Veritabanı Yönetim Sistemleri Projesi
Mağaza Otomasyon

Miray GÜRBÜZ
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
221307031

Muhammed Abdullah ACAR
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
221307038

***Özet*—Bu rapor, VTYS dersi dönem projesi için yaptığımız web tabanlı mağaza otomasyon uygulamasının ayrıntılarını içerir: Projenin amacı, veritabanı tasarımı ve işlemleri, ER diyagramı, admin ve kullanıcı panelleri üzerinden yapılabilecek işlemler vs.**

***Anahtar Kelimeler*—ASP.NET, C#, SQL, HTML/CSS, veritabanı, ER diyagramı**

I. Problem Tanımı

Giyim mağazalarının günlük işlemlerini daha kolay ve verimli bir şekilde yönetebilmesi için bir otomasyon sistemine entegre olmaya ihtiyaçları vardır. Bu şekilde işlemler otomatikleştirilecek ve daha az iş gücü harcanarak daha verimli işlemler yapılacaktır.

Bir mağazanın ihtiyaçlarını düşündüğümüzde, akla ilk gelen stok takibi oluyor. Mağazalar, birçok satış ve tedarik işlemi yapıyor ve bu işlemlerin kaydını geleneksel yöntemlerle tutmak hem zor hem de güvenilir değil.

Satış ve tedarik işlemleri sonrasında ürünlerin stok miktarında ve mağazanın kasasındaki gelir/gider durumunda değişiklik gözlemleniyor. Tüm bunların takibini ve analizini yapmak bir insan için oldukça zorlayıcı.

Enflasyon dolayısıyla sürekli değişiklik gösteren fiyatlar ürünlere de yansıyacaktır. Aynı üründen mağazada yüzlerce adet olduğunu varsayarsak her birinin fiyat etiketini teker teker değiştirmek pratik değil, tek işlemle yüzlerce ürünün fiyatını değiştirmek otomasyon sistemiyle mümkün olabilir.

Ayrıca bünyesinde birçok farklı marka ve kategoriden ürün barındıran bir mağazanın çalışanının, gelen herhangi bir müşterinin istediği marka ve kategori kombinasyonuna ait bir ürünü analog yöntemlerle arayıp müşteriye yardımcı olması zorlayıcı olacaktır. "X markasının Y ürününü istiyorum" diyen bir müşterinin talebi üzerine ürünü aramaya başlamadan önce o marka-kategori kombinasyonuna sahip ürünün mağazadaki mevcudiyet durumunu kontrol etmek daha mantıklıdır.

Müşteriler online sipariş verebilmelidir. Böylece mağaza daha fazla kişiye ulaşacak ve daha fazla kazanç sağlayacaktır.

Bir Mağaza Otomasyon Sisteminde Olması Gerekenler:
Mağaza Yönetimi İçin

- Ürün ekleme, listeleme, güncelleme, satış durumu belirleme
- Tedarik, satış kaydı oluşturma, listeleme ve bunlara bağlı olarak ilgili ürünün stok durumu ile mağazanın gelir-gider durumunun güncellemesi
- Kategori ve marka ekleme, listeleme, güncelleme
- Mağaza sistemine kayıt olmuş kullanıcı bilgilerini görüntüleme

Online Alışveriş İçin Kullanıcılar;

- Sisteme kayıt olabilmeli ve giriş yapabilmeli,
- Satışta olan ürünleri görüntüleyebilmeli,
- Marka ve kategoriye göre ürünleri listeleyebilmeli ve
- İstediği ürün için sipariş verebilmelidir.

Veritabanında Oluşturulması Gereken Tablolar:

- Ürün Tablosu
- Kategori Tablosu
- Marka Tablosu
- Satış Tablosu
- Tedarik Tablosu
- Sipariş Tablosu
- Kullanıcılar Tablosu
- Gelir Gider Tablosu
- Gelir Gider Tür Tablosu

II. Yapılan Araştırmalar

Başlangıç: Bu proje için bir web uygulaması geliştirmemiz gerekiyordu. Bunun için ilk olarak hangi framework ile çalışacağımıza karar verdik: *ASP.NET*

Daha önce bu konuda herhangi bir tecrübemiz yoktu ve birinci sınıfta Algoritma ve Programlama dersinde gördüğümüz C# programlama dilini kullanarak arkaplan kodlarını yazabileceğimizden bu framework'ü seçtik. Aynı zamanda aşina olduğumuz Visual Studio Community 2022 IDE'sini kullanarak sürükle-bırak işlemi ile web sayfaları tasarlayabiliyor oluşumuz işimizi kolaylaştırdı.

