```
========================================
       ETicaret E-Commerce Project
       Complete Source Code Compilation
========================================

Generated on: 2025-01-16
Project Path: /home/serhat/RiderProjects/ETicaret
Framework: .NET 9.0 (ASP.NET Core MVC)
Database: MySQL with Entity Framework Core

========================================
1. PROJECT CONFIGURATION FILES
========================================

------------- ETicaret.csproj -------------
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <UICulture>tr</UICulture>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="BCrypt.Net-Next" Version="4.0.3" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Relational"
Version="9.0.7" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools"
Version="9.0.7">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Pomelo.EntityFrameworkCore.MySql" Version="9.0.0-
rc.1.efcore.9.0.0" />
  </ItemGroup>

</Project>

------------- Program.cs -------------
using Microsoft.EntityFrameworkCore;
using ETicaret.Data;
using ETicaret.Services;
using ETicaret.ModelBinders;
using MySqlConnector;
using System.Globalization;

var builder = WebApplication.CreateBuilder(args);

// Configure supported cultures (Turkish and English)
var supportedCultures = new List<CultureInfo>
{
    new CultureInfo("tr-TR"),
    new CultureInfo("en-US")
};

// Add services to the container.
builder.Services.AddControllersWithViews(options =>
{
    // Add custom model binder for decimal values to handle Turkish locale
    options.ModelBinderProviders.Insert(0, new DecimalModelBinderProvider());
});
```

```
// Add localization services
builder.Services.AddLocalization(options =>
{
    options.ResourcesPath = "Resources";
});

builder.Services.Configure<RequestLocalizationOptions>(options =>
{
    // Use Turkish for UI but invariant culture for data parsing to avoid
decimal parsing issues
    options.DefaultRequestCulture = new
Microsoft.AspNetCore.Localization.RequestCulture(
        culture: "en-US", // For data parsing (uses period as decimal separator)
        uiCulture: "tr-TR" // For UI display
    );
    options.SupportedCultures = new[] { new CultureInfo("en-US"), new
CultureInfo("tr-TR") };
    options.SupportedUICultures = supportedCultures;
    options.RequestCultureProviders.Clear();
    options.RequestCultureProviders.Add(new
Microsoft.AspNetCore.Localization.AcceptLanguageHeaderRequestCultureProvider());
});

// Add authentication services
builder.Services.AddAuthentication("CookieAuth")
    .AddCookie("CookieAuth", options =>
    {
        options.LoginPath = "/Account/Login";
        options.LogoutPath = "/Account/Logout";
        options.AccessDeniedPath = "/Account/AccessDenied";
        options.ExpireTimeSpan = TimeSpan.FromMinutes(60);
        options.SlidingExpiration = true;
    });

builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("AdminOnly", policy =>
        policy.RequireClaim("IsAdmin", "true"));
});

// Add session support for shopping cart
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

// Add payment service
builder.Services.AddScoped<IPaymentService, PaymentService>();

// Add Entity Framework and MySQL
var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
{
    options.UseMySql(connectionString, new MySqlServerVersion(new Version(8, 0,
36)))
           .EnableSensitiveDataLogging(builder.Environment.IsDevelopment())
           .EnableDetailedErrors(builder.Environment.IsDevelopment());
});

var app = builder.Build();
```

```
// Ensure database is created and apply migrations
try
{
    using (var scope = app.Services.CreateScope())
    {
        var context =
scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();
        var logger =
scope.ServiceProvider.GetRequiredService<ILogger<Program>>();

        logger.LogInformation("Attempting to connect to database...");

        // Test connection first
        await context.Database.CanConnectAsync();
        logger.LogInformation("Database connection test successful!");

        // Ensure database and tables are created
        await context.Database.EnsureCreatedAsync();
        logger.LogInformation("Database schema creation successful!");

        // Test a simple query
        var productCount = await context.Products.CountAsync();
        logger.LogInformation("Database ready! Found {ProductCount} products.",
productCount);
    }
}
catch (MySqlConnector.MySqlException mysqlEx)
{
    var logger = app.Services.GetRequiredService<ILogger<Program>>();
    logger.LogError(mysqlEx, "MySQL specific error occurred: {ErrorCode} -
{Error}", mysqlEx.ErrorCode, mysqlEx.Message);
    logger.LogError("Connection String: {ConnectionString}",
connectionString?.Replace("Password=", "Password=***"));
    throw;
}
catch (Exception ex)
{
    var logger = app.Services.GetRequiredService<ILogger<Program>>();
    logger.LogError(ex, "Database connection failed: {Error}", ex.Message);
    throw;
}

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
else
{
    app.UseDeveloperExceptionPage();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

// Use request localization
app.UseRequestLocalization();

app.UseRouting();
app.UseAuthentication();
```

```
app.UseAuthorization();
app.UseSession();

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

try
{
    app.Run();
}
catch (System.IO.IOException ioEx)
{

    var logger = app.Services.GetRequiredService<ILogger<Program>>();
    logger.LogError(ioEx, "IOException occurred during application startup:
{Message}", ioEx.Message);
    logger.LogError("IOException Details - HResult: {HResult}, Source:
{Source}", ioEx.HResult, ioEx.Source);
    logger.LogError("Stack Trace: {StackTrace}", ioEx.StackTrace);

    if (ioEx.InnerException != null)
    {
        logger.LogError("Inner Exception: {InnerException}",
ioEx.InnerException.Message);
    }

    // Common IOException scenarios
    if (ioEx.Message.Contains("address already in use") ||
ioEx.Message.Contains("Address already in use"))
    {
        logger.LogError("Port binding issue detected. Try running on a different
port using: dotnet run --urls \"http://localhost:5088\"");
    }
    else if (ioEx.Message.Contains("permission") ||
ioEx.Message.Contains("access"))
    {
        logger.LogError("File permission issue detected. Check file/directory
permissions.");
    }
    else if (ioEx.Message.Contains("path") ||
ioEx.Message.Contains("directory"))
    {
        logger.LogError("Path resolution issue detected. Current working
directory: {WorkingDirectory}", Directory.GetCurrentDirectory());
        logger.LogError("Application base directory: {BaseDirectory}",
AppContext.BaseDirectory);
    }

    throw; // Re-throw to maintain normal exception handling
}
catch (Exception ex)
{
    var logger = app.Services.GetRequiredService<ILogger<Program>>();
    logger.LogError(ex, "Unexpected error during application startup:
{Message}", ex.Message);
    throw;
}

------------- appsettings.json -------------
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=localhost;Database=ETicaretDb_Dev;User=root;Password=Baba5249%;Port=3306
```

```
;Connect Timeout=60;SslMode=None;"

  },
  "DetailedErrors": true,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.Extensions.Hosting": "Debug"
    }
  }
}

------------- appsettings.Development.json -------------
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=localhost;Database=ETicaretDb_Dev;User=root;Password=Baba5249%;Port=3306
;Connect Timeout=60;SslMode=None;"

  },
  "DetailedErrors": true,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.Extensions.Hosting": "Debug"
    }
  }
}

------------- Properties/launchSettings.json -------------
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "http://localhost:5080",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "https": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "https://localhost:7042;http://localhost:5080",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}

========================================
2. MODELS & DATA LAYER
========================================

------------- Models/User.cs -------------
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

