**A)PROJENİN TANIMI**

Proje işyerine personelleri giriş çıkışları takip etmek için kullanılacak bir yüz tanıma sistemidir.

Projenin amacı, personellerin işe giriş ve çıkış saatlerini takip etmektir. Proje ile daha kısa sürede maaş hesaplayabilir ve hatalı bilgi girişinden dolayı oluşabilecek yasal riskleri azaltabilirsiniz. İhtiyaç olması durumunda; resmî kurumlara beyan edebilmek amacıyla, geriye dönük verilere ulaşabilir ve puantaj sisteminizin takibini kolaylıkla yapabilirsiniz. Merkezi süreç yönetimi sayesinde, İnsan Kaynakları ve Muhasebe programları ile kolay entegrasyon sağlayabilir ve ihtiyaçlarınıza yönelik yeni raporlar alabilirsiniz.

**B)PROJENİN SAHİP OLDUĞU NİTELİKLER**

* Firma girişinde bir yüz tanıma cihacı ile personellerin yüzleri okutulacak ve uygun eşleme bulunduğunda onay verilecektir. Personel bu işlem ile giriş/çıkış işlemini yapmış olacaktır. Yüz tanıma sistemi nedeniyle başka biri için giriş çıkış işlemi yapılamayacağından doğru sonuçlar elde edilecektir. RFID kart sistemlerinde başkasının kartını okutarak kaçak giriş çıkış yapılabilir.
* Personellerin hangi gün hangi saatler arasında işyerinde olduğu rahatlıkla tespit edilebilecektir.
* İşe gelmeyenler, erken çıkanlar, geç gelenler ve izinli personeller rapor halinde çıktı alınabilecektir.
* Fazla mesai hesaplaması yapılabilecektir.
* Bu kontroller web tabanlı olarak yerel ağ üzerinden yapılacaktır.
* İsteğe göre bir domain üzerinden dışarıdan da kontrol edilebilecektir.

## C.YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ

Yazılımın ürününün hem üretim hem de müşterideki kullanım süreci boyunca geçirdiği tüm aşamalar yazılım geliştirme yaşam döngüsü (“software development life cycle”, “SDLC”) olarak adlandırılır. Yazılım geliştirme süreci, zamanlamaya dayalı ve içerik olarak bölünmüş aşamalardan oluşmaktadır. Bu sayede yazılım planlı bir şekilde geliştirilmektedir. Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar sürekli bir döngü biçiminde ele alınır. Döngü içerisinde her hangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur. Temel yazılım geliştirme aşamaları aşağıdaki gibidir:

**Planlama:** Yazılım yaşam döngüsünün ilk aşamasıdır. Temel ihtiyaçlar belirlenir, proje için fizibilite çalışmaları yapılır (maliyetlerin ve sistemin yararlarının tanımlanması) ve proje planlaması gerçekleştirilir.

**Analiz:** Bu aşamanın amacı sistemin işlevlerini ve kesin gereksinimleri açıklığa kavuşturmak ve sonucunda bunları belirli bir formatta dokümante etmektir. Bu çalışma müşteri, yazılım mühendisi, sistem analisti, iş analisti, ürün yöneticisi vb. rollerin bir araya geldiği gruplar tarafından yapılabilir. İhtiyaçların net olmadığı durumlarda yazılım mühendisi ve müşteri arasında iletişim ve birlikte çalışmanın çok daha fazla olması gerekir. Çeşitli yazılım geliştirme metodolojilerinde bu aşamada kullanım dokümanları ve test plan dokümanları da oluşturulabilir.

**Tasarım:** Gereksinimlerin tamamlanmasıyla beraber sistem tasarım aşamasına başlanır. Yazılım ürün tasarımı, müşterinin gereksinim ve isteklerini karşılamak üzere yazılım ürününün özellikleri, yetenekleri, ve arayüzlerinin belirlenmesi etkinliğidir. İki tür tasarımdan bahsetmek mümkündür (Yüksek düzeyde tasarım — Mimari tasarım ve Detaylı tasarım). Mimari tasarım, yazılım modüllerinin genel yapıları ve organizasyon içerisindeki etkileşimleri ile ilgilenir. Sonucunda mimari tasarım dokümanları oluşturulur. Detaylı tasarım aşamasında Mimari tasarım dokümanları genelde revize edilirler. Tasarım ve analiz aşamalarının ayrımı “Problem Ne?/Problem NasılÇözülür?” sorularının kullanımı ile ilgilidir. Gereksinimlerin belirlendiği analiz aşaması problemin ne olduğu ile ilgilidir. Unutmamak gerekir ki sistemdeki tüm problemler yazılım ürününün tamamlanması ile çözülmeyecektir. Maalesef çoğu zaman Ne söylemi tasarım kararı olurken Nasıl söylemi de müşterinin gereksinimi olabilmektedir. Bu duruma dikkat etmek gerekir.

