Yazılım Laboratuvarı 3. Proje Ödevi

1. Esra Kurt
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
esrakurt221@gmail.com

2. Rumeysa Çetinalp
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
rumeysacetinalp821@gmail.com

***Özetçe*—Bu rapor, Kocaeli Üniversitesi Yazılım Laboratuvarı dersinin 3. Proje ödevi için hazırlanmıştır.**

Anahtar Kelimeler — Web Uygulaması, Eş Zamanlı Sipariş, Stok Yönetimi, ASP.NET Core MVC, Veri tabanı, Sql Server

I. ÖZET

Bu proje ASP.NET Core MVC kullanılarak eş zamanlı sipariş yönetimi ve stok güncellemelerini gerçekleştiren bir sistem tasarlamak amacıyla yapılmıştır.

II. GİRİŞ

Multithreading ve senkronizasyon mekanizmaları kullanarak eş zamanlı sipariş yönetimi ve stok güncellemelerini gerçekleştiren bir sistem geliştirmeyi amaçlayan bu projede; sistem, aynı kaynağa eş zamanlı erişim problemlerini çözmek ve bu süreçte process, thread, mutex, semafor gibi kavramların kullanımını göstermek üzere tasarlanmıştır.

Temel özellikleri müşteri yönetimi: müşteriler random sayıda (5-10 arasında) atanır ve bütçeleri (500-3000 TL) rastgele belirlenir. Premium ve Standart olmak üzere iki müşteri türü bulunur. Premium müşteriler yüksek öncelikli işlemlere sahiptir. Admin İşlemleri: Admin ürün ekleme, silme ve stok güncellemelerini yönetir. Admin işlemleri ile müşteri işlemleri eş zamanlı olarak gerçekleştirilebilir. Stok Yönetimi: Sistem başlangıçta 5 farklı ürün ve sabit stok miktarları ile tanımlıdır. Ürün stoğu yetersizse işlem reddedilir. Stoklar anlık olarak güncellenir. Bütçe Yönetimi: Müşteri işlemleri için bütçe kontrolü sağlanır. Yetersiz bütçe durumunda işlem reddedilir. Dinamik Öncelik Sistemi: Bekleme süresi ve müşteri türüne göre işlem sırası dinamik olarak güncellenir. Premium müşteriler varsayılan olarak yüksek önceliğe sahiptir. Veritabanı Tasarımı: İlişkisel tablolar üzerinden müşteri, ürün ve sipariş bilgileri yönetilir. Loglama sistemi, işlemler hakkında detaylı kayıt tutar. Kullanıcı Arayüzü: Müşteri panelinde müşteri listesi, sipariş oluşturma formu ve bekleme durumu gibi özellikler bulunur. Ürün stok durumu grafiklerle görselleştirilir. Log paneli ile gerçek zamanlı işlemler izlenebilir.

III. YÖNTEM

İlk olarak hangi programlama dilini ve geliştirme ortamını kullanacağımıza karar verdik. ASP.NET Core MVC mimarisinde ve Mssql'de karar kıldık. Ardından Veri tabanı tablolarımızı oluşturduk ve aralarındaki ilişkileri kurduk. Daha sonra Veri tabanı ve IDE bağlantısını da gerçekleştirdikten sonra veri tabanı ile ilişkili olan "AppDbContext.cs" dosyamızı ve veri tabanındaki tabloların Models classlarını oluşturuduk. Projenin ilerleyen süreçlerinde ihtiyaç duydukça Models sayfalarını artırıp başka tablolar ekledik ve yine bu süreçte ihtiyaç duydukça gerekli Controller sayfaları ve Html sayfalarını açarak ilerledik.

Projemizdeki her sayfada kullanılan metotları açıklamak üzere adım adım ilerleyelim.

AppDbContext:

Stok Yönetimi adlı ASP.NET Core MVC projesinde, AdminController üzerinden stok yönetimi ve sipariş onaylama işlemlerini gerçekleştiren bir yapı geliştirdik. Öncelikle, AppDbContext adlı bir DbContext nesnesi ile

veritabanı erişimini sağladık ve farklı işlemler için ilgili servisleri ve loglama mekanizmasını ekledik.

Projede, ürün ekleme, güncelleme, silme ve listeleme gibi temel CRUD işlemlerini kapsayan çeşitli aksiyonlar tanımladık. Ürün ekleme sırasında resim dosyalarını wwwroot/uploads dizinine yükledik ve bu dosya yolunu veritabanında ürünle ilişkilendirdik. Sipariş onaylama işlemi için, müşterilerin öncelik sırasına göre siparişleri işleme aldık. Dinamik bir öncelik algoritması oluşturarak sipariş bekleme süresi ve müşteri türüne göre sıralama yaptık. Yetersiz stok durumunda sipariş reddedildi, aksi halde sipariş onaylandı ve stok güncellendi.

