T.C.

FIRAT ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Proje Dokümantasyonu

(Thesis Control)

Proje Ekibi

Furkan AÇIKGÖZ – 15542522

Ocak – 2021

1. GİRİŞ

1.1 Projenin Amacı

Dışarıdan giriş yapılan Word (.doc, .docx) dosyalarını uygulamamız gerekli yazım kurallarını inceleyip kullanıcıya hata çıktılarını yollayacaktır.

1.2 Projenin Kapsamı

Tez yazacak olan ön lisans, lisans, yüksek lisans ve doktora öğrencilerimiz ve müşterilerimiz için kolaylık sağlamaktadır. İlerleyen aşamalarda sadece tez için de değil resmi belgelerde dilekçelerde vs. sayfa düzeni, yazım kurallarını inceleyip müşterilerimize hata çıktılarını verecektir.

- Öğrenciler
- Resmi kurumlar arası yazışma yapanlar

1.3 Tanımlamalar ve Kısaltmalar

- TC: Thesis Control

2. PROJE PLANI

2.1. Giriş

Kullanıcılarımız oturduğu yerden, daha önceden hazırladığı tezin hatalarını, sayfa düzenini, çift tırnak sayısını, paragraf sayısını, önsöz bölümü, içindekiler bölümü tutarlı mı? gibi kuralları kontrol eder. Eğer tezde bahsedilen hatalardan biri ve ya bir çoğu varsa kullanıcıya hataları çıktı olarak gönderir ve kullanıcı da bunları düzeltme şansı elde eder. En mükemmel ve hatasız şekilde tezi siz hazırlayın.

2.2. Projenin Plan Kapsamı

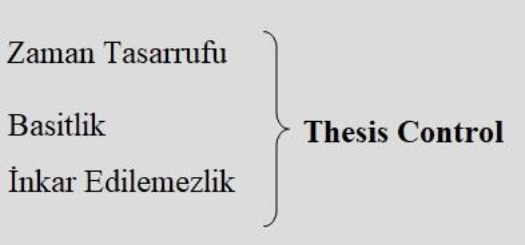
Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin doğruluğundan yola çıkıldı. Pdf veya Word dosyasının inceleyerek hatalarını vakit kaybı olmadan daha kolay bir şekilde elde edecektir.

THESİS CONTROL (TC) SİSTEMİ NEDEN GEREKLİ?

- Gözden kaçırdığınız hataların tespitinin kolaylıkla yapılabildiği için,
- Vakit nakittir sözünün doğrultusunda elinizde ki tezi saatlerce kontrol etmek yerine birkaç dakika içinde hatalarımız bulduğu için

Başta Fırat Üniversite' si olmak üzere tüm üniversiteler hatta resmi kurumlarda kullanılması için gerekli kılınmıştır.

Şekil 2.1 Vakit Nakittir



Şekil 2.2. TC Avantajları

Ölçüm Parametresi	Sayı	Ağırlık	Toplam
Kullanıcı Girdi Sayısı	1	4	4
Kullanıcı Çıktı Sayısı	1	2	2
Kullanıcı Sorgu Sayısı	1	4	4
Dışsal Arayüz Sayısı	2	6	12
Toplam			**22**

Teknik Karmaşıklık Sorusu	Puan
1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu?	1
2. Veri iletişimi gerekiyor mu?	0
3. Dağıtık işlem işlevleri var mı?	1
4. Performans kritik mi?	3
5. Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?	3
6. Sistem, çevrim içi veri girişi gerektiriyor mu?	1
7. Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?	0
8. Ana kütükler çevrim-içi olarak mı günleniyor?	0
9. Girdiler, çıktılar ya da sorgular karmaşık mı?	2
10. İçsel işlemler karmaşık mı?	2
11. Tasarlanacak kod, yeniden kullanılabilir mi olacak?	5
12. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?	4
13. Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor?	3
14. Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak?	3
TOPLAM	**28**

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 * (0.65 * 0.01 * TKF)

İN=22 * (0,65 * 0,01 * 28)

İN=4,004

Satır Sayısı = İN * 30

SATIR SAYISI=4,004 * 30 = 120,2 SATIR YAKLAŞIK 120 SATIR

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

Öncelikle projemizin türünü belirlememiz gerekiyor. Tek kişilik ekip tarafından geliştirildiği için organik projeler arasına giriyor.

Aylık Kişi Başı İş Gücü = E = a x (KSS)b
Geliştirme Süresi (Ay) = D = c x (E)d
Eleman Sayısı = E / D

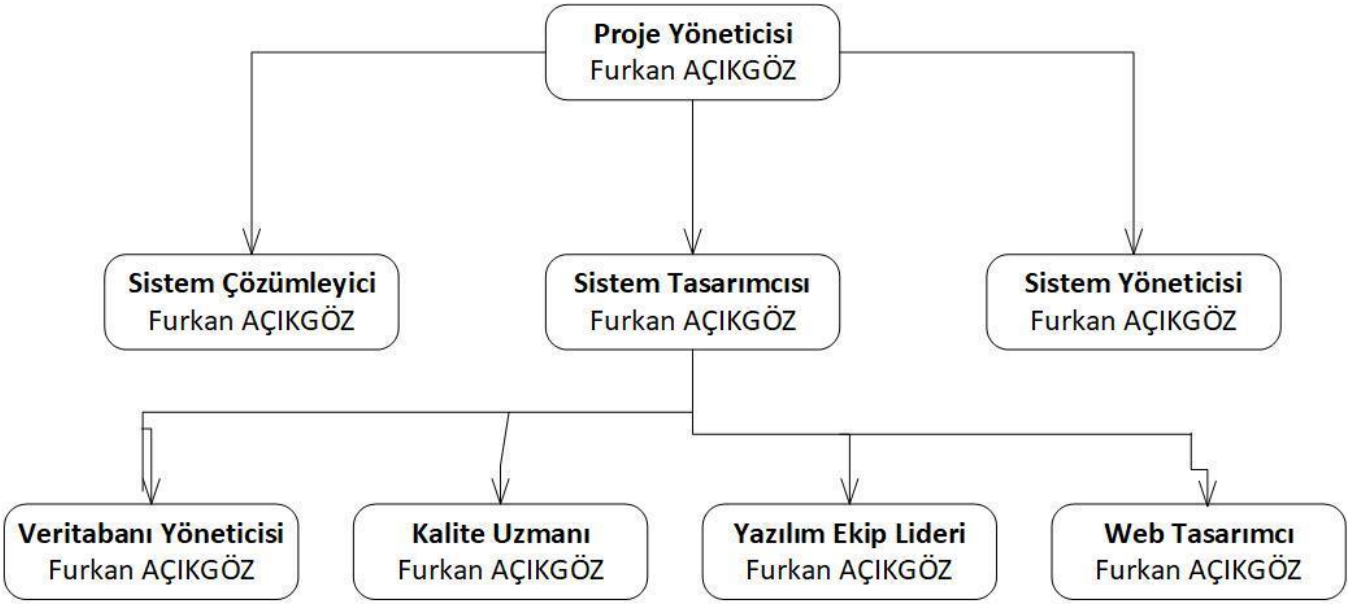
Formülde verilen değişkenler şöyle:
KSS = Kod Satır Sayısı manasına gelmektedir ve birimi bin satırdır. Projenin tahmini kaç bin satırdan oluşacağını belirtmemizi sağlar. (Kod Satırımız 120 idi; 120/1000, KSS = 0,12)

Aylık Kişi Başı İş Gücü = E = 2,4 * (0,12) * 1,05 = 0,3024
Geliştirme Süresi = D = 2,5 x (E)d = 0,28728 Ay (Yaklaşık 8 Gün)
Tahmini Gerekli Personel Sayısı = 3,18 / 1,89 = 1,052 Kişi (Yaklaşık 1 Kişi)

2.3 Proje Zaman – İş Planı

Şekil 2.3.1 Proje İş – Zaman Çizelgesi

2.3 Proje Ekip Yapısı



2.5 Önerilen Sistemin Teknik Tanımları

Basit bir tez kontrol projesi olduğundan C# yeterli olacaktır.

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

Thesis Control sistemi için JAVA ve ya C# kullanılacaktır.
Veri Tabanı olarak SQL – SERVER kullanılacaktır. (Kullanıcı üye olmak – giriş yapmak için)

2.7 Proje Standartları, Yöntem ve Metodojiler

Spiralin başladığı ilk çeyrek içinde ilk isterler toplanır ve buna göre proje planlaması yapılır. İkinci çeyrekte, ilk tanımlanan isterler 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, kullanıcı, ortaya çıkan ilk ürünü inceleyerek değerlendirme yapar, önerilerde bulunur. Bu şekilde tanımlanan ilk döngü bir sonraki döngü için bir girdi oluşturur.

Aşama	Kullanılan Yöntem/Araçlar	Ne İçin Kullanıldığı	Çıktı
Planlama	- Veri Akış Şemaları, - Süreç Belirtimleri, - Görüşme, - Maliyet Kestirim Yöntemleri - Proje Yönetim Araçları	- Süreç İnceleme - Kaynak Kestirimi - Proje Yönetimi	Proje Planı
Çözümleme	- Süreç Belirtimleri, - Veri Akış Şemaları, - Görüşme, - Nesne İlişki Şemaları, - Veri Sözlüğü	- Süreç Çözümleme - Veri Çözümleme	Sistem Çözümleme Raporu
Çözümlemeden Tasarıma Geçiş	- Akışa Dayalı Çözümleme, - Süreç Belirtimlerinin Program Tasarım Diline Dönüştürülmesi - Nesne İlişki Şemalarının Veri Tablolarına Dönüştürülmesi	- Başlangıç Tasarım - Ayrıntılı Tasarım - Başlangıç Veri - Tasarımı	Başlangıç Tasarım Raporu
Tasarım	- Yapısal Şemalar - Program Tasarım Dili - Veritabanı Tabloları - Veri Sözlüğü	- Genel Tasarım - Ayrıntılı Tasarım - Veri Tasarımı	Sistem Tasarım Raporu

Şekil 2.7.1 Proje Aşamaları

2.8 Kalite Sağlama Planı

Projedeki kalite sağlama planımız yukardaki tabloda da belli olduğu üzere;

1.**Ekonomi:** Thesis Control (TC) Sistemimizin maliyeti oldukça azdır.
2.**Tamlık**: Projede herhangi bir açık olmamalı ve programda bulunan tüm butonlar textler vs. çalışır ve tamdır.
3.**Yeniden Kullanılabilirlik**: TC her koşulda tekrardan düzenlenip kullanılabilecek.
4.**Etkinlik**: Kullanıcı sistemin her alanına hakim olduğu için sistemi etkin bir biçimde kullanacak.
5.**Bütünlük**: Admin sistemin tüm kısımlarına hakim olacak ve program bir bütün halinde çalışacaktır.
6.**Güvenilirlik**: TC güvenilirlik bakımından son derece güvenilirdir. Sistemde herhangi bir veriyi kaydetmez.
7.**Modülerlik**: Her seviyesindeki kişinin ayrı ayrı sayfalardan söz sahibi olmasını sağlar. Örneğin: Yönetim modülü, Giriş Modülü…
8.**Belgeleme**: Bu belgeden de anlaşılacağı üzere tam anlamıyla sistemin özeti olacak bu doküman oluşturulmuştur.
9.**Kullanılabilirlik**: Kullanılabilirlik olarak her seviyedeki insana hitap edeceğinden zor renkler karmaşık sistemlerden kaçınılmıştır.
10**.Temizlik:** Temiz anlaşılır bir o kadar da kolaydır.
11.**Değiştirilebilirlik**: TC veri tabanını erişme yetkisi olan ve sistem hakkında bilgisi olan herkes sistemde değişiklik yapabilecek.
12.**Esneklik**: Proje farklı platformlarda ve internet üzerinden çalışacağından gayet esnektir.
13.**Genellik**: Proje her kütüphanede kullanılabileceğinden geneldir.
14.**Sınanabilirlik**: Projedeki pilot bölge uygulaması sınana bilirliğinin göstergesidir.
15.**Taşınabilirlik**: Sistem internet üzerinden kullanılacağından herhangi bir özel cihaz gerektirmez ve istenilen cihazlarda taşınabilir ve kullanılabilir.
16.**Birlikte Çalışıla bilirlik:** Farklı programlar ile eş zamanlı çalışabilir.

