

**Proje Ekibi**

**Kadir GÜRBÜZ**

***Ocak 2021-Versiyon 1.0***

**Fırat Üniversitesi Tez Analiz Otomasyonu**

**T.C.**

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Programlama Dilleri  
  
Proje Uygulamaları ve Dökümantasyonu**

**ÖNSÖZ**

Bu proje çalışmasında örnek olarak verilen bir tez dökümanının incelesini yapıp belli kurallara göre yazılıp yazılmadığını kontrol eden bir sistem yapmamız istenmiştir.

Öncelikle bu projede yardımcı kaynaklarıyla bizlere destek sağlayan Doç Dr. Fatih ÖZKAYNAK hocama teşekkürlerimi sunarım.

**Konya,2021**

**Kadir GÜRBÜZ**

|  |  |  |  |
| --- | --- | --- | --- |
| **1. GİRİŞ** | | |  |
|  | **1.1** | Projenin Amacı | 07 |
|  | **1.2** | Projenin Kapsamı | 07 |
| **2. PROJE PLANI** | | |  |
|  | **2.1** | Giriş | 07 |
|  | **2.2** | Projenin Plan Kapsamı | 08 |
|  | **2.2.1** | Proje Zaman-İş Planı | 12 |
|  | **2.3** | Önerilen Sistemin Teknik Tanımları | 13 |
|  | **2.4** | Proje Standartları, Yöntem ve Metodolojiler | 13 |
|  | **2.5** | Konfigürasyon Yönetim Planı | 16 |
|  | **2.6** | Kaynak Yönetim Planı | 16 |
|  | **2.7** | Eğitim Planı | 17 |
|  | **2.8** | Test Planı | 19 |
|  | **2.9** | Bakım Planı | 19 |
| **3. SİSTEM ÇÖZÜMLEME** | | |  |
|  | **3.1** | Mevcut Sistem İncelemesi | 19 |
|  |  | **3.1.1** Örgüt Yapısı | 19 |
|  |  | **3.1.2** İşlevsel Model | 20 |
|  |  | **3.1.3** Varolan Yazılım/Donanım Kaynakları | 29 |
|  |  | **3.1.4** Varolan Sistemin Değerlendirilmesi | 29 |
|  | **3.2** | Gereksenen Sistemin Mantıksal Modeli | 30 |
|  |  | **3.2.1** Giriş | 30 |
|  |  | **3.2.2** İşlevsel Model | 30 |
|  |  | **3.2.3** Genel Bakış | 30 |
|  |  | **3.2.4** Sınıf Diyagramı | 31 |
|  |  | **3.2.5** İşlevlerin Sıra Düzeni | 33 |
|  |  | **3.2.6** Aktivite Diyagramı | 34 |
|  |  | **3.2.7** Sequence Diyagramı | 35 |
|  |  | **3.2.8** Başarım Gerekleri | 35 |
|  | **3.3** | Arayüz (Modül) Gerekleri | 35 |
|  |  | **3.3.1** Yazılım Arayüzü | 35 |
|  |  | **3.3.2** Kullanıcı Arayüzü | 36 |
|  |  | **3.3.3** İletişim Arayüzü | 36 |
|  |  | **3.3.4** Yönetim Arayüzü | 36 |
|  | **3.4** | Belgeleme Gerekleri | 36 |
|  |  | **3.4.1** Geliştirme Sürecinin Belgelenmesi | 36 |
|  |  | **3.4.2** Eğitim Belgeleri | 36 |
|  |  | **3.4.3** Kullanıcı El Kitapları | 36 |
| **4. SİSTEM TASARIMI** | | |  |
|  | **4.1** | Genel Tasarım Bilgileri | 37 |
|  |  | **4.1.1** Genel Sistem Tanımı | 37 |
|  |  | **4.1.2** Varsayımlar ve Kısıtlamalar | 37 |
|  |  | **4.1.3** Dış Arabirimler | 38 |
|  |  | **4.1.3.1** Kullanıcı Arabirimleri | 38 |
|  |  | **4.1.3.2** Veri Arabirimleri | 38 |
|  |  | **4.1.4** Testler | 38 |
|  |  | **4.1.5** Performans | 38 |
|  | **4.2** | Veri Tasarımı | 38 |
|  |  | **4.2.1** Tablo tanımları | 38 |
|  |  | **4.2.2** Veri Tanımları | 38 |
|  |  | **4.2.3** Değer Kümesi Tanımları | 38 |
|  | **4.3** | Süreç Tasarımı | 39 |
|  |  | **4.3.1** Genel Tasarım | 39 |
|  |  | **4.3.2** Entegrasyon ve Test Gereksinimleri | 39 |
|  | **4.4** | Ortak Alt Sistemlerin Tasarımı | 39 |
|  |  | **4.4.1** Ortak Alt Sistemler | 39 |
|  |  | **4.4.2** Modüller arası Ortak Veriler | 39 |
|  |  | **4.4.3** Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri | 39 |
|  |  | **4.4.4** Güvenlik Altsistemi | 40 |
|  |  | **4.4.5** Veri Dağıtım Altsistemi | 40 |
|  |  | **4.4.6** Yedekleme ve Arşivleme İşlemleri | 40 |
| **5. SİSTEM GERÇEKLEŞTİRİMİ** | | |  |
|  | **5.1** | Giriş | 41 |
|  | **5.2** | Yazılım Geliştirme Ortamları | 41 |
|  |  | **5.2.1** Programlama Dilleri | 41 |
|  |  | **5.2.2** Veri Tabanı Yönetim Sistemleri | 41 |
|  |  | **5.2.2.1** Veritabanı Dilleri ve Arabirimleri | 42 |
|  |  | **5.2.2.2** Veri Tabanı Sistem Ortamı | 42 |
|  |  | **5.2.2.3** VTYS'nin Sınıflandırılması | 42 |
|  |  | **5.2.2.4** Hazır Program Kütüphane Dosyaları | 42 |
|  |  | **5.2.2.5** CASE Araç ve Ortamları | 42 |
|  | **5.3** | Kodlama Stili | 42 |
|  |  | **5.3.1** Açıklama Satırları | 42 |
|  |  | **5.3.2** Kod Biçimlemesi | 42 |
|  |  | **5.3.3** Anlamlı İsimlendirme | 42 |
|  |  | **5.3.4** Yapısal Programlama Yapıları | 42 |
|  | **5.4** | Program Karmaşıklığı | 43 |
|  |  | **5.4.1** Programın Çizge Biçimine Dönüştürülmesi | 43 |
|  |  | **5.4.2** McCabe Karmaşıklık Ölçütü Hesaplama | 43 |
|  | **5.5** | Olağan Dışı Durum Çözümleme | 43 |
|  |  | **5.5.1** Olağandışı Durum Tanımları | 43 |
|  |  | **5.5.2** Farklı Olağandışı Durum Çözümleme Yaklaşımları | 43 |
|  | **5.6** | Kod Gözden Geçirme | 44 |
|  |  | **5.6.1** Gözden Geçirme Sürecinin Düzenlenmesi | 44 |
|  |  | **5.6.2** Gözden Geçirme Sırasında Kullanılacak Sorular | 44 |
|  |  | **5.6.2.1** Öbek Arayüzü | 44 |
|  |  | **5.6.2.2** Giriş Açıklamaları | 44 |
|  |  | **5.6.2.3** Veri Kullanımı | 45 |
|  |  | **5.6.2.4** Öbeğin Düzenlenişi | 45 |
|  |  | **5.6.2.5** Sunuş | 45 |
| **6. DOĞRULAMA VE GEÇERLEME** | | |  |
|  | **6.1** | Giriş | 45 |
|  | **6.2** | Sınama Yöntemleri | 46 |
|  |  | **6.2.1** Beyaz Kutu Sınaması | 46 |
|  | **6.3** | Sınama ve Bütünleştirme Stratejileri | 46 |
|  |  | **6.3.1** Yukarıdan Aşağı Sınama ve Bütünleştirme | 46 |
|  |  | **6.3.2** Aşağıdan Yukarıya Sınama ve Bütünleştirme | 46 |
|  | **6.4** | Sınama Planlaması | 47 |
|  | **6.5** | Yaşam Döngüsü Boyunca Sınama Etkinlikleri | 47 |
| **7. BAKIM** | | |  |
|  | **7.1** | Giriş | 49 |
|  | **7.2** | Kurulum | 50 |
|  | **7.3** | Yerinde Destek Organizasyonu | 50 |
|  | **7.4** | Bakım Süreç Modeli | 50 |
| **8. SONUÇ** | | |  |
| **9. KAYNAKLAR** | | |  |

|  |  |
| --- | --- |
| **1. GİRİŞ** |  |

|  |  |
| --- | --- |
| **1.1** | **Projenin Amacı** |

Bu projenin temel amacı tez analizi sırasında insan gücünden ziyade bunu daha kolaylaştırıp tez içerisinde bulunan belli başlı kurallara uygun olup olmadığını kontrol etmek ve tez kontrol işlemini çok daha efektif bir hale getirebilmek. Süre, hız ve doğruluk açısından faydalar sağlayıp tezin gerekli analizleri için öok daha kısa zamanda bunu gerçekleştirebilmek ve insan hatalarını minimum düzeye çekebilmektir.

Tezlerin her üniversite için yazım kurallarında küçük bazı değişkenlik gösteren özellikleri vardır. Burada önemli olan unsurlardan birisi de bu kuralların dinamik olabilmesini sağlayabilmektir. Bunu sağlayabilmek ve tüm üniversiteler için bunu uygun hale getirebilmek gerekir. Projemde bunu da sağlamak üzere gerekli geliştirmeler yapılmaya devam etmekte. Çoklu dil desteği ile beraber üyelik sistemi ile daha önce yapılan analiz sonuçları da kayıt altına alınabilmesi için Asp.Net Core 3.1 Web Api ile haberleşme sağlayarak burada sonuçları Web ortamında göstermek üzere geliştirmeler devam etmektedir.

|  |  |
| --- | --- |
| **1.2** | **Projenin Kapsamı** |

Bu proje ile APA kuralları ile tez yazım kuralları hibrit olarak test edilip sonuçlar elde ediliyor. Bu kapsamda tez veya apa kurallarının kontrolünü sağlamak amacıyla bu kapsamda;

* Devlet Üniversiteleri
* Özel Üniversiteler
* Apa kuralları ile doküman kontrolü sağlayan işletmeler
* Proje ofisleri

Bu uygulamadan faydalanabilir.

