

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**Yazılım Mühendisliği**

**Proje Dokümantasyonu**

***Tez Analiz***

**Proje Çalışma Grubu**

**GÜRKAN ŞENTÜRK**

ŞUBAT 2021

|  |
| --- |
| 1. GİRİŞ |
| ÖNSÖZ  1.1 Projenin Amacı  1.2 Projenin Kapsamı  1.3 Tanımlar ve Kısaltmalar |
| 2.PROJE PLANI |
| 2.1 Giriş  2.2 Projenin Plan Kapsamı  2.3 Proje Zaman-İş Planı  2.4 Proje Ekip Yapısı  2.5 Önerilen Sistemin Teknik Tanımları  2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları  2.7 Proje Standartları, Yöntem ve Metodolojiler  2.8 Kalite Sağlama Planı  2.9 Konfigürasyon Yönetim Planı  2.10 Kaynak Yönetim Planı  2.11 Eğitim Planı  2.12 Test Planı  2.13 Bakım Planı |
| 3. SİSTEM ÇÖZÜMLEME |
| 3.1 Mevcut Sistem İncelemesi  3.1.1 Örgüt Yapısı  3.1.2 İşlevsel Model  3.1.3 Veri Modeli  3.1.4 Var olan Yazılım/Donanım Kaynakları  3.1.5 Var olan Sistemin Değerlendirilmesi  3.2 Gereksenen Sistemin Mantıksal Modeli  3.2.1 Giriş  3.2.2 İşlevsel Model  3.2.3 Genel Bakış  3.2.4 Bilgi Sistemleri/Nesneler  3.2.5 Veri Modeli  3.2.6 Veri Sözlüğü  3.2.7 İşlevlerin Sıradüzeni  3.2.8 Başarım Gerekleri  3.3 Ara yüz (Modül) Gerekleri  3.3.1 Yazılım Ara yüzü  3.3.2 Kullanıcı Ara yüzü  3.3.3 İletişim Ara yüzü  3.3.4 Yönetim Ara yüzü  3.4 Belgeleme Gerekleri  3.4.1 Geliştirme Sürecinin Belgelenmesi  3.4.2 Eğitim Belgeleri  3.4.3 Kullanıcı El Kitapları |
| 4. SİSTEM TASARIMI |
| 4.1 Genel Tasarım Bilgileri  4.1.1 Genel Sistem Tanımı  4.1.2 Varsayımlar ve Kısıtlamalar  4.1.3 Sistem Mimarisi  4.1.4 Dış Arabirimler  4.1.4.1 Kullanıcı Arabirimleri  4.1.4.2 Veri Arabirimleri  4.1.4.3 Diğer Sistemlerle Arabirimler  4.1.5 Veri Modeli  4.1.6 Testler  4.1.7 Performans  4.2 Veri Tasarımı  4.2.1 Tablo tanımları  4.2.2 Tablo- İlişki Şemaları  4.2.3 Veri Tanımları  4.2.4 Değer Kümesi Tanımları  4.3 Süreç Tasarımı  4.3.1 Genel Tasarım  4.3.2 Modüller  4.3.2.1 Giriş Modülü  4.3.2.1.1 İşlev  4.3.2.1.2 Kullanıcı Arabirimi  4.3.2.1.3 Modül Tanımı  4.3.2.1.4 Modül iç Tasarımı  4.3.2.2 Kullanıcı Modülü  4.3.3 Kullanıcı Profilleri  4.3.4 Entegrasyon ve Test Gereksinimleri  4.4 Ortak Alt Sistemlerin Tasarımı  4.4.1 Ortak Alt Sistemler  4.4.2 Modüller arası Ortak Veriler  4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri  4.4.4 Güvenlik Alt sistemi  4.4.5 Veri Dağıtım Alt sistemi  4.4.6 Yedekleme ve Arşivleme İşlemleri |
| 5. SİSTEM GERÇEKLEŞTİRİMİ |
| 5.1. Giriş  5.2. Yazılım Geliştirme Ortamları  5.2.1 Programlama Dilleri  5.2.2 Veri Tabanı Yönetim Sistemleri  5.2.2.1 VTYS Kullanımının Ek Yararları  5.2.2.2 Veri Modelleri  5.2.2.3 Şemalar  5.2.2.4 VTYS Mimarisi  5.2.2.5 Veri tabanı Dilleri ve Arabirimleri  5.2.2.6 Veri Tabanı Sistem Ortamı  5.2.2.7 VTYS'nin Sınıflandırılması  5.2.2.8 Hazır Program Kütüphane Dosyaları  5.2.2.9 CASE Araç ve Ortamları  5.3. Kodlama Stili  5.3.1 Açıklama Satırları  5.3.2 Kod Biçimlemesi  5.3.3 Anlamlı İsimlendirme  5.3.4 Yapısal Programlama Yapıları  5.4. Program Karmaşıklığı  5.4.1 Programın Çizge Biçimine Dönüştürülmesi  5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama  5.5. Olağan Dışı Durum Çözümleme  5.5.1 Olağandışı Durum Tanımları  5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları  5.6. Kod Gözden Geçirme  5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi  5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular  5.6.2.1 Öbek Ara yüzü  5.6.2.2 Giriş Açıklamaları  5.6.2.3 Veri Kullanımı  5.6.2.4 Öbeğin Düzenlenişi  5.6.2.5 Sunuş |
| 6. DOĞRULAMA VE GEÇERLEME |
| 6.1. Giriş  6.2. Sınama Kavramları  6.3. Doğrulama ve Geçerleme Yaşam Döngüsü  6.4. Sınama Yöntemleri  6.4.1 Beyaz Kutu Sınaması  6.4.2 Temel Yollar Sınaması  6.5. Sınama ve Bütünleştirme Stratejileri  6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme  6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme  6.6. Sınama Planlaması  6.7. Sınama Belirtimleri  6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri |
| 7. BAKIM |
| 7.1 Giriş  7.2 Kurulum  7.3 Yerinde Destek Organizasyonu  7.4 Yazılım Bakımı  7.4.1 Tanım  7.4.2 Bakım Süreç Modeli |
| 8. SONUÇ |
| 9. KAYNAKLAR |

|  |
| --- |
| 1.GİRİŞ |

* **Projenin Amacı:**

Yazılan tezlerin belli kurallar çerçevesinde uygun formatta olup olmadığını öğrenmeyi amaçlar.

* **Projenin Kapsamı:**

.docx formatının açılabildiği tüm sistemlerde çalıştırılabilmektedir.

* **Tanımlar ve Kısaltmalar:**

|  |
| --- |
| 2.PROJE PLANI |

**2.1 Giriş:**

Bu yazılım, belli kurallara uygun olarak dokümantasyonu inceler ve bunun hakkında raporlar çıkarır.

**2.2 Projenin Plan Kapsamı:**

Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin

güvenilirliğinden yola çıkıldı. Analiz Yazılımı tamamen pratiklik ve zamandan kazanç odaklı bir

oluşumdur. Bu sayede işlemlere büyük ölçüde yardımcı olur.

**--Tez Analiz Sistemi Nasıl Çalışır?**

* Kullanıcı dokümanı seçer.
* Daha sonra Dosyayı tara butonuna basar
* Program gerekli raporları oluşturur.

