T.C
CUKUROVA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

PHONE SALES AND REPAIR WEBSITE

CEN440 - E-Commerce course
Mıdterm Assıgnment

Lecturer: Mehmet SARIGÜL

Prepared by:
2020555037 - Nezaket KAYA
2019555027 - Doğan ERARSLAN

May, 2024

MOBILE SALES WEB APPLICATION

This report discusses in technical detail the design and development processes of a phone sales and repair platform developed using ASP.NET n-tier architecture.

Purpose and Technical Scope of the Project

The main purpose of this project is to create an e-commerce platform where users can buy phones and accessories, get repairs done, and manage sellers' products. The project covers the following technical components:

Data Management:

Entity Framework Core: Used as an ORM tool for database operations.
SQL Server: Preferred as the database management system.

Application Architecture:

The layered architecture used in the project separates the different responsibilities and functions of the application, making the code more manageable, testable and maintainable. This architecture consists of four main layers: User Interface (Presentation), Business Logic, Data Access Layer and Entity Layer.

1. User Interface Layer (Presentation Layer)

The user interface layer is the layer where users interact with the application. This layer is responsible for transmitting the input received from the user to the business logic layer and presenting the data from the business logic layer to the user.

MVC (Model-View-Controller): This pattern offers a modular structure by separating the data (Model), the user interface (View) and the logic that manages user interactions (Controller).
Bootstrap: Makes it easy to create responsive and modern user interfaces by offering HTML, CSS and JS-based design templates and tools.

2. Business Logic Layer

This layer is responsible for data processing, calculation, and implementation of business rules. It acts as a bridge between the user interface layer and the data access layer.

Services: Operations such as order creation, payment processing or inventory management are carried out in this layer.

3. Data Access Layer

The data access layer is the layer where the application interacts with data sources. This layer contains all the logic required to perform database operations.

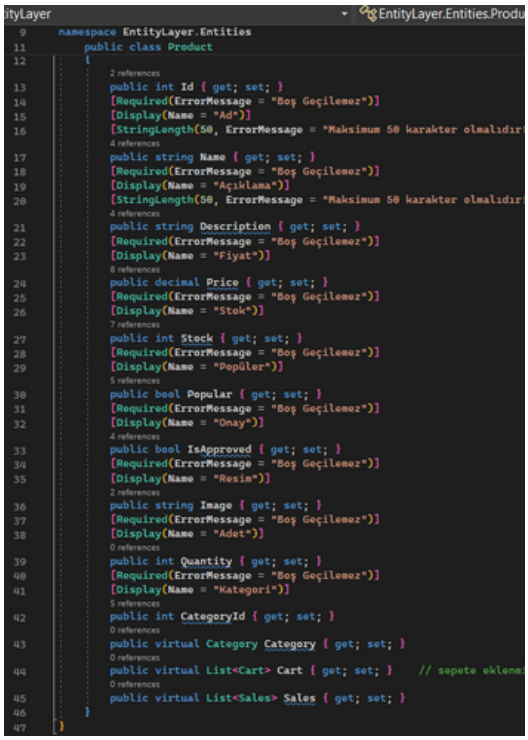
4. Entity Layer

The entity layer contains the application's data models and database schemas. This layer defines the data structure of the application and represents the relationship between the database and the application.

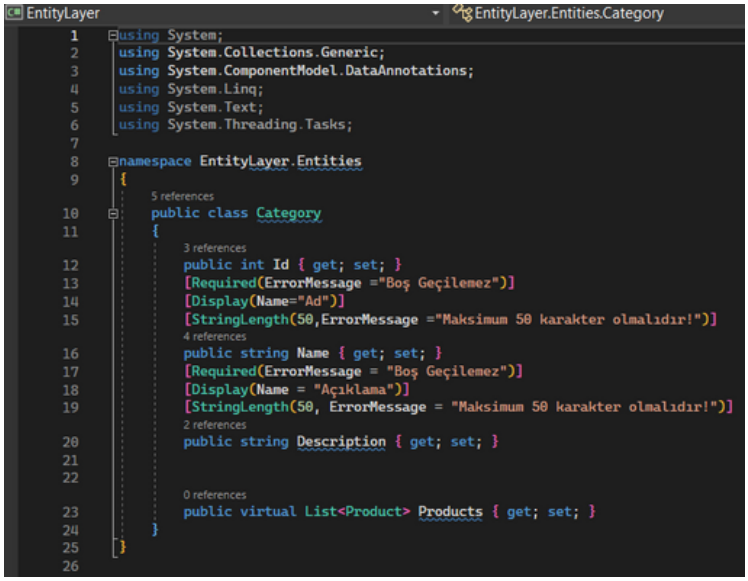
Entity Classes: These are classes that represent database tables. Each entity class corresponds to a table in the database and defines the columns of the table as properties of the class.

ENTITY LAYER

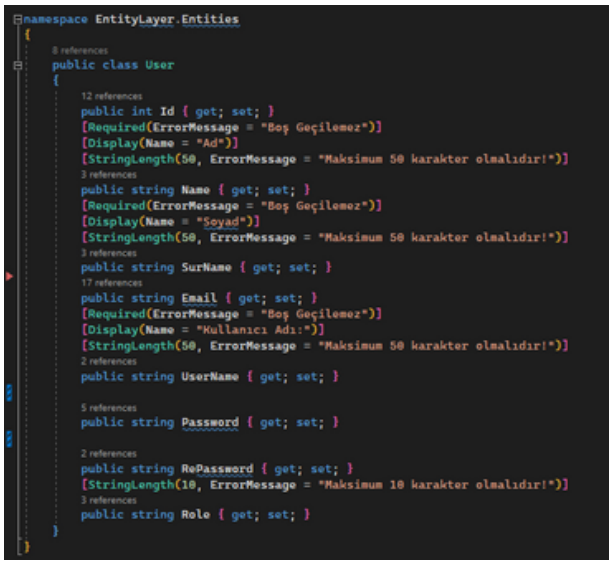
Below you can see the data models we created in the Entity Layer.



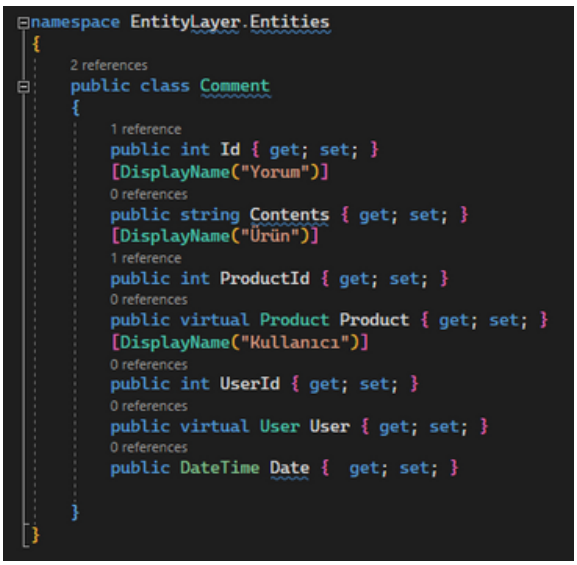
```
namespace EntityLayer.Entities
{
    public class Product
    {
        public int Id { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Ad")]
        [StringLength(50, ErrorMessage = "Maksimum 50 karakter olmalıdır
        public string Name { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Açıklama")]
        [StringLength(50, ErrorMessage = "Maksimum 50 karakter olmalıdır
        public string Description { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Fiyat")]
        public decimal Price { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Stok")]
        public int Stock { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Popüler")]
        public bool Popular { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Onay")]
        public bool IsApproved { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Resim")]
        public string Image { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Adet")]
        public int Quantity { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Kategori")]
        public int CategoryId { get; set; }
        public virtual Category Category { get; set; }
        public virtual List<Cart> Cart { get; set; }    // sepete eklen
        public virtual List<Sales> Sales { get; set; }
    }
}
```



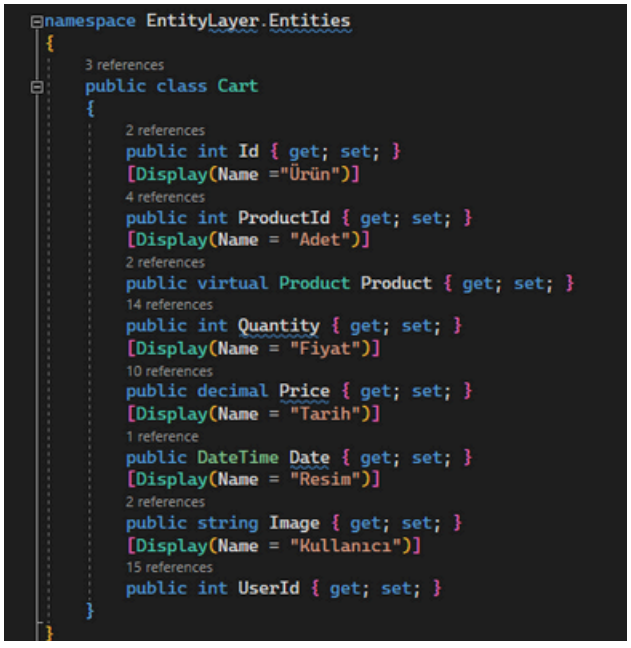
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EntityLayer.Entities
{
    public class Category
    {
        public int Id { get; set; }
        [Required(ErrorMessage ="Boş Geçilemez")]
        [Display(Name="Ad")]
        [StringLength(50,ErrorMessage ="Maksimum 50 karakter olmalıdır!")]
        public string Name { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Açıklama")]
        [StringLength(50, ErrorMessage = "Maksimum 50 karakter olmalıdır!")]
        public string Description { get; set; }