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

|  |  |  |
| --- | --- | --- |
| **2.1** | **Giriş** |  |

Bu projede tez analizini yaparken belli kurallar çerçevesinde analizler yapıp sonuçları elde etmekteyiz. Pdf veya Word dosyası olarak test edilebilen tezin ilk olarak kaynakları belirleme işlemi yapılmaktadır. Pdf olarak gelen belgeler Word dosyasına çevriliyor. Word dosyası olarak gelen dosyalar ise Pdf dosyasına çevrilip burada hibrit bir analiz gerçekleştiriliyor.

Burada kaynaklar belirlenirken “KAYNAKLAR” olarak doküman içerisinde bir arama gerçekleştirip “İÇİNDEKİLER” başlığı bulunan ve içerisinde “…..” işaret bulunan sayfayı atlayarak yapmaktadır. Daha sonra “KAYNAKLAR” başlığının bulunduğu sayfada ise kaynak numaralarından ilk 3 tanesinin kontrolünü yapıp daha sonra bulduğu zaman “KAYNAKLAR” bölümünü kopyalamaya başlamaktadır. Burada döngü ile parantez içerisindeki kaynak numaraları aratılıp ta ki bir sonraki bulunamaya kadar sayfa kopyalama devam etmektedir. Bir sonraki sayfada kontrol yapılıp var ise kopyalama gerçekleşiyor, eğer yok ise döngü kırılıyor.  
Bu sayede kaynak sayısı da belirlenmiş oluyor. Daha sonra atıf durumları kontrol ediliyor. Fakat burada atıflar kontrol edilirken eğer 1 kaynak atfı 1 farklı sayfada 1 den fazla kez tekrar ediyorsa bunu 1 sayfa diye alıyor.  
Kaynakların tümünü ise ayırıp standartlara uygun olup olmadığını kontrol etmek üzere liste haline getiriyoruz.

Çıft tırnak arasında olan cümlelerin sayısını da belirterek sonuçlar kısmına ekliyoruz.

Son olarak kaynaklar için gerekli bilgiler raporlanmak üzere kayıt altına alınıyor ve kullanıcıya sunuluyor.

Pdf ve Word dosyasını aynı anda analiz etmemin sebebi ise Word dosyalarının analizi Pdf dosyası analizine göre çok daha yavaş olması ve kaynak bulma konusunda çok sınırlı kaynağa sahip olmasıdır. Word dosyası ile sadece iki satırdan az paragraf sayılarını belirtmek ve hangi kısımların paragraf olup olmadığını anlamak için kullandım. Burada iki satırdan az paragraf belirlerken o satırın;

* İki yana yaslı olup olmadığını
* Yazılarının tümünün büyük olup olmadığını
* Şekil, Tablo, Denklem,Trial veya Resim gibi ifadelerin olmadığını
* 50 karakterden fazla 200 karakterden az olup olmadığını
* Boş karakter saymaması için boş olmadığını

Özelliklerinin tespitini yapıp daha sonra paragraf ise ve bu özellikler sağlanıyor ise uzunluğa göre rapora eklemesi için gerekli işlemleri yaptım.

Önsöz bölümünü de aratıp ilk cümlesini kontrol edip “teşekkür”,”minnettar”,”teşekkürler” gibi kelimeleri dizi olarak arattım. Bunlardan birisi var ise ihlali raporladım.

İçindekiler kısmının ayırması ve kontrolü biraz zorlu oldu denilebilir. Burada “İÇİNDEKİLER” ile başlayıp “ÖZGEÇMİŞ” ile biten bir kopyalama işlemi gerçekleştirdim. Daha sonra satırları okurken “….” İşaretine kadar kopyalama işlemi ve sondan başlayıp sayfa numarasını da” …” işaretine kadar kopyaladım ve sayfa numarasını ve başlığını kayda aldıktan sonra başlıkların tümünü sırayla arayıp bulundukları sayfanın son 10 karakterinde sayfa numaralarını arattım. Bulundukları sayfanın bir önceki ve bir sonraki sayfalarını da kontrol ederek sayfa numarasının olmaması halinde sağlayan kayıtları yanlış raporlamanın önüne bir nebze olsun geçebildim. Burada 84 sayfalık bir doküman da 1 tane hata yaparak analizi bitirdim. Bunun sebebi ise sayfa tutarlı olmasına rağmen bir önceki ve bir sonraki sayfada da sayfa numarası olmamasıydı.

Şekiller ile Tablolarda yaptığım işlemler bir birine çok benzer işlemlerdi. Burada “Şekiller Tablosu” başlığını aratıp “…” ve sayfa numaralarını kontrol edip daha sonra bulununca kopyalamaya başladım. Diğer sayfada kaydın devam edip etmediğini ise satırın ilk 5 karakterinde “Şekil ” yazısı var mı diye kontrol ettim. Yok ise döngü sonlanıyor. Bulundukları sayfa tutarlılığını da “İÇİNDEKİLER” tablosuna benzer kopyalama metodu kullandım.

Denklem numaraları ile başlık uyumunu “İÇİNDEKİLER” analizine bağlı olarak gerçekleştirdim. Burada soyutlamayı biraz ihlal ettim fakat performans artışı için bunu yaptım. Aynı şeyi Tablolar ve Şekiller analizinde de yaptım. İçindekiler kısmında ana başlık numaralarını alıp “Denklem ” yazısı sayfada var ise “Denklem “+ana\_basliklar[i]+”” olarak döngü içinde kontrol ettim. Eğer içindekiler sayfa numarasına göre uyumlu olmayan kayıt varsa onu raporladım.

Giriş bölümü kopyalanıp raporlama bölümünde aynı şekilde gösterildi.  
Son olarak güncellemeler devam etmektedir. En kısa sürede eksikler kapatılıp web ortamında kullanıma hazır olacaktır.

Denklem analizi

İki satır analizi

Çift tırnak analizi

Tutarlılık analizi

Tablo ve Şekil analizi

Kaynaklar analizi

İçindekiler analizi

Önsöz analizi

|  |  |  |
| --- | --- | --- |
| **2.2** | **Projenin Plan Kapsamı** |  |

Bu projenin amacı yönetimi kolay analizler yapmak, bu analizleri daha sonra kolay bir şekilde tekrar analiz, doğrulama, geçerleme ve analiz süresini kısaltma amacıyla yapılmıştır. Gözden kaçan denetimlerin daha hızlı yapılabilmesi, modüler ve basit bir yapıda olması sağlanmaya çalışılmıştır.

Raporlama

Geçerleme

Doğrulama

Kurallar sağlama

Gerekli analizler

Tezler

**→ Tez analizi sistemi neden gereklidir?**

* Sistem içerisinde daha hızlı işlemler yapabilme.
* 100 sayfalık bir tezin kuralları sağlama durumunu En fazla 2 dakika içerisinde analiz edebilme
* Analizde gözden kaçmaları engelleme
* Çok daha tutarlı analizler yapabilme
* Tez analizinde doğruluğu ve gerçekliği sağlamayı kolaylaştırabilme
* Zamandan tasarruf edip çok daha fazla tezi kolayca tasnifleyebilme

Bu ve buna benzer bir çok nedenden dolayı gerekli bir sistemdir.

Hızlı

Düzenli

Tez analizi sistemi

Efektif

Zaman tasarrufu

Kolay

Tez analizi sistemi

Giriş organizasyon-kapsam analizi

Denklem-başlık uyum analizi

Çift tırnak sayısı analizi  
<50

Başlık analizi  
Sayfa numarası analizi

İlk cümlede teşekkür analizi

Başlık analizi  
Sayfa numarası Analizi

Kaynak analizi  
Atıf analizi

Kaynaklar belirleme

İçindekiler belirleme

Önsöz belirleme

Tablo-Şekil belirleme

Çift tırnak belirleme

Denklem belirleme

Giriş bölümü belirleme

Raporlama

**Kütüphane bilgi sistemi nasıl çalışır?**

* Tez Pdf yada Word formatında analiz için hazırlanır fakat belli kuralları vardır.
* Bazı başlık kuralları şöyledir
  + Kaynaklar başlığı “KAYNAKLAR” olmalı ve kaynak numaraları [1],[2].. formatında olmalıdır.
  + İçindekiler başlığı “İÇİNDEKİLER” olmalı ve sonunda “ÖZGEÇMİŞ” başlığı olmalı ve başlıklar ile sayfa numaraları arasında “.” İşareti ile hiza takip çizgisi olmalı. Başlık numaraları ile başlık arasında 1 boşluk olmalı.
  + Önsöz bölümü başlığı “ÖNSÖZ” olmalı.
  + Tablolar başlığı “TABLOLAR LİSTESİ” şeklinde olmalı ve buradaki kurallar içindekiler ile aynıdır.
  + Şekiller başlığı “ŞEKİLLER LİSTESİ” şeklinde olmalı ve buradaki kurallar içindekiler ile aynıdır.
  + Denklem numaraları ise “Denklem no” şeklinde olmalı.
* Paragraflar iki yana yaslı olmalıdır.
* İçindekiler, Tablolar, Şekiller listelerinde iki satıra fazlaca yer vermek çokta iyi değildir. Fakat analizler doru olarak yapılır.
* Sayfa numaraları atlama yapılmadan verilmeli.

**Maliyet kestirimi dökümanı**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ölçüm parametresi** |  | **Ağrılık faktörü** | | |  |  |
| **Yalın** | **Ortalama** | **Karmaşık** |
| **Kullanıcı girdi sayısı** |  | 5 | 6 | 10 |  |  |
| **Kullanıcı çıktı sayısı** |  | 7 | 8 | 15 |  |  |
| **Kullanıcı sorgu sayısı** |  | 7 | 5 | 10 |  |  |
| **Dosya sayısı** |  | 5 | 4 | 10 |  |  |
| **Dışsal arayüz sayısı** |  | 3 | 2 | 5 |  |  |
|  |  |  |  |  |  |  |

**Temel karmaşıklık sorusu**

|  |  |
| --- | --- |
| **Temel karmaşıklık sorusu** |  |
|
| 1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu? | 2 |
| 2. Veri iletişimi gerekiyor mu? | 1 |
| 3. Dağıtık işlem işlevleri var mı? | 2 |
| 4. Performans kritik mi? | 3 |
| 5. Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak? | 2 |
| 6. Sistem, çevrim içi veri girişi gerektiriyor mu? | 2 |
| 7. Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu? | 2 |
| 8. Ana kütükler çevrim-içi olarak mı günleniyor? | 1 |
| 9. Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı? | 2 |
| 10. İçsel işlemler karmaşık mı? | 2 |
| 11. Tasarlanacak kod, yeniden kullanılabilir mi olacak? | 2 |
| 12. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı? | 2 |
| 13. Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor? | 2 |
| 14. Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak? | 1 |
|  | 26 |

**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=616,07

İN = AİN x (0.65 x 0.01 x TKF)

İN=616,07\*(0,65\*0,01\*48)

**İN=104,1**

Satır Sayısı = İN \*30

**SATIR SAYISI=104,1 \*30=3123 SATIR YAKLAŞIK**

Etkin Maliyet Modeli – COCOMO

* Organik proje: 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

Bizim projemiz küçük ekip tarafından geliştirilen organik proje kategorisine giriyor.