ASP.NET hakkında fikir sahibi olabilmek adına, YouTube üzerinde basit bir ASP.NET projesi geliştirilen oynatma listesinden yararlandık.[1]

Veritabanı yönetimi için Microsoft SQL Server Management Studio'yu tercih ettik. Veritabanı tasarımını kesinleştirdikten sonra projemizde kullanacağımız tabloları ve ilişkilerini oluşturduk. Ancak projemizin ilerleyen kısımlarında bunun yeterli olmadığını, VIEW ve TRIGGER kullanarak işimizin kolaylaşacağını fark ettik. Örneğin, mağazada bir tedarik işlemi yapıldığında yani tedarik tablosuna bir kayıt eklendiğinde aynı zamanda gelir gider tablosunda 'gider' olarak eklenecek bir kayıt daha oluşturulması gerekiyordu. Yine

tedarik işlemi sonrasında ilgili ürünün stoğunun tedarik miktarı kadar artması gerekiyordu. Bu işlemleri C# kodunda metotlar oluşturarak, uzun uzun SQL betikleri yazarak yapmak pratik değildi. Onun yerine veritabanında aynı işlevi sağlayacak iki adet TRIGGER oluşturmak daha mantıklıydı[2]. Uzun INNER JOIN işlemlerini tekrar tekrar yazmak yerine veritabanında bir VIEW oluşturmak daha pratik oldu[3]. Ek olarak, yazdığımız sorguların daha hızlı çalışması için INDEXler oluşturduk.

Arayüz Tasarımı: Sürükle-bırak işlemi ile sayfa tasarlayabiliyor olsak bile bu sayfaların gerek estetiği konusunda, gerek bazı özellikler ekleme konusunda belli başlı sıkıntılar yaşadık. Örneğin, her sayfanın üst kısmında bulunan bir gezinti çubuğu eklemek istiyorduk. Bunu sağlamak için hazır Bootstrap navbar şablonlarını kullandık[4] ve gezinti çubuğunun her sayfada gözükmesi için master page olarak oluşturduk[5].

Kullanıcıların girdiğinde ürünleri görüntülemesini sağlamak adına bir ürün listeleme HTML + CSS şablonu kullandık[6]. Projemize uygun şekilde düzenledik ve ürünlerin altında bulunan butonlar için başka bir CSS şablonu kullandık[7].

Diğer sayfalar için genel olarak sürükle-bırak işlemi sonrası CSS ile sayfa tasarımı yaptık.

Kodlama: Veritabanı ve arayüz tasarımı tamam. Bir sonraki aşamada veritabanına bağlantı sağlamamız gerekiyordu.

Öncelikle bu işlemler için "System.Data.SqlClient" namespace'ini kullanmalıydık[10].

Ardından kullandığımız veritabanı yönetim uygulamasından connection stringimizi aldık ve bu connection string ile bir bağlantı oluşturduk [8]. Bu işlemi tekrar tekrar yapmamak için projemizde SqlConnectionClass adında bir class oluşturduk ve bağlantı açmak, kapamak için iki adet public static metot yazdık.

Devamında yaptığımız işlemlerimiz özetle ne tür bir metot oluşturacağımızı seçmek, bağlantı açmak ve ilgili SQL komutlarını SqlCommand sınıfı ile oluşturarak işleme sokmaktı.[9]

Bu projede çoğunlukla ekleme ile ilgili metotlar yazdık; ürün ekleme, kategori-marka ekleme, kullanıcının sipariş verme işlemi, satış yapma, tedarik işlemi gerçekleştirme, kullanıcı kayıt olma işlemi vb. hepsi aslında veritabanındaki ilgili tablolara kayıt ekleme gerektiren işlemlerdi. Oluşturacağımız SqlCommand komutları içine parametreler kullanarak INSERT INTO işlemi ile bunu gerçekleştirebildik.