        public virtual List<Product> Products { get; set; }
    }
}
```



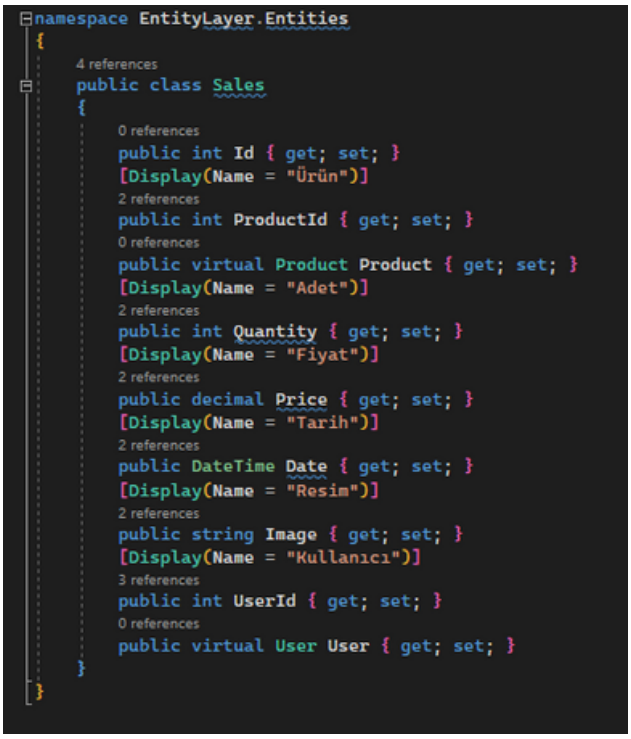
```
namespace EntityLayer.Entities
{
    public class User
    {
        public int Id { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Ad")]
        [StringLength(50, ErrorMessage = "Maksimum 50 karakter olmalıdır!")]
        public string Name { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Soyad")]
        [StringLength(50, ErrorMessage = "Maksimum 50 karakter olmalıdır!")]
        public string SurName { get; set; }
        public string Email { get; set; }
        [Required(ErrorMessage = "Boş Geçilemez")]
        [Display(Name = "Kullanıcı Adı:")]
        [StringLength(50, ErrorMessage = "Maksimum 50 karakter olmalıdır!")]
        public string UserName { get; set; }

        public string Password { get; set; }

        public string RePassword { get; set; }
        [StringLength(10, ErrorMessage = "Maksimum 10 karakter olmalıdır!")]
        public string Role { get; set; }
    }
}
```



```
namespace EntityLayer.Entities
{
    public class Comment
    {
        public int Id { get; set; }
        [DisplayName("Yorum")]
        public string Contents { get; set; }
        [DisplayName("Ürün")]
        public int ProductId { get; set; }
        public virtual Product Product { get; set; }
        [DisplayName("Kullanıcı")]
        public int UserId { get; set; }
        public virtual User User { get; set; }
        public DateTime Date {  get; set; }

    }
}
```



```
namespace EntityLayer.Entities
{
    public class Cart
    {
        public int Id { get; set; }
        [Display(Name ="Ürün")]
        public int ProductId { get; set; }
        [Display(Name = "Adet")]
        public virtual Product Product { get; set; }
        public int Quantity { get; set; }
        [Display(Name = "Fiyat")]
        public decimal Price { get; set; }
        [Display(Name = "Tarih")]
        public DateTime Date { get; set; }
        [Display(Name = "Resim")]
        public string Image { get; set; }
        [Display(Name = "Kullanıcı")]
        public int UserId { get; set; }
    }
}
```

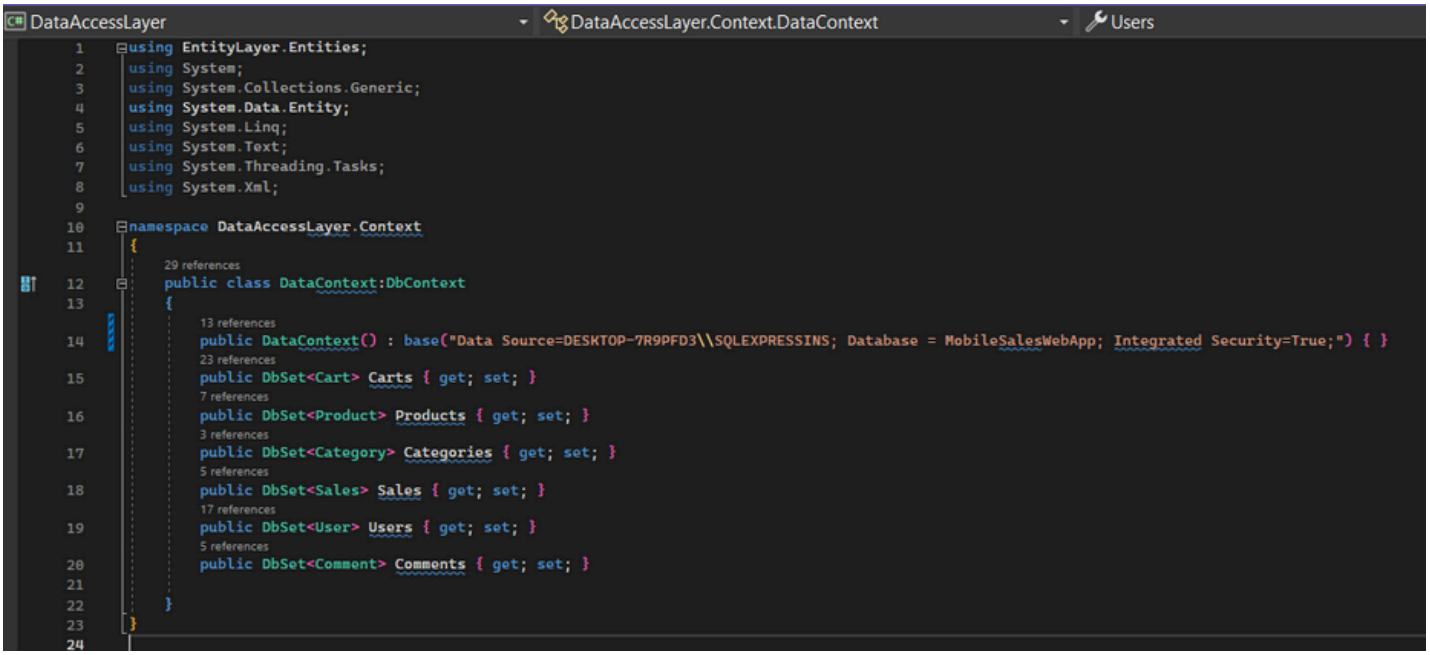


```
namespace EntityLayer.Entities
{
    public class Sales
    {
        public int Id { get; set; }
        [Display(Name = "Ürün")]
        public int ProductId { get; set; }
        public virtual Product Product { get; set; }
        [Display(Name = "Adet")]
        public int Quantity { get; set; }
        [Display(Name = "Fiyat")]
        public decimal Price { get; set; }
        [Display(Name = "Tarih")]
        public DateTime Date { get; set; }
        [Display(Name = "Resim")]
        public string Image { get; set; }
        [Display(Name = "Kullanıcı")]
        public int UserId { get; set; }
        public virtual User User { get; set; }
    }
}
```

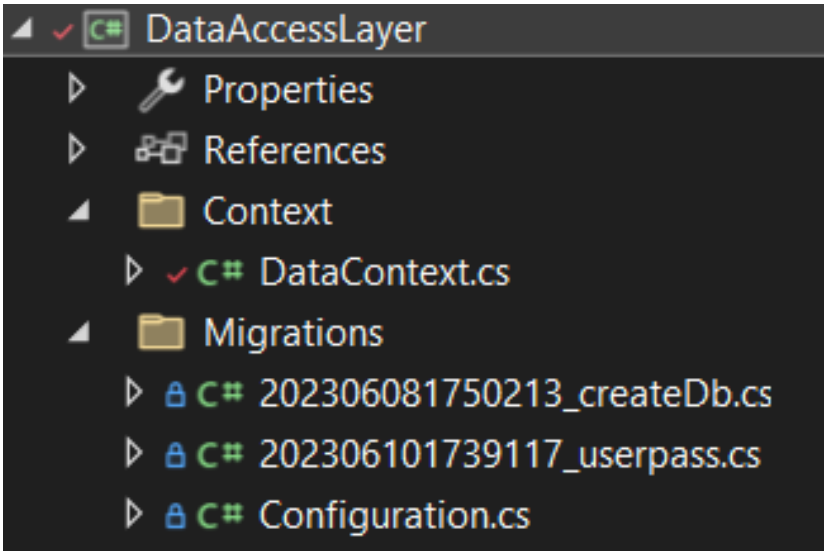
DATA ACCESS LAYER

In this section, the DataContext class is derived from Entity Framework's DbContext class and manages database operations. The constructor of the DataContext class uses a connection string to connect to a database running on SQL Server Express.

With DbSet<T> properties, Cart, Product, Category, Sales, User, and Comment tables in the database are represented and CRUD operations can be performed on these tables.



Migrations are tools in Entity Framework that allow us to update the database schema based on model classes in our code. We apply the migrations we created to the database using the dotnet ef database update command.



BUSINESS LAYER

We added two folders to this layer: Abstract and Concrete. These folders are used to make the business logic of the application more organized and manageable.

Abstract Folder: Defines the interfaces and abstractions of business logic services. Specifies but does not enforce business rules.