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

Geliştirme Süresi (Ay) = D = c x (E)d

Eleman Sayısı = E / D

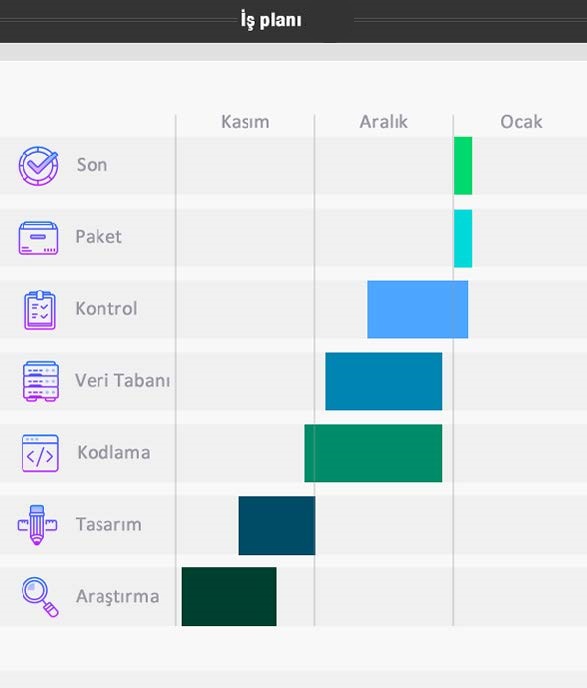
Formülde verilen değişkenler şöyle:

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\*3,121,05=**7,93**

Geliştirme Süresi =D= 2,5 x = 1 ay yaklaşık



* 1. **Önerilen Sistemin Teknik Tanımları**

**Kullanılan teknoloji**

* Gerekli kontrol kütüphaneleri
* Pdf okuma için gerekli kütüphaneler
* Word okuma için kütüphaneler
* Pdf-Word çevirimi için kütüphaneler
* C# kullanılarak geliştirildi.

|  |  |  |  |
| --- | --- | --- | --- |
| Çözümleme ve tasarım araçları | Programlama araçları | Sınama araçları | Destek araçları |
| * Smart Draw 2020 Professional * IBM Rational Software Article * Adobe Photoshop CC 2018 * Microsoft Visual Studio 2017 Professional * Pichon * Devexpress * Telerik | * C#   Sonrası için   * Javascript * Bootstrap * Web api * Asp.net core | * Microsoft Windows 8.1 * Microsoft Windows 10 | * Devexpress Components * Telerik Components |

* 1. **Proje Standartları, Yöntem ve Metodolojiler**

Spiralin başladığı ilk çeyrek içinde ilk isterler toplanır ve buna göre proje planlaması yapılır. İkinci çeyrekte, ilk tanımlanan isterlere göre risk çözümlemesi yapılır. Üçüncü çeyrekte, risk çözümlemesi sonunda ortaya çıkan isterlerin tanımlanmasındaki belirsizlikleri ortadan kaldırmak için prototipleme yöntemi kullanılır. Gerekirse benzetim(simülasyon) veya diğer modelleme kullanılarak isterlerin daha sağlıklı tanımlanması sağlanır. Dördüncü çeyrekte, müşteri, ortaya çıkan ilk ürünü inceleyerek değerlendirme yapar, önerilerde bulunur. Bu şekilde tenımlanan ilk döngü bir sonraki döngü için bir girdi oluşturur.

|  |  |  |  |
| --- | --- | --- | --- |
| Aşama | Kullanılan Yöntemler/Araçlar | Ne için kullanıldığı | Çıktı |
| Planlama | •Veri akış şemaları •Süreç belirtimleri •Görüşme •Maliyet kestirim yöntemleri •Proje yönetim araçları | •Süreç İnceleme •Kaynak Kestirimi •Proje Yönetimi | Proje planı |
| Çözümleme | •Veri Akış Şemaları •Süreç Belirtimleri •Görüşme •Nesne İlişki Şemaları •Veri Sözlüğü | •Süreç çözümleme •Veri çözümleme | Sistem çözümleme raporu |
| Çözümlemeden  tasarıma geçiş | •Akışa Dayalı Çözümleme •Süreç Belirtimlerinin Çözümlemeden Program Tasarım Diline Tasarıma Geçiş Dönüştürülmesi •Nesne İilişki Şemalarının Veri Tablolarına Dönüştürülmesi | •Başlangıç tasarım •Ayrıntılı tasarım •Başlangıç veri tasarımı | Başlangıç tasarım raporu |
| Tasarım | •Yapısal Şemalar •Program Tasarım Dili  •Veri Sözlüğü | •Genel tasarım •Ayrıntılı tasarım •Veri tasarımı | Sistem tasarım raporu |

Proje standartları yukarda belirtildiği gibidir. Bunu yanı sıra kullanılan sistem modelinde ise çağlayan model kullanılmıştır.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ekip Aylar | Ekim | | Kasım | | Aralık | |
| Sistem çözmleyici |  |  |  |  |  |  |
| Sistem yöneticisi |  |  |  |  |  |  |
| Sistem tasarımcısı |  |  |  |  |  |  |

|  |
| --- |
| * Ekonomi |
| * Tamlık |
| * Yeniden kullanılabilirlik |
| * Etkinlik |
| * Bütünlük |
| * Güvenirlik |
| * Modülerlik |
| * Belgeleme |
| * Kullanılabilirlik |
| * Temizlik |
| * Değişebilirlik |
| * Geçerlik |
| * Esneklik |
| * Genellik |
| * Sınanabilirlik |
| * Taşınabilirlik |
| * Bakılabilirlik |
| * Anlaşılabilirlik |
| * Birlikte çalışabilirlik |

Projedeki kalite sağlama planımız yukardaki tabloda da belli olduğu üzere;

1.**Ekonomi:** Ekonomik açıdan yazılımın maliyeti her ne kadar ilk seferde pahalı olsa da ileriye dönük düşünüldüğünde ve zaman tasarrufundan ötürü gayet uygundur.

2.**Tamlık**: Projede herhangi bir açık olmamalı ve programda bulunan tüm butonlar textler vs. çalışır ve tamdır.

3.**Yeniden Kullanılabilirlik**: Otomasyon her koşulda tekrardan düzenlenip kullanılabilecek.

4.**Etkinlik**: Kullanıcı sistemin her alanına hakim olduğu için sistemi etkin bir biçimde kullanacak.

5.**Bütünlük**: Sistemin tüm kısımlarına hakim olacak ve program bir bütün halinde çalışacaktır.

6.**Güvenilirlik**: Otomasyon gerekli güvenlik önlemlerinin alınması çalışmaları daha üst seviyelere çıkarılacaktır.

7.**Modülerlik**: Modülerlik otomasyonun her seviyesindeki kişinin ayrı ayrı sayfalardan söz sahibi olmasını sağlar. Örneğin: Analiz modülü, Giriş Modülü…

8.**Belgeleme**: Bu belgeden de anlaşılacağı üzere tam anlamıyla sistemin özeti olacak.

* 1. **Konfigürasyon Yönetim Planı**

Sistemin ilerde kullanıcının yeni istemlerini karşılayamaması veya sistemin yapısındaki bazı bileşenlerin değişmesi sonucu güncelliğini kaybettiğinde olası konfigürasyon planı hazırlandı.

* Analize yeni özellikler eklemek için gerekli işlemlerin yapılması
* Herhangi bir sebepten dolayı ayrılan uyumsuzluk giderme
* Sistemde herhangi bir istenmeyen durum halinde,

Durumları için konfigürasyon yönetim planı oluşturuldu.

* 1. **Kaynak Yönetim Planı**

Mevcut bir kaynağımız olmadığından kaynak olarak sadece bu proje dokümantasyonu var.

**Kaynak yönetiminde şu hususlar dikkate alınacaktır;**

* Yeterli hata bulunuyor mu?
* Kalite beklendiği gibi mi?
* Eğer yeterli sayıda hata bulunmuyorsa veya Kalite beklenenden daha iyi görünüyorsa, Kalite gerçekten daha iyi olabilir.
* Bu projede yeni bir süreç iyileştirmesi kullanılıyor mu?
* Proje elemanları yeni bir eğitim aldı mı?
* Yeni bir araç kullanılıyor mu?
* Yeterli vakit harcanıyor mu?
* İnceleyiciler yeterli hazırlık yapıyor mu?
* İnceleme toplantısında çözüm bulmak için zaman kaybediliyor mu?
  1. **Eğitim Planı**

Projeden kazanılacak en önemli olaylardan biride eğitimdir. Kullanılacak dillerin arayüz editör ve programların kullanımında hakim olunamaması halinde bu program başarıyla neticelendirilemez. Bu yüzden projede bazı eğitimler alınması gereklidir.

