



T.C.
FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Proje Dokümantasyonu

(DocChecker)

Proje Ekibi

Alperen TOKGÖZ

İbrahim ALTIKAT

Halef BUDANUR

Mehmet Sait OĞUZ

Mehmet Ali KILINÇASLAN

1. GİRİŞ
1.1 Projenin Amacı
1.2 Projenin Kapsamı
1.3 Tanımlamalar ve Kısaltmalar
2. PROJE PLANI
2.1 Giriş
2.2 Projenin Plan Kapsamı
2.3 Proje Zaman-İş Planı
2.4 Proje Ekip Yapısı
2.5 Kullanılan Özel Geliştirme Araçları ve Ortamları
2.6 Proje Standartları, Yöntem ve Metodolojiler
2.7 Kalite Sağlama Planı
2.8 Konfigürasyon Yönetim Planı
2.9 Kaynak Yönetim Planı
2.10 Eğitim Planı
2.11 Test Planı
2.12 Bakım Planı
2.13 Projede Kullanılan Yazılım/Donanım Araçlar
3. SİSTEM ÇÖZÜMLEME
3.1 Mevcut Sistem İncelemesi
3.1.1 Örgüt Yapısı
3.1.2 İşlevsel Model
3.1.3 Veri Modeli
3.1.4 Varolan Yazılım/Donanım Kaynakları
3.1.5 Varolan Sistemin Değerlendirilmesi
3.2 Gereksenen Sistemin Mantıksal Modeli
3.2.1 Giriş
3.2.2 İşlevsel Model
3.2.3 Genel Bakış
3.2.4 Bilgi Sistemleri/Nesneler
3.2.5 Veri Modeli
3.2.6 Veri Sözlüğü
3.2.7 İşlevlerin Sıra düzeni
3.2.8 Başarım Gereklere
3.3 Arayüz (Modül) Gereklere
3.3.1 Yazılım Arayüzü
3.3.2 Kullanıcı Arayüzü
3.3.3 İletişim Arayüzü

3.3.4 Yönetim Arayüzü

3.4 Belgeleme Gerekleri

3.4.1 Geliştirme Sürecinin Belgelenmesi

3.4.2 Eğitim Belgeleri

3.4.3 Kullanıcı El Kitapları

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

4.1.2 Varsayımlar ve Kısıtlamalar

4.1.3 Sistem Mimarisi

4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri

4.1.4.2 Veri Arabirimleri

4.1.4.3 Diğer Sistemlerle Arabirimler

4.1.5 Veri Modeli

4.1.6 Testler

4.1.7 Performans

4.2 Veri Tasarımı

4.2.1 Tablo tanımları

4.2.2 Tablo- İlişki Şemaları

4.2.3 Veri Tanımları

4.2.4 Değer Kümesi Tanımları

4.3 Süreç Tasarımı

4.3.1 Genel Tasarım

4.3.2 Modüller

4.3.2.1 XXX Modülü

4.3.2.1.1 İşlev

4.3.2.1.2 Kullanıcı Arabirimi

4.3.2.1.3 Modül Tanımı

4.3.2.1.4 Modül iç Tasarımı

4.3.2.2 YYY Modülü

4.3.3 Kullanıcı Profilleri

4.3.4 Entegrasyon ve Test Gereksinimleri

4.4 Ortak Alt Sistemlerin Tasarımı

4.4.1 Ortak Alt Sistemler

4.4.2 Modüller arası Ortak Veriler

4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri

4.4.4 Güvenlik Altsistemi

4.4.5 Veri Dağıtım Altsistemi

4.4.6 Yedekleme ve Arşivleme İşlemleri

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1. Giriş

5.2. Yazılım Geliştirme Ortamları

5.2.1 Programlama Dilleri

5.2.2 Veri Tabanı Yönetim Sistemleri

5.2.2.1 VTYS Kullanımının Ek Yararları

5.2.2.2 Veri Modelleri

5.2.2.3 Şemalar

5.2.2.4 VTYS Mimarisi

5.2.2.5 Veritabanı Dilleri ve Arabirimleri

5.2.2.6 Veri Tabanı Sistem Ortamı

5.2.2.7 VTYS'nin Sınıflandırılması

5.2.2.8 Hazır Program Kütüphane Dosyaları

5.2.2.9 CASE Araç ve Ortamları

5.3. Kodlama Stili

5.3.1 Açıklama Satırları

5.3.2 Kod Biçimlemesi

5.3.3 Anlamlı İsimlendirme

5.3.4 Yapısal Programlama Yapıları

5.4. Program Karmaşıklığı

5.4.1 Programın Çizge Biçimine Dönüştürülmesi

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

5.5. Olağan Dışı Durum Çözümleme

5.5.1 Olağandışı Durum Tanımları

5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

5.6. Kod Gözden Geçirme

5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

5.6.2.1 Öbek Arayüzü

5.6.2.2 Giriş Açıklamaları

5.6.2.3 Veri Kullanımı

5.6.2.4 Öbeğin Düzenlenişi

5.6.2.5 Sunuş

6. DOĞRULAMA VE GEÇERLEME

6.1. Giriş

6.2. Sınama Kavramları

6.3. Doğrulama ve Geçerleme Yaşam Döngüsü

6.4. Sınama Yöntemleri

6.4.1 Beyaz Kutu Sınaması

6.4.2 Temel Yollar Sınaması

6.5. Sınama ve Bütünleştirme Stratejileri

6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

6.6. Sınama Planlaması

6.7. Sınama Belirtileri

6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri

7. BAKIM

7.1 Giriş

7.2 Kurulum

7.3 Yerinde Destek Organizasyonu

7.4 Yazılım Bakımı

7.4.1 Tanım

7.4.2 Bakım Süreç Modeli

8. SONUÇ

9. KAYNAKLAR

1. GİRİŞ

1.1. Projenin Amacı

Projemizin amacı; Fırat Üniversitesi tez yazma kurallarını bünyesinde barındıran ve yazılan tezlerin kurallara uygun yazılıp yazılmadığını kontrol eden, kontrol sonucunda görevli akademisyene tez içerisinde kural dışı olarak belirlenen durumları filtreli bir rapor halinde sunması amaçlanmaktadır.

1.2. Projenin Kapsamı

Program tez yazım kurallarını kontrol edeceğinden, bu kuralların geçtiği her ödev ve/veya projede kullanılabilecektir. Mevcut kapsam sadece tez kontrolü iken hedef kapsam ise programın vizyonunu geliştirerek bitirme ödevleri, staj dökümanları gibi belgelerin kural kontrolünün de sağlanmasıdır.

1.3. Tanımlamalar ve Kısaltmalar

- JDBC → Java DataBase Connectivity (Java veri tabanı bağlantısı)
- İN → İşlev nokta sayısı
- AİN → Ana işlev nokta sayısı
- IDE → Integrated Development Environment (Tümleşik Geliştirme Ortamı)
- VTYS → Veri Tabanı Yönetim Sistemleri
- UML → Unified Modeling Language (Birleştirilmiş Modelleme Dili)
- SEI → Stockholm Environment Institute (Stockholm Çevre Enstitüsü)
- DKS → Değişiklik Kontrol Sistemine
- BT → Bilişim Teknolojileri

2. PROJE PLANI

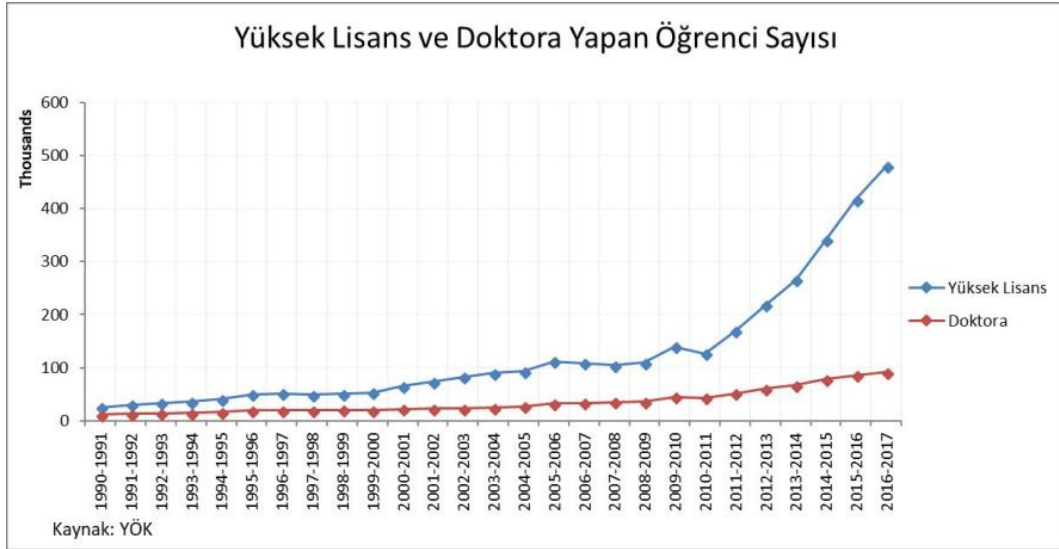
2.1. Giriş

Üniversite ve yüksek lisans eğitimlerinde hazırlanmakta olan tezleri kontrol etmek günümüzde zaman kavramının kısılması sebebiyle çok vakit almaktadır. Bununla ilgili olarak veri bütünlüğü bozulmadan öğrenciler tarafından hazırlanmış olan tez dokümanlarının kontrolü elzemdir.

2.2. Projenin Plan Kapsamı

Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin güvenilirliğinden yola çıkılmıştır. Hazırlanan tez dokümanlarının kontrolü vakit kaybına fırsat vermeden kısa sürede test edilecektir. Günümüzde yüksek lisans ve doktora yapan öğrenci sayısındaki artış göz önüne alındığında hazırlanan tezlerdeki üretkenlik kontrolünün zaman/maliyet oranını düşürmek ve işlemlere hız kazandırmak bu uygulamanın üniversitelerde hem jüriler için hemde danışman görevliler için kullanılmasını gerekli kılmıştır.

DOCHECKER NEDEN GEREKLİ?



Şekil 2.1 Yüksek Lisans ve Doktora Yapan Öğrenci Sayısı

1990 yılından 2017 yılına kadar tez çalışmaları yapan öğrencilerin grafiği yukarıda belirtilmiştir. Bu orandaki büyük ve hızlı yükseliş karşısında tez sayısının artması beraberinde bazı sorunları da getirmiş oldu.

- Tezlerde yapılmış olan intihallerin tespiti için gereklidir.
- Tezlerin uyması gereken şablonların kontrolü için gereklidir.
- Tezlerdeki şablon hatalarının tespiti için gereklidir.
- Tez içerisinde gözden kaçabilecek olan yazım hatalarının tespiti için gereklidir.
- Jüri onayı öncesi olabilecek hataların raporlanması ve hataların düzeltilmesi için gereklidir.

DOCHECKER ÇALIŞMA MANTIĞI NASILDIR?

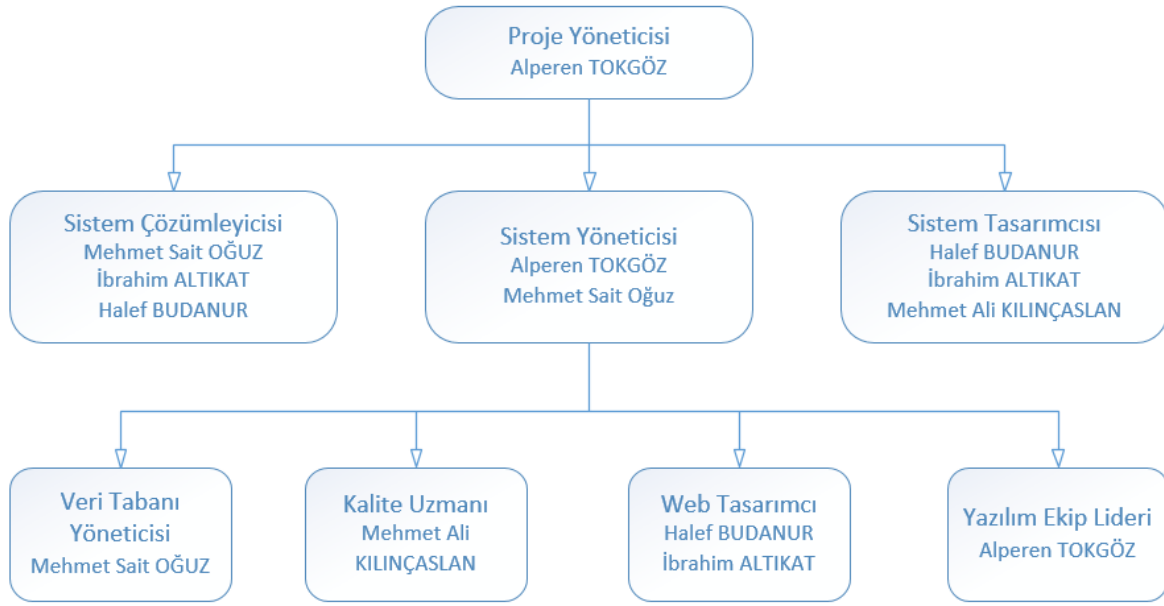
Sisteme yüklenen doc uzantılı tez taslak dosyasını; Fırat Üniversitesi Tez Yazım Kuralları dökümanında metnin font, boyut, girinti vb., sayfaların margin ölçüleri, kaynakça kullanımı gibi belirlenmiş olan kurallarına göre inceleyip kullanıcıya hatalarını ve nerede olduğunu gösterir.

2.3. Proje Zaman-İş Planı



Şekil 2.2 Proje Zaman-İş Planı

2.4. Proje Ekip Yapısı



Şekil 2.3 Proje Ekip Yapısı

Proje Yöneticisi	-Projenin yönetilmesi -Proje ekip yapısının oluşturulması -İş planlamasının yapılması
Sistem Çözümleyici	-Dökümantasyon ve raporlamanın hazırlanması
Araştırma Ekibi	-Proje için gerekli kaynakların temin edilmesi -Örnek sistemlerin incelenmesi
Veri Tabanı Ekibi	-Veri tabanı sistemlerinin oluşturulması -Veri tabanı tasarımının oluşturulması
Web Tasarım	-Arayüz tanımlamaları çalışmaları -Görsel tabanlı ve grafik tabanlı yazılım ihtiyaçlarının belirlenmesi -Tasarım ve kodlama çalışmaları
Programcı	-Proje arka plan kodlama
Test Ve Bakım Ekibi	-Sistem testlerinin yapılması -Sistem bakım planının yapılması

Şekil 2.4 Görev Açıklamaları

2.5. Kullanılan Özel Geliştirme Araçları ve Ortamları

Çözümleme ve Tasarım	Program Araçları	Sınama Araçları	Destek Araçları
CSS	ASP.NET	Microsoft Windows	Google Chrome
HTML	PYTHON	XP, 7, 8.1, 10	GitHub
	DJANGO	Google Chrome	Stackoverflow
	RestAPI	Yandex, Opera,	
	SQL Server	Mozilla, Brave,	
	Visual Studio	Chromium	

Tablo 2.1 Kullanılan Programlar

2.6. Proje Standartları, Yöntem ve Metodolojiler

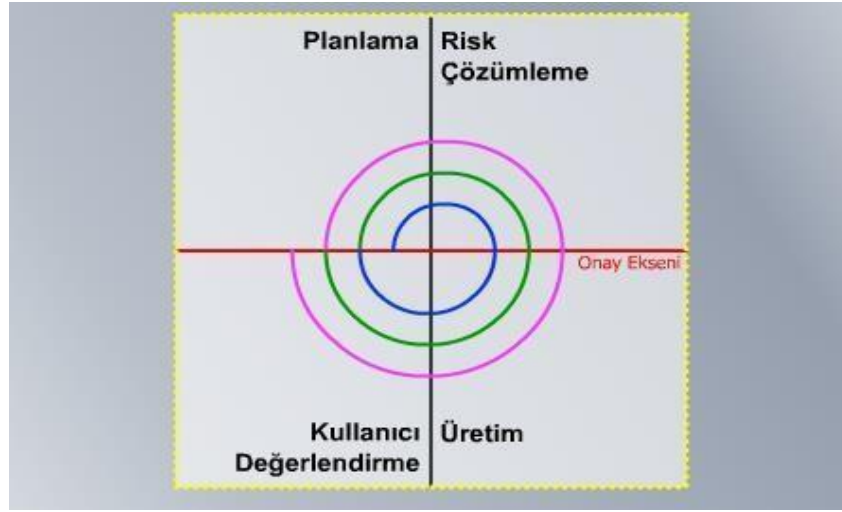
Spiralin başladığı ilk çeyrek içinde ilk isterler toplanır ve buna göre proje planlaması yapılır. İkinci çeyrekte, ilk tanımlanan ister lere göre risk çözümlemesi yapılır. Üçüncü çeyrekte, risk çözümlemesi sonunda ortaya çıkan isterlerin tanımlanmasındaki belirsizlikleri ortadan kaldırmak için prototipleme 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 tanımlanan ilk döngü bir sonraki döngü için bir girdi oluşturur.

Aşama	Kullanılan Yöntem/Araçlar	Ne İçin Kullanıldığı	Çıktı
Planlama	<ul style="list-style-type: none"> - Veri Akış Şemaları, - Süreç Belirtileri, - Görüşme, - Maliyet Kestirim Yöntemleri - Proje Yönetim Araçları 	<ul style="list-style-type: none"> - Süreç İnceleme - Kaynak Kestirimi - Proje Yönetimi 	Proje Planı
Çözümleme	<ul style="list-style-type: none"> - Süreç Belirtileri, - Veri Akış Şemaları, - Görüşme, - Nesne İlişki Şemaları, - Veri Sözlüğü 	<ul style="list-style-type: none"> - Süreç Çözümleme - Veri Çözümleme 	Sistem Çözümleme Raporu
Çözümlemeden Tasarıma Geçiş	<ul style="list-style-type: none"> - Akışa Dayalı Çözümleme, - Süreç Belirtilerinin Program Tasarım Diline Dönüştürülmesi - Nesne İlişki Şemalarının Veri Tablolarına Dönüştürülmesi 	<ul style="list-style-type: none"> - Başlangıç Tasarım - Ayrıntılı Tasarım - Başlangıç Veri Tasarımı 	Başlangıç Tasarım Raporu
Tasarım	<ul style="list-style-type: none"> - Yapısal Şemalar - Program Tasarım Dili - Veritabanı Tabloları - Veri Sözlüğü 	<ul style="list-style-type: none"> - Genel Tasarım - Ayrıntılı Tasarım - Veri Tasarımı 	Sistem Tasarım Raporu

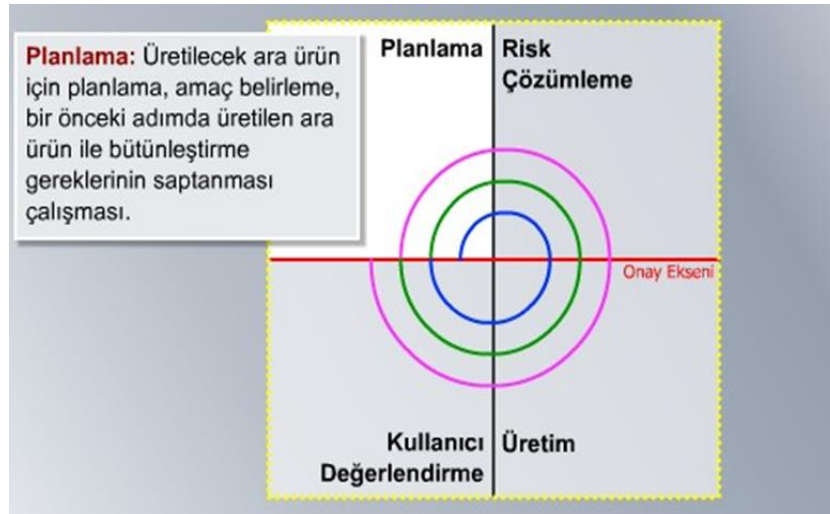
Şekil 2.5. Yazılım Yaşam Döngüsünün Açıklamaları

Projede planlama aşamasında ihtiyaç analizi yapılmış olup ne yapılacağına karar verildikten sonra tasarım aşamalarında veriler çözümlenip tasarım bitirilip proje sunuma hazır hale getirilmiştir. Projede helezonik model kullanılmıştır.

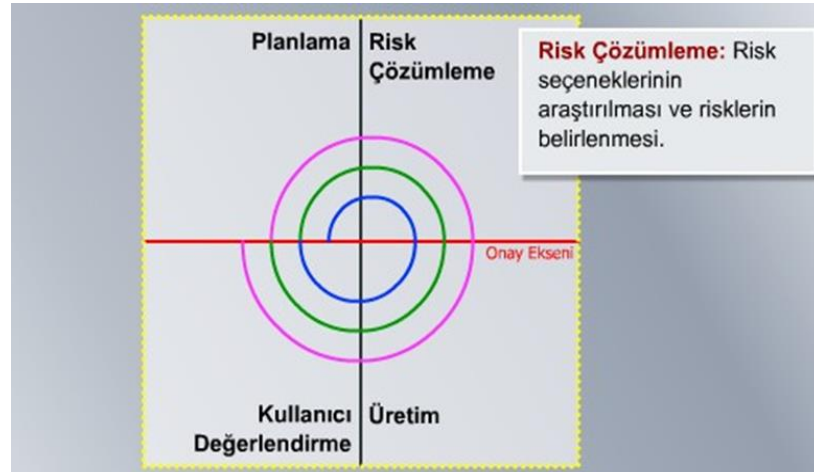
Helezonik Model Aşama Tanımları



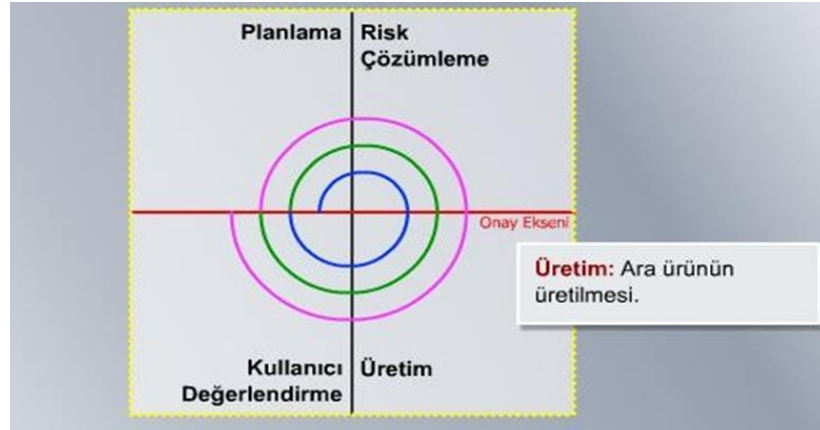
Şekil 2.6 Helezonik Model Yapısı



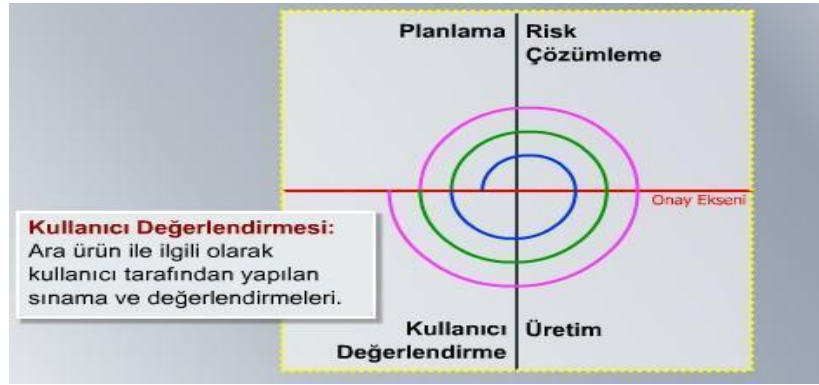
Şekil 2.7 Planlama Açıklama



Şekil 2.8 Risk Çözümleme Açıklama



Şekil 2.9 Üretim Açıklama



Şekil 2.10 Kullanıcı Değerlendirmesi Açıklama

2.7. Kalite Sağlama Planı

EKONOMİ
YENİDEN KULLANILABİLİRLİK
GÜVENİLİRLİK
DOĞRULUK
TAŞINABİLİRLİK
UYUMLULUK

Tablo 2.2 Uygulama Özellikleri

1. EKONOMİ → Az bir bütçe ile etkili işlemler yapmak hedefidir.
2. YENİDEN KULLANILABİLİRLİK → Projede kullanılan kodlar geliştirilebilir ve yeniden kullanılabilir.
3. GÜVENİLİRLİK → Sisteme yüklenen dosyalar depolanmamaktadır. Kullanıcı girişleri veritabanına şifreli bir şekilde kaydedilmektedir.
4. DOĞRULUK → Doğruluğunun kesinliği Fırat Üniversitesi tez yazım kurallarından sağlanmaktadır.
5. TAŞINABİLİRLİK → Tarayıcı tabanlı bir proje olduğundan her sisteme uygundur.
6. UYUMLULUK → Program birden fazla dil ile gerçekleştirildiğinden uyumluluk hat safhadadır. Ek olarak web tabanlı yapılacağından yayınlanabilir ise her tarayıcıda desteklediğinden uyumluluk konusunda bir problem çekilmeyecektir.