Yazılım tasarımında kullanılan en önemli tekniklerden birisi Soyutlama (Abstraction) dır. Soyutlama, problemlerin çözümlerini kolaylaştırmak için nesnelerin, olayların ve durumların bazı özelliklerin görmezden gelinmesidir. Problemi basitleştirerek en önemli kısımlarına odaklanmamızı sağlar. Modelleme is temel tasarım aracı olup statik ve dinamik modellerden bahsetmekten mümkündür. Statik model, programın çalışması sırasında değişmeyen yönlerini ifade etmek için kullanılırken (sınıf ve nesne modelleri), dinamik model programın çalışması sırasındaki işleyişi ifade etmek için kullanılır ( durum ve sıra diyagramları).

**Gerçekleştirim (Kodlama ve Test)**

Tasarım aşamasının belirli bir olgunluğa ulaşmasıyla birlikte Kodlama aşaması başlar. Müşteriye teslim edilecek ürünü programlama aşamasıdır. Kaliteli kodlama üzerine önümüzdeki yazılarda bayağı kafa yoracağız. Şimdilik kısaca değinelim. İyi kod, okunabilirliği ve bakımı kolay olan basit koddur. KISS (Keep it simple) prensibine göre yeni mezun olmuş birisine kodunuzu verdiğinde 1–2 gün içerisinde anlayabiliyor ve değiştirebiliyorsa kodunuz iyi bir koddur. İster bir şirkette çalışın ister bireysel projeler geliştirin mutlaka belirli bir kodlama kalite standardınız olsun (İsimlendirme standartları, yorum satırı kullanımları, tekrar eden kodlar, dev –if koşul blokları, aşırı benzer işlevler, uzun metotlar vb.)

Kodlama süresince ve kodlama sonrasında yapılan diğer önemli aşama test’tir. Erken test et yaklaşımı ile (early testing) hareket edip, analiz aşamasından itibaren test bakış açısına sahip olmamız hata yapma oranımızı ve maliyetleri (zaman, para, prestij vb.) düşürecektir. Birim testleri, duman testleri, yanlış değer testleri, kabul testleri, kullanım senaryo testleri, yük testleri, kullanıcı kabul testi, yoldan geçen adam testi, test otomasyonu gibi sürece ve duruma göre uygulanabilecek çok farklı kategoride ve derinlikte test türü bulunmaktadır.

**Teslim ve Bakım**

Tüm test aşamaları tamamlandıktan sonra yazılım ürünün sahaya teslim edilebilir bir versiyonu çıkartılır ve teslim aşaması gerçekleştirilir. Teslim çıktısı olarak ürün tek başına yeterli değildir. Mutlaka son kullanıcılar için kullanım kılavuzu ve versiyon fark dokümanı oluşturulmalıdır. Teslim ile birlikte bakım aşaması da başlar. Hata giderici, önleyici, altyapıyı iyileştirici, ürüne yeni özellikler ekletici gibi farklı bakım faaliyetleri mevcuttur.

Yaşam döngüsünün temel adımları çekirdek süreçler (core processes) olarak da adlandırılır. Bu süreçlerin gerçekleştirilmesi amacıyla Yazılım Belirtim Yöntemleri ve Yazılım Süreç Modelleri kullanılmaktadır.

**C.1.Yazılım Süreç Modelleri (Processes)**

* Kodla ve Düzelt (Code and Fix)
* Şelale Modeli (Waterfall Model)
* V Modeli (V-shaped Model)
* Evrimsel Geliştirme (Evolutionary Development)
* Prototipleme (Prototyping)
* Spiral Model
* Formal Sistem Geliştirme (Formal System Development)
* Yeniden kullanıma yönelik geliştirme (Re-use based development)
* Artımlı Geliştirme (Incremental Development)
* Birleşik Süreç (Unified Process)
* Çevik Modeller (Agile models: XP, Scrum)

Projemizde Evrimsel Geliştirme (Evolutionary Development) kullanılacaktır.

**Evrimsel Geliştirme (Evolutionary Development):**

Bu model ilk tam ölçekli olan modeldir. Projenin başarıya ulaşması ilk evrimin başarısına bağlıdır. “Ne istediğimi bilmiyorum ama görsem tanırım.” mantığına dayanır. Gereksinimi anlamayı basitleştirir. Risk ve hata oranı azdır. Zayıf yanıysa değişiklik denetimine sahip olmamasıdır. Bu sebeple bakımı zordur ve maliyetlidir. Müşteri geri bildirimleriyle ilerlediği için yavaştır. Düzenli bir ürün oluşumu yoktur. Kısa süreli sistemlerde ya da büyük bir sistemin parçaları (ör. Kullanıcı Arayüzleri..) için kullanılabilirler. İki çeşit evrimsel geliştirme vardır: keşifçi geliştirme (exploratory development) ve atılacak prototipleme (throw-away prototyping).

* Her aşamada üretilen test ürünlerin tam işlevselliği için
* Kurumlarda kullanılıp test edilir ve sonra güncellenir.
* Belli aralıklarla güncelleme yapılır.
* Birden fazla kuruma geliştirme yapılacağı için bu model seçilmiştir.

Evrimsel süreç modeli aşağıdaki gibi görselleştirilebilinir.