Proje kapsamında alınacak olan eğitimler;

* Adobe Photoshop CC 2018
* C#
* Devexpress
* SQL Dil Eğitimi (Güncelleme sonrası için)
* Matplotlib (Güncelleme sonrası için)
* XML(Güncelleme sonrası için)
* Telerik

Gereken eğitimlerdir.

|  |  |
| --- | --- |
| **Konu** | **Süreç** |
| **Giriş , Açıklamalar ve ön bilgi** | **1. gün** |
| **Temel Kavramlar ve Sistem için gerekli bileşenlerin tanıtılması** | **2. gün** |
| **Gerekli yöntem ve metodolojilerin nasıl kullanılacağı** | **3. gün** |
| **Gerekli Hata çözme yöntemlerinin incelenmesi** | **4. gün** |
| **Değerlendirme ve Sonuç** | **5. gün** |

Proje teslimi sonrası sistemi kullanacak olan kişiler arasından seçilen yetkili kişilere proje kullanımı olarak 1 günlük bir seminer verilecektir. Bu seminerin içeriği şu şekildedir;

* Sisteme genel bakış
* Sistemde anketin nasıl tanımlanacağı
* Sistemde yetkili yöneticilerin yetkileri
* Sistemin kısıtları açıklanarak nasıl kullanıldığına dair bilgiler verilmektedir

**Eğitimlerin zamana göre çizelgesi;**

|  |  |  |
| --- | --- | --- |
|  | **Başlama Tarihi** | **Bitiş Tarihi** |
| **• Adobe Photoshop CC 2018** | **May.24** | **Eyl.15** |
| **• C#** | **May.24** | **Eyl.15** |
| **• Devexpress** | **May.24** | **Eyl.15** |
| **• SQL Dil Eğitimi** | **May.24** | **Eyl.15** |
| **• Matplotlib** | **May.24** | **Eyl.15** |
| **• XML** | **May.24** | **Eyl.15** |

Bu şekilde bir eğitim planı uygulanmaktadır.Sistem tam olarak hazırlandığında eğitim planıda

garanti olarak ayarlanacaktır ve ona göre eğitim semineri verilecektir.Zira verilmemesi halinde kullanıcılar sistemi tam olarak kullanamayaca ve bazı açıklıklar meydana gelecektir.

* 1. **Test Planı**

Proje test ekipleri ve görevleri şu şekildedir;

* Fırat Üniversitesi bölgesindeki görev yapacak.
* Gerekli resmi kısımlarda ise resmi olarak ekip yapıları yapılacak.
* Bu ekiplerin test aşamasında yapacağı işler aşağıdaki gibidir; Başlama Tarihi Bitiş Tarihi tüm eğitimleri verilecektir.
* Bu ekiplerden kullanıcı sadece gidip imzasını kullanacak. Ön hazırlık olarak ise üniversitemizden başlamak koşuluyla gerekli hiyerarşi takip edilecek ve pilot tez analizi uygulaması gerçekleştirilecektir.
  1. **Bakım Planı**

Projenin bakım planına gelecek herg ün kullanılacak bu sistem tüm değişim ve bazı durumlarda kullanıcı eklenip çıkarılacak tüm bu sistemsel değişiklikler bakım planında yapılacaktır.

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

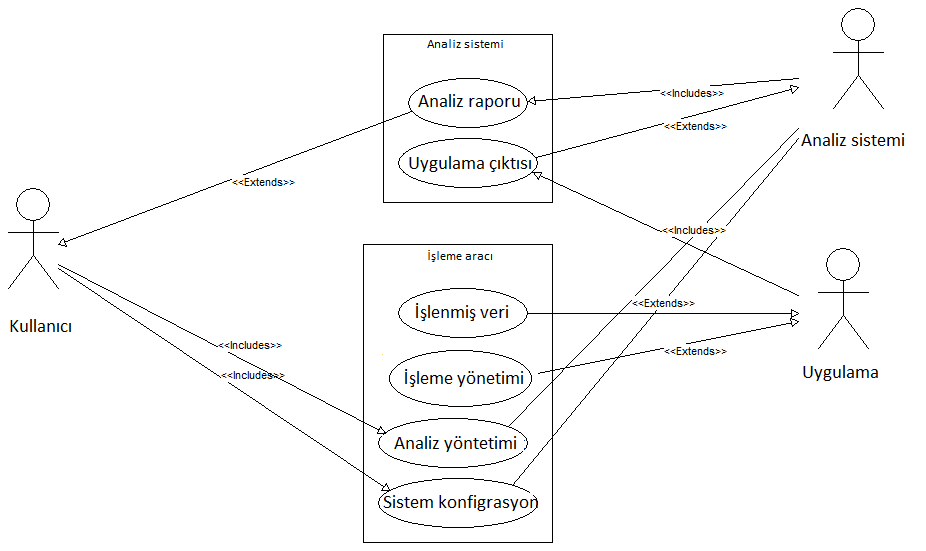
* 1. **Mevcut Sistem İncelemesi**

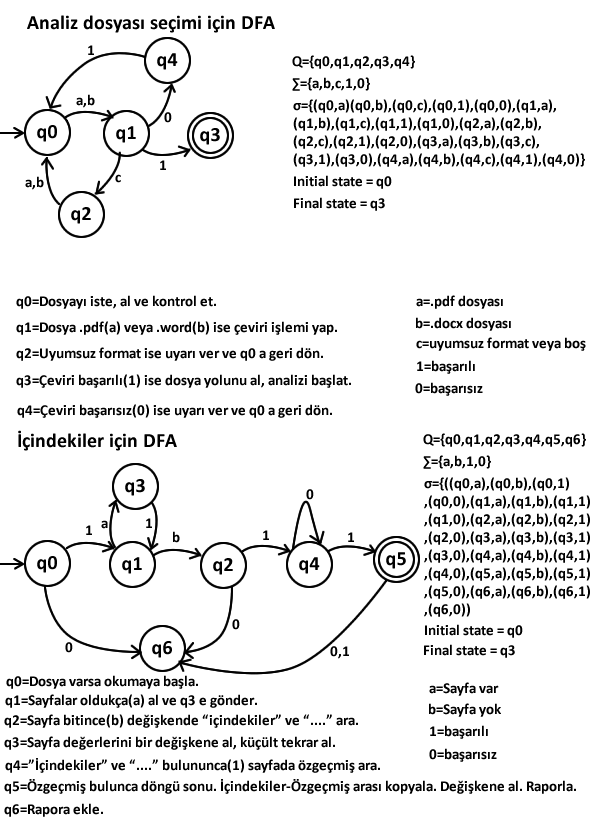
Mevcut sistem incelemesi öncelikle üniversitemizde olacağından çok zor olacaktır. Bunun yanı sıra internet üzerinden bulunan yöntemlerle de çözümler üretilecektir.

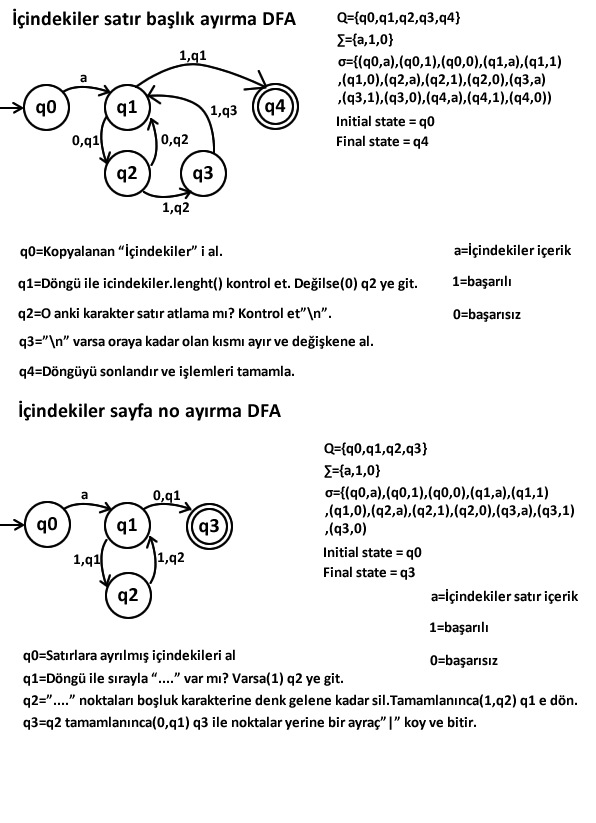
**3.1.1 Örgüt Yapısı**

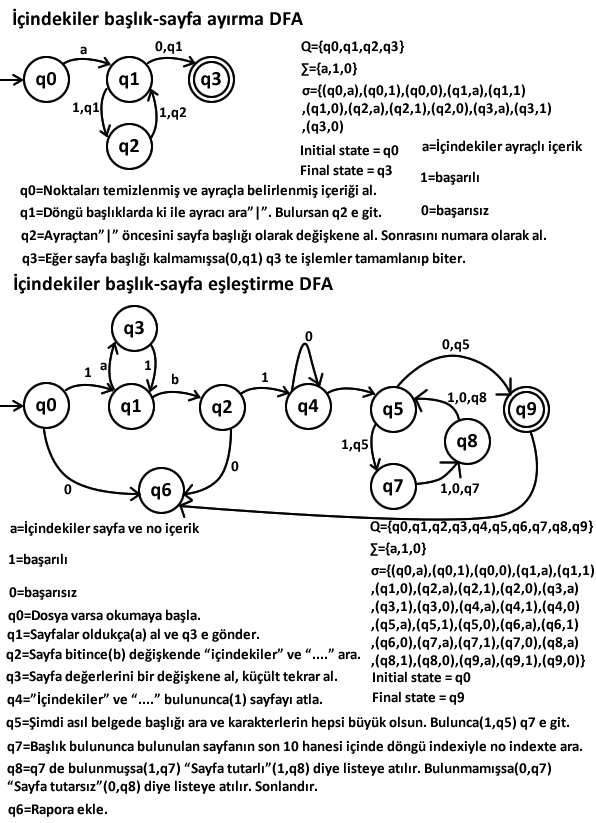
Örgüt yapısı olarak projeyi oluşturan, yöneten ve tüm gerekleri hazırlama şahsıma aittir.