Listeleme işlemleri en çok yaptığımız işlemlerden biriydi. Verileri listelemek adına DataList ve GridView kullandık. Admin panelinde verileri listelemek için çoğunlukla GridView'u kullanırken, kullanıcıların görüntülediği sayfalarda DataList kullandık. Bunun sebebi; GridView verileri tablo halinde gösterirken, DataList'in satırlar halinde gösteriyor olmasıydı. HTML/CSS şablonlarında DataList kullanmayı boş bir tuvalin üstüne resim çizmeye benzetebiliriz[11]. Tüm bunlardan bahsetmişken, verileri listelemek için öncelikle veri okuma işlemi yapmamız gerek.

Veri okuma işlemi yaparken SqlDataReader ile tek satırlık verileri okuduğumuz için sınırlaması DataReader kapanması durumu olan while döngüsü işleme aldık. İlgili satırları yazdıktan sonra DataReader'i kapattık[12].

Verileri GridView ile listelerken, GridView sütunları veritabanından olduğu gibi çekiyordu. Yani, sütun adlarını veritabanındaki nitelik adlarıyla aynı isimde oluşturuyordu. Bu sorunu çözmek ve sütunları kendimiz adlandırabilmek için GridView'un .aspx uzantılı kodda "AutoGeneratedColumns"[13] alanını "false" olarak belirledik. TemplateField'larda sütunları isimlendirdik ve TemplateField'ların içerisinde olan ItemTemplate'lerin içerisinde veritabanındaki verileri bağladık.[14]

Session İşlemleri ve Alınan Güvenlik Önlemleri:

Projemizde işlemlerin çoğu admin panelinde yapılıyor, peki admin ve kullanıcı panellerine/arayüzlerine erişim nasıl sağlanıyor? Öncelikle, yaptığımız projede kullanıcılar sisteme kayıt olabiliyor ve hesaplarına giriş yapabiliyorlar. Veritabanında oluşturduğumuz kullanıcı tablosunda her kullanıcıya ait primary key olan kullaniciID alanı mevcut, diğer alanlarımız: kullaniciAdi, kullaniciSifre, kullaniciMail. Kullanıcı kayıt olma sayfasına girdiğinde, kullanıcı adı, şifresi ve mail adresi alanlarını eksiksiz biçimde doldurmalı. Eğer alanlar doldurulmazsa, kullanıcıya label aracılığı ile "Lütfen tüm alanları doldurunuz." şeklinde bir mesaj gösteriyoruz. Kayıt işlemi esnasında C# kodlarıyla kullanıcı adı ve mail adresi alanlarını kontrol ediyoruz ve eğer daha önce kullanılmışsa kullanıcıya yine label aracılığıyla "Kullanıcı adı veya mail daha önce kullanılmış." şeklinde bir mesaj gösteriyoruz. Bu şekilde giriş esnasında herhangi bir karmaşıklık oluşmuyor.

Veritabanımızda oluşturduğumuz ilk kullanıcı hesabı "admin" kullanıcı adına sahip olan hesap olmuştu. Uygulamamızda kullanıcı adı alanı hesaba özel olduğundan, "admin" kullanıcı adlı hesaba session[14] yardımı ile admin paneline erişim izni verdik. Admin, sisteme giriş yaptığında kullanıcı arayüzü yerine çeşitli ürün, satış, tedarik vb. gibi işlemler yapabileceği admin paneline yönlendiriliyor. Admin olmayan kullanıcılar arayüz üzerinden herhangi bir şekilde admin paneline erişim sağlayamıyor. Ancak oldu ki herhangi bir kullanıcı, adres çubuğundan admin paneline girilebilecek bağlantıyı yazdı; böyle bir durumda admin kullanıcısı dışında herhangi bir kullanıcı admin paneline erişmeye çalışırsa "Erişim izniniz yok." şeklinde bir pop-up mesajı alıp kullanıcı arayüzünün anasayfası olan urunleriListele.aspx uzantılı sayfaya yönlendiriliyor. Ayrıca, session timeout'u 5 dakika olarak ayarladık. Bu şekilde 5 dakika boyunca herhangi bir işlem yapılmazsa kullanıcının oturumuna son veriliyor. Ek olarak kullanıcı, navbarda bulunan çıkış yap butonu ile dilediği zaman çıkış yapabiliyor. Çıkış yap butonuna basıldığında, session terk ediliyor, çerezler siliniyor ve kullanıcı girisYap.aspx sayfasına yönlendiriliyor.