```
namespace ETicaret.Models
{
    public class User
    {
        public int Id { get; set; }

        [Required]
        [StringLength(100)]
        public string FirstName { get; set; } = string.Empty;

        [Required]
        [StringLength(100)]
        public string LastName { get; set; } = string.Empty;

        [Required]
        [EmailAddress]
        [StringLength(200)]
        public string Email { get; set; } = string.Empty;

        [Required]
        [StringLength(255)]
        public string PasswordHash { get; set; } = string.Empty;

        [StringLength(20)]
        public string? Phone { get; set; }

        [StringLength(500)]
        public string? Address { get; set; }

        [StringLength(100)]
        public string? City { get; set; }

        [StringLength(20)]
        public string? PostalCode { get; set; }

        public bool IsActive { get; set; } = true;

        public bool IsAdmin { get; set; } = false;

        [Column(TypeName = "datetime")]
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        [Column(TypeName = "datetime")]
        public DateTime? UpdatedAt { get; set; }

        // Navigation properties
        public virtual ICollection<Order> Orders { get; set; } = new
List<Order>();
        public virtual ICollection<CartItem> CartItems { get; set; } = new
List<CartItem>();
    }
}

------------- Models/Product.cs -------------
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ETicaret.Models
{
    public class Product
    {
        public int Id { get; set; }

        [Required]
```

```
        [StringLength(100)]
        public string Name { get; set; } = string.Empty;

        [StringLength(500)]
        public string? Description { get; set; }

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

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

        public int Stock { get; set; }

        [Required]
        public int CategoryId { get; set; }

        [StringLength(200)]
        public string? ImageUrl { get; set; }

        public bool IsActive { get; set; } = true;

        public bool IsFeatured { get; set; } = false;

        [Column(TypeName = "datetime")]
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        [Column(TypeName = "datetime")]
        public DateTime? UpdatedAt { get; set; }

        // Navigation properties
        public virtual Category Category { get; set; } = null!;
        public virtual ICollection<OrderItem> OrderItems { get; set; } = new
List<OrderItem>();
        public virtual ICollection<CartItem> CartItems { get; set; } = new
List<CartItem>();
    }
}

------------- Models/Category.cs -------------
using System.ComponentModel.DataAnnotations;

namespace ETicaret.Models
{
    public class Category
    {
        public int Id { get; set; }

        [Required]
        [StringLength(100)]
        public string Name { get; set; } = string.Empty;

        [StringLength(500)]
        public string Description { get; set; } = string.Empty;

        public string? ImageUrl { get; set; }

        public bool IsActive { get; set; } = true;

        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        // Navigation property
        public ICollection<Product> Products { get; set; } = new
```

```
List<Product>();
    }
}

------------- Models/CartItem.cs -------------
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ETicaret.Models
{
    public class CartItem
    {
        public int Id { get; set; }

        [Required]
        public int UserId { get; set; }

        [Required]
        public int ProductId { get; set; }

        [Required]
        public int Quantity { get; set; }

        [Column(TypeName = "datetime")]
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        // Navigation properties
        public virtual User User { get; set; } = null!;
        public virtual Product Product { get; set; } = null!;
    }
}

------------- Models/Order.cs -------------
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel;
using ETicaret.Resources;

namespace ETicaret.Models
{
    public class Order
    {
        public int Id { get; set; }

        [Required]
        public int UserId { get; set; }

        [Required]
        [StringLength(100)]
        public string OrderNumber { get; set; } = string.Empty;

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

        [Required]
        public OrderStatus Status { get; set; } = OrderStatus.Pending;

        [StringLength(500)]
        public string? ShippingAddress { get; set; }

        [StringLength(100)]
        public string? ShippingCity { get; set; }

        [StringLength(20)]
```

```
        public string? ShippingPostalCode { get; set; }

        [StringLength(100)]
        public string? TransactionId { get; set; }

        [Column(TypeName = "datetime")]
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        [Column(TypeName = "datetime")]
        public DateTime? UpdatedAt { get; set; }

        // Navigation properties
        public virtual User User { get; set; } = null!;
        public virtual ICollection<OrderItem> OrderItems { get; set; } = new
List<OrderItem>();
    }

    public enum OrderStatus
    {
        [Description("OrderStatus_Pending")]
        Pending = 0,

        [Description("OrderStatus_Processing")]
        Processing = 1,

        [Description("OrderStatus_Shipped")]
        Shipped = 2,

        [Description("OrderStatus_Delivered")]
        Delivered = 3,

        [Description("OrderStatus_Cancelled")]
        Cancelled = 4
    }
}

------------- Models/OrderItem.cs -------------
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ETicaret.Models
{
    public class OrderItem
    {
        public int Id { get; set; }

        [Required]
        public int OrderId { get; set; }

        [Required]
        public int ProductId { get; set; }

        [Required]
        public int Quantity { get; set; }

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

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

        // Navigation properties
        public virtual Order Order { get; set; } = null!;
        public virtual Product Product { get; set; } = null!;
```

```
    }
}

------------- Data/ApplicationDbContext.cs -------------
using Microsoft.EntityFrameworkCore;
using ETicaret.Models;

namespace ETicaret.Data
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
            : base(options)
        {
        }

        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<User> Users { get; set; }
        public DbSet<Order> Orders { get; set; }
        public DbSet<OrderItem> OrderItems { get; set; }
        public DbSet<CartItem> CartItems { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // Category configuration
            modelBuilder.Entity<Category>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.Name).IsRequired().HasMaxLength(100);
                entity.Property(e => e.Description).HasMaxLength(500);
                entity.Property(e => e.ImageUrl).HasMaxLength(200);
                entity.Property(e => e.CreatedAt)
                    .HasColumnType("datetime")
                    .HasDefaultValueSql("CURRENT_TIMESTAMP");
            });

            // Product configuration
            modelBuilder.Entity<Product>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.Name).IsRequired().HasMaxLength(100);
                entity.Property(e => e.Description).HasMaxLength(500);
                entity.Property(e => e.Price).HasColumnType("decimal(18,2)");
                entity.Property(e =>
e.DiscountPrice).HasColumnType("decimal(18,2)");
                entity.Property(e => e.ImageUrl).HasMaxLength(200);
                entity.Property(e => e.CreatedAt)
                    .HasColumnType("datetime")
                    .HasDefaultValueSql("CURRENT_TIMESTAMP");
                entity.Property(e => e.UpdatedAt).HasColumnType("datetime");

