AKILLI KARGO VE SİPARİŞ YÖNETİM SİSTEMİ

PROJE RAPORU

I.Giriş

Bu proje ürün, müşteri, sipariş ve kargo süreçlerini yöneten, modüler, genişletilebilir ve bakımı kolay bir e-ticaret platformunun geliştirilmesini kapsamaktadır. Sistem, kullanıcıların ürün sipariş edebileceği, uygun kargo yöntemini seçebileceği ve teslimat sürecini takip edebileceği bir yapı sunar. Yönetici, envanter yönetimi, ürün ekleme ve stok yenileme işlemlerini gerçekleştirebilir. Sistem, Python kullanılarak geliştirilmiş olup kapsülleme, kalıtım ve çok biçimlilik gibi OOP ilkelerini temel alır. Kullanıcı etkileşimi için bir komut satırı arayüzü (CLI) sağlanmış, veri kalıcılığı ise JSON dosya depolaması ile gerçekleştirilmiştir.

Sistemin Yapabilecekleri

E-ticaret platformu aşağıdaki temel işlevleri sunar:

Ürün Yönetimi: Yeni ürün ekleme, listeleme, kategori, fiyat veya stok durumuna göre filtreleme ve detaylı stok takibi.

Müşteri Yönetimi: Müşteri kaydı, kimlik doğrulama ve sipariş geçmişi görüntüleme.

Sipariş İşlemleri: Sipariş oluşturma (standart veya ekspres), durum güncelleme ve iptal etme, otomatik stok güncellemeleriyle birlikte.

Kargo Seçenekleri: Hızlı, ekonomik ve drone kargolama yöntemleri; ağırlık, bölge veya öncelik (maliyet veya hız) bazında otomatik seçim.

Bildirim Sistemi: Sipariş durumu değiştiğinde müşterilere e-posta ve SMS bildirimleri.

Stok Kontrolü: Stok seviyelerini yönetme, düşük stok uyarıları ve işlem günlüğü tutma.

Ek Özellikler: Kullanıcı giriş/çıkış, sipariş notları, takip numarası oluşturma ve teslimat süresi tahmini.

II.Gelişme

Kullanılan OOP İlkeleri

Kapsülleme

Kapsülleme, veri ve davranışların bir sınıf içinde birleştirilmesi ve dahili detayların dış dünyadan gizlenmesi anlamına gelir. Bu, veri güvenliğini artırır ve sınıfın iç işleyişini korur. Sistemde kapsülleme şu şekilde uygulanmıştır:

Customer sınıfında, _password_hash özelliği özel (private) olarak tanımlanmış ve yalnızca authenticate yöntemi aracılığıyla erişilebilir. Bu, şifrenin doğrudan manipüle edilmesini önler.

Uygulanması: InventoryManager sınıfında, _instance, _lock ve _file_lock gibi dahili durum değişkenleri özel olarak işaretlenerek yalnızca sınıfın yöntemleri aracılığıyla erişilir. Bu, envanter yönetiminin tutarlılığını sağlar.

Avantajı: Kapsülleme, veri bütünlüğünü korur ve istem dışı değişiklikleri engeller, böylece sistem güvenilirliğini artırır.

Kalıtım

Kalıtım, bir sınıfın başka bir sınıftan özellik ve davranışları miras almasını sağlar, böylece kod tekrarını azaltır ve hiyerarşik bir yapı oluşturur. Sistemde kalıtım, özellikle tasarım desenlerinde yoğun bir şekilde kullanılmıştır:

ShippingMethod soyut temel sınıfından türeyen FastShipping, CheapShipping ve DroneShipping sınıfları, ortak bir arayüzü miras alır ve her biri kendi kargo mantığını uygular.

Uygulanması: OrderDecorator soyut sınıfından türeyen BaseOrder, FragileShipping ve InsuredShipping sınıfları, siparişlere ek hizmetler eklemek için kalıtımı kullanır. Bu, temel sipariş işlevselliğini korurken ek özellikler eklemeyi sağlar.