**C.2.YOURDAN YAPISAL SİSTEM METODOLOJİSİ**

Birbirleriyle uyumlu, yazılım yaşam döngüsü boyunca kullanılacak olan yöntemler bütününe metodoloji denir.

Kolay uygulanabilir olması, günümüzde yaygın olarak kullanılması ve birçok case aracı tarafından destekleniyor olması sebebiyle projemizde yourdan yapısal sistem metodolojisinin kullanılmasına karar verilmiştir.

Yourdan metodolojisi ile ilgili aşağıdaki tablodan detaylı bilgi edinilebilir.



**D.YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ TEMEL ADIMLARI**

**D.1.PLANLAMA**

Planlama, yazılım geliştirme yaşam döngüsünün ilk adımıdır. Bu adım projenin başarılı bir geliştirme sürecini tamamlayabilmesi için çok önemlidir. Tüm proje boyunca kullanılacak ve güncellenecek olması ne kadar önemli olduğunun göstergesidir.

Bu aşamada yapılan çalışmalar;

* Proje kapsamının belirlenmesi
* Proje kaynaklarının belirlenmesi
* Proje maliyetlerinin kestirilmesi
* Proje ekip yapısının oluşturulması
* Ayrıntılı proje planı yapılması
* Projenin izlenmesi

**D.1.1.PROJE KAPSAMININ BELİRLENMESİ**

Bu aşamada ilk olarak piyasadaki mevcut sistemler incelenmiş, eksik ya da yetersiz olan işlevler tespit edilmiştir.

Projenin tüm kurumlar için uyarlanabilir olması için farklı sektörlerdeki ihtiyaçlar ortak payda altında toplanmıştır.

Mevcut müşteri ile görüşülerek talep edilenler dinlenmiş ve müşteri tarafından oluşturulan şartname incelenmiştir.

**D.1.2.PROJE KAYNAKLARININ BELİRLENMESİ**

1. İnsan kaynakları
2. Donanım kaynakları
3. Yazılım kaynakları

**D.1.2.1.İNSAN KAYNAKLARI**

Proje Yöneticisi : Hakan SANDUK

Yazılım Ekip Lideri : Ali KORKMAZ

Front-End Geliştirici : Ahmet Çilek

Back-End Geliştirici : Muhammet ÇALIŞKAN

Veritabanı Tasarımı : Pelin ÇAKMAK

Elektronik Personeli : Mehmet ERDEM

Test Personeli : Nurgül AKMAN

**D.1.2.2. DONANIM KAYNAKLARI**

* Sunucu bilgisayar
* Yüz tanıma cihazı
* Yerel ağ alt yapısı

**D.1.2.3.YAZILIM KAYNAKLARI**

* Back-End için Visual Studio 2018 tercih edilmiştir.Asp.Net MVC Core 2 teknolojisi kullanılacaktır.
* Front-End için Visual Studio Code tercih edilmiştir.Html,Bootstrap ve Angular.Js kullanarak geliştirme yapılacaktır.
* Veri tabanı yönetim sistemi için Microsoft Sql Server 2016 tercih edilmiştir.
* Cihazlar arası iletişim TCP protokolü üzerinden sağlanacaktır.(Yüz tanıma cihazından veri alma ve gönderme)
* Versiyon kontrol sistemi için Git teknolojisi tercih edilmiştir.

**D.1.3.PROJE MALİYETİNİN HESAPLANMASI**

Yazılım projelerinde maliyet tahmini yapmak projenin başlangıcı içi en önemli adımlardan biridir. Maliyet tahmini, bir yazılım için gerekebilecek iş gücü ve zaman maliyetlerinin üretimden önce belirlenebilmesi için yapılan işlemlerdir. Bu amaçla kullanılan unsurlar; geçmiş projelere ilişkin bilgiler, proje ekibinin deneyimleri, izlenen geliştirme modeli olabilir.

**Maliyet yönetimi sayesinde;**

* Gecikmeler önlenir
* Bilgi sistemi geliştirme süreci kolaylaştırılır
* Daha etkin kaynak kullanımı sağlanır
* İş zaman planı etkin olarak gerçekleştirilir
* Ürün sağlıklı olarak fiyatlandırılır
* Ürün zamanında ve hedeflenen bütçe sınırları içerisinde bitirilir

**Gözlemlenebilecek değerler;**

•Projenin toplam süresi

•Projenin toplam maliyeti

•Projede çalışan eleman sayısı, niteliği, çalışma süresi

•Toplam satır sayısı

•Bir satırın maliyeti (ortalama)

•Bir kişi/ay’ da gerçekleştirilen satır sayısı

•Toplam işlev sayısı

•Toplam işlev maliyeti

•Bir kişi/ay’ da gerçekleştirilen işlev sayısı

•Bir kişi/ay’ da maliyeti

**İşlev Noktaları Yöntemi**

İşlev noktaları, geliştirmenin erken aşamalarında (çözümleme aşamasında) saptanan bir değerdir. Bu değer, sistemin oluşturulduğu uygulama geliştirme ortamından bağımsız olarak elde edilmektedir. İşlev noktalarının hesaplamasında problem tanımı girdi olarak alınarak üç temel adım izlenir:

