

**FIRAT ÜNİVERSİTESİ**  
TEKNOLOJİ FAKÜLTESİ  
YAZILIM MÜHENDİSLİĞİ

NESNE TABANLI PROGRAMLAMA

**RESTAURANT OTOMASYONU**

KAMİL KAPLAN

Doç. Dr. Erkan TANYILDIZI  
Arş. Gör. Zafer GÜLER

Mayıs—2017

**1 Giriş**

* 1. **Projenin Amacı**

Restaurant Otomasyon sistemi,

* 1. **Projenin Kapsamı**

Restaurant içerikli şirketlerin kullanımını kapsayan görüşme kayıt ve

**1.3 Tanımlamalar ve Kısaltmalar**

**ACCESS :** Veritabanı yönetim sistemi olup verilerin yönetilmesini sağlamak ve diğer programlama platformları ile etkileşim içinde veri bütünlüğünü saylayan ortamdır.

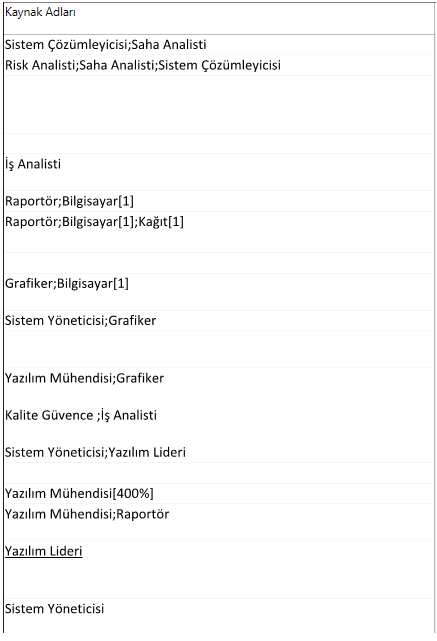
**C# :**  Nesne yönelimli bir yazılım dilidir. Yani nesneler önceden sınıflar halinde yazılıdır. Programcıya sadece o nesneyi sürüklemek ve sonrasında nesneyi amaca uygun çalıştıracak kod satırlarını yazmak kalır.

1. **Proje Planı**
   1. **Giriş**

Restaurant otomasyonu,

* 1. **Projenin Plan Kapsamı**

Proje benzer restaurant otomasyonlarından farklı olarak kapsamı dahilinde kullanıcıların takip kısmını barındırır.



****

* 1. **Proje Ekip Yapısı**

Proje masaüstü uygulaması olup tek bir kişi ile yani proje yöneticisi tarafından yapılacaktır.

Proje ERP projesi haline getirilirse, proje kapsamında Lider ve iş yöneticisi alınabilir.

* 1. **Önerilen Sistemin Teknik Tanımları**
  2. **Kullanılan Özel Geliştirme Araçları**

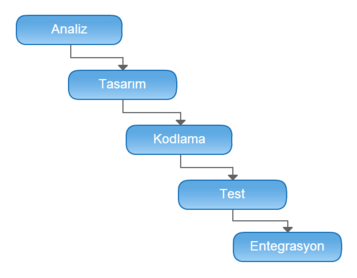
**Proje belgelendirmesi :** Microsoft Word, Microsoft Project, Microsoft Visio

**Programlama araçları :** Microsoft Visual Studio

**Veri tabanı işlemleri :** ACCESS

* 1. **Sistem Metodolojisi**

Bu proje için Şelale Modeli uygun görülmüştür. Çağlayan modeli aşağıdaki gibidir.



* + 1. **İşlevsellik**
* Gelişmiş arama özellikleri de dâhil olmalıdır.
* Hangi verilerin olacağı ve nasıl yapılandırılacağı belirlenmiştir .
* Otomasyon veri tabanı üzerinden temel bilgilere ulaşılabilecektir.
* Yüksek güvenlik standartları oluşturulmuştur.
  + 1. **Kullanılabilirlik**
* Kullanılabilirliği etkileyen genel etkenler bilinmektedir.
* Seçilen yazılımın kullanılabilirliğini etkileyen etkenler tespit edilip gerekli çalışmalar yapılmıştır.
* İyi kullanılabilirlik sağlamak için insan ve mali kaynak sağlanacaktır.
* Alt en uygun kullanılabilirlik durumunda uygulama ve iyileştirme çalışmaları yapılacaktır.
  1. **Kalite Sağlama Planı**

Ağırlıklı olarak aşağıdaki kategorilerde ki kalitenin sağlanması çalışılmıştır.

Minimum gereksinimlerin sağlanması

* 1. **Konfigürasyon Yönetim Planı**

Konfigürasyon Yönetimi süreci şunları içerir ;

Belirlenen zamanlarda, seçilen özelliğin ana hatlarını oluşturacak konfigürasyonun tanımlanması

Konfigürasyon birimlerindeki değişimlerin kontrol edilmesi

Konfigürasyon yönetimi altındaki ürünlerin oluşturulması için gerekli şartların sağlanması

Takımın sahip olduğu veri girdilerinin değişmesi

Kullanıcı sayısına göre sunucuların ve istemcilerin yetersizliği

* 1. **Kaynak Yönetim Planı**
  2. **Risk Planı**

Sahada araştırma sürecinden sonra belirlenir.

* 1. **Eğitim Planı**

Sistemin kullanımı için eğitim planı hazırlanmamıştır. Tüm menü ve buton isimleri yaptıkları işlemi anlatmaktadır.

* 1. **Test Planı**

Proje üzerinde Beta testi yapılacaktır. Projenin sonlarında gerçekleştirilecek olan sınama çalışmasında test aracı ve program kullanılarak hatalı ve eksik yönleri tespit edilmeye çalışılacaktır. Tespit edilen hata ya da eksiklikler yazılım ekibine bildirilecektir. Burada dikkat edilmesi gerekenler şu şekildedir:

* İşlevselliğin sorunları nelerdir?
* Ara yüzde nelere dikkat edilmelidir?
  1. **Bakım Planı**

Düzenli olarak yapılacak bakım sürecinde aşağıdaki maddelere uyulacaktır.

* Yeni işlev ve özelliklerin eklenmesi
* Eklenecek yeni teknolojiler
* Yeni işletim sistemleriyle uyumluluk
* Sistemde bir hata yada eksiklik oluştuğunda müdahale edilmesi
* İlgili işletim sisteminin yeni servis paketleriyle uyumluluk
* Kullanılan virüs tarayıcının yeni sürümleri ve güncellemeleriyle uyumluluk
* Yeni eğilimlere sürekli olarak uyarlanabilme özelliği
* Gelişmiş kullanışlık ve işletim güvenliği

**3 Sistem Çözümü**

**3.1 Mevcut Sistem İncelemesi**

**3.1.1 Örgüt Yapısı**

Mevcut sistemin örgüt yapısı iki alt başlıkta toplanır. Bunlar sistemi oluşturan taraf ve sistemi kullanan taraftır.