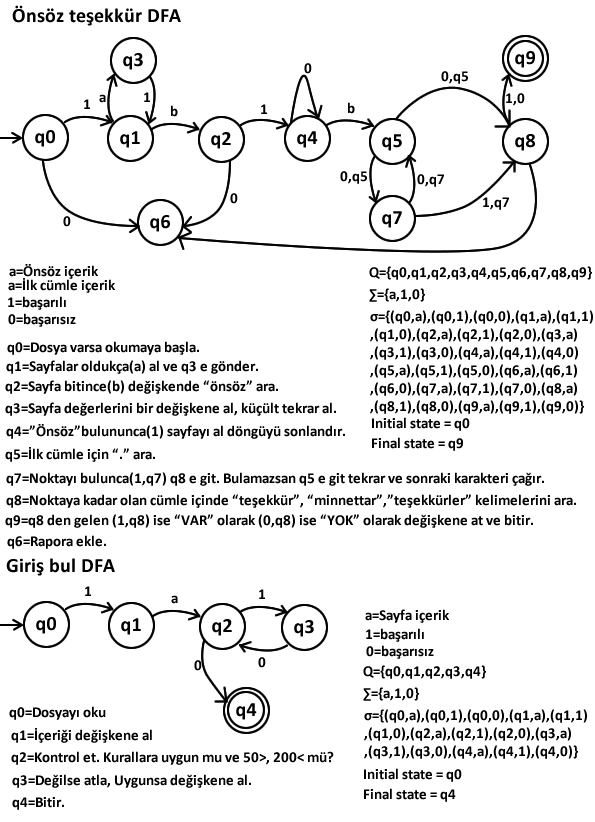
**3.1.2 İşlevsel Model (Use-Case)**

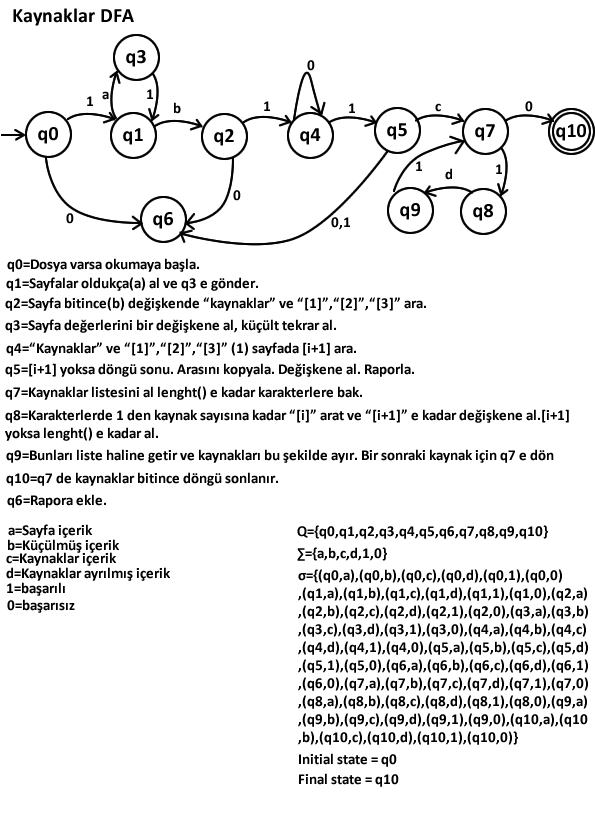
****

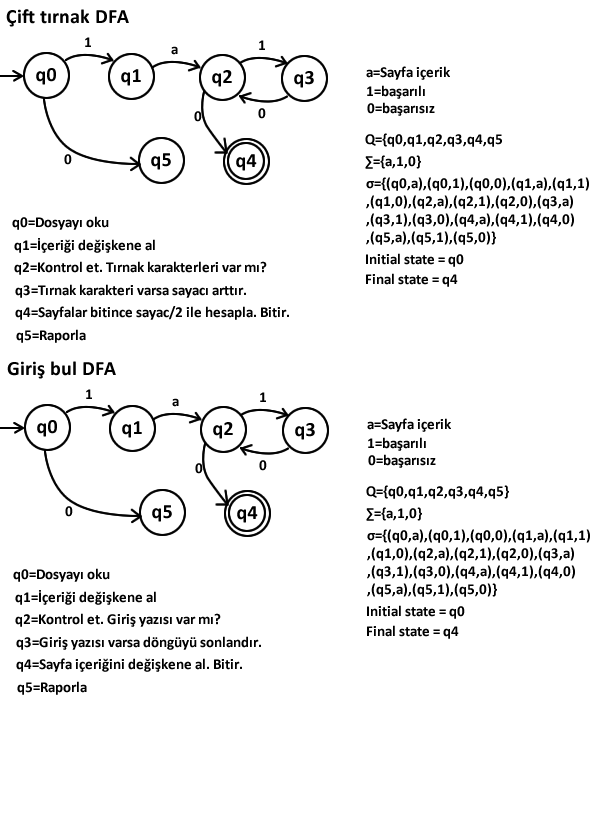
****

****

****

****

****

****

**Temel görevlendirme**



Visio ile yapılmıştır.

**3.1.3 Varolan Yazılım/Donanım Kaynakları**

* Adobe Photoshop CC 2018
* C#
* Devexpress
* SQL Dil
* XML
* Telerik
* Bilgisayar
* Visual Studio 2019
* Word



Visio ile yapılmıştır

**3.1.4 Varolan Sistemin Değerlendirilmesi**

Kullanıcıların şifrelerini yani güvenlik sözcüğünü unuttukları vakit cep telefona sms gelecek tir.

Bunun gibi avantajlar sunarak kullanıcıya daha verimli bir çalışma sağlamak amaçlarımız arasındadır.

**3.2 Gereksenen Sistemin Mantıksal Modeli**

**3.2.1 Giriş**

Mevcut sistemler incelendiğinde sonuca giden yolda epeyce bir eksikler ve resmi olmayan durumlar söz konusu artık bu sistemi Türkiye standartlarına uydurmak bize kalıyor.

**3.2.2 Genel Bakış**

Genel hatlarıyla sistemi inceleyecek olursak mevcut kütüphane yönetim sisteminde mevcut olan tüm olaylar burada da var döküman yönetim sistemiyle kıyaslayacak olursak sonuçlara hiçbir şekilde Admin dahil müdahale edilemiyor. Bunun yanı sıra olayın akış şekli USE-CASE diyagramında mevcuttur.

**3.2.3 Bilgi Sistemleri/Nesneler**

**ADMİN:** Yapının en üstünde bulunan ve en yetkili yapılanmadır.

**KULLANICI:** Yapının ikinci nesnesidir .Analiz isteminde bulunur.

**ARŞİV:** Dosyaların elektronik ortamda saklanarak evrak azalımına olanak veren bir stok nesnesidir. Gerekli bilgileri admin e göndermekle sorumludur. İkinci planda olduğu için Use-Case diyagramında işleme almadık.

** 3.2.4 Sınıf diyagramı**

Visio ile yapılmıştır.

**Kitap alış modeli**



**İşleyiş diyagramı**



**3.2.5 İşlevlerin Sıradüzeni**

Burada kaynaklar belirlenirken “KAYNAKLAR” olarak doküman içerisinde bir arama gerçekleştirip “İÇİNDEKİLER” başlığı bulunan ve içerisinde “…..” işaret bulunan sayfayı atlayarak yapmaktadır. Daha sonra “KAYNAKLAR” başlığının bulunduğu sayfada ise kaynak numaralarından ilk 3 tanesinin kontrolünü yapıp daha sonra bulduğu zaman “KAYNAKLAR” bölümünü kopyalamaya başlamaktadır. Burada döngü ile parantez içerisindeki kaynak numaraları aratılıp ta ki bir sonraki bulunamaya kadar sayfa kopyalama devam etmektedir. Bir sonraki sayfada kontrol yapılıp var ise kopyalama gerçekleşiyor, eğer yok ise döngü kırılıyor.  
Bu sayede kaynak sayısı da belirlenmiş oluyor. Daha sonra atıf durumları kontrol ediliyor. Fakat burada atıflar kontrol edilirken eğer 1 kaynak atfı 1 farklı sayfada 1 den fazla kez tekrar ediyorsa bunu 1 sayfa diye alıyor.  
Kaynakların tümünü ise ayırıp standartlara uygun olup olmadığını kontrol etmek üzere liste haline getiriyoruz.

Çıft tırnak arasında olan cümlelerin sayısını da belirterek sonuçlar kısmına ekliyoruz.

Son olarak kaynaklar için gerekli bilgiler raporlanmak üzere kayıt altına alınıyor ve kullanıcıya sunuluyor.

Pdf ve Word dosyasını aynı anda analiz etmemin sebebi ise Word dosyalarının analizi Pdf dosyası analizine göre çok daha yavaş olması ve kaynak bulma konusunda çok sınırlı kaynağa sahip olmasıdır. Word dosyası ile sadece iki satırdan az paragraf sayılarını belirtmek ve hangi kısımların paragraf olup olmadığını anlamak için kullandım. Burada iki satırdan az paragraf belirlerken o satırın;

* İki yana yaslı olup olmadığını
* Yazılarının tümünün büyük olup olmadığını
* Şekil, Tablo, Denklem,Trial veya Resim gibi ifadelerin olmadığını
* 50 karakterden fazla 200 karakterden az olup olmadığını
* Boş karakter saymaması için boş olmadığını

Özelliklerinin tespitini yapıp daha sonra paragraf ise ve bu özellikler sağlanıyor ise uzunluğa göre rapora eklemesi için gerekli işlemleri yaptım.

Önsöz bölümünü de aratıp ilk cümlesini kontrol edip “teşekkür”,”minnettar”,”teşekkürler” gibi kelimeleri dizi olarak arattım. Bunlardan birisi var ise ihlali raporladım.

İçindekiler kısmının ayırması ve kontrolü biraz zorlu oldu denilebilir. Burada “İÇİNDEKİLER” ile başlayıp “ÖZGEÇMİŞ” ile biten bir kopyalama işlemi gerçekleştirdim. Daha sonra satırları okurken “….” İşaretine kadar kopyalama işlemi ve sondan başlayıp sayfa numarasını da” …” işaretine kadar kopyaladım ve sayfa numarasını ve başlığını kayda aldıktan sonra başlıkların tümünü sırayla arayıp bulundukları sayfanın son 10 karakterinde sayfa numaralarını arattım. Bulundukları sayfanın bir önceki ve bir sonraki sayfalarını da kontrol ederek sayfa numarasının olmaması halinde sağlayan kayıtları yanlış raporlamanın önüne bir nebze olsun geçebildim. Burada 84 sayfalık bir doküman da 1 tane hata yaparak analizi bitirdim. Bunun sebebi ise sayfa tutarlı olmasına rağmen bir önceki ve bir sonraki sayfada da sayfa numarası olmamasıydı.

Şekiller ile Tablolarda yaptığım işlemler bir birine çok benzer işlemlerdi. Burada “Şekiller Tablosu” başlığını aratıp “…” ve sayfa numaralarını kontrol edip daha sonra bulununca kopyalamaya başladım. Diğer sayfada kaydın devam edip etmediğini ise satırın ilk 5 karakterinde “Şekil ” yazısı var mı diye kontrol ettim. Yok ise döngü sonlanıyor. Bulundukları sayfa tutarlılığını da “İÇİNDEKİLER” tablosuna benzer kopyalama metodu kullandım.