2.8. Konfigürasyon Yönetim Planı

Proje dokümantasyonunda oluşabilecek sorunlara ve/veya yeniliklere karşı planlamalar aşağıdaki şekilde listelenmiştir:

- Yeni kullanıcının sisteme gelmesi durumunda yeni üyelik açma işlemi
- Programa null bilgi dahil edilmek istendiğinde istisna mesajları ile uyarma işlemi
- Çıkış yapılmak istendiğinde

durumları için konfigürasyon yönetim planı oluşturuldu.

2.9. Kaynak Yönetim Planı

Kaynak olarak Programlama dersi kapsamında çekilen ders kayıtları, kullanılan tanımlamalar ve anahtar sözcükler araştırılarak gerekli doküman ve verileri elde edilmiştir. Kaynaklar belirlenirken mevcut kurallara en iyi cevap verenler seçilerek mükemmeliyetçi bir yaklaşıma varılacaktır.

2.10. Eğitim Planı

Sistem web tabanlı olduğundan kullanıcılara verilecek olan eğitim Visual Studio, SQL server, Microsoft Office programları hakkında temel bilgiler ve kullanım sırasında oluşabilecek istisnalara karşı programların hata mesajlarının açıklamalarıdır.

KONU	SÜREÇ
Giriş ve ön tanıtım	1-2. Gün
Program değişken, sembol, görünüm vb. hakkında bilgi	3-25. Gün
Kullanımının denemesi ve testi	25-29. Gün
Sonuç	30. Gün

Tablo 2.3 Eğitim Planları

Eğitim planı 1 ay içerisinde her programa uygulanacaktır. 1 günde ayrı ayrı bölümler ile 1 ayda tüm programların eğitimi bitirilecektir.

2.11. Test Planı

- Sistem kullanıma hazır bir duruma getirildiğinde sınama işlemlerine tabi tutulacaktır.
- Hata oluştuğu takdirde direk müdahale edilerek giderilecektir. Bu işlem hata ve/veya sorun kalmayınca dek sürdürülecektir.
- Hata kalmadığı anlaşıncaya dek bahsedildiği gibi kullanıma sunulacaktır.

2.12. Bakım Planı

Sistemin kurulduğu her otoparka 1-3-6-9-12 aylarda 1 defa genel bakım bunun haricinde de rutin olmayan arıza gibi durumlarda bakım yapılacaktır. Aşağıda bakım tablosu mevcuttur.

Oca k	Şuba t	Mar t	Nisa n	Mayı s	Haziran	Temmuz	Ağustos	Eylü l	Eki m	Kası m	Aralı k
+		+			+			+			+
Rutin kontroller haricinde arıza kontrolleri her zaman yapılabilmektedir.											

Şekil 2.11 Bakım Planı

2.13. Projede Kullanılan Yazılım/Donanım Araçlar

Yazılım Araçları:

Çözümleme ve Tasarım	Program Araçları	Sınama Araçları	Destek Araçları
CSS	ASP.NET	Microsoft Windows	Google Chrome
HTML	PYTHON	XP, 7, 8.1, 10	GitHub
	DJANGO	Google Chrome	Stackoverflow
	RestAPI	Yandex, Opera,	
	SQL Server	Mozilla, Brave,	
	Visual Studio	Chromium	

Tablo 2.4 Yazılımsal gereçler

Donanım Araçları: Sistem için gerekli donanım araçları; Bilgisayar ve Sunucudur.

3. SİSTEM ÇÖZÜMLEME

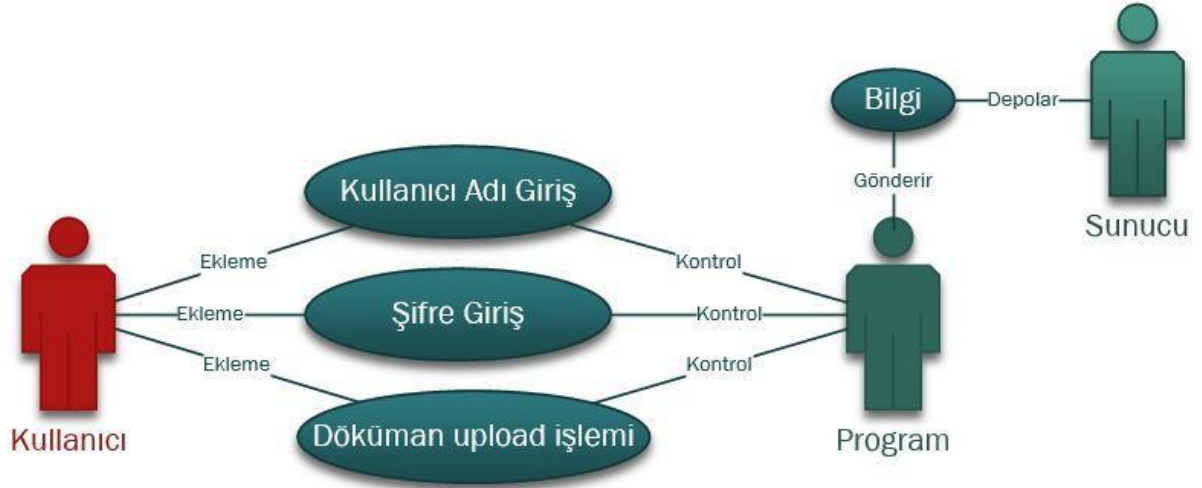
3.1. Mevcut Sistemin İncelenmesi

Yapılması planlanan uygulamamız hem üniversitemizde hem de Türkiye Cumhuriyeti'nde bir ilk olacağı için karşılaştırma veya kıyas yapılabilecek herhangi bir uygulama bulunmamaktadır. Bununla birlikte string metotları ve bunların kullanım yolları derinlemesine incelenmiş olup planlanan sisteme nasıl katkı yapacağı araştırılmıştır. Çeşitli metotlar hazırlanarak sisteme uygulanmıştır.

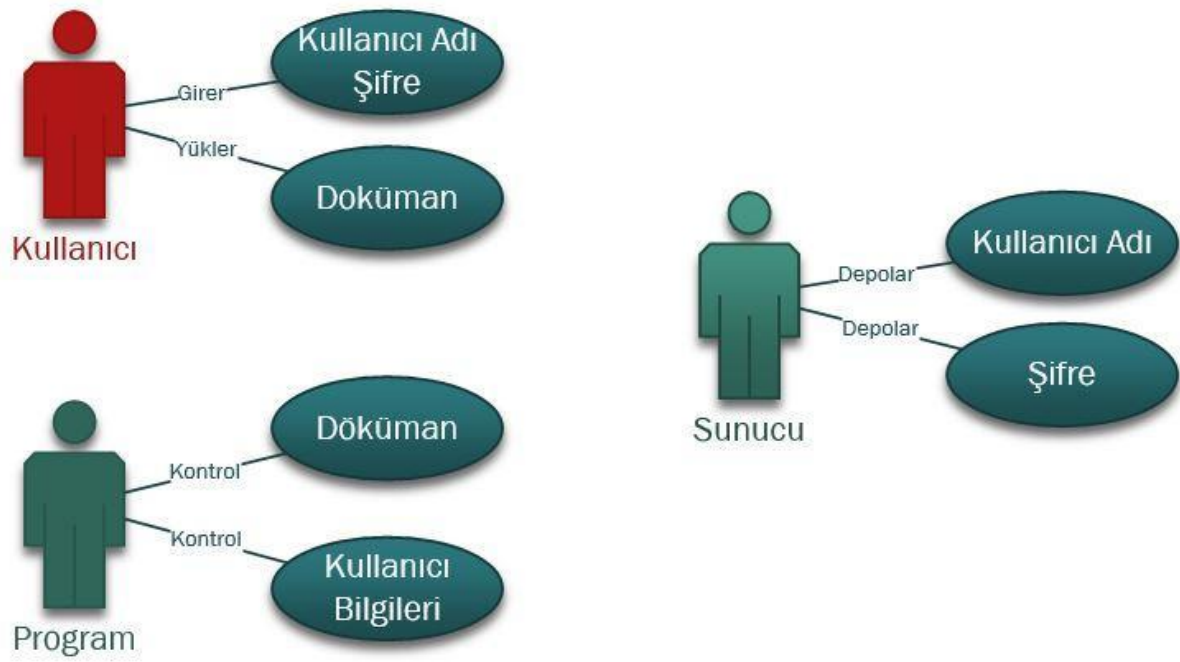
3.1.1. Örgüt Yapısı

Örgüt yapısı olarak Yazılım Mühendisliği 3. sınıf öğrencilerinden Alperen TOKGÖZ, İbrahim ALTİKAT, Halef BUDANUR, Mehmet Sait OĞUZ ve Mehmet Ali KILINÇASLAN olarak yapılan ortak çalışma ile örgüt yapısı oluşturulmuştur.

3.1.2. İşlevsel Model



Şekil 3.1 Use-Case Diyagramı



Şekil 3.2 Temel Görevlendirme

3.1.3. Veri Modeli

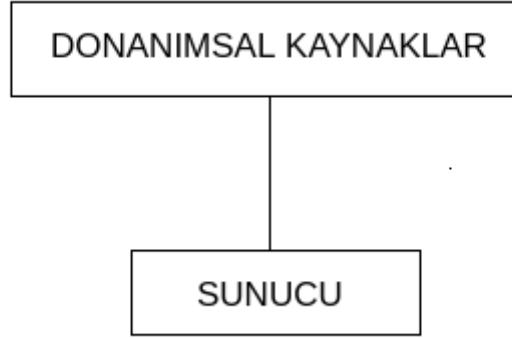
Sistemin veri modeli olan tablo ve ilişkisel yapı aşağıdaki şekil ile açıklanmıştır. Sistemdeki açıklık ve eksiklikler kendi yaptığımız sistemde karşılanmıştır.



Şekil 3.3 Veri Modeli

3.1.4. Var Olan Yazılım/Donanım Kaynakları

- VS Code, Visual Studio, Pycharm
- Web Sunucu



Şekil 3.4 Donanımsal Kaynaklar

3.1.5. Var olan Sistemin Değerlendirilmesi

Yukarıda var olan sisteme ek olarak kullanıcılar sistemde kayıtlı olan bilgileri ile sisteme giriş yapacaklardır. Eğer giriş bilgilerinde hata varsa kullanıcı sisteme giremeyecektir. Sistem üzerinde daha önce yaptığı tez kontrollerine dair raporlar veri tabanında saklanacaktır. Hataların düzeltilmesinde etkin bir kolaylık sunacaktır. Bu sayede hem danışman hatalarının hem de öğrenci hatalarının nerede olduğu ve bu hata eğilimlerinin hangi konularda yaşandığı tespit edilebilecektir.

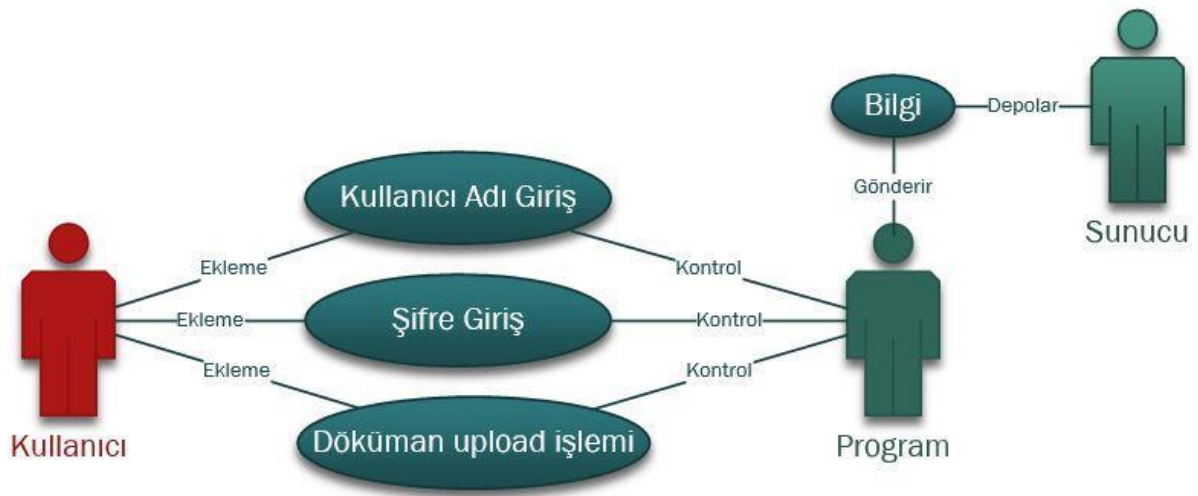
Açıklandığı üzere sistem zamandan tasarruf sağlayacaktır bununla birlikte gözden kaçabilecek hataların minimize edilmesini sağlayacaktır. Bu avantajlar sayesinde de kullanıcıya daha verimli bir zaman sunmak ve sağlamak amaçlarımız arasındadır.

3.2. Gereksenen Sistemin Mantıksal Modeli

3.2.1. Giriş

Mevcut sistem incelendiğinde eksikleri ve gereksinimleri belirledikten sonra oluşturmak istediğimiz sistem alanında tek olduğu belirtilebilecektir. Her ne kadar NLP projelerinde spell check ve bunun gibi işlemler yapılıyor olsa da bu kadar kapsamlı bir çalışma alanında öncü olacaktır. Tez kontrollerinin insan marifetiyle yapılması zamanla sorunlara yol açtı. Tarafımızca üretilecek olan sistem bu işlemleri otomatize etmek üzerine kurulmuştur.

3.2.2. İşlevsel Model



Şekil 3.5 Use-Case Diyagramı

DOCHECKER Programının Kontrol Edeceği Kurallar (Gramer) Nelerdir?

- 1) Çift tırnak içerisinde 50 den fazla karakter bulunamaz.
- 2) Onaylanan tez başlığı ile kitapçık kapağındaki tez başlığının aynı olması
- 3) Şekil üst kenarı ile metin arasında 10 punto tek aralık 2 satır boşluk (24 nk - nokta) bulunur.
- 4) Metin içerisinde, “parantez içinde yazarın soyadı, virgül, eserin yayım yılı, iki nokta üst üste, kaynağın sayfa numarası” yazılmak suretiyle (Atay, 1998:12) verilmelidir.
- 5) Tezin metin kısmı 12 punto, dipnotlar 10 punto ile yazılmalıdır.

- 6) Kaynakça da ana kaynaklardan başlayarak yazar soyadlarına göre alfabetik sırada ve aynı yazarın eserleri kullanılmışsa eserler tarih sırasıyla bibliyografik usulle yazılır.
- 7) Tezin en sonunda aday özgeçmişi bulunmak zorundadır.
- 8) Tezin Giriş başlığı kesinlikle 1.Giriş şeklinde başlamalıdır.
- 9) Kaynak bir gazete makalesi ise; yazarın soyadı, adının baş harfi, makalenin tam adı, gazetenin tam adı, yayın tarihi, sayfa numarası gibi.
- 10) Tablolar varsa tablolara atıf yapmak zorundadır.
- 11) Şekiller varsa şekillere atıf yapmak zorundadır.
- 12) Kaynaklar varsa kaynaklar belli yazım standartlarına uymak zorunda
- 13) Dergi kaynağı ise yazım şablonu şudur
- 14) Kitap kaynağı ise alıntı yazım şablonu şudur
- 15) Yazar isimleri noktalı virgül ile ayrılır.
- 16) İki satırdan az paragraf var mı ?
- 17) Önsöz bölümünün ilk paragrafında teşekkür ibaresi yer almamalı, alıyorsa rapora eklenir
- 18) İçindekiler tablosundaki bölüm sayfa numaraları ile bölümlerin geçtiği sayfa numaraları tutarlı mı ? Sayfalar tutarlı değilse raporla
- 19) Kaynak bir kitap ise, yazarın soyadı, adının baş harfi varsa cilt numarası, kaçınıcı baskı olduğu, yayınlayan kuruluşun adı, basıldığı yer, tarih ve alıntı yapılan sayfa yazılır.
- 20) Kaynak bir tez ise; yazarın soyadı, adının baş harfi, tezin tam adı, () içerisinde tezin yapıldığı Enstitü / Üniversite, yeri ve yılı ve sayfası.
- 21) Kaynak bir tebliğ ise, yazarın soyadı, adı () tarih, “ “ tebliğin tam adı, farklı karakterlerde ilmi toplantıyı hazırlayan kuruluşun ve toplantının adı, basım yeri, ve ilk ve son sayfası.
- 22) Kaynak bir makale ise; yazarın soyadı, adı () tarih, “ “ makalenin tam adı, farklı karakterde (italik-bold) derginin adı veya bilinen kısa adı, c. , s. Basım yeri, ilk ve son sayfa

aynı yazarın aynı yılda birden fazla yayını varsa tarih yanına 1998/a - 1998/b ... gibi sıralama yapılır.

- 23) Özgeçmiş en son sayfada yer almalıdır.
- 24) İçindekiler kısmında geçen her başlık metinde 1 kere daha geçmeli
- 25) Açılan bir tırnak kapatılmalı.
- 26) Kullanılan toplam parantez sayısı 2 ve 2'nin katları şeklinde olmalı
- 27) Kullanılan çift tırnak sayısı 2 ve 2'nin katları şeklinde olmalı
- 28) Türkçe özet/abstract 200–250 kelime olarak yazılmış olmalı.
- 29) Özetin sonunda keywords'ler en az üç, en fazla beş anahtar sözcük yer almıştır.
- 30) Yüksek lisans tezlerinde 30000 kelimeden fazla kelime kullanılmamalıdır.
- 31) Simgeler için ayrı alfabetik bir dizin hazırlanmalıdır.
- 32) Kısaltmalar için ayrı alfabetik bir dizin hazırlanmalıdır.
- 33) Kaynak numaraları en az 2 adet olmalı. - [1] ifadesi- en az 2 adet olmalı.
- 34) Ek'ler en az 2 adet olmalı. Ek-1 ifadesi en az 2 adet olmalı.
- 35) İç kapakta üniversite adı, tez çalışması yapılan enstitü, Yüksek Lisans Tezi, xxxx anabilim dalı, xxxx programı şeklinde satır satır yazılmış olmalı.
- 36) Tezin yazım ayı ve yılı özet kısmında ve kapak kısmında geçmelidir.
- 37) Tezin yazım ayı ve yılı en az 2 kere tez içerisinde “Ocak 2020” şeklinde geçmelidir.
- 38) Tezin yazım ayı “Abstract” kısmında ingilizce olarak geçeceği için yazım ayının doğru yazılıp yazılmadığı ingilizce olarak da kontrol edilmelidir.
- 39) Tablo isimleri tez metni içerisinde sıralı halde olmalıdır.(Tablo-2 25. sayfada olupta Tablo-1 35. sayfada olmamalıdır.)
- 40) Sayfa numaraları en az iki kere tez içerisinde bulunmalıdır.

41) Hatırat, derleme, masal, ağıt, bilmece gibi araştırmalar için başvurulacak kaynak kişiler için şahsın adı-soyadı, baba adı, doğum tarihi ve yeri, okur yazar olup olmadığı ve tahsili, halen ikâmet ettiği yer yazılmalıdır. **(Ahmet Demirkıran; Murat oğlu, 1327 (1911) Kilis doğumlu, okur-yazar değil, halen Kilis Çalkaya (Cukanlı) köyü nüfusuna kayıtlı)**

42) Kaynak gazete makalesi ise; yazarın soyadı, adının baş harfı, makalenin tam adı, gazetenin tam adı, yayın tarihi, sayfa numarası.

43) Kaynak bir Ansiklopedi maddesi ise; biliniyorsa madde yazarının soyadı, adının baş harfı, “ “ içinde maddenin tam adı, ansiklopedinin farklı karakterde yazılmış tam adı veya bilinen kısaltması, cilt/ fasikül sayısı, basım tarihi ve sayfa numarası.

44) Metinler iki tarafa yaslı olmalıdır.

45) Denklemlere atıf yapılmış mı?

46) Giriş bölümünün son bölümünde tezin organizasyonu ve kapsamına yer verilmiş mi?

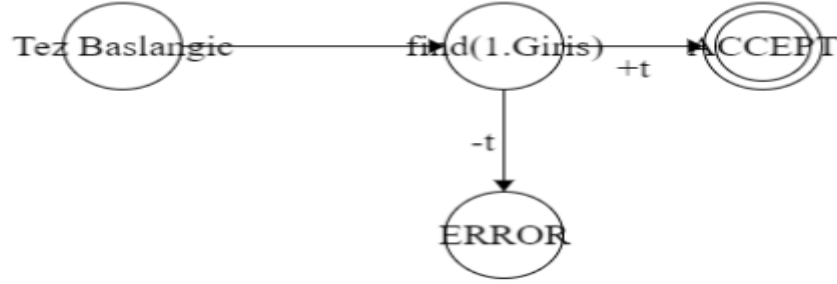
47) Tezin amacı kapsamına değiniyor mu tez amacı ve hipotezi gibi veriler geçiyor mu ?

48) Denklem numaraları bölüm başlıkları ile uyuyor mu?

