirme butonları.

Responsive Tasarım

@media sorguları ile 991px ve 575px altı ekranlarda form alanının padding ve font boyutları optimize edilmiştir.

Mobil uyumluluk yüksektir.

3. Giriş Formu

Form Özellikleri

Giriş formu POST metoduyla Giris action metoduna gönderilir.

Kullanıcıdan aşağıdaki bilgiler alınır:

E-posta (input type="email")

Şifre (input type="password")

Beni Hatırla checkbox

Şifremi Unuttum? ve Kayıt Olun linkleri yer alır.

Giriş yapan kullanıcıya özel mesajlar (başarılı/başarısız giriş) TempData ile gösterilir.

Validation

asp-validation-summary="ModelOnly" kullanılarak sunucu taraflı hata mesajları ekrana yansıtılır.

_ValidationScriptsPartial dahil edilmiştir; istemci tarafı doğrulama (client-side validation) aktiftir.

4. Geri Bildirim Sistemi

TempData["Mesaj"]: Başarılı işlemler için yeşil (success) uyarı kutusu.

TempData["Hata"]: Hatalı işlemler için kırmızı (danger) uyarı kutusu.

Kullanıcıya işlem sonrası net ve anlaşılır geri bildirim sağlanır.

5. Stil ve Görsel Yapı

CSS Özellikleri

Sayfa genelinde modern, yuvarlatılmış köşeler, gölgelendirme ve gradient renk geçişleri kullanılmıştır.

Form ve butonlar hover/focus durumlarına duyarlıdır.

Görseller ve ikonlar sayfaya zenginlik katar.

Sol Panel Özellikleri

Arka plan olarak Unsplash görseli kullanılmış.

Beyaz yazılar ve ikonlarla dikkat çekici bilgi sunumu yapılmış.

Ücretsiz kargo, güvenli alışveriş ve müşteri desteği gibi avantajlar ikonlarla desteklenmiş.

6. Animasyon ve Etkileşim

Giriş formu ve özellikler slideInUp ve fadeInLeft animasyonlarıyla canlandırılmış.

Form etkileşimi:

Odaklanılan input alanları vurgulanır.

Submit işlemi sırasında butonda spinner animasyonu gösterilir ve buton pasifleştirilir.

Bu sayede kullanıcıya hem görsel hem işlevsel geri bildirim sağlanır.

7. Güvenlik ve Erişilebilirlik

Email input için type="email" belirlenmiş, temel doğrulama sağlanmış.

required özelliği sayesinde kullanıcı boş bırakırsa istemci tarafında hata alır.

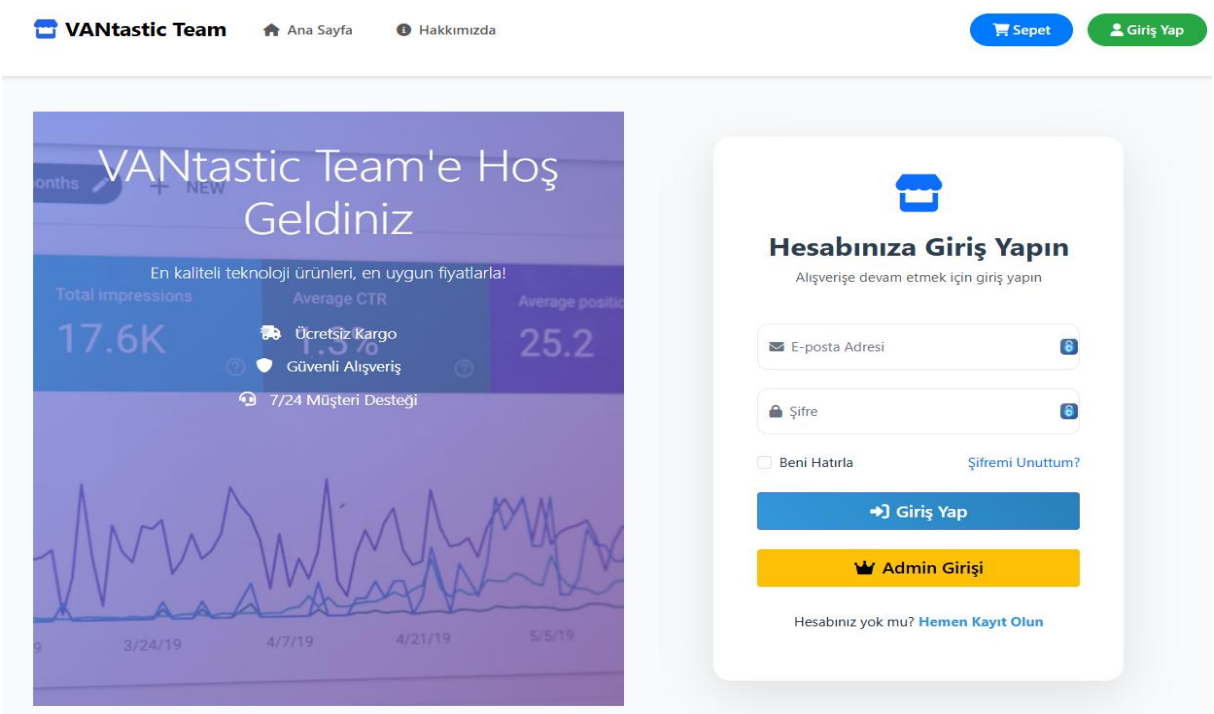
asp-validation-summary sunucu tarafı hataları gösterir.