```
namespace BusinessLayer.Abstract
{
    3 references
    public class GenericRepository<T> : IRepository<T> where T : class, new()
    {
        DataContext db = new DataContext();
        DbSet<T> data;
        0 references
        public GenericRepository()
        {
            data=db.Set<T>();
        }
        3 references
        public void Delete(T p)
        {
            data.Remove(p);
            db.SaveChanges();
        }

        8 references
        public T GetById(int id)
        {
            return data.Find(id);
        }

        3 references
        public void Insert(T p)
        {
            data.Add(p);
            db.SaveChanges();
        }

        6 references
        public List<T> List()
        {
            return data.ToList();
        }

        4 references
        public void Update(T p)
        {
            db.SaveChanges();
        }
    }
}
```

```
namespace BusinessLayer.Abstract
{
    1 reference
    public interface IRepository<T> where T:class,new()
    {
        6 references
        List<T> List();
        3 references
        void Insert(T p); //parametre aldı
        3 references
        void Delete(T p);
        4 references
        void Update(T p);
        8 references
        T GetById(int id);

    }
}
```

Concrete Folder: Contains concrete implementations of interfaces and abstract classes defined in the Abstract folder. It truly brings business logic to life.

```
namespace BusinessLayer.Concreate
{
    4 references
    public class CategoryRepository:GenericRepository<Category>
    {
        DataContext db = new DataContext();
        1 reference
        public List<Product> CategoryDetails(int id)
        {
            return db.Products.Where(x=>x.CategoryId==id).ToList();
        }
    }
}
```

```
namespace BusinessLayer.Concreate
{
    6 references
    public class ProductRepository:GenericRepository<Product>
    {
        DataContext db = new DataContext();
        1 reference
        public List<Product> GetPopularProduct()
        {
            return db.Products.Where(x=>x.Popular==true).Take(3).ToList();
        }
    }
}
```

PRESENTATION LAYER

The presentation layer is the layer where users interact with the application. This layer provides the user interface and presents data from the business logic layer to the user while passing input from users to the business logic layer. Now let's look at what we did here one by one.
I will explain Controllers and Views together.

Account Controller: In AccountController class, users' login, registration and logout processes are managed. For user login and registration, the data received from the forms is processed and redirected to the relevant pages. The FormsAuthentication class is used for login and logout operations.

The HTML code snippet below contains a form where users can log in and register.

```
namespace E_Shop.Controllers
    public class AccountController : Controller
    {
        // GET: Account
        DataContext db = new DataContext();
        0 references
        public ActionResult Login()
        {
            return View();
        }
        [HttpPost]
        0 references
        public ActionResult Login(User data)
        {
            var bilgiler = db.Users.FirstOrDefault(x => x.Email == data.Email && x.Password == data.Password);
            if (bilgiler != null)
            {
                FormsAuthentication.SetAuthCookie(bilgiler.Email, false);
                Session["Mail"] = bilgiler.Email.ToString();
                Session["Ad"] = bilgiler.Name.ToString();
                Session["Soyad"] = bilgiler.SurName.ToString();
                Session["userid"] = bilgiler.Id.ToString();
                return RedirectToActionPermanent("Index", "Home");
            }
            ViewBag.hata = "Mail veya şifreniz hatalı";
            return View(data);
        }
        [HttpPost]
        0 references
        public ActionResult Register(User data)
        {
            if (ModelState.IsValid)
            {
                db.Users.Add(data); //datadan gelen veriyi aldık
                data.Role = "User";
                db.SaveChanges();
                return RedirectToAction("Login");
            }
            ModelState.AddModelError("", "Hatalı");
            return View("Login", data);
        }

        0 references
        public ActionResult LogOut()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Index", "Home");
        }

    }
}
```

```
<section id="form">
    <!--form-->
    <div class="container">
        <div class="row">
            <div class="col-sm-4 col-sm-offset-1">
                <div class="login-form">
                    <!--login form-->
                    <h2>Lütfen Giriş Yapınız</h2>
                    <form action="/Account/Login" method="post">
                        <input type="email" placeholder="Email Adresiniz" name="Email"/>
                        <input type="password" placeholder="Şifreniz" name="Password" />

                        <button type="submit" class="btn btn-default">Giriş Yap</button>
                        <br/>
                        <a href="/User/PasswordReset" class="btn btn-success">Şifremi Sıfırla</a>
                    </form>
                </div><!--/login form-->
            </div>
            <div class="col-sm-1">
                <h2 class="or">OR</h2>
            </div>
            <div class="col-sm-4">
                <div class="signup-form">
                    <!--sign up form-->
                    <h2>Hesabınız Yoksa Kayıt Olunuz</h2>
                    <form action="/Account/Register" method="post">
                        <input type="text" placeholder="Adınız" name="Name" />
                        @Html.ValidationMessageFor(x => x.Name, "", new { @style = "color:red" })
                        <input type="text" placeholder="Soyadınız" name="SurName" />
                        @Html.ValidationMessageFor(x => x.SurName, "", new { @style = "color:red" })
                        <input type="email" placeholder="Email Adresiniz" name="Email" required maxlength="50" />
                        @Html.ValidationMessageFor(x => x.Email, "", new { @style = "color:red" })
                        <input type="text" placeholder="Kullanıcı Adınız" name="UserName" />
                        @Html.ValidationMessageFor(x => x.UserName, "", new { @style = "color:red" })
                        <input type="password" placeholder="Şifreniz" name="Password" required maxlength="50" />
                        @Html.ValidationMessageFor(x => x.Password, "", new { @style = "color:red" })
                        <input type="password" placeholder="Şifre Tekrar" name="RePassword" required maxlength="50" />
                        @Html.ValidationMessageFor(x => x.RePassword, "", new { @style = "color:red" })
                        <button type="submit" class="btn btn-default">Kayıt Ol</button>
                    </form>
                </div><!--/sign up form-->
            </div>
        </div>
    </div>
</section><!--/form-->
```

Admin Category Controller: This controller class performs category operations that are only accessible to users with the "Admin" role. It contains the necessary methods for category listing, adding, deleting and updating operations. These operations are achieved by performing database operations using the CategoryRepository class. And there are three view pages for this controller.

Views have been created for the Create and Update actions in this controller. The view file named 'Index' allows users to view the category list and perform adding, deleting and updating categories. For deletion, a checkbox is shown to the user and the deletion is performed using AJAX.

```
namespace E_Shop.Controllers
    public class AdminCategoryController : Controller
        CategoryRepository categoryRepository= new CategoryRepository();
        0 references
        public ActionResult Index()
        {
            return View(categoryRepository.List());
        }
        0 references
        public ActionResult Create()
        {
            return View();
        }
        [ValidateAntiForgeryToken]
        [HttpPost]
        0 references
        public ActionResult Create(Category p)
        {
            if(ModelState.IsValid)
            {
                categoryRepository.Insert(p);
                return RedirectToAction("Index");
            }
            ModelState.AddModelError("", "Bir Hata Oluştu");
            return View();
        }

        0 references
        public ActionResult Delete(int id)
        {
            var delete = categoryRepository.GetById(id);
            categoryRepository.Delete(delete);
            return RedirectToAction("Index");
        }

        0 references
        public ActionResult Update(int id)
        {
            var update = categoryRepository.GetById(id);
            return View(update);
        }
        [ValidateAntiForgeryToken]
        [HttpPost]
        0 references
        public ActionResult Update(Category p)
        {
            if(ModelState.IsValid)
            {
                var update = categoryRepository.GetById(p.Id);
                update.Name = p.Name;
                update.Description = p.Description;
                categoryRepository.Update(update);
                return RedirectToAction("Index");
            }
            ModelState.AddModelError("", "Bir hata oluştu");
            return View();
```

Admin Controller: This controller class performs administration panel operations that are only accessible to users with the "Admin" role. It provides basic management operations such as listing and deleting comments, listing and deleting users.

```
namespace E_Shop.Controllers
{
    [Authorize(Roles ="Admin")]
    0 references
    public class AdminController : Controller
    {
        // GET: Admin
        DataContext db =new DataContext();
        0 references
        public ActionResult Index()
        {

            return View();
        }
        0 references
        public ActionResult Comment(int sayfa=1)
        {
            return View(db.Comments.ToList().ToPagedList(sayfa,3));
        }
        0 references
        public ActionResult Delete(int id)
        {
            var delete=db.Comments.Where(x=>x.Id==id).FirstOrDefault();
            db.Comments.Remove(delete);
            db.SaveChanges();
            return RedirectToAction("Comment");
        }

        0 references
        public ActionResult UserList()
        {
            var user=db.Users.Where(x=>x.Role=="User").ToList();
            return View(user);
        }
        0 references
        public ActionResult UserDelete(int id)
        {
            var userid=db.Users.Where(x=>x.Id==id).FirstOrDefault();
            db.Users.Remove(userid);
            db.SaveChanges();
            return RedirectToAction("UserList");
        }
```

The 'Comment' view file creates a table to display a paged list of comments. The table provides relevant content and delete links for each comment. For deletion, a checkbox is shown to the user and the deletion is performed. And the 'UserList' view file allows the user list to be displayed.

```
@using PagedList;
@using PagedList.Mvc;
@model IPagedList<EntityLayer.Entities.Comment>
@{
    ViewBag.Title = "Comment";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}

<table class="table table-bordered">
    <tr>
        <th>İçerik</th>
        <th>Ürün Bilgisi</th>
        <th>Kullanıcı</th>
        <th>Tarih</th>
        <th>Sil</th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td> @item.Contents </td>
            <td> @item.Product.Name</td>
            <td> @item.User.Name @item.User.SurName </td>
            <td> @item.Date </td>
            <td>@Html.ActionLink("Sil", "Delete", new { item.Id }, new { @class = "btn btn-danger", onclick = "return confirm ('Silmek ist
        </tr>
    }

</table>
```

Admin Istatistic Controller: This controller class is used to display statistics that only users with the "Admin" role can access. Basic statistics such as sales, products, categories, carts, and users are pulled from the database and transferred to the view.