**Maliyet Kestirim Dokümanı**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ölçüm parametresi** | Sayı | **Ağırlık** | **Toplam** |
| **Kullanıcı girdi sayısı** | **11** | **4** | **44** |
| **Kullanıcı çıktı sayısı** | **9** | **7** | **63** |
| **Kullanıcı sorgu sayısı** | **3** | **3** | **9** |
| **Kütük sayısı** | **5** | **6** | **30** |
| **Dışsal ara yüz sayısı** | **0** | **0** | **0** |
| **Ana işlev nokta sayısı** |  |  | **146** |

|  |  |
| --- | --- |
| **Teknik Karmaşıklık Sorusu** | **Puan** |
| **1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu?** | **5** |
| **2. Veri iletişimi gerekiyor mu?** | **5** |
| **3. Dağıtık işlem işlevleri var mı?** | **3** |
| **4. Performans kritik mi?** | **5** |
| **5. Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?** | **2** |
| **6. Sistem, çevrim içi veri girişi gerektiriyor mu?** | **5** |
| **7. Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?** | **4** |
| **8. Ana kütükler çevrim-içi olarak mı güncelleniyor?** | **5** |
| **9. Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı?** | **3** |
| **10. İçsel işlemler karmaşık mı?** | **3** |
| **11. Tasarlanacak kod, yeniden kullanılabilir mi olacak?** | **4** |
| **12. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?** | **2** |
| **13. Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor?** | **1** |
| **14. Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak?** | **2** |
| **TOPLAM** | **49** |

**0**: Hiçbir Etkisi Yok

**1:** Çok Az etkisi var

**2**: Etkisi Var

**3**: Ortalama Etkisi Var

**4**: Önemli Etkisi Var

**5:** Mutlaka Olmalı, Kaçınılamaz

İN = AİN x (0.65 x 0.01 x TKF)

İN=146\*(0,65\*0,01\*49)

**İN=46,501**

Satır sayısı = İN\*30

**SATIR SAYISI = 46,501\*30=1.395,03 SATIR YAKLAŞIK 1400 SATIR**

Etkin Maliyet Modeli --COCOMO

**Organik Projeler İçin:** a=2,4, b=1,05, c=2,5, d= 0,38

**Yarı – Gömülü Projeler İçin:** a=3,0, b=1,12, c=2,5, d= 0,35

**Gömülü Projeler İçin:** a=3,6, b=1,20, c=2,5, d= 0,32

Projemin türü organik projeler türüne giriyor sebebi ise küçük ekip tarafından geliştirilmesi olarak gösterilebilir.

Aylık Kişi Başı İş Gücü = E = a x (KSS)b

Geliştirme Süresi (Ay) = D = c x (E)d

Eleman Sayısı = E / D

Formülde verilen değişkenler şöyle:

KSS = Kod Satır Sayısı manasına gelmektedir ve birimi bin satırdır. Projenin tahmini kaç bin satırdan oluşacağını belirtmemizi sağlar.

Aylık Kişi Başı İş Gücü = E=2,4\*11,05=2,4

Geliştirme Süresi =D= 2,5 x2,40,38= 3,48

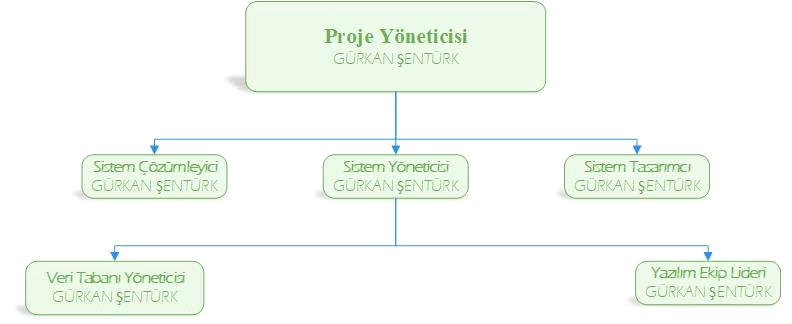
Tahmini Gerekli Personel Sayısı = 3,48 / 2,4= 1,45 yaklaşık 1.50

**2.3 Proje Zaman-İş Planı**

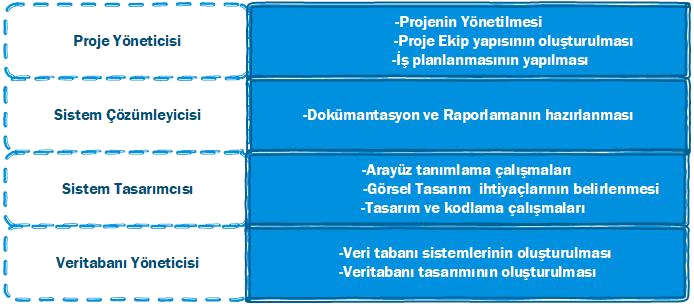
|  |  |  |  |
| --- | --- | --- | --- |
| **İş-Zaman Çizelgesi** | | | |
| **Zaman**  **İŞ** | **1.Hafta** | **2.Hafta** | **3.Hafta** |
| **Proje Teklifi** | **X** |  |  |
| **Proje Planı** | **X** |  |  |
| **Analiz** | **X** |  |  |
| **Sistem Çözümleme** | **X** | **X** |  |
| **Kullanıcı Ara yüz Tasarımı** |  | **X** | **X** |
| **Gerçekleştirim** |  | **X** | **X** |
| **Test** |  |  | **X** |
| **Sunum** |  |  | **X** |

**Şekil 2.2 Proje İş-Zaman Çizelgesi**

**2.4 Proje Ekip Yapısı**



Şekil 2.3



Şekil 2.4 Proje Ekip Görevleri

|  |  |  |  |
| --- | --- | --- | --- |
| Ekip Yapısı ZAMAN-İŞ Planı | | | |
| **AYLAR**  **EKİP** | **NİSAN** | | **MAYIS** |
| Sistem Yöneticisi |  |  |  |
| Sistem Çözümleyici |  |  |  |
| Sistem Tasarımcısı |  |  |  |
| Veri Tabanı Yöneticisi |  |  |  |

Şekil 2.5 Zaman-İş çizelgesi

**2.5 Önerilen Sistemin Teknik Tanımları**

Kullanılan Teknoloji

* Document tarayıcı

**2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları**

Çözümleme ve Tasarım Araçları

* Microsoft Visual Studio 2019
* Microsoft Visio

Programlama Araçları

* C#

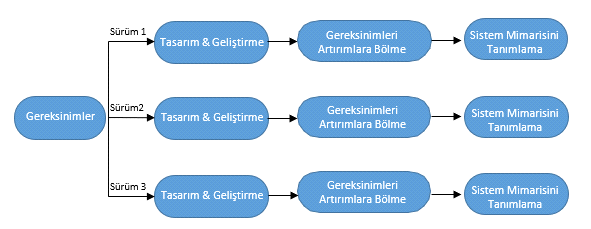
Sınama Araçları

* Microsoft Windows 10
* Microsoft Windows 8
* Microsoft Windows 7

Destek Araçlar

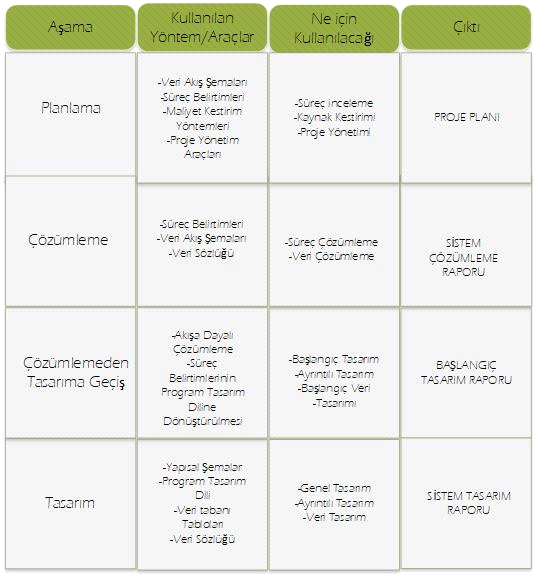
* Mozilla Firefox 11
* Google Chrome
* Opera
* Microsoft Edge

**2.7 Proje Standartları, Yöntem ve Metodolojiler**



Şekil 2.6 Artımlı Geliştirme Modeli

Kullanılan Model Artımlı Süreç Modeli. Sebebi ise oluşturduğum otomasyonda maliyetim düşük seviyelerde. Sistemde kolaylıkla değişim yapabileceğim için basit ve başarı garantisi yüksek olacak. Bu sebeple bu modelin, geliştirdiğim sistem için uygun olduğunu öngördüm. İlk olarak ilk isterler toplanır. Bu sayede mevcut problem varsa tespiti yapılır. Ardından çözümleyici yazılımın ne yapması gerektiği ortaya koyar. Ortaya koyulan bilgi ile problem karşılaştırılır sonuç olarak problemin çözüm haritası çıkarılmış olur. Bu şekilde ortaya çıkan problem çözülmüş olur.



Şekil 2.7 Proje Aşamaları

Proje standartları yukarıda belirtildiği gibidir. Sistem modeli başlangıçta da belirtiğim gibi Artımlı Süreç Modelidir.

**2.8 Kalite Sağlama Planı**



Projede kalite sağlama planım aşağı-yukarı üstte bulunan tablodaki gibidir.

