**T.C.**

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Proje Dokümantasyonu**

**(Tez Denetleme)**

**Proje Ekibi**

**175542011-İsmail BİLENER**

**185541067-Furkan TAŞKÖPRÜ**

**175541066-Beyza YILDIZ**

**175542013-Leyla ALPTEKİN**

**185541048-Öznur SÖĞÜT**

|  |
| --- |
| 1. **GİRİŞ** |
| * 1. Projenin Amacı   2. Projenin Kapsamı   3. Tanımlamalar ve Kısaltmalar |
| 1. **PROJE PLANI** |
| * 1. Giriş   2. Projenin Plan Kapsamı   3. Proje Zaman-İş Planı   4. Proje Ekip Yapısı   5. Önerilen Sistemin Teknik Tanımları   6. Kullanılan Özel Geliştirme Araçları ve Ortamları   7. Proje Standartları, Yöntem ve Metodolojiler   8. Kalite Sağlama Planı   9. Konfigürasyon Yönetim Planı   10. Kaynak Yönetimi   11. Eğitim Planı   12. Test Planı   13. Bakım Planı   14. Projede Kullanılan Yazılım/Donanım Araçlar |
| 1. **SİSTEM ÇÖZÜMLEME** |
| * 1. **Mevcut Sistem İncelemesi**      1. Örgüt Yapısı      2. İşlevsel Model      3. Veri Modeli      4. Varolan Yazılım/Donanım Kaynakları      5. Varolan Sistemin Değerlendirilmesi   2. **Gereksenen Sistemin Mantıksal Modeli**      1. Giriş      2. İşlevsel Model      3. Genel Bakış      4. Veri Modeli      5. Veri Sözlüğü      6. İşlevlerin Sıradüzeni      7. Başarım Gerekleri   3. **Arayüz (Modül) Gerekleri**      1. Yazılım Ara yüzü      2. Kullanıcı Ara yüzü      3. İletişim Ara yüzü      4. Yönetim Ara yüzü   4. **Belgeleme Gerekleri**      1. Geliştirme Sürecinin Belgelenmesi      2. Eğitim Belgeleri      3. Kullanıcı El Kitapları |
| 1. **SİSTEM TASARIMI** |
| * 1. **Genel Tasarım Bilgileri**       1. Genel Sistem Tanımı      2. Kullanıcı Arabirimleri      3. Veri Modeli      4. Testler   2. **Veri Tabanı**   3. **Süreç Tasarımı**      1. Genel Tasarım      2. Modüller      3. Kullanıcı Profilleri      4. Entegrasyon ve Test Gereksinimleri |
| 1. **SİSTEM GERÇEKLEŞTİRİMİ** |
| * 1. Giriş   2. Yazılım Geliştirme Ortamları      1. Programlama Dilleri   3. Kodlama Stili   4. Program Karmaşıklığı      1. Programın Çizge Biçimine Dönüştürülmesi      2. McCabe Karmaşıklık Ölçütü Hesaplama   5. Olağan Dışı Durum Çözümleme      1. Olağandışı Durum Tanımları      2. Farklı Olağandışı Durum Çözümleme Yaklaşımları   6. Kod Gözden Geçirme      1. Gözden Geçirme Sürecinin Düzenlenmesi         1. Öbek Ara yüzü         2. Veri Kullanımı         3. Sunuş |
| 1. **DOĞRULAMA VE GEÇERLEME** |
| * 1. Giriş   2. Sınama Kavramları   3. Doğrulama ve Geçerleme Yaşam Döngüsü   4. Sınama Yöntemleri      1. Beyaz Kutu Sınaması      2. Temel Yollar Sınaması   5. Sınama ve Bütünleştirme Stratejileri      1. Yukarıdan Aşağı Sınama ve Bütünleştirme      2. Aşağıdan Yukarıya Sınama ve Bütünleştirme   6. Sınama Planlaması   7. Sınama Belirtimleri   8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri |
| 1. **BAKIM** |
| * 1. Giriş   2. Kurulum   3. Yerinde Destek Organizasyonu   4. Yazılım Bakımı      1. Tanım      2. Bakım Süreç Modeli |
| 1. SONUÇ |
| 1. KAYNAKLAR |

1. **Giriş**
   1. **Projenin Amacı:**

Projemizin amacı verilen isterler doğrultusunda kurallara bağlı bir dil tasarımı ortaya koyabilmektir.

* 1. **Projenin Kapsamı:**

Projemizin kapsamı verilen şablonu inceleyerek seçmiş olduğumuz dil kurallarına,tez şablonuna ve kurallarına uygun olup olmadığını saptamak. Kurallara uygun olduğu takdirde rapor etmek ve diğer isterlere devam etmek fakat tersi durum için isterleri tekrar gözden geçirmek ve gerektiği durumlarda değiştirmek.

**Tanımlamalar ve Kısıtlamalar**

**2.1 Proje Planı**

Proje planımız elimizdeki verilere bağlı özgün bir tasarım ortaya koyabilmek. Kullanacağımız kaynakları daha verimli hale getirebilmek için öncelikle projemizin taslağını hazırladık. Yapmak istediğimiz şey verilen isterler doğrultusunda ilk olarak bir algoritma çıkarabilmek. Algoritmamızdaki temel sorular şunlar bu algoritma bize hangi çıkışı verecek ve bu algoritma doğru hazırlanmış bir algoritma mı ?Bu soruların cevabını aldığımız bir sistem ortaya koymak

**2.1 Giriş**

Yazılan tezlerin kontrolünü yapmak çok zaman alıcıdır. Bu sistem sayesinde .docx türündeki dosyayı sisteme yükleyerek kurallara uygun olup olamadığını hata var mı yok mu sorularının yanıtını daha kolay bir şekilde bulabiliriz.