Bu işlemlerle erişim konusunda güvenlik önlemi almış olduk.

Başka bir güvenlik önlemi olarak şifreleme algoritmalarına denk geldik. SHA256[16] şifreleme algoritması ile, kullanıcı sisteme kayıt olduğu zaman kullanıcı şifresinin veritabanına 64 haneli harf ve sayı kombinasyonu olarak aktarılmasını sağladık.

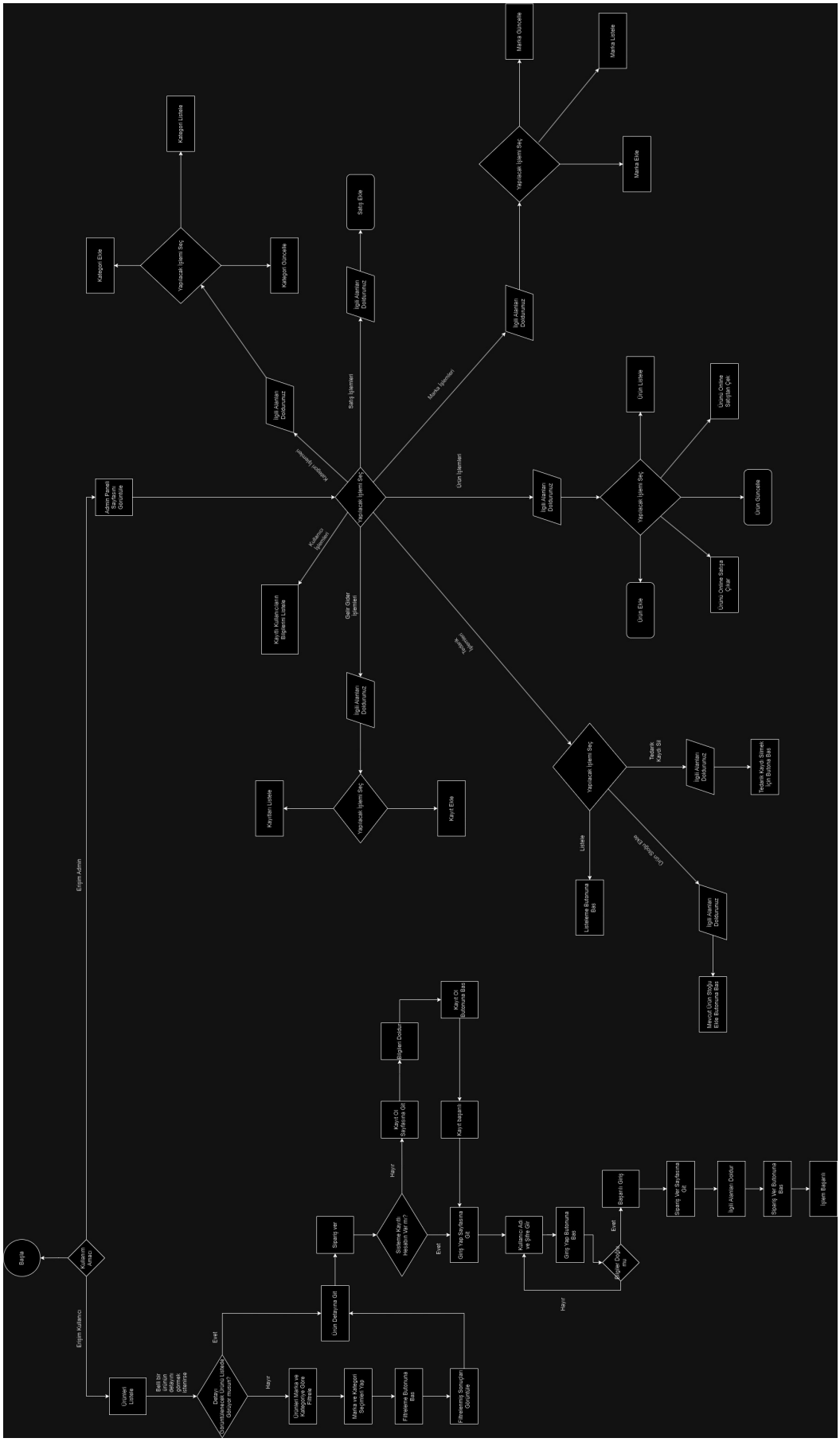

Fig. 1: **Akış Şeması** (draw.io ile oluşturulmuştur.)