Geliştirme Önerisi: CSRF güvenliği için <form> içerisine @Html.AntiForgeryToken() eklenmesi önerilir.

8. Yönetici Girişi

Kullanıcılar için ayrı bir butonla Admin Giriş sayfasına yönlendirme yapılmaktadır:

```
<a asp-action="AdminGiris" class="btn btn-warning btn-lg">
  <i class="fas fa-crown me-2"></i>Admin Girişi
</a>
```



Sepet Sayfası

Sayfa Adı:

Sepetim.cshtml (Razor View)

Sayfa Amacı:

Kullanıcının sepete eklediği ürünleri listelemek, adet güncellemek, ürün silmek ve siparişi onaylama işlemlerini gerçekleştirmesini sağlamak.

Özellikler ve İşlevler:

1. Temel Sepet Listeleme

Model.SepetUrunleri üzerinden sepet verileri listeleniyor.

Her ürün için: Adı, fiyatı, adedi, toplam tutarı ve görseli gösteriliyor.

Ürün resmi varsa görsel, yoksa alternatif simge gösterimi sağlanıyor.

2. Adet Güncelleme

Kullanıcı adet bilgisini input ile değiştirebiliyor.

Her satırda güncelleme işlemi için UrunGuncelle isimli POST formu yer alıyor.

3. Ürün Silme

Ürünü sepetten silmek için UrunSil eylemine yönlendirilen POST formu kullanılıyor.

4. Toplam Tutar Hesaplama

Sepetteki her ürünün toplamı (Fiyat * Adet) hesaplanıp en altta sepet toplamı olarak gösteriliyor.

5. Siparişi Onaylama

SiparisOnayla eylemine yönlendiren buton ile kullanıcı alışverişi tamamlayabiliyor.

6. Yapay Zeka Öneri Sistemi

Model.OnerilenUrunler içeriğiyle çalışan öneri algoritması.

Sepetteki ürünlere göre önerilen ürünler kullanıcıya kart şeklinde gösteriliyor.

Ürün açıklaması, stok durumu, fiyatı ve "Sepete Ekle" butonu yer alıyor.

Kullanıcı Deneyimini Artıran Öğeler:

Animasyonlar ve Hover Efektleri: CSS animasyonları ile sayfa geçişi ve ürün kartlarına görsel derinlik kazandırılmış.

Responsive Tasarım: Bootstrap sınıfları ile mobil uyumluluk sağlanmış.

Kullanıcı Geri Bildirimleri: TempData kullanılarak hata ve başarı mesajları gösteriliyor.

Yapay Zeka Öneri Bölümü:

Ürün tipi ve sepet içeriğine göre önerilen ürünler listeleniyor.

Her öneri kartında: Ürün görseli, Kategori (badge olarak), Açıklama (60 karakter sınır), Fiyat ve stok bilgisi, Sepete ekleme butonu bulunur.