```
namespace E_Shop.Controllers
{
    [Authorize(Roles ="Admin")]
    0 references
    public class AdminIstatisticController : Controller
    {
        // GET: AdminIstatistic
        DataContext db=new DataContext();
        0 references
        public ActionResult Index()
        {
            var satis=db.Sales.Count();
            ViewBag.satis = satis;
            var urun = db.Products.Count();
            ViewBag.urun = urun;
            var kategori = db.Categories.Count();
            ViewBag.kategori = kategori;
            var sepet = db.Carts.Count();
            ViewBag.sepet = sepet;
            var user = db.Users.Count();
            ViewBag.kullanici = user;

            return View();
        }
    }
}
```

And the following HTML code snippet creates an interface to display admin statistics.

```
<body>
    <div class="total-content">
        <div class="list_of_members">
            <div class="col-md-3">
                <div class="sales" style="background-color:#1abc9c">
                    <div class="icon-text">
                        <h3 style="color:white">@ViewBag.satis</h3>
                        <p style="color:white">Toplam Satışlar</p>
                    </div>
                    <div class="clearfix"></div>
                </div>
            </div>
            <div class="col-md-3">
                <div class="new-users" style="background-color:#2ecc71">

                    <div class="icon-text">
                        <h3 style="color:white">@ViewBag.urun</h3>
                        <p style="color:white">Ürün Sayısı</p>
                    </div>
                    <div class="clearfix"></div>
                </div>
            </div>
            <div class="col-md-3">
                <div class="orders" style="background-color:#3498db">

                    <div class="icon-text">
                        <h3 style="color:white">@ViewBag.kategori</h3>
                        <p style="color:white">Kategori Sayısı</p>
                    </div>
                    <div class="clearfix"></div>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="visitors" style="background-color:#9b59b6">

                <div class="icon-text">
                    <h3 style="color:white">@ViewBag.sepet</h3>
                    <p style="color:white">Sepet Sayısı</p>
                </div>
                <div class="clearfix"></div>
            </div>
        </div>
        <br />
        <div class="col-md-3">
            <div class="visitors" style="background-color:#ff6a00; margin-top:10px">

                <div class="icon-text">
                    <h3 style="color:white">@ViewBag.kullanici</h3>
                    <p style="color:white">Kullanıcı Sayısı</p>
                </div>
                <div class="clearfix"></div>
            </div>
            <div class="clearfix"></div>
        </div>
    </div>
</div>
<div class="clearfix"></div>
```

Admin Product Controller: This controller allows users in the administrator role to perform CRUD (Create, Read, Update, Delete) operations on products. The controller interacts with product data and performs database operations using ProductRepository and DataContext. While the Index action displays the product list by paginating, the **Create action** allows creating a new product, the **Delete action** allows deleting a product, and the **Update action** allows updating an existing product. Additionally, the **CriticalStock action** lists products with stock levels of 50 or less, and the **StockCount partial view** shows the number of products with critical stock levels.

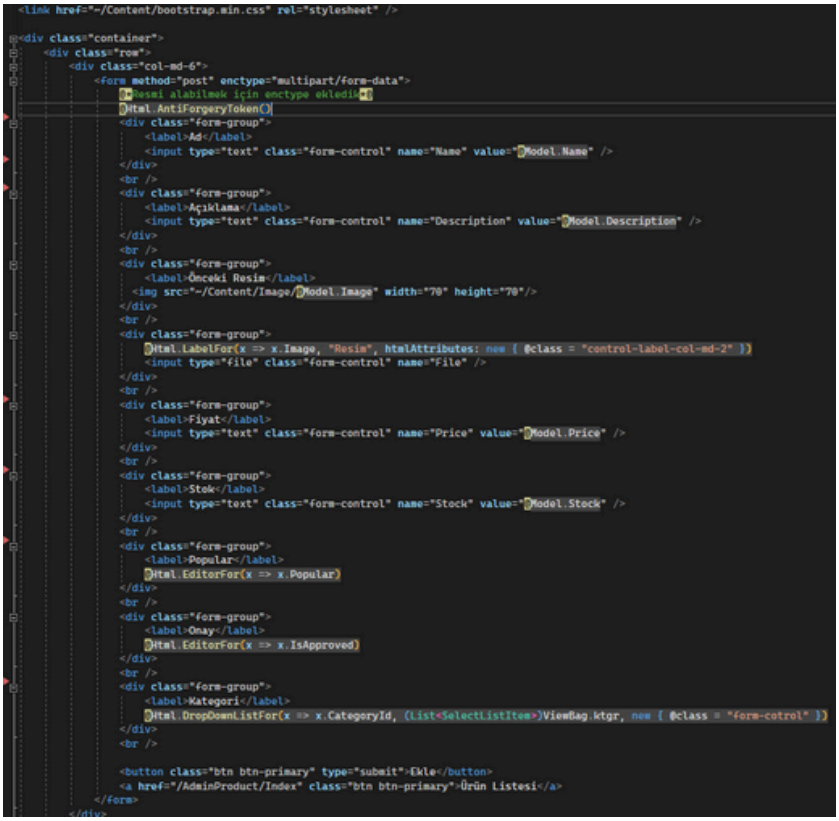
```
namespace E_Shop.Controllers
{
    [Authorize(Roles = "Admin")]
    0 references
    public class AdminProductController : Controller
    {
        ProductRepository productRepository = new ProductRepository();
        DataContext db = new DataContext();
        0 references
        public ActionResult Index(int sayfa=1)
        {
            return View(productRepository.List().ToPagedList(sayfa,3));
        }
        0 references
        public ActionResult Create()
        {
            List<SelectListItem> deger1 = (from i in db.Categories.ToList()
                                           select new SelectListItem
                                           {
                                               Text = i.Name,
                                               Value = i.Id.ToString()
                                           }).ToList();
            ViewBag.ktgr = deger1;
            return View();
        }

        [ValidateAntiForgeryToken]
        [HttpPost]
        0 references
        public ActionResult Create(Product data, HttpPostedFileBase File)
        {
            if (!ModelState.IsValid)
            {
                ModelState.AddModelError("", "Hata Oluştu");
            }

            string path = Path.Combine("~/Content/Image/" + File.FileName);
            File.SaveAs(Server.MapPath(path));
            data.Image = File.FileName.ToString();
            productRepository.Insert(data);
            return RedirectToAction("Index");

        }
```

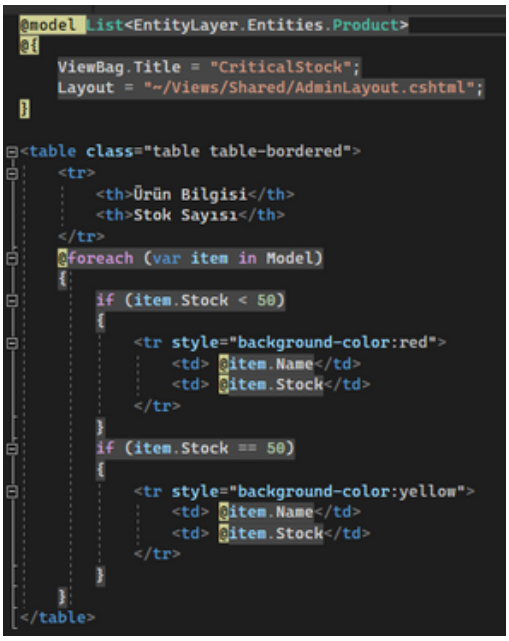
```
[ValidateAntiForgeryToken]
[HttpPost]
0 references
public ActionResult Update(Product data,HttpPostedFileBase File)
{
    var update = productRepository.GetById(data.Id);
    if (!ModelState.IsValid)
    {
        if (File == null)
        {
            update.Description = data.Description;
            update.Name = data.Name;
            update.IsApproved = data.IsApproved;
            update.Popular = data.Popular;
            update.Price = data.Price;
            update.Stock = data.Stock;
            update.CategoryId = data.CategoryId;
            productRepository.Update(update);
            return RedirectToAction("Index");
        }
        else
        {
            update.Description = data.Description;
            update.Name = data.Name;
            update.IsApproved = data.IsApproved;
            update.Popular = data.Popular;
            update.Price = data.Price;
            update.Stock = data.Stock;
            update.Image = File.FileName.ToString();
            update.CategoryId = data.CategoryId;
            productRepository.Update(update);
            return RedirectToAction("Index");
        }
    }
    return View(update);
```

```
public ActionResult Delete(int id)
{
    var delete=productRepository.GetById(id);
    productRepository.Delete(delete);
    return RedirectToAction("Index");
}
0 references
public ActionResult Update(int id)
{
    List<SelectListItem> deger1 = (from i in db.Categories.ToList()
                                   select new SelectListItem
                                   {
                                       Text = i.Name,
                                       Value = i.Id.ToString()
                                   }).ToList();
    ViewBag.ktgr = deger1;
    var update=productRepository.GetById(id);
    return View(update);
}
```

```
0 references
public ActionResult CriticalStock()
{
    var kritik = db.Products.Where(x => x.Stock <= 50).ToList();
    return View(kritik);
}
0 references
public PartialViewResult StockCount()
{
    if(User.Identity.IsAuthenticated)
    {
        var count=db.Products.Where(x=>x.Stock < 50).Count();
        ViewBag.count = count;
        var azalan = db.Products.Where(x => x.Stock == 50).Count();
        ViewBag.azalan = azalan;
    }
    return PartialView();
}
```

There are 5 view pages for the Admin product controller, including Create, Critical Stock, Index, StockCount and Update actions. The **Create.cshtml** page provides a form with several fields for creating a new product, including category selection and image upload. The **CriticalStock.cshtml** page allows the administrator to monitor critical stock situations by listing products with stock levels of 50 or less. The **Index.cshtml** page displays a paged list of products and allows edit and delete operations for each product. The **StockCount.cshtml** partial view shows the number of products at critical stock levels, and this information is often used in the main dashboard or other admin pages. Finally, the **Update.cshtml** page provides a form filled with existing product information, containing the required fields to update an existing product.





Admin Sales Controller: This controller provides a simple control for admin users to view sales. The **Index action** displays sales as a one-page list, showing 5 sales records on each page. And **Index** view allows administrators to view all sales in a one-page table, showing a specific number of sales records on each page. Thanks to page numbers, users can easily switch to different pages. This improves user experience when working with large data sets.