**Sistemi oluşturan taraf:**

**Proje Yöneticisi:** Projeyi ve ekip yapısını yöneten kişi

**Sistemi kullanan taraf:**

**Yönetici:** Sistemin yöneticisi ve temsilcisidir. Sistemim kullanan en üstü yöneticidir.

**Kullanıcı:** Sistemin kurulu olduğu donanımlarda ihtiyacına uygun işlemleri yapmaktadır.

**3.1.2 İşlevsel Model**

**3.1.4 Var Olan Yazılım / Donanım Kaynakları**

**Yazılım Kaynakları**

C#

ACSEES Database

**Donanım Kaynakları**

Serverlar

**3.2 Önerilen Sistemin Mantıksal Modeli**

**3.2.1 Giriş**

Bu bölümde önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünde çözümleme yapılır. Bu model daha çok bilgi sistemini geliştirecek teknik personele yöneliktir. Mantıksal model olarak da tanımlanır.

**3.2.2 Genel Bakış**

***Personel İşlemleri:***

Yöneticilerin yeni personel girişi, mevcut personellerin bilgilerinin düzenleyebilmesi, personel çıkışının yapılması gibi işlemleri yapar.

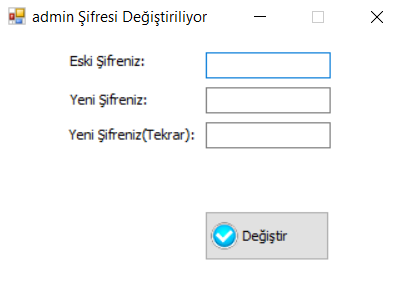
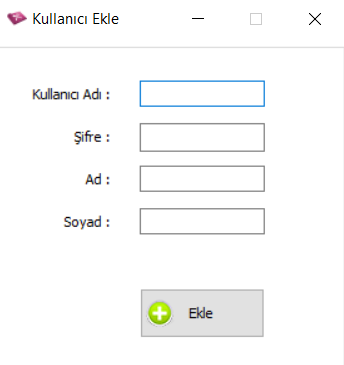
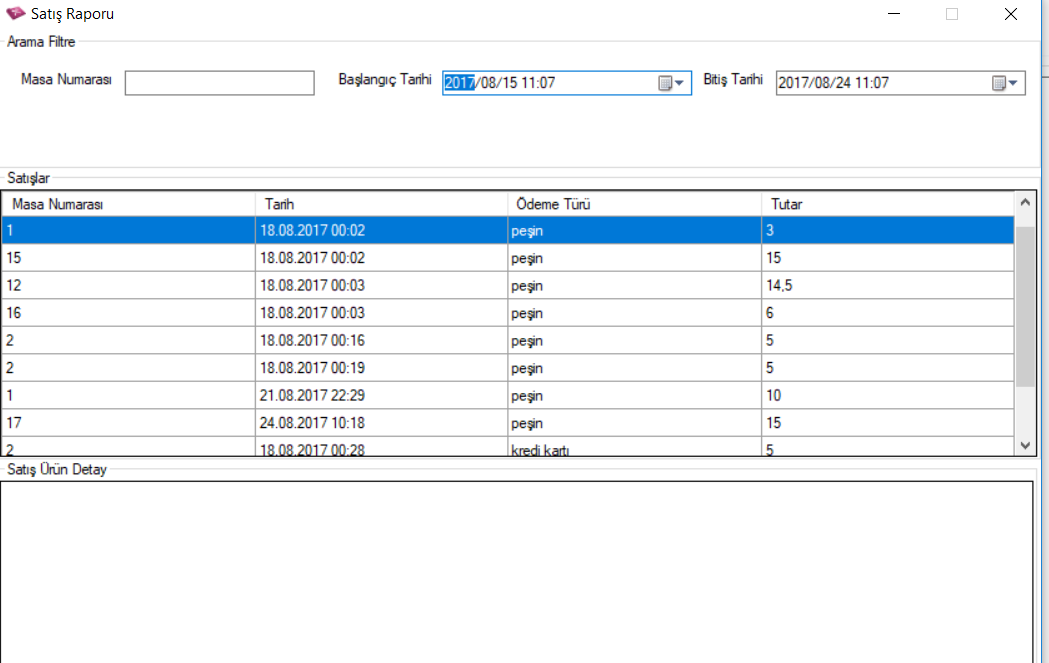
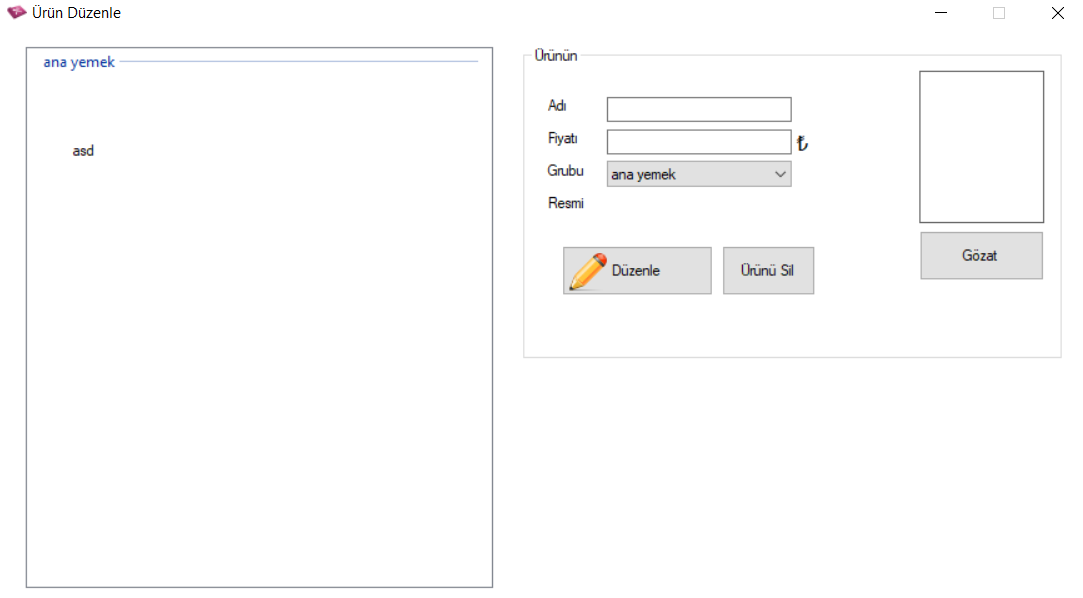
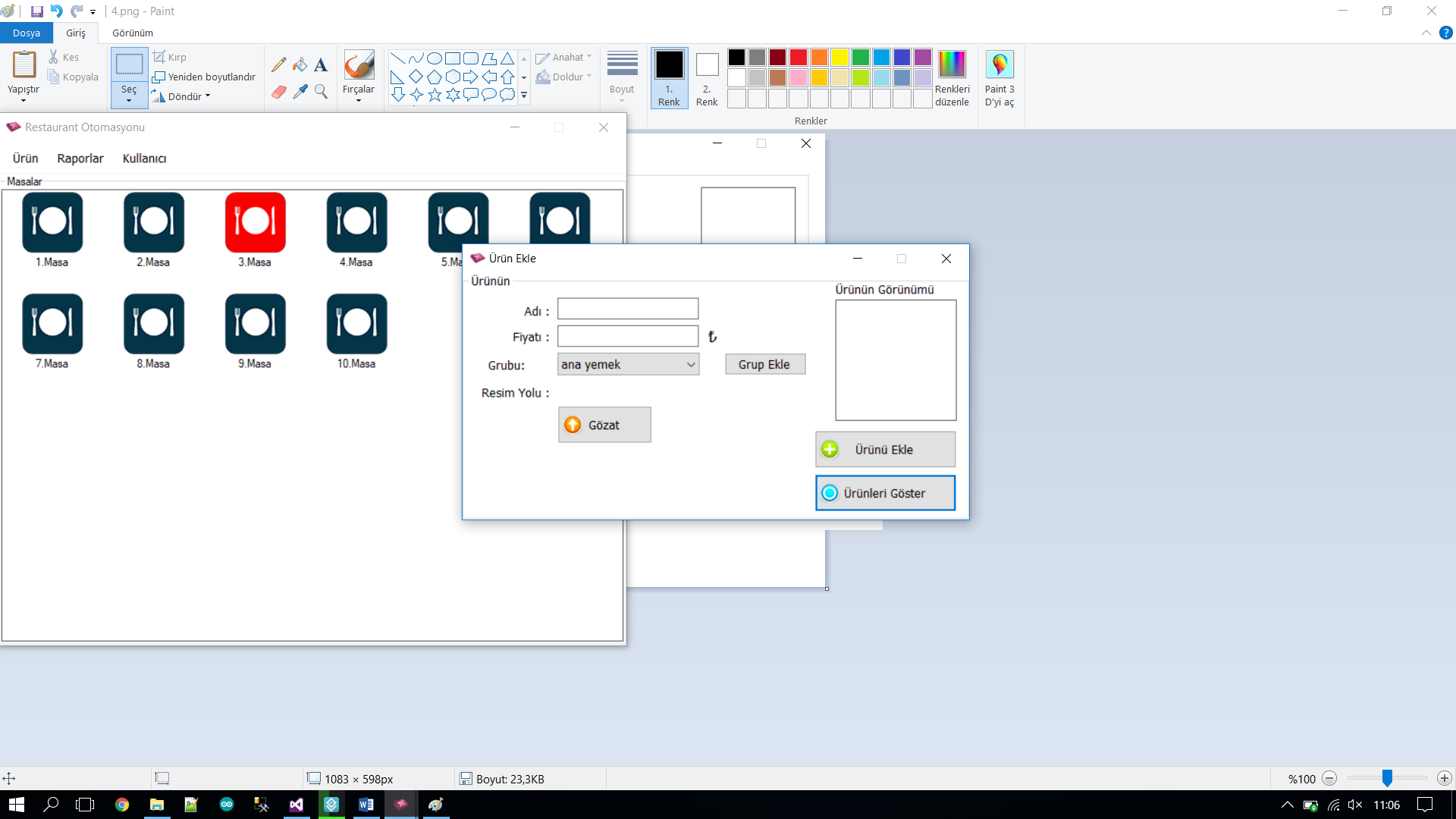
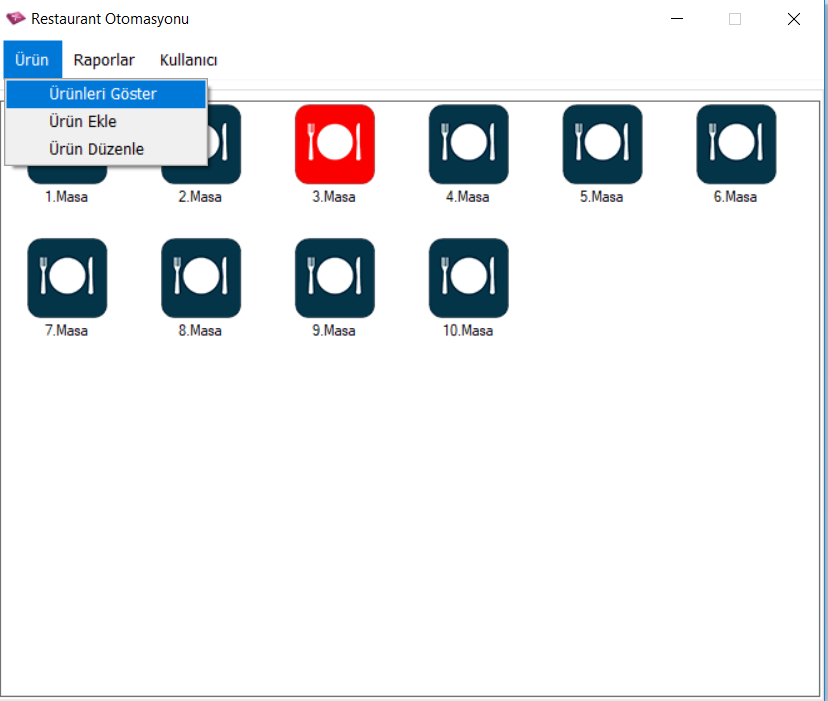
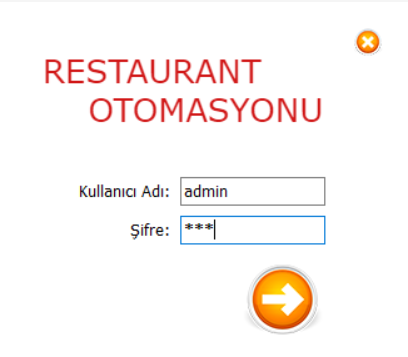
***Yönetici İşlemleri:***

Yöneticiler; sistemin gerekli düzenlemeler yapabilir, duyuru işlemlerini gerçekleştirebilir.