* **Ekonomi**: Maliyet açısından düşük olması daha fazla kazanç sağlamaktadır.
* **Tamlık**: Projede her hangi bir sorun, açık bulunmamalı, veri güvenliği sağlanmalıdır.
* **Yeniden** **Kullanılabilirlik**: Sistemi isteğe göre tekrardan dizayn edebilme imkânı olmalıdır.
* **Etkinlik**: Sistem oldukça basit ve kullanıcı bir tasarıma sahip olduğu için işlemler etkin bir şekilde kullanılacak.
* **Güvenilirlik**: Veri güvenliği için gereken önlemler alınacaktır.
* **Belgeleme**: Bu belgeden de anlaşılacağı üzere tam anlamıyla sistemin özeti olacak bu doküman oluşturulacak.
* **Kullanılabilirlik**: Oldukça sade ve basit ara yüze sahip olduğundan her seviyeden insan kullanabilecek.
* **Temizlik**: Kelime karmaşası, görüntü kirliliği olmayacak.
* **Esneklik**: Sistem farklı işletim sistemlerinde ve internet üzerinde çalışabildiğinden oldukça esnek bir yapısı vardır.
* **Genellik**: Sistem katılımın olduğu her yerde kullanılabilir.
* **Taşınabilirlik**: Sistem kamerası bulunan her türlü bilgisayarda çalışmaktadır.

**2.9 Konfigürasyon Yönetim Planı**

Sistemde meydana gelebilecek bazı sorunlar için konfigürasyon planı hazırlandı.

* Yeni Kullanıcı Eklenmesi,
* Kullanıcı Silinmesi,
* Sistemde oluşacak herhangi bir sorun,

Gibi oluşacak durumlar için yapılandırma yönetim planı oluşturuldu.

**2.10 Kaynak Yönetim Planı**

Mevcut bir kaynağım yok. Kaynak olarak gösterebileceğim yalnızca bu proje dokümantasyonu var.

**Kaynak Yönetiminde şu hususlara dikkat edilecektir;**

* Kalite beklendiği gibi mi?
* Hata mevcut mu? Mevcutsa nasıl bir çözüm yolu izlenecek?
* Proje elemanları yeni bir eğitim aldı mı?
* Yeterli vakit harcanıyor mu?

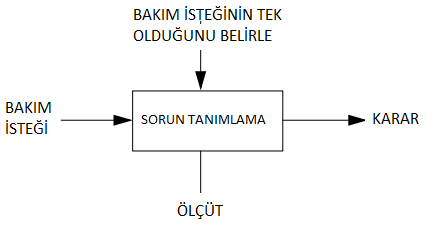
**2.11 Test Planı**

Proje test takımı ve görevleri şu şekildedir;

* Kullanıcılardan en yetkili olan kişi yönetici girişi yapıp yönetici olarak görev yapacak.
* Veri tabanı kısmını sunucuyu sağlayan firma üstlenecek.

Bu takım kontrolleri yapıp herhangi bir istenmeyen durum olduğu süreçte bizle iletişime geçecek ve sorunu ortadan kaldıracağız.

**2.12 Bakım Planı**



Proje duruma göre her gün kullanılmayacak. Yani katılımın olmadığı zamanlarda proje kullanılmayacak. Bu ve bunun gibi sistemde tüm değişimler, kullanıcı eklenip çıkarılması gibi tüm bu sistemsel değişiklikler bakım planında yapılacaktır.

|  |
| --- |
| **3. SİSTEM ÇÖZÜMLEME** |

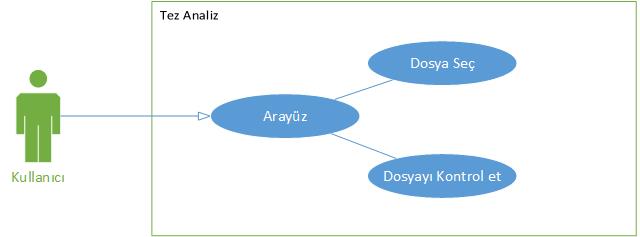
**3.1 Mevcut Sistem İncelemesi**

Mevcut sistem incelemesi alanında bir ilk olmadığı için zor olmayacaktır. Basit bir şekilde internette mevcut olan sistemler incelenip bizim sistemimize ne gibi katkı yapabileceği araştırılıp sisteme uygulanmıştır.

**3.1.1 Örgüt Yapısı**

Mevcut sistemin yapısı dolayısıyla sistemde herhangi bir örgüt yapısına ihtiyaç yoktur.

**3.1.2 İşlevsel Model**



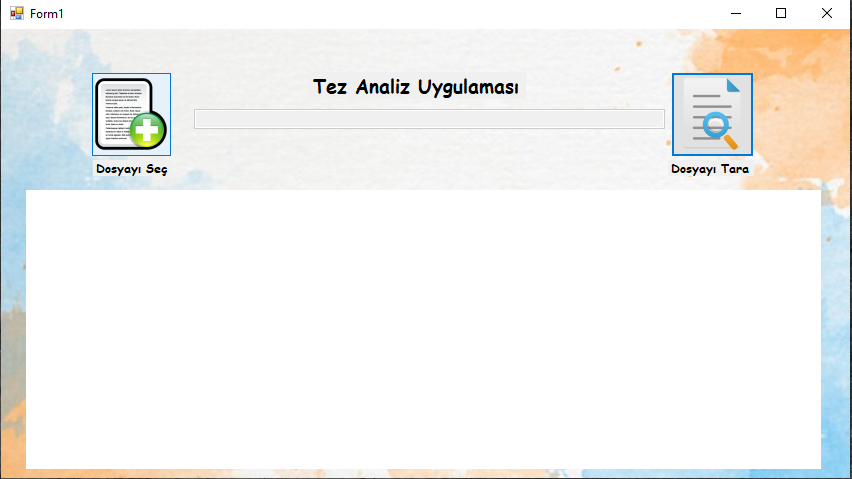
Şekil 3.1 BASİT USE-CASE Diyagramı

**3.1.3 Veri Modeli**

Veri tabanı ilişkisel veri modelinde veriler tablolar üzerinden kurulan ilişkiye dayanmaktadır.

**3.1.4 Var olan Yazılım/Donanım Kaynakları**

* Microsoft Visio
* Microsoft Visual Studio





**3.2 Gereksenen Sistemin Mantıksal Modeli**

**3.2.1 Giriş**

Mevcut sistemler incelendiğinde sonuca giden yolda biraz eksiklikler mevcut. Fakat ileri ki güncellemelerde bu eksiklikleri gidermek bize kalıyor.

Şimdi sistemin işlevsel modeline bir göz atalım

**3.2.3 Genel Bakış**

Genel hatlarıyla sistemi inceleyecek olursak mevcut olan takip sistemlerinde mevcut olan birçok özellik burada da mevcut. Hatta QR kod okuma sistemi olması ile yaptığım sistemin bu özelliği ile öne çıktığını söyleyebiliriz. Eksik yanlarımız elbette mevcut bunları gelecek güncelleme gidereceğiz.

Bunlara ek olarak olayların akış şekli USE-CASE diyagramlarında ayrıntılı olarak mevcut bulunmaktadır. Görüldüğü üzere silme gibi kritik işlemleri standart kullanıcı yapamaz sadece üst yönetici olarak adlandırdığımız standart kullanıcısı bu tip kritik işlemleri gerçekleştirme yetkisine sahiptir.

**3.2.4 Bilgi Sistemleri/Nesneler**

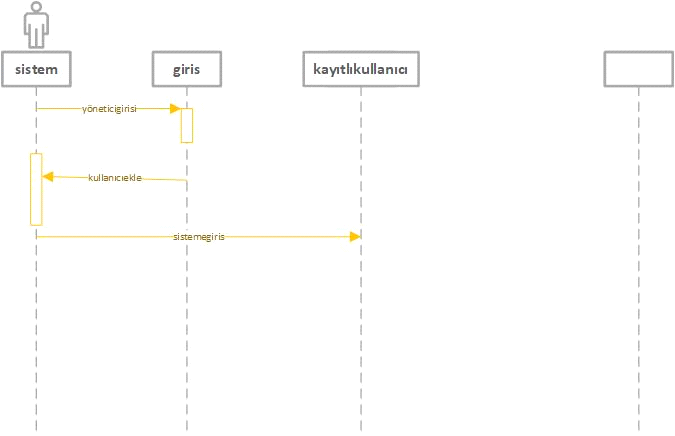
Kullanıcı: Yapının üst kısmında bulunan yetkileri sınırsız olan yapılanmadır.

