YAZILIM GEREKSİNİM ANALİZİ
GÜZ DÖNEMİ PROJE ÖDEVİ

Öğrenci No:

190601017
200601036
200601038

İçindekiler

İterasyon-1

Hafta-1: Projenin Özeti

POSBAR adını verdiğimiz bu projemizde yazarkasa satış ve barkod okuma sistemini pos cihazına entegre ederek pos cihazını sadece ödeme aracı olarak kullanmaktan kurtarıp aynı zamanda satış aracı haline getirerek pos cihazıyla satış sistemini pratikleştirmek ve yazarkasanın pos cihazı halinde daha hafif, daha kullanışlı bir halde her yerde her zaman kullanılmak üzere taşınabilirliğini sağlamak, alışveriş dünyasının ödeme ve satış problemlerine fiziksel çözümler yerine yazılımsal çözümler üreterek yükte ve pahada daha ucuz bir alternatif sağlanması hedeflenmektedir. . Böylece birden fazla satış ve ödeme aracı kullanmak yerine tek bir makineye yazılımsal işlevler yükleyerek, maliyeti azaltarak, verimliliği arttırmak amaçlanmaktadır.

Alışveriş dünyasının (giyim, gıda vb.) markalaşmış şirketleri ve vizyoner esnaf için kullanışlı olan Posbar, yazarkasa sistemini daha minimal ve işlevsel hale getirecektir. Bununla birlikte yazarkasa sisteminin kapladığı alan ve maliyetlerinin yüksek olması açısından bir alternatif oluşturan Posbar tercih edilebilir olması yönüyle de dikkatleri çekmesi açısından da avantajlıdır.

Ancak projenin pos cihazının eski işlevleri açısından alışılageldik olmaması, süregelen sistemlere uyum sağlayamaması, kart ödemelerine sağladığı kolaylığı nakit ödemelere gösterememesi; yazılımsal, donanımsal gereksinimlerin doğru ve etkili bir şekilde entegre edilememesi, taşınabilir olmasından dolayı cihazın kolay kaybolabileceği, çalınabileceği ya da düşmelere bağlı olarak göreceği hasarların çoğunlukta olması gibi birtakım zorluklar vardır. Özellikle kaybolma, çalınabilme ve hasar görme gibi olaylar bize maliyet ve zaman kaybı yaşatması dezavantaj oluşturmaktadır.

Projenin sahip olduğu dezavantajların yanında gelecekte sahip olabileceği avantajların artacağı öngörülmektedir. Zira gelişen teknoloji ile sağlanacak güvenli önlemlerin (örneğin, pos cihazına GPS uygulaması eklenerek kaybolma ve çalınma durumlarında konum tespitinin yapılabilmesi ya da yetkili olmayan kişilerin kullanmasını engellemek için pos cihazı açılışını şifre ya da parmak izi yapmak gibi vs.) alınması dezavantajlarının azalacağı öngörüsünü taşımaktadır. Projenin nihai amacı akıllı bir pos cihazı tasarlayarak yazarkasa ile yapamayacağımız birçok işlem ve işlevi tek cihazla yapmayı sağlamaktır. Alışveriş dünyasının akıllı telefonu haline gelecek olan POSBAR daha verimli ve kullanışlı bir yazarkasa alternatifidir

Hafta-2: Başlangıç Analizi

Bu projeyi işlemeden önce amacımız, karşımızdaki sorunlara cevap bulmak ve cevapların maliyetlerine göre proje çalışmasını yürütmenin ne kadar sağlıklı olduğu sonucunu bulmaktır. Burada ulaştığımız sonuca göre projemizin faydalı mı yoksa zaman kaybı mı olduğu ortaya çıkacaktır.

1) İzinler:

-Bankalarca müşterilerin kart bilgilerine ve diğer kişisel bilgilerine erişim izni.

2) Doğrulama:

-Gönderilen işlem talebinin banka tarafından onaylanması, doğrulanması

-Makinenin okuduğu barkodların doğruluğu

-Okunan barkodların yazılım tarafına dökülürken doğru çevrilmesi ve veri tabanında doğru erişim sağlama

-Doğru fiş kesimi (Fiş kesilirken aynı zamanda veri tabanı üzerinde ürün stok işlemleri, fiyatlandırma, stok takibi gibi durumların doğruca yapılması)

-Kredi/Banka kartlarının cihaz tarafından doğru okunması

3) Veri Güvenliği ve Depolama:

-Ürün, müşteri, satış geçmişi gibi saklanması zorunlu verilerin "MongoDB" veri tabanı üzerinde saklanması düşünülüyor. Aynı zamanda bu kiralama maliyetini de beraberinde getirecektir.

4) Makinenin Fiziksel İhtiyaçları:

-Kredi/Banka kartlarını okuyabilmesi için sensör

-Barkod okuması/tanıması için sensör

-Her alışveriş ya da iade sonrası işlem için fiş kesmelidir. Bunun için makinenin içerisinde bir yazma aygıtına ihtiyaç vardır. Bu direkt özel bir aygıt olmak zorunda değildir. Aynı işi yapabilecek bir ekleme yeterli olacaktır.

5) Yazılım Gereksinimleri:

-Barkod Yazılımı

-Kredi/Banka Kartı Yazılımları

-Yukarıdaki her iki yazılım ile beraber çalışacak ücretlendirme ve veri tabanı yazılımları.

-Bütün yazılımların üzerinde çalışabilmesi için gerekli ve sistem üzerindeki aygıtları denetleyerek onları yazılım arayüzüne bağlayacak olan bir gömülü sistem yazılımı.