```
namespace E_Shop.Controllers
{
    [Authorize(Roles = "Admin")]
    0 references
    public class AdminSalesController : Controller
    {
        DataContext db = new DataContext();
        // GET: AdminSales
        0 references
        public ActionResult Index(int sayfa=1)
        {
            return View(db.Sales.ToList().ToPagedList(sayfa,5));
        }
    }
}
```

```
@using PagedList;
@using PagedList.Mvc;
@model IPagedList<EntityLayer.Entities.Sales>
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}

<h2>Satış Lisesi</h2>
<table class="table table-bordered">
    <tr>
        <th>Kullanıcı Bilgisi</th>
        <th>Id</th>
        <th>Ürün Adı</th>
        <th>Adet</th>
        <th>Fiyat</th>
        <th>Resim</th>
        <th>Tarih</th>
    </tr>
    @foreach (var item in Model)
    {
        <tbody>
            <tr>
                <td>@item.User.Name @item.User.SurName</td>
                <td>@item.Id</td>
                <td>@item.Product.Name    </td>
                <td>@item.Quantity </td>
                <td>@item.Price </td>
                <td><img src="~/Content/Image/@item.Product.Image" width="90" height="90" /></td>
                <td>@Convert.ToDateTime(item.Date).ToString("dd/MM/yyyy")</td>
            </tr>
        </tbody>
    }
</table>

@Html.PagedListPager(Model, sayfa => Url.Action("Index", new { sayfa }), PagedListRenderOptions.Classic)
```

Cart Controller: This controller provides various actions for users to manage shopping cart transactions. The **Index action** fetches the products in the user's cart after user authentication, calculates the total amount and sends it to the view. The **AddCart action** adds the product to the cart with the specified product ID; If the item is already in the cart, the quantity is increased and the total price is updated. The **TotalCount action** calculates the total number of items in the user's cart and displays it in a partial view. The **DynamicQuantity action** dynamically updates the product quantity based on the specified cart ID and recalculates the total price. The **reduce action** reduces the product quantity by one based on the cart ID; If the quantity is 1, it removes the item from the cart. The **increase action** increases the product quantity by one and recalculates the total price. The **Delete action** removes the product from the cart based on the specified cart ID. Finally, the **DeleteRange action** removes all items from the user's cart after the user is authenticated.

```
public class CartController : Controller
{
    DataContext db = new DataContext();
    0 references
    public ActionResult Index(decimal? Tutar)
    {
        if (User.Identity.IsAuthenticated)
        {
            var username = User.Identity.Name;
            var kullanici = db.Users.FirstOrDefault(x => x.Email == username);
            var model = db.Carts.Where(x => x.UserId == kullanici.Id).ToList();
            var kid = db.Carts.FirstOrDefault(x => x.UserId == kullanici.Id);
            if (model != null)
            {
                if (kid == null)
                {
                    ViewBag.Tutar = "Sepetinizde ürün bulunmamaktadır.";
                }
                else if (kid != null)
                {
                    Tutar = db.Carts.Where(x => x.UserId == kid.UserId).Sum(x => x.Product.Price * x.Quantity);
                    ViewBag.Tutar = "Toplam Tutar=" + Tutar + "TL";
                }
                return View(model);
            }
        }
        return HttpNotFound();
```

```
public ActionResult TotalCount(int?count)
{
    if (User.Identity.IsAuthenticated)
    {
        var model = db.Users.FirstOrDefault(x => x.Email == User.Identity.Name);
        count = db.Carts.Where(x=>x.UserId==model.Id).Count();
        ViewBag.count = count;
        if (count == 0)
        {
            ViewBag.count = "";
        }
    }
    return PartialView();
}
0 references
public void DinamikMiktar(int id, int miktari)
{
    var model = db.Carts.Find(id);
    model.Quantity=miktari;
    model.Price=model.Price*model.Quantity;
    db.SaveChanges();
}
0 references
public ActionResult azalt(int id)
{
    var model= db.Carts.Find(id);
    if (model.Quantity == 1)
    {
        db.Carts.Remove(model);
        db.SaveChanges();
    }
    model.Quantity--;
    model.Price=model.Price+model.Quantity;
    db.SaveChanges();
    return RedirectToAction("Index");

}
0 references
public ActionResult arttir(int id)
{
    var model = db.Carts.Find(id);
    model.Quantity++;
    model.Price = model.Price * model.Quantity;
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

```
public ActionResult AddCart(int id)
{
    if (User.Identity.IsAuthenticated)
    {
        var kullaniciadi = User.Identity.Name;
        var model = db.Users.FirstOrDefault(x => x.Email == kullaniciadi);
        var u = db.Products.Find(id);
        var sepet = db.Carts.FirstOrDefault(x => x.UserId == model.Id && x.ProductId == id);
        if (model != null)
        {
            if (sepet != null)
            {
                sepet.Quantity++;
                sepet.Price = u.Price * sepet.Quantity;
                db.SaveChanges();
                return RedirectToAction("Index", "Cart");
            }
            var s = new Cart
            {
                UserId = model.Id,
                ProductId = u.Id,
                Quantity=1,
                Price=u.Price,
                Date=DateTime.Now
            };
            db.Carts.Add(s);
            db.SaveChanges();
            return RedirectToAction("Index", "Cart");
        }
    }
    return View();
```

```
public ActionResult Delete(int id)
{
    var sil = db.Carts.Find(id);
    db.Carts.Remove(sil);
    db.SaveChanges();
    return RedirectToAction("Index");
}
0 references
public ActionResult DeleteRange()
{
    if (User.Identity.IsAuthenticated)
    {
        var kulllanici = User.Identity.Name;
        var model=db.Users.FirstOrDefault(x=>x.Email == kulllanici);
        var sil = db.Carts.Where(x=>x.UserId==model.Id);
        db.Carts.RemoveRange(sil);
        db.SaveChanges();
        return RedirectToAction("Index");

    }
    return HttpNotFound();
```

The **Index view** file of the Cart Controller allows users to manage the shopping cart. The view visually presents the contents of the cart by creating a table row for each item added to the cart. User information is taken from session variables and displayed in the header. For each product, there are columns containing product information, quantity, price, date, product image, deletion and purchase transactions.
Users can increase or decrease the quantity for each product. Buttons and some input boxes are provided for these operations. Quantity changes are handled dynamically using jQuery and an AJAX request is sent to the DynamicQuantity action. The total amount is displayed as the sum of the prices of all products added to the cart.
Additionally, users can purchase all products or delete them all from the cart. Relevant buttons are provided for these operations. CSS and jQuery were used for visual editing and user interaction. This way, users can manage their carts, edit product quantities, complete their purchases, and clear their carts when necessary.

```
@model List<EntityLayer.Entities.Cart>
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h2>@Session["Ad"] @Session["Soyad"]</h2>

<h3>Sepetim</h3>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>Ürün Bilgisi</th>
            <th>Adet</th>
            <th>Fiyat</th>
            <th>Tarih</th>
            <th>Resim</th>
            <th>Sil</th>
            <th>Satın Al</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>@item.Product.Name</td>

                <td>
                    <div>
                        @Html.ActionLink("-", "azalt", new { id = item.Id }, new { @class = "btn btn-info btn-sm azalt" })
                    </div>
                    <div>
                        @Html.TextBoxFor(x => item.Quantity, new { @class = "txt", data_id = item.Id })
                    </div>
                    <div>
                        @Html.ActionLink("+", "arttir", new { id = item.Id }, new { @class = "btn btn-info btn-sm arttir" })
                    </div>
                </td>
                <td>@item.Product.Price</td>

                <td>@Convert.ToDateTime(item.Date).ToString("dd/MM/yyyy")</td>
                <td><img src="~/Content/Image/@item.Product.Image" width="90" height="90" /></td>
                <td>@Html.ActionLink("Sil", "Delete", new { id = item.Id }, new { @class = "btn btn-danger", onclick = "return confirm('Silmek İstiyor musunuz?')" })</td>
                <td><a href="/Sales/Buy/@item.Id" class="btn btn-success">Satın Al</a></td>
            </tr>
        }
    </tbody>
</table>
<div class="container">@ViewBag.Tutar</div>
<br />
<div class="container form-group">
    <a href="/Sales/BuyAll" class="btn btn-success form-control">Tümünü Satın Al</a>
</div>

<div class="container form-group">
    <a href="/Cart/DeleteRange" class="btn btn-danger form-control">Hepsini Sil</a>
</div>
```

```
<style>
    .txt {
        line-height: 18px;
        padding: 0 4px 10px 4px;
        width: 50px !important;
        height: 32px;
        font-size: 14px;
        text-align: center;
        border: solid 1px #ccc;
        margin-left: 23px;
    }

    .arttir {
        display: block;
        cursor: pointer;
        border: solid 1px #ccc;
        position: absolute;
        margin-top: -32px;
        margin-left: 75px;
    }

    .azalt {
        display: block;
        border: solid 1px #ccc;
        cursor: pointer;
        position: absolute;
        margin-right: 90px;
    }
</style>
```

```
<script src="~/Scripts/jquery-3.0.0.min.js"></script>

<script>
    $(function () {

        $('.txt').on('change', function () {

            var miktar = $(this).val();
            var sepetid = $(this).attr('data_id');
            $.ajax({
                url: '/Cart/DinamikMiktar',
                data: { id: sepetid, miktari: miktar },
                success: function (res) {
                    document.location.reload();
                }
            });
        });

    });
</script>
```

Category Controller: The **CategoryList action** in this controller returns the list of all categories as a partial view, while the **Details action** displays the details of a specific category. This controller performs category operations through CategoryRepository, a data layer class that handles business logic.