* **Veri Modeli**

Verilen veri modeli ile önceki attığımız model arasında fark yoktur fakat konuyu kavramak için tekrar gösterme ihtiyacı duyuldu.

* **İşlevlerin Sıradüzeni**

Sisteme kaydı alınan katılımcı öncelikle sistemde aktif olan(girişte girilen kullanıcı) yönetici tarafından veri tabanına gönderilir ve ilgili tabloya kayıt edilir. Bu şekilde katılımcı ekleme işlemi sağlanır. Aynı şekilde kullanıcı ve etkinlik ekleme işlemleri bu şekilde olmaktadır.



Şekil 3.11 Giriş Diyagramları

* **Başarım Gerekleri**

Mevcut sistem incelendi ve sistemin eksiklerinden yola çıkarak, sistemin başarımı için

* Tepki süresinin en aza indirilmesi,
* Anlaşıla bilirlik,
* Kullanım kolaylığı,
* Hata ve yanlışlıkların en aza indirgenmesi

Temel gereklilikler olarak tespit edilmiştir.

* **Ara yüz(Modül) Gerekleri**

**3.3.1 Yazılım Ara yüzü**

Proje çalışma sırasında böyle bir açık verilmemesine özen gösterildi.

**3.3.2 Kullanıcı Ara yüzü**

Kullanıcı ara yüzünde yapılan işlemler sınırlandırıldı. Kullanıcı ara yüzünde yalnızca ekleme işlemleri yapılmakta. Görünüm olarak ise renkler ve büyük puntolu yazılar seçilerek canlı ve sıkmayan bir görüntü sağlandı.



Şekil 3.13 Kullanıcı İş Akışı

**3.3.3 İletişim Ara yüzü**

Sistemde iletişim ara yüzü henüz eklenmedi fakat isteğe bağlı olarak eklenebilir.

* **Belgeleme Gerekleri**

**3.4.1 Geliştirme Sürecinin Belgelenmesi**

Geliştirme sürecinde belgelendirme hem şimdiki geliştirme sürecinde hem de ileriye dönük projenin tamamlama yüzdesini nerede kalınıp nerelerde eksikler olduğunu genel hatlarıyla göstermesi amacıyla yapıldı.

**3.4.2 Eğitim Belgeleri**

Herhangi bir belge bulunmamaktadır.

**3.4.3 Kullanıcı El Kitapları**

Sistem basit bir yapıya sahip olduğu için herhangi bir el kitabı mevcut değildir.

|  |
| --- |
| * **SİSTEM TASARIMI** |

* **Genel Tasarım Bilgileri**
* **Genel Sistem Tanımı**



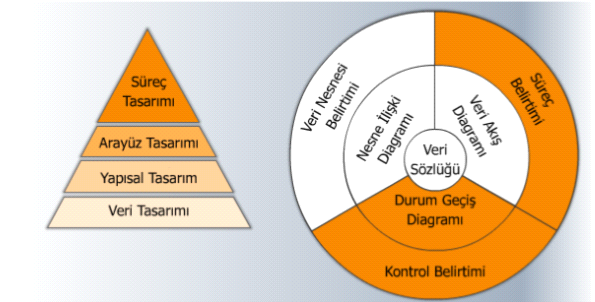
**Şekil 4.1 Genel sistem tanımı**

* **Gereksinimler**

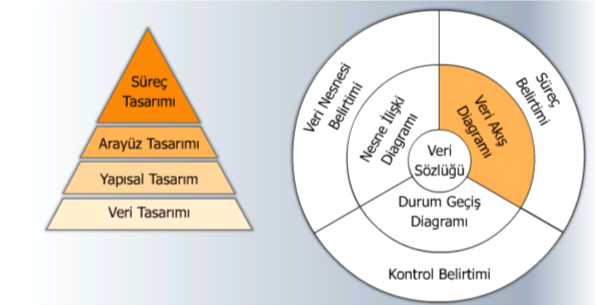
Gereksinimler bu sistemi kullanabilecek olan oluşumlardan bilgi alarak ona göre tasarım oluşturulmaya başlanacaktır.

* **İşlevsel gereksinim:** Burada ilk olarak sistem ne yapacak sorusuna cevap vereceğim. Sistem üniversitemiz içinde veya katılımın fazlaca olduğu etkinliklerde zamandan tasarruf sağlayacak.
* İkinci olarak yazılım ne yapacak sorusuna cevap verecek olursam, yazılım çevrim-içi bir sunucuyla bağlantıda olarak etkinliğe gelecek olan insanlara önceden gönderilmiş QR kodları etkinlik girişinde okuyarak kimlerin geldiğini, kimlerin gelmediğini tespit edecek ve buna göre listeleme yapacak bu şekilde hem zamandan tasarruf edilecek, kâğıttan tasarruf edilecek bunları şu şekilde açıklayabilirim normalde etkinlik girişlerinde bilgiler alınıp kâğıda işleniyor ve bu şekilde hem kuyruk oluşuyor hem de zamanda sorun yaşanıyor. Bu yazılım ile bu sorunun önüne geçilecek.
* **Tasarım**
* Öncelikle bir süreç tasarımı olacak. Ve adımları aşağıdaki gibi olacak.

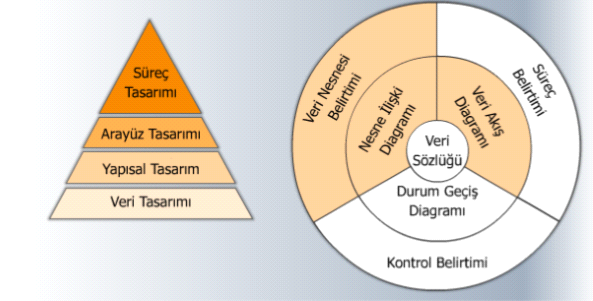




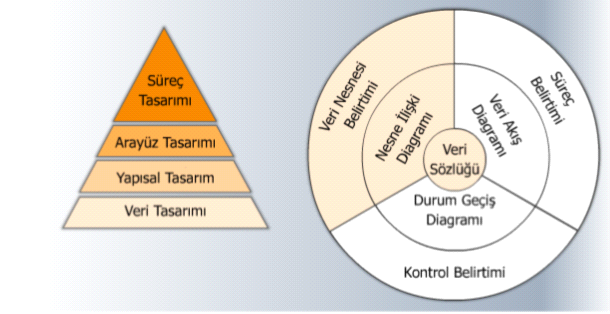
2.Süreç tasarımı bitince sırada ara yüz tasarımına geleceğiz ve ara yüz tasarımı şu şekilde gerçekleştirilecek.



3.Arayüz tasarımını da hallettikten sonra Yapısal tasarım kısmına geliyoruz ve adımlar şu şekilde oluyor.



4.Yapısal tasarımın ardından Veri tasarımı geliyor onun içinde şu adımlar izlenecek.



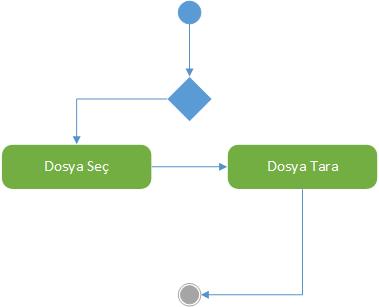
Şekiller 4.2 Tasarım Aşamaları

**4.1.2 Varsayımlar ve Kısıtlamalar**

Sistemde varsayılan değer olarak kullanıcının takılım sayısı başlangıçta 1 olarak varsayılır ve katılımı arttıkça bu sayı artacaktır.

Yanı sıra var olan kısıtlamalar ise şunlardır:

* Yalnızca yönetici tarafından eklenenen kullanıcılar olması,
* Veri aktarımı yapması
* **Sistem Mimarisi**

****

**Şekil 4.3 Genel Sistem Mimarisi**

Sistem mimarisinin akış diyagramında olmasının temel nedeni sistemin çalışma mantığının nasıl olduğu ve nasıl bir yoldan gidileceğinin anlaşılmasını sağlamaktır.

* **Dış Arabirimler**

**4.1.4.1. Kullanıcı Arabirimleri**

Kullanıcı arabirimlerinin ilk başında sistem giriş ekranı bulunacak. Ve buradan giriş yapılacak.