**2.2 Projenin Plan Kapsamı**

Sisteme yüklenen dosyaları, belirlenen kurallara göre algoritma kurarak bizi doğru sonuçlara götürmesi planlanmaktadır.

**2.3 Proje Zaman-İş Planı**

* Başlangıç olarak yapmak istediğimiz programın gereksinim analizini belirledik.
* Proje ile ilgili detaylı bir araştırma yaptık.
* Proje ekibimiz arasında görev dağılımını gerçekleştirdik.

.

**2.4 Proje Ekip Yapısı**

İsmail Bilener

Beyza Yıldız

Öznur Söğüt

Leyla Alptekin

Furkan Taşköprü

**2.5 Önerilen Sistemin Teknik Tanımları**

Verilen dosyayı masaüstü uygulamamız dosyanın içeriğini aşağıdaki önerilen sisteme göre yapacaktır:

* Python 3.x

**2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları**

* Visual Studio Code

**2.7 Proje Standartları, Yöntem ve Metodolojiler**

Projemizi Sprial modelle yapacağız.

Spiralin başladığı ilk çeyrek içindeki gerekler toplanılır alternatif belirlenir daha sonra karşılaşabilecek sorunlar değerlendirilir belirlenen hedefe ulaştıktan sonra geliştirmek için tartışılır bir sonraki ürün belirlenir. Aynı döngü maksimum performansa gelene kadar devam eder.

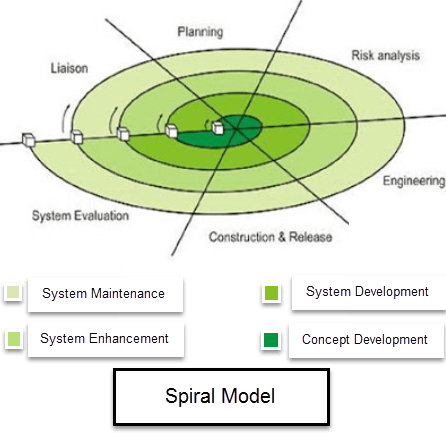
**Sprial Model**

* Amaç belirlenir
* Alternatifler belirlenir
* Alternatifler değerlendirilir
* Risk analizi yapılır
* Geliştirme yapılır
* Bir sonraki ürün belirlenir
* Bir sonraki aşama planlandırılır
* Kullanıcı değerlendirmeleri alınır

**NEDEN SPRİAL MODEL**

Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. Yazılımı kullanacak personelin sürece erken katılması ilerde oluşabilecek istenmeyen durumları engeller.

Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca var olan sorunlarla karşılaştıkları için daha kolay izleme ve sorunları daha iyi analiz ederler.



**2.8 Kalite Sağlama Planı**

Kalite planlama yazılım sürecinin erken bir evresinde başlanacaktır. Kalite planı istenen ürün kalitelerini ortaya koyar. Bunlara nasıl değer biçileceğini de tanımlamalıdır. Bu yüzden yüksek kaliteli yazılımın gerçekte ne anlama geldiğini tanımlar. Böyle bir tanım olmazsa, farklı mühendisler zıt yönlerde çalışabilirler böylece farklı ürün özellikleri en iyi duruma getirilmeye çalışılır.

* Organizasyon safhasında, yüksek kaliteli bir yazılım elde etmek için, organizasyon süreçlerine ve standartları için bir iskelet yapı oluşturmakla projemize başladık.
* Proje safhasında kalite yönetimi, belirlediğimiz kalite süreçlerinin uygulamaları projemize dâhil etmekle ve bu planladığımız süreçlerin takip edilip edilmediği kontrol etmekle yükümlüdür.
* Proje safhasında, kalite yönetimi ve oluşturduğumuz plan ile kalite hedeflerini belirledik, hangi süreç ve standartları kullanılacağımızı belirledik.

**2.9 Konfigürasyon Yönetim Planı**

Gerekli yazılımsal ve donanımsal güncellemeleri kullanıcı feedbackleri ve yazılım mühendislerimizin kaynak kodlu hataları tespiti yapılacak ve sistemimiz sürekli internet ağına bağlı olduğu için gerekli güncelleme hata görüldüğü takdirde düzeltilecektir. Programım sürekli kullanıcı ile iç içe olduğu kullanıcıdan dönüt aldığı anda gerekli bakımlar tarafımızca yapılacaktır. Büyük bakımlar gerektiği zaman gerektiğinde versiyon güncellemesi olacaktır. Gerekli konfigürasyon planı bu şekilde hazırlanmıştır.

**2.10 Kaynak Yönetimi**

Tüm grup üyeleri ile gerekli araştırmalar ve çözümleme aşamasını yaptık. Gerekli işlemler grup üyeleri tarafından bölünerek paylaşıldı. Projemiz için gerekli insan ve yazılım kaynakları tarafımızca araştırılmakta ve proje süresince gerekli analizler yapılarak oluşturulacaktır.

**2.11** **Eğitim Planı**

Ekip elamanlarıyla birlikte projeyi tam olarak anladıktan sonra ekibimizle işi ve işin nasıl yapacağını anlatarak, tartışarak en verimli hale getirmek için çalışmaktır. Müşteriye programın nasıl kullanacağına dair bilgilendirme yapılacaktır.

**2.12** **Test Planı**

İlk baş bir Beta Sürümü yayınlayacağım. Beta Sürümü başarılı olursa, dönüt alırsam bunu uygulama haline geçireceğim. Oluşturulan tasarımın grubumuz tarafından kontrol edildikten sonra hataların ve eksiklerin giderilmesi, en iyi durumun sağlanması sistemden en iyi verim alana kadar bu döngü devam eder.