2.9 Konfigürasyon Yönetim Planı

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

- Tez kuralları değişmiş ise kullanıcı yeni ayarları kendisi girebilmesi
- Sistemde herhangi bir istenmeyen durum halinde,

Durumları için konfigürasyon yönetim planı oluşturuldu.

2.10 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?

2.11 Eğitim Planı

Projeden kazanılacak en önemli olaylardan biride eğitimdir. Kullanılacak dillerin arayüz editör ve programların kullanımında hâkim olunamaması halinde bu program başarıyla neticelendirilemez. Bu yüzden projede bazı eğitimler alınması gereklidir.
Proje kapsamında alınacak olan eğitimler;
• JAVA Dil Eğitimi
• C# Dil Eğitimi
• SQL Dil Eğitimi

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;

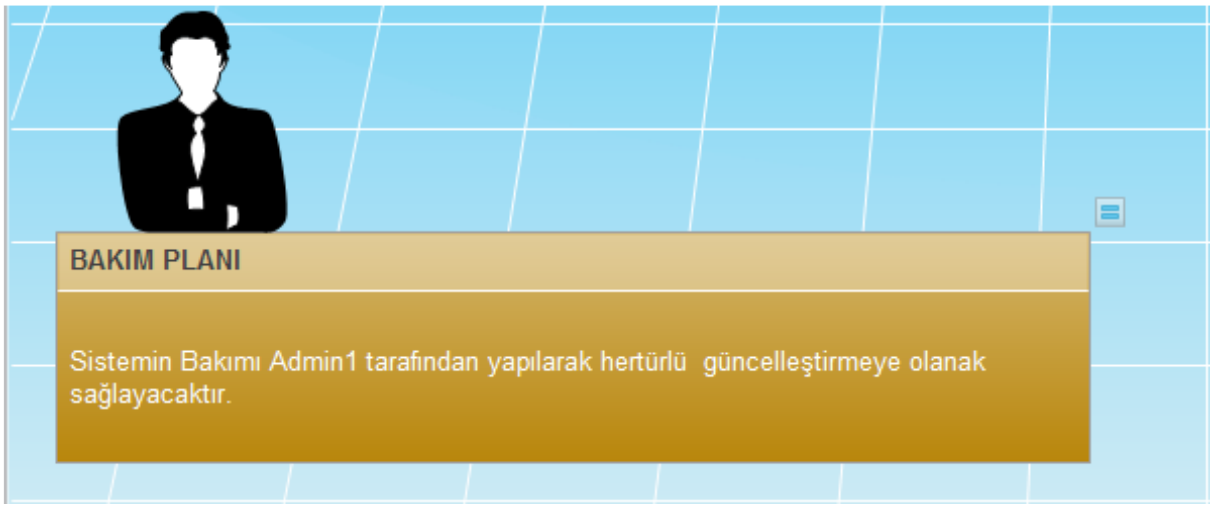
- Sisteme genel bakış
- Sistemde anketin nasıl tanımlanacağı
- Sistemde yetkili yöneticilerin yetkileri
- Sistemin kısıtları açıklanarak nasıl kullanıldığına dair bilgiler verilmektedir

2.12 Test Planı

Tezi kullanan öğrenciler tarafından test edilecektir. Gerekli kısımlarda ise resmi olarak ekip yapıları yapılacaktır.

2.13 Bakım Planı

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



Şekil 2.13.1 Proje Bakım Planı

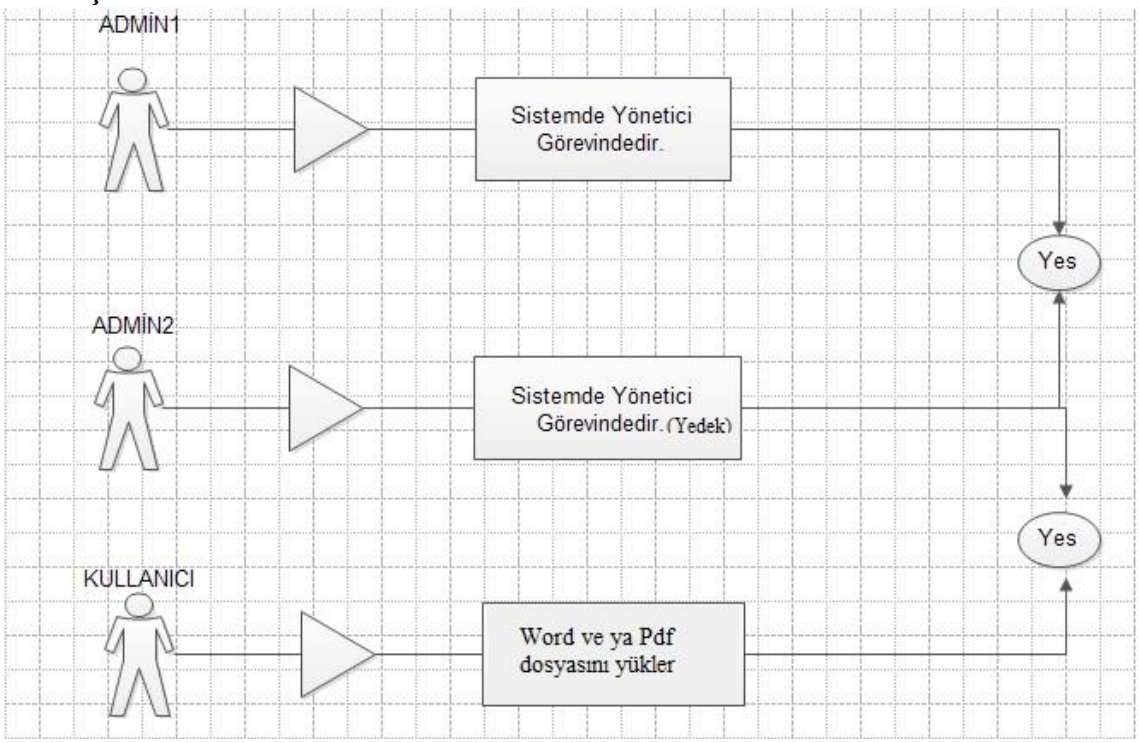
3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

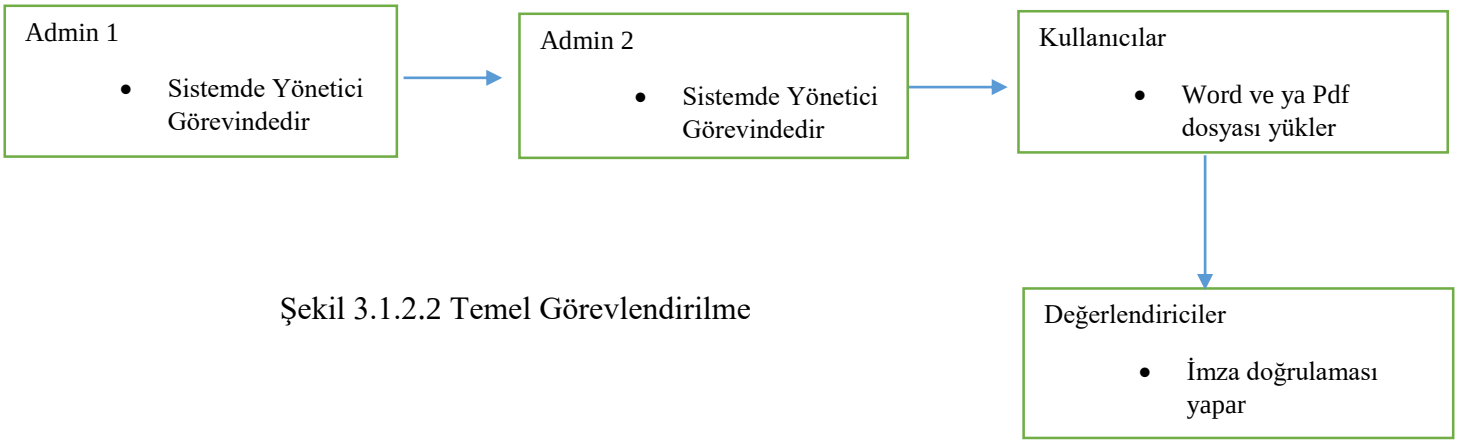
Mevcut sistem incelemesi TC sistemimiz daha önce kullanılmadığından zor olacaktır. Bunun E-SERTİFİKA incelenerek sistemin genel yapısı incelenmiş ve bizim sistemimize nasıl katkı yapılacağı araştırılmış sistemimize uygulanmıştır.

3.1.1 Örgüt Yapısı

3.1.2 İşlevsel Model



Şekil 3.1.2 Use-Case Diyagramı



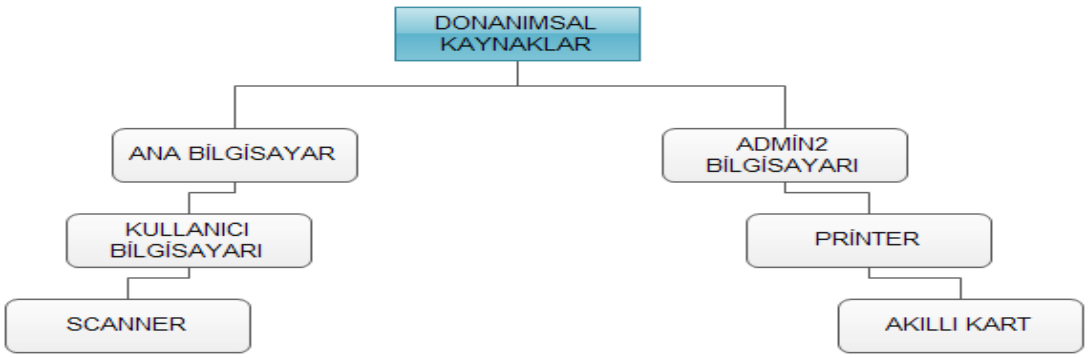
Şekil 3.1.2.2 Temel Görevlendirilme

3.1.3 Veri Modeli

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

3.1.4 Var olan Yazılım/Donanım Kaynakları

- Elektronik Sertifika Sağlayıcılar,
- Ms Word,
- Argo UML,
- RSA.



3.1.5 Var olan Sistemin Değerlendirilmesi

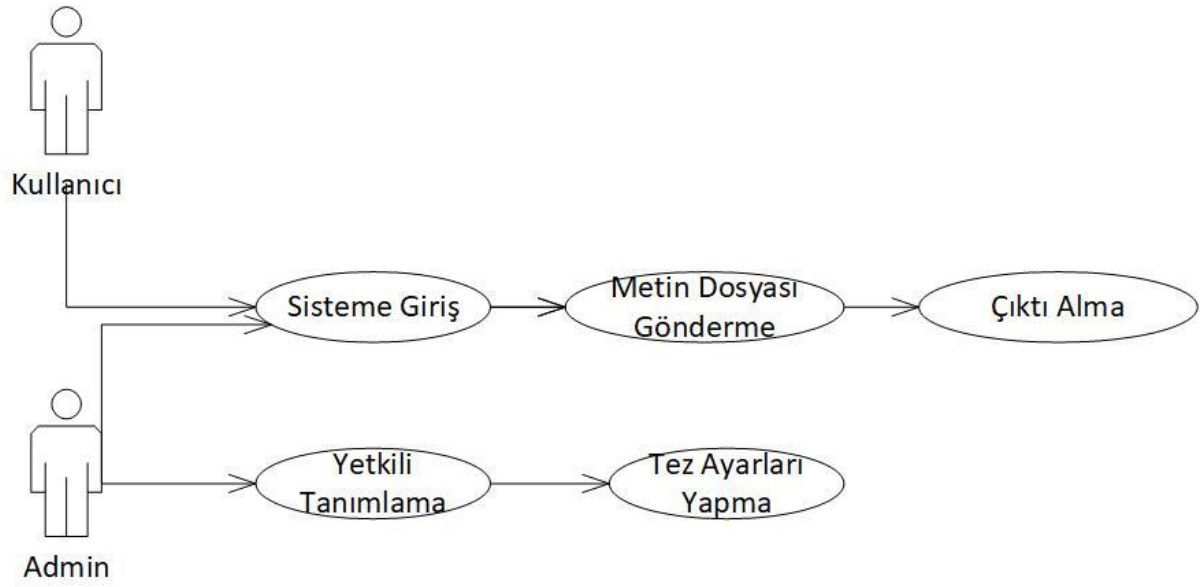
Daha önce de söylediğimiz gibi kullanıcılarım TC sistemine üye olup giriş yapacaklar ve sisteme Word ve ya PDF dosyasını yükleyecekler.