**4.1.4.2 Veri Arabirimleri**

Veri arabirimlerinde sistem çevrimiçi bir sunucu bağlı olduğu için arada XML ve PHP arabirimleri olacaktır. Sisteme girilen veriler sunucuda kayıt altında tutulacak ve iletişim sağlanacaktır.

**4.1.4.3 Diğer Sistemlerle Arabirimler**

Henüz bu arabirim kullanılmamakta.

* **Testler**

Genel olarak testler iki adımda gerçekleşecek.

**Beta Aşaması**: Kullanıcının kendi sisteminde gözetmen eşliğinde yapılacak olan test

**Alfa Aşaması**: Sistemin geliştirildiği sistemde kullanıcıların gelip yerinde test etmesi ile yapılacak olan test.

* **Performans**

Sistemin tasarıma uyguluk performansı;

Tasarımı yapılan sistemin oturtulması ve işleyiş performansı değerlendirilecek

Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecek

Veri yapısının sistemle performansı;

Veri yapısının sistemle oturtulmuş ve çalışma zamanındaki uyumluluk düzeyindeki performansı değerlendirilecek.

**4.2 Veri Tasarımı**

**4.2.1 Tablo Tanımları**

Sistemde temel olarak 4 tablo bulunmakta. Sistemin özelliği gereği istenildiği kadar tablo oluşturulabilir veya silinebilir. Sistemde her varlık katılımcı olduğu ana tabloyu katılımcılar tablosu olarak yarattık. Ayrıca sistemde herhangi bir yetkiye sahip kullanıcılarında kayıtlı olduğu bir tablo mevcut.

**4.2.3 Veri Tanımları**

Veri tipi olarak int kullanılmasının temel sebebi alınan değerin sayısal bir değer olmasıdır. Text olarak kullanılan değerler ise kelime içeren değerleri tutacağı için kullanıldı.

**4.2.4 Değer Kümesi Tanımları**

Değer kümesi bulunmamakta.

**4.3 Süreç Tasarımı**

**4.3.1 Genel Tasarım**

Oluşturduğum sistemde ilk olarak görsel ara yüz oluşturuldu ardından yapısal ara yüz ile paralel olarak veri tabanı modeli oluşturuldu.

**4.3.2 Modüller**

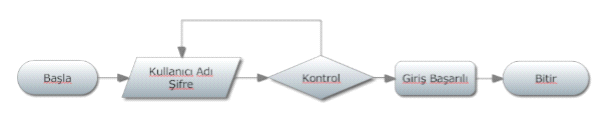
**4.3.2.1.1 İşlev**

Kullanıcının ana ekrana ulaşması için geçmesi gereken parçadır.

**4.3.1.2 Kullanıcı Arabirimi**

**4.3.1.3 Modül Tanımı**

**4.3.1.4 Modül İç Tasarımı**



Kullanılan değişkenler:

* private Connection con: Veri tabanı için bağlantı nesnesi.
* private String host: Veri tabanının bulunduğu yol.
* private String kullaniciAdi: Veri tabanına bağlantı için kullanıcı adı.
* private String sifre: Veri tabanına bağlantı için şifre

|  |  |
| --- | --- |
| Metot adı | Tanım |
| Button2\_click | Buton2’ye basınca dosya seçen metot |
| Buton1\_click | Seçilen dosyayı tarayan metot |

**4.3.2.2 Yönetici Modülü**

**4.3.2.3 İşlev**

Sistem içindeki tüm arabirimler aslında yönetici parça diyebiliriz. Ara yüzler birbirinin aynısı fakat yaptıkları işlevler farklı.

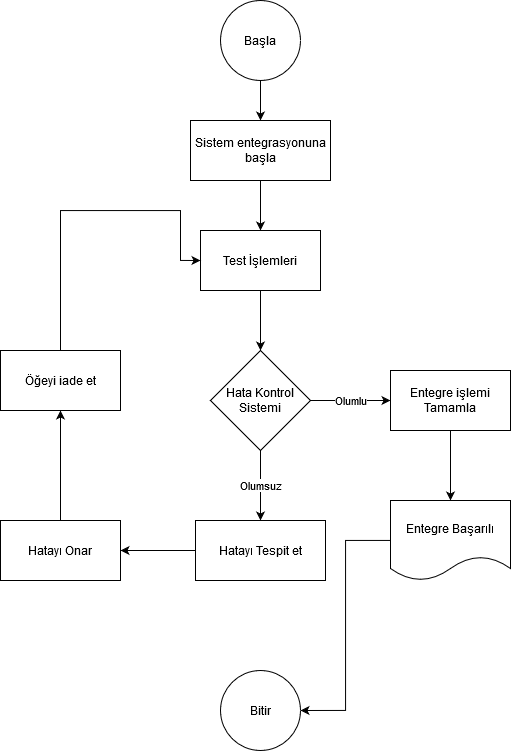
**4.3.3 Kullanıcı Profilleri**

KULLANICI: Tüm kısımlara erişimi olan üst seviye birimdir.

**4.3.4 Entegrasyon ve Test Gereksinimleri**

Sistem basit olduğu için kolayca her sistemde uyum sağlayabilmektedir.

Bunu basit bir şekilde akış diyagramında gösterelim.



**4.4 Ortak Alt Sistemlerin Tasarımı**

**4.4.1 Ortak Alt Sistemler**

Kullanılacak tek ortak sistem kullanılan sunucu olacaktır.

**4.4.2 Modüller arası Ortak Veriler**

Modüller arasında ilişkiler gösterilmiştir.

**4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri**

Ortak veriler için böyle bir parça kullanılmadı.

**4.4.4 Güvenlik Alt Sistemi**

Yazılım sistemlerinde güvenlik en önemli şeylerin başında gelir. Oluşturulan sistemde verilerin güvenliği açısından başta da belirtildiği gibi silme işlemleri sadece üst kullanıcıya yani admin kullanıcısına açıktır. Yazılım sistemlerinin güvenilirliğe ilişkin nicelikleri, kullanıcıların gereksinimlerini karşılayacak şekilde ortaya koymak ve güvenilirliğin hesaplanmasına yönelik verileri toplama, istatistiksel tahminleme, ölçütlerin tespiti, yazılıma ait mimari özelliklerin belirlenmesi, tasarım, geliştirme ve bunlara yönelik çalışma ortamının belirlenmesi ve modellenmesini kapsamaktadır.

**4.4.5 Veri Dağıtım Alt sistemi**

Veri dağıtımında sunucudan veri alış-verişi mevcut olacak. Yani programda yapılan işlemler veri tabanına iletilecek ve aynı zamanda istenildiği takdirde veri tabanında veri çekilebilecek.

**4.4.6 Yedekleme ve Arşivleme İşlemleri**

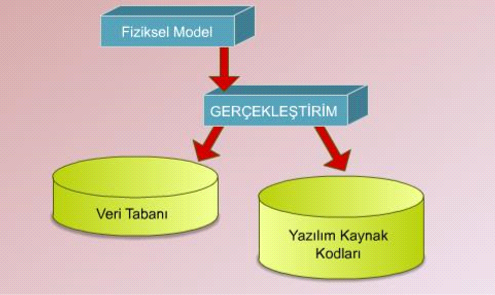
İşlemler sırasında verilerin herhangi bir şekilde zarar görmesi durumunda sistemin çalışmasında ciddi sorunlar oluşabilir. Bu durumun devam etmesi halinde depolanan verilerin sağlığı kullanıcı veya kurumlara çok ciddi zarar verebilir.

Bu nedenle yedekleme işlemi sistemin çalışmasına göre günlük olarak takip edilmelidir. Aksi takdirde büyük sorunlara yol açabilir.

|  |
| --- |
| * SİSTEM GERÇEKLEŞTİRİMİ |

**5.1 Giriş**

Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir. Her şeyden önce yazılımın geliştirilmesi için uygun ortam seçilmelidir.