**2.13** **Bakım Planı**

Programım sürekli kullanıcı ile iç içe olduğu kullanıcıdan dönüt aldığı anda gerekli bakımlar tarafımızca yapılacaktır. Büyük bakımlar gerektiği zaman gerektiğinde versiyon güncellemesi olacaktır.

**2.14 Projede Kullanılan Yazılım/Donanım Araçları**

Visual Studio Code

**3.Sistem Çözümleme**

Yazılan kod ve kod satırları aracılığıyla oluşturulan sisteme yollanan uzantılar ve ya gösterilen kaynak neticesinde istenen tezin “Fırat Üniversitesi Tez Yazma Kuralları” hususunda uygun olup olmadığını geri bildirim(Feedback) yöntemleriyle tarafımıza iletilecektir.

**3.1 Mevcut Sistem İncelemesi**

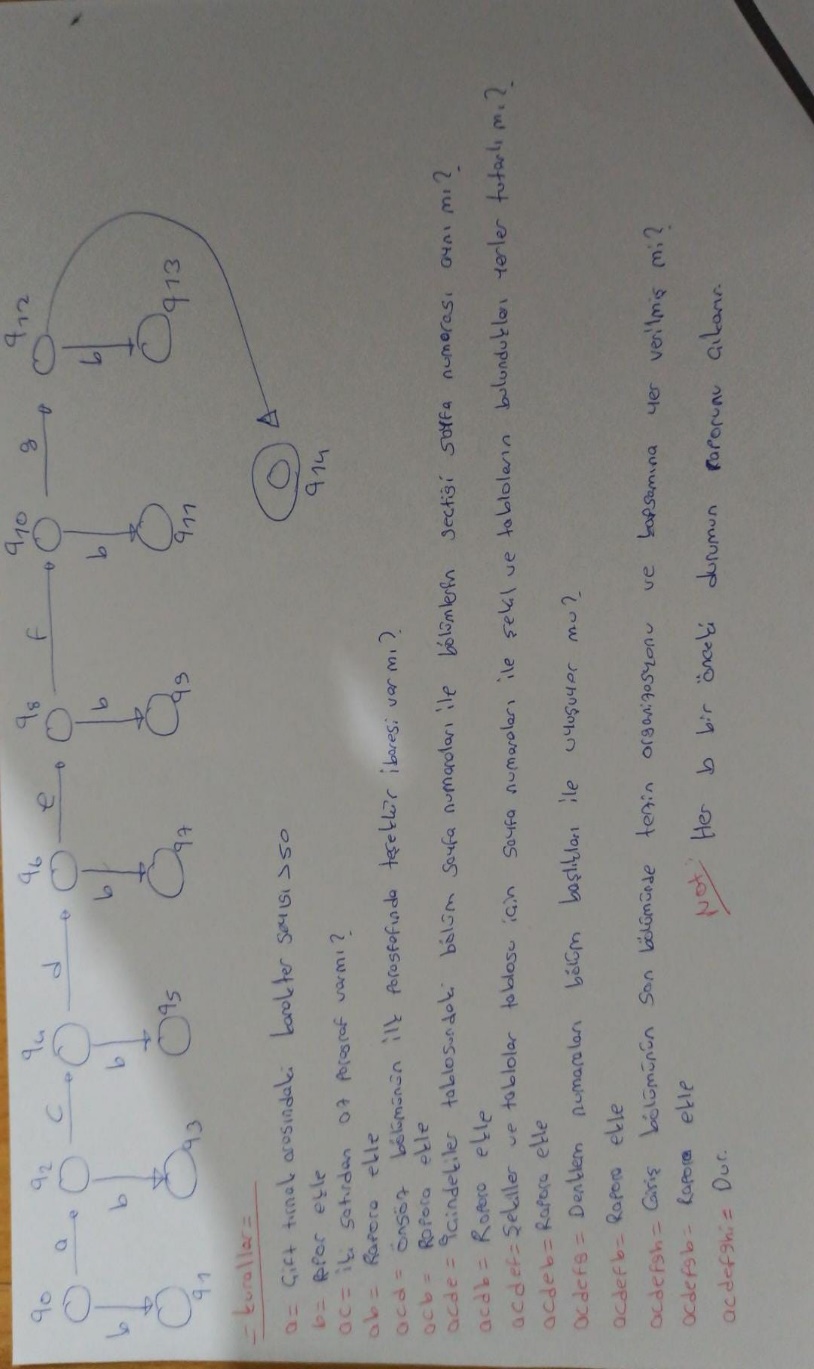
**3.1.1 Örgüt Yapısı**

* Yönetici
* Yönetici yardımcısı
* Kullanıcı

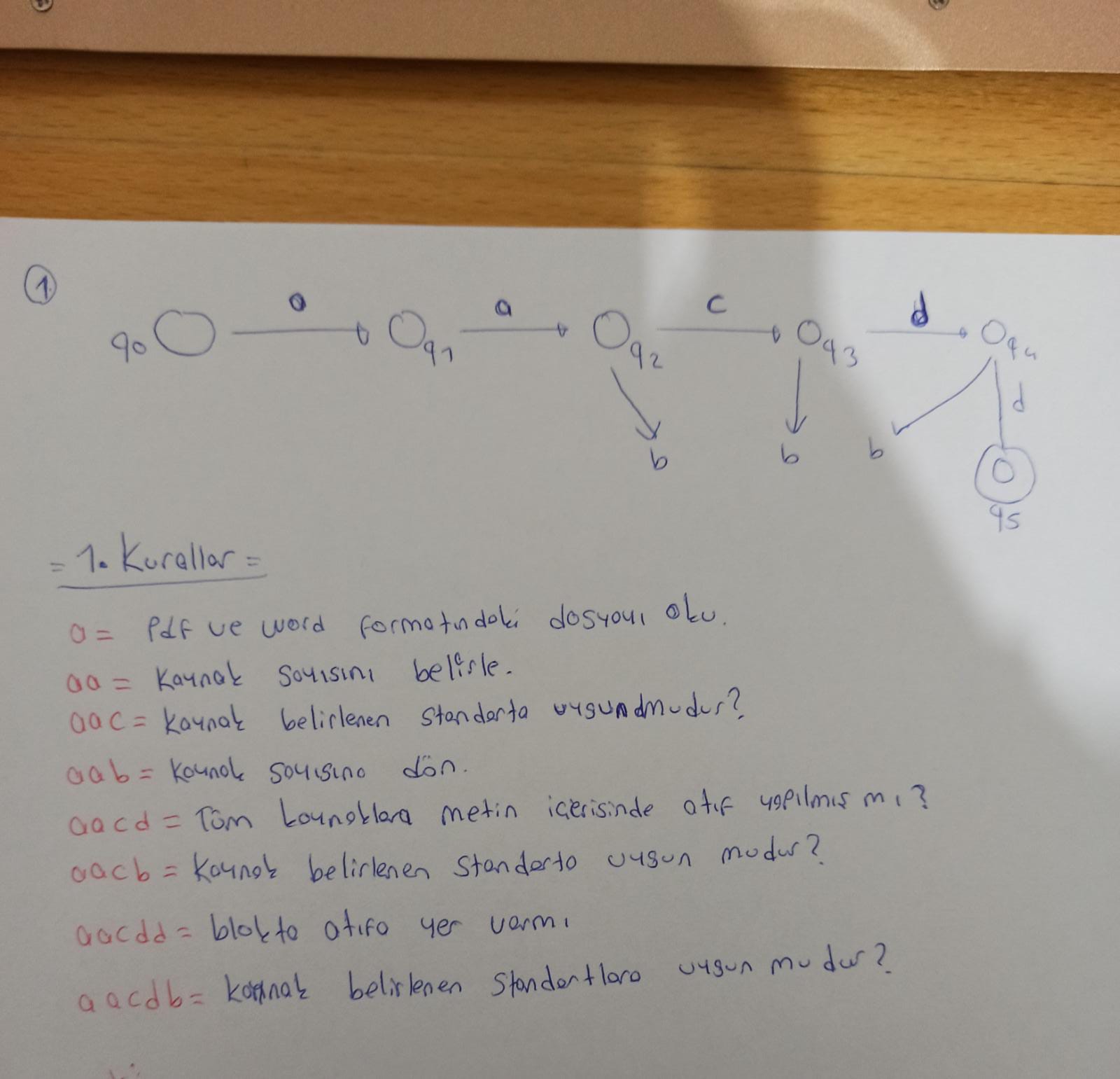
**3.1.2 İşlevsel Model**

* Kullanıcı programı çalıştırır
* Kullanıcı Word dosyasının ismini sisteme yazar
* Kullanıcı rapor oluşturulmasını ister
* Sistem gerekli düzenlemeleri yapar(hataları belirtir)
* Sistem raporu çıkarır

**3.1.3 Veri Modeli**



**Şekil3.1.3.1: DFA**



**Şekil3.1.3.2: DFA**

**3.1.4 Var Olan Yazılım/Donanım Kaynakları**

Visual Studio Code

**3.1.5 Var Olan Sistemin Değerlendirilmesi**

Uygulamayı hazır hale getirdiğimizde uygulama içinde puanlama ve yorum kısmı olacak bu sayede uygulamamızın başarılı olup olmadığını neleri geliştirebileceğimizi hangi yerlerde düzeltme yapabileceğimizi anlamış olacağız.

* + 1. **Gereksenen Sistemin Mantıksal Modeli:**

**3.2.1 Giriş**

**3.2.2 İşlevsel Model**

* Kullanıcı programı çalıştırır
* Kullanıcı Word dosyasının ismini sisteme yazar
* Kullanıcı rapor oluşturulmasını ister