                // Foreign key relationship
                entity.HasOne(e => e.Category)
                    .WithMany(e => e.Products)
                    .HasForeignKey(e => e.CategoryId)
                    .OnDelete(DeleteBehavior.Restrict);
            });

            // User configuration
            modelBuilder.Entity<User>(entity =>
```

```
            {
                entity.HasKey(e => e.Id);
                entity.Property(e =>
e.FirstName).IsRequired().HasMaxLength(100);
                entity.Property(e => e.LastName).IsRequired().HasMaxLength(100);
                entity.Property(e => e.Email).IsRequired().HasMaxLength(200);
                entity.Property(e =>
e.PasswordHash).IsRequired().HasMaxLength(255);
                entity.Property(e => e.Phone).HasMaxLength(20);
                entity.Property(e => e.Address).HasMaxLength(500);
                entity.Property(e => e.City).HasMaxLength(100);
                entity.Property(e => e.PostalCode).HasMaxLength(20);
                entity.Property(e => e.CreatedAt)
                    .HasColumnType("datetime")
                    .HasDefaultValueSql("CURRENT_TIMESTAMP");
                entity.Property(e => e.UpdatedAt).HasColumnType("datetime");

                // Unique email constraint
                entity.HasIndex(e => e.Email).IsUnique();
            });

            // Order configuration
            modelBuilder.Entity<Order>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e =>
e.OrderNumber).IsRequired().HasMaxLength(100);
                entity.Property(e =>
e.TotalAmount).HasColumnType("decimal(18,2)");
                entity.Property(e => e.ShippingAddress).HasMaxLength(500);
                entity.Property(e => e.ShippingCity).HasMaxLength(100);
                entity.Property(e => e.ShippingPostalCode).HasMaxLength(20);
                entity.Property(e => e.CreatedAt)
                    .HasColumnType("datetime")
                    .HasDefaultValueSql("CURRENT_TIMESTAMP");
                entity.Property(e => e.UpdatedAt).HasColumnType("datetime");

                // Foreign key relationship
                entity.HasOne(e => e.User)
                    .WithMany(e => e.Orders)
                    .HasForeignKey(e => e.UserId)
                    .OnDelete(DeleteBehavior.Restrict);
            });

            // OrderItem configuration
            modelBuilder.Entity<OrderItem>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e =>
e.UnitPrice).HasColumnType("decimal(18,2)");
                entity.Property(e =>
e.TotalPrice).HasColumnType("decimal(18,2)");

                // Foreign key relationships
                entity.HasOne(e => e.Order)
                    .WithMany(e => e.OrderItems)
                    .HasForeignKey(e => e.OrderId)
                    .OnDelete(DeleteBehavior.Cascade);

                entity.HasOne(e => e.Product)
                    .WithMany(e => e.OrderItems)
                    .HasForeignKey(e => e.ProductId)
                    .OnDelete(DeleteBehavior.Restrict);
            });
```

```
            // CartItem configuration
            modelBuilder.Entity<CartItem>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.CreatedAt)
                    .HasColumnType("datetime")
                    .HasDefaultValueSql("CURRENT_TIMESTAMP");

                // Foreign key relationships
                entity.HasOne(e => e.User)
                    .WithMany(e => e.CartItems)
                    .HasForeignKey(e => e.UserId)
                    .OnDelete(DeleteBehavior.Cascade);

                entity.HasOne(e => e.Product)
                    .WithMany(e => e.CartItems)
                    .HasForeignKey(e => e.ProductId)
                    .OnDelete(DeleteBehavior.Cascade);
            });

            // Seed data
            SeedData(modelBuilder);
        }

        private void SeedData(ModelBuilder modelBuilder)
        {
            // Kategori verilerini ekle
            var categories = CategorySeedData.GetCategories();
            for (int i = 0; i < categories.Length; i++)
            {
                categories[i].Id = i + 1;
                categories[i].CreatedAt = DateTime.UtcNow;
            }
            modelBuilder.Entity<Category>().HasData(categories);

            // Ürün verilerini ekle
            modelBuilder.Entity<Product>().HasData(
                new Product
                {
                    Id = 1,
                    Name = "Dizüstü Bilgisayar Pro 15",
                    Description = "16GB RAM ve 512GB SSD'li yüksek performanslı
dizüstü bilgisayar",
                    Price = 1299.99m,
                    DiscountPrice = 1199.99m,
                    Stock = 10,
                    CategoryId = 1,
                    ImageUrl = "/images/products/laptop1.jpg",
                    IsActive = true,
                    IsFeatured = true,
                    CreatedAt = DateTime.UtcNow
                },
                new Product
                {
                    Id = 2,
                    Name = "Kablosuz Kulaklık",
                    Description = "Gürültü engelleme özellikli premium kablosuz
kulaklık",
                    Price = 29.99m,
                    Stock = 50,
                    CategoryId = 1,
                    ImageUrl = "/images/products/headphones1.jpg",
                    IsActive = true,
```

```
                    IsFeatured = false,
                    CreatedAt = DateTime.UtcNow
                },
                new Product
                {
                    Id = 3,
                    Name = "Dijital Fotoğraf Makinesi",
                    Description = "24MP sensörlü profesyonel dijital fotoğraf
makinesi",
                    Price = 19.99m,
                    Stock = 100,
                    CategoryId = 1,
                    ImageUrl = "/images/products/camera1.jpg",
                    IsActive = true,
                    IsFeatured = true,
                    CreatedAt = DateTime.UtcNow
                },
                new Product
                {
                    Id = 4,
                    Name = "Akıllı Saat",
                    Description = "Kalp ritmi monitörlü fitness takip akıllı
saati",
                    Price = 39.99m,
                    Stock = 25,
                    CategoryId = 1,
                    ImageUrl = "/images/products/watch1.jpg",
                    IsActive = true,
                    IsFeatured = false,
                    CreatedAt = DateTime.UtcNow
                }
            );

            // Yönetici kullanıcısı verilerini ekle (Parola: admin123)
            modelBuilder.Entity<User>().HasData(
                new User
                {
                    Id = 1,
                    FirstName = "Yönetici",
                    LastName = "Kullanıcı",
                    Email = "yonetici@eticaret.com",
                    PasswordHash =
BCrypt.Net.BCrypt.HashPassword("admin123"), // admin123
                    IsActive = true,
                    IsAdmin = true,
                    CreatedAt = DateTime.UtcNow
                }
            );
        }
    }
}

------------- Data/CategorySeedData.cs -------------
using ETicaret.Models;

namespace ETicaret.Data
{
    public static class CategorySeedData
    {
        public static Category[] GetCategories()
        {
            return new[]
            {
                new Category
```