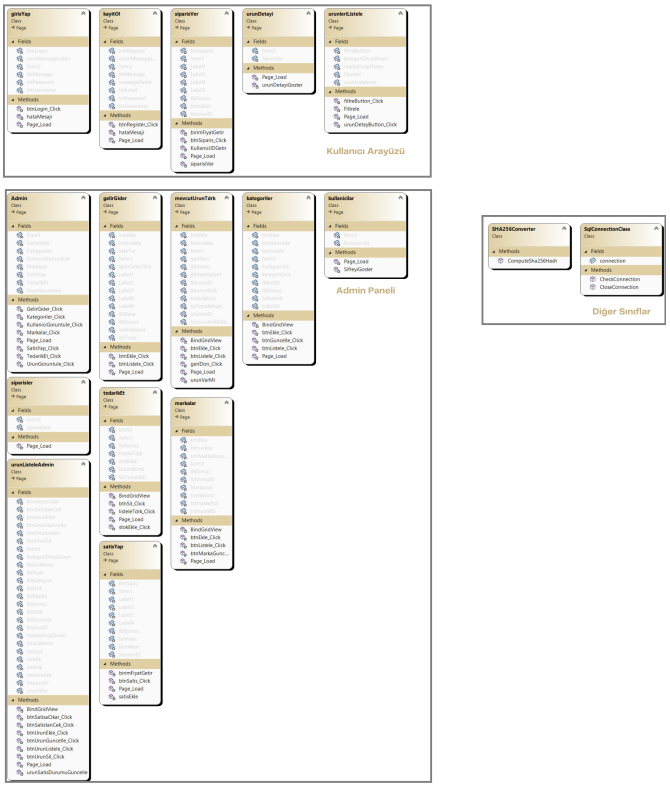

Fig. 2: **Yazılım Mimarisi:** Proje Sınıf Diyagramı (Visual Studio Community 2022 ile oluşturulmuştur.)

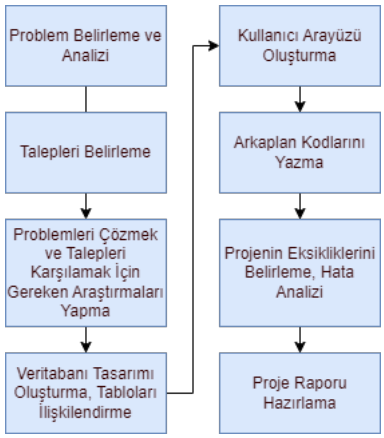

Fig. 3: **Yazılım Mimarisi:** Proje Geliştirme Aşamaları (draw.io ile oluşturulmuştur.)

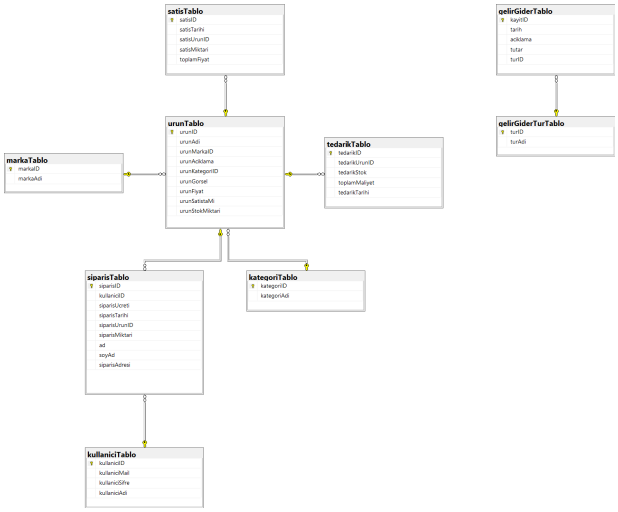

Fig. 4: **Veritabanı ER Diyagramı** (Microsoft SQL Server Management Studio ile oluşturulmuştur.)