**3.3.1 Yazılım ve Donanım Ara Yüzü**

Otomasyonun görünüşü sade ve anlaşılır olacağından dolayı kullanıcıların ve yöneticilerin kolaylıkla anlayabileceği şekilde ayarlanmıştır

**3.3.2 Kullanıcı Ara Yüzü**



**3.3.4 Kullanıcı ve Yönetici Ara Yüzü**

**Kullanıcı Ara Yüzü:**

Kullanıcıların kullanımı için 2 temel parçadan oluşur. Bunlar:

* Yöneticiler
* Personeller

**Yönetici Ara Yüzü:**

Şuan Aktif Değildir.

**3.4 Belgeleme Gerekleri**

**3.4.1 Geliştirme Sürecinin Belgelenmesi**

Belgelendirme MS Word ortamında raporlandırılacaktır ve bu rapor, proje ile ilgili tüm çalışmaları içerecektir. Proje, şelale modeli gereği belgelendirme projenin her aşamasında raporlanacaktır. Belgelendirme işlemlerini şu ana başlıklarda sıralayabiliriz:

* Proje planı belgelendirilmesi
* Sistem çözümlemesinin belgelendirilmesi
* Sistem tasarımının belgelendirilmesi
* Sistem gerçekleştirilmesinin belgelendirilmesi
* Doğrulama ve geçerlemenin belgelendirilmesi
* Sistem bakımının belgelendirilmesi

**3.4.2 Eğitim Belgeleri**

Eğitim belgeleri için her hangi bir dokümantasyon hazırlanmamıştır.

**3.4.3 Kullanıcı El Kitapları**

Kullanıcı el kitapları yerine gerekli olan bilgiler mesajlarla gösterilecektir.

**4 Sistem Tasarımı**

**4.1 Genel Tasarım Bilgileri**

**4.1.1 Genel Sistem Tanımı**

Yapısal sistem geliştirme metodolojisine göre yapılan çözümlemenin tasarıma dönüştürülmesi, tasarım kavramları, veri tasarımları açıklanmaktadır. Sistem için gerekecek ortak alt sistemler tanıtılmaktadır.

Sistemde kullanılacak veri yapılarının ve veri tabanının yedeklenir ve arşivlenir ortamları ana bilgisayarda saklanacak. Kurulacak sunucular merkezi bilgi sistemi olacaktır. Sistemin tüm bilgi alışı ve akışı bu sunuculardan olacaktır. Sistem verilerinin sunucularda sistematik şekilde saklanması için SQL veri tabanı yönetim sistemi (VTYS) kullanılacaktır.

**4.1.3 Sistem Mimarisi**

Sistem mimarisinde; veri tabanı, kullanıcı ve sunucu katmanları olmak üzere 3 bölümden oluşan mimari kullanılacaktır. Bu mimariyi kullanmanın amacı, bir kullanıcı ve veri tabanı arasında kullanıcının veri isteklerine hizmet etmektir. Yazılımcı için esnek ve yeniden kullanılabilir bir model sunar.

İşlemlerin gerçekleşmesinin birbiri ile ilişkili üç ayrı katmanda yapılması için tasarlanmış bir programlama modelidir.

Bu mimarideki üç katmanı, ara yüz veya sunum katmanı, işleme veya mantık katmanı ve veri veya bilgi katmanı oluşturmaktadır. Kullanıcıların istekleri birinci katman aracılığı ile alınır. Aynı zamanda yapılan işlemlerin kullanıcının anlayacağı bir şekilde sunulması da ara yüz veya sunum katmanında gerçekleşir.

İkinci katman işlemlerin yapıldığı katmandır. Ara yüz katmanından alınan istekler değerlendirilir.

İlgili ve gerekli veriler üçüncü katmandan alınır ve işlenir. Yapılan işlemler sonucu kaydedilen veriler üçüncü katmana kaydedilir ve kullanıcıya sunulması gereken veriler uygun bir şekilde birinci katmana iletilir. Üçüncü katman veri ve bilgilerin depolandığı ve yönetildiği katmandır.



**4.1.4 Dış Arabirimler**

**4.1.4.1 Kullanıcı Arabirimleri**

Kullanıcı arabirimleri arayüzler kısmında anlatıldığı gibi tasarlanmıştır

**4.1.4.2 Veri Arabirimleri**

Yönetici, personel ve kullanıcıların sisteme girdiği ve sistemde kullandığı veriler bir sunucu içerisindedir.

**4.1.4.3 Diğer Sistemlerde Arabirimler**

Otomasyon sisteminin yönetim arayüzü ise diğer bir arabirimi oluşturmaktadır. Yönetim panelinden yetkili olan kişinin sisteme giriş, tüm işlemleri yapabilme ve tüm sorguları çalıştırabilme yetkisi vardır. Yönetici olan kişi ….

**4.1.6 Testler**

Yazılım geliştirme çalışmalarında hem ALFA hem de BETA test aşamalarında ki sistemin sınanması gereken alanlar ortaktır.

**4.1.7 Performans**

Sistemin performansını etkileyen etkenlerin test verileri değerlendirilecektir.

* Sistemin Tasarımına Uygunluk Performansı : Tasarımı yapılan sistemin kararlılığı ve işleyiş performansı değerlendirilecektir.
* Veri Yapısının Sistemle Performansı : Veri yapısının sistemle kararlılığı ve çalışma zamanındaki uyumluluk düzeyindeki performansı değerlendirilecektir.