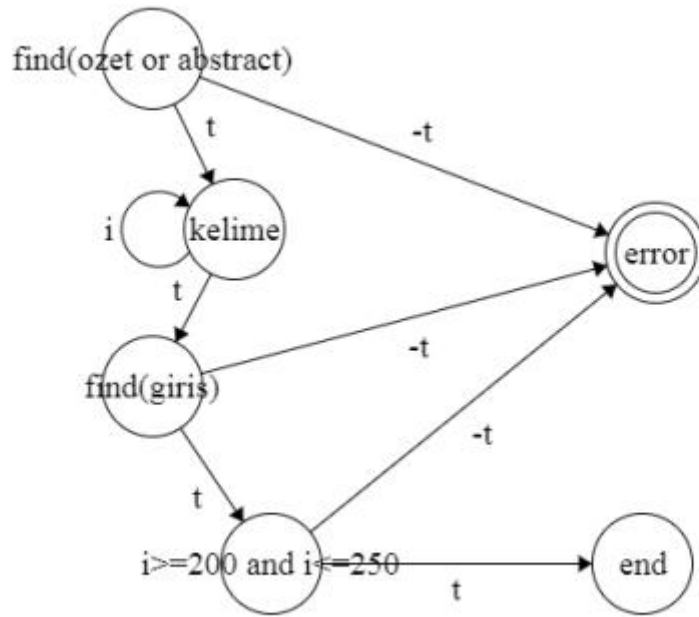
49) Tablo numaraları bölüm başlıkları ile uyuyor mu?

50) Şekil numaraları bölüm başlıkları ile uyuyor mu?

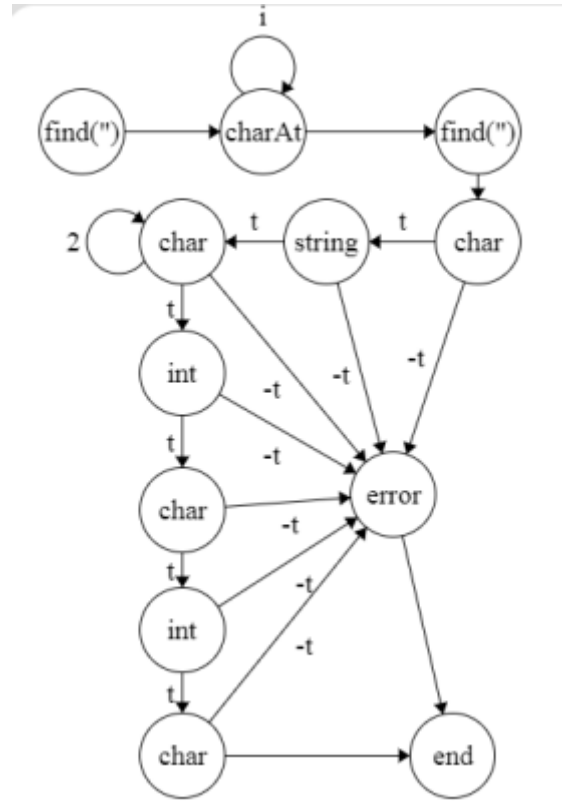
DOCHECKER Programına Ait DFA (Deterministic Sonlu Otomata) Diyagramları Nelerdir?



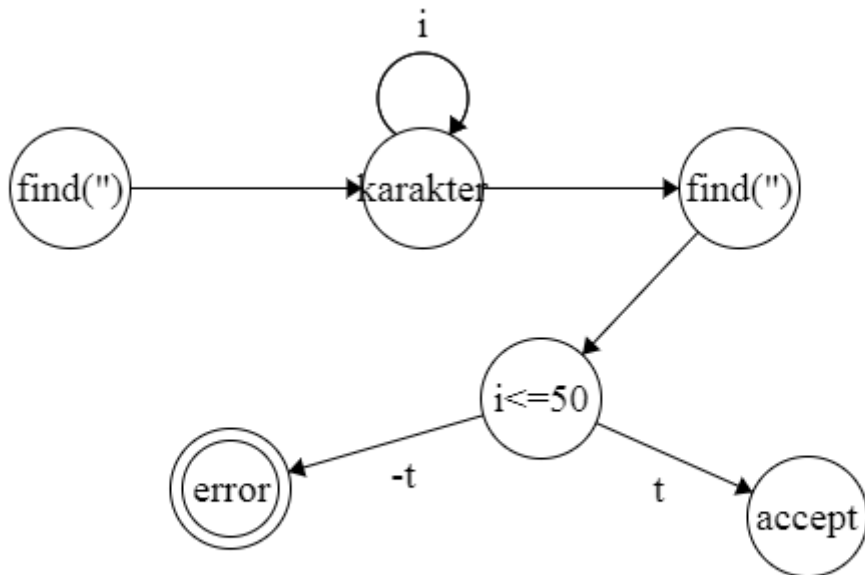
Şekil 3.6 Giriş Kontrol



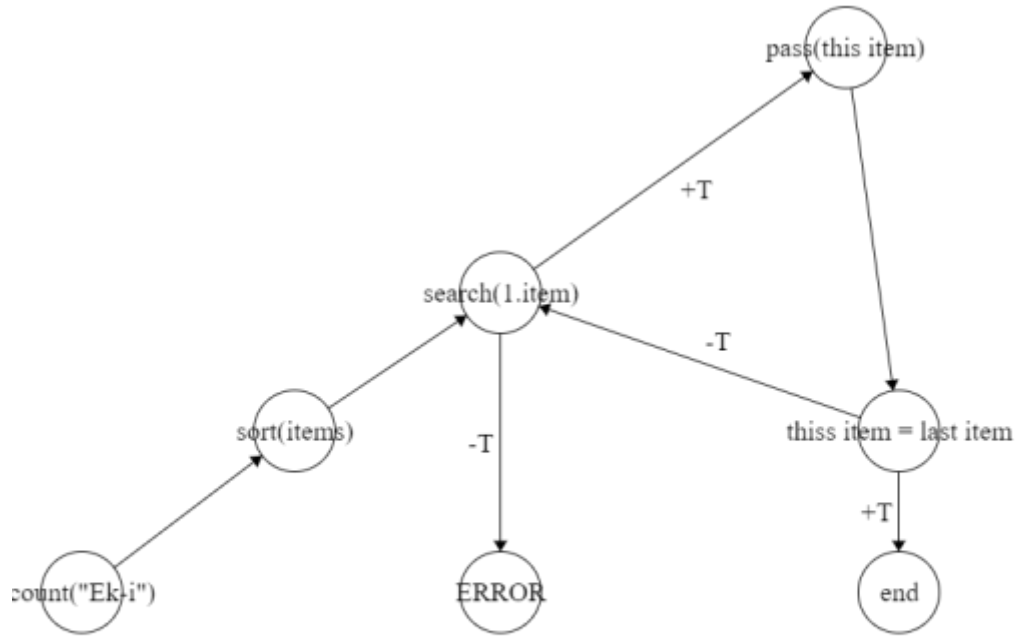
Şekil 3.7 Özet veya Abstract kelime sayısı kontrol



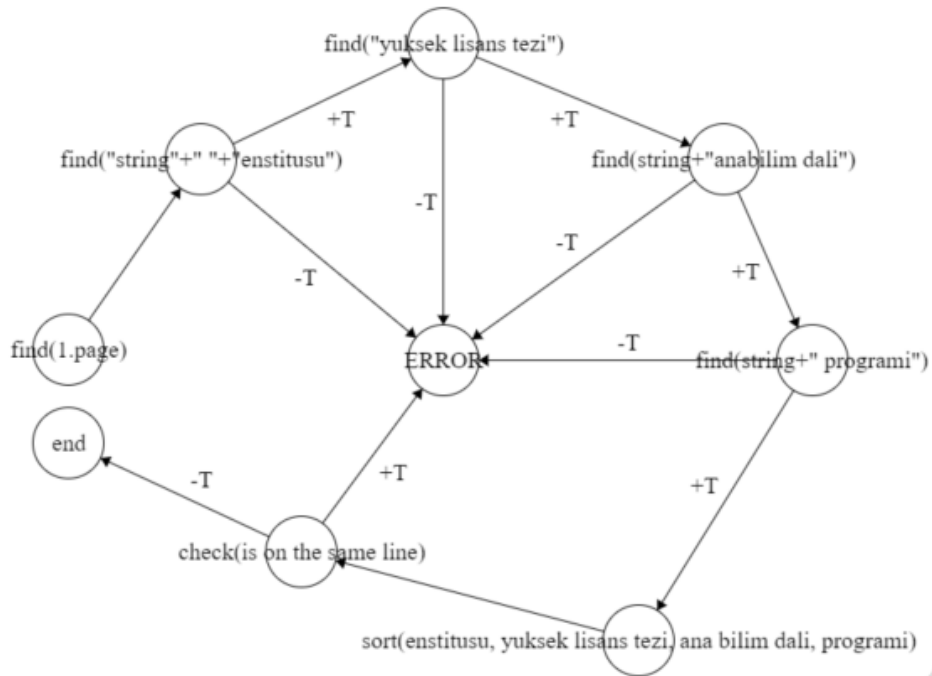
Şekil 3.8 Alıntı kontrol



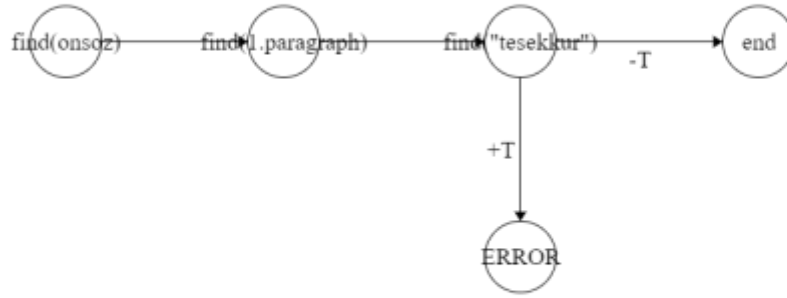
Şekil 3.9 Çift tırnak arası karakter kontrol



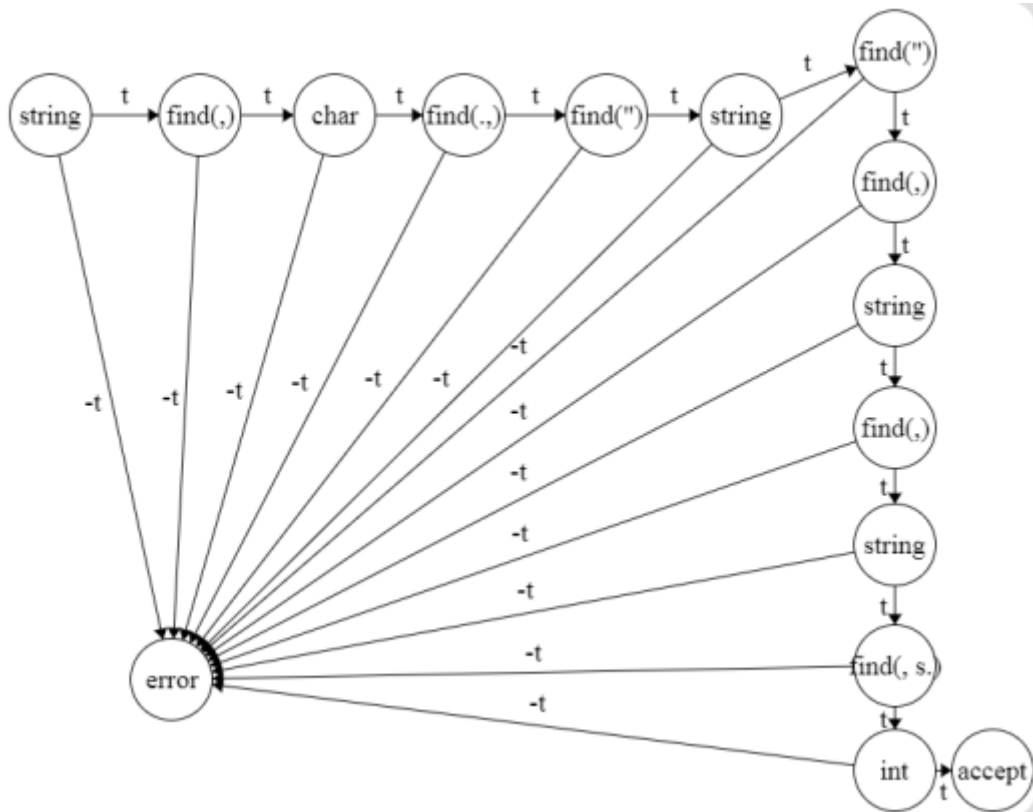
Şekil 3.10 Ek sayısı kontrol



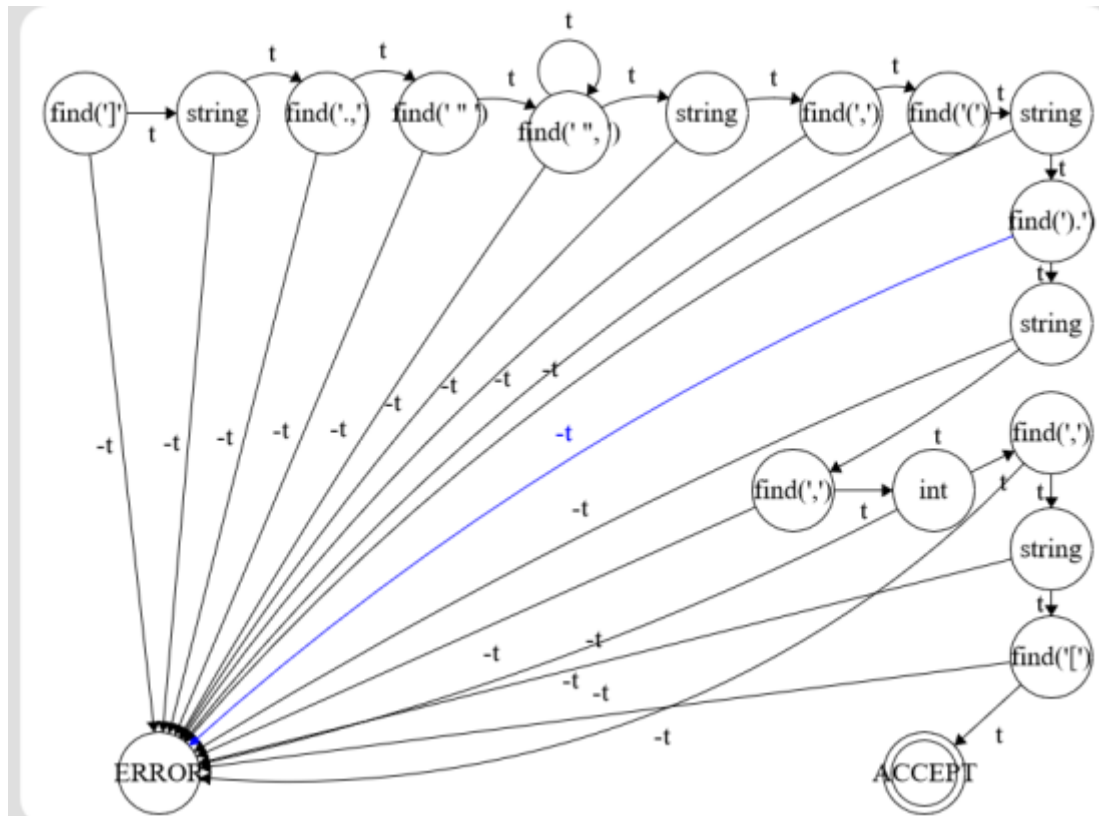
Şekil 3.11 İç kapak yazılma şekli kontrol



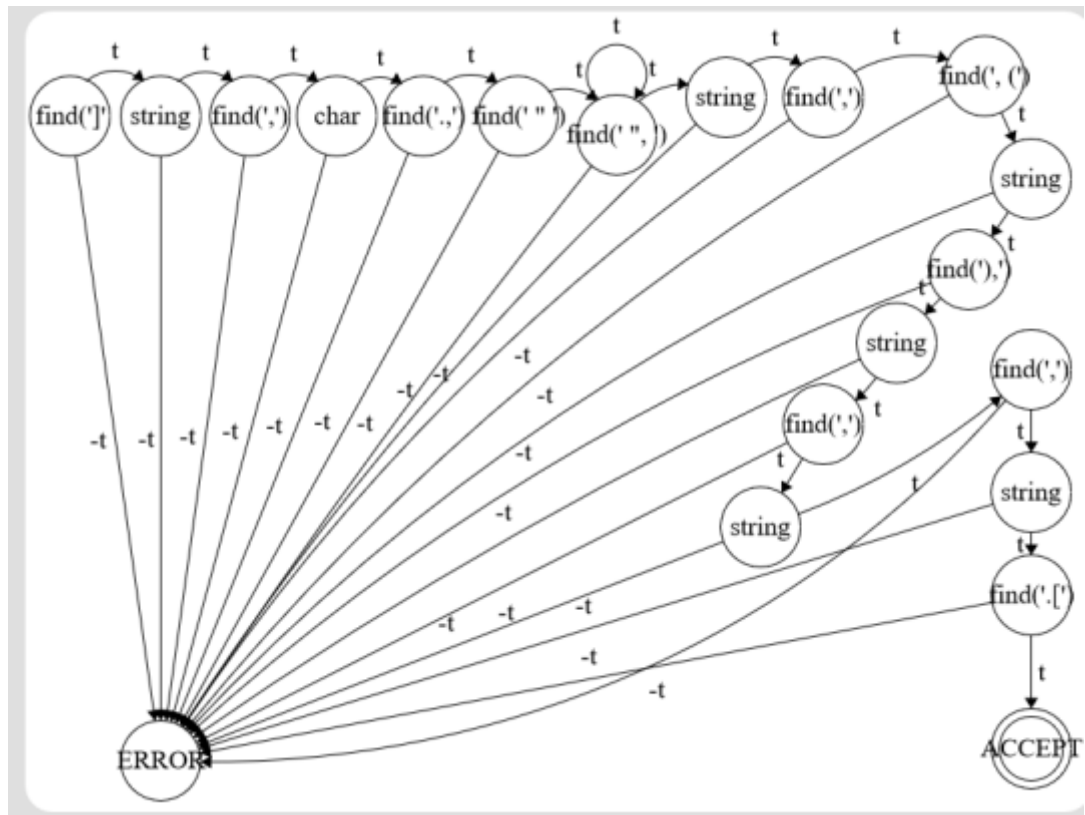
Şekil 3.12 İlk paragrafta teşekkür ibaresi kontrol



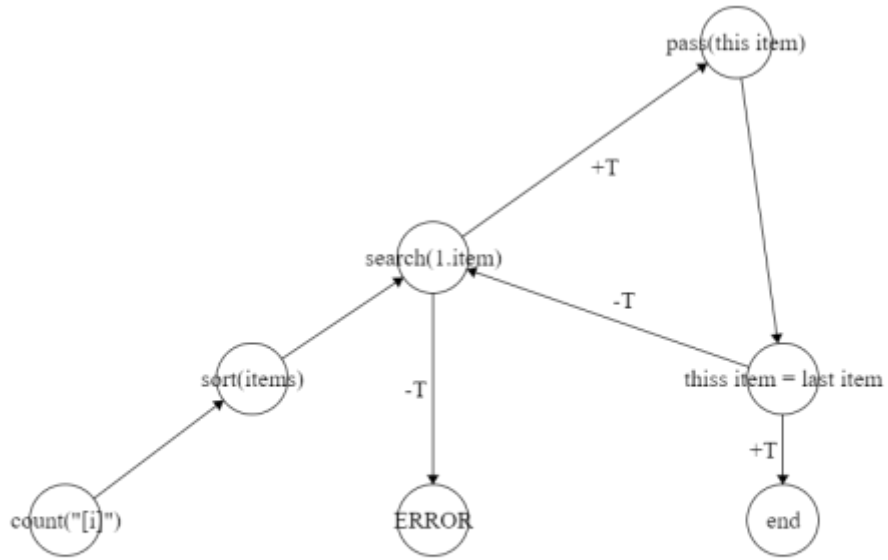
Şekil 3.13 Kaynak (Ansiklopedi Maddesi) kontrol



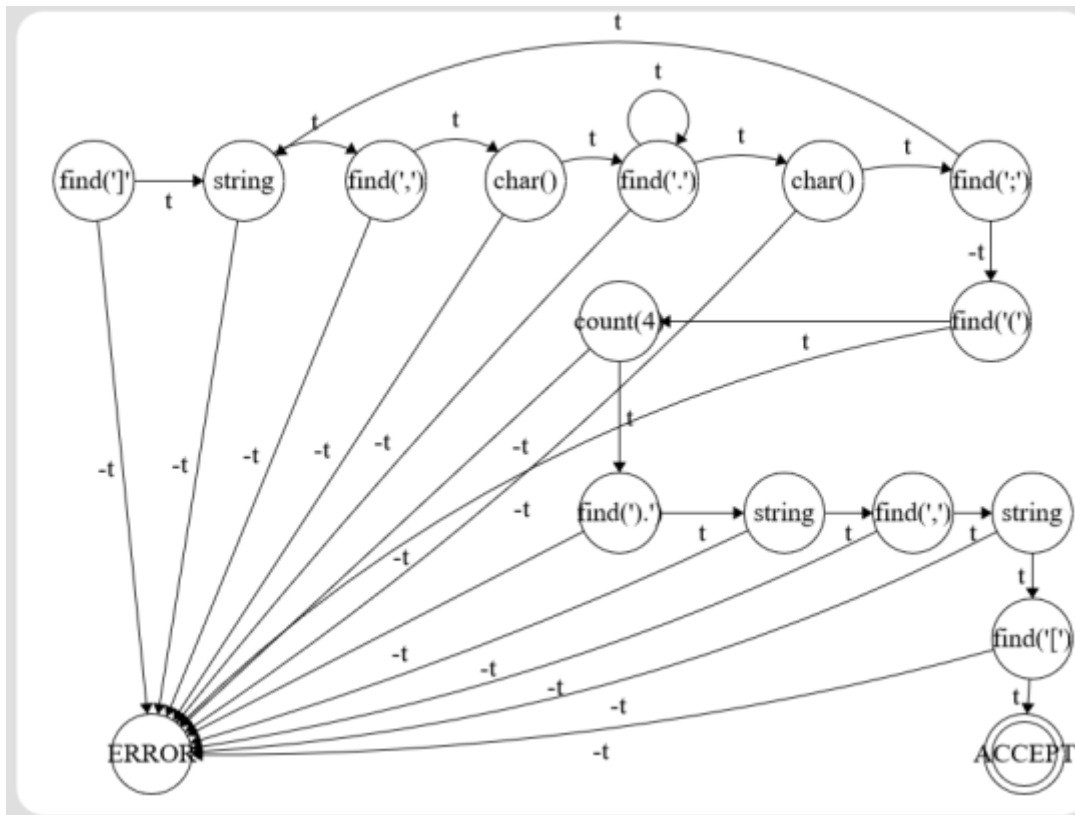
Şekil 3.16 Kaynak (Makale) kontrol



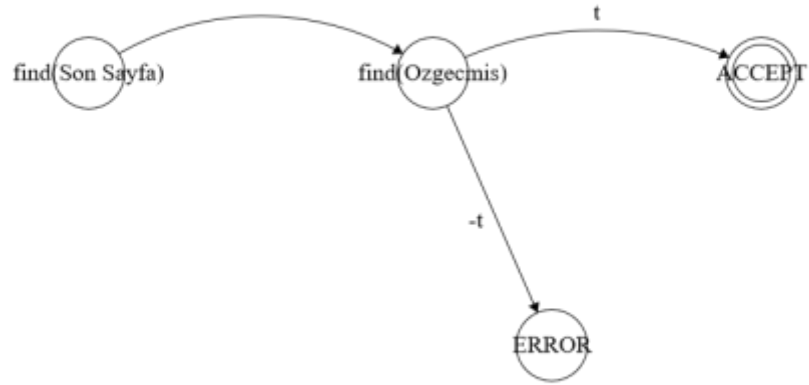
Şekil 3.17 Kaynak (Tebliğ) kontrol



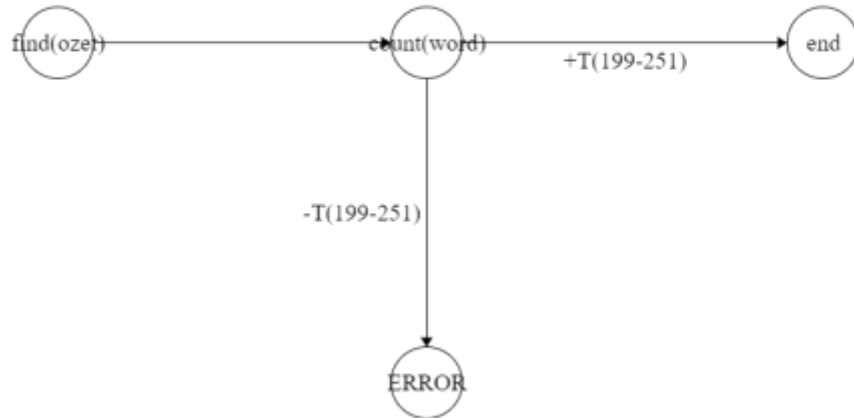
Şekil 3.18 Kaynak numaraları sayısı kontrol