1. Problemin bilgi ortamının incelenmesi
2. Problemin teknik karmaşıklığının incelenmesi
3. İşlev noktası hesaplama

Özellikle, çözümleme çalışması, yapısal yöntemlerle yapıldığında, işlev noktaları yarı otomatik bir biçimde kolayca elde edilebilir. İşlev noktaları yönteminin proje geliştirme sürecinin erken aşamalarında oldukça etkili olduğu belirtilmektedir

**1.Problemin bilgi ortamının incelenmesi;**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ölçüm Parametresi** | **Sayı** |  | **Ağırlık Faktörü** | | |  |  |
| Yalın | Karmaşık | Ortalama |
| Kullanıcı Girdi Sayısı | 15 | x | 3 | 4 | **6** | = | 80 |
| Kullanıcı Çıktı Sayısı | 10 | x | 4 | 5 | **7** | = | 70 |
| Kullanıcı Sorgu Sayısı | 20 | x | 3 | **4** | 6 | = | 80 |
| Kütük Sayısı | 4 | x | **7** | 10 | 15 | = | 28 |
| Dışsal Arayüz Sayısı | 0 | x | **5** | 7 | 10 | = | 0 |
| **Toplam Sayı** | | | | | | = | 258 |

**Kullanıcı girdileri:** Yazılıma girdi olarak verilen her farklı uygulama bileşeni bir kullanıcı girdisi olarak sayılır. Kullanıcı girdilerinin sayısı hesaplanırken, alan bazında değil, daha genel olarak mantıksal kayıt bazında uygulama yapılmalıdır.

**Kullanıcı çıktıları:** Kullanıcıyı ilgilendiren her tür mantıksal çıktı, kullanıcı çıktısı olarak sayılmalıdır. Bu kapsamda kullanıcı çıktılarına örnek olarak, raporlar, ekran çıktıları, hata iletileri vb. verilebilir.

**Kullanıcı sorguları:** Çevrim içi olarak bilgisayara verilen bir girdi sonucu yine çevrim içi olarak bir kullanıcı çıktısı alınması biçiminde tanımlanmaktadır. Her farklı sorgu ayrı olarak sayılmalıdır.

**Kütükler:** Her mantıksal bilgi yığını ya da kütük ayrı olarak sayılmalıdır.

**Dışsal arayüzler:** Geliştirilmesi öngörülen bilgi sisteminin, gerek kurum içinde gerekse kurum dışında bir başka bilgi sistemi ile bilgisayar ortamında bir iletişimi (çevrim dışı disket vb. ya da çevrim-içi doğrudan bağlantı) söz konusu ise bu durum bir dışsal arayüz olarak sayılmalıdır.

Bu bilgiler ışığında AİN (Ayarlanmamış İşlev Nokta sayısı)= 258 olarak hesaplanmıştır.

**2.Problemin teknik karmaşıklığının incelenmesi;**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TFK Soruları** | **Hiç etkisi yok(0)** | **Çok Az etkisi var(1)** | **Etkisi var(2)** | **Ortalama etkisi var(3)** | **Önemli etkisi var(4)** | **Mutlaka olmalı, kaçınılmaz(5)** |
| Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu? |  |  |  |  |  | X |
| Veri iletişimi gerekiyor mu? |  |  |  |  |  | X |
| Dağıtık işlem işlevleri var mı? |  |  | X |  |  |  |
| Performans kritik mi? |  |  |  |  | X |  |
| Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak? |  | X |  |  |  |  |
| Sistem, çevrim içi veri girişi gerektiriyor mu? |  |  |  |  |  | X |
| Ana kütükler çevrim-içi olarak mı günleniyor? |  |  | X |  |  |  |
| Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı? |  | X |  |  |  |  |
| İçsel işlemler karmaşık mı? |  | X |  |  |  |  |
| Tasarlanacak kod, yeniden kullanılabilir mi olacak? |  |  |  |  |  | X |
| Dönüştürme ve kurulum tasarımda dikkate alınacak mı? |  |  |  |  | X |  |
| Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor? |  |  |  |  |  | X |
| Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak? |  |  |  |  |  | X |

Buna göre TFK (Teknik Karmaşıklık Faktörü) = 40 olarak hesaplanmıştır.

3.İşlev Noktası Hesaplama

Bu son adımda, bilinen ve sık kullanılan deneysel formül kanalıyla ilk iki adım sonucundaki bulgular kullanılarak işlev nokta sayısı belirlenir.

Gerek AİN, gerekse TKF önceki bölümlerde açıklandığı gibi hesaplandıktan sonra, geliştirilecek bilgi sistemine ilişkin İşlev Nokta Sayısı (İN) aşağıdaki formül kullanılarak hesaplanır.

İN(İşlev Nokta sayısı)=AİN x (0.65x0.01xTKF)

Üretkenlik=İN/Kişi-Ay

Kalite=Hatalar/İN

Maliyet=Fiyat/FP

İN=258\*(0.65\*0.01\*40)

İN=67.08

İN≈ 68 olarak hesaplanmıştır.