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

TC sistemi basit ve kullanışlı bir sistem olduğu için herhangi bir hukuk standartlarını ihlal etmemektedir.

3.2.2 İşlevsel Model



Şekil 3.2.2 Use – Case Diyagramı

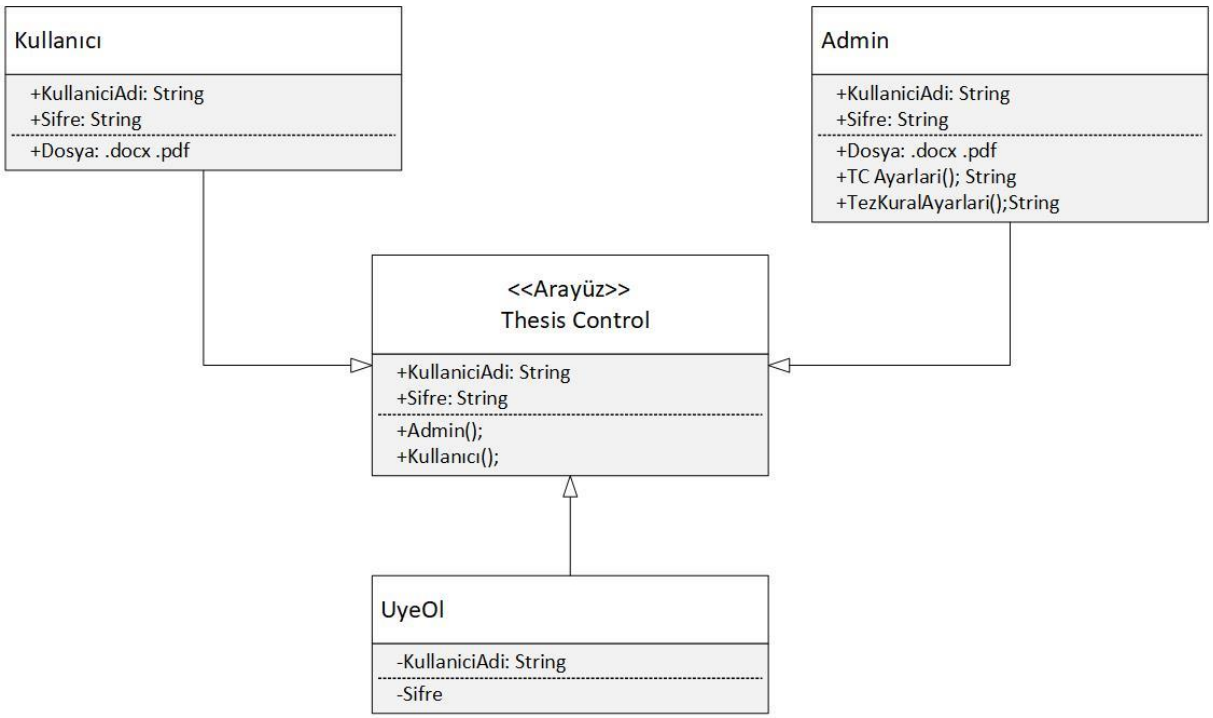
3.2.3 Genel Bakış

Genel hatlarıyla sistemi inceleyecek olursak mevcut yönetim sisteminde mevcut olan tüm olaylar burada da var doküman yönetim sistemiyle kıyaslayacak olursak sonuçlara hiçbir şekilde Admin1 dâhil müdahale edilemiyor. Bunun yanı sıra olayın akış şekli USE-CASE diyagramında mevcuttur. Yani dosya, veri, hata çıktısı hiçbir şekilde başkası tarafından alınamaz, görülemez, işlem yapılamaz.

3.2.4 Bilgi Sistemleri/Nesneler

ADMİN1: Yapının en üstünde bulunan ve en yetkili yapılanmadır
KULLANICI: Yapının ikinci nesnesidir. Thesis Control sistemine gerekli olan Word ve ya PDF dosyasını yükleyen kişidir.

3.2.5 Veri Modeli



Şekil 3.2.5 Sınıf Diyagramı

3.2.6 Veri Sözlüğü

KullaniciAdi = * String tipinde bir değerden oluşur *
Sifre= *String tipinde bir değerden oluşur*
Dosya = *Programa girilecek olan Word ve ya PDF dosyasından oluşur*

3.2.7 İşlevlerin Sıradüzeni

Kullanıcı üye olur. Üye olduktan sonra kullanıcı adı ve şifre ile sisteme giriş yapar.
Tez analizini yapmak için gerekli olan dosyayı sisteme yükler ve çıktısını alır. Admin ise gerekli tez ayarlarını kontrol eder TC sisteminin bakımını yapar.

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 kolaylığı
- Anlaşılabilirlik
- Tarafsızlık

Temel gereklilikler olarak tespit edilmiştir.

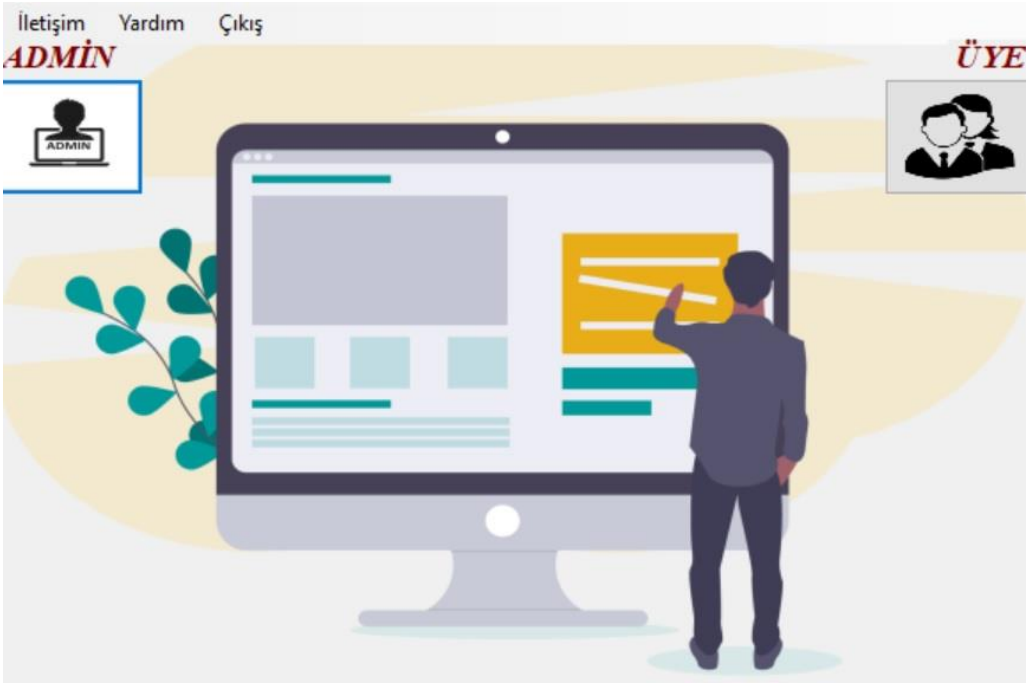
3.3 Arayüz (Modül) Gerekleri

3.3.1 Yazılım Arayüzü

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

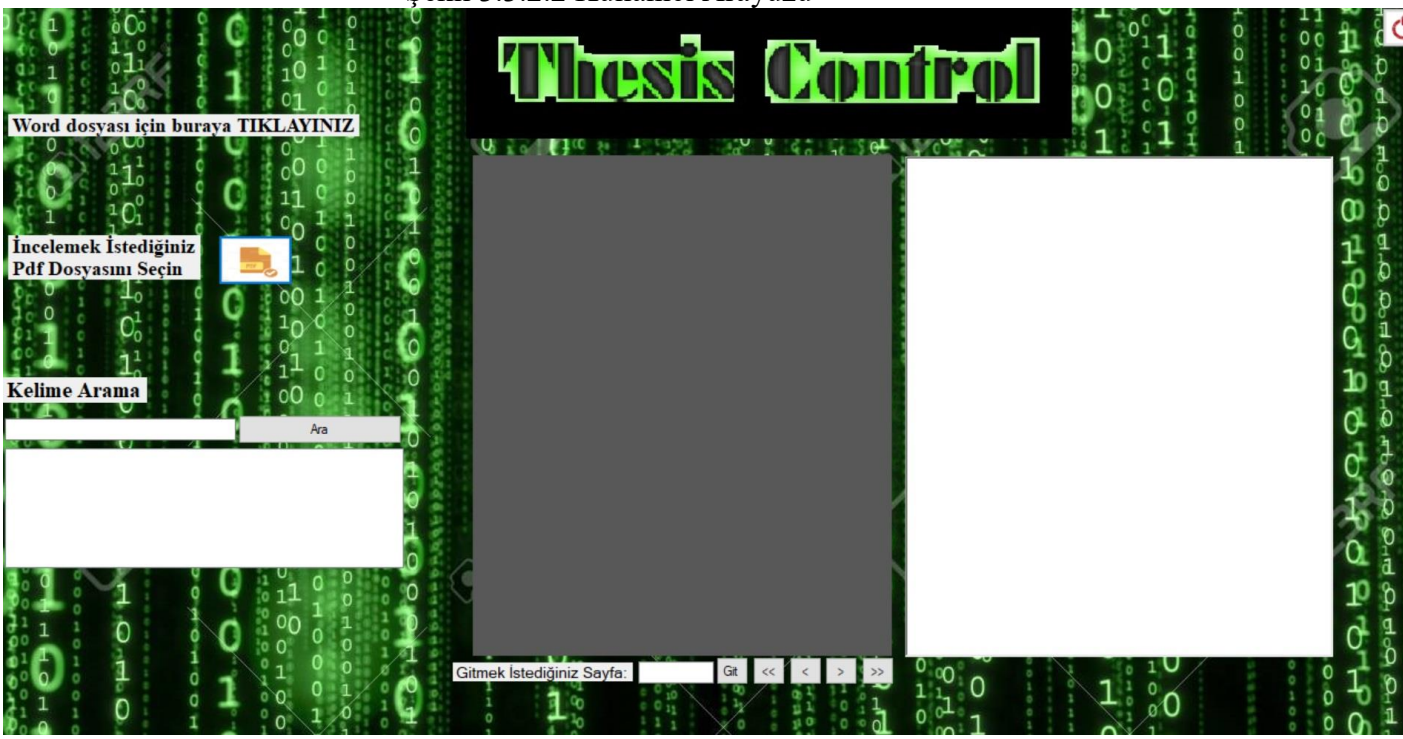
3.3.2 Kullanıcı Arayüzü

Kullanıcı arayüzü tarafsız bir şekilde, Thesis Control sistemini kullanacak kullanıcılarımız için rahat büyük puntolu arayüz tasarlandı.