**5.2 Yazılım Geliştirme Ortamları**

Yazılımın geliştirme ortamı, tasarım sonunda üretilen fiziksel modelin, bilgisayar ortamında çalışması için gerekli olan;

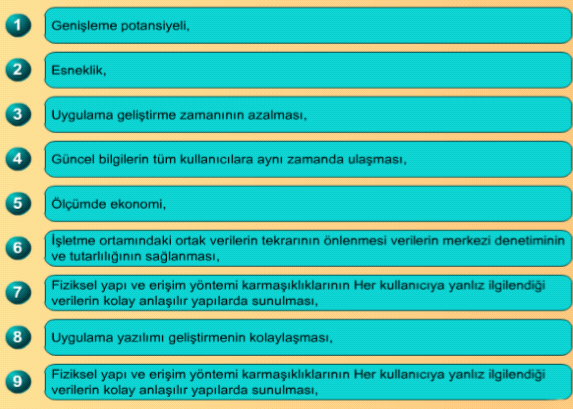
* Programla Dili
* Veri Tabanı Yönetim Sistemi

CASE araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

* **Programlama Dilleri**

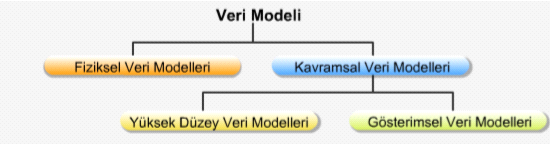
Sistemde kullanılan başlıca dil C# programlama dilidir. Sebebi ise C# dilinin güvenli bir dil olması. Ayrıyeten herhangi bir dil kullanılmamakta.

**5.2.2.1 VTYS Kullanımının Ek Yararları**



**Şekil 5.3 VTYS ’nin Ek yararları**

**5.2.2.2 Veri Modelleri**



**Şekil 5.4 Veri Modelleri**

Fiziksel Veri Modelinin yaptığı şey MYSQL de verilerin tablolar içindeki alanlarda saklamak.

Kavramsal Veri Modelinde ise iki ana başlık altında inceleriz.

Üçlü şema mimarisinde görülen yapıların kullanıcı gereksinimlerinden yola çıkılarak aşamalı bir şekilde fiziksel olarak gerçekleştirilmesidir.

* Gereksinimlerin Belirlenmesi
* Kullanılacak Veri Tipi
* Kullanılacak Veri grupları
* Sisteme konulacak kurallar
* Kavramsal Model
* Kullanıcıdan elde edilecek gereksinimler ile analiz çalışması ile bağlantılı verilerin gruplandırma yapılarak modellenmesi gerekmektedir. Bu modeli grafiksel olarak varlık bağıntısı seçenekleri ile gösterebiliriz.
* Mantıksal Model
* Veri tabanı tasarımlarımızın ilişkisel veri tabanı modelinde tablolar ile ifade edilebilmesi için yapılması gereken dönüşümü içerir.
* Fiziksel Model
* Fiziksel olarak sistemin kurulması sağlanır. Kullanılacak VTYS ile ilgili ilk temas burada kurulur.



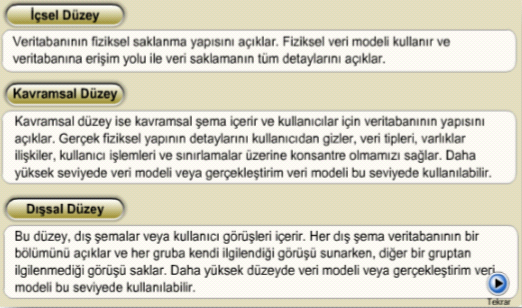
Şekil 5.6 Kullanıcı Şemaları

**5.2.2.3 Şemalar**

Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlandırılmaktadır. Ver tabanı şeması, tasarım sırasında belirtilir ve sıkça değişmesi beklenmez. Pek çok veri modeli şeması, belli gösterim biçimine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek örneğini göstermez.

**5.2.2.4 VTYS Mimarisi**

Veri Tabanı Yönetim Sistemi mimarisini üç başlık şeklinde ele aldık.



**5.2.2.5 Veri Tabanı Dilleri ve Arabirimleri**

Sistem veri tabanı dili olarak SQL kullandık. Arabirim olarak MYSQL kullandık.

**1.1.1.1 Veri Tabanı Sistem Ortam**

Veri tabanı sistemin oluştururken phpMyAdmin sistem ortamını kullandık. Veri tabanı işlemlerinin hemen hemen hepsini burada gerçekleştirdik.

**1.1.1.2 VTYS’nin Sınıflandırılması**

Bizim kullandığımız veri modeli en çok kullanılan veri modellerinden biri olan ilişkisel veri modelidir.

**1.1.1.3 Hazır Program Kütüphane Dosyaları**

Böyle bir şeye ihtiyaç yok.

**1.1.1.4 CASE Araç ve Ortamları**

Case aracı olarak DRAW. İO ve Microsoft Visio ürünlerini kullandık.

**1.2 Kodlama Stili**

Herhangi bir kodlama biçimine bağlı kalmadık. Kendi stilimizi kendimiz oluşturduk.

**5.3.1 Açıklama Satırlar**

Açıklama satırlar yapılan işlemlerin başına bu işlemin ne yaptığı hakkında kısa bilgiler olarak yazıldı.

**5.3.2 Kod Biçimlemesi**

Kod biçimlemesinde iç içe bir hiyerarşi kullandık.

**5.3.3 Anlamlı İsimlendirme**

Sistem kodlamasının genel yapısında kullanılan değişkenlerin veri tabanında karşılığı yine kodlamada kullanılan adla yapıldı ki tam anlamıyla karşılaştırılabilsin.

**5.3.4 Yapısal Programlama Yapıları**

Genel olarak 3 Başlıkta incelenirse;

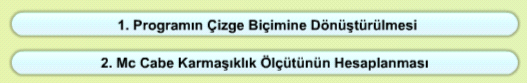
* **Döngü Yapıları:**
* Tıpkı ardışık işlemler gibi alt alta birkaç satır yazacağımıza tek bir döngüyle bu işlemleri kolayca hallettik
* **Koşullu İşlem Yapıları:**
* Bu yapıları ise neredeyse programın tamamında kullandık zaten karşılaştırma bulunan her yerde mevcut durumda.
* **Ardışık işlem Yapıları:**
* Bu yapılar ise genellikse fonksiyon gibi tekrarlı yapılarda tek bir seferde çözdük.

**1.3 Program Karmaşıklığı**

Programın karmaşıklığını ölçmek için McCabe karmaşıklık ölçütü kullanılmıştır.

Bu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu bundan esinlenmiştir.

McCabe ölçütü, bir programda kullanılan “koşul” deyimlerinin program karmaşıklığını etkileyen en önemli unsur olduğu esasına dayanır ve iki aşamada uygulanır.



**5.4.1 Programın Çizge Biçimine Dönüştürülmesi**

* **Olağan Dışı Durum Çözümleme**

Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.

**5.5.1 Olağandışı Durum Tanımları**

Olağan dışı durumlar için try-Catch blokları devrede hazır beklemektedirler.

**5.5.2 Farklık Olağandışı Durum Çözümleme Yaklaşımları**



Şekil 5.11 Olağan Dışı Halde Yapılacaklar

* **Kod Gözden Geçirme**

Kod gözden geçirme teslim sırasında çok önemli bir rol oynar ve şart olan bir durumdur. Program sınamadan ile kod gözden geçirmeyi birbirinden ayırmak gerekir. Program sına, programın işletimi sırasında ortaya çıkabilecek yanlış ya da hataları yakalamak amacıyla yapılır. Kod gözden geçirme işlemi ise, program hataların %3-5 oranında kesimi yakalanabilmektedir. Eğer programı yazan kişi, yazdığı programın hemen sonra bir kod inceleme sürecine girdi olacağını bilerek program yazdığında daha etkin, az hatalı ve okunabilir programlar elde edilebilmektedir.

**5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi**

Gözden geçirme sürecinin temel özelliklerini sıralayalım;

* Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır. Biçiminde özetlenebilir. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir.
* Hataların bulunması, ancak düzeltilmemesi hedeflenir,
* Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyici kullanılmasıdır. Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımı yapacak ekipten seçilmesinde yarar vardır.

**5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular**

Program incelenirken aşağıdaki soruların yanıtları aranır. Bunlara ek olaraktan sorular eklenebilir amaca bağlı olarak.

**5.6.2.1 Öbek Ara yüzü**

Öbekleri test etmek için belli sorular sorduk;

* Öbek tek giriş ve tek çıkış mı?
* Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?
* Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?
* Her öbek tek bir işlevsel amacı yerine getiriyor mu?

Şeklinde sorular sorduk.

**5.6.2.2 Giriş Açıklamaları**

Oluşturduğumuz giriş açıklamalarını test etmek için belli sorular sorduk;

* Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?
* Giriş açıklama satırları, Öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?
* Her paragrafı açıklayan kısa açıklamalar var mı?
* Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?
* Giriş açıklama satırları, çıktıları (parametre, kütük vb.) ve hata iletilerini tanımlıyor mu?
* Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?
* Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?
* Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?
* Giriş açıklama satırları, öbeğin amacını açıklıyor mu?