Buna göre;

|  |  |
| --- | --- |
| **Programlama Platformu** | **Satır Sayısı/İN(Ortalama)** |
| Assembly Dili | 300 |
| COBOL | 100 |
| FORTRAN | 100 |
| Pascal | 90 |
| C | 90 |
| Ada | 70 |
| Nesne Kökenli Diller | 30 |
| 4. Kuşak Dilleri | 20 |
| Kod Üreticiler | 15 |

İN değeri 68 bulunduktan sonra kullanılan dil grubuna göre yaklaşık satır sayısı hesaplanmıştır. Kullanılan programlama platformu nesne kökenli olduğu için yaklaşık satır sayısı kestirimi;

Satır Sayısı =68\*30=2040 olacaktır.

Buna göre;

* İş Gücü (K) K=a\*Sb
* Zaman (T) T=c\*Kd

a,b,c,d : her bir model için farklı katsayılar

S : bin türünden satır sayısı

İş gücü ve zaman değerleri aşağıdaki gibi hesaplanmıştır;

Ayrıca COCOMO Modeli, farklı türdeki yazılım projelerine farklı biçimde uygulanır.Projeler; Ayrık Projeler, Yarı-Gömülü Projeler, Gömülü Projeler olarak sınıflandırılır. Projemiz küçük boyutlu ve deneyimli personeller tarafından gerçekleştirileceği için ayrık proje olarak değerlendirilirmiştir.

Sıramatik sisteminin satır sayısı 1000 ile birkaç 10.000 satır aralığında olduğundan ayrık proje olarak değerlendirilmiştir. Bunun yanında Yazılımın zaman ve iş-gücü maliyet kestiriminde proje ekibinin özelikleri , projenin geliştirilmesinde kullanılacak araç , yöntem ve ortam dikkate alındığından model olarak Ara Model seçilmiştir.

**Ara Model 3 aşamadan meydana gelir:**

1. Aşama: İş gücünü Hesaplama
2. Maliyet Çarpanını Hesaplama
3. İlk İş gücü değerini Düzeltme

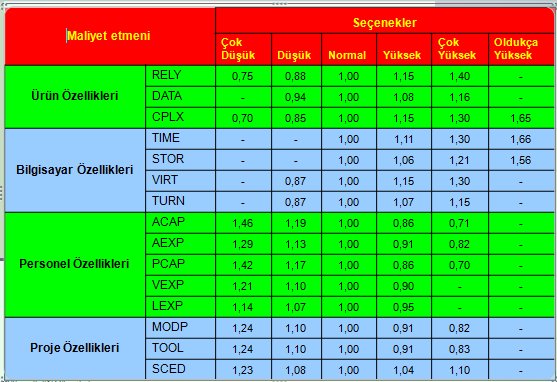
**İş gücü** =K= 3,2\* S1,05 (S = bin türünden satır sayısı)

K=3,2\*S1,05

**Zaman** =T= 2.5 \* K0.38 (K= Düzeltilmiş iş gücü T= zaman)

**Maliyet Çarpanı**=C= C1\*C2\*C3\*C4….

**Düzeltilmiş İş gücü** = K\*C (C = maliyet çarpanı)



Buna göre projenin satır sayısı ortalama olarak *46*00 satır olacağı belirlenmiş ve maliyet kestirimi için ;

K= 3,2 \* 4,61,05

K= 15,89

C=1,15\*1,16\*1\*1,11\*1\*1,07\*1,46\*1,29\*0,86\*1,21\*1,14\*1,24\*0,83\*1

C=3.64

Düzeltilmiş İş Gücü= 15,89\*3.64

Düzeltilmiş İş Gücü=57.84

T= 2.5 \* 57.840.38

T=11.68

T≈12 ay (yaklaşık değer)

Projenin tamamlanma süresi 12 ay çıkmış olup, bir aylık giderin 10.000 dolar olduğu göz önünde bulundurulursa toplam proje maliyeti 120.000 dolar olarak hesaplanmıştır.

**D.1.5PROJE İŞ ZAMAN PLANININ OLUŞTURULMASI**

* **Proje planlama aşaması (01.07.2019-01.08.2019)**

Proje kapsamının belirlenmesi, proje kaynaklarının belirlenmesi, proje ekip yapısının oluşturulması, proje maliyetlerinin kestirilmesi, ayrıntılı proje planının yapılması işlemleri için 30 günlük zaman dilimi uygun görülmüştür.