* Sistem gerekli düzenlemeleri yapar(hataları belirtir)
* Sistem raporu çıkarır

**3.2.3 Genel Bakış**

Veri ihtiyaçlarımızı tanımladık. İlişkiler için daha önce belirlenen bağlayıcı ayrıntıların doğrulanması ve ayarlanmasını yaptık.

**3.2.4 Veri Modeli**

Kullanılan sistemde veri tabanı olmadığından veri modeli bulunamamaktadır.

**3.2.5 Veri Sözlüğü**

**Veri sözlüğü ne işe yarar?**

Entegrasyon zorluğu

Kişilere ya da firmalara bağımlılık

Veri anahtarı yokluğu

Bilgiye erişme güçlüğü

Tekrarlı ve çelişen veriler

Bilgi çıkarımına hazır olmama

* + 1. **İşlevlerin Sıradüzeni**
* Word formatındaki dosyayı oku.
* Kaynak sayısını belirle.
* Kaynak belirtilen standartlara uygun mudur?
* Tüm kaynaklara metin içinde atama yapılmış mı?
* Kaynak belirlenen standartlara uygun mu?
* Blokta atıflara yer var mı?
* Kaynak belirlenen standartlara uygun mu?
* Karakter sayısı>50 midir?
* İki satırdan az paragraf var mı?
* Önsöz bölümünün ilk paragrafında teşekkür ibaresi var mı?
* İçindekiler tablosundaki bölüm sayfa numaraları ile bölümlerin geçtiği sayfa numarası aynı mı?
* Şekiller ve tablolar tablosu için sayfa numaraları ile şekil ve tabloların bulundukları yerler tutarlı mıdır?
* Denklem numaraları bölüm başlıkları ile uyuşuyor mu?
* Giriş bölümünün son bölümünde tezin organizasyonu ve kapsamına yer verilmiş mi?
  + 1. **Başarım Gerekleri**
* Sistem hızlı çalışmalı
* Proje ekip yapısı düzgün seçilmeli
* Personel eğitimi yapılmalı
* Ara yüz kullanıcının anlayabileceği basitlikte olmalı
* Güvenlik sorunları, sistem zafiyetleri düzgün kontrol edilmeli
* Kullanıcıdan geri dönüş alınmalı