-Tamamen kesin olmamakla beraber kriptografi yazılımı. (Eğer maliyet konusunda çok sorun yaşanmazsa ve güvenlik tehdidi varsa önlem amaçlı kullanılacaktır.)

-Muhasebe yazılımı

-Sistem entegrasyonu

-Test

6) Maliyet (Tahmini):

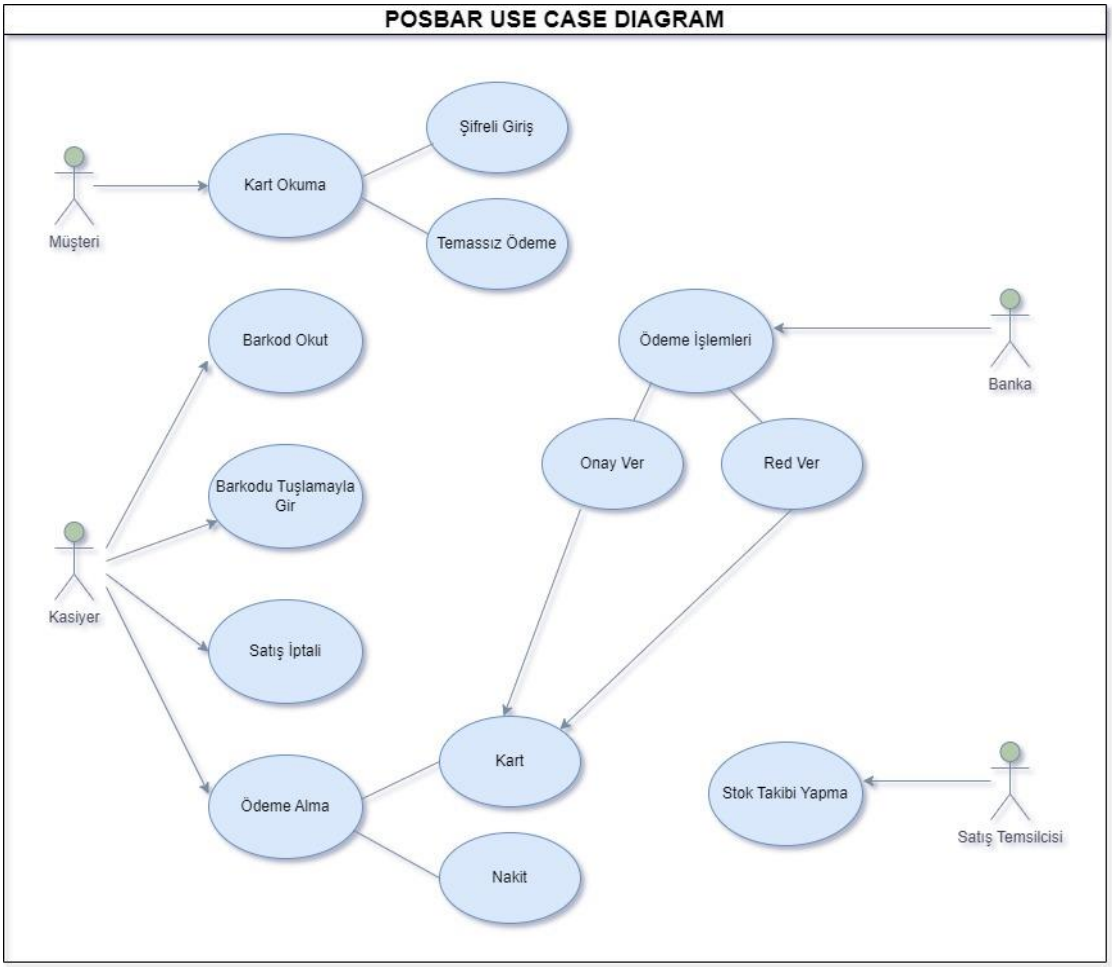
-Gerekli yazılımların tasarımları, gerçekleştirilmesi için yazılım/bilgisayar mühendisi- 3
-Düzenli kontroller için bakım ekibi - 1
-Cihaz için gerekli sensörler ve yazıcı.
-Veri tabanı sunucusu için bu alanda çalışan bir yazılım/bilgisayar mühendisi ve veri tabanı sunucu kiralama
-Kriptografi için yazılım/bilgisayar mühendisi (kesin değil?)