Şekil 3.3.2 Uygulama Arayüzü

Şekil 3.3.2.2 Kullanıcı Arayüzü



Şekil 3.3.2.2 Kullanıcı Arayüzü

Şekil 3.3.2.3 Kullanıcı Arayüzü

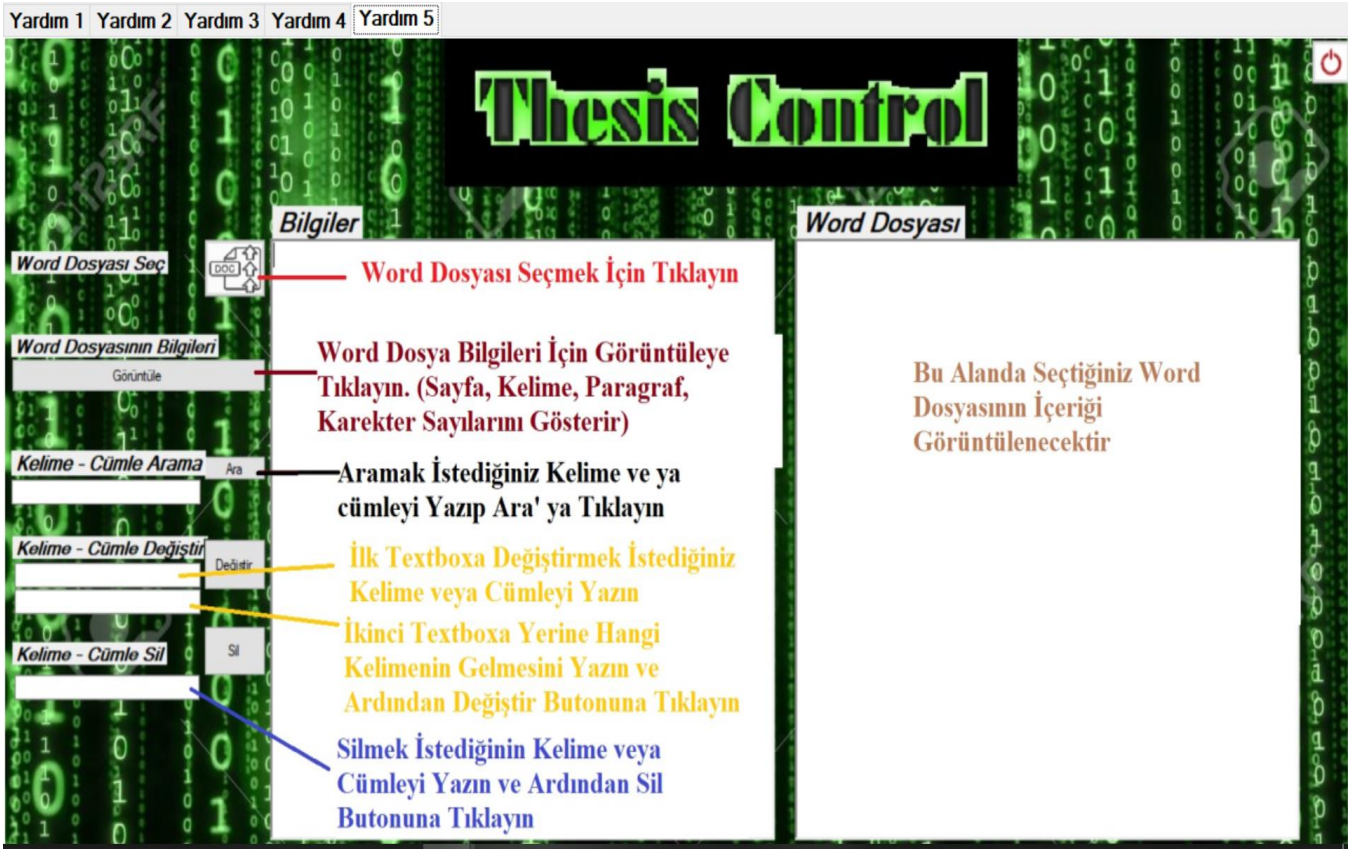
Şekil 3.3.2.4 Kullanıcı Arayüzü

3.3.3 İletişim Arayüzü

Projemizde iletişim modülümüz yer almaktadır. Kullanıcılardan gelen feedbackleri değerlendirip sürekli olarak projemizde yenilemeler yapacağız.

Şekil 3.3.3 İletişim Arayüzü

Şekil 3.3.4 İletişim Arayüzü



Şekil 3.3.5 Yardım Arayüzü

3.3.4 Yönetim Arayüzü

Thesis Control projemizde adminin görevleri, gerekli iyileştirmeleri yapmak, kullanıcı arayüzü değiştirmek, gerekli görüldüğünde üye olan kullanıcıların üyeliklerini silmek ve ihtiyaç dâhilinde yeni adminler ekleyebilmesi olacaktır.

Şekil 3.3.4 Yönetim Arayüzü

Şekil 3.3.5 Yönetim Arayüzü

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.

3.4.2 Eğitim Belgeleri

Mevcut bir belgemiz bulunmamaktadır.

3.4.3 Kullanıcı El Kitapları

TC sisteminin kullanımı oldukça basit olduğundan el kitabına ihtiyaç duyulmamıştır.

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

Şekil 4.1.1 Genel Sistem Tanımı

- **Gereksinimler**

Gereksinimler kullanıcı ihtiyaçlarına göre belirlenmiştir. Kullanıcı eklenmesini istediği özelliği TC sisteminin iletişim kısmında bize yazarak ulaşabilirler.

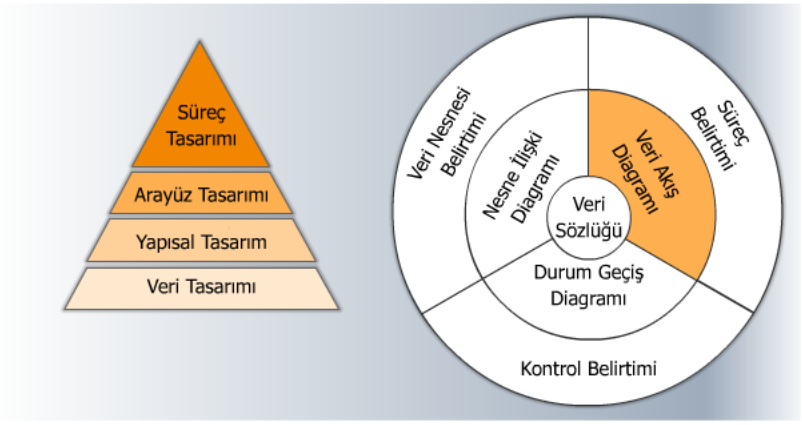
- **İşlevsel Belirtimler**

Bu kısımda ilk önce sistemin ne yapacağı sorusuna cevap verelim. Dışarıdan girilecek olan Word veya PDF dosyasını tez kurallarına göre kontrol edip bize hataların çıktısını verecektir.

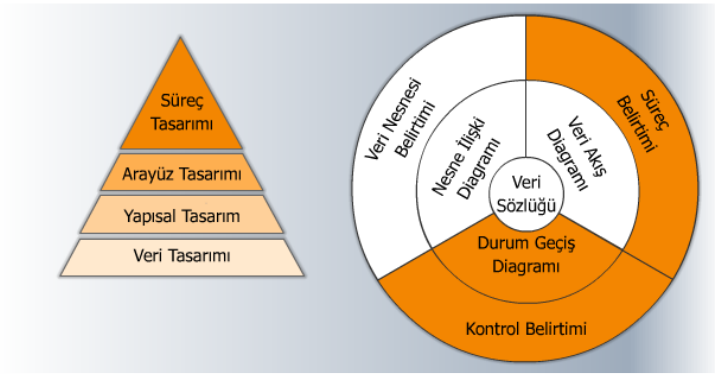
- **Tasarım**

Tasarım aşamasında neler olacağını grafiksel olarak aktarmak isterim.

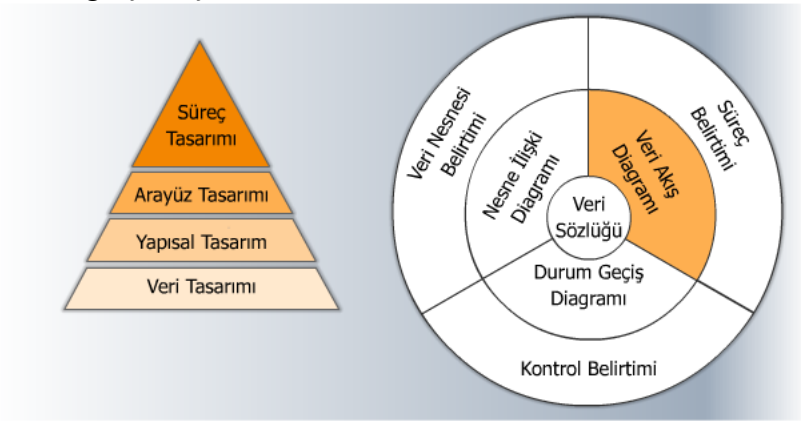
1. İlk önce bir Süreç tasarımı olacak ve adımlar aşağıdaki gibi olacak.



Şekil 4.1.1. Tasarım Aşaması

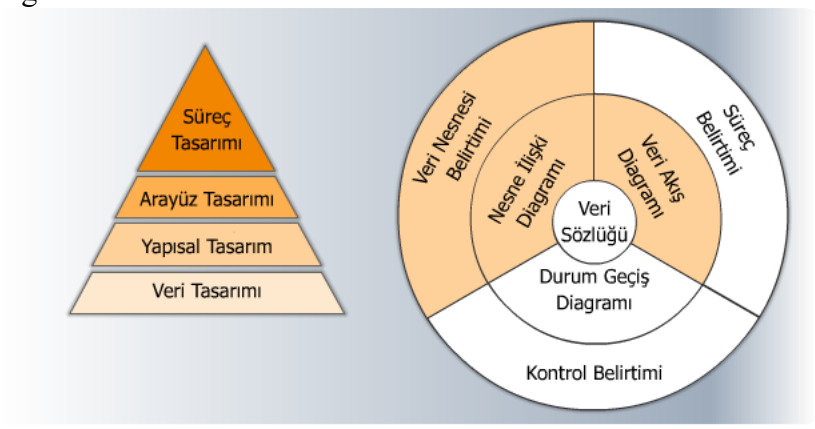


2.Süreç tasarımı bittikten sonra sıra Arayüz tasarımına geldi ve arayüz tasarımı aşağıda göründüğü adımlarla gerçekleştirildi.



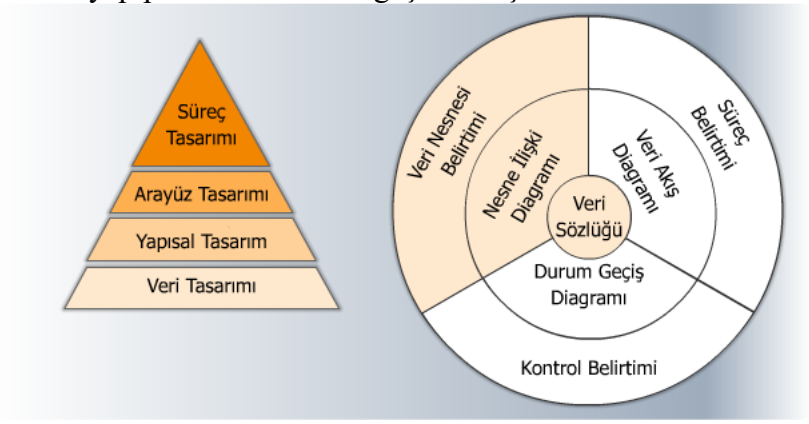
Şekil 4.1.1.2 Tasarım Aşaması

3.Arayüz tasarımını aştıktan sonra sora yapısal tasarıma geldi. Yapısal tasarımda izlenen yollar aşağıdaki gibi oldu.



Şekil 4.1.1.3 Tasarım Aşaması

4.Yapısal tasarımı da yapıp Veri tasarımına geçildi ve şu adımlar izlendi.