**3.3** **Arayüz (Modül) Gerekleri**

**3.3.1 Yazılım Ara yüzü**

Projenin yazılım ara yüzü kaynak kodları tarafımızca kapalı kaynak kodu olarak belirlenmiştir. Bu yüzden tamamlanan projemizin kodları yalnızca bizim ekibimiz tarafından düzenlenecektir.

**3.3.2 Kullanıcı Ara yüzü**

Kullanıcı programa gerekli dokümanın adını yazdığında sistem belirlenen çıktıları verecektir.

Tasarladığımız program kullanıcıya kolaylık sağlayacak ve işlemini daha rahat ve hızlı yapmasına yardımcı olacaktır.

**3.3.3 İletişim Ara yüzü**

Tasarladığımız projede herhangi bir iletişim ara yüzü bulunmamaktadır.

**3.3.4 Yönetim Ara yüzü**

Yönetim ara yüzü olarak komut satırı (cmd.exe)

**3.4 Belgeleme Gerekleri**

**3.4.1 Geliştirme Sürecinin Belgelenmesi**

Geliştirme sürecinin belgelenmesinin sebebi hem yeni geliştiricileri projeyi hızlı tanıması hem de hata ile karşılaştığımızda daha hızlı çözüme kavuşabilmemize olanak sağlamasıdır.

**3.4.2 Eğitim Belgeleri**

Şuan için herhangi bir eğitim belgemiz bulunmamaktadır.

**3.4.3 Kullanıcı El Kitapları**

Sitemimiz bir eğitime gerek olmadan kullanıcının anlayabileceği düzeyde basit olacaktır.

1. **Genel Tasarım Bilgileri**

**4.1.1 Genel Sistem Tanımı**

Komut satırı (Windows Command Processor) aracılığıyla “.docx” uzantısı ile verilen dosyanın içeriği ve stil yapısının uygulanan ya da uygun görülen standartlara göre uygunluğunu, kabul görünürlüğünü gösteren bir sistem olarak tasarlanmıştır.

**4.1.2 Kullanıcı Arabirimleri**

* Komut satırı (Windows Command Processor).
* Terminal.

**4.1.3 Veri Modeli**

Tasarladığımız Projenin hiç bir aşamasında Veri tabanına ihtiyaç duyulmadığından kullanılmamıştır. Bu yüzden veri modeli bulunmamaktadır.

**4.1.4 Testler**

**Yazılım Testi**

Yazılım kodlama aşamasında programcı tarafından oluşabilecek hataları gidermek amacıyla düzenli bir şekilde yapılır. Programcı sisteme kaynak olarak gösterdiği “.docx” uzantılı veriden alınan geri bildirimler (Feedback) aracılığı ile yazılımı test etme fırsatına sahip olacaktır.

**Sürekli Sistem testi**

Oluşturulan programın test ekibi tarafından test edildikten sonra oluşan hataların ve eksiklerin alanında uzman kişiye belirtilmesi ve en iyi durumun sağlanması ile gerçekleşir. Programdan ve cihaz hızından en iyi verimi alana kadar bu döngü devam eder.

* 1. **Veri Tabanı**

Oluşturulan projede veri tabanına ihtiyaç olmadığından veri tabanı bulunmamaktadır.

**4.3 Süreç Tasarımı**

**4.3.1 Genel Tasarım**

Oluşturulan UML diyagramından faydalanarak sistemdeki görsel ve işitsel bilgileri işlenmesi vasıtasıyla “Virtual Reality” ortamının oluşturulması.

* + 1. **Modüller**

**1.Yönetici Modülü:**

* Var olan sistemde yapılacak güncellemeler.
* Sistemin Testerlar tarafından test edilmesi.

**2.Kullanıcı Modülü:**

* Kullanıcıların görsel ve işitsel sohbet etmesi.

**4.3.3 Kullanıcı Profilleri**

**Yönetici:** Kullanıcının hareketliliğini ve davranışını izler.

**Kullanıcı:** “.docx” uzantılı dosyayı test etmek için sisteme yükler.

**4.3.4 Entegrasyon ve Test Gereksinimleri**

Var olan sistemde hataların, kodlardaki bozuklukların ve kullanıcının isteklerini göz önünde bulundurarak yapılacak güncellemelerdir. Entegrasyon testleri entegrasyon planına göre yazılım test uzmanı tarafından yapılmaktadır.

1. **SİSTEM GERÇEKLEŞTİRİMİ**

**5.1. Giriş**

Sistem gerçekleştirme, çalışan bir sistem üreten fiziksel ve kavramsal kaynakların temini ve sistemle bütünleştirilmesi olarak tanımlanmaktadır. Bu işlemler yazılımın kodlanmasıyla başlayıp test aşaması sonrası sistemin kullanıma hazır hale gelmesiyle sona erer.

Visual Studio Code ortamında gerçekleştirdiğimiz, Python ile oluşturduğumuz formda kullanıcının adını girdiği “.docx” uzantılı dosya üzerinde belirlediğimiz tez kontrol kurallarına uygun olup olmadığını raporlayan programdır.