Şeklinde oldu.

**5.6.2.3 Veri Kullanımı**

Oluşturduğumuz veri kullanımlarını test etmek için belli sorular sorduk bu sorular;

* Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı?
* Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?
* İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?
* Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?

**5.6.2.4 Öbeğin Düzenlenişi**

* Modüller birleşimi uyumlumu?
* Modüller arası veri aktarımları sağlanıyor mu?
* Bütün parçalar birleştiğinde sistem çalışıyor mu? Gözden geçirme sırasında referans alınacak sorular olacaktır.

**5.6.2.5 Sunuş**

Artık son kısma gelindiğinde ise şu sorular soruldu;

* Öbek yapısı içerisinde akıllı “programlama hileleri” kullanılmış mı?
* Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?
* Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşıla bilirliği kolaylaştıracak biçimde anlamlı mı?
* Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?
* Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?

|  |
| --- |
| **6.DOĞRULAMA VE GEÇERLEME** |

**6.1 Giriş**

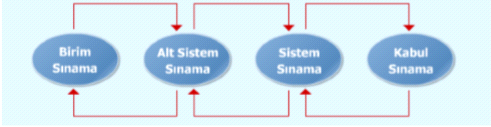
Geliştirilecek bilgi sistemi yazılımının doğrulanması ve geçerlemesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler;

* 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 belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru. Açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.



Şekil 6.1 Doğrulama Geçerleme

**6.2 Sınama Kavramları**



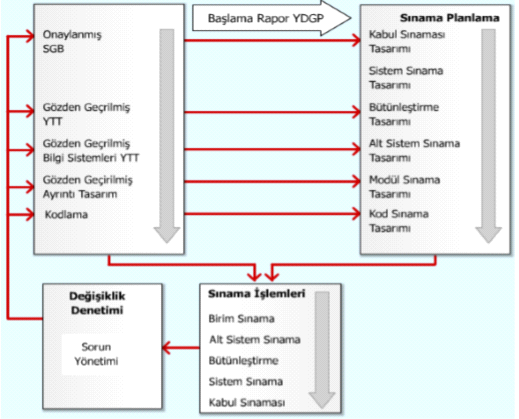
**Birim Sınama:** Sistemin birimleri teker teker sınandı ve sonuçları çıkarıldı.

**Alt Sistem Sınama:** Birimlerin birleşmesiyle parçalar oluşturulup bunların kendi içinde denemeleri yapıldı. Eksikler giderildi.

**Sistem Sınama:** Sistemin bütün olarak sınanması yapıldı ve programın eksiksiz olduğu onaylandı.

**Kabul Sınama:** Sistem ilk örnek çıkartılıp gerçek veriler girildi ve sorunsuz olduğu bir kez daha onaylandı.

**6.3 Doğrulama ve Geçerleme Yaşam Döngüsü**



Şekil 6.2 Yaşam Döngüsü

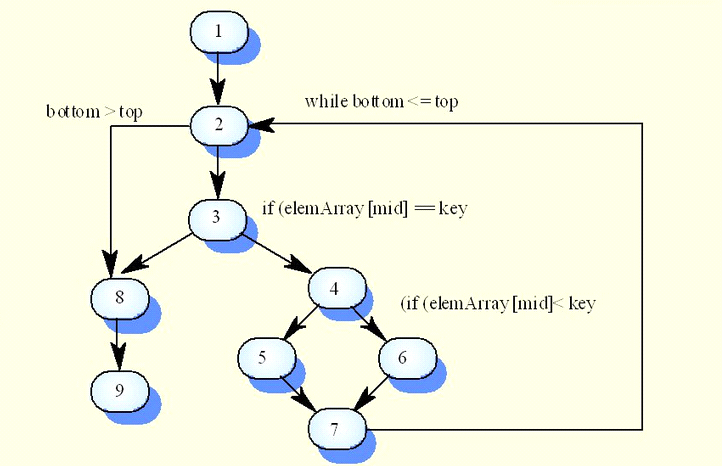
**6.4 Sınama Yöntemleri**

Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir “sonra” operasyonu olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken çaba türüdür.

**6.4.1 Beyaz Kutu Sınaması**

Denetimler arasında:

* İç veri yapılarının denenmesi yapıldı,
* Bütün döngülerin sınır değerlerinde sınanması,
* Bütün bağımsız yolların en azından bir kere sınanması,
* Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması.



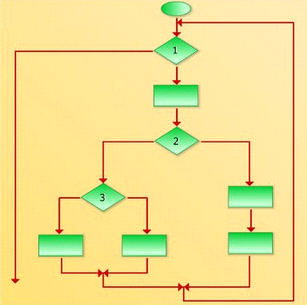
Şekil 6.3 Beyaz Kutu Sınaması

**6.4.2 Temel Yollar Sınaması**

Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün testidir. Sistem şartnamesinin gerekleri incelenir.

Kullanım Profili

* Rastgele Test
* Belgelenmiş özelliklere göre test
* Sonlu durum makinesi
* Karar tablosu
* Uç değerler analizi
* Eş değerlere bölme



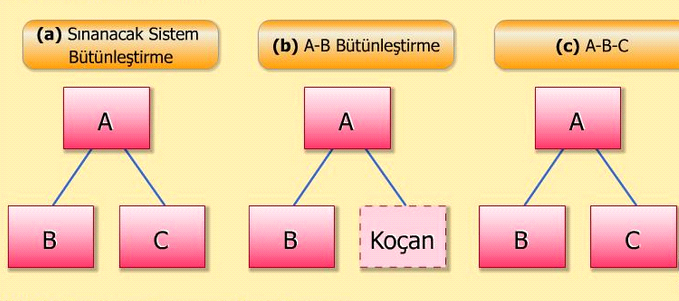
Şekil 6.4 Temel Yollar Sınaması Basit Şablon

**6.5 Sınama ve Bütünleştirme Stratejileri**

Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir. İşlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

**6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme**

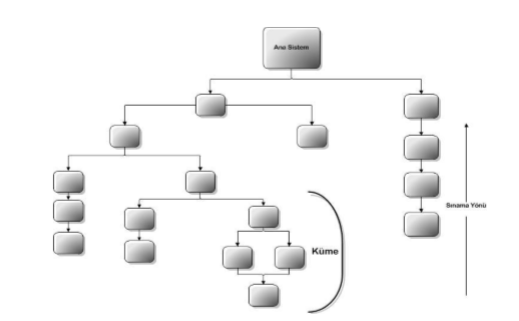
Önce sistemin en üst düzeylerinin sınanması ve daha sonra aşağı düzeyleri, ilgili parçaların takılarak sınanmaları söz konusudur. En üst noktadaki bileşen, bir birim/parça/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir. Ancak bu en üstteki bileşenin tam olarak sınanması için alttaki bileşenlerle olan bağlantılarının da çalışması gerekir. Genel hatlarıyla özetlemek gerekirse şu mantıkla sitem sınaması yapıldı.



Şekil 6.5 Yukarıdan Aşağı Sınama

**6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme**

Aşağıdan yukarıya bütünleştirme de ise, Yukarıdan aşağıya yönetiminin tersi bir uygulama yapılır. Önce en alt düzeydeki birimler sınanır ardından üstteki birimlere geçilip oradaki birimler sınanır.



Şekil 6.7 Aşağıdan Yukarı Bütünleştirme

**6.6 Sınama Planlaması**

Test raporu hazırlanırken şu özellikler mutlaka planda belirtilmelidir;

**Test Planı kimliği:** Test planının adı veya belge numarası

**Giriş:** Test edilecek yazılımın elemanlarının genel tanıtım özetleri. Ayrıca bu plan kapsamı ve başvurulan belgeler. Kısaltmalar ve terim açıklamaları bu bölümde bildirilmelidir.

**Test edilecek sistem:** Sistemde bileşenleri sürüm sayıları olarak sıralar ve sistemin özelliklerini bileşenlerini ve nasıl kullanıldıkları açıklanmalıdır. Ayrıca sistemde test edilmeyecek parçalar belirtilmelidir.