Denklem numaraları ile başlık uyumunu “İÇİNDEKİLER” analizine bağlı olarak gerçekleştirdim. Burada soyutlamayı biraz ihlal ettim fakat performans artışı için bunu yaptım. Aynı şeyi Tablolar ve Şekiller analizinde de yaptım. İçindekiler kısmında ana başlık numaralarını alıp “Denklem ” yazısı sayfada var ise “Denklem “+ana\_basliklar[i]+”” olarak döngü içinde kontrol ettim. Eğer içindekiler sayfa numarasına göre uyumlu olmayan kayıt varsa onu raporladım.

Giriş bölümü kopyalanıp raporlama bölümünde aynı şekilde gösterildi.

**3.2.6 Aktivite diyagramı (Activity diagram)**



**3.2.7 Sequence diyagramını**



Raporla

Rapor oluştur

Göster

Analizi onayla

Analizi sağla

Kullanım kontrol

Analiz tamamla

Analiz onayla

Kullanıcı onayla

Detay

Format onayla

Belge mevcut

Belge kontrol

**3.2.8 Başarım Gerekleri**

Mevcut sistemler incelendi ve mevcut sistemin eksiklerinden yola çıkılarak, sistemin başarımı için

* Sistemin sonuç üretim doğrulukları
* Tepki sürelerinin en aza indirilmesi
* Mali külfetin azaltılması
* Hile hata ve yanlışlıkların en aza indirilmesi
* Kullanım kolaylılığı
* Anlaşılabilirlik
* Tarafsızlık

temel gereklilikler olarak tespit edilmiştir.

* 1. **Arayüz (Modül) Gerekleri**

**3.3.1 Yazılım Arayüzü**

Projenin çalışması esnasında böyle bir açık verilmemesine özen gösterildi. Gerekli olan her türlü değişiklik seurce kodları üzerinden yapılıp tekrar derlenecek.

**3.3.2 Kullanıcı Arayüzü**

Projede kullanıcının arayüzü tasarlanırken herhangi bir şekilde renkler seçilerek tarafsız rahat büyük puntolu yazılı bir arayüz tasarlanacaktır.



bilgi

veri

veri

veri

bilgi

**3.3.3 İletişim Arayüzü**

İletişimle ilgili olarak iletişim modülümüzü yazacağız lakin üniversite ile yapılan anlaşmada bu modül konmayacak denirse çünkü bu onların hakkı olası bir kötü durumda muhatap biz olmamamız açısından koyulmayabilir.

**3.3.4 Yönetim Arayüzü**

Projenin %90 ı genelde yönetimsel arayüzlerden oluşacak sadece personelin göreceği ekran kullanıcı arayüzüne girecek dışındaki her yer yönetimsel arayüz olacak. Yöneticinin ekrana girdiğinde karşılaşacağı arayüzdür.İki arayüz vardır:Biri Kullanıcı Arayüzü diğeri ise yönetimsel arayüzdür.Yani bu arayüz admin tarafından sisteme girişlerde gerçekleşecektir.

**3.4 Belgeleme Gerekleri**

**3.4.1 Geliştirme Sürecinin Belgelenmesi**

Geliştirme sürecinde genel olarak belgelendirilmesi hem ileriye dönük hem de şimdiki geliştirme sürecinde projenin tamamlanma yüzdesini nerede kalınıp nerelerde eksikler olduğunu genel hatlarıyla göstermesi amacıyla yapıldı. Bunun yanı sıra projeye yeni dahil olan personellerin olaya hakimiyeti açısından bu yönteme başvuruldu.

**3.4.2 Eğitim Belgeleri**

Mevcut bir belgemiz bulunmamaktadır.

**3.4.3 Kullanıcı El Kitapları**

Bu kısma projenin en son safhasında kullanıcılara verilecek eğitimlerden pilot uygulamalardan yola çıkılarak hazırlanacak.Yani proje sonunda rahat ve kolay kullanımdan dolayı bir eğitim semineri ve bir kullanım kitapcığı hazırlanacaktır.

|  |
| --- |
| **4. SİSTEM TASARIMI** |

**4.1 Genel Tasarım Bilgileri**

**4.1.1 Genel Sistem Tanımı**

**İşlevsel belirtimler**

**(Mantıksal model)**

**Gereksinimler**

**Tasarım**

* **Gereksinimler**

Gereksinimler kısmında üniversitede bir anket düzenlenecek ve ona göre bir tasarım oluşturmaya başlanacaktır.Çünkü her ne kadar da Teknoloji Çağı’nda yaşasakda bilgisayarı açıp kapamayı zor yapan insanlar var.

* **İşlevsel Belirtimler**

Bu kısımda ilk önce sistemin ne yapacağı sorusuna cevap verelim. Sistem Türkiye de kâğıt üstünde yapılan ve birçok külfeti olan sistemin dijital ortama aktarılması diye özetlenebilir. Seçim öncesinde gerekli hiyerarşiler gene dijital ortamda yapılarak imza ortamı oluşturulacak ve personel imzasını atacak,dosya ve evrak paylaşımı yapacaktır.

İkinci olarak yazılım ne yapacağı sorusuna cevap vermek gerekirse, yazılım ilk aşamada Admin ile her kullanıcı arasındaki iletişimi sağlayacak. Bundan sonra imzalama sırasında kimlik kontrolü yapmasını ve personelin rahat bir şekilde ortam sağlayacak.

* **Tasarım**

Süreç tasarımı,arayüz tasarımı,yapısal tasarım ve veri tasarımı olacaktır.

1. Veri akış diyagramı
2. Süreç belirtimi
3. Kontrol belirtimi
4. Durum geçiş diyagramı
5. Veri nesnesi belirtimi
6. Nesne ilişkisi diyagramı
7. Veri sözlüğü

Yapılacaktır.

**4.1.2 Varsayımlar ve Kısıtlamalar**

Sistemde varsayılan değerler bulunmamakta bunun yanı sıra başlıca kısıtlamalar şunlardır:

* Kullanıcı olması
* Dosya paylaşımını sağlaması
* Daha önce bu konuda eğitilmiş olması
* Dosya içeriğinin gizli tutulması

**4.1.3 Dış Arabirimler**

**4.1.3.1 Kullanıcı Arabirimleri**

Kullanıcı arabirimlerin ilk başında sistem giriş ekranı bulunacak. Ve her birimin kendine ait ekranları olacak. Kullanıcı girişi bu arabirimde amaçlanmıştır.Kullanıcı bilgilerini girerek bu sisteme giriş yapmış olacaktır.

**4.1.3.2 Veri Arabirimleri**

Veri arabirimlerinde sistem çalışacağından veritabanından kayıt yazıp okurken arada XML ve C#-Devexpress-Telerik arabirimleri olacaktır. Veriler,dosyalar,evraklar bunlar veri halinde iletim sağlanacaktır. Eskiden var olan dosya ve evraklarda optik olarak okutulup arşivde tutulacaktır.

**4.1.4 Testler**

Genel hatlarıyla testlerimiz iki aşamada gerçekleştirilecek. Bilinen adıyla pilot bölge uygulaması yapılacak.

**Alfa Aşaması:** Sistemin geliştirildiği yerde kullanıcıların gelerek katkıda bulunması sistemi test etmesi ile yapılacak.

**Beta Aşaması:** Kullanıcı, geliştirilen sistemi kendi yerleşkesinde, bir gözetmen eşliğinde yapılacak.

**4.1.5 Performans**

Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecek

Sistemin Tasarıma Uygunluk Performansı;

Tasarımı yapılan sistemin stabilitesi ve işleyiş performansı değerlendirilecek.

Veri Yapısının Sistemle Performansı;

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

**4.2 Veri Tasarımı**

**4.2.1 Tablo tanımları**

Sistem 5 tablodan oluşmaktadır. Bunun yanı sıra kullanıcının ekstra istekleri doğrultusunda ek tablolara açık bir sistem tasarlanmaktadır. İlk olarak sistemi kullanacak her varlığın insan olduğunu kabul edersek bu insanları tek bir tabloda tutmak en mantıklısı olacaktır böyle bir sistem ile ANA tablosunu yaptık. Bunun yanı sıra her kuruluş için ayrı tablo yapıldı ve kuruluşların içindeki en yetkili kişi tablonun ikincil anahtarı oldu ve ANA tablosu ile ilişkilendirildi.

**4.2.2 Veri Tanımları**

Veri tipi olarak integer(int) kullanılmasının amacı sayısal değerleri almaktan ötürüdür. String olarak kullanılan değerler kelime içeren değerleri tutacağından ötürü kullanıldı. Boolean veri tipi işte işin can alıcı noktası burası boolean veri tipi iki faklı değer alır ya truedir ya false eğer ki görevlendirme varsa true olur.

**4.2.3 Değer Kümesi Tanımları**

**Sayım, Sonuc:** Kullanılanlar doğrultusunda oluşacak olan değerdir.

**Yetki:** Yetki kişinin derecesine göre değişen bir değerdir.

**4.3 Süreç Tasarımı**

**4.3.1 Genel Tasarım**

Genel olarak analiz tasarımı oluşturulup daha sonra geliştirmeler yapıldı.

**4.3.2 Entegrasyon ve Test Gereksinimleri**

Sistemimizin daha öncede belirttiğimiz gibi veri tabanlarıyla entegrasyon halinde olması gerekmez. Bunun yanı sıra pilot bölgedeki test süresince 1.derece yazılım yetkilileri orda bulunmalı ve sistem testini gözlemleyip notlar çıkartmalı.

**4.4 Ortak Alt Sistemlerin Tasarımı**

**4.4.1 Ortak Alt Sistemler**

Analiz sistemi ortak bir sistemdir.

**4.4.2 Modüller arası Ortak Veriler**

Modüller arasında ilişkili veri modelinde olduğu üzere ortak veriler mevcuttur.

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

Raporlar bir başka sistemde derlenip ortak olarak sunulmaktadır.

**4.4.4 Güvenlik Alt sistemi**

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.