Stok güncellemelerinde ve sipariş işlemlerinde veri tutarlılığını sağlamak için SemaphoreSlim kullanarak eş zamanlı erişimi kontrol altına aldık. Ayrıca, müşteri harcamaları belirli bir limiti aştığında müşteri türünü dinamik olarak "Standard"dan "Premium"a yükselttik. Bu değişiklik sırasında da SemaphoreSlim kullanarak thread-safe işlemler gerçekleştirdik.

Loglama mekanizmasını entegre ederek, her bir işlemin başarılı ya da hatalı durumlarını sistem günlüğüne kaydettik ve adminin bu logları görebileceği bir görünüm hazırladık. Ayrıca, kilitlenmiş ürünler ve aktif siparişlerin durumlarını ayrı birer aksiyon üzerinden JSON formatında döndürerek dinamik bir frontend desteği sağladık.

Son olarak, tüm bu işlemleri modern yazılım geliştirme prensiplerine uygun bir şekilde modüler, güvenli ve performanslı hale getirmeye odaklandık. Bu sayede sistem, eş zamanlı erişim sorunlarını minimize ederek güvenilir bir stok ve sipariş yönetimi sağlıyor.

AdminController:

Stok Yönetimi projesindeki AdminController'ı detaylandırarak, ürün, sipariş ve müşteri yönetimi gibi temel işlemleri gerçekleştirdiğimiz bir kontrolcü sınıfını kapsıyor. Projemizde, ASP.NET Core MVC framework kullanarak kullanıcı dostu bir yönetim paneli oluşturduk. Bu kontrolcüde, ürünlerin eklenmesi, güncellenmesi, silinmesi ve listelenmesi gibi temel işlemleri yaptık. Ayrıca, siparişlerin dinamik öncelik hesaplaması ile sıralanmasını ve stok kontrollerinin gerçekleştirilmesini sağladık.

Ürün ekleme sırasında, görselleri wwwroot/uploads dizinine yükledik ve veritabanında görsel yollarını ilişkilendirdik. Güncelleme işlemlerinde ürünlere ait detayların yanı sıra, ürün görsellerini de değiştirme imkânı sunduk. Ürün silme işlemi, eş zamanlılık sorunlarını önlemek amacıyla SemaphoreSlim kullanılarak gerçekleştirildi. Tüm bu işlemlerde başarılı ve başarısız durumlar için loglama mekanizmasını devreye aldık.

Sipariş işlemlerinde, siparişlerin durumlarını "Beklemede", "Onaylandı" ve "Reddedildi" olarak güncelledik. Bekleyen siparişleri, müşterilerin türüne (Standard veya Premium) ve sipariş bekleme sürelerine göre dinamik olarak sıraladık. Stok yeterli ise siparişi onaylayıp, müşterinin bütçesini ve harcamalarını güncelledik; aksi durumda siparişi reddettik. Ayrıca, müşterilerin toplam harcaması belirli bir eşiği geçtiğinde, müşteri türünü "Standard"dan "Premium"a yükselttik.

Eş zamanlı erişim sorunlarını yönetmek için, ürün ve müşteri seviyesinde SemaphoreSlim nesneleri ile kilit mekanizmaları oluşturduk. Örneğin, bir ürün üzerinde güncelleme işlemi devam ederken başka bir adminin aynı ürüne erişimini engelledik. Ayrıca, kilitli ürünlerin 2 dakika sonra otomatik olarak serbest bırakılmasını sağladık.

Veritabanı işlemlerini test edebilmek için TestDb metodu ile bağlantıyı kontrol ettik. Admin kullanıcılarına, sistem loglarını görüntüleme ve JSON formatında sipariş verilerini alma imkânı sunduk. Sipariş sıralama ve stok kontrol işlemleri için gerekli verileri dinamik olarak düzenledik ve arka planda performansı artırmaya yönelik iyileştirmeler yaptık.

Sonuç olarak, bu kontrolcü sınıfı ile ürün, müşteri ve sipariş yönetimi süreçlerini modern, performanslı ve güvenli bir yapı ile geliştirdik. Aynı zamanda loglama ve eş zamanlılık yönetimini projeye entegre ederek, sistemi daha stabil ve ölçeklenebilir hale getirdik.

UserDashboardController:

Stok Yönetimi projesi için UserDashboardController adını verdiğimiz bir kullanıcı paneli kontrolcüsü geliştirdik. Bu kontrolcü, kullanıcıların ürünleri görüntülemesi, profillerini yönetmesi, alışveriş sepetlerini düzenlemesi ve sipariş işlemlerini

gerçekleştirmesi gibi temel işlevleri içeriyor. Ayrıca, kullanıcı dostu bir deneyim sağlamak için dinamik veri yönetimi ve loglama mekanizmalarını entegre ettik.

Dashboard (Ürün Listesi): Kullanıcıların tüm ürünleri görüntüleyebildiği bir kontrol noktası oluşturduk. Ürünler belirli kategorilere göre filtrelenebilirken, kilitlenmiş ürünler (örneğin başka bir admin tarafından düzenlenmekte olanlar) listeden hariç tutuluyor. Kullanıcının alışveriş sepeti de dinamik olarak hazırlanıyor ve kullanıcı paneline entegre ediliyor.