```
                {
                    Name = "Elektronik",
                    Description = "Telefon, laptop, tablet ve diğer elektronik
cihazlar",
                    ImageUrl = "/images/categories/electronics.jpg",
                    IsActive = true
                },
                new Category
                {
                    Name = "Giyim",
                    Description = "Erkek, kadın ve çocuk giyim ürünleri",
                    ImageUrl = "/images/categories/clothing.jpg",
                    IsActive = true
                },
                new Category
                {
                    Name = "Kitap",
                    Description = "Roman, bilim, eğitim ve referans kitaplar",
                    ImageUrl = "/images/categories/books.jpg",
                    IsActive = true
                },
                new Category
                {
                    Name = "Ev & Bahçe",
                    Description = "Mobilya, dekorasyon ve bahçe ürünleri",
                    ImageUrl = "/images/categories/home-garden.jpg",
                    IsActive = true
                },
                new Category
                {
                    Name = "Spor",
                    Description = "Spor aletleri, giyim ve aksesuarlar",
                    ImageUrl = "/images/categories/sports.jpg",
                    IsActive = true
                }
            };
        }
    }
}

========================================
3. CONTROLLERS
========================================

------------- Controllers/HomeController.cs -------------
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ETicaret.Data;
using ETicaret.ViewModels;

namespace ETicaret.Controllers
{
    public class HomeController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly ILogger<HomeController> _logger;
        private readonly IWebHostEnvironment _webHostEnvironment;

        public HomeController(ApplicationDbContext context,
ILogger<HomeController> logger, IWebHostEnvironment webHostEnvironment)
        {
            _context = context;
            _logger = logger;
            _webHostEnvironment = webHostEnvironment;
```

```
        }

        public async Task<IActionResult> Index()
        {
            var viewModel = new HomeViewModel
            {
                FeaturedProducts = await _context.Products
                    .Include(p => p.Category)
                    .Where(p => p.IsActive && p.IsFeatured)
                    .Take(8)
                    .ToListAsync(),

                NewProducts = await _context.Products
                    .Include(p => p.Category)
                    .Where(p => p.IsActive)
                    .OrderByDescending(p => p.CreatedAt)
                    .Take(8)
                    .ToListAsync(),

                Categories = await _context.Categories
                    .Where(c => c.IsActive)
                    .ToListAsync()
            };

            return View(viewModel);
        }


        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View();
        }
    }
}

------------- Controllers/ProductsController.cs -------------
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ETicaret.Data;
using ETicaret.ViewModels;
using ETicaret.Models;

namespace ETicaret.Controllers
{
    public class ProductsController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly ILogger<ProductsController> _logger;

        public ProductsController(ApplicationDbContext context,
ILogger<ProductsController> logger)
        {
            _context = context;
            _logger = logger;
        }

        public async Task<IActionResult> Index(int? categoryId, string? search,
```

```
int page = 1, int pageSize = 12)
        {
            var query = _context.Products
                .Include(p => p.Category)
                .Where(p => p.IsActive);

            // Filter by category
            if (categoryId.HasValue)
            {
                query = query.Where(p => p.CategoryId == categoryId.Value);
            }

            // Filter by search term
            if (!string.IsNullOrWhiteSpace(search))
            {
                query = query.Where(p => p.Name.Contains(search) ||
                                         p.Description!.Contains(search));
            }

            var totalProducts = await query.CountAsync();
            var totalPages = (int)Math.Ceiling(totalProducts /
(double)pageSize);

            var products = await query
                .OrderBy(p => p.Name)
                .Skip((page - 1) * pageSize)
                .Take(pageSize)
                .ToListAsync();

            var categories = await _context.Categories
                .Where(c => c.IsActive)
                .ToListAsync();

            var viewModel = new ProductListViewModel
            {
                Products = products,
                Categories = categories,
                SelectedCategoryId = categoryId,
                SearchTerm = search,
                CurrentPage = page,
                TotalPages = totalPages,
                PageSize = pageSize
            };

            return View(viewModel);
        }

        public async Task<IActionResult> Details(int id)
        {
            var product = await _context.Products
                .Include(p => p.Category)
                .FirstOrDefaultAsync(p => p.Id == id && p.IsActive);

            if (product == null)
            {
                return NotFound();
            }

            // Get related products from the same category
            ViewBag.RelatedProducts = await _context.Products
                .Include(p => p.Category)
                .Where(p => p.CategoryId == product.CategoryId &&
                           p.Id != product.Id &&
                           p.IsActive)
```

```
                .Take(4)
                .ToListAsync();

            return View(product);
        }
    }
}

------------- Controllers/AccountController.cs -------------
[First 50 lines shown due to length - full content continues with authentication
logic, login, logout, and registration methods]

using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;
using ETicaret.Data;
using ETicaret.ViewModels;

namespace ETicaret.Controllers
{
    public class AccountController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly ILogger<AccountController> _logger;

        public AccountController(ApplicationDbContext context,
ILogger<AccountController> logger)
        {
            _context = context;
            _logger = logger;
        }

        [HttpGet]
        public IActionResult Login(string? returnUrl = null)
        {
            ViewBag.ReturnUrl = returnUrl;
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(LoginViewModel model, string?
returnUrl = null)
        {
            ViewBag.ReturnUrl = returnUrl;

            if (!ModelState.IsValid)
            {
                return View(model);
            }

            try
            {
                // Find user by email
                var user = await _context.Users
                    .FirstOrDefaultAsync(u => u.Email == model.Email &&
u.IsActive);

                if (user == null)
                {
                    ModelState.AddModelError("", "Geçersiz giriş bilgileri.");
                    return View(model);
```

```
                }

                // Verify password (using BCrypt)
                if (!BCrypt.Net.BCrypt.Verify(model.Password,
user.PasswordHash))
                {
                    ModelState.AddModelError("", "Geçersiz giriş bilgileri.");
                    return View(model);
                }

                // Create claims and authentication logic continues...
                // [Remaining authentication code omitted for brevity]
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Login error for user {Email}",
model.Email);
                ModelState.AddModelError("", "Giriş sırasında bir hata
oluştu.");
                return View(model);
            }
        }

        // Additional methods: Logout, Register, etc.
    }
}

------------- Controllers/CartController.cs -------------
[Shopping cart controller with session-based cart management, checkout process,
and payment integration - full implementation included in original file]

------------- Controllers/AdminController.cs -------------
[Comprehensive admin panel controller with CRUD operations for products,
categories, users, and orders - full implementation included in original file]

------------- Controllers/OrderController.cs -------------
[Order management controller for user order history and details - full
implementation included in original file]