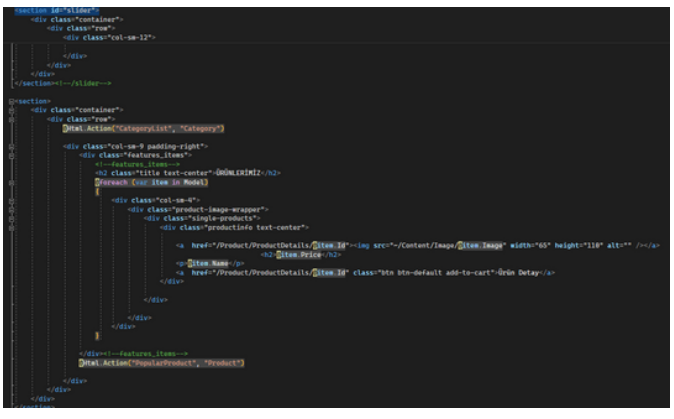
```
namespace E_Shop.Controllers
{
    0 references
    public class CategoryController : Controller
    {
        // GET: Category
        CategoryRepository categoryRepository= new CategoryRepository();
        0 references
        public PartialViewResult CategoryList()
        {
            return PartialView(categoryRepository.List());
        }
        0 references
        public ActionResult Details(int id)
        {
            var cat = categoryRepository.CategoryDetails(id);
            return View(cat);
        }
    }
}
```

CategoryList view displays the category list as a side menu. A category list creates a list item for each category, showing the category name and the number of products in that category. It also provides redirects to detailed category pages via clickable links to each category.

```
@model List<EntityLayer.Entities.Category>

<div class="col-sm-3" style="margin-top:30px">
    <div class="left-sidebar">
        <div class="brands_products">
            <h2>KATEGORİLER</h2>
            <div class="brands-name">
                @foreach (var item in Model)
                {
                    <ul class="nav nav-pills nav-stacked">
                        <li><a href="/Category/Details/@item.Id"><span class="pull-right">(@item.Products.Count)</span>@item.Name</a></li>
                    </ul>
                }
                <a href="/Product/AllProduct/" class="btn btn-outline-primary" style="position:center"><span class="pull-right"></span> Tümünü Gör</a>
            </div>
        </div>
    </div>
</div>
```

The Details view file contains the design of the product details page of an e-commerce site. The slider section consists of three different slides. Each slide contains a title, a subtitle, a description, and a button.
The products section has an area where the category list is on the left and the products are listed in the main content section. The category list provides quick access to different product categories. In the main content section, products are listed in individual boxes. Each product box contains a product image, price and name. Additionally, under each product there is a link to view the details of the product. Popular tools such as Bootstrap and jQuery were used for visual design and user experience.

```
<section id="slider">
    <div class="container">
        <div class="row">
            <div class="col-sm-12">
                <div id="slider-carousel" class="carousel slide" data-ride="carousel">
                    <ol class="carousel-indicators">
                        <li data-target="#slider-carousel" data-slide-to="0" class="active"></li>
                        <li data-target="#slider-carousel" data-slide-to="1"></li>
                        <li data-target="#slider-carousel" data-slide-to="2"></li>
                    </ol>

                    <div class="carousel-inner">
                        <div class="item active">
                            <div class="col-sm-6">
                                <h1><span>E</span>-SHOPPER</h1>
                                <h2>Free E-Commerce Template</h2>
                                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. </p>
                                <button type="button" class="btn btn-default get">Get it now</button>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/girl1.jpg" class="girl img-responsive" alt="" />
                                <img src="~/eshop/images/home/pricing.png" class="pricing" alt="" />
                            </div>
                        </div>
                        <div class="item">
                            <div class="col-sm-6">
                                <h1><span>E</span>-SHOPPER</h1>
                                <h2>100% Responsive Design</h2>
                                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. </p>
                                <button type="button" class="btn btn-default get">Get it now</button>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/girl2.jpg" class="girl img-responsive" alt="" />
                                <img src="~/eshop/images/home/pricing.png" class="pricing" alt="" />
                            </div>
                        </div>

                        <div class="item">
                            <div class="col-sm-6">
                                <h1><span>E</span>-SHOPPER</h1>
                                <h2>Free Ecommerce Template</h2>
                                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. </p>
                                <button type="button" class="btn btn-default get">Get it now</button>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/girl3.jpg" class="girl img-responsive" alt="" />
                                <img src="~/eshop/images/home/pricing.png" class="pricing" alt="" />
                            </div>
                        </div>

                    </div>

                    <a href="#slider-carousel" class="left control-carousel hidden-xs" data-slide="prev">
                        <i class="fa fa-angle-left"></i>
                    </a>
                    <a href="#slider-carousel" class="right control-carousel hidden-xs" data-slide="next">
                        <i class="fa fa-angle-right"></i>
                    </a>
```



Home Controller: This controller creates the home page that the user encounters when they log in to the site and lists the available products.

```
public class HomeController : Controller
{
    // GET: Home
    ProductRepository productRepository=new ProductRepository();
    0 references
    public ActionResult Index(int sayfa=1)
    {
        return View(productRepository.List().ToPagedList(sayfa,3));
    }
}
```

Index view file creates the design of the home page of an e-commerce site.
The slider section consists of three different slides. Each slide contains a different repair service. There are right and left arrow signs to switch between slides. The Products section includes a list of categories on the left and an area where products are listed in the main content section. The category list provides quick access to different product categories. In the main content section, products are listed in individual boxes. Each product box contains a product image, price and name.

```
<section id="slider">
    <!--slider-->
    <div class="container">
        <div class="row">
            <div class="col-sm-12">
                <div id="slider-carousel" class="carousel slide" data-ride="carousel">
                    <ol class="carousel-indicators">
                        <li data-target="#slider-carousel" data-slide-to="0" class="active"></li>
                        <li data-target="#slider-carousel" data-slide-to="1"></li>
                        <li data-target="#slider-carousel" data-slide-to="2"></li>
                    </ol>

                    <div class="carousel-inner">
                        <div class="item active">
                            <div class="col-sm-6">

                                <h1>Ekran Değişimi</h1>
                                <p>Telefonunuzun ekranı kırıldı mı veya çatladı mı? Profesyonel ekran değişimi hizmetimiz ile telefonunuzun ekranını ilk günkü gibi yeniliyoruz.</p>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/ekran.jpg" class="girl img-responsive" alt="" />
                            </div>
                        </div>
                        <div class="item">
                            <div class="col-sm-6">

                                <h1>Pil Değişimi</h1>
                                <p>Telefonunuzun pili hızlı mı tükeniyor? Pil ömrü azalan veya şişme gibi sorunlar gösteren pillerinizi değiştiriyoruz.</p>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/pil.jpg" class="girl img-responsive" alt="" />
                            </div>
                        </div>

                        <div class="item">
                            <div class="col-sm-6">

                                <h1>Şarj Soketi Tamiri</h1>
                                <p>Telefonunuz şarj olmuyor mu? Şarj bağlantısında sorun mu var? Şarj soketi tamiri veya değişimi ile bu problemi çözüyoruz. </p>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/soket.jpg" class="girl img-responsive" alt="" />
                            </div>
                        </div>
                        <div class="item">
                            <div class="col-sm-6">

                                <h1>Diğer Hizmetler</h1>
                                <p>İhtiyacınız olan tüm bakım ve onarım işlemleri için bizimle iletişime geçebilirsiniz.</p>
                                <p><br>0555 555 55 55</br></p>
                            </div>
                            <div class="col-sm-6">
                                <img src="~/eshop/images/home/onarım.jpg" class="girl img-responsive" alt="" />
                            </div>

                        </div>

                    </div>
```

```
<section>
    <div class="container">
        <div class="row">
            @Html.Action("CategoryList", "Category")

            <div class="col-sm-9 padding-right">
                <div class="features_items">
                    <!--features_items-->
                    <h2 class="title text-center">ÜRÜNLERİMİZ</h2>
                    @foreach (var item in Model)
                    {
                        <div class="col-sm-4">
                            <div class="product-image-wrapper">
                                <div class="single-products">
                                    <div class="productinfo text-center">

                                        <a href="/Product/ProductDetails/@item.Id"><img src="~/Content/Image/@item.Image" width="65" height="150" alt="" /></a>
                                        <h2>@item.Price</h2>
                                        <p>@item.Name</p>
                                        <a href="/Product/ProductDetails/@item.Id" class="btn btn-default add-to-cart">Ürün Detay</a>
                                    </div>

                                </div>

                            </div>
                        </div>
                    }

                    @Html.PagedListPager(Model, sayfa => Url.Action("Index", new { sayfa }), PagedListRenderOptions.Classic)


                </div><!--features_items-->
                @Html.Action("PopularProduct", "Product")

            </div>
        </div>
    </div>
</section>
```

Product Controller: The PopularProduct action returns a partial view to display popular products. It retrieves popular products using the ProductRepository class and passes them to the view via ViewBag. The ProductDetails action is used to display details of a specific product. It retrieves the details of a particular product using the ProductRepository class and also pulls data from the Comments table to retrieve comments for that product.Comment HTTP POST action allows users to comment on the product. Comment data is retrieved and added to the Comments table, and the changes are saved in the database. If the user is not authenticated, comments cannot be added and they remain on the same page. The AllProduct action returns a partial view to list all products.

```
public class ProductController : Controller
{
    // GET: Product
    ProductRepository productRepository = new ProductRepository();
    DataContext db = new DataContext();
    0 references
    public PartialViewResult PopularProduct()
    {
        var product = productRepository.GetPopularProduct();
        ViewBag.popular = product;
        return PartialView();
    }
    [Route("Product/{id}/{name}")]
    0 references
    public ActionResult ProductDetails(int id)
    {
        var details = productRepository.GetById(id);
        var yorum = db.Comments.Where(x => x.ProductId == id).ToList();
        ViewBag.yorum = yorum;
        return View(details);
    }
    [HttpPost]
    0 references
    public ActionResult Comment(Comment data)
    {
        if (User.Identity.IsAuthenticated)
        {
            db.Comments.Add(data);
            db.SaveChanges();
            return RedirectToAction("Index", "Home");
        }
        return View();
    }
    0 references
    public PartialViewResult AllProduct()
    {
        var allproduct = productRepository.List();
        return PartialView(allproduct);
    }
}
```