Hafta-3: Kullanım Durumları

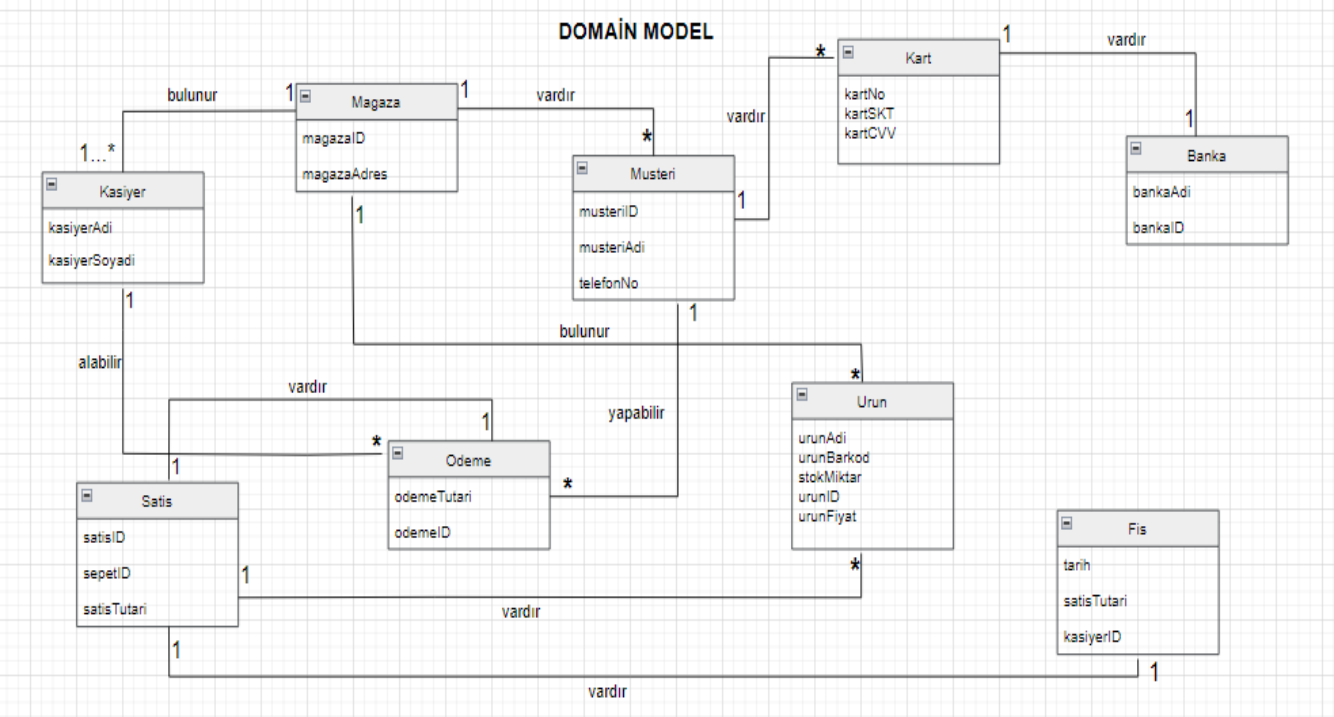
- **Kart Okutma**
  - Temassız Ödeme
    - Kart Tanıma
    -Banka Doğrulaması
    -Onay/Ret İşlemi

  - Şifreli Giriş
    - Kart Tanıma
    - Banka Doğrulaması
    - Onay/Ret İşlemi
- **Barkod Okutma**
  - Barkod Doğrulaması
  -Ürün Bilgilerini Görüntüle
  -Ürün Havuzuna Ekle
- **Satış İptali**
  - Ürünü Havuzdan Sil
  -Geçerli Ürün Havuzunu Sil
- **Barkodu Tuşlamayla Gir**
  -Barkod Doğrulaması
  - Ürün Bilgilerini Görüntüle
  -Ürün Havuzuna Ekle
- **Ödeme Alma**
  -Kart ile
    - Satış tutarı ve kart bilgisinin bankaya gönderilmesi
    - Ürünlerin stoktan düşmesi
  -Nakit ile
    - Ürünlerin stoktan düşmesi
- **Ödeme İşlemleri**
  -Onay ver
    - Hesaptan kesim yapılması
    - Kullanıcıya işlem bildirilmesi
  -Ret Ver
    - Cihaza ret bildirimlerinin gönderilmesi
    - Kullanıcıya işlem ret bildirimi

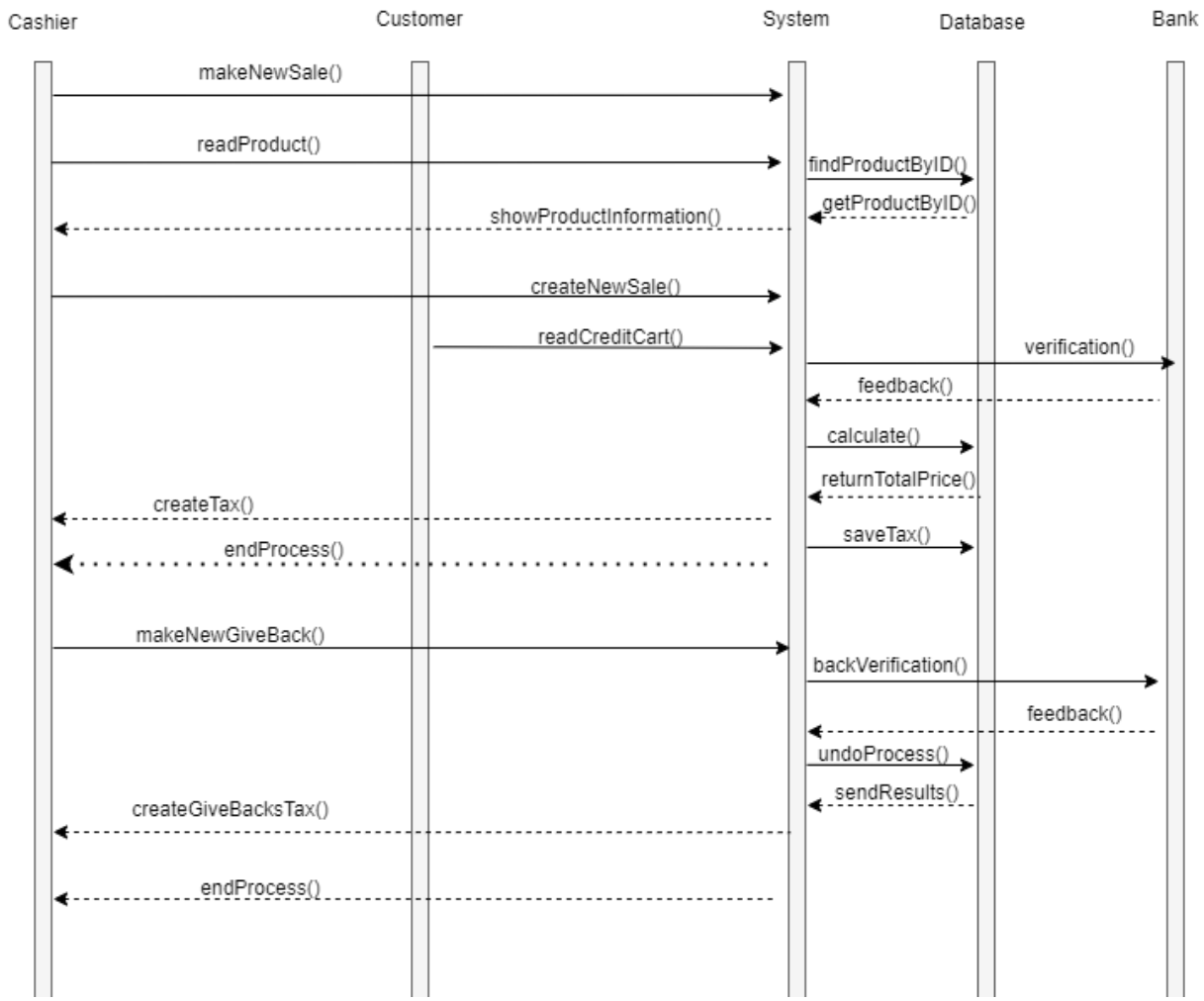
- **Stok Takibi Yapma**
  -Ürün stoklarını kontrol et
  -Tükenmekte olan ürünlerin bildirimi