Avantajı: Kalıtım, kod tekrarını önler ve sınıflar arasında ortak davranışları paylaşarak bakım kolaylığı sağlar.

Çok Biçimlilik

Çok biçimlilik, farklı sınıfların aynı arayüzü farklı şekillerde uygulayabilmesi anlamına gelir. Sistemde çok biçimlilik, özellikle Strateji ve Dekoratör desenlerinde belirgindir:

ShippingMethod arayüzünü uygulayan sınıflar (FastShipping, CheapShipping, DroneShipping), calculate_cost ve estimate_delivery_days yöntemlerini farklı mantıklarla uygular. Bu, sistemin kargo yöntemlerini dinamik olarak değiştirmesine olanak tanır.

Uygulanması: Order.update_status yöntemi, sipariş durumu değiştiğinde farklı bildirim türlerini (EmailNotification, SMSNotification) aynı update arayüzü üzerinden çağırır.

Avantajı: Çok biçimlilik, sistemin esnekliğini artırır ve farklı davranışların tek bir arayüz üzerinden yönetilmesini sağlar.

Soyutlama

Soyutlama, yalnızca gerekli detayların gösterilmesi ve karmaşıklığın gizlenmesi anlamına gelir. Sistem, soyut sınıflar ve arayüzler aracılığıyla soyutlamayı yoğun bir şekilde kullanır:

ShippingMethod ve OrderDecorator gibi soyut temel sınıflar, somut sınıfların uyması gereken bir arayüz sağlar. Bu, kargo yöntemleri ve sipariş dekoratörlerinin tutarlı bir şekilde uygulanmasını garanti eder.

Uygulanması: NotificationObserver soyut sınıfı, bildirim kanallarının ortak bir update yöntemine sahip olmasını sağlar, böylece farklı bildirim türleri aynı arayüzü kullanır.

Avantajı: Soyutlama, kodun okunabilirliğini ve bakımını kolaylaştırır, çünkü geliştiriciler yalnızca arayüzle ilgilenir ve dahili detayları bilmek zorunda kalmaz.

Kullanılan Tasarım Desenleri

Proje, yapısını ve sürdürülebilirliğini artırmak için çeşitli yazılım tasarım desenlerini kullanır.

Strategy Deseni

Strateji deseni, farklı kargo yöntemlerini (Hızlı Kargolama, Ekonomik Kargolama, Drone Kargolama) dinamik olarak seçmek için kullanılır. Bu desen, sipariş ağırlığı, bölge ve kullanıcı tercihlerine (maliyet

veya hız önceliği) göre uygun kargo yöntemini belirler.

Uygulanması:

ShippingMethod soyut temel sınıfı, kargo stratejileri için calculate_cost, estimate_delivery_days ve is_available yöntemlerini tanımlar.

Somut sınıflar (FastShipping, CheapShipping, DroneShipping) bu yöntemleri özel mantıkla uygular:

- FastShipping: Daha yüksek maliyet, hızlı teslimat (2-3 gün), ağırlık ≤ 20 kg için uygundur.
- CheapShipping: Düşük maliyet, yavaş teslimat (7-10 gün), tüm ağırlıklar için uygundur.
- DroneShipping: Sabit maliyet, en hızlı teslimat (1-2 gün), ağırlık ≤ 5 kg ve yerel bölgeler için uygundur.

ECommerceSystem.select_shipping_method, önceliğe (maliyet veya hız) göre en uygun kargo yöntemini seçer.

Avantajı: Bu desen, yeni kargo yöntemlerinin mevcut kodu değiştirmeden kolayca eklenebilmesini sağlar ve Açık/Kapalı İlkesine uygunluk sunar.

Observer Deseni

Gözlemci deseni, sipariş durumu değişikliklerinde müşterilere birden fazla kanal (e-posta ve SMS) üzerinden bildirim göndermek için kullanılır.

Uygulanması:

NotificationService sınıfı, NotificationObserver nesnelerinin bir listesini tutar (EmailNotification, SMSNotification).

Sipariş durumu değiştiğinde (Order.update_status), NotificationService.notify yöntemi, tüm kayıtlı gözlemcilere biçimlendirilmiş mesajlar gönderir.