1. Model seçme ve düzenlemeye yönelik faaliyetlerin temelinde uygun hedeflerin tespit edilmesi bulunmaktadır.

2. Hata ve aksaklıkların analiz edilmesi için uygun verilerin tanımlanması gerekmektedir. Örneğin, arıza veya hataları önemine göre sınıflandırmak, hatalar arası ortalama süreyi bulmak, hata nedenlerini araştırmak, hataları bulmaya yönelik test verilerine karar vermek.

3. Belirtilen hedeflere yönelik veriler modellenir.

4. Geçmişe yönelik verilerin zaman bilgilerini de içerecek şekilde elde edilerek yazılım geliştirme sürecine dâhil etmek.

5. Yazılım geliştirme sürecinin modellenmesi, hata ile karşılaşılıp, test sürecine başlamak ve model doğrulama işlemlerine gerçekleştirmek.

6. Güvenilirlik tahminleme modelinin seçilmesini sağlamak.

7. Güvenilirlik modeli tarafından kullanılacak olan parametrelerini tespit etmek.

8. Verilen bir noktayı kullanarak gelecekteki olası hatalar hakkında tahmin yapmak.

9. Tahmin edilen hata ve arıza oranları ile gerçekleşen değerleri kıyaslamak.

**4.4.5 Veri Dağıtım Altsistemi**

Veri dağıtımından ziyade veri alma üzerine bir sistem kuruldu. Dağıtılan sistemlerde zararlı kişilerin müdahalesi kolay olmaktadır bunun yanı sıra bizim sistemimizde terminaller serverlardan veri çekecek serverlar terminallere veri göndermeyecek.

**4.4.6 Yedekleme ve Arşivleme İşlemleri**

Depolanan verilerin, herhangi bir nedenle zarar görmesi, sistemin çalışma süreçlerinde ciddi zararlar oluşturabilir. Yaşanabilecek bir felaket durumu sonrasında, depolanan verilerin geri yüklenememesi, sistemin sağlandığı kullanıcılara veya kurumlara çok ciddi zararlar verebilir.

Bu nedenle sistemin çalışma süreçlerine bağlı olarak, yedekleme sistemleri kurulmalı ve yedekleme işlemleri günlük olarak takip edilmelidir.

Yedekleme sistemlerinin kurulumu; yedeklenecek veri miktarı, yedekleme sıklığı, yedeklenen verinin zaman içerisinde değişme oranı ve maksimum veri kaybı gibi parametrelere bağlıdır.

Sistemin birden fazla sunucusunun eş zamanlı yedekleme işlemini yapabilmesi, işletim sistemlerinin kayıt dosyalarını tam ve eş zamanlı olarak yedekleyebilmesi ve işletim sistemleri üzerinde çalışan veri tabanı uygulamasının yedeklerini sistem kapatılmadan alabilmesi gerekmektedir.

|  |
| --- |
| **5. 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. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

**5.2 Yazılım Geliştirme Ortamları**

Yazılım geliştirme ortamı, tasarım sonunda üretilen fiziksel modelin, bilgisayar ortamında çalıştırılabilmesi için gerekli olan:

* Programlama Dili
* Hazır Program Kitapçıkları

CASE Araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

**4.2.1 Programlama Dilleri**

Sistemde kullanılan başlıca programlama dillerini daha öncelerde de yazmıştık fakat burada bir kez daha dile getirelim. Kendi içlerinde sınıflandırılan bu diller arasından bizim seçtiğimiz sınıf şüphesiz ki veri işleme yoğunluklu uygulamalarda kullanılacak dillerden olacaktır. Bu yüzden daha çok görsel programlamaya önem verilecek kullandığımız diller arasında C# başlıcadır.

**4.1.2 Veri Tabanı Yönetim Sistemleri**

Devexpress ile hazırladığımız sistem ara yüz burada ki kullanıcı/programcının kullandığı uygulama kısmına denk geliyor.

C# ise veriye erişen yazılım oluyor.

**1. Gereksinimlerin belirlenmesi**

* Veri tipleri
* Veri grupları
* Veriler ile ilgili kurallar
* Veriler üzerinde yapılması gereken işlemler

**2. Kavramsal Model**

Kullanıcıdan elde edilecek gereksinimler ile ilgili bir analiz çalışmasının yapılması ve birbiriyle bağlantılı verilerin gruplanarak bir düzenleme içinde modellenmesi gerekmektedir. Bu modeli grafiksel olarak varlık bağıntı seçenekleri ile gösteririz.

**3. Mantıksal Model**

Veri tabanı tasarımlarımızın ilişkisel veritabanı modelinde tablolar ile ifade edilebilmesi için yapılması gereken dönüşümü içerir.

**4. Fiziksel Model**

Fiziksel olarak sistemin kurulması sağlanır. Kullanılacak VTYS ile ilgili ilk temas burada kurulur.

**4.1 Şemalar**

Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak önemlidir. Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlandırılır. Veri tabanı şeması, tasarım sırasında belirtilir ve sıkça değişmesi beklenmez. Pek çok veri modeli şemaları, diyagramlar halinde göstermek için belli gösterim biçimlerine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek örneğini göstermez

**5.2.2.1 Veritabanı Dilleri ve Arabirimleri**

Sistemimizde veritabanı dili olarak SQL kullanılabilir.

**5.2.2.2 Veri Tabanı Sistem Ortam**

Tüm Yükleme, yedekleme, performans ölçme, sıralama, veri sıkıştırma, ve benzeri fonksiyonları yerine getirmek amacıyla Sql Server Management Studio ortamını kullanılabilir.

**5.2.2.3 VTYS'nin Sınıflandırılması**

En fazla kullanılan veri modelleri ilişkisel, ağ, hiyerarşik, nesne-yönelimli ve kavramsal modellerdir. Bizim Kullandığımız ise ilişkisel veri modelidir.

**5.2.2.4 Hazır Program Kütüphane Dosyaları**

Zaten entegre olduğu için böyle bir şeye ihtiyaç duymadık.

**5.2.2.5 CASE Araç ve Ortamları**

Case araçları olarak ise Microsoft un Visio Project ürünlerini kullandık. Bunun yanı sıra Microsoft Visio yazılımı yazılımını kullandık.

**5.3 Kodlama Stili**

Kendimize has kodlama bicini kullandık herhangi bir hazır düzene bağlı kalmadık Bakım programcımıza da aynı stil üzerine eğitim verdik ve sorunları ortadan kaldırdık.

**5.3.1 Açıklama Satırları**

Açıklama satırları karmaşık her modül için yapıldı.

**5.3.2 Kod Biçimlemesi**

Kod biçimlemesine değinmek gerekirse alt alta oluşan kodlarda tabi indexleri kullandık ve iç içe bir biçimde hiyerarşi oluşturduk.

**5.3.3 Anlamlı İsimlendirme**

Sistem kodlamasının genel yapısında kullanılan değişkenler için anlamlı isimlendirme yaptık.

**5.3.4 Yapısal Programlama Yapıları**

Genel olarak 3 başlıkta incelersek:

* **Ardışık işlem yapıları:** Bu tür yapılarda genellikle fonksiyon, altprogram ve buna benzer tekrarlı yapıları tek bir seferde çözdük.
* **Koşullu işlem yapıları:** Bu yapıları ise neredeyse programın tamamında kullandık karşılaştırma yapılan her yerde bunlara yer verildi.
* **Döngü yapıları:** Tıpkı ardışık işlemler gibi alt alta birkaç satır yazıcığımıza tek bir döngüyle bu sorunların üstesinden geldik.

**5.4 Program Karmaşıklığı**

Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür. Bu bölümde bu ölçüt anlatılmaktadır. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir.

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

**5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama**

**k = 13** Kenar sayısı **d = 9** Düğüm sayısı **p = 1** Bileşen sayısı

**V(G)=k-d+2p (Formülüyle bulunur)**

V(G)= k – d + 2p

VG= 30 – 25+2\*31

= 67

**5.5 Olağan Dışı Durum Çözümleme**

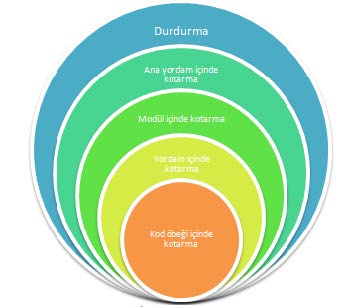
Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.

**5.5.1 Olağandışı Durum Tanımları**

Olağandışı gelişen durumlarda try-catch blokları devreye girecek ve program kırılmadan çalışmasına devam edebilecek şekilde tasarladık.

**5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları**

Tüm olağan dışı durumlarda program kırılmadan hata mesajlarıyla tekrar başa dönecek şekilde tasarladık.



**5.6 Kod Gözden Geçirme**

Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeden okunabilir bir program yazamaz. Hiçbir yazı editörün onayını almadan basılamayacağı gibi hiçbir program da incelenmeden, gözden geçirilmeden işletime alınmamalıdır. Kod gözden geçirme ile program sınama işlemlerini birbirinden ayırmak gerekir.

Program sınama, programın işletimi sırasında ortaya çıkabilecek yanlış ya da hataları yakalamak amacıyla yapılır. Kod gözden geçirme işlemi ise, programın kaynak kodu üzerinde yapılan bir incelemedir. Kod gözden geçirmelerinde program hatalarının %3-5 oranındaki 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 özellikleri;

* 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.
* 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.

**5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular**

Bir program incelenirken, programın her bir öbeği (yordam ya da işlev) aşağıdaki soruların yanıtları aranır. Bu sorulara ek sorular eklenebilir. Bazı soruların yanıtlarının "hayır" olması programın reddedileceği anlamına gelmemelidir.

**5.6.2.1 Öbek Arayüzü**

Oluşturduğumuz öbekleri test etmek için belli sorular sorduk bu sorular:

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

Şeklinde oldu.

**5.6.2.2 Giriş Açıklamaları**

Oluşturduğumuz giriş açıklamalarını test etmek için belli sorular sorduk bu sorular:

* Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?
* Giriş açıklama satırları, öbeğin amacını açıklıyor mu?
* Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?
* Giriş açıklama satırları, çıktıları (parametre, kütük vb) ve hata iletilerini tanımlıyor mu?
* Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?
* Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?
* 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?

Şeklinde oldu.

**5.6.2.3 Veri Kullanımı**

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

* İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?
* Değişken adları,işlevlerini yansıtacak biçimde anlamlı mı?
* Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?
* Her değişken tek bir amaçla mı kullanılıyor?
* Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?
* Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

Şeklinde oldu.