========================================
4. VIEW MODELS
========================================

------------- ViewModels/AccountViewModel.cs -------------
using System.ComponentModel.DataAnnotations;
using ETicaret.Models;

namespace ETicaret.ViewModels
{
    public class LoginViewModel
    {
        [Required(ErrorMessage = "E-posta adresi gereklidir")]
        [EmailAddress(ErrorMessage = "Geçersiz e-posta formatı")]
        public string Email { get; set; } = string.Empty;

        [Required(ErrorMessage = "Şifre gereklidir")]
        [DataType(DataType.Password)]
        public string Password { get; set; } = string.Empty;

        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
```

```
        [Required(ErrorMessage = "Ad gereklidir")]
        [StringLength(100)]
        public string FirstName { get; set; } = string.Empty;

        [Required(ErrorMessage = "Soyad gereklidir")]
        [StringLength(100)]
        public string LastName { get; set; } = string.Empty;

        [Required(ErrorMessage = "E-posta adresi gereklidir")]
        [EmailAddress(ErrorMessage = "Geçersiz e-posta formatı")]
        public string Email { get; set; } = string.Empty;

        [Required(ErrorMessage = "Şifre gereklidir")]
        [StringLength(100, MinimumLength = 6, ErrorMessage = "Şifre en az 6
karakter olmalıdır")]
        [DataType(DataType.Password)]
        public string Password { get; set; } = string.Empty;

        [Required(ErrorMessage = "Şifre onayı gereklidir")]
        [DataType(DataType.Password)]
        [Compare("Password", ErrorMessage = "Şifre ve şifre onayı eşleşmiyor")]
        public string ConfirmPassword { get; set; } = string.Empty;

        [Phone]
        public string? Phone { get; set; }
    }

    public class AdminDashboardViewModel
    {
        public int TotalProducts { get; set; }
        public int TotalCategories { get; set; }
        public int TotalUsers { get; set; }
        public int TotalOrders { get; set; }

        public List<Product> RecentProducts { get; set; } = new();
        public List<Order> RecentOrders { get; set; } = new();
        public List<Category> Categories { get; set; } = new();
        public List<User> RecentUsers { get; set; } = new();
    }
}

------------- ViewModels/CartViewModel.cs -------------
using ETicaret.Models;

namespace ETicaret.ViewModels
{
    public class CartViewModel
    {
        public List<CartItemViewModel> Items { get; set; } = new();
        public decimal TotalAmount { get; set; }
        public int TotalItems { get; set; }
    }

    public class CartItemViewModel
    {
        public int Id { get; set; }
        public Product Product { get; set; } = null!;
        public int Quantity { get; set; }
        public decimal TotalPrice => Product.Price * Quantity;
    }
}

------------- ViewModels/CheckoutViewModel.cs -------------
using System.ComponentModel.DataAnnotations;
```

```
using ETicaret.Models;

namespace ETicaret.ViewModels
{
    public class CheckoutViewModel : IValidatableObject
    {
        public List<CartItemViewModel> Items { get; set; } = new();
        public decimal TotalAmount { get; set; }
        public int TotalItems { get; set; }
        public ShippingInfo Shipping { get; set; } = new();
        public PaymentInfo Payment { get; set; } = new();

        [StringLength(1000)]
        [Display(Name = "Sipariş Notları")]
        public string? OrderNotes { get; set; }

        public IEnumerable<ValidationResult> Validate(ValidationContext
validationContext)
        {
            yield break; // Return empty validation results
        }
    }

    public class ShippingInfo
    {
        [Required]
        [StringLength(500)]
        [Display(Name = "Kargo Adresi")]
        public string Address { get; set; } = string.Empty;

        [Required]
        [StringLength(100)]
        [Display(Name = "Şehir")]
        public string City { get; set; } = string.Empty;

        [StringLength(20)]
        [Display(Name = "Posta Kodu")]
        public string? PostalCode { get; set; }

        public bool IsValid => !string.IsNullOrWhiteSpace(Address) &&
                               !string.IsNullOrWhiteSpace(City);
    }

    public class PaymentInfo
    {
        private const string CreditCardPattern = @"^[0-9\s]{13,19}$";
        private const string ExpirationDatePattern = @"^(0[1-9]|1[0-2])\/([0-9]
{2})$";
        private const string CVVPattern = @"^\d{3,4}$";

        [Required]
        [RegularExpression(CreditCardPattern, ErrorMessage = "Geçerli bir kredi
kartı numarası giriniz")]
        [Display(Name = "Kredi Kartı Numarası")]
        public string CardNumber { get; set; } = string.Empty;

        [Required]
        [RegularExpression(ExpirationDatePattern, ErrorMessage = "MM/YY
formatında tarih giriniz")]
        [Display(Name = "Son Kullanma Tarihi")]
        public string ExpirationDate { get; set; } = string.Empty;

        [Required]
        [RegularExpression(CVVPattern, ErrorMessage = "CVV 3 veya 4 haneli
```

```
olmalıdır")]
        [Display(Name = "CVV")]
        public string CVV { get; set; } = string.Empty;

        [Required]
        [StringLength(100)]
        [Display(Name = "Kart Sahibi Adı")]
        public string CardholderName { get; set; } = string.Empty;

        public bool IsValid => !string.IsNullOrWhiteSpace(CardNumber) &&
                               !string.IsNullOrWhiteSpace(ExpirationDate) &&
                               !string.IsNullOrWhiteSpace(CVV) &&
                               !string.IsNullOrWhiteSpace(CardholderName);
    }

    public class OrderConfirmationViewModel
    {
        public Order Order { get; set; } = null!;
        public List<OrderItem> OrderItems { get; set; } = new();
    }
}

------------- ViewModels/HomeViewModel.cs -------------
using ETicaret.Models;

namespace ETicaret.ViewModels
{
    public class HomeViewModel
    {
        public List<Product> FeaturedProducts { get; set; } = new();
        public List<Product> NewProducts { get; set; } = new();
        public List<Category> Categories { get; set; } = new();
    }
}

------------- ViewModels/ProductListViewModel.cs -------------
using ETicaret.Models;

namespace ETicaret.ViewModels
{
    public class ProductListViewModel
    {
        public List<Product> Products { get; set; } = new();
        public List<Category> Categories { get; set; } = new();
        public int? SelectedCategoryId { get; set; }
        public string? SearchTerm { get; set; }
        public int CurrentPage { get; set; } = 1;
        public int TotalPages { get; set; }
        public int PageSize { get; set; } = 12;
    }
}

========================================
5. SERVICES & COMPONENTS
========================================

------------- Services/IPaymentService.cs -------------
namespace ETicaret.Services
{
    public interface IPaymentService
    {
        Task<PaymentResult> ProcessPaymentAsync(PaymentRequest request);
    }
```

