**T.C.**

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Proje Dokümantasyonu**

**(Tez Denetleme)**

**Proje Ekibi**

**175541023-Gülsena BULGURLU**

**175541027-Merve KARLI**

**175541062-Barış Emre TEKSİN**

**175541003-Nurullah KAZAN**

**175541054-Güven KUŞİN**

**15.01.2021 – 1.0.0**

|  |
| --- |
| 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 oluşturulan tezleri belirli kurallar doğrultusunda kontrol etmektir.

* 1. **Projenin Kapsamı:**

Projemizde “.docx” türündeki dosyaları giriş olarak ele alıp belirli kontrollerden geçirerek yazının belirtilen kurallara uygun olup olmadığını kontrol etmek.

* 1. **Tanımlamalar ve Kısıtlamalar**

-Sisteme giriş

-Dosya yükleme

-Raporlama

**2.1 Proje Planı**

**2.1 Giriş**

Yazılan tezlerin hata kontrolünü yapmak çok zaman alır. Bu sistem sayesinde .docx türündeki dosyayı sisteme yükleyerek kurallara uygun mu? Hata var mı? Sorularının yanıtını daha kolay ve hızlı bir şekilde bulabiliriz.

**2.2 Projenin Plan Kapsamı**

Sisteme yüklenen dosyaları, istenilen kurallara göre algoritma kurarak bizi doğru çıktılara götürmesi planlanmaktadır.

**2.3 Proje Zaman-İş Planı**

-Başlangıç olarak yapmak istediğimiz programın gereksinim analizini tespit ettik.

-Proje ile ilgili araştırmalar yaptık.

-Proje ekibi arasında görev dağılımını gerçekleştirdik.

.

**2.4 Proje Ekip Yapısı**

Merve Karlı

Gülsena Bulgurlu

Barış Emre Teksin

Güven Kuşin

Nurullah Kazan

**2.5 Önerilen Sistemin Teknik Tanımları**

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

* Python 3.x

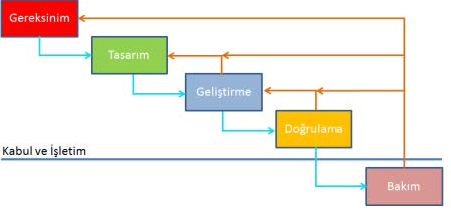
**2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları**

* Visual Studio Code

**2.7 Proje Standartları, Yöntem ve Metodolojiler**

Projemizi çağlayan (şelale) modeli ile yapacağız.

Şelale yönteminde yazılım geliştirme süreci analiz, tasarım,​​ kodlama, test, sürüm ve bakım​​ gibi safhalardan oluşur.​​ Geleneksel yazılım metotlarında bu safhalar şelale​​ modelinde olduğu gibi lineer olarak işler. Her safha,​​ başlangıç noktasında bir önceki safhanın ürettiklerini bulur.​​ Kendi bünyesindeki değişikler doğrultusunda teslim​​ aldıklarını bir sonraki safhanın kullanabileceği şekilde​​ değiştirir.​​ ​​



-Karmaşık ve nesne yönelimli projeler için uygun değildir.

-Devam eden ve uzun projeler için zayıftır.

-Projede oluşabilecek her türlü değişime elverişsiz, katı bir modeldir. Yapılan her değişiklik maliyeti büyük oranda arttırır.

-Müşteri memnuniyetini sağlamak çok zordur çünkü gelişim ve değişime açık bir model değildir.​​

-Model safhalardan oluştuğu için ürün son safhada tamamlanır, ​​ gereksinimlerin iyi tanımlanmadığı müşterinin ne istediğinin anlaşılmadığı bir projede bu durum projenin bittikten sonra iptal edilmesine ve başka gerginliklere sebep olmaktadır.

-Kullanımı ve anlaması basittir.

-Yönetimi kolaydır.

-Projenin safhaları ayrı olduğundan iş bölümü ve iş planı projenin en başında net bir şekilde bellidir. Bu durum projenin yönetimini de oldukça kolay hale getirir.​​

-Şelale Modeli çok küçük ve gereksinimleri çok iyi anlaşılmış projelerde iyi çalışır.​​

**2.8 Kalite Sağlama Planı**

-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**

Yazılım yönünden ve donanım yönünden güncellemeleri, kullanıcı geri bildirimleri(Feedback) ve sürekli internet ağına bağlı olan programımız için gerekli güncellemeleri yapacaktır. Kaynak kodlu hataları tespit edecektir. Gerekli yapılandırma planımız bu şekilde hazırlanmıştır.