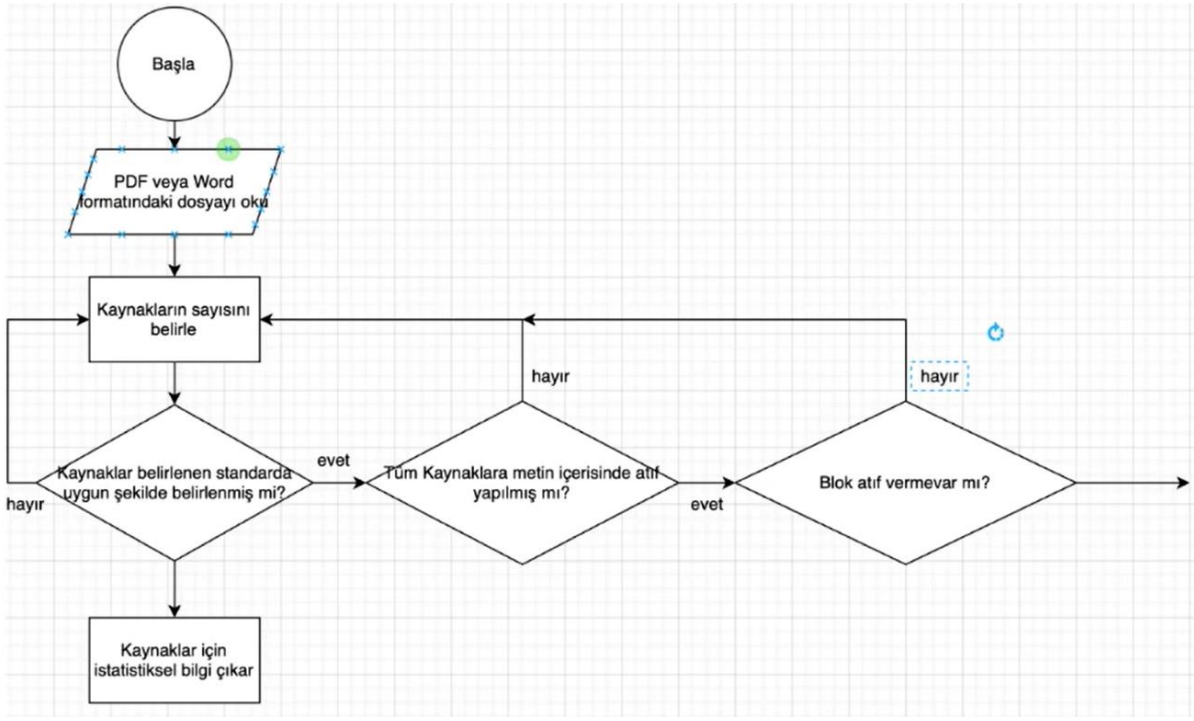


Şekil 4.1.1.4 Tasarım Aşaması

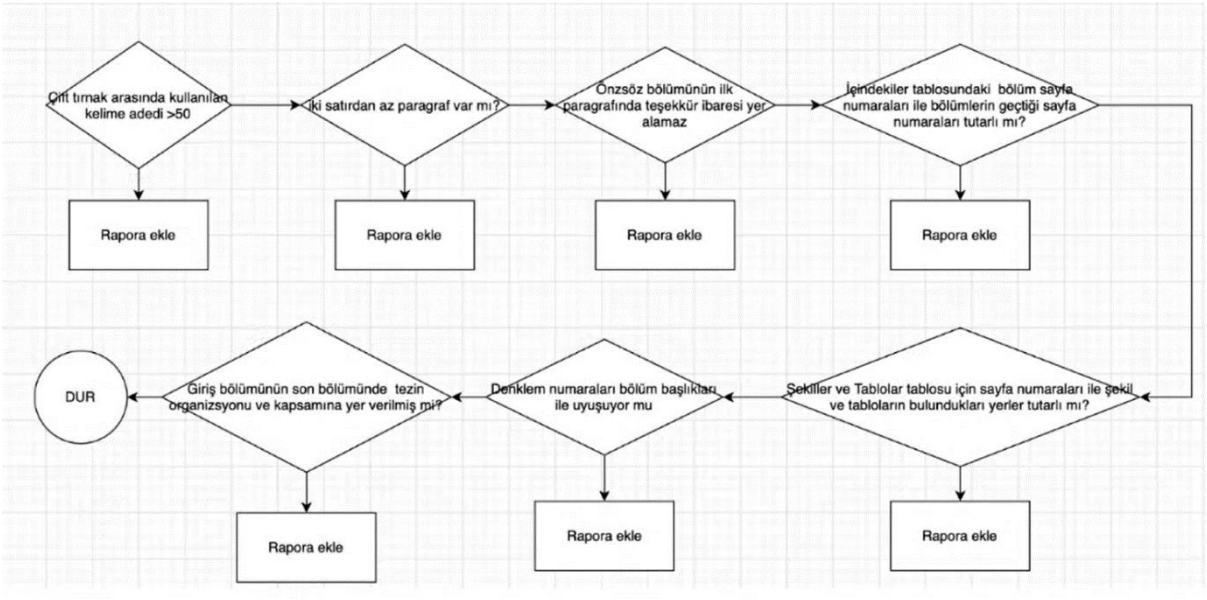
4.1.2 Varsayımlar ve Kısıtlamalar

TC sisteminde varsayım ve kısaltma bulunmamaktadır.

4.1.3 Sistem Mimarisi



Şekil 4.1.3 Sistem Mimarisi



Şekil 4.1.3.2 Sistem Mimarisi

Sistemin mimarisinin akış diyagramı şeklinde verilmesinin temel nedeni sistemin işleyiş mantığının nasıl olduğu ve nasıl bir yol çizileceğinin bilinmesidir. Akış diyagramı sistemin temel mantığı hakkında bize fikir verecektir.

4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri

Kullanıcı arabirimlerin ilk başında sistem giriş ekranı bulunacak ve üyeliği olmayan için üyelik oluşturabileceklerdir.

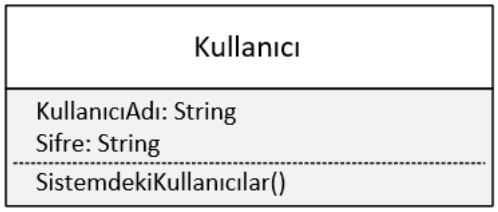
4.1.4.2 Veri Arabirimleri

Kullanıcılar üyeliklerini oluştururken bilgileri veri tabanında saklanacaktır. Kullanılacak veri tabanı da SQL – Server olacaktır.

4.1.4.3 Diğer Sistemlerle Arabirimler

Şuan tam bilgi sahibi olmadığımız için bu arabirim kullanılmayacaktır.

4.1.5 Veri Modeli



Şekil 4.1.5 Veri Tabanı Diyagramı

4.1.6 Testler

Thesis Control sistemi admin kendisi gerekli testleri ve kontrolleri yaptıktan sonra ekstra bir teste gerek yoktur.

4.2 Veri Tasarımı

4.2.1 Tablo tanımları

Thesis Control sistemi için tek tablo yeterli olacaktır. Kullanılan bu tabloda kullanıcının üyelik bilgilerini taşıyacak (Kullanıcı adı, şifre, adı, soyadı, e-mail)

Kullanıcı Tablosu	
Veri Adı	**Veri Tipi**
KullanıcıId	İnt (Primary Key)
KullanıcıAdı	Varchar(50)
KullanıcıSifresi	Varchar(50)
KullanıcıTipId	İnt (Foregin Key)
Adi	Varchar(50)
Soyadı	Varchar(50)
Email	Varchar(50)

Şekil 4.2.1 Veritabanı Kullanıcı Tablosu

4.2.2 Tablo- İlişki Şemaları

Tablo ilişki şeması hazırlandığında buraya eklenecektir.

4.2.3 Veri Tanımları

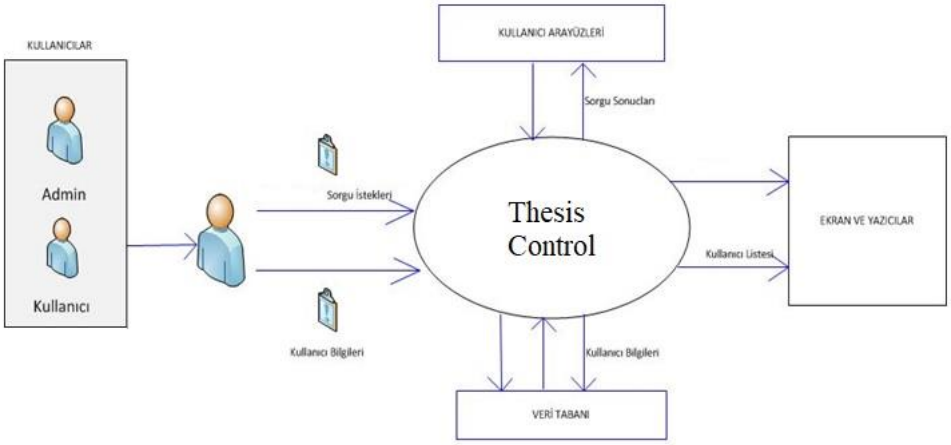
Veri tipi olarak integer(int) kullanılmasının amacı sayısal değerleri almaktan ötürüdür. String olarak kullanılan değerler kelime içeren değerleri tutacağından ötürü kullanıldı.

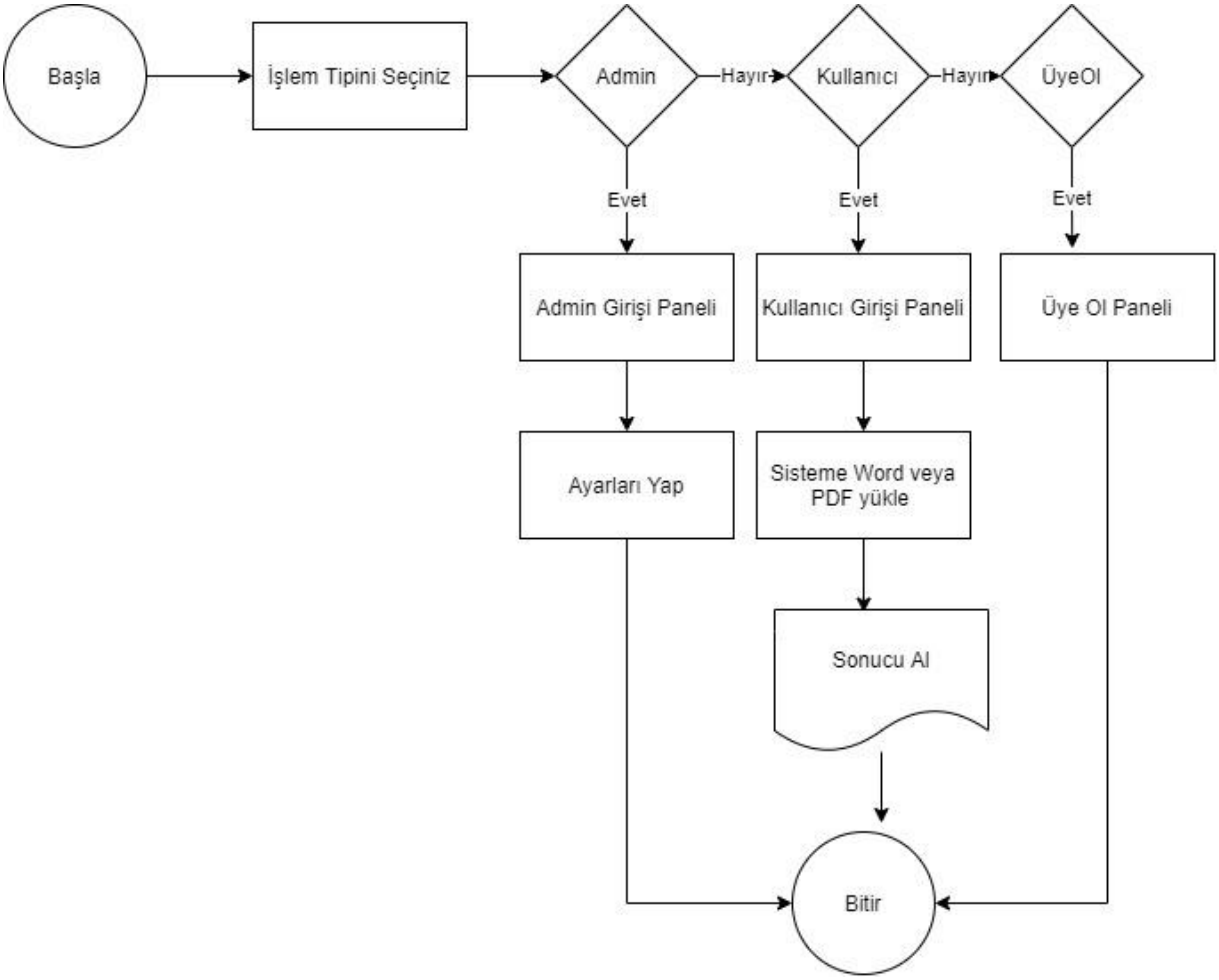
4.2.4 Değer Kümesi Tanımları

4.3 Süreç Tasarımı

4.3.1 Genel Tasarım

Genel olarak tasarımda ilk önce veri tabanı modeli oluşturuldu. Ardından giriş modülü onun ardından yönetici modülleri ve en sonda kullanıcı arayüzü oluşturduk.