The **AllProduct view** file is used to list all products on an e-commerce site.

```
<body>
    <br />
    <header id="header">
        <!--header-->
        @Html.Action("CategoryList", "Category")
        <div class="col-sm-9 padding-right">
            <div class="features_items">
                <!--features_items-->
                <h2 class="title text-center">Tüm Ürünler</h2>
                @foreach (var item in Model)
                {

                    <div class="col-sm-4">
                        <div class="product-image-wrapper">
                            <div class="single-products">
                                <div class="productinfo text-center">
                                    <img src="~/Content/Image/@item.Image" alt="" />
                                    <h2>@item.Price</h2>
                                    <p>@item.Description</p>
                                    <a href="/Cart/AddCart/@item.Id" class="btn btn-default add-to-cart"><i class="fa fa-shopping-cart"></i>Sepete Ekle</a>
                                </div>

                            </div>
                        </div>


                    </div>



                }

            </div><!--features_items-->
        </div>
    </header>

    <script src="js/jquery.js"></script>
    <script src="js/price-range.js"></script>
    <script src="js/jquery.scrollUp.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/jquery.prettyPhoto.js"></script>
    <script src="js/main.js"></script>
</body>
</html>
```

The **PopularProduct view** file is used to show popular products on an e-commerce site. **ProductDetails view** file is used to show details of a product. The page contains the product's image, price, description, stock and other features.

```
<h2 class="title text-center">POPÜLER ÜRÜNLERİMİZ</h2>

<div id="recommended-item-carousel" class="carousel slide" data-ride="carousel">
    <div class="carousel-inner">
        <div class="item active">
            @foreach (var item in ViewBag.popular)
            {
                <div class="col-sm-4">
                    <div class="product-image-wrapper">
                        <div class="single-products">
                            <div class="productinfo text-center">
                                <a href="/Product/ProductDetails/@item.Id">
                                    <img src="~/Content/Image/@item.Image" width="30" height="150" alt="" />
                                </a>
                                <h2>@item.Price</h2>
                                <p>
                                    <a href="/Product/ProductDetails/@item.Id">@item.Name</a>
                                </p>
                            </div>

                        </div>
                    </div>
                </div>
            }


        </div>
        <div class="item">
            @foreach (var item in ViewBag.popular)
            {
                <div class="col-sm-4">
                    <div class="product-image-wrapper">
                        <div class="single-products">
                            <div class="productinfo text-center">
                                <a href="/Product/ProductDetails/@item.Id">
                                    <img src="~/Content/Image/@item.Image" width="30" height="150" alt="" />
                                </a>
                                <h2>@item.Price</h2>
                                <p>
                                    <a href="/Product/ProductDetails/@item.Id">@item.Name</a>
                                </p>
                            </div>

                        </div>
                    </div>
                </div>}

        </div>
    </div>
    <a class="left recommended-item-control" href="#recommended-item-carousel" data-slide="prev">
        <i class="fa fa-angle-left"></i>
    </a>
    <a class="right recommended-item-control" href="#recommended-item-carousel" data-slide="next">
        <i class="fa fa-angle-right"></i>
    </a>
</div>
```

```
<html lang="en">
<body>
    <section>
        <div class="container">
            <div class="row">
                @Html.Action("CategoryList", "Category")
                <div class="col-sm-9 padding-right">
                    <div class="product-details" >
                        <!--product-details-->
                        <div class="col-sm-5">
                            <div class="view-product">
                                <img src="~/Content/Image/@Model.Image" style="width:300px; height:240px; margin-top:20px" alt="@Model.Name" />

                            </div>
                        </div>
                        <div class="col-sm-7">
                            <div class="product-information">

                                <h2>Ürün Adı: @Model.Name</h2>
                                <span>
                                    <span>@Model.Price TL</span>

                                    <a class="btn btn-default cart" style="margin-top:15px" href="/Cart/AddCart/@Model.Id">
                                        <i class="fa fa-shopping-cart"></i>
                                        Sepete Ekle
                                    </a>
                                </span>
                                <p><b>Kategori:</b> @Model.Category.Name</p>
                                <p><b>Stok:</b> @Model.Stock</p>
                                <p>
                                    <b>Popüler:</b>
                                    @if (Model.Popular == true)
                                    {
                                        <b style="color:green"> Evet </b>
                                    }
                                    else
                                    {
                                        <b style="color:red"> Hayır </b>
                                    }
                                </p>
                                <p>
                                    <b>Onay:</b>
                                    @if (Model.IsApproved == true)
                                    {
                                        <b style="color:green"> Evet </b>
                                    }
                                    else
                                    {
                                        <b style="color:red"> Hayır </b>
                                    }
                                </p>
                                <p><b>Açıklama:</b> @Model.Description</p>
                                <a href=""><img src="images/product-details/share.png" class="share img-responsive" alt="" /></a>
                            </div><!--/product-information-->
                        </div>
                        <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
                            Yorum Yap
                        </button>
```

```
<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <form method="post" action="/Product/Comment">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">Yorum Yazınız</h5>

                </div>
                <div class="modal-body">
                    <textarea class="form-control" name="Contents" placeholder="Yorum Yazınız..."></textarea>
                    <input type="hidden" name="ProductId" value="@Model.Id" />
                    <input type="hidden" name="UserId" value="@Session["userid"]" />
                    <input type="hidden" name="Date" value="@DateTime.Now" />
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Kapat</button>
                    <button type="submit" class="btn btn-primary">Gönder</button>
                </div>
            </form>
        </div>
    </div>
</div>


<hr />
<div class="row">
    <div class="col-md-4 col-sm-4">


        @foreach (var item in ViewBag.yorum)
        {
            <h4 style="color:coral">
                @item.Date <br /><br />     @item.User.Name @item.User.SurName
            </h4>
            <p>@item.Contents </p>
            <hr />
        }



    </div>
</div><!--/product-details-->


@Html.Action("PopularProduct", "Product")
```

Sales Controller: This Controller class processes purchases in response to various HTTP requests. The **Index** method retrieves the products purchased by the logged in user and presents them to the user. The **Buy** method redirects to the purchase page of a particular product. The **Buy2** method purchases a specific card item. If the model validation is valid, it ensures that the products from the cart are transferred to the sales table and the cart is cleared. The **Buyall** method calculates the total amount of the products in the entire basket and presents it to the user. The **BuyAll2** method buys all cart items. A sales record is created for each product and the cart is cleared. The user is then directed to the shopping cart.

```
public class SalesController : Controller
{
    DataContext db = new DataContext();
    0 references
    public ActionResult Index(int sayfa = 1)
    {
        if (User.Identity.IsAuthenticated)
        {
            var kullaniciadi = User.Identity.Name;
            var kullanici = db.Users.FirstOrDefault(x => x.Email == kullaniciadi);
            var model = db.Sales.Where(x => x.UserId == kullanici.Id).ToList().ToPagedList(sayfa, 5);
            return View(model);
        }
        return HttpNotFound();
    }
    0 references
    public ActionResult Buy(int id)
    {
        var model = db.Carts.FirstOrDefault(x => x.Id == id);
        return View(model);
    }
    [HttpPost]
```

```
public ActionResult Buyall(decimal? Tutar)
{
    if (User.Identity.IsAuthenticated)
    {
        var kullaniciadi = User.Identity.Name;
        var kullanici=db.Users.FirstOrDefault(x=>x.Email == kullaniciadi);
        var model = db.Carts.Where(x => x.UserId == kullanici.Id).ToList();
        var kid=db.Carts.FirstOrDefault(x=>x.UserId==kullanici.Id);
        if (model != null)
        {
            if(kid == null)
            {
                ViewBag.Tutar = "Sepetinize ürün bulunmamaktadır.";
            }
            else if(kid !=null)
            {
                Tutar=db.Carts.Where(x=>x.UserId==kid.UserId).Sum(x=>x.Product.Price*x.Quantity);
                ViewBag.Tutar = "Toplam Tutar=" + Tutar + "TL";
            }
            return View(model);
        }
        return View();
    }
    return HttpNotFound();
}
```

```
public ActionResult Buy2(int id)
{
    try
    {
        if (ModelState.IsValid)
        {
            var model = db.Carts.FirstOrDefault(x => x.Id == id);
            var satis = new Sales //sepettekileri alıp sarışlara atıp sepeti temizleme
            { // satın alınan ve sepet birbirine eşleniyor
                UserId = model.UserId,
                ProductId = model.ProductId,
                Quantity = model.Quantity,
                Image = model.Image,
                Price = model.Price,
                Date = DateTime.Now,
            };
            db.Carts.Remove(model);
            db.Sales.Add(satis);
            db.SaveChanges();
            ViewBag.islem = "Satın alma işlemi başarılı bir şekilde gerçekleşmiştir";
        }
    }
    catch (Exception)
    {
        ViewBag.islem = "Satın alma işlemi başarısız";
    }
    return View("islem");
```

```
public ActionResult BuyAll2()
{
    var username=User.Identity.Name;
    var kullanici=db.Users.FirstOrDefault(x=>x.Email == username);
    var model = db.Carts.Where(x => x.UserId == kullanici.Id).ToList();
    int row = 0;

    foreach (var item in model)
    {
        var satis = new Sales
        {
            UserId = model[row].UserId, //ne kadar veri varsa satırınca
            ProductId = model[row].ProductId,
            Quantity = model[row].Quantity,
            Price = model[row].Price,
            Image = model[row].Image,
            Date = DateTime.Now
        };
        db.Sales.Add(satis);
        db.SaveChanges();
        row++;
    }
    db.Carts.RemoveRange(model); //Tüm verileri sildirir.
    db.SaveChanges();
    return RedirectToAction("Index","Cart");
```