Profil Yönetimi: Kullanıcılar, profil bilgilerini (ad, soyad, e-posta) güncelleyebilir ve şifrelerini değiştirebilir. Tüm değişiklikler loglanıyor ve gerekli durumlarda hata mesajları ile kullanıcı bilgilendiriliyor. Ayrıca, kullanıcı bütçesinin belirli bir aralıkta (500 TL ile 3000 TL) güncellenmesine izin verdik.

Alışveriş Sepeti Yönetimi: Kullanıcıların alışveriş sepetlerine ürün ekleyebilmesi ve çıkarabilmesi için detaylı bir yapı oluşturduk. Sepete eklenen ürünler, maksimum 5 adet kuralıyla sınırlandırılıyor ve bu sınır aşıldığında kullanıcı bilgilendiriliyor. Sepetten çıkarma işlemlerinde loglama yapılarak detaylı kayıt tutuluyor.

Satın Alma İşlemi: Kullanıcılar, sepetteki ürünleri satın alırken bütçe kontrolü ve stok kontrolü yapılır. Yeterli bütçe varsa siparişler oluşturulup "Beklemede" statüsüyle kaydediliyor ve sepet temizleniyor. Kullanıcının siparişleri loglanarak ileride raporlama veya hata çözümleme işlemlerinde kullanılabiliyor.

Arama İşlevi: Kullanıcıların ürün adı veya kategorilere göre hızlıca arama yapabilmesini sağladık. Arama sonuçları dinamik olarak döndürülüyor.

Dinamik ve Güvenli Yapılar: Kullanıcı işlemlerinin daha güvenli hale getirilmesi için, tüm kritik işlemleri HttpContext.Session kullanarak oturum bazlı gerçekleştirdik. Ayrıca, dinamik loglama mekanizmasıyla her işlemin kaydını tutarak sistemin izlenebilirliğini artırdık. Sepet ve sipariş işlemleri sırasında hata durumları detaylı bir şekilde ele alındı ve kullanıcı deneyimi olumsuz etkilenmesin diye geri bildirimler sağlandı.

Sonuç olarak, bu kontrolcü ile kullanıcıların ürünleri kolayca görüntüleyebileceği, kişisel bilgilerini yönetebileceği ve alışveriş işlemlerini gerçekleştirebileceği bir kullanıcı paneli geliştirdik. Tüm bu süreçler boyunca güvenliğe, veri tutarlılığına ve kullanıcı memnuniyetine odaklandık.

HomeController:

Stok Yönetimi projesi için bir HomeController geliştirdik. Bu kontrolcü, kullanıcıların ve adminlerin giriş yapmasını, kayıt işlemlerini, müşteri verilerinin oluşturulmasını ve hata yönetimini kapsayan temel işlevleri sağlar. Ayrıca, kullanıcı deneyimini kolaylaştırmak ve güvenliği artırmak için modern şifre hashleme yöntemlerini ve oturum yönetimini entegre ettik.

Kullanıcılar, kayıt sırasında gerekli bilgileri doldurarak sisteme dahil olabilirler. Şifre alanı hashlenerek güvenli bir şekilde saklanır. Ayrıca, kullanıcıların bütçesi 500 TL ile 3000 TL arasında olmalıdır; bu aralık dışındaki değerler reddedilir. Yeni kayıt olan kullanıcılar için varsayılan olarak "Standard" müşteri tipi atanır ve diğer alanlara varsayılan değerler eklenir. Kayıt işlemi başarılı olduğunda, kullanıcı giriş yapması için yönlendirilir.

Kullanıcıların giriş yapması sırasında e-posta ve şifre doğrulaması gerçekleştirilir. Şifreler, hashlenmiş olarak saklandığı için doğrulama işlemi HashHelper yardımıyla yapılır. Başarılı girişlerde, oturum yönetimi ile kullanıcının kimliği HttpContext.Session içerisine kaydedilir. Giriş sırasında yanlış şifre veya e-posta girildiğinde kullanıcı bilgilendirilir. Admin giriş işlemi için ayrı bir yöntem tanımladık ve sadece admin hesapları üzerinde işlem yapılmasını sağladık.

Projenin başlatılmasında örnek müşteri verilerini otomatik olarak oluşturmayı kolaylaştırmak için bir işlem ekledik. Rastgele olarak "Premium" ve "Standard" müşteri hesapları oluşturuluyor. Bu sayede, sistemin test edilmesi ve tanıtımı için hazır veriler sağlanmış oluyor. Şifreler bu aşamada da hashleniyor ve güvenli bir şekilde saklanıyor.

Herhangi bir hata durumunda, kullanıcıya anlamlı bir hata mesajı iletilir ve hata detayları loglanır. Özellikle giriş, kayıt ve müşteri oluşturma işlemlerinde meydana gelen hatalar, ILogger kullanılarak detaylı bir şekilde kayıt altına alınır.

Güvenlik: Şifreler hashlenerek saklanır ve hiçbir zaman düz metin olarak tutulmaz.Oturum Yönetimi: Kullanıcı ve admin giriş işlemleri sırasında oturum bazlı kimlik doğrulama uygulanır.Hata Yönetimi ve Loglama: Her

işlem için anlamlı hata mesajları ve detaylı loglama yapılır.Veritabanı Tohumlama: Test ve geliştirme süreçlerini kolaylaştırmak için rastgele müşteri hesapları oluşturulabilir.Kullanıcı Dostu: Kayıt ve giriş işlemleri sırasında, eksik veya hatalı bilgiler kullanıcıya açık bir şekilde belirtilir.

Bu kontrolcü, sistemin giriş ve kayıt gibi temel fonksiyonlarını başarıyla yönetirken, kullanıcı deneyimini güvenlik ve verimlilikle birleştirir. Tüm işlemler sırasında loglama ve hata yönetimi mekanizmalarıyla sistemin stabilitesi ve izlenebilirliği artırılmıştır.

LogService:

Bu kodda, LogService adını verdiğimiz bir hizmet sınıfı geliştirdik. Bu sınıf, Stok Yönetimi projesi kapsamında sistemde gerçekleşen olayların loglanmasını sağlamak amacıyla oluşturulmuştur. Loglama işlemleri, veritabanında Logs tablosuna kayıt yapılarak sistemin izlenebilirliğini artırır. LogService, AppDbContext üzerinden veritabanına erişir ve log kayıtlarını güvenli bir şekilde eklemek için bir kilit mekanizması olarak Semaphore kullanır. Böylece, aynı anda birden fazla thread'in log yazma işlemi yapması engellenir ve veri tutarlılığı korunur.

Loglama işlemi, AddLog metodu aracılığıyla gerçekleştirilir. Bu metot, bir kullanıcı veya sistem olayıyla ilgili log detaylarını alır ve bir log nesnesi oluşturarak veritabanına kaydeder. Her bir log kaydında, müşteri ID'si, log türü (örneğin, "Bilgilendirme" veya "Hata"), log detayları ve isteğe bağlı olarak bir sipariş ID'si gibi bilgiler bulunur. Loglama işlemi sırasında, Semaphore ile kilit alınır ve işlem tamamlandıktan sonra kilit serbest bırakılır. Bu mekanizma, çoklu thread erişiminden kaynaklanabilecek veri tutarsızlıklarını önler ve işlemlerin güvenli bir şekilde gerçekleştirilmesini sağlar.

Bu yapı, sistemin her noktasında merkezi bir loglama çözümü sunar. Örneğin, bir sipariş oluşturma, kullanıcı kaydı veya hata durumunda ilgili detaylar veritabanına loglanabilir. Bu sayede, sistemde gerçekleşen tüm önemli olaylar kolayca izlenebilir ve hata ayıklama süreçleri daha verimli bir şekilde yürütülebilir. Loglama işlemleri performanslı, güvenilir ve bakım açısından kolay bir yapıda tasarlanmıştır.

HashHelper:

Bu kodda, HashHelper adını verdiğimiz bir sınıf oluşturarak projede kullanıcı şifrelerinin güvenli bir şekilde işlenmesini sağladık. Şifreleme işlemleri için güvenilir bir yöntem olan SHA256 algoritmasını kullandık. Amacımız, kullanıcı şifrelerini düz metin olarak saklamaktan kaçınmak ve bunun yerine hashlenmiş bir formatta güvenli bir şekilde saklamak. Böylece, veritabanına yetkisiz erişim durumunda bile kullanıcı şifreleri korunmuş oluyor.

HashPassword metoduyla, bir şifreyi alıp SHA256 algoritmasıyla hashledik ve çıkan sonucu Base64 formatında döndürdük. Bu sayede, her şifre işlemi sabit uzunlukta, insan tarafından okunamaz bir formata dönüştürülmüş oluyor. Öte yandan, VerifyPassword metodu ile, kullanıcının giriş sırasında girdiği şifreyi tekrar hashleyip, veritabanında saklanan hash ile karşılaştırdık. Bu karşılaştırma sayesinde, düz metin şifrelerle çalışmadan kullanıcı doğrulama işlemini güvenli bir şekilde gerçekleştirebildik.

Bu yapı sayesinde, kullanıcı şifrelerinin güvenliğini sağlamış olduk. Sistemimizde hiçbir zaman düz metin şifre saklamıyor ve hashleme işlemi sayesinde yetkisiz erişimler durumunda bile şifrelerin güvenliğini koruyoruz. Ayrıca, SHA256 algoritması modern şifreleme standartlarına uygun olduğu için sistemimizi güvenilir ve güçlü bir hale getirdik. Bu sayede kullanıcılarımızın verilerini korurken, şifre işlemlerini performanslı bir şekilde yönetebiliyoruz.

Program:

Bu kodda, Stok Yönetimi projesi için uygulama yapılandırmasını ve başlangıç işlemlerini gerçekleştiren bir Program.cs dosyası hazırladık. Projenin temel yapı taşlarını burada belirleyerek uygulamayı çalıştırmadan önce tüm gerekli servisleri ve ayarları sisteme entegre ettik.

Öncelikle, Dependency Injection (Bağımlılık Enjeksiyonu) kullanarak projede ihtiyacımız olan servisleri ekledik. AppDbContext, veritabanına bağlanmak için yapılandırıldı ve DefaultConnection adıyla tanımlanan bağlantı dizesi ile çalışacak şekilde ayarlandı. Loglama işlemleri için LogService sınıfını sisteme tanıttık. Ayrıca, oturum yönetimini etkinleştirerek kullanıcı oturumlarının 30 dakika

boyunca geçerli olmasını sağladık. Oturum çerezlerini güvenli ve erişimi kısıtlı olacak şekilde yapılandırdık.

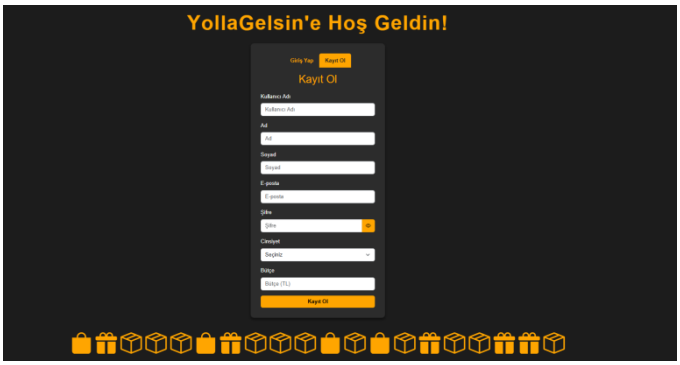
Modern web uygulamalarında güvenlik ve veri paylaşımı açısından kritik bir yapı olan CORS (Cross-Origin Resource Sharing) politikasını ekledik. "AllowAll" adıyla bir politika tanımlayarak herhangi bir kaynaktan gelen isteklerin tüm başlıklarına ve yöntemlerine izin verdik. Bu, farklı kaynaklardan yapılacak API çağrılarının sorunsuz çalışmasını sağladı.

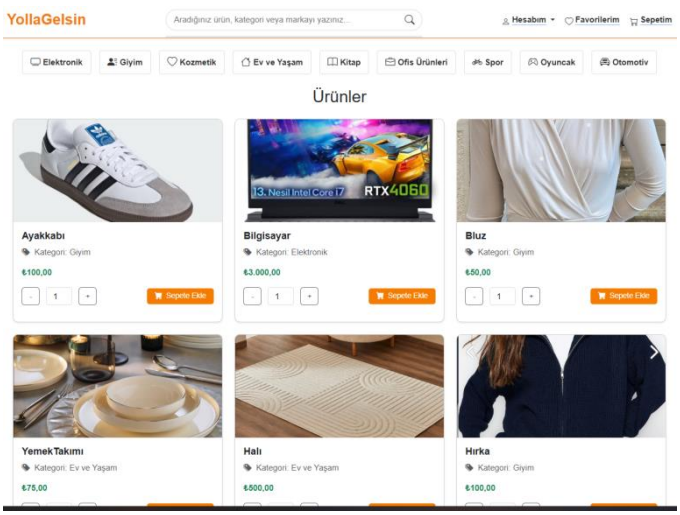
Uygulama başlatıldığında, veritabanını hazır hale getirmek ve örnek müşteri verileri oluşturmak amacıyla InitializeCustomers metodunu çağırdık. Bu işlem için HomeController sınıfını kullanarak, rastgele oluşturulan müşterilerin veritabanına eklenmesini sağladık. Bu, sistemin test edilmesi ve başlatılması için kullanışlı bir yöntemdir.

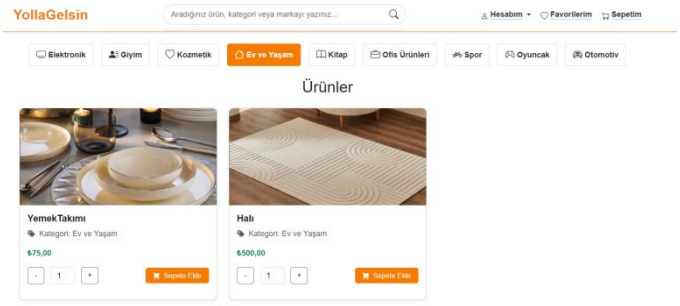
Son olarak, uygulamanın yönlendirme kurallarını belirledik ve varsayılan olarak Home kontrolcüsünün Login aksiyonuna yönlendirme yaptık. HTTPS yönlendirmesi ve statik dosya servislerini etkinleştirerek güvenli bir kullanıcı deneyimi sunduk. Uygulama, UseCors, UseSession, ve diğer middleware'lerle modern bir yapı sunacak şekilde düzenlendi.

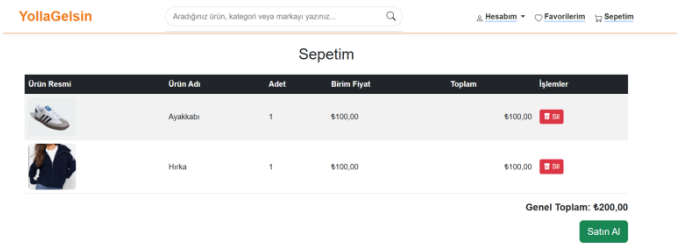
Bu yapılandırma, hem güvenli hem de esnek bir uygulama başlatma süreci oluştururken, modern web geliştirme standartlarına uygun bir temel sağladı. Proje, bu yapı sayesinde kolayca ölçeklenebilir, genişletilebilir ve yönetilebilir bir hale geldi.

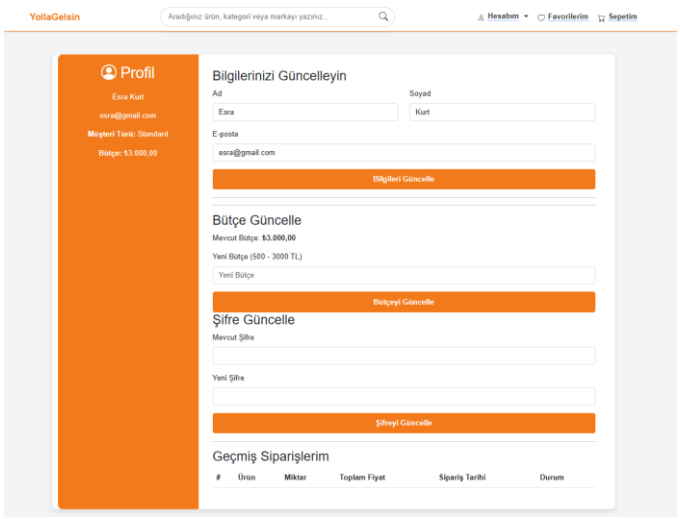
IV. DENEYSEL SONUÇLAR

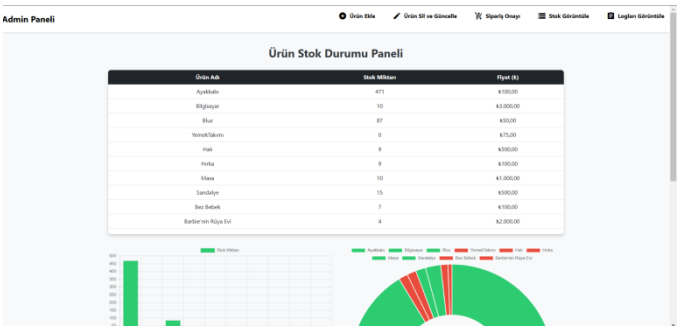


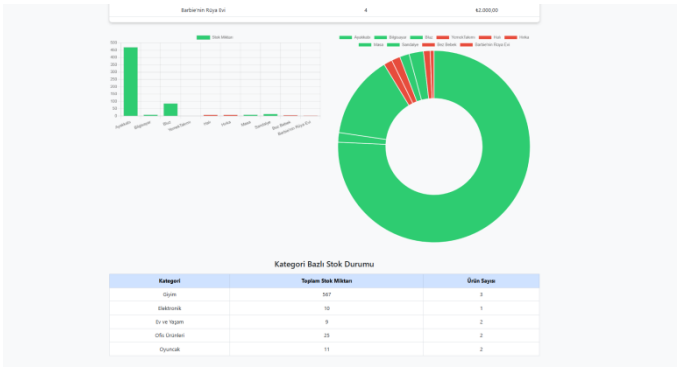


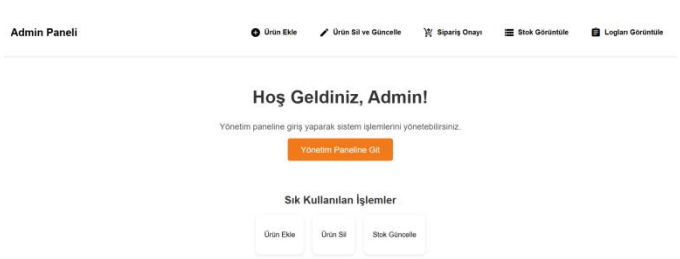


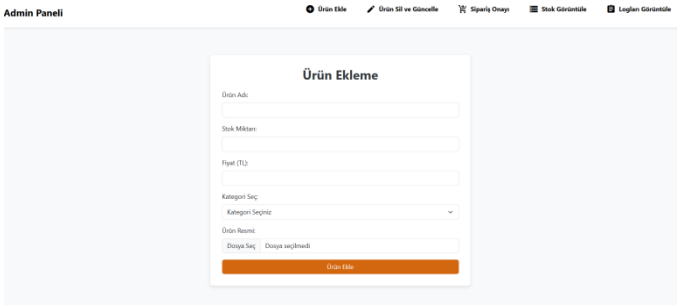



YollaGelsin
Hesabım
Favorilerim
Sepetim
Profil
Esra Kurt
esra@gmail.com
Müşteri Türü: Standard
Bütçe: ₺3.000,00
Bilgilerinizi Güncelleyin
Ad
Esra
Soyad
Kurt
E-posta
esra@gmail.com
Bilgileri Güncelle
Bütçe Güncelle
Mevcut Bütçe: ₺3.000,00
Yeni Bütçe (500 - 3000 TL)
Yeni Bütçe
Bütçeyi Güncelle
Şifre Güncelle
Mevcut Şifre
Yeni Şifre
Şifreyi Güncelle
Geçmiş Siparişlerim
Ürün Miktar Toplam Fiyat Sipariş Tarihi Durum

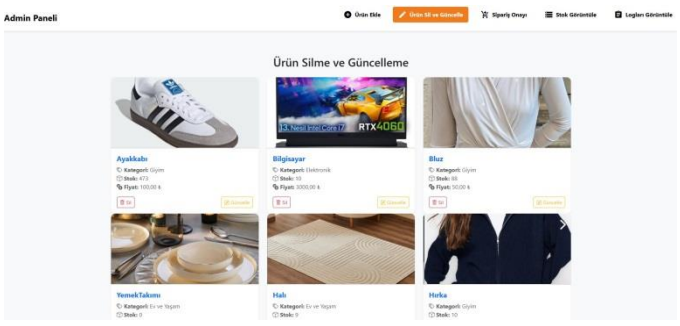

Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Ürün Stok Durumu Paneli
Ürün Adı
Stok Miktarı
Fiyat (₺)
Ayakkabı 471 ₺100,00
Bilgisayar 10 ₺3.000,00
Bluz 87 ₺50,00
YemekTakımı 0 ₺75,00
Halı 9 ₺500,00
Hırka 9 ₺100,00
Masa 10 ₺1.000,00
Sandalye 15 ₺500,00
Bez Bebek 7 ₺100,00
Barbie'nin Rüya Evi 4 ₺2.000,00

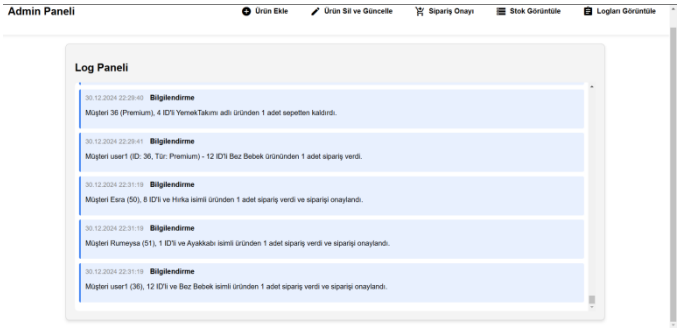

Kategori Bazlı Stok Durumu
Kategori
Toplam Stok Miktarı
Ürün Sayısı
Giyim 567 3
Elektronik 10 1
Ev ve Yaşam 9 2
Ofis Ürünleri 25 2
Oyuncak 11 2

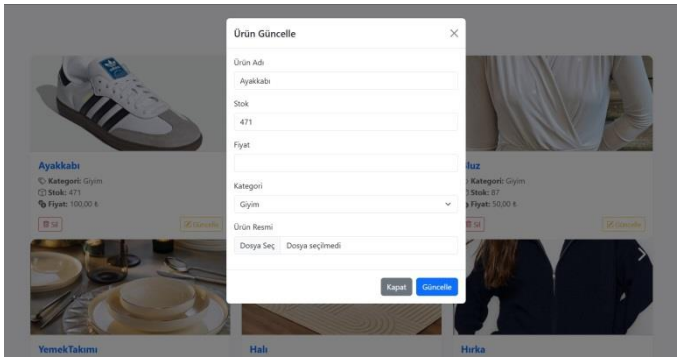

Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Hoş Geldiniz, Admin!
Yönetim paneline giriş yaparak sistem işlemlerini yönetebilirsiniz.
Yönetim Paneline Git
Sık Kullanılan İşlemler
Ürün Ekle
Ürün Sil
Stok Güncelle


Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Ürün Ekleme
Ürün Adı:
Stok Miktarı:
Fiyat (TL):
Kategori Seç:
Kategori Seçiniz
Ürün Resmi:
Dosya Seç
Dosya seçilmedi
Ürün Ekle


Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Bekleyen Siparişler
Müşteri Adı
Ürün Adı
Adet
Toplam Fiyat
Sipariş Tarihi
Bekleme Süresi (dk)
Müşteri Türü
Öncelik Skoru
Esra Hırka 1 100 30.12.2024 22:28 1 dk Normal 63.48
Rumeysa Ayakkabı 1 100 30.12.2024 22:28 1 dk Normal 56.00
user1 Bez Bebek 1 100 30.12.2024 22:29 0 dk Premium 31.41
Tümünü Onayla
Toplam Bekleyen Sipariş Sayısı: 3


Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Ürün Silme ve Güncelleme
Ayakkabı
Bilgisayar
Bluz
YemekTakımı
Halı
Hırka


Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Log Paneli
Bilgilendirme
Müşteri 36 (Premium), 4 ID'li YemekTakımı adlı üründen 1 adet sepetten kaldırdı.
Bilgilendirme
Müşteri user1 (ID: 36, Tür: Premium) - 12 ID'li Bez Bebek ürününden 1 adet sipariş verdi.
Bilgilendirme
Müşteri Esra (50), 8 ID'li ve Hırka isimli üründen 1 adet sipariş verdi ve siparişi onaylandı.
Bilgilendirme
Müşteri Rumeysa (51), 1 ID'li ve Ayakkabı isimli üründen 1 adet sipariş verdi ve siparişi onaylandı.
Bilgilendirme
Müşteri user1 (36), 12 ID'li ve Bez Bebek isimli üründen 1 adet sipariş verdi ve siparişi onaylandı.

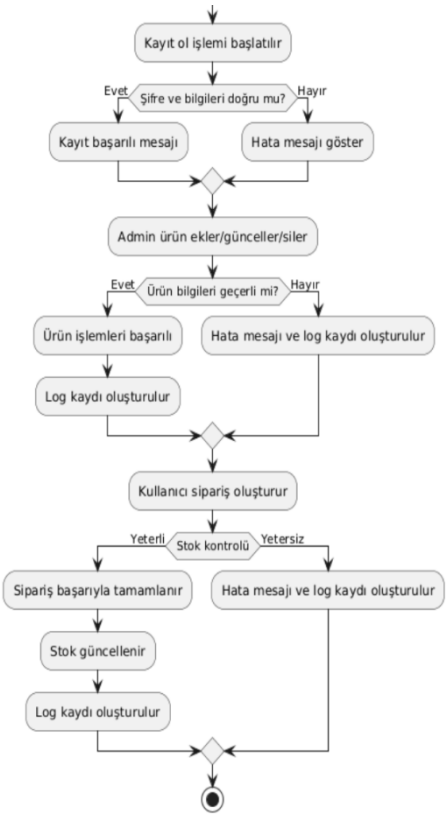

Ürün Güncelle
Ürün Adı
Ayakkabı
Stok
471
Fiyat
Kategori
Giyim
Ürün Resmi
Dosya Seç
Dosya seçilmedi
Kapat
Güncelle
Ayakkabı
YemekTakımı
Halı
Hırka


Admin Paneli
Ürün Ekle
Ürün Sil ve Güncelle
Sipariş Onayı
Stok Görüntüle
Logları Görüntüle
Log Paneli
Bilgilendirme
Müşteri 51 (Standard), 1 ID'li Ayakkabı adlı üründen 1 adet sepete ekledi.
Bilgilendirme
Müşteri Rumeysa (ID: 51, Tür: Standard) - 1 ID'li Ayakkabı ürününden 1 adet sipariş verdi.
Bilgilendirme
Müşteri 36 (Premium) bütçesini 3000 TL olarak güncelledi.
Bilgilendirme
Müşteri 36 (Premium), 4 ID'li YemekTakımı adlı üründen 1 adet sepete ekledi.
Hata
Müşteri user1 (ID: 36) - Ürün stoğu yetersiz: YemekTakımı (ID: 4) için istenen miktar: 1, mevcut stok: 0.

SONUÇ

Sonuç olarak bu Asp.Net Core Mvc ve Mssql kullanarak bir eticaret uygulaması yaptık. Böylece web sitesi uygulaması yapma becerilerimizi pekiştirdik ve veri tabanı ile ilgili işlemlere daha da hakim olduk. Thread, semafor, mutex gibi yapıları pratikte uygulama fırsatı elde ettik.

AKIŞ DİYAGRAMLARI



KAYNAKLAR

GitHub - Brktrlw/Stok-Muhasebe-Yazilimi: Tedarikçilerden ürün alış ve satışlara,müşteri ekleme satış oluşturma vb gibi bir çok özelliği ile ücretsiz çoğu kobinin kullanabileceği muhasabe yazılımı. (n.d.). GitHub. https://github.com/Brktrlw/Stok-Muhasebe-Yazilimi

Build software better, together. (n.d.). GitHub. https://github.com/topics/stock-tracking

Murat Çıplak. (2023, November 2). *Asp.Net Core Mvc 6.0 E-Ticaret Projesi* [Video]. YouTube. https://www.youtube.com/watch?v=kbwWyIyNNdM

Ali Zorlu. (2023, October 30). *7) SemaphoreSlim- .Net Parallel Programming* [Video]. YouTube. https://www.youtube.com/watch?v=CP5qjOXWwCk

ER DİYAGRAMI