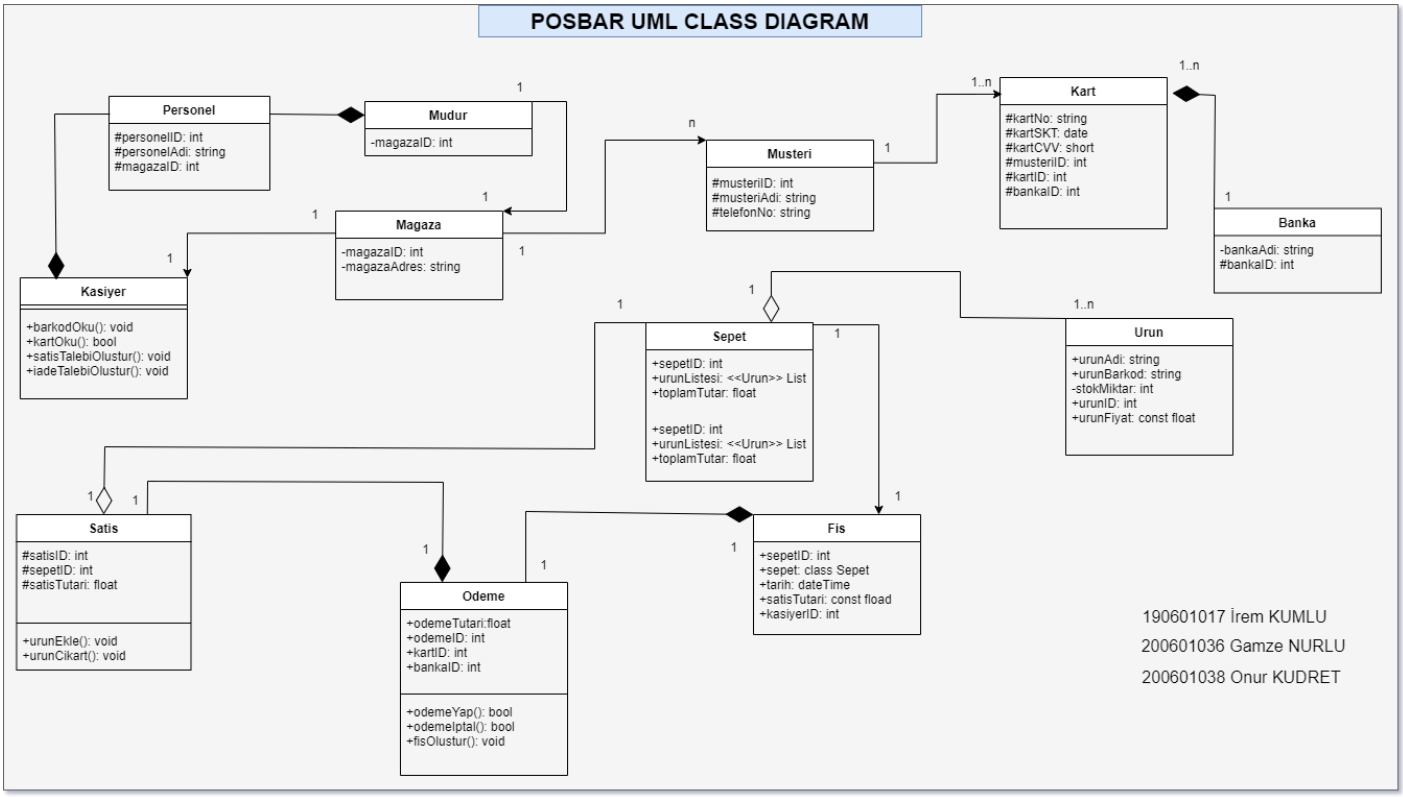
Hafta-4: Değerlendirme ve Alan Modeli



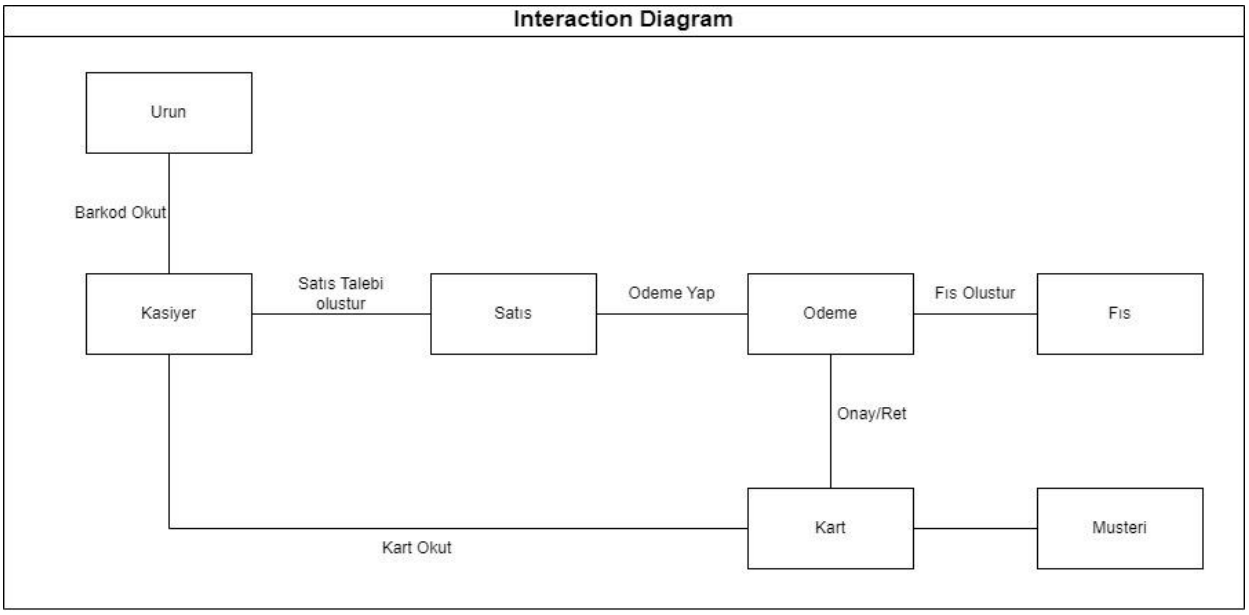
Hafta-5: Sistem Geçiş Diyagramları