* **Sistem çözümleme işlemleri (01.08.2019-15.08.2019)**  
  Gereksinimlerin belirlenmesi, kullanıcı ara yüz tasarımlarının hazırlanması, piyasadaki mevcut sistemlerin analizi için yaklaşık 2 haftalık zaman dilimi uygun görülmüştür.
* **Tasarım aşaması (16.08.2019-30.10.2019)**  
  Veri tasarımı, yapısal tasarım ve kullanıcı tasarımlarının yapılması için yaklaşık 70-75 günlük bir zaman dilimi uygun görülmüştür.
* **Proje gerçekleştirim aşaması(01.11.2019-01.04.2020)**  
  Yazılım geliştirme ortamlarının belirlenmesi, kodlama stilinin kararlaştırılması, kodlamanın gerçekleştirilmesi, hata durumlarının yönetilmesi, yazılan kodun gözden geçirilmesi gibi işlemler için yaklaşık 5 aylık zaman dilimi uygun görülmüştür.
* **Yazılım doğrulama ve geçerleme aşaması(01.04.2020-01.05.2020)**  
  Tüm sisteminin doğruluğunun sınanması için 30 günlük zaman dilimi uygun görülmüştür.
* **Kurulum ve eğitim aşaması(01.05.2020-01.06.2020)**  
  Sistemin kurulumu ve kurum personellerinin sistemle ilgili eğitimlerinin tamamlanması için 30 günlük zaman dilimi uygun görülmüştür.
* **Hata kontrolleri, genel kontrol ve bakım işlemleri(01.06.2020-01.08.2020)**  
  Sistem kurulduktan sonraki 2 ay boyunca sistemin doğru çalıştığı gözlemlenecek olup meydana gelen hatalara müdahale edilecektir. Ayrıca bu süre zarfı içerisinde istenilen iyileştirmeler ve değişiklikler gerçekleştirilecektir.

**E.SİSTEM ÇÖZÜMLEME**

**E.1.TANIMLAMALAR**

Personel: Giriş/Çıkış yapacak kişi

Sistemi kullanacak yetkili: Web arayüz tarafında kontrolleri, rapor alma vb işleri yapacak kişi

**E.2.GEREKSİNİMLER**

Gereksinim; sistemin amaçlarını yerine getirme yeteneği olan bir özellik ya da belirtim olarak tanımlanmaktadır.

Gereksinim türleri şu şekilde sıralanabilir;

* + Fiziksel çevre
  + Ara yüzler
  + Kullanıcı ve insan etmeni
  + İşlevsellik
  + Belgeleme
  + Veri
  + Kaynaklar
  + Güvenlik
  + Kalite güvencesi

**E.3.SİSTEM ÇÖZÜMLEME**

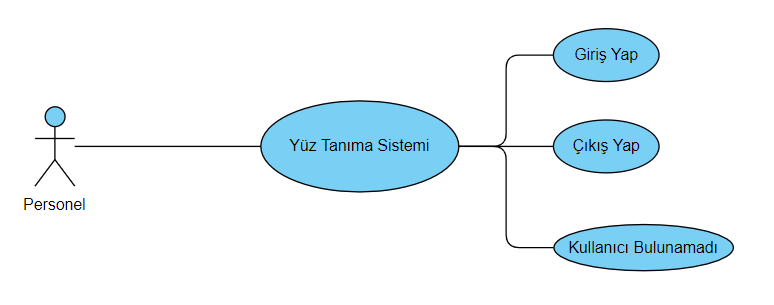
**E.3.1.Mevcut Sistemin İncelenmesi**

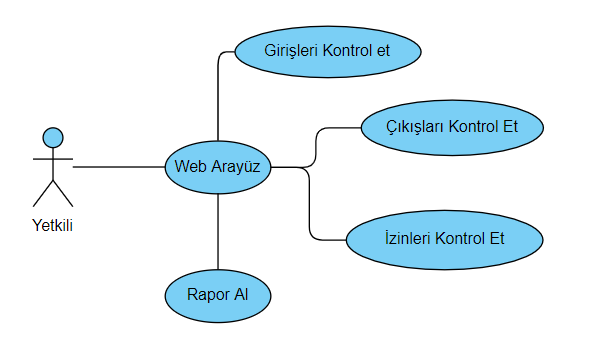
Firmada yüz tanıma sistemi ve web tabanlı arayüzün kullanımı test edilmiştir.

**E.3.2Mevcut Sistemin Modellenmesi**

Mantıksal model olarak da adlandırılan bu adım önerilen sistemin işlevsel yapısını, veri yapısını ve kullanıcı ara yüzünü oluşturur.

**E.3.2.1İşlevsel Modelleme**

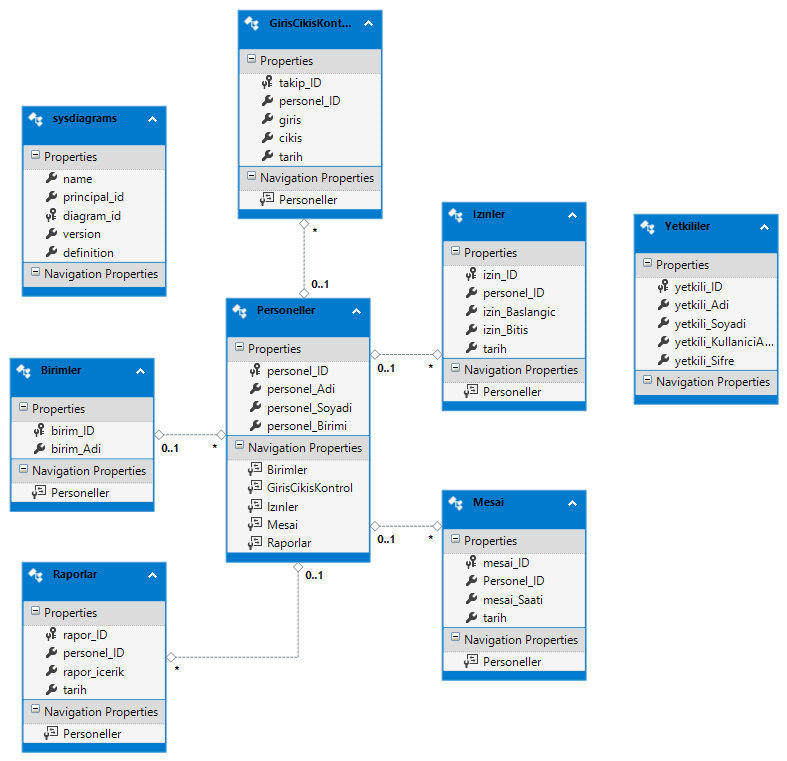
İşlevsel modelleme için use-case diyagramından faydalanılmıştır. 