**2.10 Kaynak Yönetimi**

Tüm grup üyeleri ile gerekli araştırmalar ve çözümleme aşamasını yaptık. Gerekli işlemler grup üyeleri tarafından bölünerek paylaşıldı.

Sistem Gerçekleştirimi: Barış Emre Teksin, Güven Kuşin, Merve Karlı, Gülsena Bulgurlu, Nurullah Kazan

Dokümantasyon ve gerekli araştırmalar:

Barış Emre Teksin: Sonuç, Yöneticilik, Sistem Gerçekleştirimi

Nurullah Kazan: Proje Planı, Kaynaklar

Merve Karlı: Sistem Çözümleme, Bakım

Gülsena Bulgurlu: Sistem Tasarımı, Giriş

Güven Kuşin: Doğrulama ve Geçerleme

**2.11** **Eğitim Planı**

Grubumuzdaki bireylerle proje hakkında araştırmalar yapıp birbirimize aktaracağız.

Müşteriye programın nasıl kullanacağına dair bilgilendirme yapılacaktır.

**2.12** **Test Planı**

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**

Sistem üzerinden oluşturulacak hataların bakım ekibi tarafından düzeltilmesi ve duruma bağlı olarak sürekli kontrolünün yapılmasıdır.

**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**

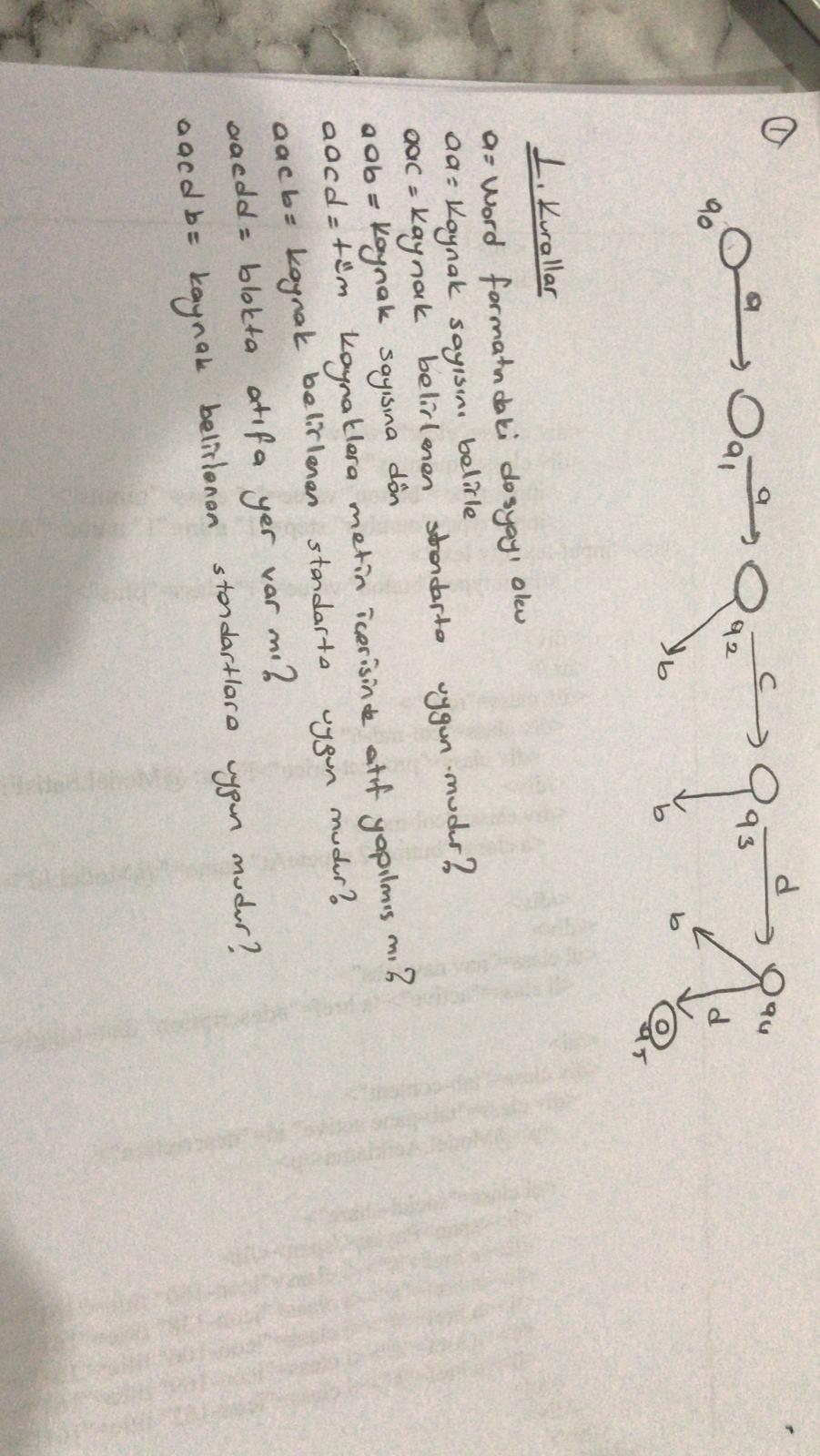
**3.1.1 Örgüt Yapısı**

* Yönetici
* Yönetici yardımcısı
* Kullanıcı

**3.1.2 İşlevsel Model**

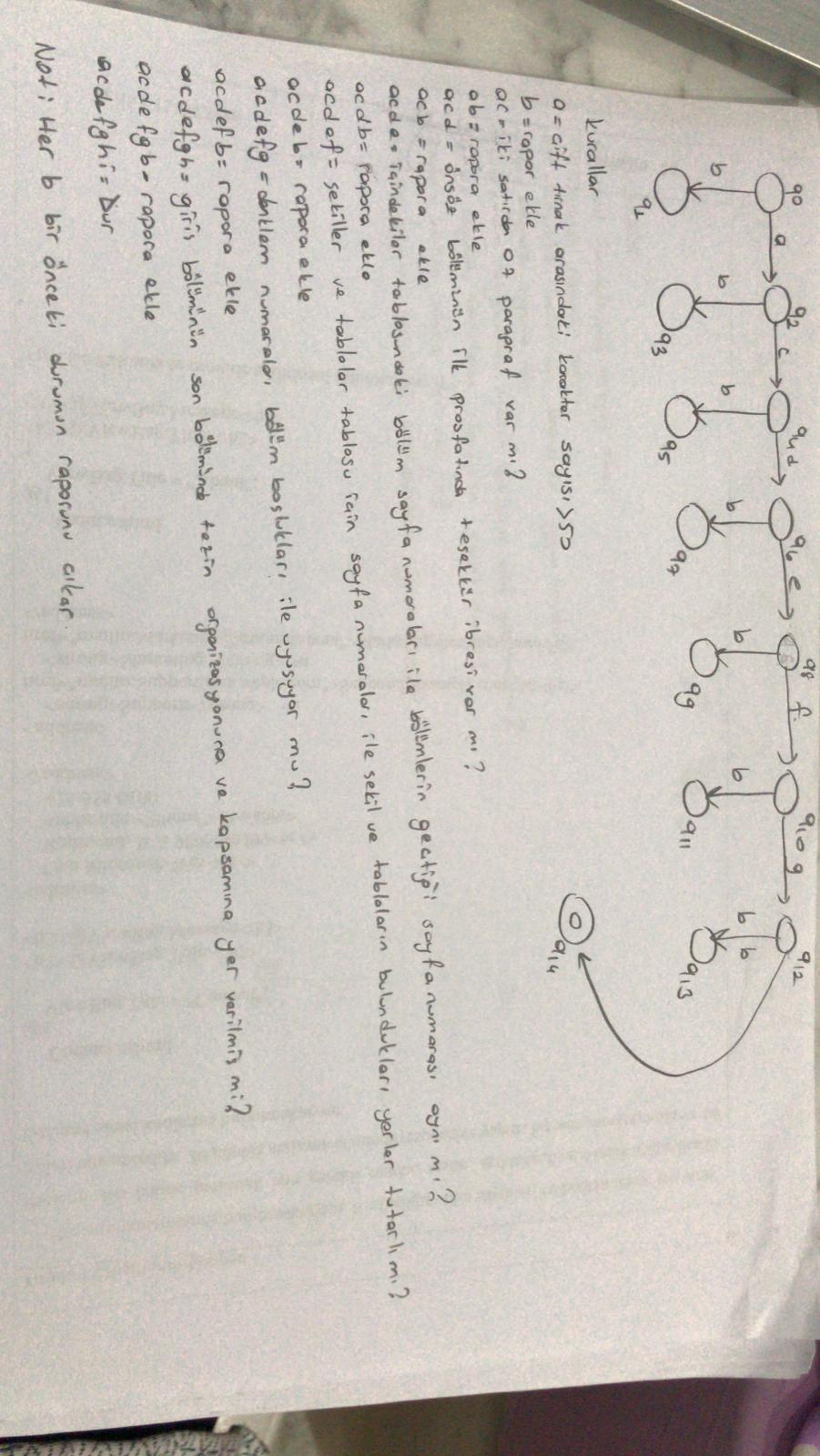
* Kullanıcı programı çalıştırır
* Kullanıcı Word dosyasının ismini yazar
* Kullanıcı rapor oluşturulmasını ister
* Sistem gerekli düzenlemeleri yapar
* Sistem raporu çıkarır