Şekil 4.3.1. Giriş Sayfası Modülleri Akış Şeması

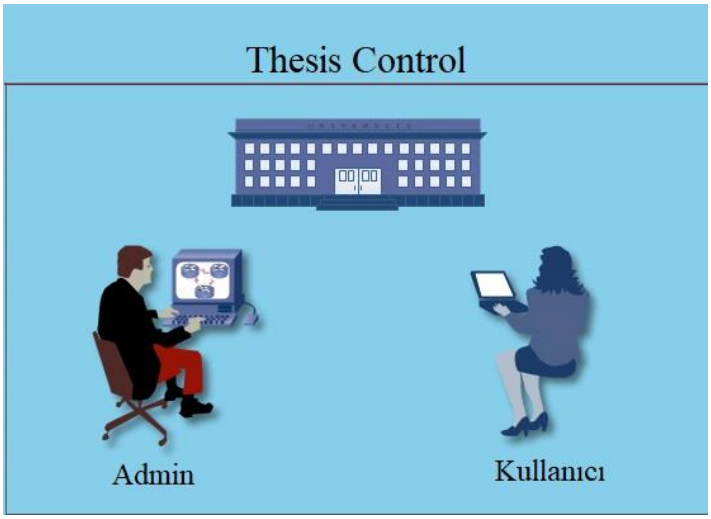
4.3.2 Modüller

4.3.2.1 Giriş Modülü

4.3.2.1.1 İşlev

Kullanıcının sisteme müdahale edilebileceği ekrana erişmesi için aşması gereken bir modüldür.

4.3.2.1.2 Kullanıcı Arabirimi



Şekil 4.3.2.1.2 Kullanıcı Modülleri

4.3.2.1.3 Modül Tanımı
4.3.2.1.4 Modül iç Tasarımı

4.3.2.2 Yönetici Modülü
4.3.2.3 İşlev

Sistem içindeki tüm arabirimler kullanıcı odaklıdır. Arayüzler birbirlerinin benzeridir. Buton isimleri farklıdır.

4.3.3 Kullanıcı Profilleri

ADMİN1: Yapının en üstünde bulunan ve en yetkili yapılanmadır. Admin2'ye ve kullanıcıya yetki verir.
ADİMN2: Admin2 esasen yoktur ama admin1 gerekli görürse admin2 atayıp kendisine belirli yetkiler verebilir.
KULLANICI: Yapının üçüncü nesnesidir ama tüm proje kullanıcı odaklıdır. Kullanıcı tez kontrol sistemini doğrudan kullanabilmektedir.

4.3.4 Entegrasyon ve Test Gereksinimleri

Sistemimizin daha öncede belirttiğimiz gibi basit bir proje olduğundan test gereksinimlerine fazla ihtiyaç duyulmamaktadır. Proje tamamlandıktan sonra admin gerekli testleri yapar ve kullanıcıya Thesis Control programını sunar.

4.4 Ortak Alt Sistemlerin Tasarımı

4.4.1 Ortak Alt Sistemler

Kullanılacak ortak sistem bulunmamaktadır.

4.4.2 Modüller arası Ortak Veriler

Kullanılacak ortak veriler bulunmamaktadır.

4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri

Ortak veri olmadığından bu modüle de gerek duyulmadı.

4.4.4 Güvenlik Alt sistemi

Yazılım sistemlerinin güvenilirliğe ilişkin nicelikleri, kullanıcıların gereksinimlerini karşılayacak şekilde ortaya koymak ve güvenilirliğin hesaplanmasına yönelik verileri toplama, istatistiksel tahminlere, 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 dağıtım alt sistemi bulunmamaktadır.

4.4.6 **Yedekleme ve Arşivleme İşlemleri**

Depolanan verilerin (Kullanıcı üye bilgileri), herhangi bir nedenle zarar görmesi, sistemin çalışma süreçlerinde zararlar oluşturabilir. Yaşanabilecek bir felaket durumu sonrasında, depolanan verilerin geri yüklenememesi, sistemin sağlandığı kullanıcılara zararlar verebilir.

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

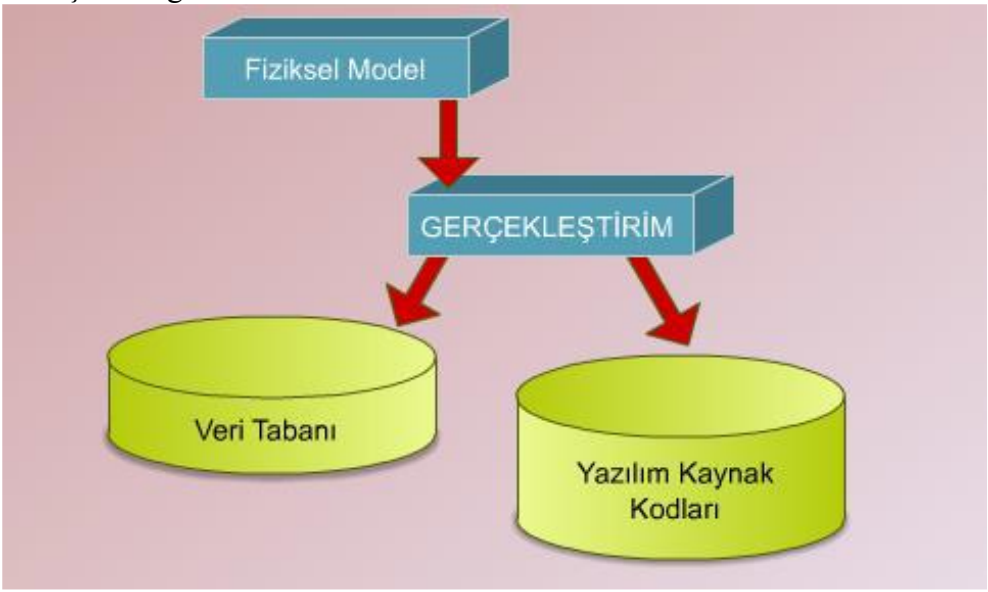
Yedekleme sistemlerinin kurulumu; yedeklenecek veri miktarı, yedekleme sıklığı, yedeklenen verinin zaman içerisinde değişme oranı ve maksimum veri kaybı gibi parametrelere bağlıdır.

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.



Şekil 5.1 Sistem Gerçekleştirim Modeli

5.2 Yazılım Geliştirme Ortamları

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

- Programlama Dili
- Veri Tabanı Yönetim Sistemi
- Hazır Program Kitapçıkları

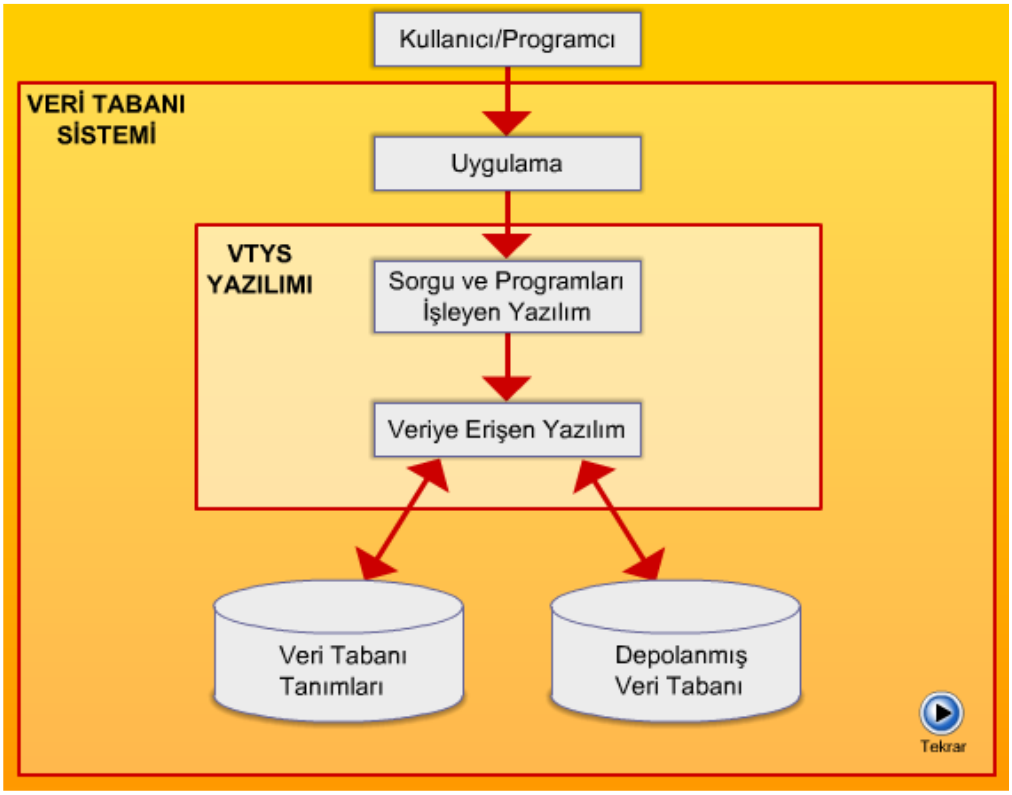
CASE Araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

5.2.1 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. C# ve SQL – SERVER veya MYSQL yeterli olacaktır.