Hafta-6: Sınıf Diyagramları



Hafta-7: Sınıf Etkileşim Diyagramları



1- Kasiyer ürün barkodunu okutur.

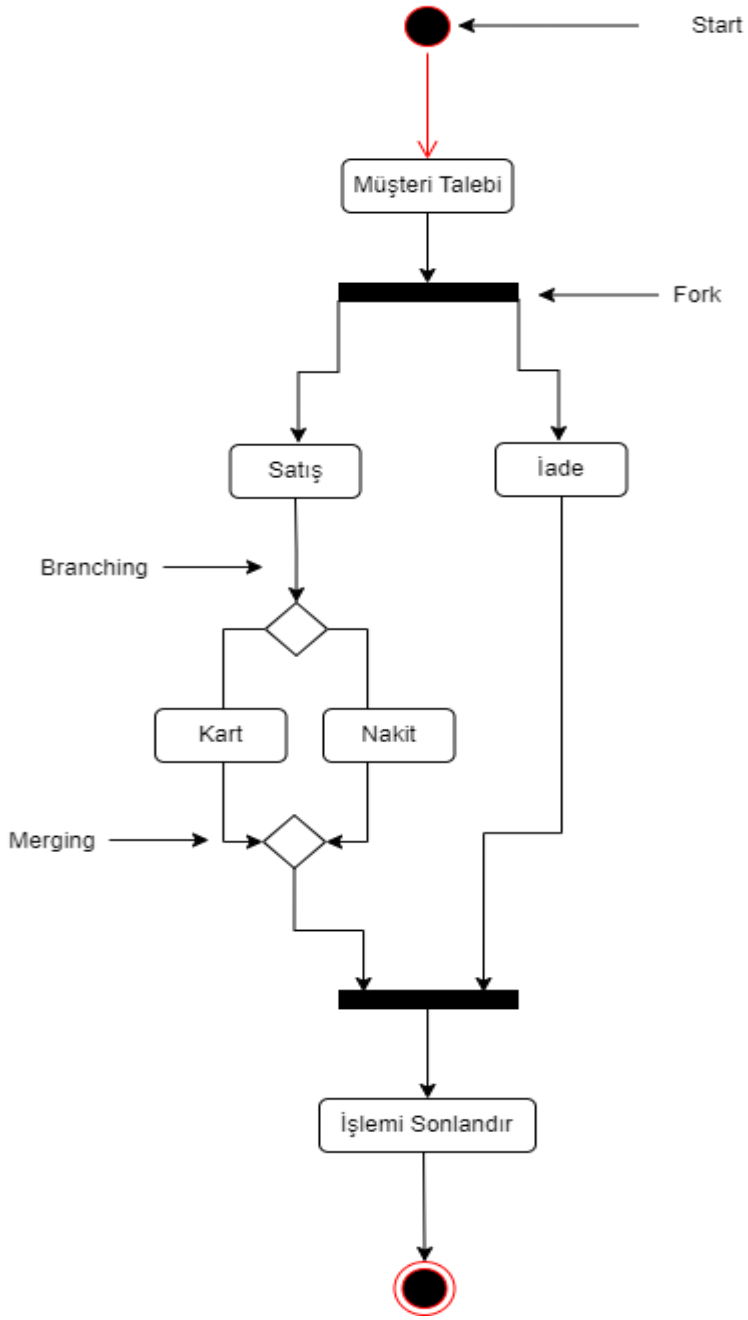
2- Okunan barkod ile satış talebi oluşur.