**3.1.3 Veri Modeli**



**Şekil3.1.3.1: DFA**

DFA yani Deterministic Finite Automata sonlu sayıda duruma sahip ve her durumdan belirlenen alfabenin tüm harfleri ile bir çıkışa sahip makine türüdür. DFA’lar durum sayıları azaltılarak indirgenebilir ve daha ekonomik ve hızlı çalışan makineler elde edilebilir. Bu açıdan DFA optimizasyonu önemlidir.



**Şekil3.1.3.2: DFA**

**3.1.4 Var Olan Yazılım/Donanım Kaynakları**

Visual Studio Code

**3.1.5 Var Olan Sistemin Değerlendirilmesi**

Verilen Word dosyasının müşteri tarafından belirlenen sistem kısıtlarına uygunluğunu sorgular.

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

**3.2.1 Giriş**

**3.2.2 İşlevsel Model**

* Kullanıcı programı çalıştırır
* Kullanıcı Word dosyasının ismini yazar
* Kullanıcı rapor oluşturulmasını ister
* Sistem gerekli düzenlemeleri yapar
* Sistem raporu çıkarır

**3.2.3 Genel Bakış**

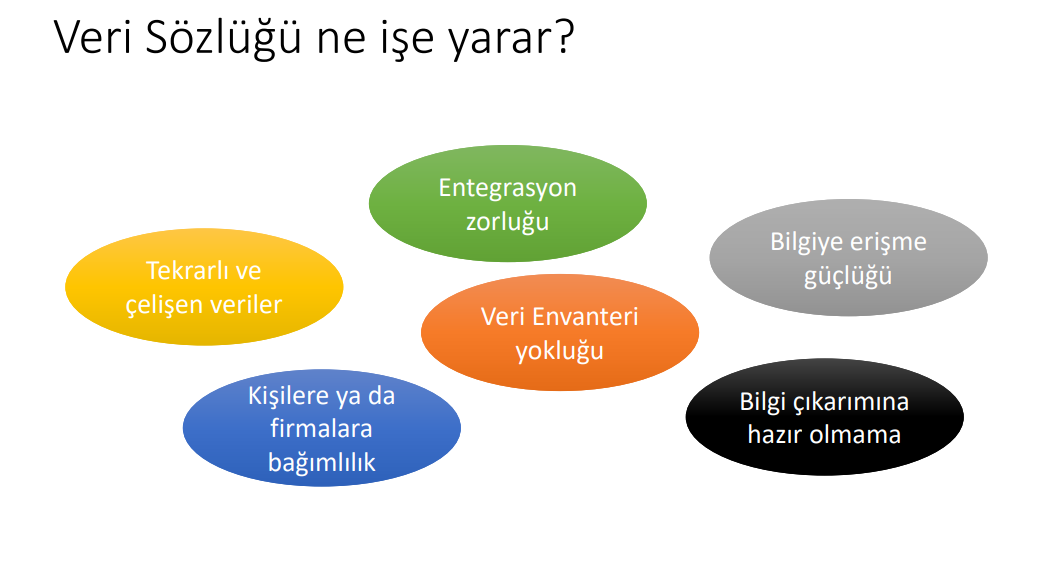
Veri ihtiyaçlarımızı tanımladık. İlişkiler için daha önce belirlenen bağlayıcı ayrıntıların doğrulanması ve ayarlanmasını yaptık.

**3.2.4 Veri Modeli**

Bir VTYS(Veri Tabanı Yönetim Sistemi)'yi kullanarak oluşturulacak her veri tabanında yer alacak veriler ve veriler arası ilişkiler, mantıksal düzeyde ilgili veri modeline göre düzenlenir; bu veri modeli kullanılarak veri tabanının kavramsal ve dış şemaları oluşturulur.

Kullanılan sistemde veri tabanı olmadığından veri modeli bulunamamaktadır.

**3.2.5 Veri Sözlüğü**



* + 1. **İşlevlerin Sıradüzeni**

-Word formatındaki dosyayı oku. Evet

-Kaynak sayısını belirle. Evet

-Kaynak belirtilen standartlara uygun mudur? Evet

-Tüm kaynaklara metin içinde atama yapılmış mı? Evet

-Kaynak belirlenen standartlara uygun mu? Evet

-Blokta atıflara yer var mı? Evet

-Kaynak belirlenen standartlara uygun mu? Evet

-Karakter sayısı>50 midir? Evet

-İki satırdan az paragraf var mı? Evet

-Önsöz bölümünün ilk paragrafında teşekkür ibaresi var mı? Evet

-İçindekiler tablosundaki bölüm sayfa numaraları ile bölümlerin geçtiği sayfa numarası aynı mı? Evet

-Şekiller ve tablolar tablosu için sayfa numaraları ile şekil ve tabloların bulundukları yerler tutarlı mıdır? Evet

-Denklem numaraları bölüm başlıkları ile uyuşuyor mu? Evet

-Giriş bölümünün son bölümünde tezin organizasyonu ve kapsamına yer verilmiş mi? Evet

* + 1. **Başarım Gerekleri**

-Sistem hızlı çalışmalı

-Ara yüz kullanıcının anlayabileceği basitlikte olmalı

-Proje ekip yapısı düzgün seçilmeli

-Personel eğitimi yapılmalı

-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 projemiz tamamlandığında kodları yalnızca bizim ekibimiz tarafından düzenlenecektir.

**3.3.2 Kullanıcı Ara yüzü**

Kullanıcı tasarladığımız programa gerekli dokümanın adını yazdığında sistem belirlenen çıktıları verecektir.

**3.3.3 İletişim Ara yüzü**

İletişim ara yüzümüz 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 sürecimiz tüm ayrıntılarıyla belgelenecektir.

Belgelemenin sebebi hata ile karşılaştığımızda daha hızlı çözüme kavuşabilmemize olanak sağlamasıdır.

**3.4.2 Eğitim Belgeleri**

Şuan için belgemiz bulunmamaktadır.

**3.4.3 Kullanıcı El Kitapları**

Kullanıcıya sistem el kitabı oluşturmadık. Sistemimiz 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**

Veri tabanı kullanılmadığından veri modeli bulunmamaktadır.

**4.1.4 Testler**

**Yazılım Testi**

Yazılım testimiz programcı kodlama yaptığı esnada gerçekleşiyor. 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 edilmesi ve oluşan hataların uzman kişiye yönlendirilmesiyle gerçekleşir. Programdan ve cihaz hızından en iyi verimi alana kadr bu işlem devam eder.

* 1. **Veri Tabanı**

Oluşturulan projede 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 Test ekibi tarafından test edilmesi.

**2.Kullanıcı Modülü:**

* Kullanıcıların görsel ve işitsel sohbet etmesi.

**4.3.3 Kullanıcı Profilleri**

**Yönetici:** Kullanıcının hareketliliğini izler.

**Kullanıcı:** Test edilecek “.docx” uzantılı dosyayı sisteme yükler.

**4.3.4 Entegrasyon ve Test Gereksinimleri**

Var olan kodlardaki bozuklukları düzeltip ve kullanıcının isteklerini göz önünde bulundurarak yapılacak yenilemelerdir.

1. **SİSTEM GERÇEKLEŞTİRİMİ**

**5.1. Giriş**

Python ile oluşturduğumuz formda kullanıcının adını girdiği “.docx” uzantılı dosya üzerinde belirlediğimiz kurallara uygun olup olmadığını belirleyen 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.

