YAZILIM GELİŞTİRME ORTAM ARAÇLARI VİZE PROJESİ

AKYÜZ MARKET

1. ALAN

1. HAFTA

1. Proje Tanımı ve Planlama:

Projenin kapsamının belirlenmesi

Hedef müşteri kitlesi ve kullanıcı ihtiyaçlarının analizi

Proje takviminin oluşturulması

Ekip üyelerinin rollerinin belirlenmesi

2. Gereksinim Analizi:

Müşteri ile toplantılar yaparak gereksinimlerin belirlenmesi

İhtiyaç duyulan özelliklerin ve modüllerin tanımlanması

Gereksinim belgesinin oluşturulması ve müşteri onayı

3. Tasarım:

Sistem mimarisinin belirlenmesi

Veritabanı modelinin oluşturulması

Arayüz tasarımının yapılması

Modül ve bileşenlerin ilişkilerinin belirlenmesi

4. Geliştirme:

Yazılım kodlarının yazılması

Modüllerin aşamalı olarak geliştirilmesi

Sürekli entegrasyon ve testlerin yapılması

Hata düzeltmeleri ve iyileştirmeler

2. HAFTA

5. Test Etme:

Birim testlerin yapılması

Entegrasyon testlerinin gerçekleştirilmesi

Sistem testlerinin ve kullanılabilirlik testlerinin uygulanması

Performans testlerinin yapılması

6. Dağıtım ve Uygulama:

Stok takip sisteminin kullanıma sunulması

Kullanıcı eğitimleri ve dokümantasyonun sağlanması

İlk kullanıcı geri bildirimlerinin toplanması

7. Bakım ve Destek:

Sistem üzerindeki güncellemelerin ve değişikliklerin yönetilmesi

Kullanıcıların yaşadığı sorunlara teknik destek sağlanması

Sistem performansının izlenmesi ve gerekirse iyileştirmelerin yapılması

8. Proje Değerlendirmesi:

Proje sürecinin gözden geçirilmesi

Başarı kriterlerinin analizi

Eksikliklerin ve başarıların belirlenmesi

Gelecekteki projeler için öğrenilen derslerin kaydedilmesi

Bu adımlar, genel bir yazılım geliştirme sürecini temsil eder ve projenin karmaşıklığına, büyüklüğüne ve gereksinimlerine bağlı olarak detaylandırılabilir.

***

2. ALAN

1. İteratif ve Artırımsal Gelişim:

Projenin küçük parçalara bölünerek, her bir parçanın ayrı ayrı geliştirilmesi ve test edilmesi.

Her iterasyon sonunda müşteriden geri bildirim alınarak, gereksinimlerin güncellenmesi.

2. Scrum Meetings:

Günlük "stand-up" toplantılar düzenlenerek ekip üyelerinin günlük ilerlemeleri paylaşması.

Sprint planlama toplantıları ile bir sonraki iterasyonun planının yapılması.

3. Müşteri Katılımı:

Müşteri, projenin her aşamasında aktif olarak yer alarak, gelişmeleri izler ve geri bildirim sağlar.

Müşteri talepleri, sık sık gerçekleşen toplantılarda değerlendirilir.

4. Değişen Gereksinimlere Hızlı Adaptasyon:

Proje süresince ortaya çıkan değişikliklere hızlı bir şekilde adapte olunması.

Yeni gereksinimlerin ve önceliklerin sürekli değerlendirilmesi.

5. Sık Sık Teslimatlar:

İterasyonlar sonucunda sık sık çalışan bir ürün sunulması.

Bu sayede müşteri sürekli olarak sistemi kullanabilir ve geri bildirim sağlayabilir.

6. Continuous Integration:

Kodun sürekli olarak entegre edilmesi ve test edilmesi.

Hata düzeltmeleri ve iyileştirmelerin hızlı bir şekilde uygulanması.