III. Genel Yapı

- **Geliştirme ortamları:** Visual Studio Community 2022, Microsoft SQL Server Management Studio
- **Kullanılan programlama dilleri:** C#, JavaScript
- **Kullanılan işaretleme dilleri:** HTML, CSS
- **Kullanılan frameworkler:** ASP.NET, Bootstrap
- **Veritabanı yönetimi için:** SQL (Structured Query Language), Microsoft SQL Server Management Studio

Özet: Bu projede giyim mağazalarının günlük işlemlerini daha kolay ve verimli şekilde yönetebilmesi amaçlanarak web tabanlı bir otomasyon sistemi oluşturulmuştur. Projenin veritabanı 9 tablo ve bu tablolar arasında 7 ilişki içermektedir. INDEX, TRIGGER ve VIEW kullanılarak veritabanı zenginleştirilmiştir.

A. Admin ve Kullanıcı Arayüzleri

Admin arayüzünde: Mağazadaki ürünler listelenebilir, yeni ürün eklenebilir, bu ürünlerin bilgileri güncellenebilir, ürünlerin kullanıcı arayüzündeki görünürlüğü (satış durumu) değiştirilebilir.

Mağazadaki ürün satışlarını, tedarikleri ve müşteri siparişlerini listelemek, el ile yeni satış ya da tedarik girmek ve bu işlemlerin ardından mağazanın gelir gider durumunu takip etmek mümkündür. Aynı zamanda satış, tedarik veya müşteri siparişi sonrasında ürünlerin stok durumu gözlemlenebilir.

Admin, mağazada kayıtlı olan marka ve ürün kategorilerini görüntüleyebilir, güncelleyebilir ve mağazaya yeni marka veya ürün kategorisi ekleyebilir. Mağaza sistemine kayıt olmuş kullanıcıların bilgilerini listeleyebilir.

Kullanıcı arayüzünde: Kullanıcı sisteme kayıt olup giriş yapabilir.

Giriş yapmamış kullanıcı satışta olan ürün listesini görüntüleyebilir, istediği marka ve kategoriye göre filtrelenmiş sonuçları alabilir. İlgili ürünün detaylarını görüntüleyebilir.

Giriş yapan kullanıcı, giriş yapmamış kullanıcının yaptığı tüm işlemleri yapabilir. Ek olarak sipariş verebilir.

B. Alınan Güvenlik Önlemleri

Kullanıcının oluşturduğu şifre veritabanında güvenlik amaçlı şifrelenerek saklanır.

Admin arayüzüne yalnızca admin erişebilir. Kullanıcı erişimi engellenmiştir.

REFERANSLAR

[1] https://www.youtube.com/watch?v=7_ZdsXMUri8&list=PLD54hVH5EaMZCz7d9H5TFmXeqH8rTiTB8

[2] https://furkanalaybeg.medium.com/sqlde-tetikleyiciler-triggers-nedir-6bd50b72e56d

[3] https://www.w3schools.com/sql/sql_view.asp

[4] https://www.w3schools.com/bootstrap/bootstrap_navbar.asp

[5] https://learn.microsoft.com/en-us/previous-versions/aspnet/wtxbf3hh(v=vs.100)

[6] https://codepen.io/aaronbarnard/pen/oeWvJo

[7] https://getcssscan.com/css-buttons-examples

[8] https://www.afguven.com/asp-net-database-baglantisi-yapmak.html

[9] https://learn.microsoft.com/tr-tr/dotnet/api/system.data.sqlclient.sqlcommand?view=dotnet-plat-ext-8.0

[10] https://learn.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=net-8.0

[11] https://learn.microsoft.com/tr-tr/dotnet/api/system.web.ui.webcontrols.datalist?view=netframework-4.8.1

[12] https://stackoverflow.com/questions/4018114/read-data-from-sqldatareader

[13] https://docs.devexpress.com/AspNet/DevExpress.Web.ASPxGridView.AutoGenerateColumns

[14] https://learn.microsoft.com/en-us/aspnet/web-forms/overview/data-access/custom-formatting/using-templatefields-in-the-gridview-control-cs

[15] https://www.gencayyildiz.com/blog/asp-net-session-kullanimi/

[16] https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm