Eş Zamanlı Sipariş Yönetimi ve Stok Güncelleme Sistemi

1st Eyüp Ensar TOĞUŞLU
Kocaeli University
Coumputer Engineer
Kocaeli, Turkey
eyupensar89544@gmail.com

2nd Mahir Enes AKPINAR
Kocaeli University
Coumputer Engineer
Kocaeli, Turkey
mahirennesakpinar@gmail.com

***Abstract*—Bu çalışma, eş zamanlı sipariş işleme ve stok güncellemelerini yönetmek için bir sistem sunmaktadır. Temel amaç, paylaşılan kaynaklara eş zamanlı erişim sorunlarını, çoklu iş parçacığı kullanımı ve senkronizasyon mekanizmalarıyla çözmektir. İş parçacıkları, mutex, semaforlar ve işlem önceliği gibi kavramlardan yararlanılarak, verimli ve çakışmasız operasyonlar sağlanmıştır. Sistem, C programlama dili ve SQL Server veritabanı kullanılarak geliştirilmiştir. Deneysel sonuçlar, önerilen sistemin eş zamanlı işlemleri etkin bir şekilde yönettiğini ve veri tutarlılığını sağladığını göstermektedir.**

I. Özet

Eş Zamanlı Sipariş ve Stok Yönetimi Uygulaması, Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü Yazılım Laboratuvarı I dersi kapsamında tasarlanmış bir yazılım projesidir. Bu proje, müşteri siparişlerini ve stok güncellemelerini dinamik ve eş zamanlı olarak ele alan yenilikçi bir sistem geliştirmeyi hedeflemektedir. Multithreading ve senkronizasyon mekanizmalarını etkin bir şekilde kullanarak, gerçek dünya senaryolarındaki benzer sorunları çözmek amacıyla yapılandırılmıştır.

Proje, yazılım mühendisliğinin hem teorik hem de pratik yönlerini birleştirerek öğrencilerin ileri düzeyde beceriler kazanmasını sağlamayı amaçlamaktadır. Modern yazılım geliştirme tekniklerinin uygulanmasını gerektiren proje, karmaşık sistemlerde eş zamanlılık ve veri tutarlılığı gibi kritik konuların anlaşılmasına olanak tanımaktadır. Özellikle aynı anda birden fazla kullanıcının sisteme erişim sağlayarak işlem yapabildiği senaryolarda, kaynaklara doğru şekilde erişim sağlamak büyük önem taşır. Bu bağlamda, C# dilinin güçlü nesne tabanlı programlama yetenekleri ve MSSQL veritabanının sağlam veri yönetim altyapısı, projenin başarısında kilit rol oynamaktadır.

Proje kapsamında ele alınan ana bileşenler aşağıdaki gibi sıralanabilir:

- **Müşteri Yönetimi:** Sistem, başlangıçta rastgele atanan müşteri bilgilerini yönetir. Her müşteri bir kimlik numarasına, bir bütçeye ve harcama bilgilerine sahiptir. Müşteriler Premium veya Standart olarak sınıflandırılmıştır. Premium müşteriler öncelikli işlem sırasına sahip olup, bu özellik dinamik bir şekilde güncellenmektedir.
- **Stok Yönetimi:** Sistem başlangıçta tanımlanmış ürün stoklarını yönetir. Her ürün için ürün adı, stok miktarı ve fiyat bilgileri bulunmaktadır. Stok miktarları müşterilerin satın alma işlemlerine bağlı olarak güncellenmektedir. Ayrıca, sistem, stok yetersizliği durumlarında ilgili işlemleri iptal ederek veri tutarlılığını sağlamaktadır.
- **Dinamik Önceliklendirme:** Sistem, müşteri türüne ve bekleme sürelerine göre dinamik bir önceliklendirme mekanizması uygular. Örneğin, Premium müşteriler varsayılan olarak daha yüksek bir önceliğe sahipken, Standart müşterilerin bekleme sürelerine dayalı olarak öncelikleri artmaktadır. Dinamik önceliklendirme algoritması, müşteri türüne göre farklı bir temel öncelik skoru atar ve müşterinin bekleme süresine bağlı olarak bu skoru dinamik şekilde günceller. Aşağıdaki örnek, bu mekanizmanın nasıl çalıştığını açıklamaktadır:
  - **Örnek 1:** Bir Premium müşteri, sipariş butonuna bastıktan sonra 5 saniye beklemişse ve bekleme süresi ağırlığı 0.5 olarak belirlenmişse, bu müşterinin toplam öncelik skoru şu şekilde hesaplanır:
    $$\text{Öncelik Skoru} = 15 + (5 \times 0.5) = 17.5$$
  - **Örnek 2:** Bir Standart müşteri, sipariş butonuna bastıktan sonra 10 saniye beklemişse, bu müşterinin öncelik skoru şu şekilde hesaplanır:
    $$\text{Öncelik Skoru} = 10 + (10 \times 0.5) = 15$$
  - **Kritik Durum:** Eğer bir Standart müşteri uzun süre beklemiş ve öncelik skoru Premium müşterilerden daha yüksek hale gelmişse, sistem otomatik olarak bu müşteriye öncelik tanır.
- **Loglama ve İzleme:** Tüm sistem işlemleri ayrıntılı bir şekilde loglanmaktadır. Log kayıtları, işlem türü, müşteri bilgileri, zaman damgaları ve işlem sonuçlarını içermektedir. Bu özellik, sistemin geçmiş performansını izlemek ve potansiyel sorunları tespit etmek için önemlidir.
- **Kullanıcı Arayüzü (UI):** Kullanıcı dostu bir arayüz, müşterilerin ve admin kullanıcılarının işlemlerini kolayca gerçekleştirmesine olanak tanır. Stok durumları, sipariş sıralamaları ve işlem sonuçları gibi bilgiler, görsel grafiklerle ve tablolarda sunulmaktadır.

Bu projenin geliştirilmesi sırasında karşılaşılan en önemli zorluklardan biri, eş zamanlı erişim sorunlarının çözülmesidir. Multithreading kullanılarak, aynı anda birden fazla işlemin sorunsuz bir şekilde gerçekleştirilmesi sağlanmıştır. Mutex ve semafor gibi senkronizasyon araçları, kaynakların doğru sırayla kullanılmasını garanti altına almıştır. Ayrıca, dinamik öncelik hesaplama algoritması sayesinde müşterilerin bekleme süreleri minimize edilmiştir.

II. Giriş

Hızlı tempolu iş ortamlarında etkin ve güvenilir sipariş ve stok yönetimi kritik öneme sahiptir. Paylaşılan kaynaklara eş zamanlı erişim, doğru bir şekilde yönetilmediğinde veri çakışmalarına veya tutarsızlıklara neden olabilir. Özellikle e-ticaret gibi sektörlerde, anlık olarak değişen müşteri taleplerine uyum sağlamak hayati önem taşır. Geleneksel yöntemler bu tür gereksinimleri karşılamakta yetersiz kalabilir. Bu bağlamda, eş zamanlı erişim sorunlarını çözmek ve operasyonel verimliliği artırmak için yenilikçi bir yaklaşım gereklidir.

Bu çalışmada, masaüstü uygulamaların sağladığı avantajlar dikkate alınmıştır. Web tabanlı sistemlere kıyasla masaüstü uygulamalar, daha hızlı yanıt süresi, çevrimdışı çalışma imkanı ve yerel kaynakların daha etkin kullanımı gibi avantajlar sunar. Masaüstü ortamı, özellikle yüksek performans gerektiren işlemler ve yoğun veritabanı işlemleri için idealdir. Ayrıca, C programlama dili bu bağlamda güçlü bir seçenek olarak öne çıkmaktadır. .NET ekosistemiyle sağlanan geniş kütüphane desteği, zengin kullanıcı arayüzü oluşturma imkanı ve yüksek performanslı uygulamalar geliştirme yeteneği, C'ı tercih edilen bir dil haline getirmiştir.

Bu projede kullanılan sistem, hem müşteri hem de admin işlemlerinin eş zamanlı olarak yürütülmesini sağlamak üzere tasarlanmıştır. Özellikle, birden fazla müşterinin aynı anda aynı ürünü satın almak istemesi durumunda çakışmaları önleyecek bir senkronizasyon mekanizması geliştirilmiştir. Premium müşterilere öncelik tanınması, dinamik öncelik hesaplama algoritması ile desteklenmiştir. Bu algoritma, bekleme sürelerini ve müşteri türlerini dikkate alarak öncelikleri otomatik olarak belirlemektedir.

Sistemde kullanılan veritabanı, ilişkisel bir model üzerine kurulmuş olup müşteriler, ürünler, siparişler ve loglar için ayrı tablolar barındırmaktadır. Bu yapı, tüm işlemlerin hızlı ve güvenilir bir şekilde gerçekleştirilmesini sağlamaktadır. Ayrıca, sistemdeki işlemlerin izlenebilirliği, gerçek zamanlı loglama mekanizması ile mümkün kılınmıştır. Örneğin, müşterilerin satın alma işlemleri sırasında karşılaşabilecekleri hata mesajları veya başarılı işlemler loglanarak kayıt altına alınmaktadır. Bu loglar, sistem performansını değerlendirmek ve potansiyel sorunları hızlı bir şekilde çözmek için kullanılmaktadır.

Projenin bir diğer önemli özelliği ise dinamik stok yönetimi sistemidir. Admin, stokları güncelleyebilir veya yeni ürünler ekleyebilirken, müşteriler aynı anda bu stok bilgilerine erişebilir. Stokların güncel tutulması, çakışmaları önlemek için mutex mekanizmaları ile sağlanmıştır. Örneğin, bir müşteri satın alma işlemi sırasında ürün stoklarının yetersiz olması durumunda işlem reddedilmekte ve bu durum log kaydı olarak tutulmaktadır.

Sonuç olarak, bu çalışma, eş zamanlı erişim problemlerini çözmek ve operasyonel verimliliği artırmak için çoklu iş parçacığı ve senkronizasyon mekanizmalarını bir araya getiren kapsamlı bir çözüm sunmaktadır. Geliştirilen sistem, kullanıcı dostu bir arayüz ve esnek bir mimari ile desteklenmiş olup, hem bireysel hem de kurumsal kullanımlar için uygundur.

III. Yöntem

A. Sistem Mimarisi

Sistem, tüm süreçlerin koordinasyonunu sağlamak üzere üç ana modülden oluşmaktadır:

- **Sipariş Yönetim Modülü**: Kullanıcı siparişlerini alır, önceliklendirir ve onaylar. Her bir sipariş, müşteri türü ve siparişin sisteme giriş zamanı gibi kriterlere göre sıralanır.
- **Stok Yönetim Modülü**: Stok güncellemelerini takip eder ve çakışmaları önlemek için senkronizasyon mekanizmaları kullanır. Bu modül, mevcut stok bilgilerini düzenli olarak günceller ve eksik stoklar için uyarılar sağlar.
- **Kullanıcı Yönetim Modülü**: Kullanıcı hesaplarının ve yetkilerinin yönetimini gerçekleştirir. Ayrıca, sistem üzerindeki tüm işlemlerin kayıt altına alınmasını sağlar.

B. Veritabanı Yapısı

Veritabanı, ilişkisel bir model üzerine inşa edilmiştir ve aşağıdaki tablolar ve sütunları içermektedir:

- **Customers**:
  - **CustomerID**: Benzersiz müşteri kimliği.
  - **CustomerName**: Müşteri adı.
  - **Budget**: Müşteri bütçesi.
  - **CustomerType**: Müşteri türü (örneğin, Premium veya Standart).
  - **TotalSpent**: Müşterinin toplam harcaması.
- **Products**:
  - **ProductID**: Benzersiz ürün kimliği.
  - **ProductName**: Ürün adı.
  - **Stock**: Mevcut stok miktarı.
  - **Price**: Ürün fiyatı.
- **Orders**:
  - **OrderID**: Benzersiz sipariş kimliği.
  - **CustomerID**: Siparişi veren müşteri kimliği.
  - **ProductID**: Sipariş edilen ürün kimliği.
  - **Quantity**: Sipariş edilen ürün miktarı.
  - **TotalPrice**: Siparişin toplam fiyatı.
  - **OrderDate**: Siparişin verildiği tarih.
  - **OrderStatus**: Sipariş durumu (örneğin, Bekliyor, Onaylandı, İptal).
- **Logs**:
  - **LogID**: Benzersiz log kaydı kimliği.
  - **CustomerID**: Log ile ilişkili müşteri kimliği.
  - **OrderID**: Log ile ilişkili sipariş kimliği.
  - **LogDate**: Log kaydının oluşturulduğu tarih.
  - **LogType**: Log türü (örneğin, Bilgi, Hata).
  - **LogDetails**: Log detayları.

C. Algoritmalar ve Teknikler

Loglama Mekanizması: Sistem, her işlem için detaylı bir loglama mekanizması sağlar. Bu mekanizma, sistemin güvenilirliğini artırır ve hata tespiti için kullanılır. Loglama süreci şu şekilde işler:

- **Log Oluşturma:**
  - Sipariş oluşturma, güncelleme ve stok işlemleri sırasında loglar otomatik olarak kaydedilir.
  - Kayıtlar, işlem türüne (örneğin, Bilgi, Hata) ve detayına göre kategorize edilir.
- **Log Kaydı:**
  - Tüm loglar **Logs** tablosunda saklanır. Bu tablo, işlem tarihini (LogDate), türünü (LogType), müşteri ve sipariş kimliklerini (CustomerID, OrderID) ve işlem detaylarını (LogDetails) içerir.

D. Örnek Loglama Kod Parçası

```
def log_ekle(customer_id, order_id,
    log_turu, log_detaylari):
    log_id = benzersiz_id_olustur()
    log_date = mevcut_tarih()
    veritabanına_ekle(log_id, customer_id
        , order_id, log_date, log_turu,
        log_detaylari)
```

Listing 1. Loglama Fonksiyonu

- **Log Analizi:**
  - Loglar düzenli olarak analiz edilerek sistem performansı hakkında bilgi elde edilir.
  - Hata logları incelenerek sistemdeki potansiyel sorunlar belirlenir.

```
function oncelik_hesapla(musteri_turu,
    siparis_yasi):
    temel_puan = 20 eğer musteri_turu == '
        Premium' ise aksi halde 10
    oncelik_puani = temel_puan + (siparis_yasi
         * 0.5)
    return oncelik_puani
```

Listing 2. Öncelik Hesaplama Algoritması

Mutex Kullanarak Senkronizasyon:

```
mutex.kilit();
try {
    stok_guncelle();
} finally {
    mutex.serbest_birak();
}
```

Listing 3. Mutex Kullanımı

E. UML Diyagramı

Sistemin genel işleyişini ve bileşenler arasındaki ilişkileri gösteren UML diyagramı aşağıda verilmiştir.

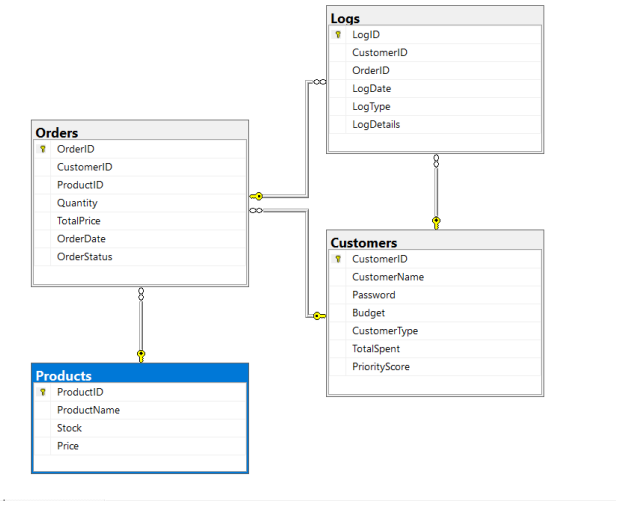

Fig. 1. Sistem UML Diyagramı

IV. Deneysel Sonuçlar

Deneysel sonuçlar kapsamında sistemin farklı panelleri incelenmiş ve aşağıda detaylı olarak sunulmuştur:

A. Giriş Paneli

Kullanıcıların sisteme giriş yaptığı alan. Burada kullanıcı adı ve şifre girişinden sonra yetkilere göre yönlendirme yapılır.

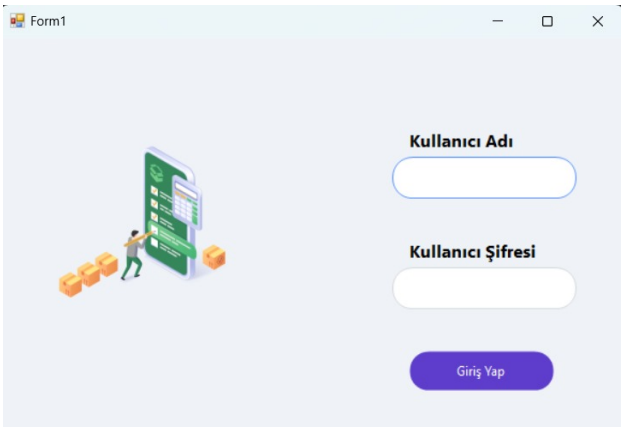

Fig. 2. Giriş Paneli

B. Kullanıcı Paneli

Ürünlerin listelendiği ve sepete eklenebildiği ana panel. Ayrıca, toplam tutar ve stok bilgileri dinamik olarak gösterilir.

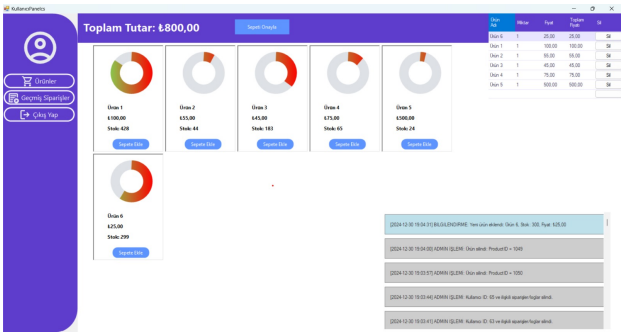

Fig. 3. Kullanıcı Paneli

C. Kullanıcı Geçmiş Siparişler Paneli

Kullanıcıların geçmiş siparişlerini tarih ve durum bazında görebildiği ekran.

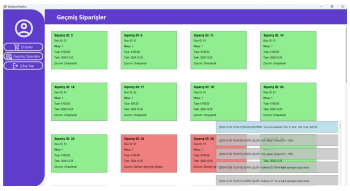

Fig. 4. Kullanıcı Geçmiş Siparişler Paneli

D. Admin Sipariş Onaylama Paneli

Yönetici yetkisine sahip kullanıcıların siparişleri onaylayabildiği ve durumlarını takip edebildiği ekran.

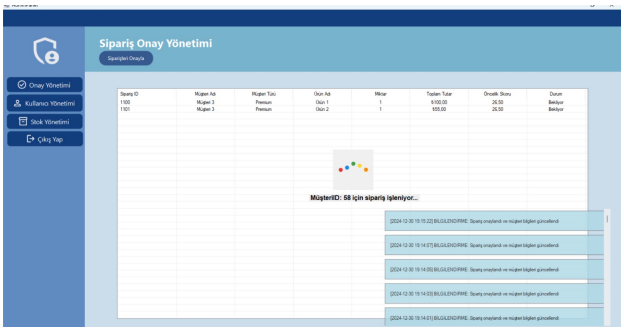

Fig. 5. Admin Sipariş Onaylama Paneli

E. Admin Kullanıcı Yönetimi Paneli

Kullanıcı hesaplarının listelendiği ve düzenlenebildiği ekran. Burada kullanıcı türü (Premium veya Standart) de belirtilir.