3- Oluşan satış ödeme talebinde bulunur.

4- Kart ile yapılan ödeme işlemi onay veya ret geri bildirimi ile fiş oluşur. Satış işlemi gerçekleşir.

Hafta-8: Aktivite Diyagramları ve Modelleme

AKTİVİTE DİYAGRAMI



Hafta-9: Tasarım Desenleri

Tasarım desenleri, yazılım tasarımında yaygın olarak ortaya çıkan sorunlara tipik çözümlerdir. Kodumuzda yinelenen bir tasarım problemini çözmek için özelleştirebileceğimiz önceden hazırlanmış B planı gibidirler. Ek olarak, tüm modeller amaçlarına göre kategorize edilebilir. Örneğin;

Singleton, bir sınıfın yalnızca bir nesneye sahip olmasını sağlayan bir tasarım desenidir. Peki neden birisi bir sınıfın kaç nesnesi olduğunu kontrol etmek istesin? Bunun nedeni, bir veritabanı veya dosya gibi bazı paylaşılan kaynaklara erişimi kontrol etmektir. Buna çözüm olarak şu yapılır, diğer nesnelerin Singleton sınıfıyla new operatorünü kullanmasını önlemek için geçici constructor private edilir. Örneğin çalışmamızdaki veritabanı sınıfını bir Singleton sınıfı gibi davranır. Bu sınıfın public bir constructor'u yoktur, bu nedenle nesnesini almanın tek yolu getInstance yöntemini çağırmaktır. Bu yöntem, ilk oluşturulan nesneyi önbelleğe alır ve onu sonraki tüm çağrılarda döndürür. Singleton ya yeni bir nesne yaratır ya da önceden oluşturulmuşsa var olan bir nesneyi döndürür böylece.

Kullandığımız bir sonraki Factory Method deseni ile nesne yaratılma işini inheritance yoluyla client-side'dan ayırıp sub-classes'lara vermek amaçlanır. Aynı interface'i veya abstract sınıfı implement etmiş factory nesnelerinin üretiminden sorumlu pattern'dir.

Geliştirmekte olduğumuz uygulamaya yeni bir feature eklerken en az dokunuş ile client'ı bu duruma hiç sokmadan yapabilmeyi amaçlıyoruz. Factory pattern de bu amaca yönelik olarak önerilen en önemli pattern'lerden birisi olduğu için bu deseni tercih ettik.

Factory Method'da bir sınırlama vardır: alt sınıflar, yalnızca bu ürünlerin ortak bir temel sınıfa veya interface'e sahip olması durumunda farklı türde ürünler döndürür. Ayrıca, temel sınıftaki factory yönteminin dönüş türü bu interface olarak bildirilmelidir.

Oluşturduğumuz projenin genişlemesi ve kapsam alanın artması durumunda kodumuzla birlikte çalışması gerekecek olan nesnelerin tam türlerini ve bağımlılıklarını önceden bilememiz durumunda da Factory method kullana bilecek olmamız büyük avantajdır.

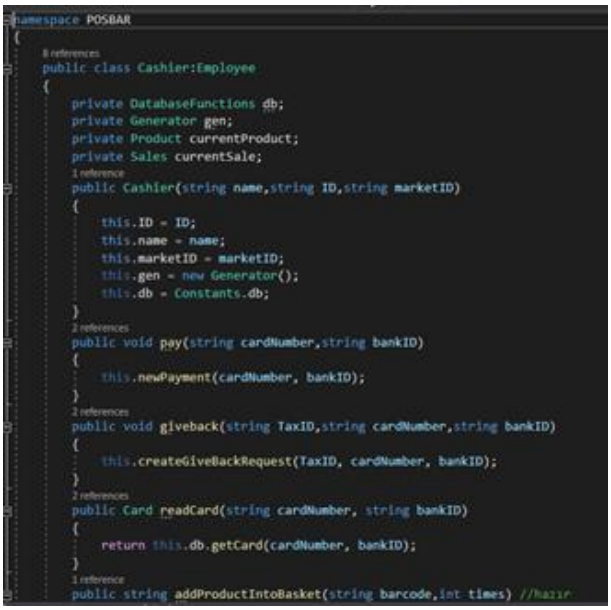
Örneğin, uygulamamıza yeni bir ürün türü eklemek için yalnızca yeni bir creator alt sınıfı oluşturmamız ve içindeki Factory yöntemini override etmemiz gerekir.

Factory Design Pattern, veri tabanı bağlantıları, dosya sistemleri ve ağ kaynakları gibi büyük, kaynağı yoğun nesnelerle uğraşırken, var olan nesneleri her seferinde yeniden oluşturmak yerine yeniden kullanarak sistem kaynaklarından tasarruf etmemizi sağlar.

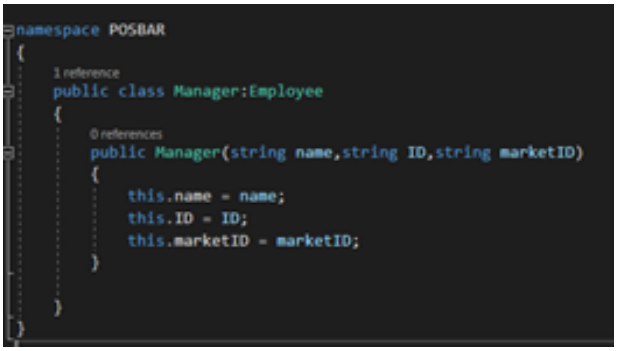
Factory Design Pattern avantajları: Ürün oluşturma kodunu programda tek bir yere taşıyarak kodun desteklenmesini kolaylaştırabiliriz.