```
    public class PaymentRequest
    {
        public string CardNumber { get; set; } = string.Empty;
        public string ExpirationDate { get; set; } = string.Empty;
        public string CVV { get; set; } = string.Empty;
        public string CardholderName { get; set; } = string.Empty;
        public decimal Amount { get; set; }
        public string Currency { get; set; } = "USD";
        public string OrderNumber { get; set; } = string.Empty;
    }

    public class PaymentResult
    {
        public bool IsSuccess { get; set; }
        public string TransactionId { get; set; } = string.Empty;
        public string ErrorMessage { get; set; } = string.Empty;
        public string BankResponse { get; set; } = string.Empty;
        public DateTime ProcessedAt { get; set; }
    }
}

------------- Services/PaymentService.cs -------------
[Payment service implementation with simulation of bank API calls, card
validation, and various payment scenarios]

------------- Extensions/EnumExtensions.cs -------------
using System.ComponentModel;
using System.Reflection;
using Microsoft.Extensions.Localization;
using ETicaret.Resources;

namespace ETicaret.Extensions
{
    public static class EnumExtensions
    {
        public static string GetLocalizedDescription(this Enum enumValue,
IStringLocalizer<SharedResource> localizer)
        {
            FieldInfo? field =
enumValue.GetType().GetField(enumValue.ToString());

            if (field?.GetCustomAttribute<DescriptionAttribute>() is
DescriptionAttribute attribute)
            {
                return localizer[attribute.Description];
            }

            return enumValue.ToString();
        }

        public static string GetLocalizedDescription(this Enum enumValue)
        {
            FieldInfo? field =
enumValue.GetType().GetField(enumValue.ToString());

            if (field?.GetCustomAttribute<DescriptionAttribute>() is
DescriptionAttribute attribute)
            {
                var resourceManager = SharedResource.ResourceManager;
                var culture = System.Globalization.CultureInfo.CurrentUICulture;

                string? localizedValue =
resourceManager?.GetString(attribute.Description, culture);
```

```
                return localizedValue ?? enumValue.ToString();
            }

            return enumValue.ToString();
        }
    }
}

------------- ModelBinders/DecimalModelBinder.cs -------------
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System.Globalization;

namespace ETicaret.ModelBinders
{
    public class DecimalModelBinder : IModelBinder
    {
        public Task BindModelAsync(ModelBindingContext bindingContext)
        {
            if (bindingContext == null)
                throw new ArgumentNullException(nameof(bindingContext));

            var valueProviderResult =
bindingContext.ValueProvider.GetValue(bindingContext.ModelName);

            if (valueProviderResult == ValueProviderResult.None)
                return Task.CompletedTask;

            bindingContext.ModelState.SetModelValue(bindingContext.ModelName,
valueProviderResult);

            var value = valueProviderResult.FirstValue;

            if (string.IsNullOrEmpty(value))
            {
                if (bindingContext.ModelType == typeof(decimal?))
                {
                    bindingContext.Result = ModelBindingResult.Success(null);
                }
                else
                {
                    bindingContext.Result = ModelBindingResult.Success(0m);
                }
                return Task.CompletedTask;
            }

            // Try parsing with different cultures
            decimal result;

            // First try with invariant culture (dot as decimal separator)
            if (decimal.TryParse(value, NumberStyles.Number,
CultureInfo.InvariantCulture, out result))
            {
                bindingContext.Result = ModelBindingResult.Success(result);
                return Task.CompletedTask;
            }

            // Then try with Turkish culture (comma as decimal separator)
            if (decimal.TryParse(value, NumberStyles.Number, new
CultureInfo("tr-TR"), out result))
            {
                bindingContext.Result = ModelBindingResult.Success(result);
                return Task.CompletedTask;
            }
```

```
            // Finally try with current culture
            if (decimal.TryParse(value, NumberStyles.Number,
CultureInfo.CurrentCulture, out result))
            {
                bindingContext.Result = ModelBindingResult.Success(result);
                return Task.CompletedTask;
            }

            bindingContext.ModelState.TryAddModelError(bindingContext.ModelName,
"Invalid decimal value");
            bindingContext.Result = ModelBindingResult.Failed();

            return Task.CompletedTask;
        }
    }
}

------------- ModelBinders/DecimalModelBinderProvider.cs -------------
using Microsoft.AspNetCore.Mvc.ModelBinding;

namespace ETicaret.ModelBinders
{
    public class DecimalModelBinderProvider : IModelBinderProvider
    {
        public IModelBinder GetBinder(ModelBinderProviderContext context)
        {
            if (context == null)
                throw new ArgumentNullException(nameof(context));

            if (context.Metadata.ModelType == typeof(decimal) ||
                context.Metadata.ModelType == typeof(decimal?))
            {
                return new DecimalModelBinder();
            }

            return null;
        }
    }
}

------------- Filters/AdminOnlyAttribute.cs -------------
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

namespace ETicaret.Filters
{
    public class AdminOnlyAttribute : ActionFilterAttribute
    {
        public override void OnActionExecuting(ActionExecutingContext context)
        {
            // Check if user is accessing admin panel
            var isAdminPath =
context.HttpContext.Request.Path.StartsWithSegments("/Admin");

            if (isAdminPath)
            {
                // In production, implement proper authentication check here
            }

            base.OnActionExecuting(context);
        }
    }
}
```

```
------------- Resources/SharedResource.cs -------------
using System.Resources;

namespace ETicaret.Resources
{
    public class SharedResource
    {
        private static readonly Lazy<ResourceManager> _resourceManager = new
Lazy<ResourceManager>(
            () => new ResourceManager("ETicaret.Resources.SharedResource",
typeof(SharedResource).Assembly)
        );

        public static ResourceManager ResourceManager => _resourceManager.Value;
    }
}

========================================
6. VIEWS & UI (Key Views)
========================================

------------- Views/Shared/_Layout.cshtml -------------
<!DOCTYPE html>
<html lang="tr">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - ETicaret</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-
primary shadow mb-3">
            <div class="container">
                <a class="navbar-brand fw-bold" asp-area="" asp-
controller="Home" asp-action="Index">
                    <i class="fas fa-shopping-bag me-2"></i>ETicaret
                </a>
                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target=".navbar-collapse">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-
content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-white" asp-controller="Home"
asp-action="Index">
                                <i class="fas fa-home me-1"></i>Ana Sayfa
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white" asp-
controller="Products" asp-action="Index">
                                <i class="fas fa-box me-1"></i>Ürünler
                            </a>
                        </li>
                    </ul>
                    <ul class="navbar-nav">
                        <li class="nav-item">
```