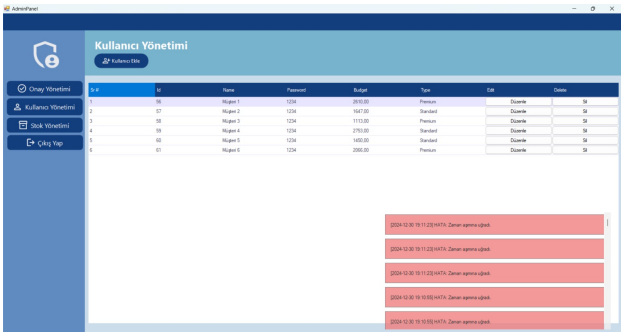

Fig. 6. Admin Kullanıcı Yönetimi Paneli

F. Admin Stok Yönetim Paneli

Stok bilgilerinin güncellenebildiği, ürünlerin eklenip çıkarılabildiği ekran.

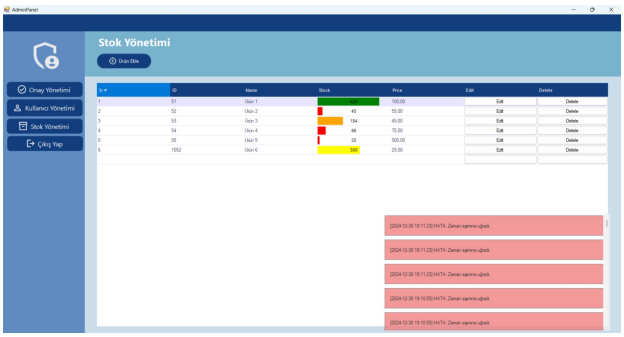

Fig. 7. Admin Stok Yönetim Paneli

G. Ürün Ekleme Paneli

Yeni ürünlerin isim, stok ve fiyat bilgileriyle sisteme eklenebildiği basit bir form.

Fig. 8. Ürün Ekleme Paneli

V. SONUÇ

Bu çalışmada, eş zamanlı sipariş işleme ve stok yönetimi için çoklu iş parçacığı ve senkronizasyon mekanizmalarını kullanan bir sistem geliştirilmiştir. Sistem, hem müşteri hem de yönetici işlemlerini optimize eden kullanıcı dostu bir platform sunmaktadır.

Geliştirilen sistemin temel özellikleri arasında:

- Dinamik önceliklendirme algoritmaları ile müşteri işlemlerinin sıralanması,
- Gerçek zamanlı stok yönetimi ve eş zamanlı erişim sorunlarını önlemek için mutex tabanlı mekanizmaların kullanılması,
- Kullanıcı ve yönetici panelleri aracılığıyla sezgisel bir arayüz,
- Veritabanı ile uyumlu işlem izleme ve loglama mekanizmaları bulunmaktadır.

Deneysel sonuçlar, sistemin eş zamanlı işlem yükünü başarılı bir şekilde yönettiğini ve veri tutarlılığını koruduğunu göstermiştir. Gelecekte, bu sistemin daha geniş ölçekli veri setleri ile test edilmesi ve makine öğrenimi algoritmaları ile işlem önceliklendirme yeteneklerinin artırılması planlanmaktadır.

REFERENCES

[1] https://www.youtube.com/watch?v=baGYmzqGnzM
[2] https://www.youtube.com/watch?v=J4RQFtiA4a4t=618s
[3] https://www.geeksforgeeks.org/mutex-vs-semaphore/
[4] https://stackoverflow.com/questions/34524/what-is-a-mutex