**4.2 Veri Tasarımı**

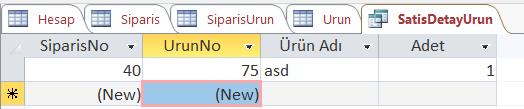
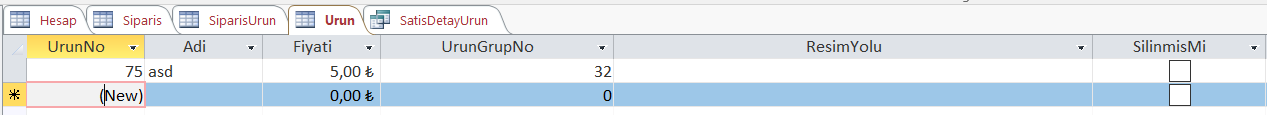
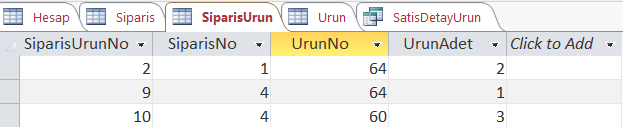
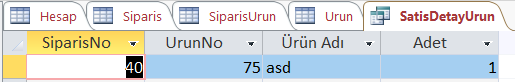
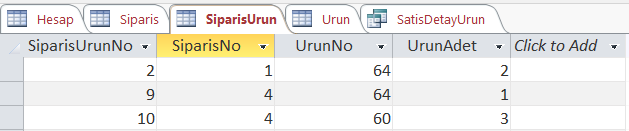
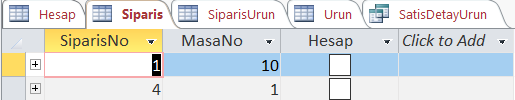
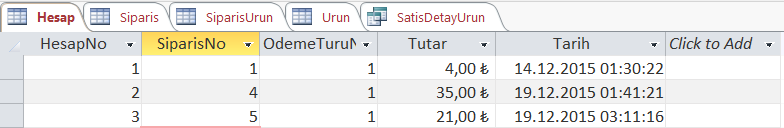
Veri Tasarımında aşağıdaki maddeler dikkate alınarak ele alınmıştır.

* Değişik veri yapılandırılmaları değerlendirilmelidir.
* Kullanılacak veri yapılarını oluşturan birimlerin türleri, sınırları ile birbirleri ile olan ilişkileri bir veri sözlüğü içinde toplanmıştır.
* Veri yapılarının tanımlanmasında döngüsel bir yol izlenmesi daha uygundur.
* Veri yapıları yalnızca kendilerini kullanan modüllere görünür olmalıdır.
* Karmaşık veri yapılarının kullanımı gerekiyorsa kullanılacak program dili, soyut veri türlerini desteleyebilecek şekilde seçilmelidir.

**4.2.1 Tablo Tanımları**

Veri tabanındaki tablolarda kullanılan isimler kullanılmıştır

**4.2.2 Tablo – İlişki Şemaları**



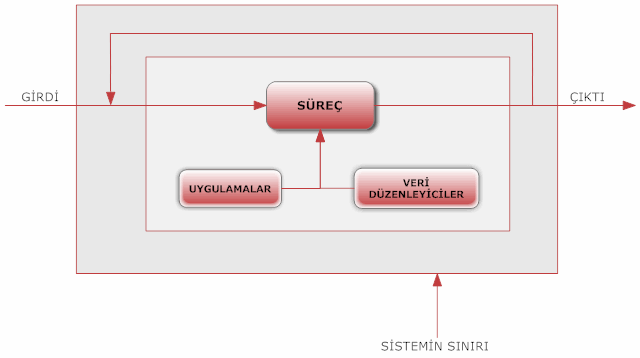
**4.2.3 Veri Tanımları**

Veri tabanındaki tüm sütunlar bağlı oldukları diğer tabloların sütunları ile aralarındaki ilişkiyi, veri tabanı ilişki modeli üzerinden gösterir.

**4.2.4 Değer Kümesi Tanımları**

Yönetici, personel ve kullanıcıların kayıt bilgileri tek bir tabloda tutulur. Giriş yaparken bu kişilerin yetki ayrımının yapılması için yetki numarası verilmelidir. Yönetici – 1, Personel – 2, Kullanıcı – 0 değerlerini almalıdır.

**4.3.1 Genel Tasarım**



**4.3.2 Modüller**

Modüller, tanımlayıcı isimleri olan, tanımlanmış işlevlere sahip asıl sistemi oluşturan birimlerdir. Tasarımı daha başarılı yapmanın diğer yolu da problemi uygun parçalara bölerek yazılımı modüler bir şekilde geliştirmektir.

**4.3.2.1 Yönetici Modülü**

Yönetici modülü sistem işleyişinin kontrol eden kullanıcıdır.

**4.3.2.1.1 İşlev**

Kullanıcı ve yöneticilerin sisteme giriş yaptığı yerdir.

**4.3.2.1.2 Kullanıcı Arabirimi**

Kullanıcı ve yöneticiler için ayrı ayrı giriş sistemimiz bulunacaktır.

**4.3.2.1.3 Modül Tanımı**

Kullanıcılar üye adı ve şifresi ile sisteme giriş yapabilirler.

Yöneticiler panelinden kullanıcı adı ve şifrelerini girerek sisteme giriş yapabilirler.

**4.3.2.2 Kullanıcı Modülü**

Kullanıcı hakkında yapılması gereken değişikliklerin bulunduğu ara yüzdür

**4.3.4 Entegrasyon ve Test Gereksinimleri**

Veri tabanı sisteme bir değişiklik yapılmadan entegre edilecektir. Veri tabanı yük altında sistemle uyumlu ve doğru çalışıyor mu, sistemle veri tabanı uyumluluğu veri tabanı ve sistem arasında ki süre verme tepkisinin hızlılığı veri tabanının sistemdeki bilgilerin okuma doğrululuğu kontrol edilir. Fonksiyonel gereksinimlerin bazıları geliştirilen sistemin diğer mevcut sistemler ile veri alışverişi yapmasını gerektirebilir. Entegrasyon testi hem ALFA, hem BETA aşamalarında gerçekleştirilen bir test alanıdır ve yapılacak olan testler şunlardır;

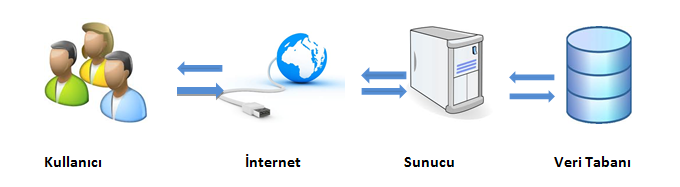
* Sistem veri tabanı uygunluğu
* Veri tabanının sisteme en kısa sürede tepki verme performansı
* Veri tabanından bilgileri okuma doğruluğu
* Veri tabanına sistem bilgilerinin kayıt doğruluğu
* Veri tabanının yük altında sistemle uyumluluk ve doğru çalışma performansı

**4.4.4 Güvenlik Alt Sistemi**

İyi bir yazılım güvenilirlik modelinin sahip olması gereken özellikler aşağıda verilmektedir:

* Gelecekte oluşacak hatalara ait iyi bir tahminlemeye olanak vermesi
* Yararlı nicelikleri hesaplaması
* Basit olması
* Geniş bir uygulama alanının bulunması
* Kabul edilebilir varsayımları desteklemesi
* Parametrelerin kolay hesaplanması