**5.4 Program Karmaşıklığı**

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

**5.4.1 Programın Çizge Biçimine Dönüştürülmesi**

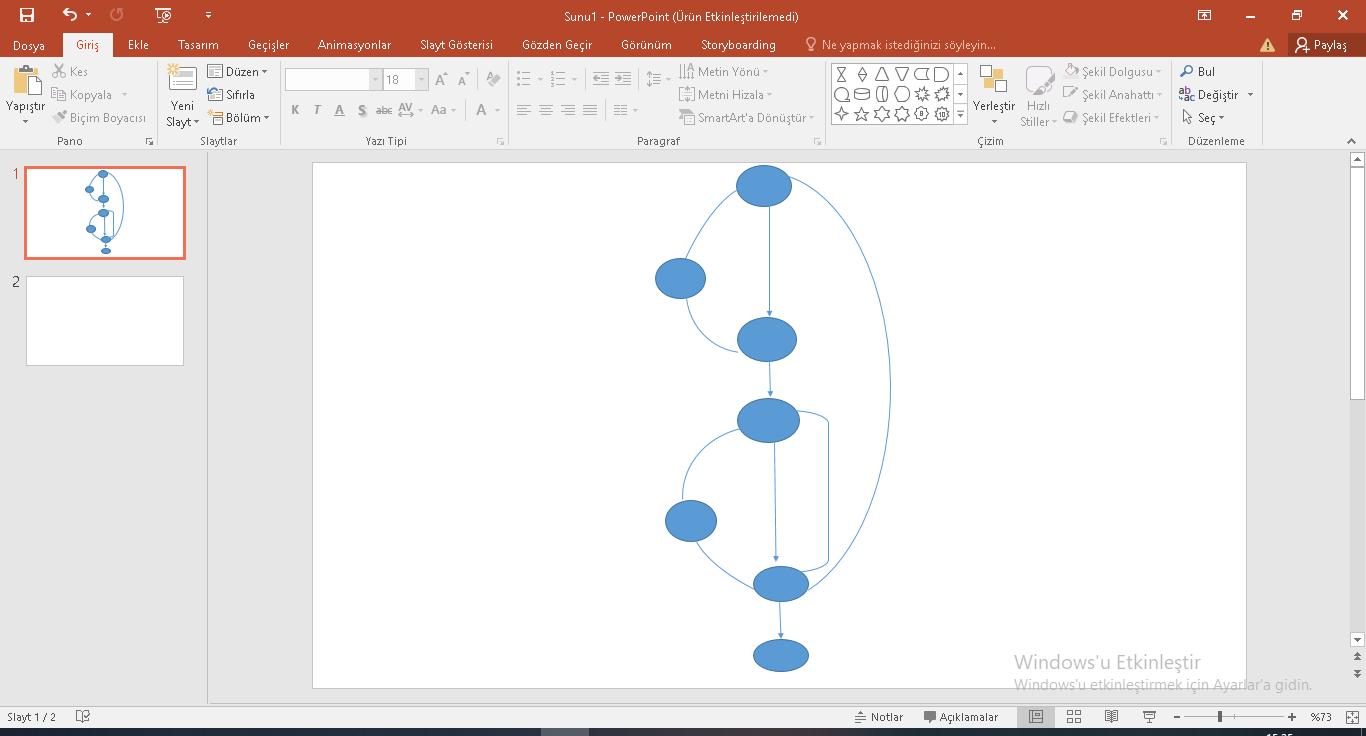
Programımız bir ana program ve onunla ilgili birden fazla alt programdan oluşuyor.

Ana programı ve gerekse alt programların tümünü, McCabe Karmaşıklık ölçütünün hesaplanmasından önce çizge biçimine dönüştürdük.

Projemizde yer alan örnek bir kısa kod parçası:

# 

Kısa kod parçasına ait çizgesi:



* + 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**

Programımızın işletiminin sonlandırılması işleminin -elektrik kesintisi vb donanım hataları dışında- bütünüyle program denetimini kontrol ettik. Bu denetim olağan dışı sonlandırma işlemini de kapsamaktadır. Bu nedenle program kodlarımızı oluşturulurken, olağandışı durumları da dikkate aldık ve bu durumlardaki program davranışına ilişkin yöntemler geliştirdik.

* + 1. **Olağandışı Durum Tanımlar**

Olağan dışı durumlar, programlama dili tarafından tanımlı durumlar olduğu gibi kullanıcı tarafından da tanımlanabilmektedir.

Bu tür olağan dışı durumlar, olağan dışı durumu tanımlayan ve sonucunda "doğru" ya da "yanlış" değeri üreten biri mantıksal yordam ya da fonksiyon tanımından oluşmaktadır.

Python, ayrıca yorumlanabilir, interaktif ve nesne odaklı bir programlama dilidir. Bu dil modüller, olağandışı durumlar, dinamik yazım, oldukça yüksek dinamik veri türleri ve sınıfları ile birlikte çalışmaktadır.

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

**-**Anında Durdurma: Hata bulunduğunda program açıklayıcı bir ileti vererek sona erer.

**-**Hata Kodu Verme: İlgili program modülünün başarısız şekilde çıkıldığında, programın bir hata kodu vermesi biçiminde bir yaklaşımdır.

**-**Hata Yordamı Çalıştırma: Yordamın beklenmeyen bir değer geri döndürmesi gibi bir yaklaşımdır.

* 1. **Kod Gözden Geçirme**

Kod gözden geçirme ile program sınama işlemlerini birbirinden ayırmak gerekir.