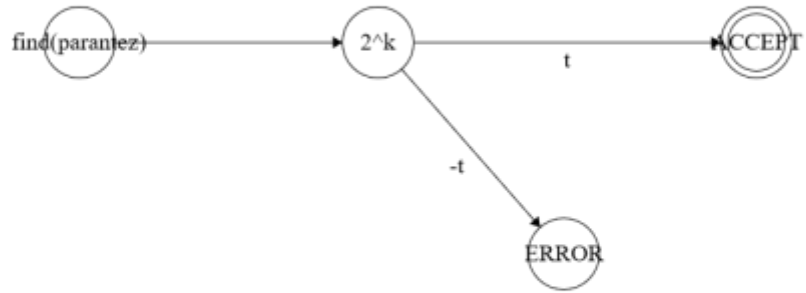
Şekil 3.19 Kaynak (Kitap) kontrol



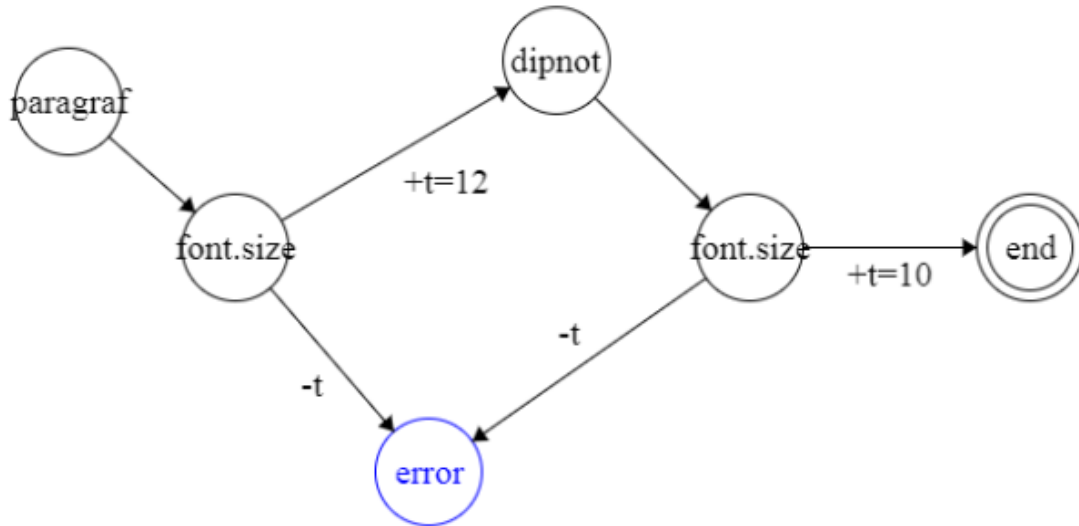
Şekil 3.20 Özgeçmiş sayfa kontrol



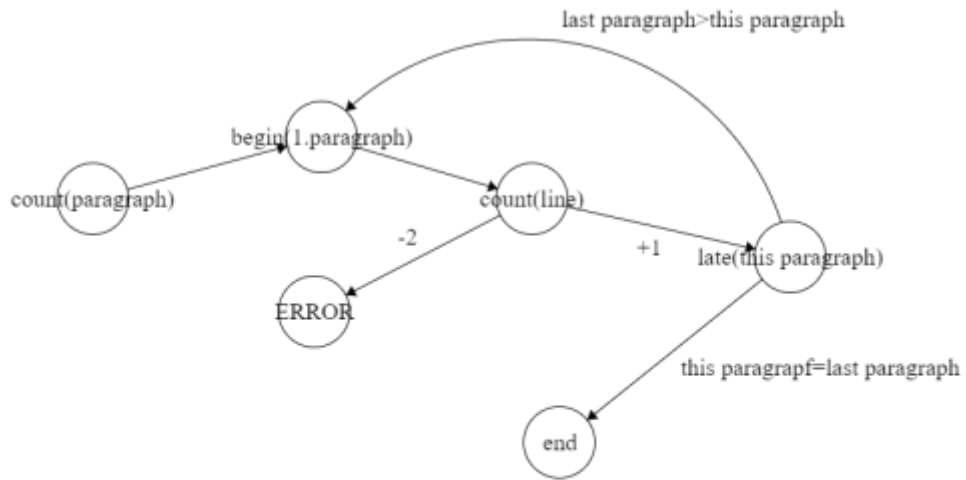
Şekil 3.21 Özet kelime sayısı kontrol



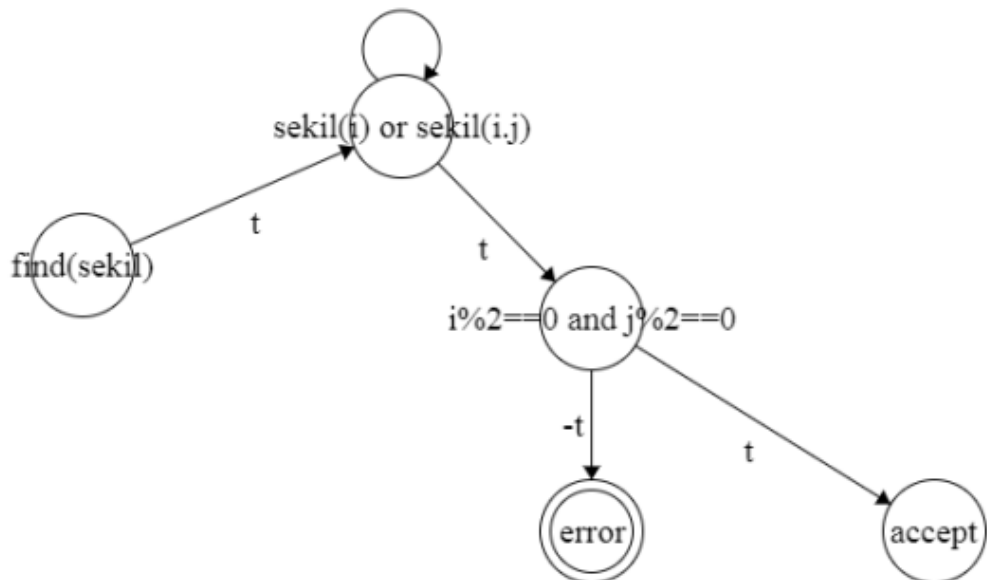
Şekil 3.22 Parantez kontrol



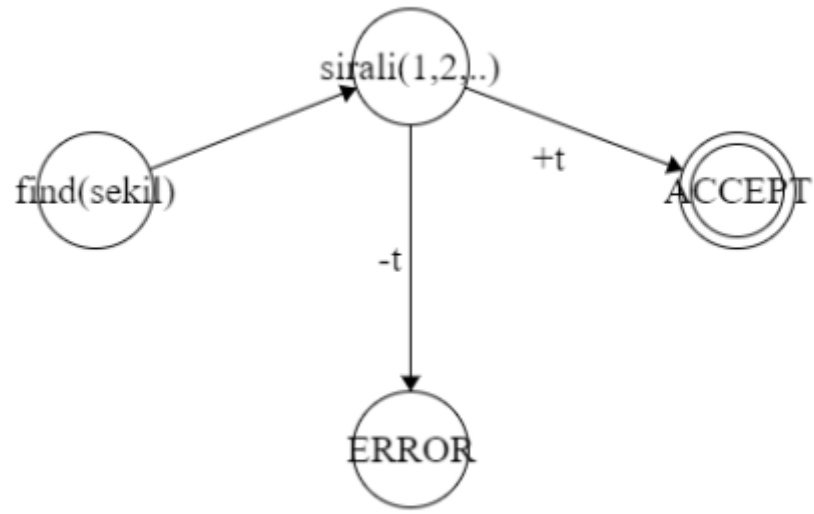
Şekil 3.23 Punto kontrol



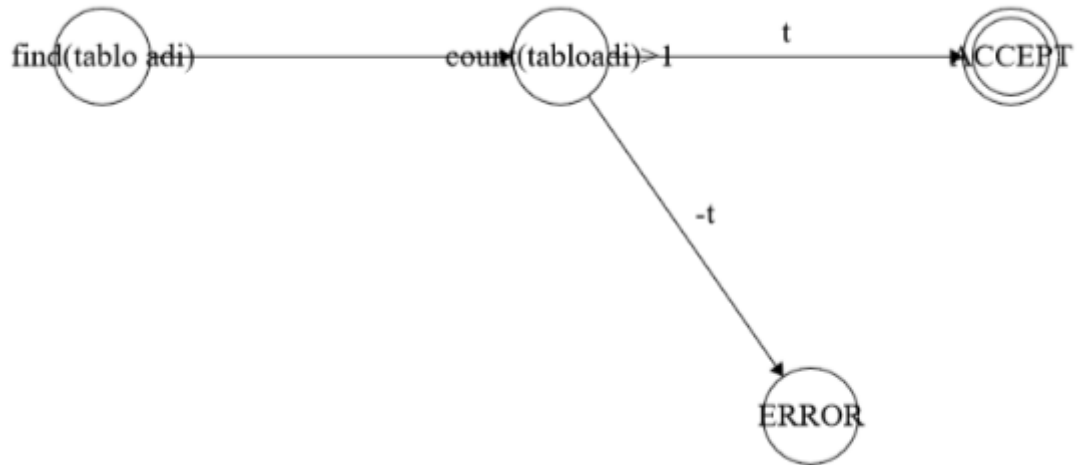
Şekil 3.24 Satır sayısı kontrol



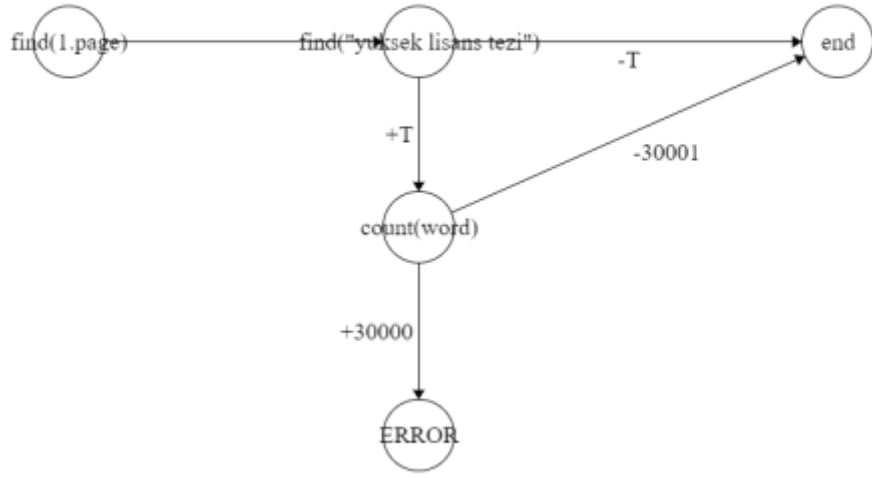
Şekil 3.25 Şekiller sayısı tutarlılığı kontrol



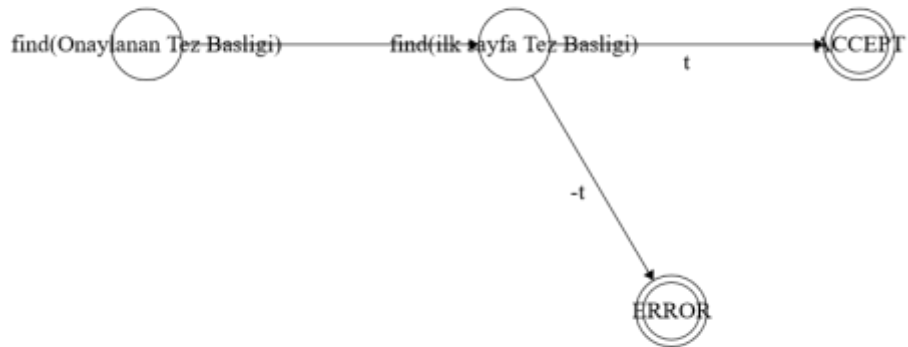
Şekil 3.26 Şekiller sıra doğruluğu kontrol



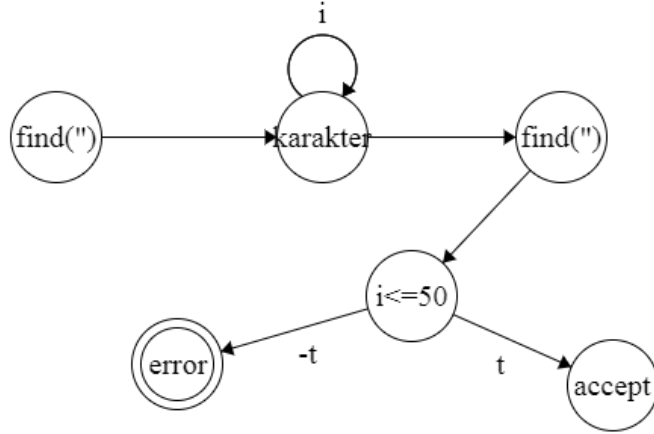
Şekil 3.27 Tablo sayısı kontrol



Şekil 3.28 Tez kelime sınırı kontrol



Şekil 3.29 Başlık kontrol



Şekil 3.30 Çift tırnak işareti karakter sayısı kontrol

3.2.3. Genel Bakış

Oluşturulan sistem incelendiğinde kullanıcıdan alınacak olan girdi dosyası üzerinden çeşitli string işlemleri gerçekleştirilerek geriye döndüreceği değerler üzerinden hareket edilerek sonuçlar elde edilecektir. Bu sonuçların raporlanması ile kullanıcılara bilgilendirme ve daha sonraki kontrolleri manuel gerçekleştirilmesine imkan verebilmek amacıyla checklist mantalitesinde çıktı vermektir.

3.2.4. Bilgi Sistemleri/Nesneler

Kullanıcı: Kullanıcı adı ve şifre ile sisteme giriş sağlayan ve üzerinden işlem yapılacak olan dökümanı sisteme yükleyecek olan aktördür.

Program: Kullanıcının bilgilerini kontrol eden ve döküman işlemlerini gerçekleştiren nesnedir.

Sunucu: Gerekli yapı nesnelerinden biridir. Depolama işlemleri sağlanacak olan aktördür.

3.2.5. Veri Modeli



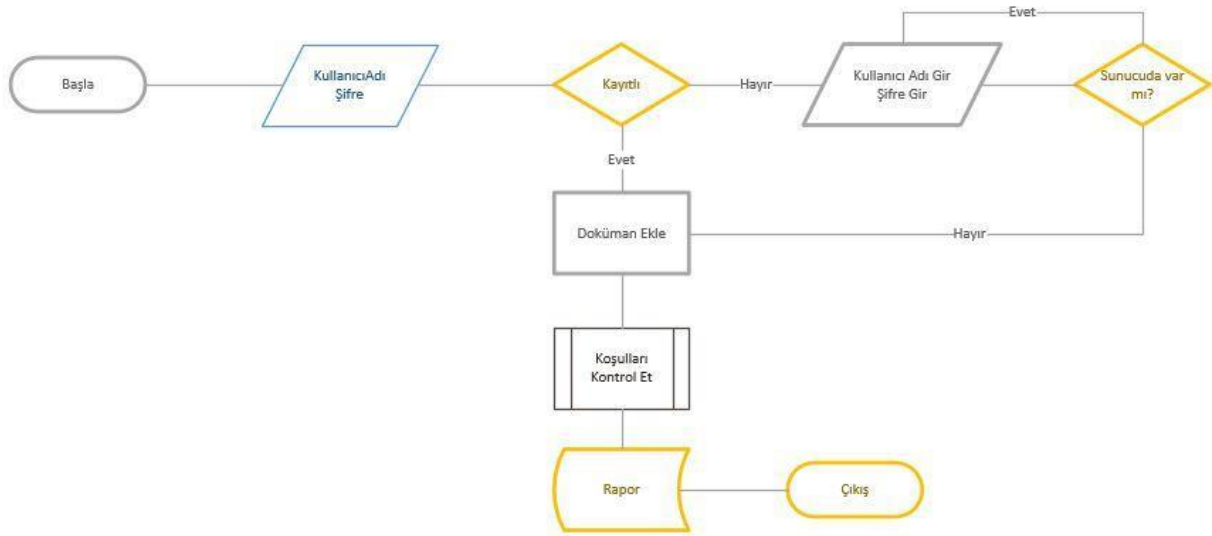
Şekil 3.31 Veri Modeli

3.2.6. Veri Sözlüğü

- Ana bilgilerin tanımlanması
 - ✓ kID □ *Kullanıcıya ait sistemde tanınmasını sağlayan integer değer*
 - ✓ kullanıcıAdi □ *String türündeki değer*
 - ✓ sifre □ *String türünde değer*
 - ✓ dosyaID □ *integer türünde bir değer*
 - ✓ doc □ *String türünde işlem yapılacak dosyayı oluşturan değer*
 - ✓ rapor □ *kontrol işlemlerinin sağlanmasının ardından görüntülenecek olan string türünde bir değer*

3.2.7. İşlevlerin Sıra Düzeni

Programa gelen müşteriden kullanıcı adı ve şifre istenilerek sunucuda olup olmadığının kontrolü sağlanacaktır. Kontrolün ardından kullanıcı ana işlem sayfasına yönlendirilecek ve yükleyeceği dökümanın kural kontrolü sağlanıp rapor sunulacaktır.



Şekil 3.32 İşleyiş

3.2.8. Başarım Gereklileri

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

- Hazırlanan tezin plagiarizm kontrolünün yapılması
- Hızlı giriş/çıkış işlemleri
- Veri tabanı bağlantısı
- Kullanım kolaylığı
- İTÜ NLP Api ihtiyacı
- Metot çalışma zamanlarının iyileştirilmesi

temel gereklilikler olarak tespit edilmiştir.

3.3. Arayüz (parça) Gereklilikleri

3.3.1. Yazılım Arayüzü

Gerekli olan her türlü değişiklik source kodları üzerinden yapıp tekrar derlenecektir.

3.3.2. Kullanıcı Arayüzü

Kullanıcının rahat kullanımına yönelik tasarım ve işleyiş sisteme eklenerek hız, tasarım, güvenlik gibi temel özellikler sağlanmıştır. Projede kullanıcı için arayüzü

tasarlanırken uyumlu renkler seçilerek rahat büyük puntolu yazı tipleri seçilerek sade bir arayüz tasarlanacaktır.

3.3.3. İletişim Arayüz

İletişim mail yoluyla sağlayabilecektir. Güvenlik ve istismar amacıyla site üzerinden bir iletişim formu ile değil mail yoluyla feedbackler alınabilecektir.

3.4. Belgeleme Gerekleri

3.4.1. Geliştirme Sürecinin Belgelenmesi

Yazılan kodların yanı sıra tasarımların tamamlanabilmesi amacıyla ilham kaynağı niteliğindeki kaynak ve dökümanlar, veritabanı bağlantısı için gerekli araştırmalar ve buna bağlı bilgilerin hepsi dokümantasyona dâhil edilmiştir. 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 yapılmıştır. Bunun yanı sıra projeye yeni dahil olabilecek personellerin olaya hakimiyeti açısından bu yönetime uygun bulunmuştur.

3.4.2. Eğitim Belgeleri

Sistemin kullanımı son derece basittir herhangi bir eğitim verilmeye ihtiyaç duyulmamaktadır. Hedeflenen kullanıcı modeli sebebiyle eğitime ihtiyaç duyulmayacağı öngörülmüştür. Bununla beraber risk faktörlerini göz ardı etmeden hareket edilmesi gerekeceği için form tasarımında arka planda nasıl yapılacağına dair gösterim yer alması planlanmaktadır.

3.4.3. Kullanıcı El Kitapları

Sistem web tabanlı olduğu için herhangi bir el kitapçığı hazır lanmayacaktır.

4. SİSTEM TASARIMI

4.1. Genel Tasarım Bilgileri

4.1.1. Genel Sistem Tanımı

4.1.1.1. Gereksinimler

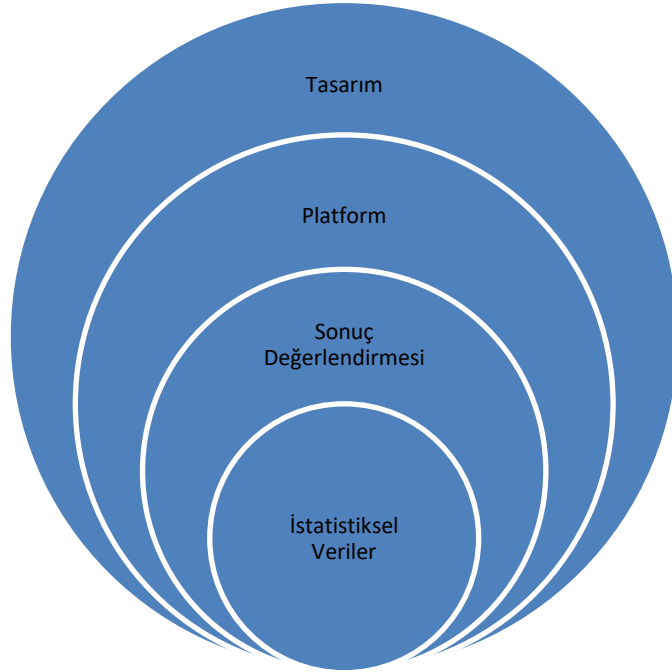
Sistem tasarımı gereksinimleri belirlemek amacıyla kullanıcılara sorma yöntemi ile istatistiksel sonuçlar elde edilecektir. Elde edilen bu istatistiksel sonuçlara göre gözden kaçırılmış olabilecek mantıksal hatalar, fikirler ve gereksinimler kullanıcılardan toplanacaktır.