Acil bildirimler (örn. Yolda, Dağıtımda, Teslim Edildi durumları) için mesajlar "[URGENT]" öneki ile vurgulanır.

Avantajı: Bu desen, bildirim mantığını sipariş işleme sürecinden ayırarak, yeni bildirim kanallarının (örn. push bildirimleri) kolayca eklenebilmesini sağlar.

Factory Method Deseni

Fabrika Yöntemi deseni, farklı sipariş türlerini (standart veya ekspres) ve ilgili kargo yapılandırmalarını oluşturmak için kullanılır.

Uygulanması:

OrderFactory soyut temel sınıfı, create_order yöntemini tanımlar.

Somut fabrika sınıfları (StandardOrderFactory, ExpressOrderFactory) sipariş oluşturma mantığını işler:

- StandardOrderFactory: Kullanıcı tarafından seçilen veya varsayılan kargo yöntemleriyle sipariş oluşturur.
- ExpressOrderFactory: Ağırlık ve bölge kısıtlamalarına göre en hızlı kargo yöntemini (Drone veya Hızlı) otomatik olarak atar.

ECommerceSystem.create_order, kullanıcı tarafından belirtilen sipariş türüne göre uygun fabrikayı kullanır.

Avantajı: Bu desen, sipariş oluşturma mantığını kapsüller ve yeni sipariş türlerinin (örn. abonelik tabanlı siparişler) kolayca eklenmesini sağlar.

Singleton Deseni

Tekil desen, envanter ve stok yönetimini merkezi bir şekilde kontrol etmek için InventoryManager sınıfının tek bir örneğinin olmasını sağlar.

Uygulanması:

InventoryManager sınıfı, eşzamanlı oluşturmayı önlemek için _lock ile iplik güvenli bir tekil uygulama kullanır.

Ürün envanterini, stok seviyelerini, düşük stok uyarılarını ve işlem günlüklerini yönetir; veriler JSON dosyasına kaydedilir.

_load_from_json ve _save_to_json veri kalıcılığını, update_stock ve restock_product ise doğru stok takibini sağlar.

Avantajı: Tekil desen, sistem genelinde tutarlı bir envanter durumu garanti ederek, çok kullanıcılı ortamlarda veri tutarsızlıklarını önler.

Decorator Deseni

Dekoratör deseni, siparişlere premium hizmetler (Kırılacak Eşya Koruması, Sigortalı Gönderim) eklemek için kullanılır ve maliyet ile açıklamaları dinamik olarak genişletir.

Uygulanması:

OrderDecorator soyut temel sınıfı, maliyet ve açıklama yöntemleri için arayüz tanımlar.

BaseOrder, temel Order nesnesini sarar; FragileShipping ve InsuredShipping, sırasıyla sabit (10 TL) ve yüzdesel (toplam maliyetin %5'i) ücretler ekler.

ECommerceSystem.create_order, kullanıcı tarafından seçilen premium hizmetlere göre dekoratörleri uygular.

Avantajı: Bu desen, temel Order sınıfını değiştirmeden premium hizmetlerin esnek bir şekilde eklenmesini sağlar ve genişletilebilirliği artırır.

Ek Detaylar

Veri Kalıcılığı: Sistem, ürünleri, kullanıcıları, siparişleri ve işlem günlüklerini siparisdatabase.json dosyasında saklar ve iplik güvenli dosya işlemleri için _file_lock kullanır.

Sipariş Durumu Yönetimi: OrderStatus enum'u, durumları (Beklemede, Hazırlanıyor, Yolda, Dağıtımda, Teslim Edildi, İptal Edildi) tanımlar ve kalan teslimat günlerine göre otomatik durum güncellemeleri yapar.

Teslimat Tahmini: Order.get_estimated_delivery yöntemi, teslimat tahminlerini hesaplar ve teslimat zaman çizelgesi ilerledikçe durum değişikliklerini tetikler.