***

3. ALAN

UML Sınıf Diagramı:

Market Sınıfı:

Özellikler (Alanlar): marketAdi, adres, telefon, stokListesi, personelListesi

Metodlar: stokEkle(), stokSil(), personelEkle(), personelSil()

Ürün Sınıfı:

Özellikler: urunAdi, urunKodu, fiyat, adet

Metodlar: stokGuncelle(), fiyatGuncelle()

Personel Sınıfı:

Özellikler: personelAdi, personelSoyadi, personelID, pozisyon

Metodlar: satisYap(), stokKontrol()

Sipariş Sınıfı:

Özellikler: siparisNo, tarih, urunListesi, toplamFiyat

Metodlar: siparisOlustur(), siparisOnayla()

StokTakipSistemi Sınıfı:

Özellikler: marketListesi, siparisListesi

Metodlar: marketEkle(), marketSil(), siparisRaporuOlustur()

UML Kullanım (Use Case) Diagramı:

Market Personeli Kullanım Durumu:

Giriş Yapma:

Stok Kontrolü Yapma

Satış Yapma

Yönetici Kullanım Durumu:

Giriş Yapma:

Market Ekleme

Market Silme

Rapor Oluşturma

Müşteri Kullanım Durumu:

Giriş Yapma:

Ürünleri İnceleme

Sipariş Verme

Sipariş İptal Etme

Stok Takip Sistemi Kullanım Durumu:

Market Ekleme:

Market Silme

Rapor Oluşturma

Sipariş Takibi Yapma

***

Örnek Kodlama:

```
Class Urun:
    def _init_(self, urun_id, adi, stok_adedi, birim_fiyat):
        self.urun_id = urun_id
        self.adi = adi
        self.stok_adedi = stok_adedi
        self.birim_fiyat = birim_fiyat

class StokTakipSistemi:
    def _init_(self):
        self.urunler = []

    def urun_ekle(self, urun):
        self.urunler.append(urun)
        print(f"{urun.adi} ürünü sisteme eklendi.")

    def stok_sorgula(self, urun_id):
        for urun in self.urunler:
            if urun.urun_id == urun_id:
                print(f"{urun.adi} ürününün stok adedi: {urun.stok_adedi}")
                break
```

```
        else:
            print("Ürün bulunamadı.")

    def urun_sat(self, urun_id, adet):
        for urun in self.urunler:
            if urun.urun_id == urun_id:
                if urun.stok_adedi >= adet:
                    urun.stok_adedi -= adet
                    toplam_fiyat = adet * urun.birim_fiyat
                    print(f"{urun.adi} ürününden {adet} adet satıldı. Toplam fiyat: {toplam_fiyat}")
                    break
                else:
                    print(f"Yetersiz stok! Stok adedi: {urun.stok_adedi}")
                    break
        else:
            print("Ürün bulunamadı.")

# Örnek Kullanım
if _name_ == "_main_":
    # Stok Takip Sistemi oluştur
    market = StokTakipSistemi()

    # Ürünleri oluştur
    urun1 = Urun(1, "Beyaz Ekmek", 100, 2.5)
    urun2 = Urun(2, "Sıvı Yağ", 50, 15.0)

    # Ürünleri sisteme ekle
    market.urun_ekle(urun1)
    market.urun_ekle(urun2)

    # Stok durumunu sorgula
```

```
market.stok_sorgula(1)

# Ürün satışı yap
market.urun_sat(1, 20)

# Stok durumunu güncelle ve tekrar sorgula
market.stok_sorgula(1)
```

Bu örnek, bir market stok takip sistemini temsil eder. Şimdi, bu örneğin süreçlerini adım adım açıklayalım:

Urun Sınıfı:

Urun sınıfı, bir ürünün temel özelliklerini içerir: ürün ID'si, adı, stok adedi ve birim fiyatı.

StokTakipSistemi Sınıfı:

StokTakipSistemi sınıfı, marketin stok takibiyle ilgili işlemleri gerçekleştiren bir sınıftır.

init metodu, sistemi başlatır ve urunler adlı bir liste oluşturur.

urun_ekle metodu, yeni bir ürünü sisteme ekler.

stok_sorgula metodu, bir ürünün stok durumunu sorgular.

urun_sat metodu, bir ürünü belirli bir adette satar ve stok durumunu günceller.

Örnek Kullanım:

StokTakipSistemi sınıfından bir örnek oluşturulur (market).