**E.3.2.2.Veri Modelleme**

Veri modeli olarak Nesne İlişki Şemaları ve Veri Sözlüğü kullanılmıştır.

**E.3.2.2.1.Nesne-İlişki Şeması**



**E.3.2.2.2Veri Sözlüğü**

**1.Yetkililer Tablosu**

Sisteme giriş yapma yetkisi bulunan yetkililerin bulunduğu tablodur.

* **yetkili\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **yetkili\_Adi:** Yöneticinin adı
* **yetkili\_Soyadi:** Yöneticinin soyadı
* **yetkili\_KullaniciAdi:** Yöneticinin sisteme giriş yapacağı kullanıcı adı
* **yetkili\_Sifre:** Yöneticinin sisteme giriş yapacağı şifre bilgisini saklar.

**2.Birimler Tablosu**

Birimler hakkında bilgilerin bulunduğu tablodur.

* **birim\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **birim\_Adi:** Yöneticinin adı

**3.Raporlar Tablosu**

Raporlar hakkında bilgilerin bulunduğu tablodur.

* **rapor\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **Personel\_ID:** Raporu tutulan personelin ID bilgisi
* **Rapor\_icerik:** Hazırlanmış raporun içeriği
* **tarih:** Raporun hazırlandığı tarih

**4.Izınler Tablosu**

İzinli personellerin bulunduğu tablodur. Personel giriş yapmadığında bu tablo kontrol edilir. İzinli ise işlem yapılmaz.

* **izin\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **personel\_ID:** İzinli personelin ID bilgisi
* **izin\_Baslangic:** İzin başlangıç tarihi
* **izin\_Bitis:** İzin bitiş tarihi
* **tarih:** İzin işleminin yapıldığı tarih

**5.Personeller Tablosu**

Personellerin gerekli bilgilerinin bulunduğu tablodur. Detay bilgiler ana sistemde bulunur.

* **personel\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **personel\_Adi:** Personelin ad bilgisi
* **personel\_Soyadi:** Personelin soyad bilgisi
* **personel\_Birimi:** Personelin birim bilgisi
* **6.GirisiCikisKontrol Tablosu**

Personellerin giriş çıkış bilgilerinin bulunduğu tablodur.

* **takip\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **personel\_ID:** Personelin ID si
* **giris:** Personelin giriş saati
* **cikis:** Personelin çıkış saati
* **tarih:**İşlemin yapıldığı tarih

**7.Mesai Tablosu**

Personellerin mesai bilgilerinin bulunduğu tablodur. Giriş çıkış saatleri kontrol edilerek otomatik ekleme yapılır.

* **mesai\_ID:** Otomatik olarak artan int tipinde primary key özelliği bulunan alandır.
* **personel\_ID:** Personelin ID si
* **mesai\_saati:** Yapılan mesai saati
* **tarih:**İşlemin yapıldığı tarih

**E.4.KULLANICI ARAYÜZLERİ**

Web tabanlı hazırlanacak olan sistemin ara yüz tasarım ve kodlamaları responsive(duyarlı) olacaktır.

Yetkili kişiler arayüz üzerinden giriş-çıkışları görebilecek ve raporlama ile izin işlemlerini yapabileceklerdir.

**F. TASARIM**

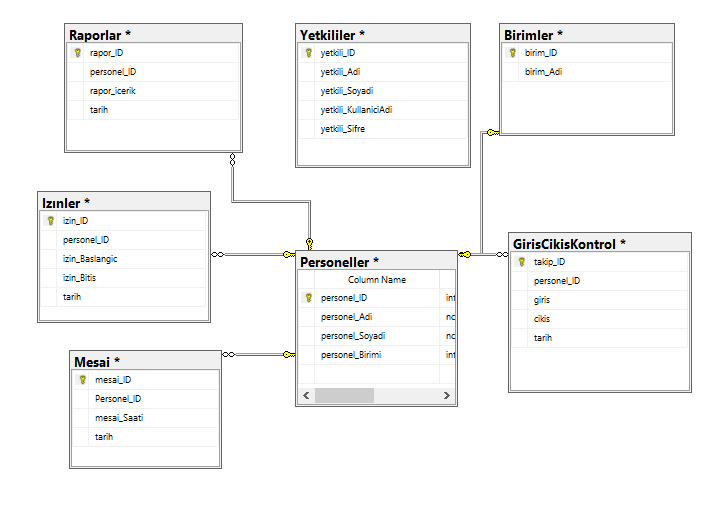
Tasarım, Sistem Analizi çalışması sonucunda üretilen Mantıksal Modelin Fiziksel Modele dönüştürülme çalışmasıdır.