**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**

Her öbek tek bir işlevsel amacı yerine getiriyor mu? Evet

Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi? Evet

Öbek tek girişli ve çıkışlı mı? Evet

Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu? Evet

**5.6.1.2 Giriş Açıklamaları**

Öbek, doğru biçimde giriş açıklama satırları içeriyor mu? Evet

Giriş açıklama satırları, öbeğin amacını açıklıyor mu? Evet

Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu? Evet

Giriş açıklama satırları, çıktıları ve hata iletilerini tanımlıyor mu? Evet

Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu? Evet

Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu? Hayır

Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu? Evet

Giriş açıklama satırları, öbeği yazan kişi ve yazıldığı tarihle ilgili bilgileri içeriyor mu? Evet

Her paragrafı açıklayan kısa açıklamalar var mı? Var

**5.6.1.3 Veri Kullanımı**

İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı? Evet

Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı? Evet

Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı? Hayır

Her değişken tek bir amaçla mı kullanılıyor? Hayır

Tanımlanan her gösterge değişken için bellek ataması yapılmış mı? Evet

**5.6.1.4 Sunuş**

Her satır, en fazla bir deyim içeriyor mu? Evet

Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşıla bilirliği kolaylaştıracak biçimde anlamlı mı? Evet

Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı? Evet

Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantez kullanılmış mı? Evet

Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı? Evet

Öbek yapısı içerisinde akıllı ”programlama hileleri” kullanılmış mı? Hayır

**6. SİSTEM GERÇEKLEŞTİRİMİ**

**6.1 Giriş**

Sistem gerçekleştirim aşamaları bu bölümde anlatılacaktır.

**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?

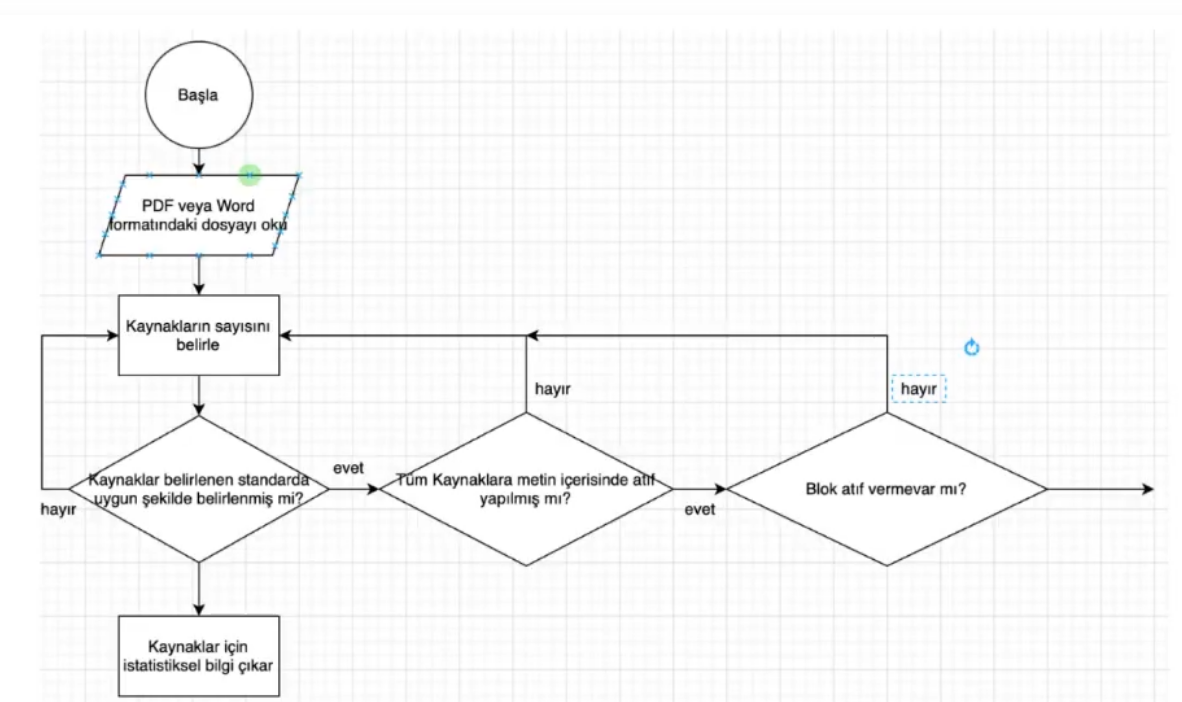
-İçindekiler tablosundaki bölüm sayfa numaraları ile bölümlerin geçtiği sayfa numaraları tutarlı mı? Tutarlıysa rapora eklenmiş mi?