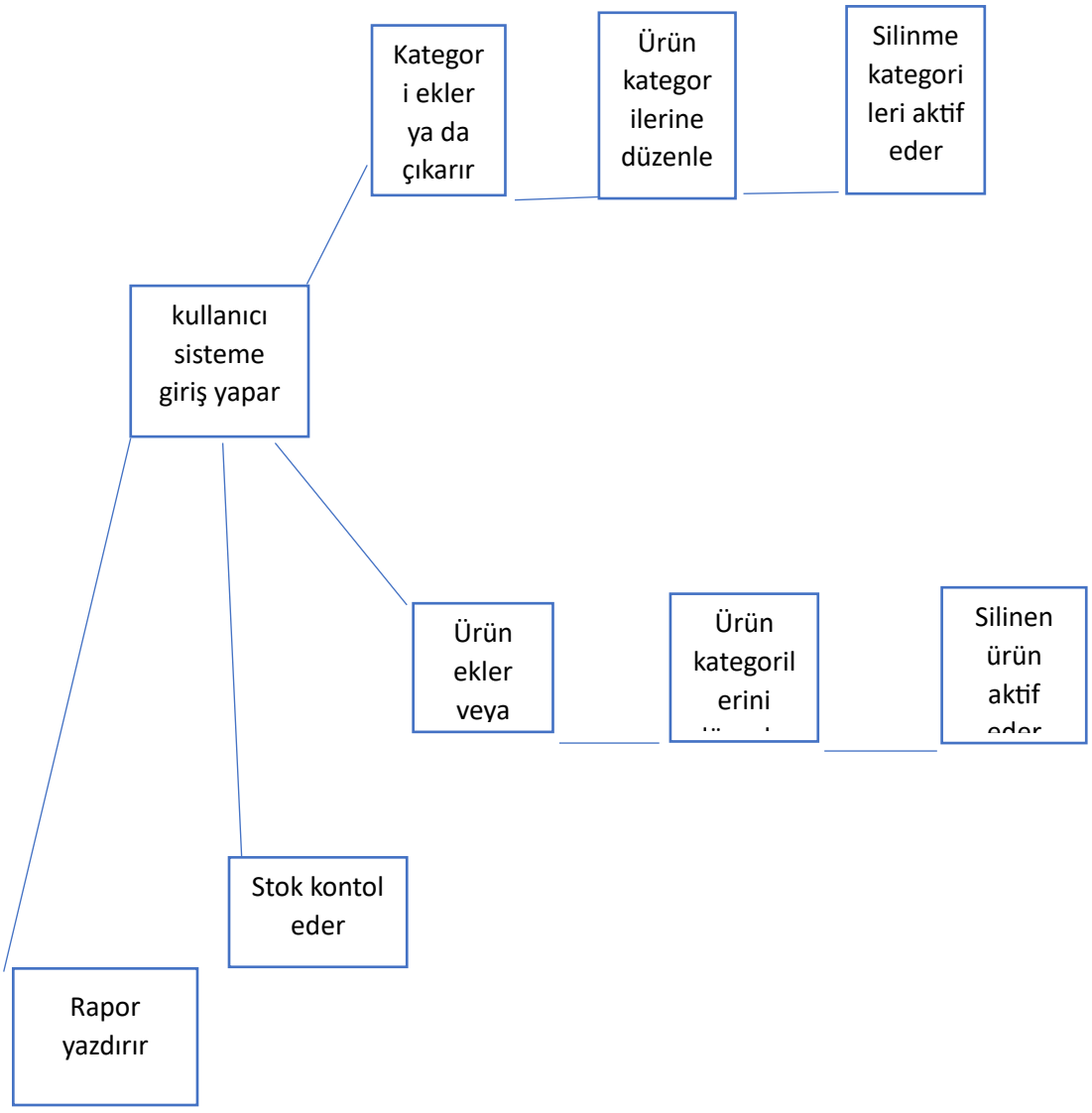
Urun sınıfından iki ürün oluşturulur (ekmek ve yağ).

Ürünler sisteme eklenir (urun_ekle).

Bir ürünün stok durumu sorgulanır (stok_sorgula).

Bir ürün satılır (urun_sat).

Satış sonrasında stok durumu tekrar sorgulanır (stok_sorgula)


Kategor i ekler ya da çıkarır
Ürün kategor ilerine düzenle
Silinme kategori leri aktif eder
kullanıcı sisteme giriş yapar
Ürün ekler veya
Ürün kategoril erini
Silinen ürün aktif
Stok kontol eder
Rapor yazdırır