Hafta-10: Sınıf Uygulanması

```
namespace POSBAR
{
    2 references
    public abstract class Employee
    {
        10 references
        public string ID { get; protected set; }
        7 references
        public string name { get; protected set; }
        4 references
        public string marketID { get; protected set; }

    }
}
```



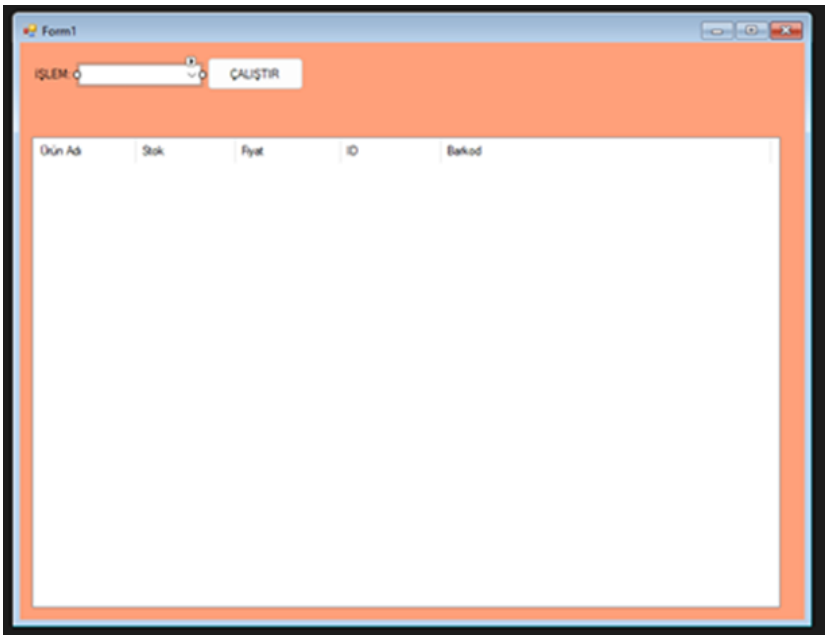
```
namespace POSBAR
{
    8 references
    public class Cashier:Employee
    {
        private DatabaseFunctions db;
        private Generator gen;
        private Product currentProduct;
        private Sales currentSale;
        1 reference
        public Cashier(string name,string ID,string marketID)
        {
            this.ID = ID;
            this.name = name;
            this.marketID = marketID;
            this.gen = new Generator();
            this.db = Constants.db;
        }
        2 references
        public void pay(string cardNumber,string bankID)
        {
            this.newPayment(cardNumber, bankID);
        }
        1 reference
        public void giveback(string TaxID,string cardNumber,string bankID)
        {
            this.createGiveBackRequest(TaxID, cardNumber, bankID);
        }
        2 references
        public Card readCard(string cardNumber, string bankID)
        {
            return this.db.getCard(cardNumber, bankID);
        }
        1 reference
        public string addProductIntoBasket(string barcode,int times) //hazır
```

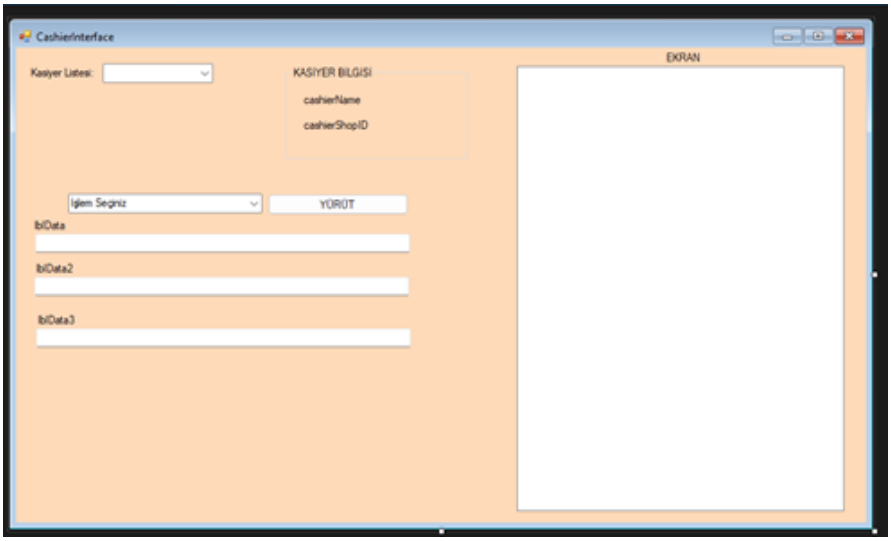


```
namespace POSBAR
{
    1 reference
    public class Manager:Employee
    {
        0 references
        public Manager(string name,string ID,string marketID)
        {
            this.name = name;
            this.ID = ID;
            this.marketID = marketID;
        }

    }
}
```

```
14 references
public class Product
{
    public string name;
    public string barcode;
    public int amount;
    public float price;
    7 references
    public string ID { get; private set; }
    1 reference
    public Product(float price,int amount,string barcode,string name,string ID)
    {
        this.price = price;
        this.amount = amount;
        this.barcode = barcode;
        this.name = name;
        this.ID = ID;
    }

    2 references
    public string getProductInfos()
    {
        return name + "\n" + barcode + "\n" + ID + "\n" + price.ToString()+"\n"+amount+"\n";
    }
}
```

```
namespace POSBAR
{
    6 references
    public class Bank
    {
        private string name;
        private string ID;
        private List<Card> cards;
        1 reference
        public Bank(string name,string ID)
        {
            this.name = name;
            this.ID = ID;
            this.cards = new List<Card>();
        }
        1 reference
        public void addNewCard(Card c)
        {
            this.cards.Add(new Card(c.getCardNumber(this.ID), c.date, c.CVV, c.customerID, c.cardID, this.ID, c.limit));
        }
        1 reference
        public Card getCard(string cardNumber)
        {
            return this.cards.Find(x=> x.confirm(cardNumber,this.ID));
        }
        1 reference
        public bool confirm(string bankID)
        {
            return bankID == this.ID;
        }
    }
}
```