4.1.1.2. İşlevsel Belirtiler (Mantıksal Model)

Bu kısımda ilk önce sistemin ne yapacağı sorusuna cevap vermemiz gerekmektedir. Sistem Türkiye Cumhuriyeti'nde artan eğitim/öğrenim oranı ile tez çalışmalarında da yüksek bir artış meydana getirmiştir. Bu doğal ilerlemenin sonucu olarak hazırlanan tez sayısında da artış meydana gelmiştir. Sadece yüksek lisans ve doktora öğrencileri baz alınmamıştır. Günümüzde online eğitim sistemine geçilmesi ve öğrencilerini kaliteli yetiştirmek isteyen öğretim üyelerinin de vermiş olduğu ödevler için tez hazırlaması istenmektedir. Bu artan ihtiyaç göz önüne alındığı zaman teknolojinin bu alanda olması kaçınılmaz hale gelmiştir. Bir tezin düzenli olup olmadığı her ne kadar göz ile kontrol edilebilse de günümüzün hızlı dünyasında zaman kavramının kısalığını göz ardı edemeyeceğimiz için bu işlemi manuel yapmak yerine öncelikli olarak teknolojiyi kullanmakta fayda olacağı aşîkardır. Zaman/Maliyet planlaması yapıldığında bile bunun elzem olduğu aşîkardır.

4.1.1.3. Tasarım

Tasarım aşamasında gereksinimler bölümünde belirttiğimiz istatistiksel veriler sonucunda tasarım bölümünü sürdüreceğiz. Grafiksel olarak açıklayacak olursak:



Şekil 4.1 Tasarım modellemesi

4.1.2. Varsayımlar ve Kısıtlamalar

Varsayılan değerler

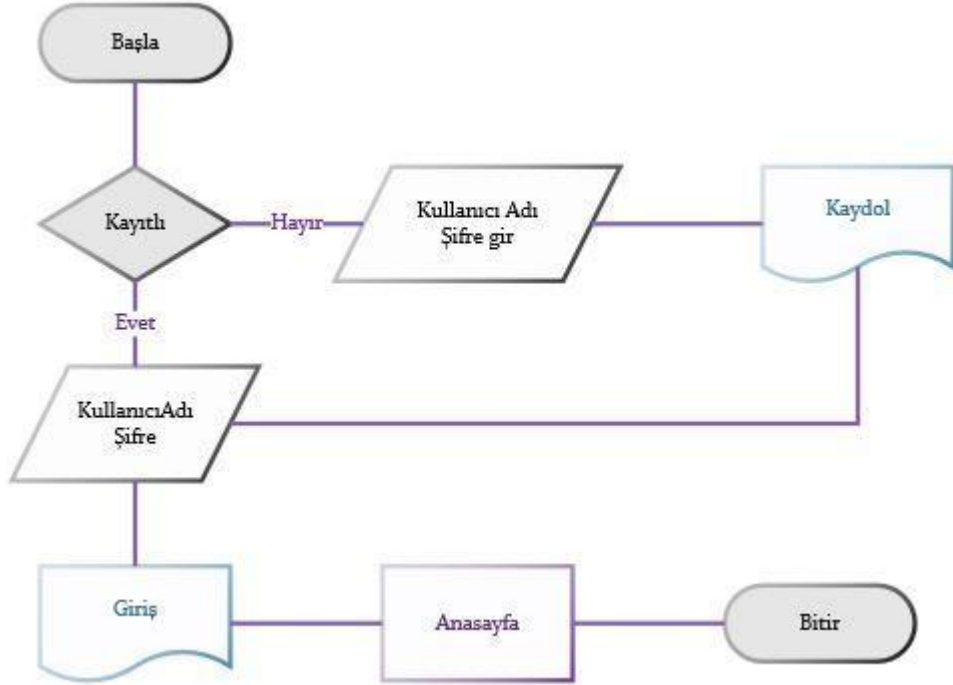
- Üniversite adının tez kapağında “Fırat Üniversitesi” şeklinde olması varsayılan değerdir.

Kısıtlamalar

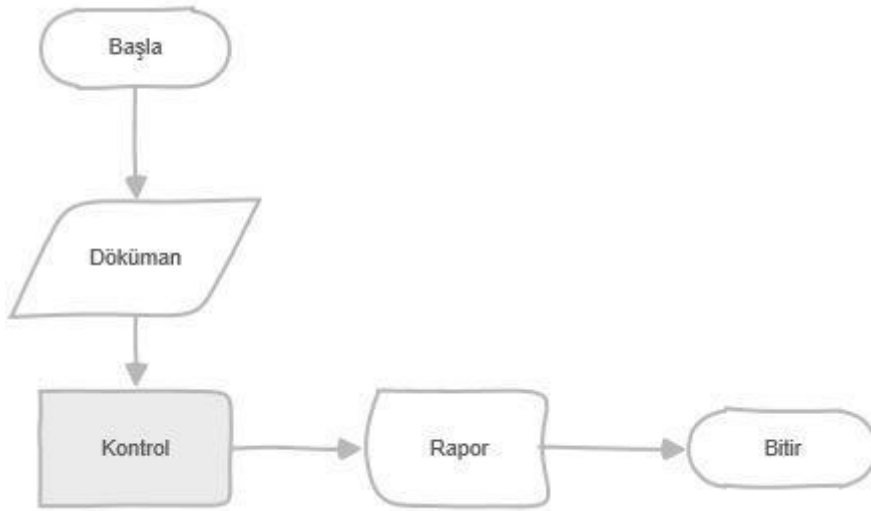
- Her üniversitenin tez hazırlama planı ve programı farklıdır bundan dolayı sadece Fırat Üniversitesine özel olarak hazırlanmıştır.

4.1.3. Sistem Mimarisi

Sistemin tam anlamıyla çalışma mantığı ve veri tabanı aşağıdaki diyagramlar da verilmiştir. Sistem giriş ve ana ekran formlarından oluşmaktadır. (Şekil 4.4)



Şekil 4.2 Giriş Yapısı



Şekil 4.3 Genel Sistem Mimarisi

Sistemin temel işleyiş yapısı ve veri tabanı hakkında gerekli açıklamalar yapılmış olup temel işleyiş hakkında da bilgi verilmiştir. Sistemde herhangi bir aksamaya karşı hata mesajları mevcut olacaktır.

4.1.4. Dış Arabirimler

4.1.4.1. Kullanıcı Arabirimleri

Sistemin en başında ana ekrana geçiş için kullanıcıdan şifre talep edilecektir. Kullanıcı bu şifreyi doğru bilmesi ile ana ekrana erişim sağlayacaktır. Ana ekranda sistem kullanıcısı tarafından yapılabilecek olan dosya yükleme, rapor indirme gibi işlemler programa kazandırılmıştır.

4.1.4.2. Veri Arabirimleri

Veriler DataBase yardımı ile yedeklenecektir. Bundan dolayı SQL Server Management kullanımı arabirim niteliğinde gerekli bir ihtiyaçtır.

4.1.4.3. Diğer Sistemlerle Arabirimler

Tümleşik olarak başka bir sistemle çalışmamaktadır.

4.1.5. Veri Modeli



Şekil 4.4 Veri modeli

4.1.6. Testler

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

Alfa aşaması

Başlangıç zamanından tasarım aşamasına gelene kadar yapılan testlerdir. Bu testlere karşılaşılan kod ve tasarımsal hatalar dâhil edilmekte olup çözüme kavuşturulmuştur.

Beta aşaması

Bakım aşamasında ve sistemin kurulup diğer süreçlerinin testleri bu aşamada yapılmakta olup test sonuçlarında çıkan sorunlar bertaraf edilmektedir.

4.1.7. Performans

Sistemin en iyi performansı verdiği yapı tespit edilerek onun üzerinden yapılacak tüm kurulumlar ve çalıştırmaların performanslarının belirlenen yapıya uygun olup olmadığı gözlemlenerek varsa eksik veya aksak yönler giderilecektir.

Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecektir. Sistemin Tasarıma Uygunluk Performansı kontrol edilecek olup tasarımı yapılan sistemin stabilitesi ve işleyiş performansı değerlendirilecektir.

4.2. Veri Tasarımı

4.2.1. Tablo tanımları

Sisteme kayıt edilecek olan kullanıcının bilgilerini tutmak üzere Giriş tablosu, döküman dosyasının tutulduğu Dosya tablosu, sistemin kural kontrolü sonucunda oluşturulan raporun tutulduğu Çıkış tablosu mevcuttur.

Giriş Tablosu	
Veri Adı	Veri Tipi
kID	Integer
kullaniciAdi	varChar(25)
sifre	varChar(25)

Tablo 4.1 Giriş Tablosu

Dosya Tablosu	
Veri Adı	Veri Tipi
dosyaID	İnteger
doc	varChar(MAX)

Tablo 4.2 Dosya Tablosu

Çıkış Tablosu	
docID	Veri Tipi
dosyaID	İnteger
rapor	varChar(MAX)

Tablo 4.3 Çıkış Tablosu

4.2.2. Tablo - İlişki Şemaları

4.2.3. Veri Tanımları

Bool veri tipi raporlamalar da işlemin yani hesaplanan metodun sonucunun değerini ifade etmesi amacıyla kullanılmıştır. Metinsel verileri saklamak amacıyla varChar tipi kullanılmıştır.

4.2.4. Değer Kümesi Tanımları

4.3. Süreç Tasarımı

4.3.1. Genel Tasarım

Genel sistem tasarımı öncelik kullanılacak olan ana ekranın şablon ve düzenine verilmiştir. Sonrasında bu şablona bağlı kalarak veri tabanı oluşturulmuş ve sistem kullanıma hazır hale getirilmiştir.

TEZ Kontrol Giriş Sayfası



HOŞ GELDİNİZ...

Lütfen Aşağıdaki Bilgileri Doldurunuz

Kullanıcı Adı

Parola

Üye değilseniz kayıt olmak için [tıklayınız](#)

Şekil 4.5 Giriş Ekranı

TEZ Kontrol Kayıt Sayfasına Hoş Geldiniz...



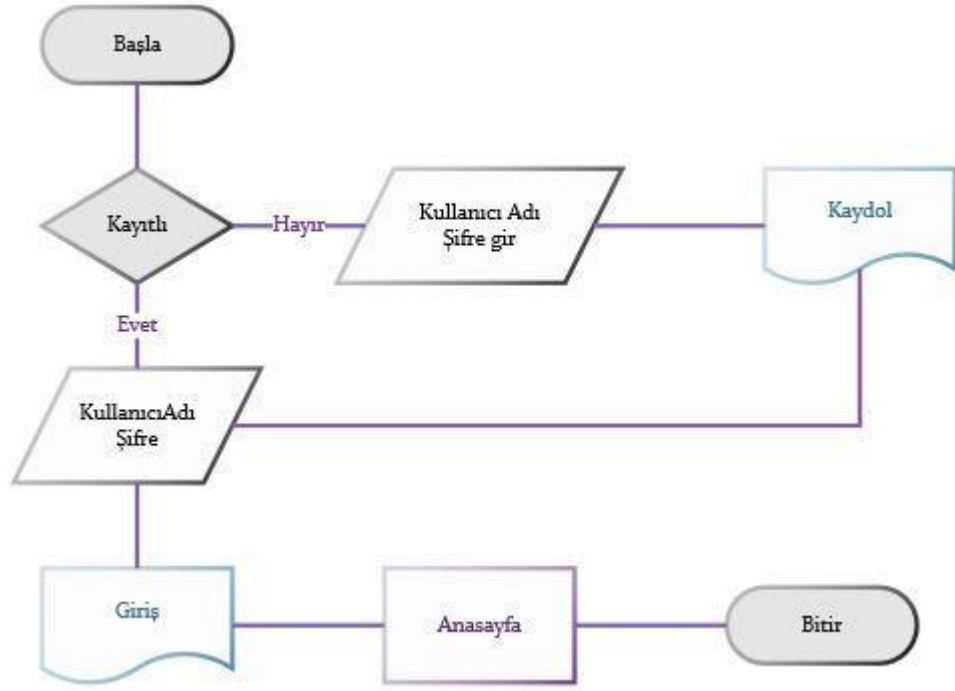
Lütfen Aşağıdaki Bilgileri Doldurunuz...

Kullanıcı Adı

Şifre

Şifre (Tekrar)

Şekil 4.6 Yeni Kayıt Ekranı



Şekil 4.7 Giriş tasarım/işleyiş

4.3.2. Modüller

4.3.2.1. Giriş Modülü

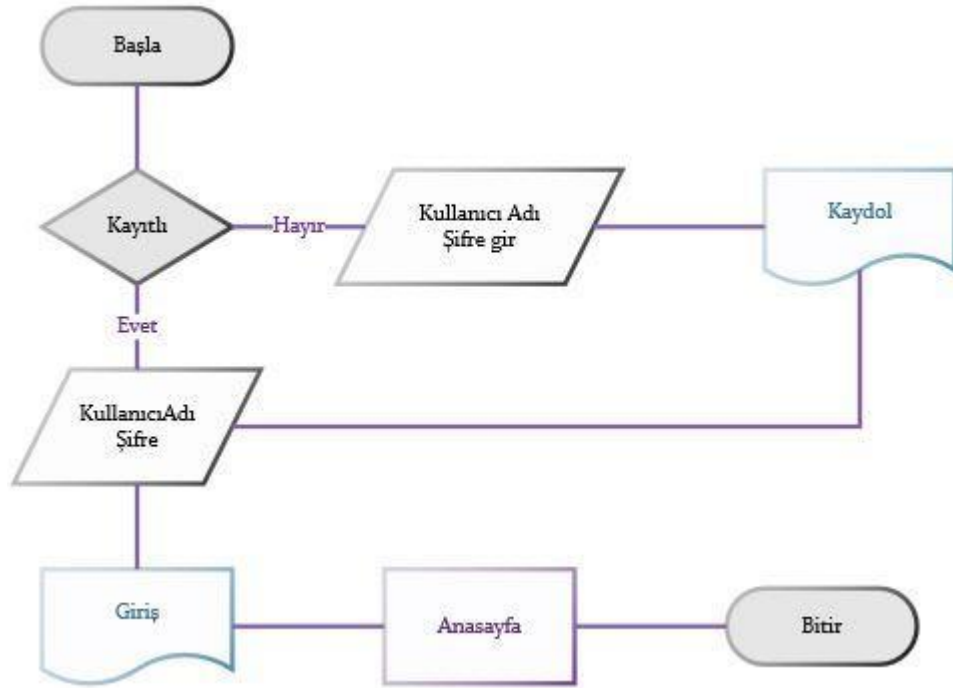
4.3.2.1.1. İşlev

Kullanıcının işlem yapabilmesi için ilk olarak kullanıcı girişini gerçekleştirmesi yada kayıt olması gerekmektedir. Giriş yaptıktan sonra ana sayfaya yönlendirilecek ve işlemleri gerçekleştirebilecektir.

4.3.2.1.2. Kullanıcı Arabirimi

Sisteme giriş izni verilen kişiyi belirtmektedir.

4.3.2.1.3. Modül iç Tasarımı



Şekil 4.8 Şifre işleyiş

Sistemde veri tabanı bağlantısını sağlayan bağlantı nesnesi ve aynı zamanda veritabanı için kullanıcıAdı/sifre vardır.

METOD ADI	TANIM
Şifre()	İlk girişte çalışandan alınıp doğruluğu kontrol eder.
Ekle()	Kullanıcının döküman eklemesini sağlar.
Çıkış()	Rapor bilgisini döndürür.
Kapat()	Programdan çıkışı sağlar.

Tablo 4.4 Metotlar

4.3.2.2. Yönetici Modülü

4.3.2.2.1. İşlev

Yapılan yükle, tara ve rapor kontrol olarak saydığımız işlemlerin kontrollerini sağladığından dolayı ayrı ayrı tekrar yazmaya ihtiyaç duyulmamaktadır.

4.3.3. Kullanıcı Profilleri

Kullanıcılar: Web ortamında bulunduğundan bütün kitleye hitab etmektedir.

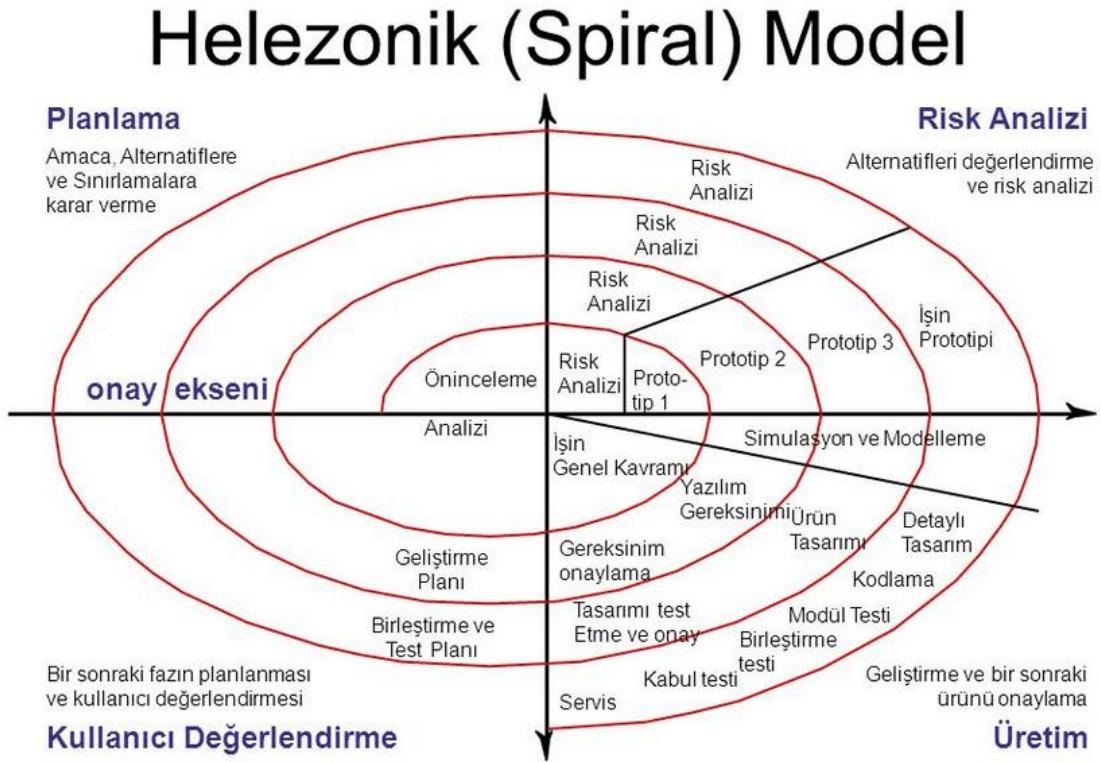
4.3.4. Entegrasyon ve Test Gereksinimleri

Sistem kendine has çalışma yapısı ile gerekli olan tüm işlemleri içinde barındırmaktadır. Test gereksinimleri için takım içerisinde bir teknisyene belirlenmiştir. Kurulum için herhangi bir platforma ihtiyaç yoktur sistem internet bağlantısı olduğu sürece çalışacaktır.

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1. Giriş

Gerçekleştirim çalışması için helezonik model kullanıldı. Böylece projeyi aşamalara bölerek ve her aşamada risk analizi yaparak oluşan hataları daha erken fark edilip giderilerek zaman ve maliyet açısından ciddi bir avantaj elde edildi. Aynı zamanda her döngüde risk analizi yapılarak oluşan veya oluşabilecek risklere karşı döngü sonunda planlama aşamasında bir çözümleme getirilerek yeni döngüye başlandı. Her döngü sonunda elimizde bir prototip olduğundan dolayı aşama aşama hataları giderilerek ilerleyerek en hatasız ürünü ortaya çıkarma fırsatı yakaladığımız gibi kullanıcıya da sistemi daha erken gösterebilme olanağı sağlanmış oldu.



Şekil 5.1 Sistem gerçekleştirim modeli

5.2. Yazılım Geliştirme Ortamları

Tasarımın sonunda ürettiğimiz fiziksel modelin, web de çalıştırılabilmesi için lazım olan Programlama Dili ve Veri Tabanı Yönetim Sistemi olarak iki bölümde incelendi.

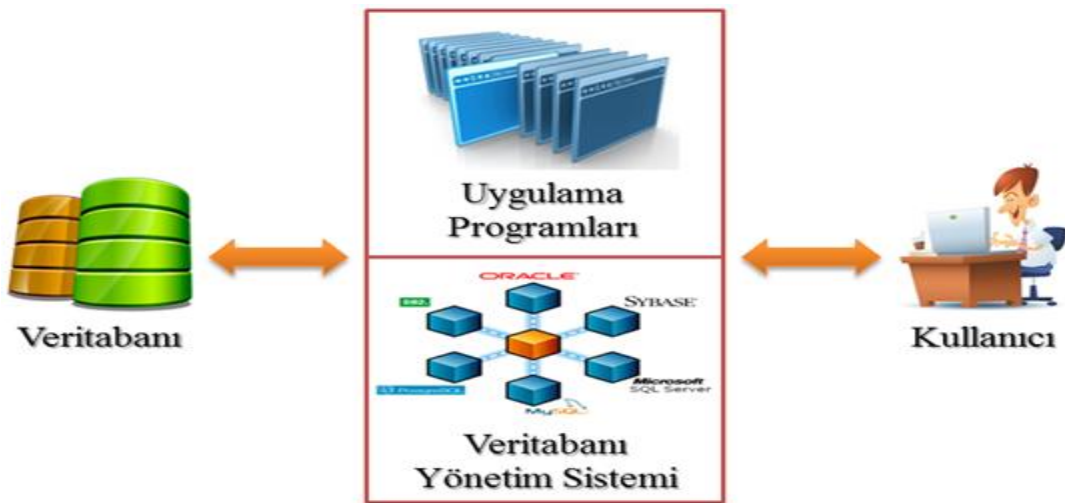
5.2.1. Programlama Dilleri

İşlemleri daha hızlı yapabilmek ve zengin kütüphanelerden yararlanarak tasarımıımızdan kodlarımıza kadar her işemi en iyi şekilde yapabilmeyi amaçladık. Erken bağdaştırma, tam zamanında derleme, doğal iyileme ve tamponlama hizmetleri gibi avantajlar sağladığından dolayı üstün bir performans sağlaması, zengin ve nitelikli araç desteği sağlaması, güçlü ve esnek bir yapıya sahip olması, basit ve yönetilebilir olması, mobil programla desteği sağlaması, çoklu dil desteği sağlaması gibi avantajlar sağladığı için ASP.NET kullanıldı. Sistemle tümleşik bir yapı oluşturmak amacıyla SQL Server Management kullanıldı.