**5.2. Yazılım Geliştirme Ortamları**

* Visual Studio Code

**5.2.1 Programlama Dilleri**

* Python

**5.3.Kodlama Stili**

Kodlama stili olarak mevcudiyette kullanılan “Python Programlama Dili” için en uygun stil olarak görülen “pep8” kullanılmaktadır. PEP8 anlaşılır ve okunabilir bir Python kodu için uymamız gereken kuralları karşımıza çıkartır.

**5.4 Program Karmaşıklığı**

Program karmaşıklığını ölçmek için birçok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe tarafından 1976 yılında geliştirilen modeldir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir. Çevrimsel karmaşıklık olan McCabe test edilmesi güç olan ve güvenilir olmayan program modüllerini ifşa etmeyi amaçlar. McCabe önce programın akış diyagramına dönüştürülmesini, daha sonra karmaşıklık ölçütünü hesaplamamızı sağlar.

**5.4.1 Programın Çizge Biçimine Dönüştürülmesi**

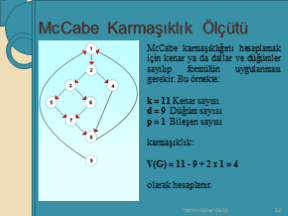
Çevrimsel karmaşıklık kısaca CC olarak adlandırılan bu yöntem, temel olarak kontrol akış grafı kullanarak ilgili modülün karmaşıklığını ölçmeye çalışır. Bunu graf üzerinde bulunan düğüm ve ayrıtların varlığına göre yapar. Graf üzerindeki her bir düğüm, program kodu içerisinde bulunan mantıksal ifadelere göre oluşturulur. Bu ifadeler if, switch, while, for, goto gibi ifadelerdir. Her bir lojik deyimi kontrol grafı üzerinde düğümlere dönüştürülerek, deyimler ayrıtlar ile birbirlerine bağlanır. Program kodundaki her bir dallanma kontrol grafı üzerindeki ayrıt yönünü belirler.

Tasarladığımız programımız bir ana program ve onunla ilgili birden fazla alt programdan oluşuyor.

Ana programı ve alt programların tümünü, McCabe Karmaşıklık ölçütünün hesaplanması için önce akış diyagramına (çizge biçimine) dönüştürdük.

* + 1. **McCabe Karmaşıklık Ölçütü Hesaplama**

Ana program ve alt programlar çizge biçimine dönüştürüldükten sonra programın McCabe karmaşıklık ölçütü (V(G)) aşağıdaki biçimde hesaplanır.

V(G)= k-d+2p = 9 - 7 + 2\*1

=4

k: kenar sayısı =9

d: düğüm sayısı =7

p: bileşen sayısı=1

* 1. **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.

* + 1. **Olağandışı Durum Tanımlar**

Olağan dışı durumlarda hata mesajına yönlendiriyor.

* + 1. **Farklı Olağandışı Durum Çözümleme Yaklaşımları**

**----**

* 1. **Kod Gözden Geçirme**

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**

Hataların bulunması, ancak düzeltilmemesini hedefledik. Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyici kullandık.

Kalite çalışmalarının bir parçası olarak ele aldık ve sonuçları düzenli ve belirlenen bir biçimde sakladık.

**5.6.1.1 Öbek Ara yüzü**

------

**5.6.1.2 Giriş Açıklamaları**

------

**5.6.1.3 Veri Kullanımı**

Verilen girdilere göre veri tipimiz belirlenmiştir. Yüklenen tezin boyutuna göre veri kullanımı belirlenir.

**6. SİSTEM GERÇEKLEŞTİRİMİ**

**6.1 Giriş**

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

* Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.
* Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.  Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.

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

**6.2 Sınama kavramları**

Sistemin istenilenleri karşılayıp karşılamadığını ekibimizce kontrol ettik.

-PDF veya Word formatındaki dosyalar okunuyor mu?

-Kaynak sayısı belirlenmiş mi?

-Kaynaklar uygun mu?

-Tüm kaynaklara metin içinde atıf yapılmış mı?

-Blok atıf verme var mı?

-Kaynaklar için istatistiksel bilgi çıkmış mı?

-Çift tırnak arasında kullanılan kelime sayısı 50’den fazla mı?

-Fazlaysa rapora eklenmiş mi?

-İki satırdan az paragraf var mı? Varsa rapora eklenmiş mi?

-Önsöz bölümünün ilk paragrafında teşekkür ibaresi var mı? Varsa rapora eklenmiş mi?

-İçindekiler tablosundaki bölüm sayfa numaraları ile bölümlerin geçtiği sayfa numaraları tutarlı mı? Tutarlıysa rapora eklenmiş mi?