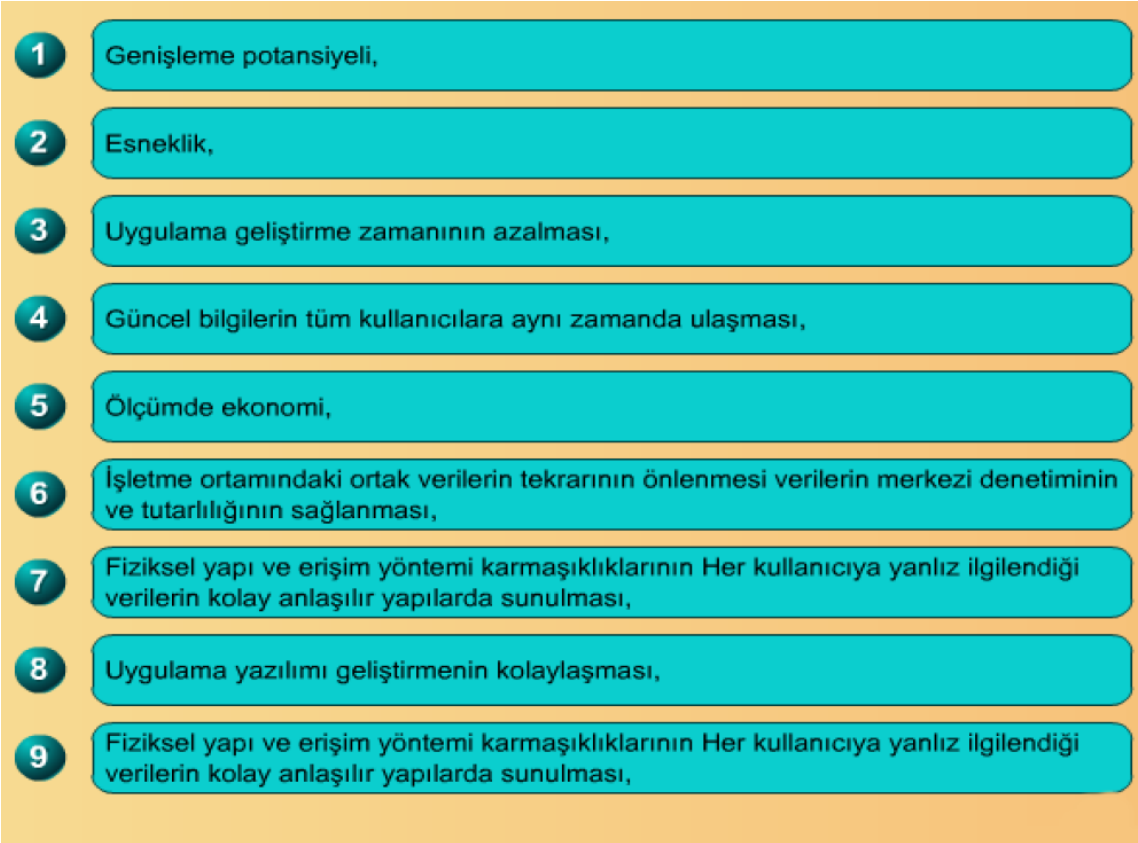
5.2.2 Veri Tabanı Yönetim Sistemleri

Veri tabanı tanımları ve Depolanmış veri tabanı ise SQL ile çözümleniyor. Veri tabanı yönetiminde kullandığımız hiyerarşiyi bir göstermek gerekirse:



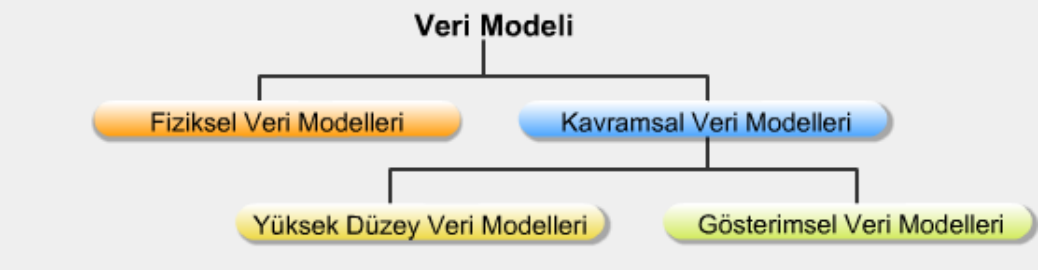
Şekil 5.2.2 Veri Tabanı Sistemi

5.2.2.1 VTYS Kullanımının Ek Yararları



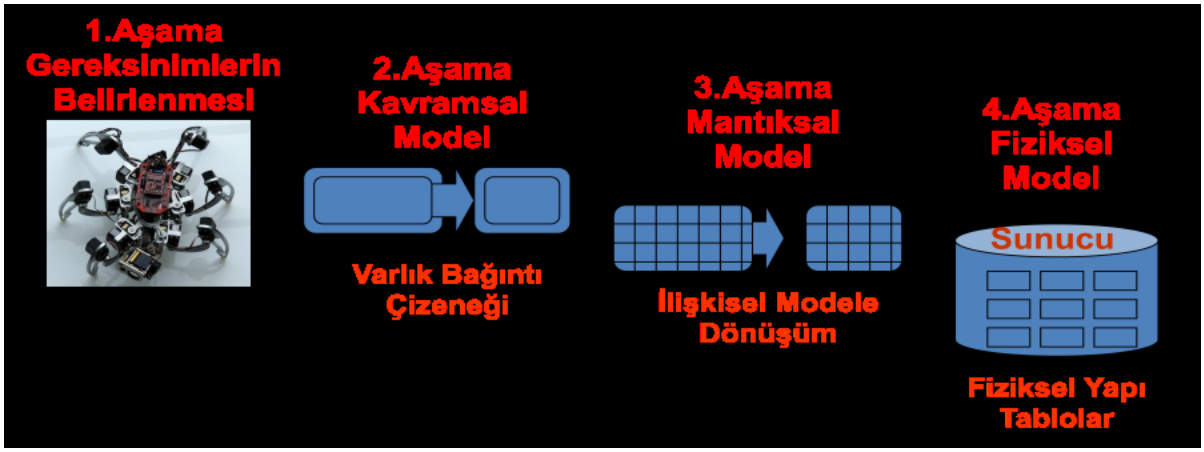
Şekil 5.2.2.1 VTYS' nin Ek Yaraları

5.2.2.2 Veri Modelleri



Şekil 5.2.2.2 Veri Modeli

Fiziksel Veri Modelinde verilerin MYSQL de tablolar içindeki alanlarda saklanacağı ve birbirleriyle ilişki içinde olduğunu söyleyebiliriz. Kavramsal Veri Modelini iki ana başlık altında inceleyebiliriz.



Şekil 5.2.2.2.2 Şema Mimarisi

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

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.

5.2.2.3 Şemalar

Bu alan daha sonra güncellenecektir

5.2.2.4 VTYS Mimarisi

VTYS mimarisini üç başlık altında ele aldık bunları özetlemek gerekir ise:

İçsel Düzey

Veritabanının fiziksel saklanma yapısını açıklar. Fiziksel veri modeli kullanır ve veritabanına erişim yolu ile veri saklamanın tüm detaylarını açıklar.

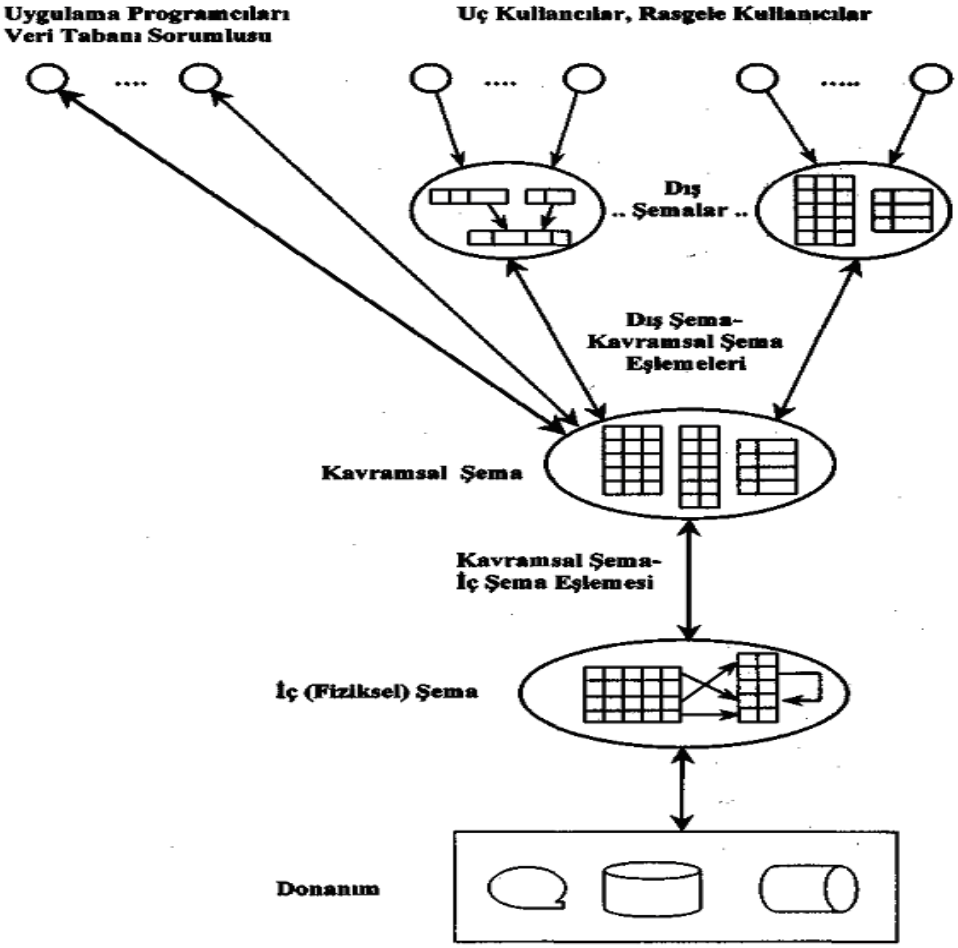
Kavramsal Düzey

Kavramsal düzey ise kavramsal şema içerir ve kullanıcılar için veritabanının yapısını açıklar. Gerçek fiziksel yapının detaylarını kullanıcıdan gizler, veri tipleri, varlıklar ilişkiler, kullanıcı işlemleri ve sınırlamalar üzerine konsantre olmamızı sağlar. Daha yüksek seviyede veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

Dışsal Düzey

Bu düzey, dış şemalar veya kullanıcı görüşleri içerir. Her dış şema veritabanının bir bölümünü açıklar ve her gruba kendi ilgilendiği görüşü sunarken, diğer bir gruptan ilgilenmediği görüşü saklar. Daha yüksek düzeyde veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

Tekrar



Şekil 5.2.2.4 Veri Tabanı Şemaları

5.2.2.5 Veritabanı Dilleri ve Arabirimleri

Sistemimizde Veritabanı dili olarak SQL kullanıldı. MYSQL arabirimini yeterli görüldü.

Veri Tanımlama Dili (VTD)	Kavramsal şemaları tanımlamak üzere veritabanı yönetici ve tasarımcısı tarafından kullanılır
Saklama Tanımlama Dili (STD)	İçsel şemayı tanımlamak için kullanılır.
Görüş Tanımlama Dili (GTD)	Görüş tanımlama dili kullanıcı görüşlerini tanımlamak ve kavramsal şemaya dönüştürmek amacıyla kullanılır.
Veri İşleme Dili (VİD)	Veri işleme dilidir. Veri tabanı oluşturulduktan sonra Veri tabanına veri eklemek, değiştirmek, silmek veya eklenmiş veriyi getirmek amacıyla kullanılır.

Şekil **5.2.2.5** Veri Tabanı Dilleri ve Arabirimler

5.2.2.6 Veri Tabanı Sistem Ortam

Şimdiye kadar MYSQL üzerinden gittik. Gerekli görülürse phpMyAdmin den destek alacağız.

5.2.2.7 VTYS' nin Sınıflandırılması

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

5.2.2.8 Hazır Program Kütüphane Dosyaları

İhtiyaç duyulmadı.

5.2.2.9 CASE Araç ve Ortamları

Case araçları olarak ise Microsoft un Visio Project ürünlerini kullandık. Bunun yanı sıra draw.io kullandık.

5.3 Kodlama Stili

Kendime has kodlama biçimi 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 satırın sonunda yapıldı. Ve biten her günün sonunda kodun kalındığı yere o günün tarihi atıldı.

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 değişkenlerin veri tabanında karşılığı varsa önce “tabloadı_islevadı_sayısı” seklinde bir anlamlı isimlendirme yaptık.

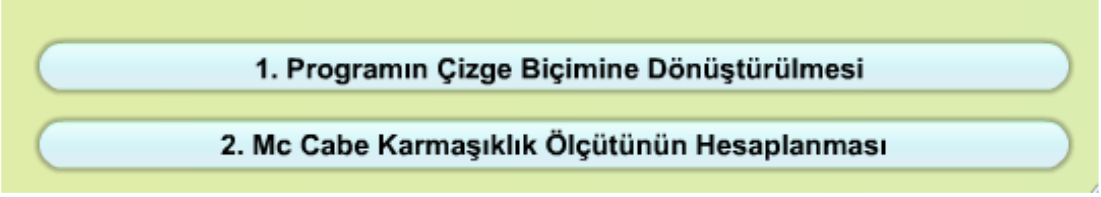
5.3.4 Yapısal Programlama Yapıları

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:



Şekil 5.4 Program Karmaşıklığı

5.4.1 Programın Çizge Biçimine Dönüştürülmesi

Daha sonra güncellenecektir.