5.2.2. Veri Tabanı Yönetim Sistemleri

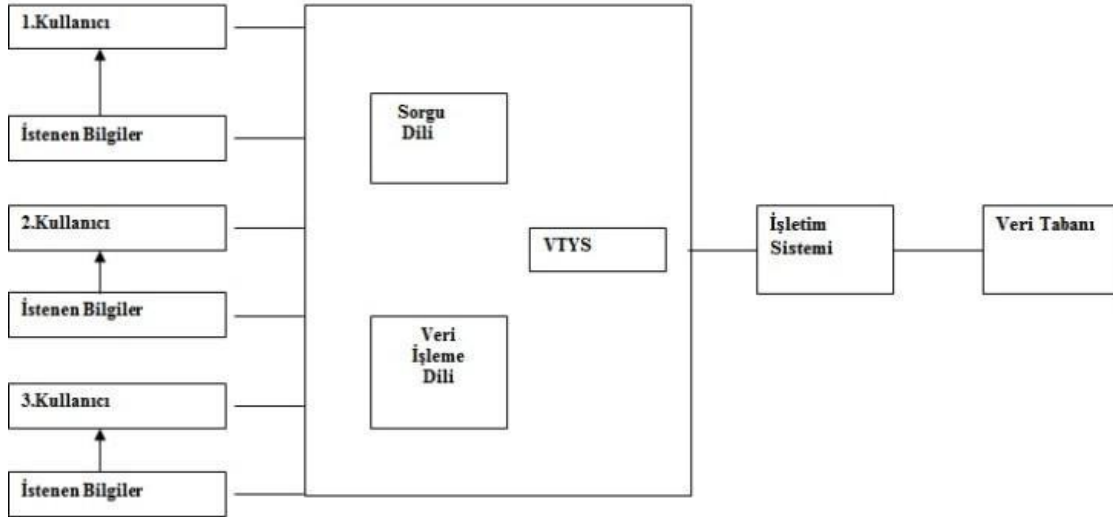
Web uygulamamızı ASP.NET ile yazıp SQL Server Management veritabanını kullanarak daha fonksiyonel bir kullanım sağlamayı amaçladık.



Şekil 5.2 Veri Tabanı Sistemi

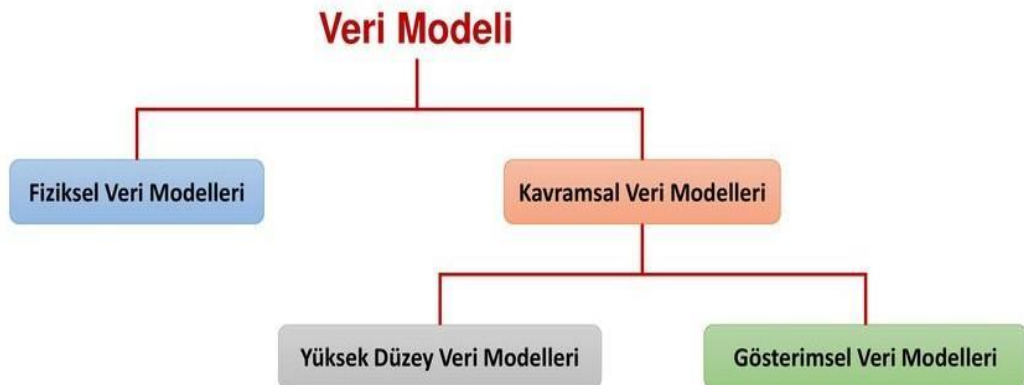
5.2.2.1. VTYS Kullanımının Ek Yararları

1. Genişleme potansiyeli,
2. Esneklik,
3. Uygulama geliştirme zamanından tasarruf,
4. Ölçümde ekonomi,
5. Ortak verilerin tekrarının önlenmesi,
6. Denetimin ve tutarlılığın sağlanması,
7. Verilerin kolay ve anlaşılır yapılarda sunulması,
8. Yazılımı geliştirmenin kolaylaşması,
9. Güncel bilgilerin tüm kullanıcılara aynı zamanda ulaşması



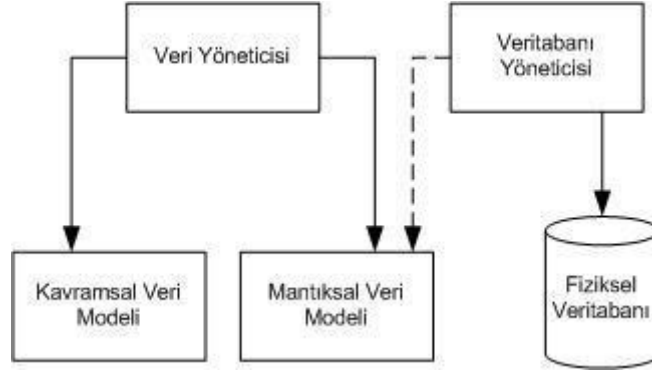
Şekil 5.3 VTYS nin Ek Yararları

5.2.2.2. Veri Modelleri



Şekil 5.4 Veri Modeli

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



Şekil 5.5 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 gösterilir.

3. Mantıksal Model

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

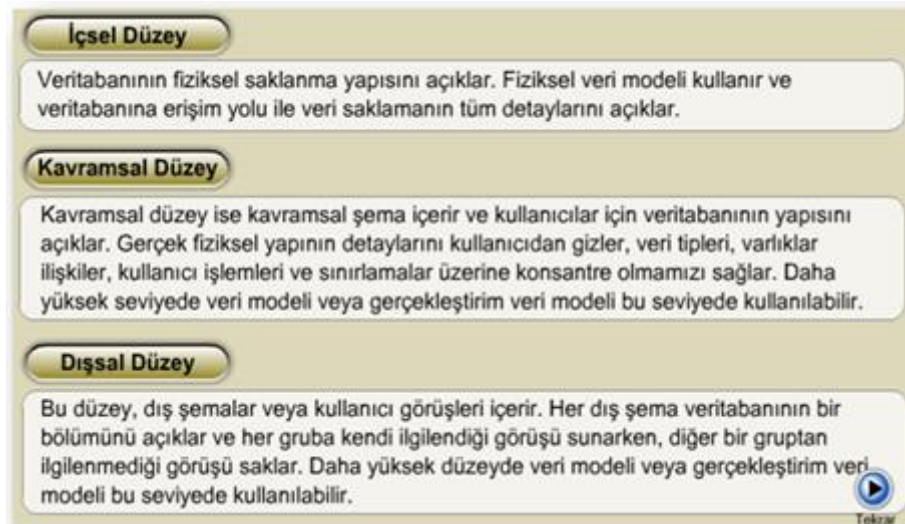
4. Fiziksel Model

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

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.

5.2.2.3. VTYS Mimarisi

Üç ana başlıkta inceleyelim.



Şekil 5.6 VTYS Mimarisi

Şekil 5.6 da görüldüğü gibi içsel düzey, kavramsal düzey ve dışsal düzey olarak incelenip tanımları verildi.

5.2.2.4. Veri Tabanı Dilleri ve Arabirimler

Veri tabanı için gerekli olunan tüm işlemleri SQL Server Management ile halledilerek sistem veritabanı çalışma yapısı sağlandı.

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 (VID)	Veri işleme dilidir. Veri tabanı oluşturulduktan sonra Veri tabanına veri eklemek, değiştirmek, silmek veya eklenmiş veriyi getirmek amacıyla kullanılır.

Şekil 5.7 Veri Tabanı Dilleri ve Arabirimler

5.2.2.5. Veritabanı Sistem Ortamı

Veri tabanında yükleme/raporlama gibi tüm işlemleri SQL Server Management kullanılarak tüm yedekleme ve görüntüleme işlem yapıları sağlanmış olmaktadır.

5.2.2.6. VTYS' nin Sınıflandırılması

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

5.2.2.7. Hazır Program Kütüphane Dosyaları

Uygulamamızda kullandığımız ASP.NET programlama dilindeki kütüphaneler yeterince kapsamlı ve zengin olduğundan dolayı hazır program kütüphane dosyalarına ihtiyaç duyulmadı.

5.2.2.8. CASE Araç ve Ortamları

Case işlemlerimiz Visual Studio ve Visual Studio Code sistemleri kullanılarak gerçekleştirildi. Aynı zamanda <http://madebyevan.com/> üzerinden Uml diyagramı çizimleri yapıldı.



Visual Studio Code

5.3. Kodlama Stili

Kod yapılarımız oluşturulurken hem bizim ne durumda olursa olsun veya ne zaman olursa olsun rahatlıkla analiz edebileceğimiz hem de dışarıdan herhangi bir uzman kişinin veya herhangi bir yazılımcının hiçbir problem veya aksaklık yaşamadan analiz edebileceği biçimde tasarlandı. Böylelikle kod karmaşasından, bundan kaynaklanabilecek yaşanması muhtemel problemlerin ve bunun sebep olacağı zaman kaybının önüne geçilmeye çalışıldı.

5.3.1. Açıklama Satırları

Açıklama satırlarını metotların başında kullandık ve metot içinde herhangi bir karmaşıklık gördüğümüzde veya sezdiğimizde kullandık. Böylelikle metotlarımızın daha anlaşılır ve çözümlenebilir olması sağlanmaya çalışıldı.

5.3.2. Kod Biçimlenmesi

Kodlarımız ASP.NET programlama dilinin kendine ait kodlama biçimiyle yazıldı. Bu süreçte kodlamanın başından sonuna kadar kodlarla ilgili herhangi bir biçimsel problemle karşılaşmadı.

5.3.3. Anlamlı İsimlendirme

Projenin genelinde standartlara bağlı kalınarak ikinci harfin küçük olması durumunda büyük harfle başlanarak ve daha anlaşılabilir olması amacıyla kullanılan metot, sınıf veya değişkenlerde kullanım amacına veya kullanım yerine göre adlandırma yapılmaya çalışıldı.

Benimsediğimiz bu adlandırma kurallarına projenin başından sonuna kadar istisnai durumlar haricinde uyulmaya çalışıldı.

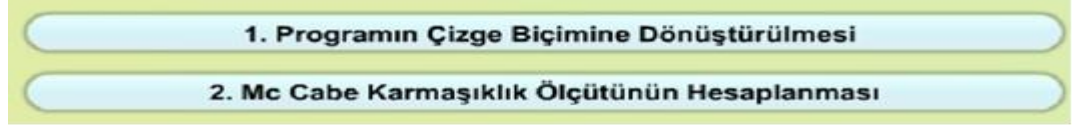
5.4. Program Karmaşıklığı

Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Lakin bu modellerin en eskisi ve diğer modellerin çoğuna yol gösterici olan McCabe karmaşıklık ölçüsü kullanıldı. Bu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir.

Bu ölçüt bir programda kullanılan ‘koşul’ deyimlerinin program karmaşıklığını etkileyen en önemli neden olduğu esasına dayanmaktadır.

İki aşamada uygulanır.

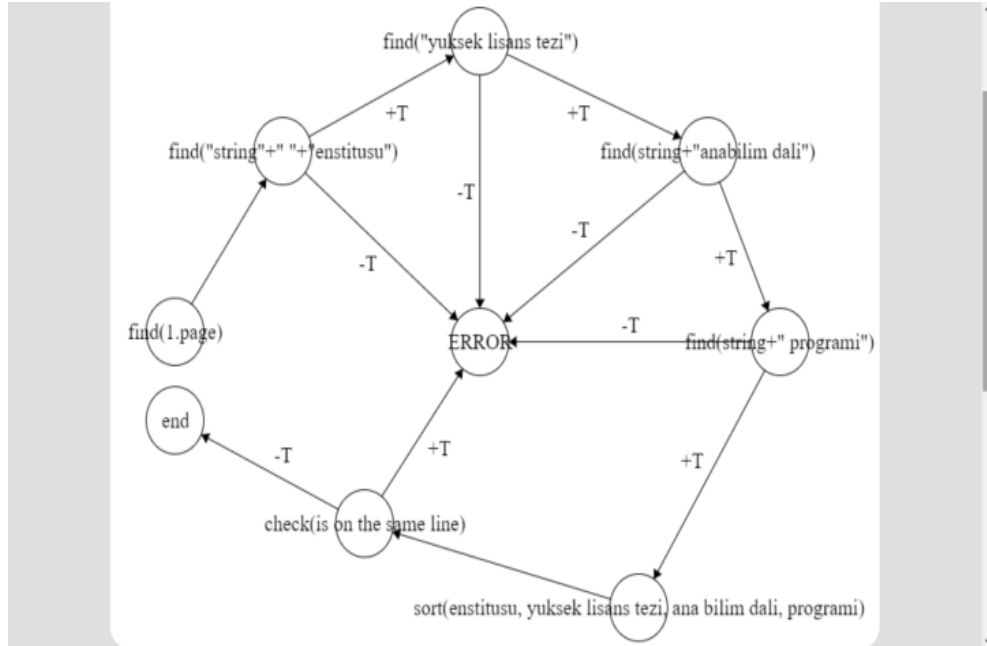
1. Programın çizgi biçimine dönüştürülmesi
2. McCabe karmaşıklık ölçütünün hesaplanması



Şekil 5.8 McCabe adımları

5.4.1. Programın Çizge Biçimine Dönüştürülmesi

İlk aşamada projenin çizgi biçimine dönüştürülmesi kısmında daha önceden her metot için ayrı hazırladığımız UML diyagramları esas alındı.



Şekil 5.9 Önceden hazırladığımız örnek bir diyagram

5.4.2. McCabe Karmaşıklık Ölçütü Hesaplama

M McCabe karmaşıklık ölçütü hesaplaması için daha önceden her metot için hazırladığımız diyagramlardan birini kullanalım.

k = 12 Kenar sayısı

d = 9 Düğüm sayısı

p = 2 Bileşen sayısı

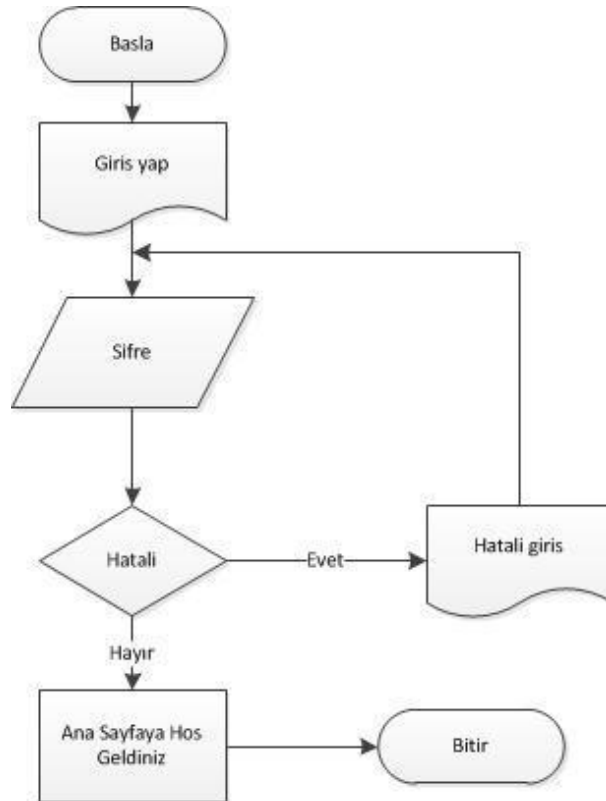
$V(G)=k-d+2*p$ (Karmaşıklık ölçütü formülü)

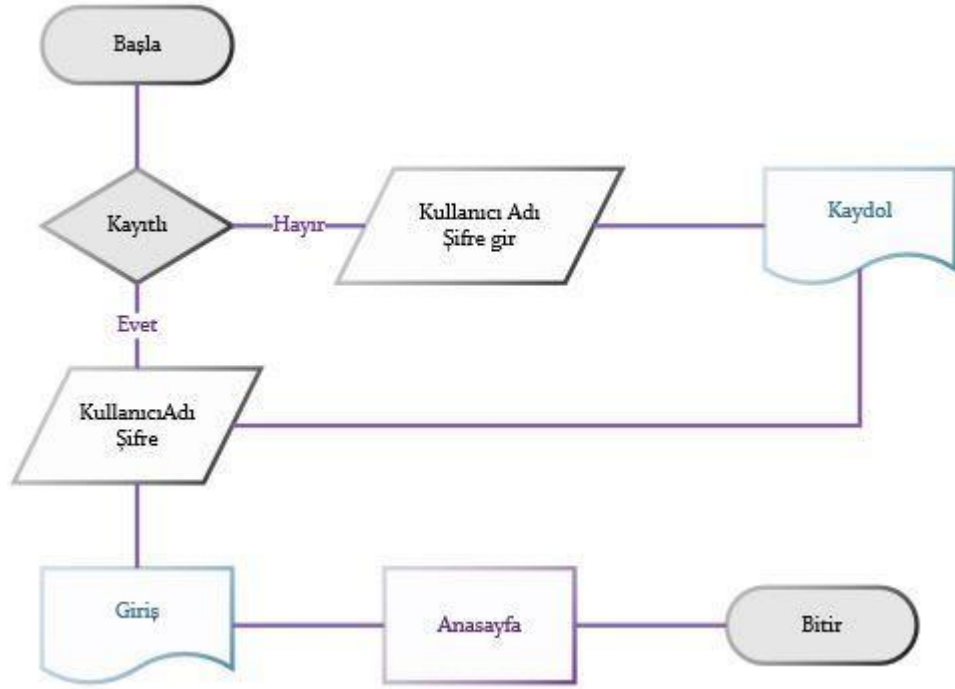
$$V(G)=k-d+2*p$$

$$V(G) = 12 - 9 + 2 * 2$$

$$= 7$$

SEI tarafından kabul edilen risk değerlendirme değerlerine göre ‘1-10’ arasındaki karmaşıklık ölçütüne yani fazla riskli olmayan basit bir modül olarak değerlendirmeye alınmıştır. Daha önceden hazırladığımız bütün diyagramlara karmaşıklık ölçüsü hesaplaması uyguladık. Çoğunluğunda 1-10 arasında değerler aldık. 10 dan fazla ölçüt aldığımız durumlarda karmaşıklığı azaltmak için diyagramlarda düzenleme yoluna başvurduk.





Şekil 5.10 Akış Diyagramları

5.5. Olağan Dışı Durum Çözümleme

Herhangi bir şekilde programın çalışmasının aksadığı, geçersiz veya yanlış veri oluşumu ya da başka sebeplerden dolayı programın sonlanması veya aksaması gibi durumlar olağan dışı durum olarak tanımlanmaktadır.

Bu sebeple kodlar oluşturulurken olağan dışı durumlar da dikkate alınarak bu durumda programın davranışlarına yönelik yöntemler geliştirilmeye çalışıldı.

5.5.1. Olağan Dışı Durum Tanımları

Ortaya çıkan olağan dışı durumlar karşısında TeamViewer yardımıyla ekrana ortaya çıkan problem hakkında bilgiler verilerek programın çalışmasına engel olacak bir problem değil ise çalışmaya devam etmesi veya programın çalışmasına engel olacak bir problem ise sayfanın otomatik olarak yeniden başlatılması seçenekleri değerlendirmeye alındı.

5.5.2. Farklı Olağandışı Durum Çözümleme Yaklaşımları

Yukarıda da belirtildiği gibi programın sonlanması veya sayfanın yeniden başlatılması seçenekleri değerlendirilecektir.

Şayet problem bütünüyle sistemin hepsini etkileyecek düzeyde ise programın çalışma yapısı bozulmayacak şekilde yerinde destek veya teknik destek seçenekleri de değerlendirmeye alınacaktır.

5.6. Kod Gözden Geçirme

Amacımız yaptığımız programa başlamadan önce mevcut sistemleri incelerken kod yapısının nasıl olduğu, nerelerde hata veya aksaklık olduğunu tespit ederek oluşturduğumuz sistemi en iyi şekilde oluşturmaktır. Bu sebeple kod gözden geçirme kısmında mevcut sistemlerin kod yapısı ve yeni eklenecek özelliklerin kod yapısı incelendi.

5.6.1. Gözden Geçirme Sürecinin Düzenlenmesi

Gözden geçirme sürecinin temel özellikleri;

- Hataların bulunmasını ancak düzeltilmemesini amaçladık.
- Olabildiğince az kişi ile yapmaya çalıştık. Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımını yapmasını planladığımız kişilerden oluşmasına dikkat ettik.
- Kalite çalışmalarının bir parçası olarak ele aldık ve sonuçları düzenli bir biçimde saklamaya çalıştık. Bu kısımda cevabını aradığımız temel soru programın yazıldığı gibi çalışıp çalışmadığını belirlemek oldu.

5.6.2. Gözden Geçirme Sırasında Kullanılacak Sorular

Programımızı incelerken, programın her bir işlevi için aşağıdaki sorulara cevap vermeye çalıştık.

5.6.2.1. Öbek Arayüzü

Sorular ve cevaplar aşağıda mevcuttur.

1. Her öbek tek bir işlevsel amacı yerine getiriyor mu?
 - Anlam karmaşıklığını azaltmak amacıyla her öbeğe tek işlev yaptırma amaçlanmıştır.
2. Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?

- Yukarıdaki soruda da belirtildiği gibi anlam karmaşıklığını giderebilmek amaçlı isimlendirmeler özenle seçilmiştir.
3. Öbek tek giriş ve tek çıkış mı?
 - Evet, öbek tek giriş ve çıkışlıdır.
 4. Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?
 - Evet, değiştirebilme özelliği mevcuttur.

5.6.2.2. Giriş Açıklamaları

Sorular ve cevaplar aşağıda mevcuttur.

1. Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?
 - Evet, açıklama satırları mevcuttur.
2. Giriş açıklama satırları, öbeğin amacını açıklıyor mu?
 - Tanımlamalara özen göstererek açıklamalar eklenmiştir.
3. Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?
 - Sistemimizde kullanılan tüm parametreler ve işlemler tanıtılmıştır.
4. Giriş açıklama satırları, çıktıları (parametre, kütük vb) ve hata iletilerini tanımlıyor mu?
 - Evet, tanımlamalar mevcuttur.
5. Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?
 - Kurallara uymaya çalışılıp açıklamalar öyle yapılmıştır.
6. Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?
 - Evet, içte yapılan işlemlerde açıklanmıştır.
7. Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?
 - Olağandışı durumlar için açıklama satırları mevcuttur.
8. Giriş açıklama satırları, Öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?
 - Evet, içermektedir.
9. Her paragrafı açıklayan kısa açıklamalar var mı?
 - Evet, açıklamalar her paragraf başında bulunmaktadır.

5.6.2.3. Veri Kullanımı

Sorular ve cevaplar aşağıda mevcuttur.

1. İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?
 - Birbiriyle ilişkili veriler gruplanmıştır.
2. Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı?
 - Anlamlı isimlendirmeler kullanılmıştır.
3. Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?
 - Değişkenler arasında kullanım uzaklığı yoktur.
4. Her değişken tek bir amaçla mı kullanılıyor?
 - Her değişken tek bir işlev ile görevlendirilmiştir ki karmaşıklık önlenesin.
5. Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?
 - Kimi yerlerde dizin dışı kullanımda gerektiği için genel tanımlamalarda mevcuttur.
6. Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?
 - Her gösterge için yer mevcuttur.

5.6.2.4. Öbeğin Düzenlenişi

Sorular ve cevaplar aşağıda mevcuttur.

1. Modüller birleşimi uyumlu mu?
 - Her birleşimin amacı olduğundan açıklamalar mevcuttur.
2. Modüller arası veri aktarımları sağlanıyor mu?
 - İletişim mevcuttur.
3. Bütün modüller birleştiğinde sistem çalışıyor mu?
 - Sistem aktif ve kullanılabilir vaziyettedir.

5.6.2.5. Sunuş

Sorular ve cevaplar aşağıda mevcuttur.