**4.4.5 Veri Dağıtım Alt Sistemi**



**5 Sistem Gerçekleştirimi**

**5.1 Giriş**

Proje geliştirme sürecinde gerçekleştirim bölümünde tasarım aşamasının planına göre bir yol izlenir. Tasarlanan mimari, ara yüz, veri tabanı bağlantıları ve içerik tasarım süreçleri somut olarak gerçekleştirilmeye başlanır. Gerçekleştirme aşamasında şunlar yapılmalı:

* Geliştirilecek yazılımlar için programlama işleminin gerçekleştirilmesi
* Programların birim ve sistem testlerinin yapılması
* Kullanıcı ve işletim dosyalarında gereken güncellemelerin yapılması
* Uygulama aşaması için geçiş planının hazırlanması

**5.2 Yazılım Geliştirme Ortamı**

**5.2.1 Programlama Dilleri**

Masaüstü uygulamasını yapabilmek için C# programlama dili kullanılmıştır.

**5.2.2 Veri Tabanı Yönetim Sistemleri**

Projenin veri tabanı yönetim sistemi olarak SQL Server Management Studio (SQL) kullanılmıştır.

**5.2.2.1 Veri Tabanı Yönetim Sistemi Kullanımın Ek Yararları**

Aynı veri değişik kişilerin PC’lerinde veya değişik bilgisayarlarda tekrar tekrar tutulmaz. Veri Tekrarı (“Data Redundancy”) azaltılır ya da yok edilir.

**Veri Tutarlılığı:** Aynı verinin değişik yerlerde birkaç kopyasının bulunması “bakım” zorluğu getirir: bir yerde güncellenen bir adres bilgisi başka yerde güncellenmeden kalabilir ve bu durum veri tutarsızlığına (“Data Inconsistency”) yol açar.

**Veri Paylaşımı / Eşzamanlılık:** Veri tabanı yönetim sistemi (VTYS) kullanılmadığı durumlarda veriye sıralı erişim yapılır. Yanı birden çok kullanıcı aynı anda aynı veriye erişemez. Bir VTYS’de ise aynı veri tabanlarına saniyede yüzlerce, binlerce erişim yapılabilir.

**Veri Bütünlüğü:** Bir tablodan bir yayın kaydı silinirse, yayın var olduğu diğer tüm tablolardan silinmelidir.

**Veri Güvenliği:** Verinin isteyerek ya da yanlış kullanım sonucu bozulmasını önlemek için çok sıkı mekanizmalar mevcuttur. Veri tabanına girmek için kullanıcı adı ve şifreyle korumanın yanı sıra kişiler sadece kendilerini ilgilendiren tabloları ya da tablo içinde belirli kolonları görebilirler.

**Veri Bağımsızlığı:** Programcı, kullandığı verilerin yapısı ve organizasyonu ile ilgilenmek durumunda değildir. Veri bağımsızlığı, VTYS’lerinin en temel amaçlarındandır.

**5.2.2.2 Veri Modelleri**

Sistemin veri tabanı oluşturulurken ilişkisel veri modeli baz alınacaktır. Kısaca bahsedecek olursak ilişkisel veri modeli, verilerin tablolarda satır ve sütunlar hâlinde tutulduğu ve yüksek bir veri tutarlılığına sahip veri depolama sistemidir. İlişkisel veri tabanını çeşitli tablolar arasında organize edilmiş verilerden oluşan veri tabanı olarak açıklayabiliriz. Bu farklı tablolar arasındaki veriler, çeşitli anahtarlar vasıtası ile birbirlerine bağlanırlar. İlgili tablolarda, sütunlar arasında bir anahtar sütun yer alır. Bu anahtar sütun aracılığı ile birden çok tablo verileri birbiriyle bağlantı sağlayabilir ve herhangi bir sorgulamada birlikte görüntülenebilir. Örneğin oluşturulacak veri tabanında yayınlar aynı tablonda bir sütunda listelenebilir.

**5.2.2.3 Veri Tabanı Yönetim Sistemi Mimarisi**

VTYS mimarisi üç düzeyden oluşur. Bunlar;

**Dış Seviye**

Veri eklemek-silmek-değiştirmek için dış seviye kullanılır. Kullanıcıya en yakın düzeydir.

Kütüphane otomasyonunda ise bu yüzeyler yayın arama, kullanıcının listeleri, yöneticinin personel ekleme, silme ve düzenlenme gibi işlemler kullanıcılar en yakın düzeydir.

**Kavramsal Seviye**

İç seviye ile dış seviye arasındaki köprüyü oluşturur. Veri tabanı içeriği hakkındaki bütün bilgilerin var olduğu kısımdır. Otomasyonda ise kullanıcıların formlara doldurdukları bilgileri alıp veri tabanına gönderilen bölümdür. Bu işlemi ise JSF yüklemektedir.

**İç Seviye**

Verilerin saklanış biçimi hakkında ve bilgisayar tarafından işlenmesiyle ilgili seviyedir. Verilerin nerede, hangi konumda ve nasıl saklanacağına dair bilgilerin olduğu kısımdır. Bu seviye kullanıcıları ilgilendirmez, kullanıcılar dış seviye ve kavramsal seviye ile ilgilenebilir.

**5.2.2.4 Veri Tabanı Dilleri ve Arabirimleri**

Veri tabanını tasarladıktan sonra Veri Tanımlama Dili(VTD), Saklama Tanımlama Dili (STD), Görüş Tanılama Dili (GTD) ve Veri İşleme Dilleri ayrı ayrı kullanmaktan ziyade hepsini içine alınan SQL kullanılacağız. Böylece veri tabanı sistemi yönetimi karmaşıklıktan uzaklaştırılarak daha sade bir hal

alacaktır.

**5.2.2.6 Veri Tabanı Sistem Ortamı**



**5.2.2.8 Hazır Program Kütüphane Dosyaları**

Otomasyonumuzda, Visual Studio ortamından veri tabanı bağlantılarını sağlayan System.Data.SqlClient kütüphanesini kullanıp veri tabanı bağlantılarını yapmak amacıyla kullanılmıştır.