```
                            <a class="nav-link text-white position-relative"
asp-controller="Cart" asp-action="Index">
                                <i class="fas fa-shopping-cart me-1"></i>Sepet
                                <span class="position-absolute top-0 start-100
translate-middle badge rounded-pill bg-danger" id="cart-count">
                                    0
                                </span>
                            </a>
                        </li>
                        @if (User.Identity?.IsAuthenticated == true)
                        {
                            @if (User.HasClaim("IsAdmin", "true"))
                            {
                                <li class="nav-item">
                                    <a class="nav-link text-white"
href="@Url.Action("Index", "Admin")">
                                        <i class="fas fa-crown
me-1"></i>Yönetici Paneli
                                    </a>
                                </li>
                            }
                            <!-- User dropdown menu -->
                        }
                        else
                        {
                            <li class="nav-item">
                                <a class="nav-link text-white"
href="@Url.Action("Login", "Account")">
                                    <i class="fas fa-sign-in-alt me-1"></i>Giriş
Yap
                                </a>
                            </li>
                        }
                    </ul>
                </div>
            </div>
        </nav>
    </header>

    <div class="container">
        <!-- Alert messages for success/error feedback -->
        @if (TempData["Success"] != null)
        {
            <div class="alert alert-success alert-dismissible fade show"
role="alert">
                <i class="fas fa-check-circle me-2"></i>@TempData["Success"]
                <button type="button" class="btn-close" data-bs-
dismiss="alert"></button>
            </div>
        }

        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted bg-light mt-5">
        <div class="container py-4">
            <div class="row">
                <div class="col-md-4">
                    <h5><i class="fas fa-shopping-bag me-2"></i>ETicaret</h5>
                    <p>Kaliteli ürünler için tek durak alışveriş merkeziniz.</p>
                </div>
                <div class="col-md-4">
```

```
                    <h6>Hızlı Bağlantılar</h6>
                    <ul class="list-unstyled">
                        <li><a href="#" class="text-muted">Hakkımızda</a></li>
                        <li><a href="#" class="text-muted">İletişim</a></li>
                        <li><a href="#" class="text-muted">Gizlilik
Politikası</a></li>
                    </ul>
                </div>
                <div class="col-md-4">
                    <h6>Müşteri Hizmetleri</h6>
                    <ul class="list-unstyled">
                        <li><a href="#" class="text-muted">Yardım
Merkezi</a></li>
                        <li><a href="#" class="text-muted">Kargo
Bilgileri</a></li>
                        <li><a href="#" class="text-muted">İadeler</a></li>
                    </ul>
                </div>
            </div>
            <hr>
            <div class="text-center">
                &copy; 2025 - ETicaret - Tüm hakları saklıdır
            </div>
        </div>
    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>

    <script>
        // Update cart count on page load
        $(document).ready(function() {
            updateCartCount();
        });

        function updateCartCount() {
            $.get('/Cart/GetCartCount', function(data) {
                $('#cart-count').text(data.count);
                if (data.count > 0) {
                    $('#cart-count').show();
                } else {
                    $('#cart-count').hide();
                }
            });
        }
    </script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>

------------- Views/Home/Index.cshtml -------------
@model HomeViewModel
@{
    ViewData["Title"] = "ETicaret'e Hoşgeldiniz";
}

<!-- Hero Section -->
<div class="jumbotron bg-primary text-white rounded mb-5 p-5">
    <div class="container">
        <h1 class="display-4 fw-bold">
            <i class="fas fa-shopping-bag me-3"></i>ETicaret'e Hoşgeldiniz
        </h1>
```

```
        <p class="lead">Uygun fiyatlarla harika ürünleri keşfedin. Hemen
alışveriş yapın ve hızlı, güvenilir teslimatın tadini çıkarın!</p>
        <a class="btn btn-light btn-lg" asp-controller="Products" asp-
action="Index" role="button">
            <i class="fas fa-search me-2"></i>Hemen Alışveriş Yap
        </a>
    </div>
</div>

<!-- Categories Section -->
@if (Model.Categories.Any())
{
    <section class="mb-5">
        <h2 class="text-center mb-4">
            <i class="fas fa-th-large me-2"></i>Kategoriye Göre Alışveriş
        </h2>
        <div class="row">
            @foreach (var category in Model.Categories)
            {
                <div class="col-md-4 mb-3">
                    <div class="card h-100 shadow-sm border-0 category-card">
                        @if (!string.IsNullOrEmpty(category.ImageUrl))
                        {
                            <img src="@category.ImageUrl" class="card-img-top"
alt="@category.Name" style="height: 200px; object-fit: cover;">
                        }
                        else
                        {
                            <div class="card-img-top bg-light d-flex align-
items-center justify-content-center" style="height: 200px;">
                                @switch (category.Name.ToLower())
                                {
                                    case "elektronik":
                                        <i class="fas fa-laptop fa-3x text-
primary"></i>
                                        break;
                                    case "giyim":
                                        <i class="fas fa-tshirt fa-3x text-
success"></i>
                                        break;
                                    case "kitap":
                                        <i class="fas fa-book fa-3x text-
info"></i>
                                        break;
                                    case "ev & bahçe":
                                        <i class="fas fa-home fa-3x text-
warning"></i>
                                        break;
                                    case "spor":
                                        <i class="fas fa-dumbbell fa-3x text-
danger"></i>
                                        break;
                                    default:
                                        <i class="fas fa-box fa-3x text-
secondary"></i>
                                        break;
                                }
                            </div>
                        }

                        <div class="card-body text-center">
                            <h5 class="card-title">@category.Name</h5>
                            <p class="card-text text-
muted">@category.Description</p>
```

```
                            <a asp-controller="Products" asp-action="Index" asp-
route-categoryId="@category.Id"
                               class="btn btn-outline-primary">
                                @category.Name İncele
                            </a>
                        </div>
                    </div>
                </div>
            }
        </div>
    </section>
}

<!-- Featured Products Section -->
@if (Model.FeaturedProducts.Any())
{
    <section class="mb-5">
        <h2 class="text-center mb-4">
            <i class="fas fa-star me-2"></i>Öne Çıkan Ürünler
        </h2>
        <div class="row">
            @foreach (var product in Model.FeaturedProducts)
            {
                <div class="col-md-3 col-sm-6 mb-4">
                    <div class="card h-100 shadow-sm product-card">
                        @if (!string.IsNullOrEmpty(product.ImageUrl))
                        {
                            <img src="@product.ImageUrl" class="card-img-top"
alt="@product.Name" style="height: 200px; object-fit: cover;">
                        }
                        else
                        {
                            <div class="card-img-top bg-light d-flex align-
items-center justify-content-center" style="height: 200px;">
                                <i class="fas fa-image fa-3x text-muted"></i>
                            </div>
                        }

                        @if (product.DiscountPrice.HasValue)
                        {
                            <div class="position-absolute top-0 start-0">
                                <span class="badge bg-danger
m-2">Kampanya!</span>
                            </div>
                        }