1. Her satır, en fazla bir deyim içeriyor mu?
 - Evet, anlam karmaşıklığını önlemek hedefidir.

2. Bir deyimnin birden fazla satıra taşması durumunda, bölünme anlaşıla bilirliği kolaylaştıracak biçimde anlamlı mı?
 - Evet, mesela ekleme/silme/güncelleme gibi veri tabanı işlemleri + sembolünden ayrılmıştır.
3. Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?
 - Evet, yalınlık esastır.
4. Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?
 - Evet, karmaşıklık bu nedenle az çıkmıştır.
5. Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?
 - İlk olarak belirlediğimiz stil sürekli kullanılmıştır.
6. Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?
 - Sistemin if else bloklarında buna ihtiyaç duyulup kullanılmıştır.

6. DOĞRULAMA VE GEÇERLEME

6.1. Giriş

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

- Yazılım belirtilmelerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtilmeleri 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 belirtilmelerin önceki belirtilmelerle karşılaştırılması.

Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçekleşmesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.



Şekil 6.1 Doğrulama ve gerçekleştirme(geçerleme)

6.2. Sınama Kavramları



Şekil 6.2 Sınama Kavramları

Birim Sınama:

Bu kısımda sistemin sınanması için okul sınıf arkadaşlarımıza, bölüm fark etmeksizin okuldaki arkadaşlarımıza, aktif olarak çalışan yazılımcı ve yazılım mühendisi arkadaşlarımıza gönderildi ve onların sistemi sınaması istendi.

Aynı zamanda ekiptekiler olarak her birimiz ayrı ayrı sistemimizi sınadık.

Alt Sistem Sınama:

Birimlerimiz birleştirilip modüllerimiz oluşturulduktan sonra bunların kendi içinde sınanması sağlandı. Çoğunlukla sistemimizin arayüzünde görülen eksiklikler ve hatalar giderildi. Sistemimizin her aşamasında sistem sılandıktan sonra raporumuz hazırlanıp sistemimizin bütün aşamaları için sınama ve sınama onayı yapıldı.

Sistem sınaması:

Birim sınaması aşamasında hazırlanan rapor ve alt sistem sınamasında hazırlanan rapor bir araya getirilerek genel bir rapor hazırlandı. Bu raporda ekibimizin tek tek onayı alınıp, birim sınamasında sistemimizi gönderdiğimiz sınıf arkadaşlarımızın, diğer öğrenci arkadaşlarımızın, aktif olarak çalışan yazılımcı ve yazılım mühendisi arkadaşlarımızın onayı alındı.

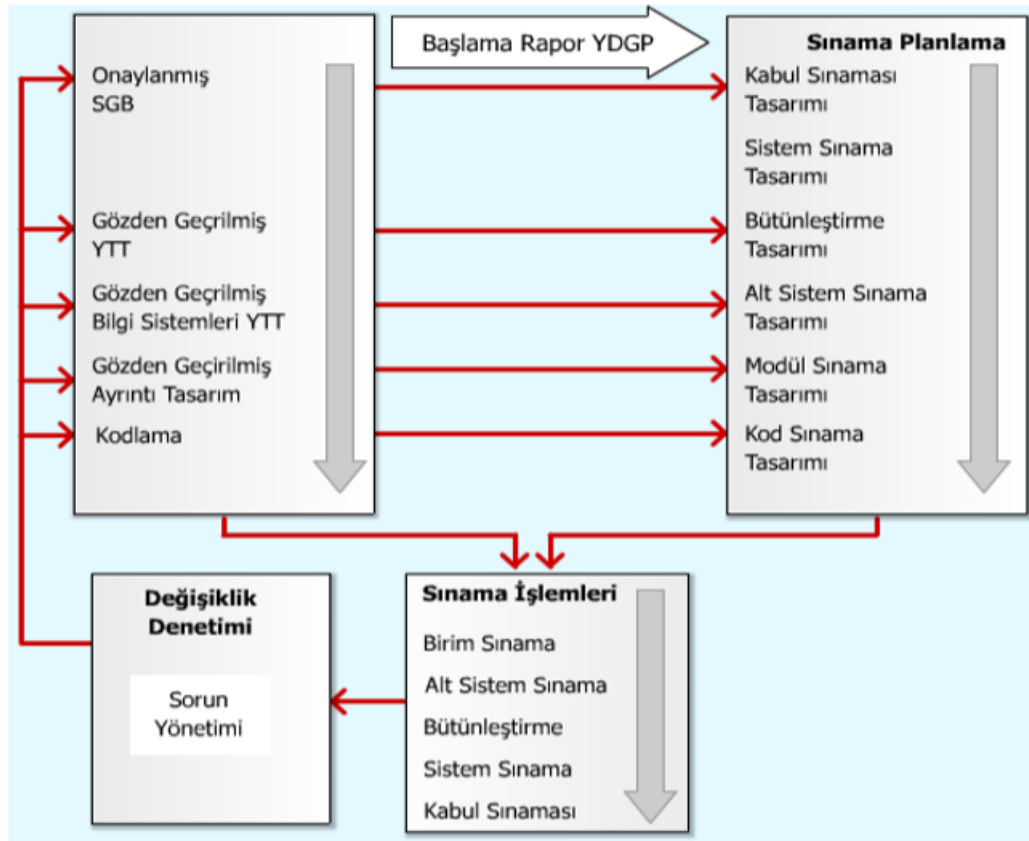
Bu aşamada hazırlanan rapor sonunda sistemde herhangi bir problem veya eksiklik tespit edilmemiştir.

Kabul sınaması:

Sistem sınaması aşamasında sonuçlanan rapor nihayetinde sistemimizde herhangi bir aksaklığa veya hataya rastlanmadı.

Sonuç itibari ile sistemimiz mükemmel bir biçimde çalıştı.

6.3. Doğrulama ve Geçerleme Yaşam Döngüsü



Şekil 6.3 Doğrulama/Geçerleme yaşam döngüsü

6.4. Sınama Yöntemleri

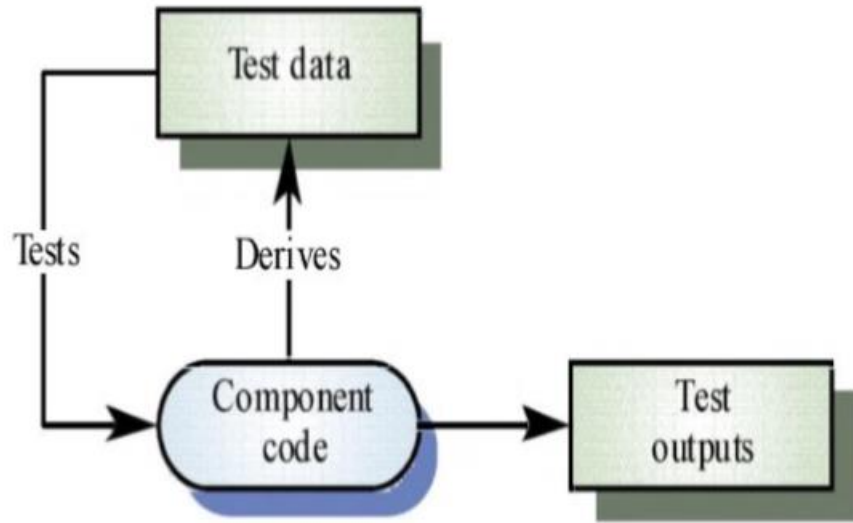
Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir "sonra" operasyonu olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çaba türüdür.

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 denemesi yapıldı
- Bütün işlevsel noktalar sınandı
- Ödeme sistemi entegrasi sınandı
- Arayüzler ve girişler sınandı



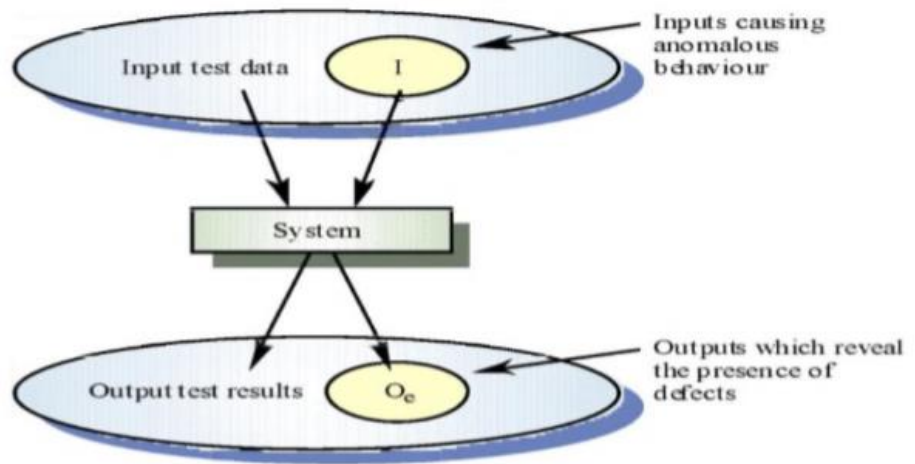
Şekil 6.4 Beyaz kutu sınaması

6.4.2. Temel Yollar Sınama

Temel Yollar Sınaması

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

- Eş değerlere bölme
- Uç değerler analizi
- Karar Tablosu
- Sonlu durum makinesi
- Belgelenmiş özelliklere göre test
- Rastgele test
- Kullanım profili



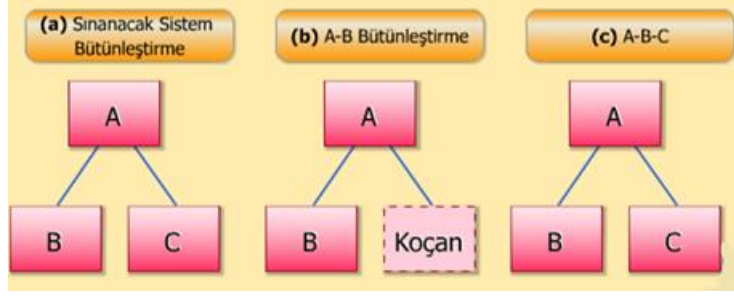
Şekil 6.5 Sınama yapısı

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 bağlıdır. 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

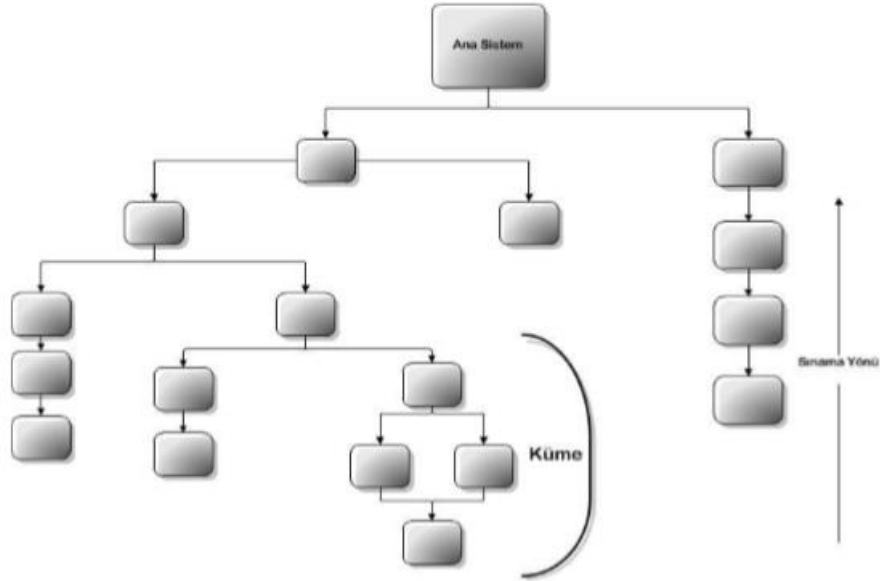
Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur. En üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınıandıktan sonra alt düzeye geçilmelidir. Ancak bu en üstteki bileşenin tam olarak sınanması için alttaki bileşenlerle olan bağlantılarının da çalışması gerekir.



Şekil 6.6 Yukarıdan aşağı

6.5.2. Aşağıdan Yukarıya Sınama ve Bütünlendirme

Aşağıdan yukarı bütünlendirmede 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. Projede bu sistemin kullanılmıştır. Önce alt sistem sonra üst sistemler sınanmıştır.



Şekil 6.7 Aşağıdan yukarı

6.6. Sınama Planlaması

Aşağıda tanımlar belirtilmiştir.

Test planı kimliği: Test planının adı veya belge numarası

Giriş: Test edilecek yazılımın elemanlarının genel tanıtım özetleri. Ayrıca bu plan kapsamı ve başvuru belgeleri. 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 Belirtileri

İşleyiş yapısı ve bilgiler aşağıda mevcuttur.

- ☒ Sınanan program modülü ya da modüllerinin adları
- ☒ Sınama türü, stratejisi (beyaz kutu, temel yollar vb.)
- ☒ Sınama verileri
- ☒ Sınama senaryoları

gibi bilgiler mevcuttur.

Sınama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir.

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

Sınama işlemi sonrası bu belirtilme

- ❖ Sınamayı yapan
- ❖ Sınama tarihi
- ❖ Bulunan hatalar ve açıklamalar

Türündeki bilgiler eklenerek sınama raporu oluşturulur.

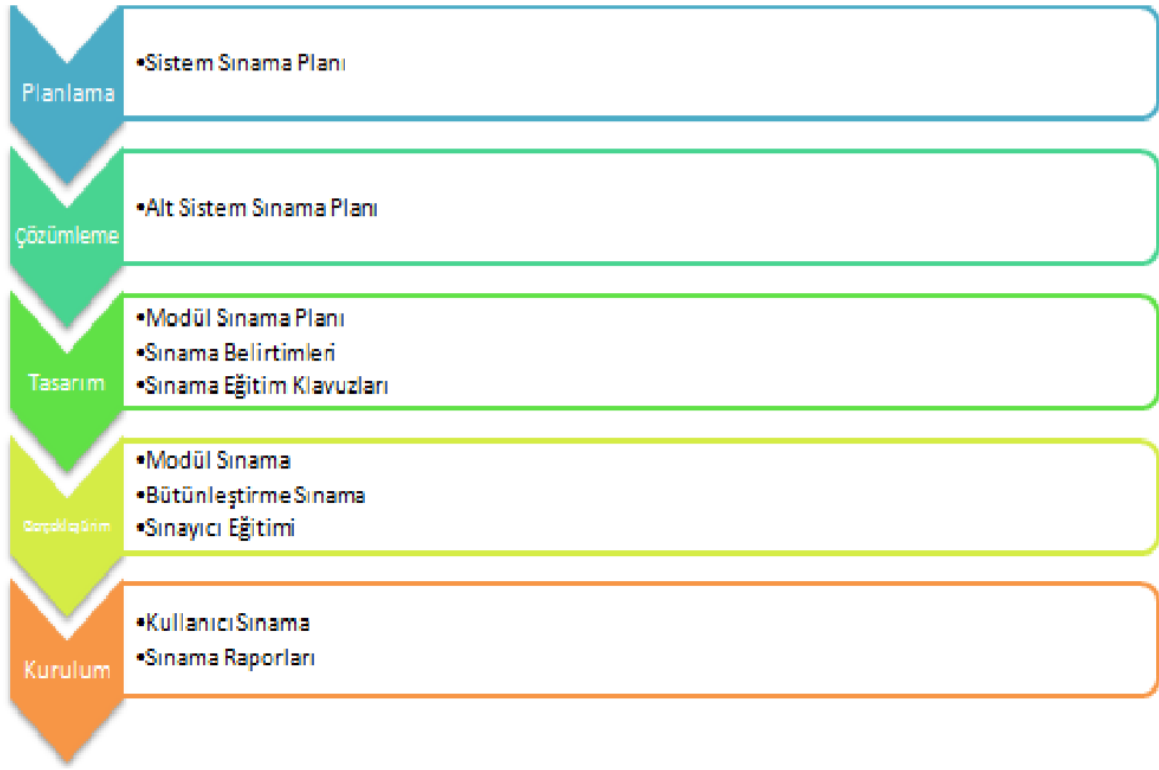
Sınama raporları, sınama bitiminde imzalanır ve yüklenici ile iş sahibi arasında resmi belge niteliği oluşturur.

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 ayrıntılanır. Tasarım aşamasında sınama plana detaylandırılır ve sınama belirtilmeleri oluşturulur. Bu oluşumlar daha sonra eğitim ve el kitabında kullanılır.

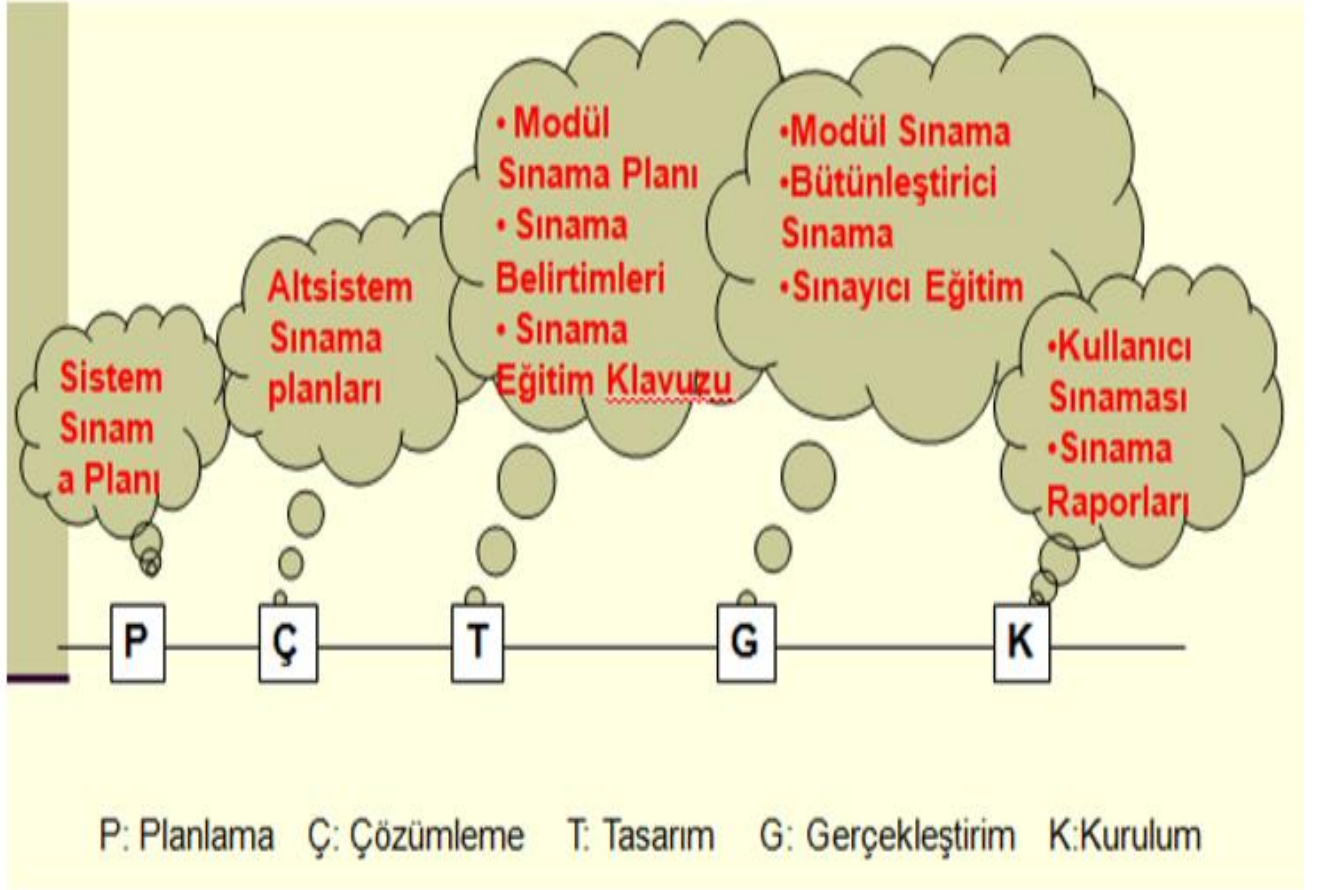
Gerçekleştirim aşamasında teknik sınamalar yapılır sınama raporları hazırlanır ve elle tutulur ilk testler yapılır. Kurulum aşamasında sistemle ilgili son sınamalar yapılır ve sınama raporları hazırlanır.







Şekil 6.8 Yaşam Döngüsü Sinama Etkinlikleri

Sinama 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.



Gerçekleştirim Ve Doğrulama-Zaman Diyagramı			
Zaman	7. Hafta	8. Hafta	9. Hafta
Gerçekleştirme-Doğrulama			
Gerçekleştirme			
Doğrulama			

Şekil 6.9 Yaşam Döngüsü ve sınamalar

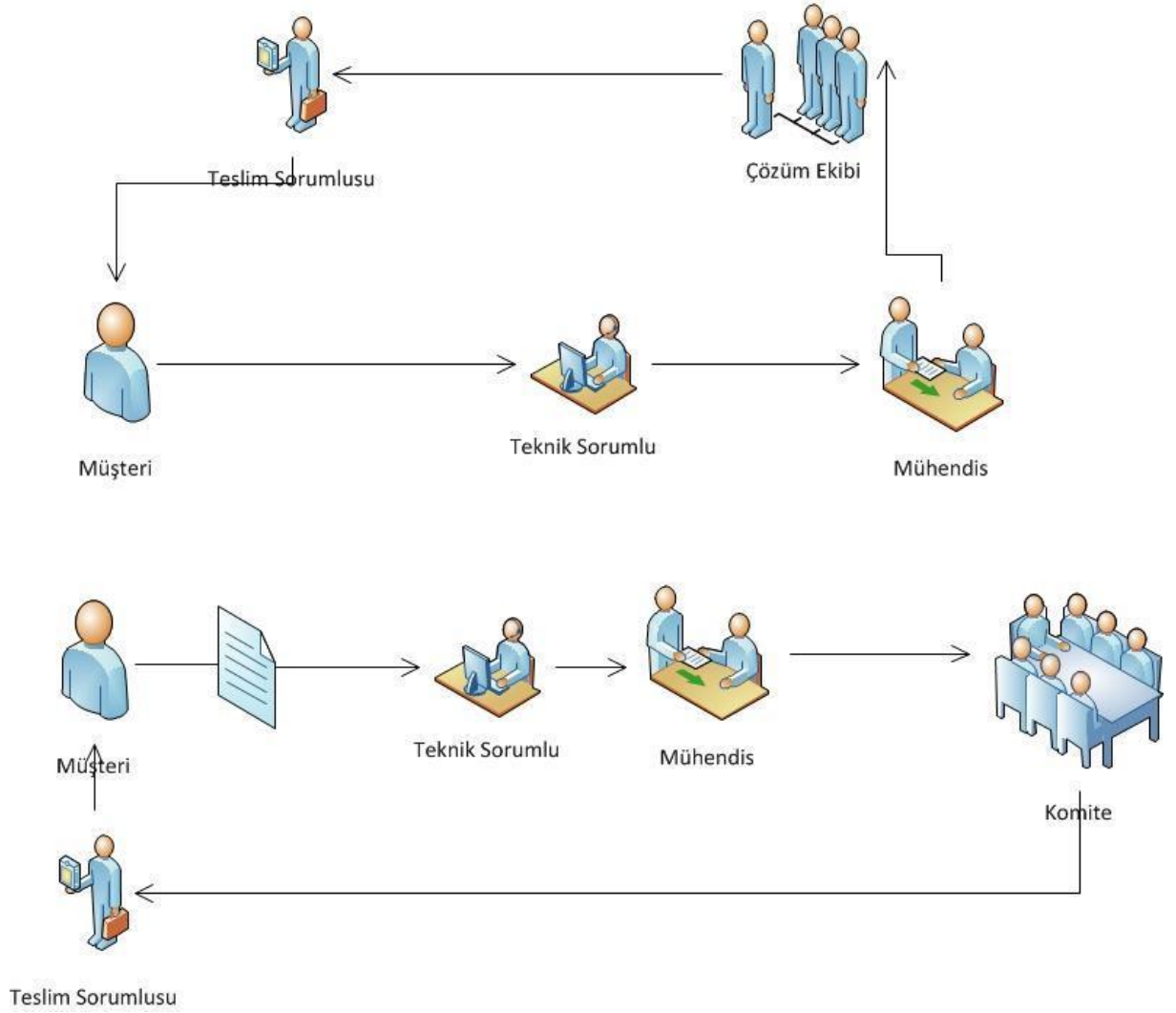
7. BAKIM

7.1. Giriş

Bakım aşamasının aslında hiç bitmeyen bir aşama olduğu bilinmektedir. Çünkü herhangi bir problem ile karşılaştığımızda problemin giderileceğinden ve buna ek olarak rutin kontrolleri yapacağımızdan bu aşamanın daimi bir şekilde devam edeceği görülmektedir.