**5.2.2.9 CASE Araç ve Ortamları**

Yazılım geliştirme ortamları yapmış olduğumuz projenin gerçekleştirilebilme imkanı sunan yazılım editörleridir. Bunlar;

* Visual Studio 2015
* SQL
* Fireworks Vektörel Çizim Programı

**5.3 Kodlama Stili**

Hangi platformda geliştirilirse geliştirilsin yazılımın belirli bir düzende kodlanması, yazılımın yaşam döngüsü açısından önem kazanmaktadır. Etkin kod yazılım stili için kullanılan yöntemler:

• Açıklama Satırları

• Kod Yazım Düzeni

• Anlamlı İsimlendirme

• Yapısal Programlama Yapıları

**5.3.1 Açıklama Satırları**

Sistemin kodlanmasında hemen hemen her kod bloğunda açıklama satırı bırakılmıştır. İlerde sistemin güncelleştirilmesinde, bakımında ya da kodlayan ekibin değişikliğinden projenin yarıda kalmasını ya da baştan kodlamasını önlemek adına disiplinli ve anlaşır bir şekilde açıklama satırları bırakılmıştır. Satırlar anlaşılması kolay ve kısa şekilde yazılmaya dikkat edilmiştir. Bu satırlar kod bloğunun işlevini, eksikliğini, düzeltilen yanlışlar hakkında bilgiler gibi içeriklerle yazılır.

**5.3.2 Kod Biçimlemesi**

Kodlar, güncellemeyi ya da bakımı kolaylaştırmak adına düzenle yazılmıştır. Bir satıra birden fazla kod cümleciği eklemek yerine satır satır yazılmıştır. Operatörler kullanılırken aralarından boşluklar konulmuştur. Kod blokları iç içe girdikçe satır başlarının da içeri doğru kaymasına özen gösterilmiştir.

**5.3.3 Anlamlı İsimlendirme**

Kullanılan tanımlayıcıların (değişken adları, dosya adları, veri tabanı tablo adları, fonksiyon adları, yordam adları gibi) anlamlı olarak isimlendirilmeleri anlaşılabilirliği büyük ölçüde etkilemektedir.

**5.3.4 Yapısal Programlama Yapıları**

Yapılar;

* Ardışıl İşlem Yapıları
* Koşullu İşlem Yapıları
* Döngü Yapıları

Teorik olarak da herhangi bir bilgisayar programının sadece bu yapıların kullanılması ile yazılabileceği kanıtlanmıştır.

**5.4 Program Karmaşıklığı**

Gerçekleştirilen programın karmaşıklığını ölçerken eski ve yol gösterici bir McCabe ölçütünden yararlanacağız. Bu ölçüt iki adımdan oluşur bunlar:

• Programın çizge biçiminde gösterilmesi

• McCabe karmaşıklık ölçütü hesaplama

**5.5 Olağan Dışı Durum Çözümleme**

Gerçekleştirdiğimiz sistem kullanıcı dostu yaklaşımından dolayı olağan dışı durumlarda olaya anında müdahale edip kullanıcıyı zor durumda bırakmamak için önlem alır. Bu önlemler if – else yapıları ya da try – catch blokları ile sağlanmaktadır.

**5.5.2 Farklı Olağan Dışı Durum Çözümleme Yaklaşımları**

Modül testlerinin başarılı geçmesi durumunda:

• Modül içindeki değişken ya da fonksiyonların ayrı ayrı doğruluk testlerine tabi tutulması ön görülmektedir.

**5.6 Kod Gözden Geçirme**

**5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi**

Gözden geçirme sürecinde şu adımlar takip edilir:

• Hatalar bulunacak ancak düzeltilmeyecektir.

• Kalite çalışmalarının bir parçası olarak ele alınacak ve sonuçlar düzenli ve belirlenen bir biçimde saklanacak.

**6 Doğrulama ve Geçerleme**

**6.1 Giriş**

Geliştirdiğimiz yazılımının doğrulanması ve geçerlenmesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler:

* Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.
* Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.
* Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.
* Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklenmesi için sınamaların hazırlanıp yürütülmesi.

**6.2 Sınama Kavramları**

Sınama ve bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerin seçimi gerekmektedir. Bütünleştirme işleminde, en küçük birimlerden başlanarak sistem düzeyine çıkılmaktadır. Bu değişik düzeylere hitap edecek sınama yöntemleri olmalıdır

**Birim Sınama:** Bağlı oldukları diğer sistem unsurlarından bütünüyle soyutlanmış olarak birimlerin, doğru çalışmalarının belirlenmesi amacıyla yapılır. Birimler, ilişkili yapıtaşlarının bütünleştirilmesinden oluşurlar.

**Alt Sistem Sınama:** Alt sistemleri ise, modüllerin bütünleştirilmesi ile ortaya çıkar. Yine bağımsız olarak sınamaları yapılmalıdır. Bu düzeyde en çok hata, ara yüzlerde bulunmaktadır, ara yüz hatalarına yönelik sınamalara yoğunlaşılmalıdır.

**Sistem Sınaması:** Üst düzeyde bileşenlerin sistem ile olan etkileşimlerinde çıkacak hatalar aranmaktadır. Ayrıca belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

**Kabul Sınama:** Çalıştırılmadan önce sistemin son sınamasıdır. Artık yapay veri yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması ve beta sınaması olarak da bilinir. Alfa sınamada, sınamanın geliştirici organizasyonun yerleşkesinde, sistemi kullananların da gelerek katkıda bulunması içerilir. Daha sonra ürünün pazarlama işlemi sırasında beta sınama denilen, sınama, sistemi kullananın kendi yerleşkesinde, geliştirici gözetiminde yapılır.

**6.3 Doğrulama ve Geçerleme Yaşam Döngüsü**

**6.4 Sınama Yöntemleri**

**6.4.1 Beyaz Kutu Sınaması**

Beyaz kutu sınamasında kullandığımız temel sınama yöntemleri şunlardır;

* Bütün bağımsız yollar en azından bir kere sınandı
* Bütün mantıksal karar noktalarında 2 değişik karar için sınamalar yapıldı
* Bütün döngülerin sınır değerlerinde sınandı
* İç veri yapıları denendi

**6.4.2 Temel Yollar Sınaması**

Çevrimsel karmaşıklık sayısı E – N + 2 formülüne göre hesaplanır.

E : Toplam dal sayısı N : Toplam düğüm sayısı

**6.5 Sınama ve Bütünleştirme Stratejileri**

**6.5.1 Yukarıdan Aşağıya Sınama ve Bütünleştirme**

En üst düzey sistem modellerinin sınanmasından sonra yukarıdan aşağıya doğru olan program akışındaki modüllerin eklenerek sınanmasıdır.

Yönetici, Personel ve Kullanıcı modüllerinde sistemin en üst düzey modüllerinin alt modüllere ayrılarak sınanacaktır.