                        <div class="card-body d-flex flex-column">
                            <h6 class="card-title">@product.Name</h6>
                            <p class="card-text text-muted small flex-grow-
1">@product.Description</p>
                            <div class="price-section mb-2">
                                @if (product.DiscountPrice.HasValue)
                                {
                                    <span class="h6 text-primary fw-
bold">@product.DiscountPrice.Value.ToString("C")</span>
                                    <span class="text-muted text-decoration-
line-through ms-2">@product.Price.ToString("C")</span>
                                }
                                else
                                {
                                    <span class="h6 text-primary fw-
bold">@product.Price.ToString("C")</span>
                                }
                            </div>
```

```
                            <div class="d-flex justify-content-between align-
items-center">
                                <a asp-controller="Products" asp-
action="Details" asp-route-id="@product.Id"
                                   class="btn btn-outline-primary btn-sm">
                                    <i class="fas fa-eye me-1"></i>Gör
                                </a>
                                <form asp-controller="Cart" asp-
action="AddToCart" method="post" class="d-inline">
                                    <input type="hidden" name="productId"
value="@product.Id" />
                                    <button type="submit" class="btn btn-primary
btn-sm">
                                        <i class="fas fa-cart-plus
me-1"></i>Sepete Ekle
                                    </button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            }
        </div>
    </section>
}

<style>
    .product-card:hover {
        transform: translateY(-2px);
        transition: transform 0.2s ease-in-out;
    }

    .category-card:hover {
        transform: translateY(-2px);
        transition: transform 0.2s ease-in-out;
    }

    .jumbotron {
        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    }
</style>

========================================
7. CSS & JAVASCRIPT
========================================

------------- wwwroot/css/site.css -------------
html {
  font-size: 14px;
}

@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

.btn:focus, .btn:active:focus, .btn-link.nav-link:focus, .form-
control:focus, .form-check-input:focus {
  box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;
}

html {
  position: relative;
```

```
  min-height: 100%;
}

body {
  margin-bottom: 60px;
}

.product-card {
    transition: transform 0.2s ease-in-out;
}

.product-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 25px rgba(0,0,0,0.1);
}

.category-card {
    transition: transform 0.2s ease-in-out;
}

.category-card:hover {
    transform: translateY(-3px);
    box-shadow: 0 6px 20px rgba(0,0,0,0.1);
}

.footer {
    background-color: #f8f9fa;
    padding: 2rem 0;
    margin-top: auto;
}

------------- wwwroot/js/site.js -------------
// Please see documentation at https://docs.microsoft.com/aspnet/core/client-
side/bundling-and-minification
// for details on configuring this project to bundle and minify static web
assets.

// Write your JavaScript code.

// Cart functionality
function addToCart(productId, quantity = 1) {
    fetch('/Cart/AddToCart', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `productId=${productId}&quantity=${quantity}`
    })
    .then(response => {
        if (response.ok) {
            updateCartCount();
            // Show success message
            showAlert('Ürün sepete eklendi!', 'success');
        }
    });
}

function updateCartCount() {
    fetch('/Cart/GetCartCount')
    .then(response => response.json())
    .then(data => {
        const cartBadge = document.getElementById('cart-count');
        if (cartBadge) {
            cartBadge.textContent = data.count;
```

```
            cartBadge.style.display = data.count > 0 ? 'block' : 'none';
        }
    });
}

function showAlert(message, type = 'info') {
    const alertDiv = document.createElement('div');
    alertDiv.className = `alert alert-${type} alert-dismissible fade show`;
    alertDiv.innerHTML = `
        ${message}
        <button type="button" class="btn-close"
data-bs-dismiss="alert"></button>
    `;

    const container = document.querySelector('.container');
    container.insertBefore(alertDiv, container.firstChild);

    // Auto dismiss after 3 seconds
    setTimeout(() => {
        alertDiv.remove();
    }, 3000);
}

// Form validation helpers
function validateForm(formSelector) {
    const form = document.querySelector(formSelector);
    if (!form) return true;

    const inputs = form.querySelectorAll('input[required], select[required],
textarea[required]');
    let isValid = true;

    inputs.forEach(input => {
        if (!input.value.trim()) {
            input.classList.add('is-invalid');
            isValid = false;
        } else {
            input.classList.remove('is-invalid');
        }
    });

    return isValid;
}

// Initialize tooltips and other Bootstrap components
document.addEventListener('DOMContentLoaded', function() {
    // Initialize tooltips
    var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-
toggle="tooltip"]'));
    var tooltipList = tooltipTriggerList.map(function (tooltipTriggerEl) {
        return new bootstrap.Tooltip(tooltipTriggerEl);
    });

    // Initialize popovers
    var popoverTriggerList = [].slice.call(document.querySelectorAll('[data-bs-
toggle="popover"]'));
    var popoverList = popoverTriggerList.map(function (popoverTriggerEl) {
        return new bootstrap.Popover(popoverTriggerEl);
    });

    // Update cart count on page load
    updateCartCount();
});
```

==
8. PROJECT SUMMARY
==

This ETicaret (E-Commerce) project is a full-featured ASP.NET Core MVC application with the following characteristics:

FRAMEWORK & TECHNOLOGY:
- .NET 9.0 (ASP.NET Core MVC)
- Entity Framework Core 9.0.7
- MySQL Database (Pomelo.EntityFrameworkCore.MySql)
- Bootstrap 5 for responsive UI
- Font Awesome for icons
- jQuery for client-side interactions

KEY FEATURES:
- User Authentication & Authorization (Cookie-based)
- Product Catalog with Categories
- Shopping Cart (Session-based)
- Order Management
- Payment Processing (Simulated)
- Admin Panel for CRUD operations
- Responsive Design
- Turkish Language Support
- Image Upload for Products/Categories

DATABASE MODELS:
- User (Authentication & Profile)
- Product (Catalog items)
- Category (Product categorization)
- Order & OrderItem (Purchase history)
- CartItem (Shopping cart)

CONTROLLERS:
- HomeController (Landing page)
- ProductsController (Product browsing)
- CartController (Shopping cart & checkout)
- AccountController (Login/Register)
- AdminController (Administrative functions)
- OrderController (Order history)

SECURITY FEATURES:
- Password hashing with BCrypt
- Anti-forgery tokens
- Admin-only areas
- Input validation
- SQL injection protection via EF Core

BUSINESS LOGIC:
- Stock management
- Price calculations with discounts
- Order processing workflow
- Payment validation
- User session management

The application follows MVC architectural patterns, implements proper separation of concerns, uses dependency injection, and includes comprehensive error handling. It's designed for a Turkish e-commerce environment with localization support and responsive design principles.

==
END OF SOURCE CODE COMPILATION
==
Generated: 2025-01-16 03:04:00 UTC

Total Lines: ~8000+ across all files