**Test edilecek ana fonksiyonlar:** Sistemin test edilecek ana fonksiyonlarının kısa bir tanıtımı yapılmalıdır.

**Test edilmeyecek ana fonksiyonlar:** Sistemde test edilmeyecek fonksiyonları ve bunların neden test edilmedikleri açıklanacaktır.

**Geçti/Kaldı Kriterleri:** Bir test sonucunda sistemin geçmiş veya kalmış sayılacağını açıklanmalıdır.

**Test dokümanı:** Test süresince yapılan işlemleri alınan raporları elde edilen bilgileri rapor içinde sunulmalıdır.

**Sorumluluklar:** Hangi kişilerin nelerden sorumlu olduğu ve test takım lideri bilgileri mutlaka raporda belirtilmelidir.

**Riskler ve Önlemler:** Test planında varsayılan ve olası yüksek riskli durumları belirtir ve bu durumların olması durumunda, etkilerinin en aza indirilebilmesi için alınması gereken önlemleri açıklar.

**6.7 Sınama Belirtimleri**

Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir.

Sınama verilerinin otomatik sınama verisi üreten programlardan yararlanılabilir. Çünkü elle hazırlanması zaman alıcı olabilir.

Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Sınama belirtimlerinin hazırlanmasındaki temel amaç, etkin sınama yapılması için bir rehber oluşturmasıdır.

**6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

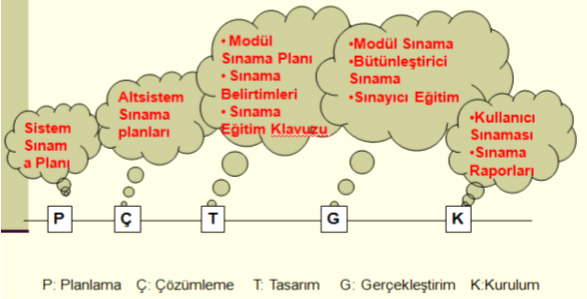
Planlama aşamasında genel planlama sınaması gerçekleştirilir. Bu olan tüm planların basit bir ön hazırlığı niteliğindedir. Çözümleme aşamasında sınama planı alt sistemler bazında ayrıntılandırılır. Tasarım aşamasında sınama plana detaylandırılır ve sınama belirtimleri oluşturulur. Bu oluşumlar daha sonra eğitim ve el kitabında kullanılır. Gerçekleştirim aşamasında teknik sınamalar yapılır sınama raporları hazırlanır ve elle tutulur ilk testler yapılır. Kurulum aşamasında sistemle ilgili son sınamalar yapılır ve sınama raporları hazırlanır.



**Şekil 6.8 Sınama Etkinlikleri**

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: BT projesinin 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.



|  |  |  |
| --- | --- | --- |
| Zaman  Gerçekleştirme-  Doğrulama | 2.HAFTA | 3.HAFTA |
| Gerçekleştirme |  | X |
| Doğrulama |  | X |

Tablo Gerçekleştirme aşama ve tarihleri

|  |
| --- |
| **7.BAKIM** |

**7.1 Giriş**

Sistemin tasarımı bitirdikten sonra bazı zaman aralıklarında sistemin bakıma sokulması gerekiyor. Bu sistem oldukça sağlam ve geç yıpranma yaşayacak bir sistem olduğu için bakım aşamasına gelmesi uzun süreler alabilir. Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı temel olarak alınmıştır.



Şekil 7.1 Bakım Aşamaları

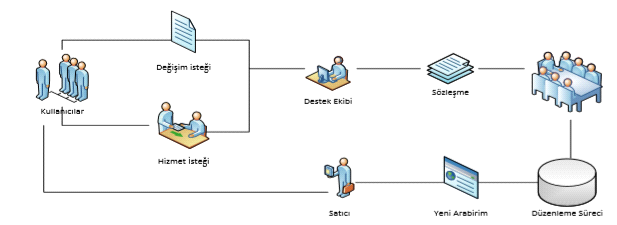
**7.2 Kurulum**

Sistem kurulumu oldukça basit, yalnızca uygulamanın yükleneceği firmanın veri tabanı kaydı uygulamayla bütünleştirip veri aktarımı yapılması gerekiyor. Bu işlemde oldukça basit olduğu için kurulum kısmı fazla üzerine durulacak bir şey değil.

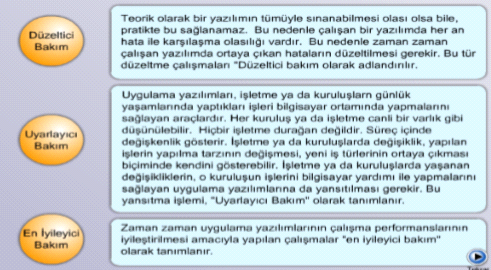
**7.3 Yerinde Destek Organizasyonu**

Bu konuda destek muhatabı benim. Yani olası bir hatada bizzat ben müdahale edeceğim.

**7.4 Yazılım Bakımı**



**7.4.1 Tanım**



**7.4.2 Bakım Süreç Modeli**

Bakım süreç modeli yukarıdaki yapılan işlemlerin tümünün baştan yapılması demektir. Bunları adım adım inceleyeceğiz.



Şekil 7.4 Bakım Süreç Modeli

**1.Adım:** Sorunu Tanımlama Süreci

İlk önce bakım ne için yapılır bu sorunun cevabına bakıyoruz.



Şekil 7.5 Sorun Tanımlama

**2.Adım:** Çözümleme Süreci

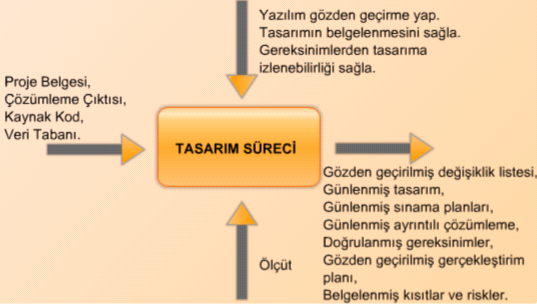
Sorun tanımlamadan çıkan karar doğrultusunda problemi kâğıt üzerinde çözelim.



Şekil 7.6 Çözümleme Süreci

**3.Adım:** Tasarım Süreci

Çözümlenen sistem sonucunda tasarımı güncelleştirme sırasında.



Şekil 7.7 Tasarım Süreci

**4.Adım:** Gerçekleştirim Süreci

Tasarımı yapılan sistemin gerçekleşme aşamasına gelelim.



Şekil 7.8 Gerçekleştirim Süreci

**5.Adım:** Sistem Sınama Süreci

Şimdi sistemin sınama sürecini tekrar ele alırsak;



Şekil 7.9 Sınama Süreci

**6.Adım:** Kabul Sınaması Süreci

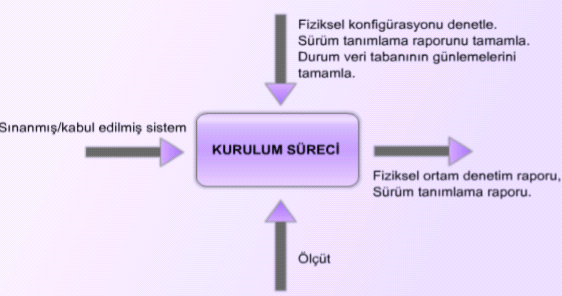
Müşteri karşısında sistemi sınadığımız zaman;



Şekil 7.10 Kabul Sınama Süreci

**7.Adım:** Kurulum Süreci

Sınamayı geçen sistem şimdi kurulum aşamasına giriyor.



7.11 Şekil Kurulum Süreci

|  |
| --- |
| **8.SONUÇ** |

Sonuç olarak sistem hayata geçtiği zaman neler olacak ne işe yarayacak bunları gözler önüne serdik.

Bu sistem sayesinde hem zaman hem maddi olarak kazanç sağlanacak.

Sistem sayesinde personel yorulmayacak ve farklı işlerde daha fazla güç sarf etme imkânı doğacak ki bu işletmenin yükselmesi için inanılmaz bir fırsat olarak görebiliriz.

|  |
| --- |
| 9.KAYNAKLAR |

İnternet Kaynakları

1. . http://www.kriptarium.com/pd.html

2. <http://muhammetbaykara.com/2017/04/05/ymh-114-ornek-taslak-proje/>

3. <https://en.wikipedia.org/wiki/Software_verification_and_validation>

4. https://stackoverflow.com/questions/31376177/parse-word-doc-docx-document

|  |
| --- |
|  |
|  |