Raporlar, Kalite Yönetim Birimine sunularak değerlendirilecek. Sistemin işleyiş ya da yapısı ile ilgili sorunlar olması dâhilinde sistem tasarımcısı ve sistem çözümleyicisi tarafından yeniden değerlendirilecektir.

**6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme**

En alt düzey sistem modellerinin sınanmasından sonra aşağıdan yukarıya doğru olan program akışındaki modüllerin eklenerek sınanmasıdır. En sonunda tüm sistemin sınanmasıdır.

Raporlar, Kalite Yönetim Birimine sunularak değerlendirilecektir. Sistemin işleyiş ya da yapısı ile ilgili sorunlar olması dahilinde sistem tasarımcısı ve sistem çözümleyicisi tarafından yeniden değerlendirilecektir. Sistem modüllerinin sorunları çözümlemesi durumunda en alt modüller eklenerek, bağlı oldukları modüller oluşturulacak. Oluşturulan bu modüllerle sisteme bağlı olacak modüller aralarında ilişkisel bir sınamaya tabi tutulacak. Sistemdeki bu modüllerin uyumlu ve kararlı çalışmaları durumunda sisteme eklenerek, sistemin genel yapısı oluşturulacaktır.

**6.6 Sınama Planlaması**

Sınama işlemi çok kapsamlıdır. Bir plan güdümünde gerçekleştirilmelidir. Yazılım yaşam döngüsünün süreçlerine koşut olarak, farklı ayrıntı düzeylerinde birden fazla sınama planı hazırlanır.

**6.7 Sınama Belirtimleri**

Sistemin sınama işlemleri aşağıdaki ayrıntıları içerir:

• Sınanan program modülü ya da modüllerinin adları

• Sınama türü, stratejisi (beyaz kutu, temel yollar vb.)

• Sınama verileri,

• Sınama senaryoları

Sınama işlemi sonrasında bu belirtimlere:

• Sınamayı yapan

• Sınama tarihi

• Bulunan hatalar ve açıklamaları

Bilgileri eklenerek sınama raporları oluşturulur.

**6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

**Planlama**

Planlama sırasında genel sınama planı oluşturulur. Söz konusu plan tüm sınama etkinliklerini çok genel hatlarıyla tanımlar ve sınama planlamasında verilen bilgileri içerir.

**Çözümleme**

Çözümleme aşamasında, sistemler ve alt sistemler ortaya çıkarılır ve sınama planı alt sistemler bazında ayrıntılandırılır.

Sınama sırasında bulunan her hata için, değişiklik kontrol sistemine (DKS), "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

• **Onulmaz Hatalar:** Projenin gidişini bir ya da birden fazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.

• **Büyük Hatalar:** Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.

• **Küçük Hatalar:** Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır.

• **Şekilsel Hatalar:** Heceleme hatası gibi önemsiz hatalardır.

**7 Bakım**

**7.1 Giris**

Sınama işlemleri bitirilen yazılımın sunucuya yüklenmesi ve sistemin ayarlamalarının yapılması gerekmektedir. Yazılım kullanıma geçtikten sonra, yaşam döngüsünün en önemli ve hiç bitmeyecek aşaması olan "bakım" aşaması başlayacaktır. Hata giderme ve yeni özellikler yapma aşaması teslimden sonra başlayacaktır. Bu bölümde yapılan açıklamalarda IEEE 1219-1998 standardı temel alınmıştır.

**7.2 Kurulum**

Otomasyon masaüstü bilgisayarlar için yapıldığından otomasyonun kurulumu son derece basittir. Yapılması gereken tek şey .exe uzantılı dosyayı çalıştırılmasıdır.

**7.3 Yerinde Destek Organizasyonu**

Bu uygulama için otomasyonun kullanıldığı yerde destek ekibi bulundurulmasına gerek yoktur. Oluşacak olan sorun veya taleplerde geri bildirim yapılarak sorunun çözüme ulaşması sağlanılır.

**7.4 Yazılım Bakımı**

Yazılımın yaşamına devam edebilmesi için gerekli değişikliklerin uygulanmasıdır. Genel hatlarıyla 3 bakım türü vardır:

***Düzeltici Bakım:*** Tespit edilen hataların giderilmesi işlemidir. Kodlama hatalarını düzeltmek genelde az maliyetlidir. Tasarımdan kaynaklı hataların giderilmesi ise bazı sistem bileşenlerinin baştan yazılmasını vb. gerektirebilir ve nispeten yüksek maliyetlidir.

***Uyarlayıcı Bakım:*** Yazılımın yeni bir çalışma ortamına uyarlanmasıdır. Bu bir donanım platformu değişikliği olabileceği gibi (32 bit’ten 64 bit’e geçiş gibi) farklı bir işletim sistemine uyarlama şeklinde de olabilir. Ayrıca, veri tabanı sistemi değişikliği de bu türden bir bakım olarak görülebilir (MS SQL Server bağımlı kodların Oracle'a uyarlanması gibi).

***İyileştirici Bakım:*** Sisteme yeni işlev ve özelliklerin eklenmesi, performansın arttırılması gibi bakım çalışmalarıdır.

**7.4.1 Tanım**

Bakım, işletime alınan yazılımın sağlıklı olarak çalışması ve ayakta tutulabilmesi için yapılması gereken çalışmalar bütünü olarak tanımlayabiliriz. Bakım, yazılım yaşam döngüsünün en önemli ve en maliyetli aşamalarından biridir. Bakım süreci, yazılım yaşam döngüsünde "buzdağının görünmeyen kısmı" olarak adlandırılır. Bakım maliyetleri, zaman zaman üretim maliyetlerinin %60'ını geçer. IEEE 1219 standardı temel alınarak yazılım bakımıyla ilgili kavramları açıklamak gerekir.

**7.4.2 Bakım Süreç Modeli**

IEEE 1219 standardı tarafından önerilen bakım süreç modeli aşağıdaki şekilde belirtilen şablonu kullanarak bakım süreçlerini tanımlamaktadır.



IIEEE 1219 standardı tarafından önerilen bakım süreç modeli;

Sorun Tanımlama/Sınıflandırma

Çözümleme

Tasarım

Gerçekleştirim

Sistem Sınama

Kabul Sınama

Kurulum

Yukarıda belirtilen maddelerin sırası şeklinde gelişen süreç çerçevesinde bir yazılımın bakım süreci gerçekleştirilmelidir.

**8 Sonuç**

Restaurant otomasyonunda işlerin kolaylaşması ve aynı zamanda kullanım kolaylığı ile anlaşılır hale getirilmiştir.