Kullanıcı Arayüzü: CLI, kullanıcı kaydı, giriş, ürün tarama, sipariş oluşturma ve yönetimsel görevler için sezgisel bir arayüz sağlar.

III.Sonuç

E-ticaret platformu, proje gereksinimlerini başarıyla karşılayarak işlevsel, modüler ve kullanıcı dostu bir sistem sunar.

Güçlü Yönler

Modülerlik ve Genişletilebilirlik: Strateji, Gözlemci, Fabrika Yöntemi, Tekil ve Dekoratör desenlerinin kullanımı, yeni özelliklerin (örn. ek kargo yöntemleri veya bildirim kanalları) kolayca eklenmesini sağlar.

Sağlam İşlevsellik: Sistem, ürün yönetimi, sipariş işleme, kargo seçimi ve bildirimleri sorunsuz bir şekilde yönetir ve geçersiz

girişler ile stok eksiklikleri için hata işleme sunar.

Veri Kalıcılığı: JSON tabanlı depolama, veri sürekliliğini sağlar ve iplik güvenli işlemler yarış koşullarını önler.

Kullanıcı Deneyimi: CLI, hem müşteri hem de yönetici etkileşimlerini destekleyen bir arayüz sunar ve açık hata mesajları içerir.

Ek Özellikler: Kullanıcı kimlik doğrulama, sipariş notları, takip numarası oluşturma ve dinamik teslimat tahmini, minimum gereksinimlerin ötesinde işlevsellik sunar.

Zayıf Yönler

Sınırlı Ölçeklenebilirlik: JSON dosya tabanlı depolama, yüksek işlem hacimli büyük ölçekli sistemler için verimsiz olabilir. Bir veritabanı (örn. SQLite veya PostgreSQL) performansı artırabilir.

Basitleştirilmiş Bildirim Sistemi: Bildirim sistemi, e-posta ve SMS çıktılarını yazdırma ifadeleriyle simüle eder. Gerçek e-posta/SMS API'leriyle entegrasyon, üretim kullanımı için gerekli olurdu.

Hata Kurtarma: Temel hata işleme uygulanmış olsa da, bozuk JSON dosyaları veya ağ hataları (API entegrasyonu durumunda) için gelişmiş kurtarma mekanizmaları eksiktir.

Test Eksikliği: Kod tabanında, güvenilirliği sağlayacak ve bakımı kolaylaştıracak otomatik testler bulunmamaktadır.

Bölgesel Kısıtlamalar: Kargo yöntemleri, basitleştirilmiş bir bölge modeli (yerel veya yerel olmayan) varsayar. Daha ayrıntılı bir bölge sistemi, gerçekçiliği artırırdı.

Yapılabilecek İyileştirmeler

Bu zayıf yönleri ele almak için gelecekteki yinelemelerde şu adımlar atılabilir:

- JSON depolamayı, daha iyi ölçeklenebilirlik için ilişkisel bir veritabanıyla değiştirme.
- Gerçek bildirim servisleriyle entegrasyon (örn. e-posta için SMTP, SMS için Twilio).
- pytest gibi çerçevelerle birim testleri ekleme.
- Daha ayrıntılı coğrafi verilerle bölge modelini geliştirme.
- Tkinter veya Flask/Django ile grafiksel bir kullanıcı arayüzü (GUI) veya web arayüzü ekleme.

Sonuç olarak, e-ticaret platformu, OOP ve tasarım desenlerini etkin bir şekilde kullanarak işlevsel ve genişletilebilir bir sistem sunar. Modülerlik ve temel işlevsellik açısından başarılı olsa da, ölçeklenebilirlik ve gerçek dünya entegrasyonu yönleri, üretim ortamına geçiş için iyileştirilebilir.

Kaynakça:

https://github.com/rayyildiz/design-patterns

https://b47uh4n.medium.com/nesne-tabanl%C4%B1-programlamada-tasar%C4%B1m-desenleri-design-patterns-773df916a3e7

https://www.oodesign.com/

https://www.geeksforgeeks.org/python-oops-concepts/

https://www.tutorialspoint.com/python/python_oops_concepts.htm

https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/