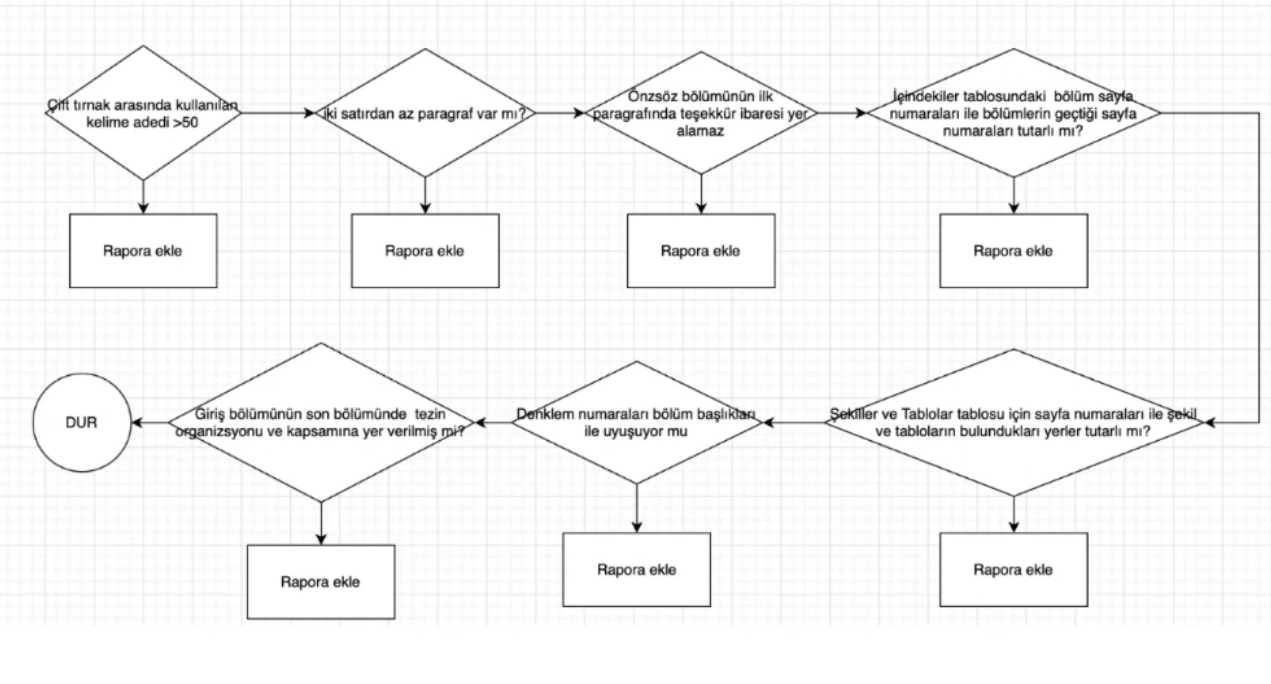
-Şekiller ve Tablolar tablosu için sayfa numaraları ile şekil ve tabloların bulundukları yerler tutarlı mı? Tutarlıysa rapora eklenmiş mi?

-Denklem numaraları ile bölüm başlıkları tutarlı mı? Tutarlıysa rapora eklenmiş mi?

-Giriş bölümünün son bölümünde tezin organizasyon ve kapsamına yer verilmiş mi? Varsa rapora eklenmiş mi?

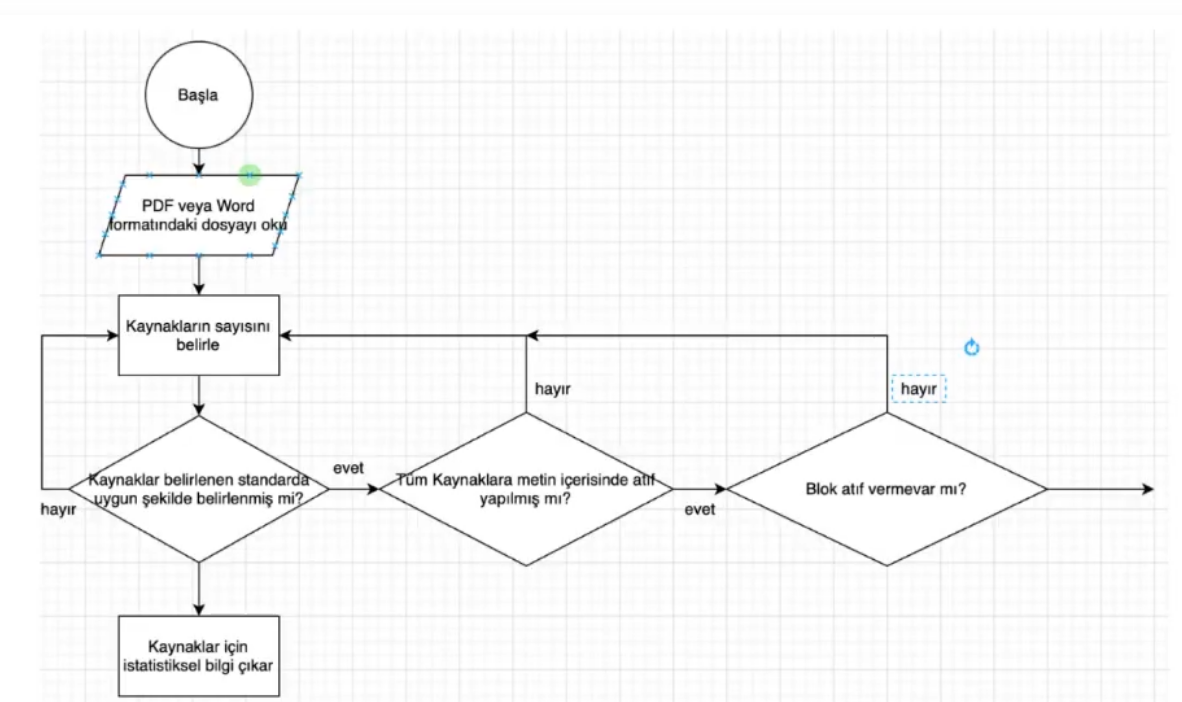
**6.3 Doğrulama ve Geçerleme Yaşam Döngüsü**

* Doğrulama ve geçerleme işlemleri yazılım üretim yaşam döngüsünün tüm süreçlerinde ve bu süreçlere koşut olarak sürdülürdü.
* Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapıldı.
* Planlama, genel olarak, birim, alt sistem, bütünleştirme, sistem ve kabul sınamalarının tasarımlarını içerdi.
* Gerçekleştirim aşamasının sonunda ise söz konusu planlar uygulandı.
* Uygulama sonucu elde edilen bulgular, Yazılım Doğrulama ve Geçerleme raporları biçiminde sürekli olarak raporlandı.
* Bu bilgiler, değişiklik denetim sistemi ve sorun yönetim sistemlerinde girdi olarak kullanılır.
* Değişiklik Denetim sistemi, sınama süresince elde edilen bulguların izlenmesi amacıyla oluşturulan bir sistemdir.