Ortada sistemle ilgili bir problem yoksa ve müşterinin herhangi bir isteği varsa müşteri sistemde nasıl bir değişiklik istediğini ekibimizce belirlediğimiz teknik sorumlu görevlisine iletacaktır. Teknik sorumlu görevlisi bu isteği bize bildirecektir. Bu istek doğrultusunda nelerin değişeceği ve kimlerin bu durumla ilgileneceği belirlenip ilgilenecek arkadaşlarımızdan oluşturduğumuz ekibe bu düzenlemenin sağlanması için görev verilecektir. Belirlediğimiz bu ekip tarafından, istekler doğrultusunda değişiklikler yapılarak ve sistemin tüm sınamaları yapılarak programın kullanmaya hazır bir hale getirilip müşteriye teslim edilmek üzere teslim sorumlusu olarak belirlediğimiz arkadaşımıza verilecektir.

Eğer ortada sistemle ilgili bir hata varsa müşteri tarafından bu hata rapor haline getirilip belirlediğimiz teknik sorumlu görevlisine teslim edilecektir. Teknik sorumlu görevlisi bu durumu çözüme kavuşturamadığı durumda rapor bize yönlendirilecektir. Tarafımızca bu hatanın nereden kaynaklandığı tespit edilip bu durumu çözebilecek teknik ekip oluşturulur. Bu ekip hatayı çözüp gerekli tüm sınamaları yaparak programın hatasız bir şekilde çalışmasını sağlayacaktır. Durum çözümlendikten sonra tarafımızca onaylanır ise belirlediğimiz teslim sorumlusu görevlisine programı müşteriye teslim etmek üzere aktarılacaktır.



Şekil 7.1 Bakım

7.2. Kurulum

Kurulum işleminin gerçekleştirilmesi için iki adet programa ihtiyaç vardır. Bu programlardan birincisi Microsoft SQL Server bir diğeri ise Microsoft Visual Studio'dur.

- İlk aşama olarak GitHub platformunda mevcut olan .bacpac uzantılı veri tabanını SQL'e import edilecektir.
- Visual Studio yardımıyla proje başlatılacaktır.
- Sisteme login işlemi (Kullanıcı eğer kayıtlı değilse kayıt sayfasına yönlendirilerek sisteme kaydı sağlanacaktır) yapıldıktan sonra sistem kullanıma hazır hale gelecektir.
- Anasayfaya erişim işleminin ardından fileupload nesnesi yardımı ile sisteme dosya yüklenir analiz et butonuna tıklanılarak kuralların kontrolü sağlanır.

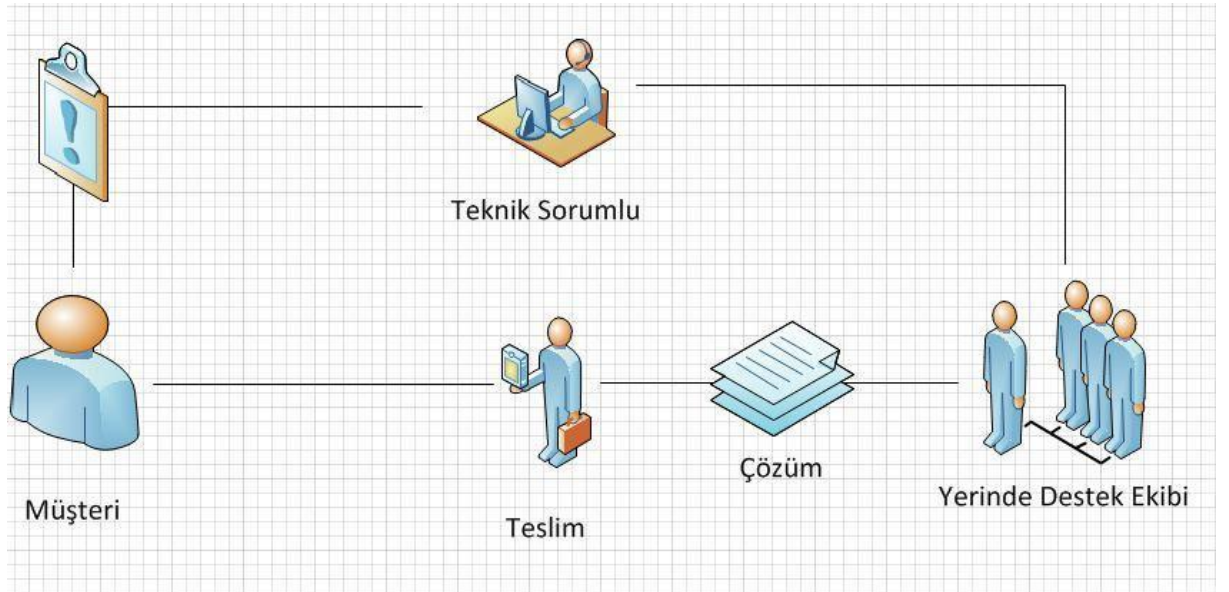
Yukarıdaki kurulum aşamaları sistemsel gereksinimleri (Windows7 üstü) karşılayan bir bilgisayara kurulduktan sonra sınanacak ve doğruluğu tespit edilecektir. Herhangi bir problem olmadığı anlaşıldıktan sonra dağıtım işlemi tüm hızıyla devam edecektir.

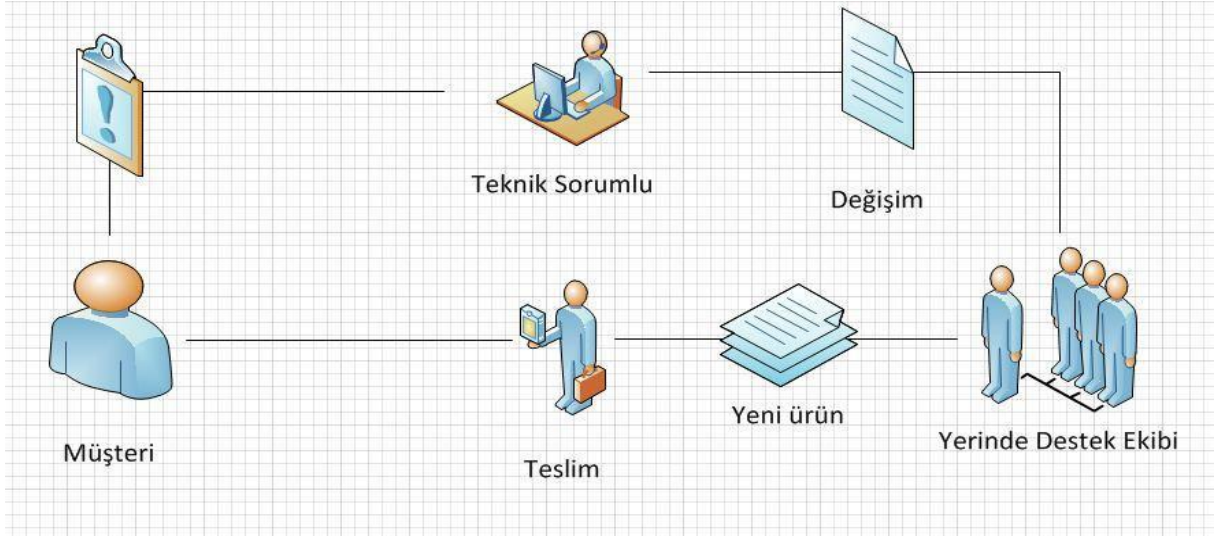
7.3. Yerde Destek Organizasyonu

Bizim belirlediğimiz teknik sorumlu arkadaşımız herhangi bir probleme karşı her zaman hazır olacaktır. Müşteri teknik sorumluya bir problem raporu gönderdiğinde veya bir istekte bulunduğunda kendisi gerek telefon görüşmesi yoluyla veya TeamViewer kullanarak durumu çözebilir ise müşteriyi bekletmeden ve aksaklık yaşayıp zaman kaybetmesine neden olmadan müşteri memnuniyetini sağlamış olacağız.

Şayet sistemimizin kullanım alanı artarsa veya müşteri sayılarında artış gözlemlersek teknik sorumlu olarak belirlediğimiz arkadaşlarımızın sayısında arttırarak müşteri memnuniyetini sağlamak hedefimiz olacaktır.

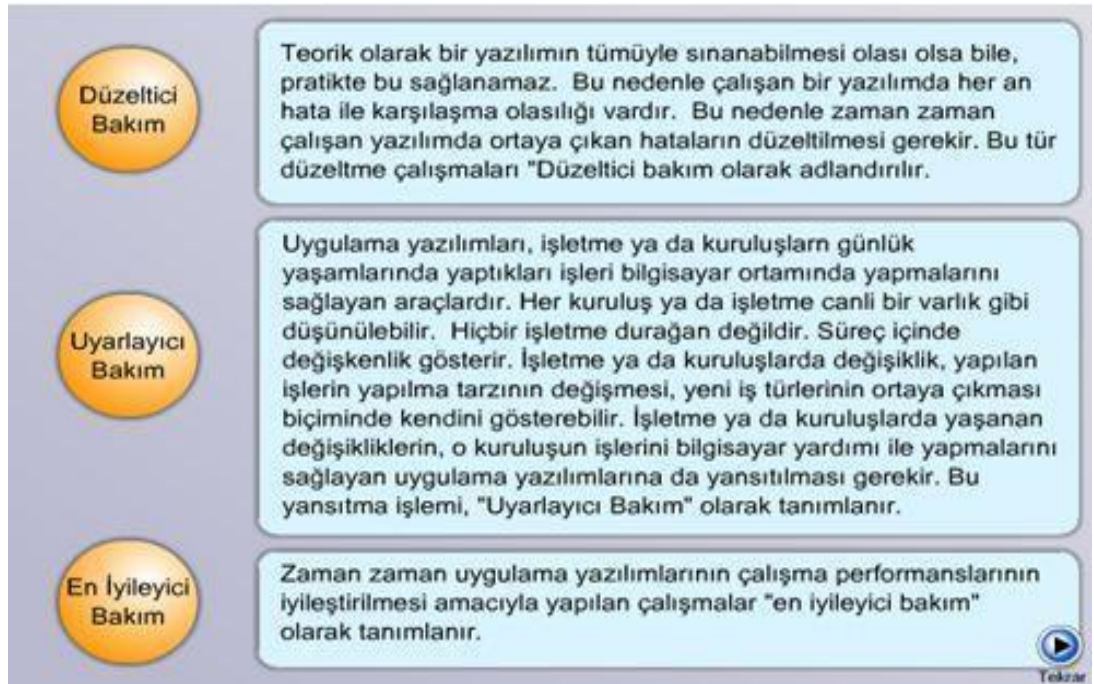
7.4. Yazılım Planı





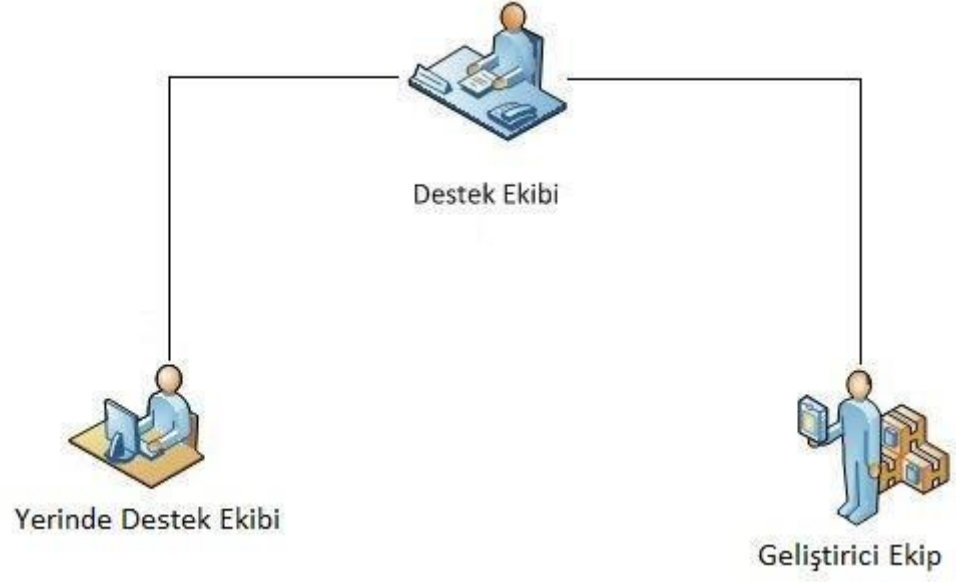
Şekil 7.2 Yazılım Planları

7.4.1. Tanım



Şekil 7.3 Bakım Çeşitleri

7.4.2. Bakım Süreç Modeli



Şekil 7.4 Bakım Süreç Modeli

1. Sorun Tanımlama Süreci

Bu süreçte hedefimiz sorunun ne olduğunu belirlemektir. Bu kısımda bakım isteğinin sebebinin yeni bir işlev veya özellik eklenmesi yönünde ya da sistemin çalışmasına etki eden bir problemin çözülmesi yönünde mi olduğunu tespit ederiz. Bu istekler arasında performans artırımına yönelik istekler de eklenebilir.

Bu süreçte bakım isteği için yeni bir dosya hazırlayıp numaralandırırız. Yapılacak değişiklikleri türüne göre sınıflandırırız. İsteği onaylama, reddetme veya ayrıntılı incelemeye alınması yönünde karar alırız. Zaman kestirimi yapıp değişiklikler birden fazla ise önceliklendirmesini yaparız.

Problemin ne olduğunu tespit edildikten ve nasıl bir yol izleneceği konusunda karar alındıktan sonra çözümleme sürecine geçeriz.



Şekil 7.5 Sorun Tanımlama

2. Çözümleme Süreci

Bu süreçte sorun tanımlama sürecinde bakım isteği için hazırlanmış rapor alınıp, projeye ilgili bilgi ve belgeleri bir araya getirip isteğin yerine getirilmesi için gerekli planı hazırlarız.

Bu süreci iki aşamada gerçekleştiririz. Bunlardan biri olurluk aşaması bir diğeri de ayrıntılı çözümleme aşamasıdır.

Olurluk aşamasında değişikliğin sisteme etkisini inceleriz. İnsan faktörlerini göz önünde bulundurarak değişikliğin getireceği maliyeti hesaplarız, sistemin güvenliğine nasıl etki edebileceğini inceleriz, gerçekleştireceğimiz prototiplemeyi içeren seçenekleri inceleriz ve yapacağımız değişimin yararlarını inceleriz.

Ayrıntılı çözümleme aşamasında değişiklik isteği için ayrıntılı gereksinim tanımlamasını gerçekleştiririz. Bu değişiklikten sonra etkilenecek yazılım öğelerini belirleriz.(tasarım, kod, gereksinimler...) Bu öğelerden değişime gidilecek olanları belirleriz.

Bu süreçte son olarak kullanıcıya en az etki edecek şekilde Gereksinimlerinde nasıl karşılanacağı bilgilerini de içeren gerçekleştirim planımızı hazırlayıp tasarım sürecine geçeriz.



Şekil 7.6 Çözümleme Süreci

3. Tasarım Süreci

Bu süreçte yapacağımız değişiklikten etkilenebilecek tüm proje bilgi ve belgeleri üzerinde çalışma yapıp çözümlediğimiz sistem tasarımını oluştururuz.

Bu süreçte etkilenen modüllerin tanımlamasını yaparız ve modül belgelerinde yapılması gereken değişiklikleri gerçekleştirip gerçekleştirim sürecine geçeriz.



Şekil 7.7 Tasarım Süreci

4. Gerçekleştirim Süreci

Bu süreçte temel olarak tasarım raporunu ve kaynak kodları alıp değişiklik isteğini gerçekleştiren kod parçalarını bir araya getirip istenilen değişikliklerin gerçekleşeceği yeni yazılım kodlarımızı oluşturmak hedefimizdir. Bununla beraber değişiklikten sonra sınama için hazırladığımız bilgi ve raporları hazırlayıp eğitim için gerekli belgeleri hazırlarız.

Değişiklik isteklerini gerçekleştiren kodları var olan programa ekledikten sonra değişime uğrayan modüllere birim sınama uygularız. Bu aşamada riskleri gidermek amacıyla sürekli olarak risk çözümleme uygularız.



Şekil 7.8 Gerçekleştirim Süreci

5. Sistem Sınama Süreci

Bu süreçte var olan programa değişiklikler uygulandıktan sonra elde ettiğimiz yeni sürümü belirlediğimiz standartlara bağlı kalarak bütünüyle beraber sistem üzerinde gerekli sınamaları gerçekleştiririz.

Bu süreci müşteriden bağımsız bir şekilde gerçekleştirmek tercihimizdir.

Bu süreçte belirlediğimiz standartlara bağlı kalarak sonuçlanmış kod ve yazılımı gözden geçirip sınamalar ile ilgili bilgilerin kaydedilmesini sağlarız. Risk çözümlememizi gerçekleştirip yeni yazılımı ortam yönetimi altında denetlenmesini sağlarız.



Şekil 7.9 Sınama Süreci

6. Kabul Sınaması Süreci

Bu süreç kullanıcılar veya kullanıcı temsilcileri ile beraber gerçekleşecektir. Amacımız kullanıcıların veya kullanıcı temsilcilerinin değişiklikleri içeren yeni yazılımı kabul etmeleridir.



Şekil 7.10 Kabul Sınama Süreci

7. Kurulum Süreci

Bu süreçte geliştirdiğimiz yeni yazılım sürümünü uygulama alanına aktarıp kurulumunu gerçekleştirmeyi hedeflemekteyiz.

Bu süreçte fiziksel ortamın denetimi gerçekleştirip kullanıcılara gerekli bilgi ve belgeleri ulaştırıp bilgilendirilmelerini sağlarız. Var olan sistemdeki bilgileri yedekleyip geliştirdiğimiz yeni sürüme entegre ederiz.



Şekil 7.11 Kurulum Süreci

8. SONUÇ

Amaç ve kapsam hedeflerinde belirtilen planlamaya yönelik olarak tez dokümanlarının kontrolü ve doğruluğunun tespiti hedeflenmiştir. Sistem üzerinde çeşitli geliştirmeler yapılması halen gereklidir. Ancak büyük oranda YÖK Tez sisteminden elde edilen test verileri ile zaman/tasarruf/hız etkenlerine fayda sağladığı görülmüştür.

Sisteme entegre edilmesi planlanan ancak İTÜ tarafından api sağlanamaması sebebiyle türkçe kelimelerin teyiti ve doğruluğunun kontrolü sağlanamamıştır. Bununla beraber tezlerdeki bir çok hata tespit edilmiştir. Sistemin online olması bizim dış dünyaya açılmamızı sağlamıştır. Birçok platformda çalışabilecek şekilde olması sistemin geliştirilmesine ve güncelleme çalışmalarımıza da katkı sağlayacaktır.

Sistem sayesinde zaman kavramının çok değerli olması ve kontrol edilmesi gereken tez işlemlerinin otomatize edilmesi kullanıcılara çok büyük bir katkı sağlayacaktır.

9. KAYNAKLAR

1. <http://muhammetbaykara.com/2019-2020-bahar-yariyili-dersler/ymh114-yazilim-muhendisliginin-temelleri/> (srs dökümanı genel)
2. <https://www.youtube.com/watch?v=tNsmna9BBow&t=3351s> (JDBC işlemleri)
3. https://www.reqview.com/papers/ReqView-Example_Software_Requirements_Specification_SRS_Document.pdf (srs hakkında bilgiler)
4. <https://www.e-iceblue.com/Tutorials.html>
5. <https://www.w3schools.com/asp/default.ASP>
6. <https://www.tutorialspoint.com/asp.net/index.htm>
7. [https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document\(srs hakkında bilgiler\)](https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document(srs)
8. <http://hrzafer.com/netbeans-java-mysql-ornek-veritabani-uygulamasi-2>(otomasyon)
9. <https://youtu.be/7dCdDuN7CBU> (resim ekleme)
10. https://www.youtube.com/watch?v=95_IGIJJ-4s (Visio sequence diyagramı)
11. <https://www.youtube.com/watch?v=s9TEdnHz9aI> (Visio durum durum)
12. <https://www.youtube.com/watch?v=aXpQW1x6yCs&t=187s> (Visio sınıf)
13. https://www.youtube.com/watch?v=GgW_V6ZJBQg (Visio use case)
14. <https://www.youtube.com/watch?v=4dwaLtPkNTs&t=4s> (Visio etkinlik)
15. <https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.word?view=word-pia>
16. <https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.word.paragraph?view=word-pia>
17. <https://stackoverflow.com/questions/60202540/what-is-a-paragraph-microsoft-office-interop-word-paragraph>
18. <https://dottutorials.net/programmatically-generate-word-files-using-interop-in-net-core/>
19. <https://stackoverflow.com/questions/6215552/how-to-detect-empty-paragraph-in-word-document-using-microsoft-office-interop-wo>
20. <https://www.dotnetperls.com/split>
21. <https://www.dotnetperls.com/list>
22. <https://www.tutorialsteacher.com/articles/search-value-in-array-csharp>
23. <http://www.kriptarium.com/pd.html>
24. http://fbe.firat.edu.tr/sites/fbe.firat.edu.tr/files/Tez_Yazim_Kilavuzu_Eylul_2019_Revize_17-07-2020.pdf

Proje teslim klasörü içerisinde yer alan tüm bilgi, belge ve dokümanlar tarafımdan gözden geçirilmiştir. Ekip yapısı içerisinde belirtilen görev ve sorumluluklardan tarafıma atanan işleri titizlikle gerçekleştirdiğimi beyan ederim. Programlama dilleri dersi için hazırlanan tez doküman kontrolü “DocChecker” bilgim ve aktif katılımım ile hazırlanmıştır. Alınan kararların oy çokluğu ile alınması durumunda ekip üyeleri tarafından yapılan itirazlar sözlü olarak sunulmuştur ve bilgim dahilindedir.

Alperen
TOKGÖZ

İbrahim
ALTIKAT

Halef
BUDANUR

Mehmet Sait
OĞUZ

Mehmet Ali
KILINÇASLAN