The **Buy view** shows the user information such as product name, quantity, price and allows the user to confirm this information and make the purchase. There is also an option to cancel on the form. This view retrieves the details of the Cart asset and displays it to the user. After reviewing these details, the user can make the purchase or cancel the transaction.

```
@model EntityLayer.Entities.Cart
@{
    ViewBag.Title = "Buy";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<div class="modal-header">
    <h2>Satın Al</h2>
</div>
<div class="container">
    @*//<form></form> da kullanılabilir.*@
    @using (Html.BeginForm("Buy2", "Sales", FormMethod.Post))
    {
        <div class="modal-body">
            @Html.HiddenFor(x => x.Id)

            <div class="form-group">
                Kullanıcı Adı Soyadı:@Session["Ad"] @Session["Soyad"]
            </div>
            <div class="form-group">
                Ürün Bilgisi: @Html.DisplayFor(x => x.Product.Name, new { @class = "form-control" })
            </div>
            <div class="form-group">
                Adet: @Html.DisplayFor(x => x.Quantity, new { @class = "form-control" })
            </div>
            <div class="form-group">
                <img src="@Model.Product.Image" width="90" height="90" />
            </div>
            <div class="form-group">
                Fiyat: @Html.DisplayFor(x => x.Price, new { @class = "form-control" })
            </div>
        </div>
        <div class="modal-footer" style="float:right">
            <button class="btn btn-success">Satın Al</button>
            <a href="/Cart/Index" class="btn btn-dark">İptal</a>
        </div>
    }
</div>
```

```
<div class="container">
    <div class="container">
        <h2>Merhaba @Session["Ad"] @Session["Soyad"] Satın Alma Sayfasına Hoşgeldiniz</h2>
    </div>


    <div class="container">
        @using (Html.BeginForm("BuyAll2", "Sales", FormMethod.Post))
        {

            foreach (var item in Model)
            {
                <div class="form-group">
                    <h5>ID: @Html.DisplayFor(x => item.Id)</h5>
                </div>
                <div class="form-group">
                    <h5>Ürün Ad: @Html.DisplayFor(x => item.Product.Name)</h5>
                </div>
                <div class="form-group">
                    <h5>Adet: @Html.DisplayFor(x => item.Quantity)</h5>
                </div>
                <div class="form-group">
                    <h5>Tutar: @Html.DisplayFor(x => item.Price)</h5>
                </div>
                <div class="form-group">
                    <h5>Resim: <img src="~/Content/Image/@item.Product.Image" width="50" height="50" /></h5>
                </div>

            }
            <p style="color:red; text-align:center">@ViewBag.Tutar</p>
            <div class="container">
                <div class="form-group" style="float:left">
                    <button class="btn btn-success">Onay</button>
                </div>
                <div style="float:right">
                    <button href="/Cart/Index" class="btn btn-danger">İptal</button>
                </div>
            </div>

        }
    </div>
</div>
```

The **BuyAll view** creates a page for the user to purchase all the items in the cart.

The **Index view** file is used to display a sales list. The page starts with a table containing table titles and sales. Sales are listed using a loop within the table. For each sale, ID, product name, quantity, price, image and date columns are displayed.

```
<h2>Satış Lisesi</h2>
<table class="table table-bordered">
    <tr>
        <th>Id</th>
        <th>Ürün Adı</th>
        <th>Adet</th>
        <th>Fiyat</th>
        <th>Resim</th>
        <th>Tarih</th>
    </tr>
    @foreach (var item in Model)
    {
        <tbody>
            <tr>
                <td>@item.Id</td>
                <td>@item.Product.Name    </td>
                <td>@item.Quantity </td>
                <td>@item.Price </td>
                <td><img src="~/Content/Image/@item.Product.Image" width="90" height="90" /></td>
                <td>@Convert.ToDateTime(item.Date).ToString("dd/MM/yyyy")</td>
            </tr>
        </tbody>
    }
</table>

@Html.PagedListPager(Model, sayfa => Url.Action("Index", new { sayfa }), PagedListRenderOptions.Classic)
```

User Controller: This Controller class allows users to update their profiles and reset their passwords. Update action is used to update the user profile. This action accepts HTTP POST requests and retrieves the user's new information and updates it in the database. The PasswordReset action is used to reset the user's password. The action retrieves an email address, sends a reset code to that address, then updates the user's password with that code. If no user matching the given email address is found, it returns a warning message.

```
public class UserController : Controller
{
    DataContext db=new DataContext();
    // GET: User
    0 references
    public ActionResult Update()
    {
        var username = (string)Session["Mail"];
        var degerler = db.Users.FirstOrDefault(x => x.Email == username);
        return View(degerler);
    }
    [HttpPost]
    0 references
    public ActionResult Update(User data)
    {
        var username = (string)Session["Mail"];
        var user = db.Users.Where(x => x.Email == username).FirstOrDefault();
        user.Name = data.Name;
        user.Email = data.Email;
        user.SurName = data.SurName;
        user.Password = data.Password;
        user.RePassword = data.RePassword;
        user.UserName = data.UserName;
        db.SaveChanges();
        return RedirectToAction("Index","Home");
    }

    0 references
    public ActionResult PasswordReset()
    {
        return View();
    }
```

```
public ActionResult PasswordReset(string eposta)
{
    var mail=db.Users.Where(x=>x.Email == eposta).SingleOrDefault();
    if(mail != null)
    {
        Random rnd=new Random();
        int yenisifre=rnd.Next();
        User sifre = new User();
        mail.Password = (Convert.ToString(yenisifre));
        db.SaveChanges();
        WebMail.SmtpServer = "smtp.gmail.com";
        WebMail.EnableSsl = true;
        WebMail.UserName = "selvi.kdb@gmail.com";
        WebMail.Password = "SELvi.1999";
        WebMail.SmtpPort = 587;
        WebMail.Send(eposta, "Giriş Şifreniz", "Şifreniz" + yenisifre);
        ViewBag.uyari = "Şifreniz başarıyla gönderilmiştir";
    }
    else
    {
        ViewBag.uyari = "Hata Oluştu Tekrar Deneyiniz";
    }
    return View();
```

The **Update view** is used to update users' profiles. The form allows the user to update their first name, last name, username, email address, and password using the properties of the User class. When users enter their updated information and click the "Update" button, this information is posted as an object of the User class.

The **PasswordReset view** is used to reset users' passwords. When users enter their e-mail addresses and click on the "Reset Password" button, the passwords of users with matching e-mail addresses in the system are reset and a new password is created.

```
@model EntityLayer.Entities.User
@{
    Layout = null;
}
<link href="~/eshop/css/bootstrap.min.css" rel="stylesheet" />

<form method="post">
    <div class="container">
        <br/>
        <div class="form-group">
            <label>Ad:</label>
            <input type="text" class="form-control" name="Name" value="@Model.Name" />
            @Html.ValidationMessageFor(x => x.Name, "", new { @style = "color:red" })
        </div>
        <div class="form-group">
            <label>Soyad:</label>
            <input type="text" class="form-control" name="SurName" value="@Model.SurName"/>
            @Html.ValidationMessageFor(x => x.SurName, "", new {@style="color:red"})
        </div>
        <div class="form-group">
            <label>Kullanıcı Adı:</label>
            <input type="text" class="form-control" name="UserName" value="@Model.UserName" />
            @Html.ValidationMessageFor(x => x.UserName, "", new { @style = "color:red" })
        </div>
        <div class="form-group">
            <label>EMail</label>
            <input type="email" class="form-control" name="Email" value="@Model.Email" />
            @Html.ValidationMessageFor(x => x.Email, "", new { @style = "color:red" })
        </div>
        <div class="form-group">
            <label>Şifre</label>
            <input type="password" class="form-control" name="Password" value="@Model.Password" />
            @Html.ValidationMessageFor(x => x.Password, "", new { @style = "color:red" })
        </div>
        <div class="form-group">
            <label>Şifre</label>
            <input type="password" class="form-control" name="RePassword" value="@Model.RePassword" />
            @Html.ValidationMessageFor(x => x.RePassword, "", new { @style = "color:red" })
        </div>

        <button class="btn btn-primary">Güncelle</button>
    </div>
</form>
```

```
<div class="alert alert-success">@ViewBag.uyari</div>
@using (Html.BeginForm("PasswordReset", "User", FormMethod.Post, new { enctye = "multipart/form-data" }))
{
    <br />
    <br />
    <div class="container">
        <div class="form-group">
            <input type="email" name="eposta" placeholder="E-Mail Adresinizi Girin"/>
        </div>
        <div class="form-group">
            <input type="submit" value="Şifreyi Sıfırla" class="btn btn-success"/>
        </div>

    </div>
```

OTHER FOLDERS AND FEATURES

Admin Layout and Layout are view layout methods used in ASP.NET MVC applications. These structures are used to manage repetitive structures and style definitions in web applications.

Layout is the basic layout template generally used on all pages of the web application. This template determines the overall look and style of the web application. All pages are created according to this layout and the content is placed within this layout.

Admin Layout is a general layout template used for a specific section such as an admin panel or backend. This layout represents a special interface that administrators or system administrators typically access.

The **AdminRoleProvider.cs** page contains the AdminRolProvider class, which is a custom role provider class for managing user roles in ASP.NET applications. Role providers are used to define users' roles, determine which roles users are in, and perform other functions associated with the roles.

The **NiceAdmin folder** contains code template we use in our project.