**5.6.2.4 Öbeğin Düzenlenişi**

* Modüller birleşimi uyumlumu?
* Modüller arası veri aktarımları sağlanıyor mu?
* Bütün modüller birleştiğinde sistem çalışıyor mu?

Gözden geçirme sırasında referans alınacak sorular olacaktır.

**5.6.2.5 Sunuş**

Artık son kısma gelindiğinde ise şu sorular soruldu:

* Her satır, en fazla bir deyim içeriyor mu?
* Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?
* Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?
* Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?
* Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?
* Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

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

**6.1. Giriş**

Geliştirilecek bilgi sistemi 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 biçiminde özetlenebilir.



**Birim Sınama:** Sistemin birimleri olan kütüphaneler içerisinde sırasıyla kendi içlerinde birimleri sınandı ve sonuçları çıkartıldı.

**Alt Sistem Sınama:** Birimlerin birleşmesiyle modüller oluşturulup bunların kendi içinde sınaması yapıldı. Genel olarak arayüzde ki eksiklikler giderildi.

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

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

**6.2 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 bir çaba türüdür.

**6.2.1 Beyaz Kutu Sınaması**

Denetimler arasında:

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

**6.3 Sınama ve Bütünleştirme Stratejileri**

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

**6.3.1 Yukarıdan Aşağı Sınama ve Bütünleştirme**

Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur. En üst noktadaki bileşen, bir birim/modül/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.

**6.3.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme**

Aşağıdan yukarı bütünleştirmede ise, önceki yöntemin tersine uygulama yapılır. Önce en alt düzeydeki işçi birimleri sınanır ve bir üstteki birimle sınama edilmesi gerektiğinde bu üst bileşen, bir '**sürücü**' ile temsil edilir. Yine amaç, çalışmasa bile arayüz oluşturacak ve alt bileşenin sınanmasını sağlayacak bir birim edinmektir. Fakat bu sınama sistemi kullanılmadı.

**6.4 Sınama Planlaması**

Bir tablo ile özetlemek gerekirse şu şekilde özetleyebiliriz.

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.

**6.5 Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

Bütün bu etkinlikleri bir hiyerarşi altında incelemek gerekirse:

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

Bu ayrıtılar temel olarak:

* sınanan program modülü ya da modüllerinin adları,
* sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
* sınama verileri,
* sınama senaryoları

türündeki bilgileri içerir.

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.

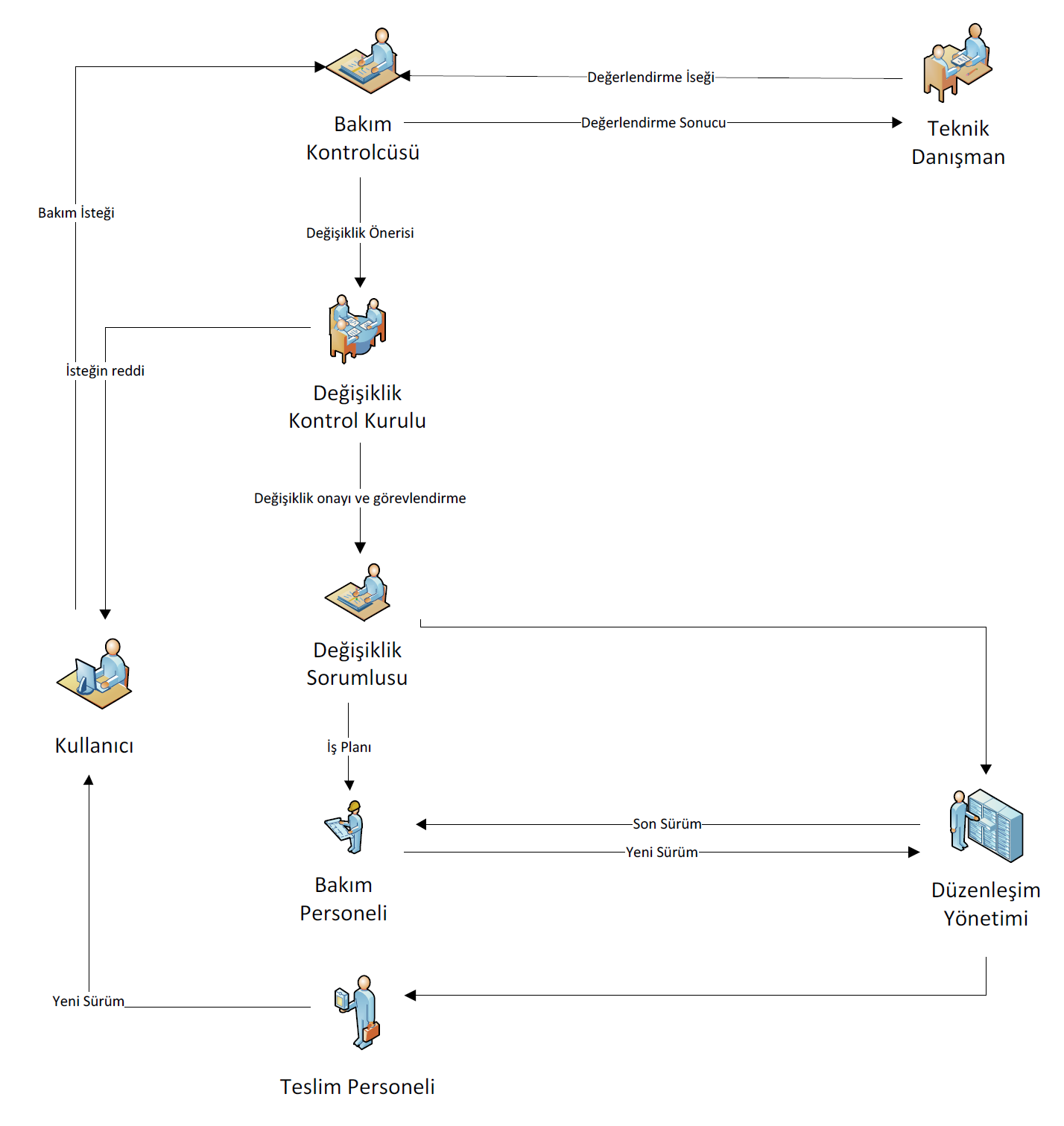
• **Küçük Hatalar:** Projeyi engellemeyen, ve giderilmesi az çaba gerektiren hatalardır.

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

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

**7.1 Giriş**

Sistemin tasarımı bittikten sonra artık seçimden seçime sistemin bakıma sokulması gerekir daha öncede belirttiğimiz gibi sistem hassas ve hata kabul etmeyecek bir sistemden bahsediyoruz. Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı baz olarak alınmıştır.



**7.2 Kurulum**

Sistem kurulumuna değinmek gerekirse devlet güvencesinde verilecek olan serverlara yüklenecek olan sistemimizde FTP arayüzü ile dosyaları servera aktaracağız ve internet explorer olan tüm cihazlarda çalışacak.

**7.3 Yerinde Destek Organizasyonu**

Bu konuyla ilgili pilot bölgede bizzat desteği ben vereceğim bunun yanında sistem canlandırılıp gerçeğe geçirilirse sistem tanımlaması kurulum için bölgelerde bayilik sistemi gibi alt kuruluşlara yetki verilecek eğer profesyonel destek istenirse yol uçak masrafını karşılamak şartıyla bölgeye yetkili gönderilip orada bir organizasyon yapılacaktır.

**7.4 Bakım Süreç Modeli**

Aslına bakmak gerekirse bakım süreç modeli yukardaki yapılan işlemlerin tümünün baştan yapılması demek bunları adım adım bir inceleyelim.

|  |
| --- |
| **8. SONUÇ** |

Sonuç olarak sistem hayata geçirildiği zaman neler değişeceği gözler önüne serdik. Bunun yanı sırabasit ama bir o kadarda güvenli olan bu sistemle ek masraflar ortadan kalkacak hataları ortadan kaldırılacak ve adil tez analizi sistemi gerçekleştirilmiş olacak.

Hem personel için hem yetkililer için oturdukları yerden işletebilecekleri bu sistem sayesinde artık eskisi kadar yorulmayacak ve herkes bu sisteme minnettar kalacak. Kurumumuz içinde iyi bir referans olacak olan bu sistem sayesinde hem biz hem kullanıcılar için çok güzel olacak. Bu zorlu süreç boyunca bize destek olan herkese bir kez daha teşekkürlerimi iletiyorum.

|  |
| --- |
| **KAYNAKLAR** |

[1] Vitosh. (2020, 12 22). *C# – Count Words in a Text – ABC Analysis for Linguists – Useful code*. Vitoshacademy: https://www.vitoshacademy.com/c-count-words-in-a-text-abc-analysis-for-linguists/ adresinden alındı

[2] Brock, N. (2020, 12 25). *c# - Reading PDF documents in .Net - Stack Overflow*. Stackoverflow: https://stackoverflow.com/questions/83152/reading-pdf-documents-in-net adresinden alındı

[3] Deenathayalan, M. (2020, 12 21). *How to Create Word Document Using C#*. C-sharpcorner: https://www.c-sharpcorner.com/UploadFile/muralidharan.d/how-to-create-word-document-using-C-Sharp/ adresinden alındı

[4] Granot, U. (2020, 12 27). *GitHub - Uzi-Granot/PdfFileAnaylyzer: PDF File Analyzer With C# Parsing Classes (Version 2.1)*. Github: https://github.com/Uzi-Granot/PdfFileAnaylyzer adresinden alındı

[5] Granot, U. (2020, 12 28). *PDF File Analyzer With C# Parsing Classes (Version 2.1) - CodeProject*. Codeproject: https://www.codeproject.com/Articles/450254/PDF-File-Analyzer-With-Csharp-Parsing-Classes-Vers adresinden alındı

[6] Sautin, M. (2020, 12 20). *Document in SautinSoft.Document library*. Sautinsoft: https://sautinsoft.com/products/document/help/net/developer-guide/document.php adresinden alındı

[7] Özkaynak, F. (2020, 9 10). *pd - Kriptarium*. Kriptarium: http://www.kriptarium.com/pd.html adresinden alındı

[8] Üniversitesi, F. (2020, 9 11). *Yeni Tez Yazım Kılavuzu -2019 | Fen Bilimleri Enstitüsü*. Fırat Üniversitesi: http://fbe.firat.edu.tr/tr/node/324 adresinden alındı