** 6.4 Sınama Yöntemleri**

Yapılabilecek denetimler arasında:

* Bütün bağımsız yolların en azından bir kere sınandı.
* Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapıldı.
* Bütün döngülerin sınır değerlerinde sınandı.

****

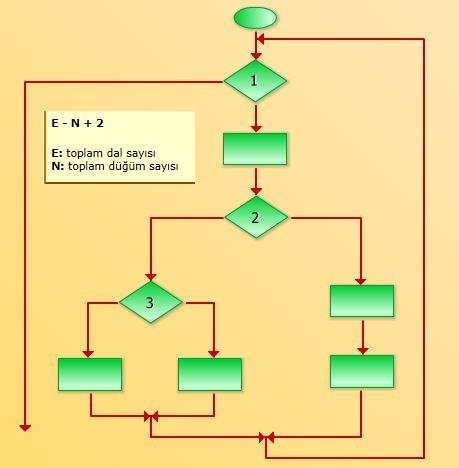
**6.4.1 Beyaz Kutu Sınaması**

-Kod yapısı ekibimizce bilindiği için bu yöntemi tercih ettik.

-Beyaz kutu**,** testleri geliştirilen yazılımın kod yapısı bilinerek gerçekleştirilen test tasarım tekniğidir.

-Beyaz kututestleri geliştirilen yazılımın içyapısı ve iş akışlarıyla ile ilgilenir.

**6.4.2 Temel Yollar Sınaması**



**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**

Sistemimiz projemiz bittiğinde en üstten başlayarak sınanmıştır. Daha sonra alt sistemlerle var olan problem belirlenmemiştir.

**6.5.2 Aşağıdan Yukarı Sınama ve Bütünleştirme**

Sistemimizde önce en alt kademeden başlayıp işçi birimleri sınadık. Sürücüler de dâhil olmak üzere sistemimizde herhangi bir sorunla karşılaşmadık.

**6.6 Sınama Planlaması**

**Birim(Modül) Sınama Planlanması:** Birimler tek tek gerekli düzeylerde denenmiş ve bir soruna rastlanmamıştır.

**Alt Sistem Sınama Planlaması:** İşçi birimler üzerinde başlanıp birimler tek tek gerekli düzeylerde denenmiş ve bir soruna rastlanmamıştır.

**Bütünleştirme Sınama Planlaması:** Sistemimiz üzerinde tüm aşamalar teker teker denenmiş ve bir soruna rastlanmamıştır.

**Sistem Sınama Planlaması:** Bütün düzeylerde birimler sisteme entegre edilerek sınama yapılmış ve bir soruna rastlanmamıştır.

**6.7 Sınama Belirtimleri**

* Sınanan program modülü ya da modüllerinin adları
* Sınama türü
* Stratejisi (beyaz kutu, temel yollar vb.)
* Sınama verileri
* Her sınama planı, sınama etkinliklerinin sınırlarını, yaklaşımını, kaynaklarını ve zamanlamasını tanımlamıştır. Sınama senaryoları olarak belirlenen her kriteri projemiz sağlamıştır.

**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 ayrı tutulur. 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.

**7. BAKIM**

**7.1 Giriş**

Projemizin en önemli kısımlarından biri bakım kısmıdır. Karşılaşılan sorun veya eksik bir durum söz konusu olunca müdahale edilmiştir. Yazılım bakım faaliyetleri yazılım yaşam döngüsünün en son yapılan faaliyeti olması açısından tüm diğer faaliyetler ile ilişkili, çeşitli uzmanlık alanlarını içeren ve müşteriyi de içine alacak şekilde bir takım çalışmasını gerektiren, karmaşık faaliyetler bütünüdür. Yönetim, iletişim, yazılım ve bilgisayar bilimlerine ait bilgilerin kullanıldığı, görev ve süreç tabanlı ilişkilerin ve rollerin bulunduğu bir alandır. Bakım ile ilgili gereksinimler, çevrimiçi yardım dokümantasyonları, dinamik kütüphaneler, güvenlik, diğer sistemlerle olan iletişim ve ilişkiler, veri toplama araçları ve toplanan verilerin saklanmasına yönelik yapılar, bakımın kaynağını oluşturmaktadır.

**7.2 Kurulum**

Kurulum yapmayı bilmeyen biri için bile oldukça kolay bir şekilde tasarlanmıştır. Setup dosyamızı indirerek ileri, ileri, ileri, kur kısımlarına basarak kurulum sağlanacaktır.

**7.3 Yerinde Destek Organizasyonu**

**-----**

* 1. **Yazılım Bakımı**

**7.4.1 TANIM**

Yaptığımız sistemde olabilecek her türlü sıkıntıda sistemimizin bakımı ekibimizce gerçekleştirilip sorunların en kısa sürede giderilmesi amaçlanmaktadır.

**7.4.2 BAKIM SÜREÇ MODELİ**

Sistem üzerinden ekibimize gelen bildirimler yine ekibimizce kontrol edilip sorun giderilecektir.

**8.Kaynakça**

<https://python-docx.readthedocs.io/en/latest/>

<https://www.horato.com/tr/post/yazilim-mimarisi-software-architecture-60>

<https://medium.com/architectural-patterns/pipe-and-filters-mimarisi-fee576ef6fb0>

<https://www.google.com/search?q=veri+modeli&oq=veri+modeli&aqs=chrome..69i57j0l3j0i22i30l4.2668j0j4&sourceid=chrome&ie=UTF-8>

<https://en.wikipedia.org/wiki/NVivo>

<https://en.wikipedia.org/wiki/Atlas.ti>

<http://www.kriptarium.com/pd.html>