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

Daha sonra güncellenecektir.

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

Gözden geçirme sırasında referans alınacak sorular olacaktır.

5.6.2.5 Sunuş

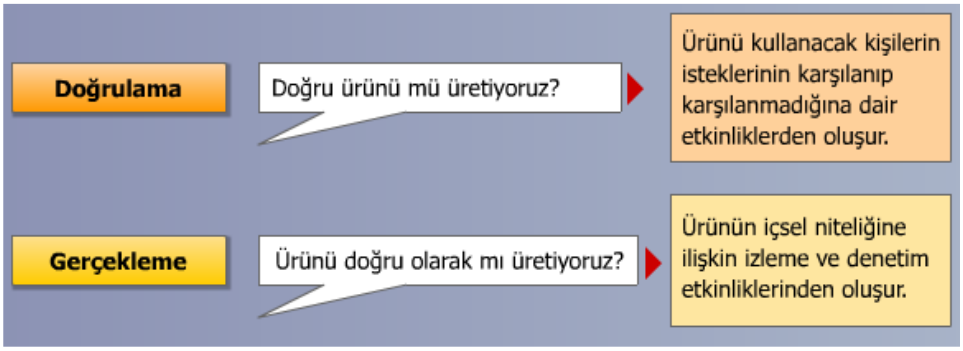
Artık son kısma gelindiğinde ise şu sorular soruldu:

- 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

Bu bölümde Doğrulama ve Geçerleme adımları anlatılacaktır.
Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklemesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.



Şekil 6.1.1 Doğrulama Geçerleme

6.2. Sınama Kavramları

Birim Sınama: Sistemin birimleri olan YSK-YSK il-YSK İlçe-Sandık Kurulu-Seçmen sırasıyla kendi içlerinde birimleri sınandı ve sonuçları çıkartıldı.

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

Sistem Sınama: Sistemin bütün olarak sınanması yapıldı ve onaylandı.

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

6.3. Doğrulama ve Geçerleme 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 bir çaba türüdür.

6.4.1 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.

6.4.2 Temel Yollar Sınaması

Setup dosyasının kurulumu farklı bilgisayarlarda denendi ve başarıyla kurulum tamamlandı. Sistemin çalışmasında herhangi bir aksilik bulunmadı.

6.5. 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.

6.5.1 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.

6.5.2 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.

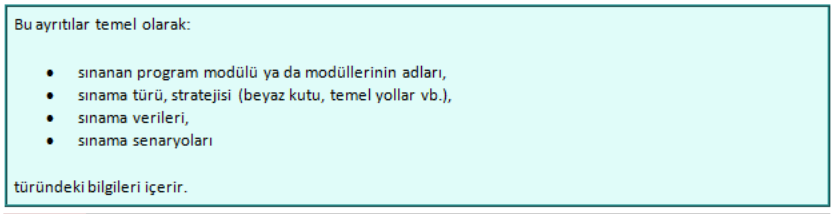
6.6. Sınama Planlaması

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.

6.7. Sınama Belirtimleri

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



Şekil 6.7.1 Sınama Betimleri

Kod çalışma durumu kontrol edildi herhangi bir eksik görülmedi.
Admin ve kullanıcı için Thesis Control programının eksiksiz çalıştığı test edildi

6.8. 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.

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

C# tan hazırladığım setup kurulumu ile kurulumu çok hızlı ve basit bir şekilde tamamlayabiliyoruz. Kurulan bilgisayarlar da Microsoft Visual Studio 2017 olsun veya olmasın .exe uzantılı dosyamız hatasız çalışmaktadır.

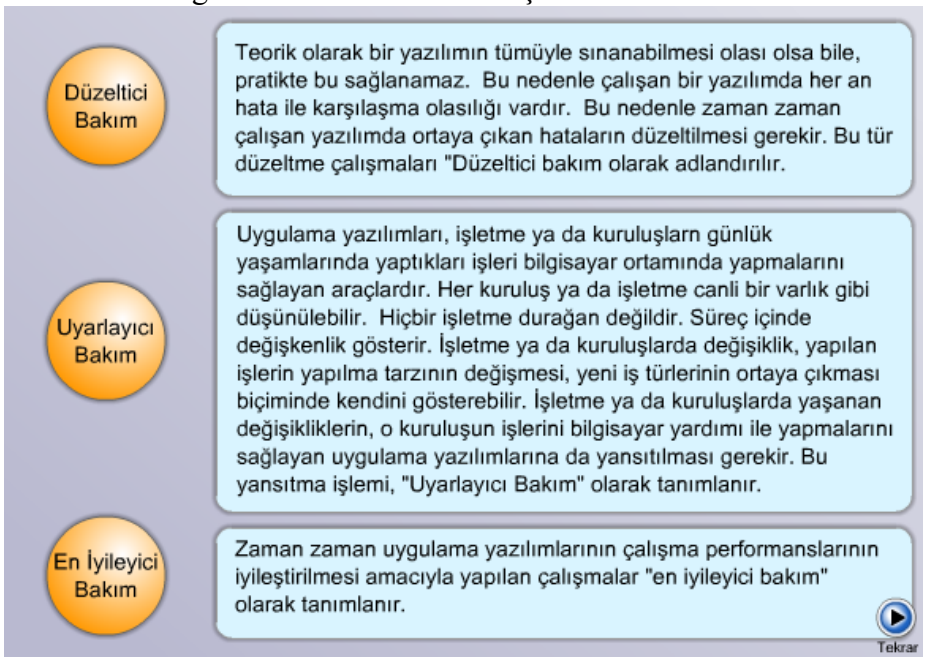
7.3 Yerinde Destek Organizasyonu

Yerinde deste organizasyonuna gerek duyulmamaktadır. Ama Thesis Control uygulamamızda iletişim bölümünden mail atarak ulaşabilirler.

7.4 Yazılım Bakımı

7.4.1 Tanım

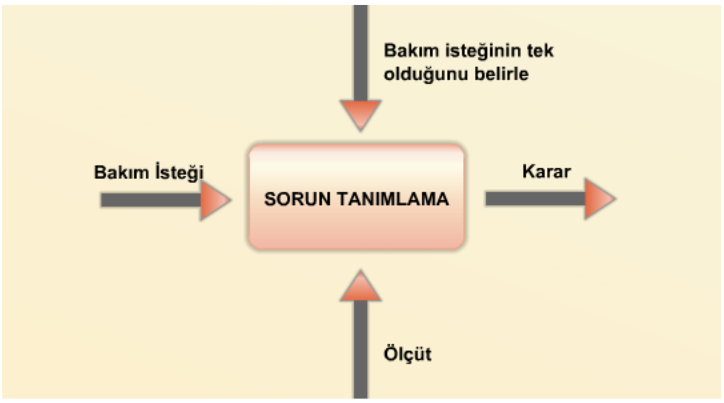
Bakım gerekli olursa tarafımdan çözülecektir.



Şekil 7.4.1.1 Bakım Çeşitleri

7.4.2 Bakım Süreç Modeli

1. Adım: Sorunu Tanımlama Süreci
İlk önce bakım ne için yapılıyor sorun ne buna bir bakalım.

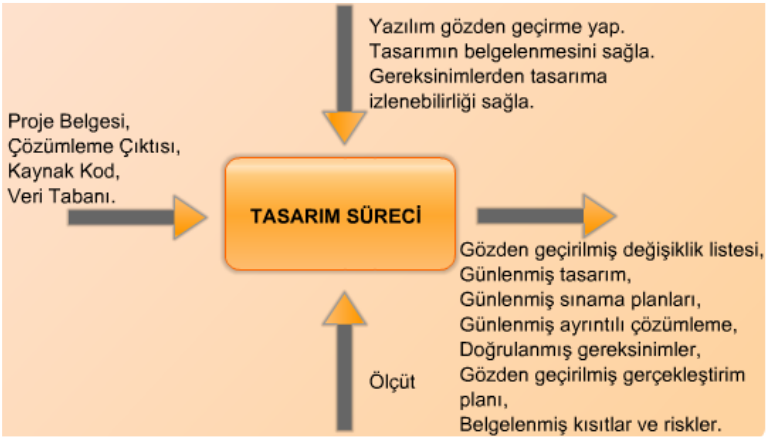


Şekil 7.4.2.1 Sorun Tanımlama

2. Adım: Çözümleme Süreci

Şekil 7.4.2.2 Çözümleme Süreci

3. Adım: Tasarım Süreci



Şekil 7.4.2.3 Tasarım Süreci

4. Adım: Gerçekleştirim Süreci

Şekil 7.4.2.4 Gerçekleştirim Süreci

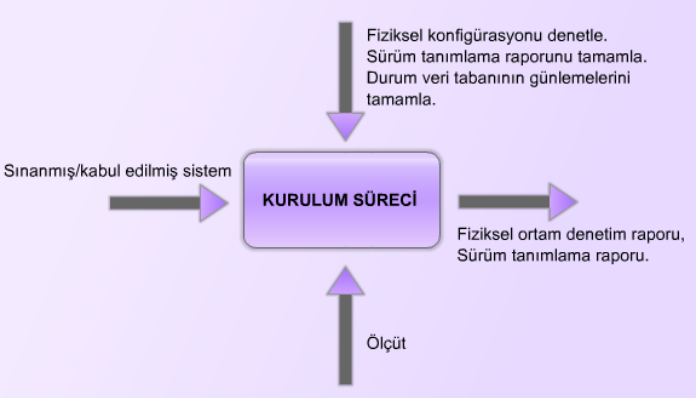
5. Adım: Sistem Sınama Süreci

Şekil 7.4.2.5 Sınama Süreci

6. Adım: Kabul Sınaması Süreci

Şekil 7.4.2.6 Kabul Sınama Süreci

7. Adım: Kurulum Süreci



Şekil 7.4.2.7 Kurulum Süreci

8. SONUÇ

Thesis Control sistemi için yola çıktığımda yapmak istediğim uygulamayı tam olarak tasarlayamadım. Pdf ve ya Word üzerinden işlemler yapan güzel bir sistem geliştirdim. Bunun yanı sıra kendimi geliştirmiş oldum. Thesis Control sistemini ve dokümantasyonu incelediğiniz için teşekkür ediyorum.

9. KAYNAKLAR

İnternet Kaynakları

Pdf Kaynaklarım
1-) https://stackoverflow.com/questions/35173818/total-page-number-with-itextpdf
2-) https://stackoverflow.com/questions/6734374/get-only-word-count-from-pdf-document
3-) https://stackoverflow.com/questions/10960801/itext-sharp-get-total-number-of-bookmarks
4-) https://www.c-sharpcorner.com/blogs/reading-contents-from-pdf-word-text-files-in-c-sharp1
5-) https://kb.itextpdf.com/home/it7kb/faq/how-to-restrict-the-number-of-characters-on-a-single-line

Word Kaynaklarım
1-) https://social.msdn.microsoft.com/Forums/tr-TR/5e378daa-a2e4-4f56-b432-a96ae953def8/microsoftofficeinteropworddll-ile-docx-oluturmak?forum=csharptr
2-) https://stackoverflow.com/questions/16918066/word-count-using-microsoft-office-interop-word
3-) https://sautinsoft.com/products/document/help/net/developer-guide/counting-words-paragraphs-net-csharp-vb.php
4-) https://sautinsoft.com/products/document/help/net/developer-guide/find-and-replace-text-in-docx-document-net-csharp-vb.php
5-) https://sautinsoft.com/products/document/help/net/developer-guide/delete-text-from-docx-document-net-csharp-vb.php