**F. 1.Genel Tasarım**

Sistem kurum bünyesinde çalışacak olup, personellerin giriş çıkış işlemleri, izin işlemleri ve rapor işlemleri yapılabilecektir.

**F. 2.Veri Tasarımı**

Veri tabanı tablolarının ve aralarındaki ilişkilerin yapıldığı aşamadır.  
İlgili veri tabanı diyagramı aşağıda verilmiştir.



**G. GERÇEKLEŞTİRİM**

Tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmasıdır.

**G.1.YAZILIM GELİŞTİRME ORTAMI**

Yazılım kodlama kısmında kullanılacak platformlardır.

• Ara yüzlerin geliştirilmesi ortamı için Visual Stucdio Code kullanılacaktır.

• Arayüz tasarımında Angular.js kullanılacaktır.

• Back-end tarafında Asp.Net MVC Core 2 kullanılacaktır.

• SQL SERVER 2016 kullanılacaktır.

**G.2.PROGRAMLAMA DİLLERİ**

•Asp.Net

•SQL Server

**G.3.VERİ TABANI YÖNETİM SİSTEMİ**

* Microsoft MSSQL kullanılacaktır.

**G.4.KODLAMA STİLİ**

•Proje için dokümantasyon hazırlanacak olsa da, gerekli görülen kod satırlarına açıklama satırlarının da eklenmesi gerektiği kararlaştırıldı.

•Değişken, sınıf ve metot isimlerinin Türkçe olmasına karar verildi.

**G.5.KARMAŞIKLIK ÖLÇÜTÜ**

Sistemin daha verimli olması için karmaşıklık ölçütü yapılmıştır.

McCabe karmaşıklık ölçütü yapılmıştır ve sistemin karmaşıklık ölçütü gayet iyidir.

**G.6.KOD GÖZDEN GEÇİRME**

Karmaşıklık ölçütünden sonra bakım ekibi tarafından kod gözden geçirmesi uygulanacaktır. Yazılım geliştirme aşamasındaki stres ve dalgınlıkla yapılan hatalar kod gözden geçirme aşamasında fark edilebilir. Bu yüzden kod gözden geçirme aşaması yapılacak ve bulunursa hatalar giderilecektir.

## H. YAZILIM DOĞRULAMA VE GEÇERLEME

## GİRİŞ

## 1 Tanım ve Kısaltmalar

Doğrulama: Ürünü kullanacak olan kişilerin isteklerinin karşılanıp karşılanmadığının tespiti için yapılan etkinlikler bütünüdür.

Geçerleme: Ürünün içsel niteliğine ilişkin izleme ve denetim etkinliklerinin bütünüdür.

## 2 SINAMA YÖNETİMİ

Bu projenin sınanması aşamasında beyaz kutu testi ve kara kutu testi uygulanmıştır.

## 2.1 Beyaz Kutu Sınaması

Projedeki bütün bağımsız yollar en az bir kez sınanmıştır.Bunun yanında bütün döngülerin sınır değerlerinde sınama işlemi yapılarak, iç veri yapıları denenmiştir.Son olarak bütün mantıksal noktalarında karar noktalarında iki değişik karar için sınamalar yapılıp tüm sonradan oluşabilmek hata riski azaltılmıştır.

**2.2 Kara Kutu Sınaması**

Program iş akış diyagramı hazırlanıp çizge olarak kullanılmıştır ve çevrimsel karmaşıklık sayısı ve bağımsız yol sayısı hesaplanmıştır.

## 3 SINAMA AYRINTILARI

## 3.1 Sınanan Birimler

Sınama aşamasında genel olarak alt sistem, sistem ve modül sınamaları yapılmıştır.

## 3.2 Sınama ve Bütünleştirme Stratejileri

Uygulamada sandviç adı verilen hem üstten hem alttan sınama işlemi gerçekleştirilmiştir. Üstten sınama işlemi Yukarıdan Aşağıya Sınama ve Bütünleştirme Stratejisi ile en üst düzeylerin sınanması ve sonra aşağıya doğru sınanması, alttan sınama işlemi Aşağıdan Yukarıya Sınama ve Bütünleştirme Stratejisi ile alt düzeylerden başlayıp sonra yukarı doğru sınanması şeklinde gerçekleştirilmiştir.

## 3.3 Sınama Sonuçları

Belirtilen yöntemlerle doğrulama ve geçerleme işlemleri yapıldı. Tüm sistemin olası hataları aranıp gözden geçirildi. Bulunan hatalar ya da eksiklikler giderildi. Belirlenen hedefe ulaşılarak, neticede gerçekleştirilen yazılım kuruma entegrasyonu sağlanacak şekle başarıyla getirilmiş oldu.