Form1
İŞLEM:
ÇALIŞTIR
Ürün Adı
Stok
Fiyat
ID
Barkod


CashierInterface
Kasiyer Listesi:
KASİYER BİLGİSİ
cashierName
cashierShopID
EKRAN
İşlem Seçiniz
YÜRÜT
lblData
lblData2
lblData3

Hafta-11: Değerlendirme

POSBAR proje içeriğimizde, satış sistemini pratikleştirmek amacıyla pos cihazı ve barkod okuma sistemini tek bir cihazda entegre ederek alışveriş dünyasının ödeme ve satış problemlerine fiziksel çözümler yerine yazılımsal çözümler ürettik. Bu amaç doğrultusunda öncelikle, proje içerisinde düşünülen artı ve eksilerin belirledik. Bunları belirlerken kapsamlı ve etkili bir vizyon üzerinden ilerledik. Birçok sınıf oluşturduk ve bu sınıflar arasındaki ilişkileri büyük bir titizlikle inşa ettik. Örneğin ödeme sınıfımız; ödeme yap, ödeme iptal ve fiş oluştur metotlarını içermektedir. Bu sınıfımızı, yine oluşturduğumuz satış ve fiş sınıfları ile birebir iletişim kurdurarak; ödeme işlemi ve sonuç çıktısı olan fiş oluşumunu pos sistemimize entegre ettik.

İşçi sınıfımızı abstract class belirleyerek bir ana sınıf oluşturmayı hedefledik. Böylelikle gereksiz kod yazımının önüne geçtik. Kasiyer ve yönetici sınıflarımız işçi sınıfımızdan kalıtım alarak sınıflar üzerinde ortak metot ve özellikler kullanılmasını sağladık bu da kodumuzu pratik ve daha kullanışlı hale getirdi.

İleriki vadede karşılaşabileceğimiz problemleri göz önünde bulundurarak Singelton ve Factory tasarım desenlerini kullandık. Bu tasarım desenleri projemizi pratikleştirmemize de katkı sağladı Geliştirmekte olduğumuz uygulamaya projenin ileriki aşamalarında yeni bir feature eklerken en az dokunuş ile client'ı bu duruma hiç sokmadan yapabilmek amacıyla Factory metodunu kullanmayı planladık.

Hafta-12: İterasyon-2 Değerlendirme Toplantısı ve Yapılacaklar

İTERASYON	HATA-EKSİKLİK	DÜZELTME ÖNERİSİ	AÇIKLAMA
İTERASYON-2	Use-Case Diyagramları üzerinde eksik kullanım durumlarının olması.	Use-Case Diyagramına "Kasiyer" sınıfı için ödeme işlevi eklenmesi ve diyagrama yansıtılması.	"Kasiyer" sınıfı üzerinde ödeme gerçekleştirmek için cihazı kontrol edebilecek bir işlev bulunmamaktadır. Bu yüzden sadece ürün ekleme/çıkarma işlemleri yapılıyor ancak yetersiz kalmaktadır.
İTERASYON-3	Domain Model üzerinde veri tabanından bahsedilmemesi.	Domain model üzerinde güncelleme yaparak veri tabanını alanlar arasında ilişkilendir	Domain Model üzerinde veri tabanı için bir alan ve diğer alanlarla ilişkileri bahsedilmediği için yazılım alanında zorluklar yaşanmıştır.
İTERASYON-4	Nakit ödeme yöntemlerinin yetersiz olması.	POSBAR ve alan maliyeti ile alakalı değişiklikler yapmak.	Cihaz geliştirilirken sadece sanal ödeme yöntemleri üzerine yoğunlaşıldığı için normal ödeme yöntemlerinde çıkmaktadır.
İTERASYON-5	Ürün ekleme/çıkarma işlemlerinin "Sepet"/ "Satış" sınıfları arasında ilişkilendirilerek işlenmesi.	Ayrı ayrı sınıflar üzerinde ilişkilendirme yoluyla veri akışı yapmak yerine sınıflar arası ilişkileri kullanarak hareket etme.	Sepet ve Satış sınıfları üzerinde ürün ekleme/çıkarma işlemleri için fazla değişkenler kullanarak veri ya da verinin adresi taşınmaktadır. Bunun ise tek sebebi yanlış sınıf tasarımıdır.
İTERASYON-6	Use-Case Diyagramları üzerinde eksik kullanım durumlarının olması.	Kart sınıfı üzerine sınıf özelliklerini geri döndüren bir işlev eklenmesi ve diyagrama yansıtılması.	Kart sınıfı üzerinde kart bilgilerine erişmek için geliştirilmiş bir yöntem olmadığı için güvenlik konularına önem verildiği zaman bu sınıfa ait hiçbir özelliğe erişilmemektedir.