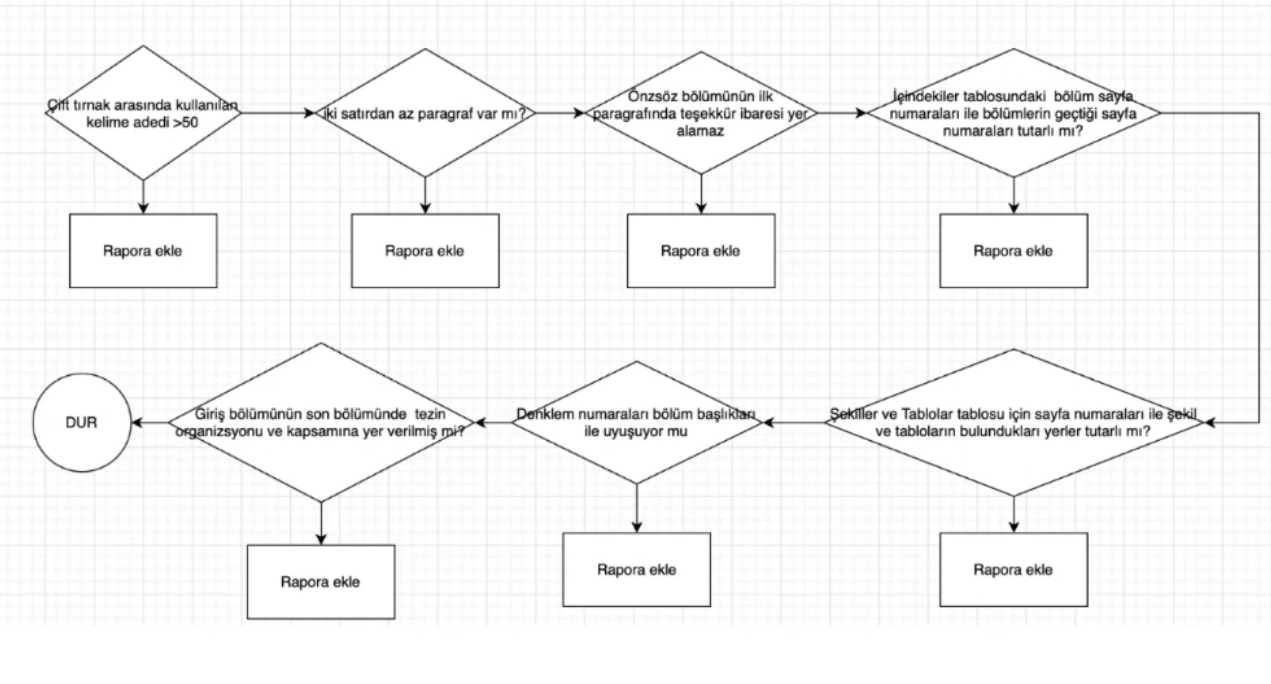
-Şekiller ve Tablolar tablosu için sayfa numaraları ile şekil ve tabloların bulundukları yerler tutarlı mı? Tutarlıysa rapora eklenmiş mi?

-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**

****

****

**6.4 Sınama Yöntemleri**

Sistemi test etmek için aşağıdaki sınama yöntemlerini tercih ettik.

**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**

Ekibimiz tarafından toplamda 15defa sınama yapılmış ve sistemimiz bütün testleri başarıyla geçmiştir.

**6.5 Sınama ve Bütünleştirme Stratejileri**

Sistemimiz sınamaları başarıyla geçtikten sonra bütün bileşenler üzerinde yukarıdan aşağı ilgili modülleri takarak ve aşağıdan yukarı olacak şekilde bileşenleri tekrardan sınadık.

**6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme**

Sistemimiz ekibimizce en üst düzeyden en alta kadar ilgili modülleri kullanarak testlerimizi tekrardan gerçekleştirdik ve sistemimiz bu testleri de başarıyla geçti.

(Resimler eklenecek.)

**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.(Resimler eklenecek.)

**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**

Aşağıdaki aşamaların hepsi teker teker kontrol edilmiş ve bir soruna rastlanmamıştır. Evet.

Temel kod yapısı kontrol edildi mi? Edildi ise rapora eklendi mi? Evet.

Beyaz kutu sınama verileri rapora eklendi mi? Evet.

Kullanıcı senaryoları test edildi mi? Sonuçlar rapora eklendi mi? Evet.

**6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

Yaptığımız sınamalar sonucu sistemimiz %100 tam başarı ile testleri geçtiği için Değişiklik Kontrol Sistemi(DKS)’ne herhangi bir kayıt girilmemiştir.

**7 DOĞRULAMA VE GEÇERLEME**

**7.1 GİRİŞ**

**7.2 KURULUM**

Kurulumu kullanıcının anlayabileceği düzeyde basit olacaktır.

**7.3 YERİNDE DESTEK ORGANİZAYSONU**

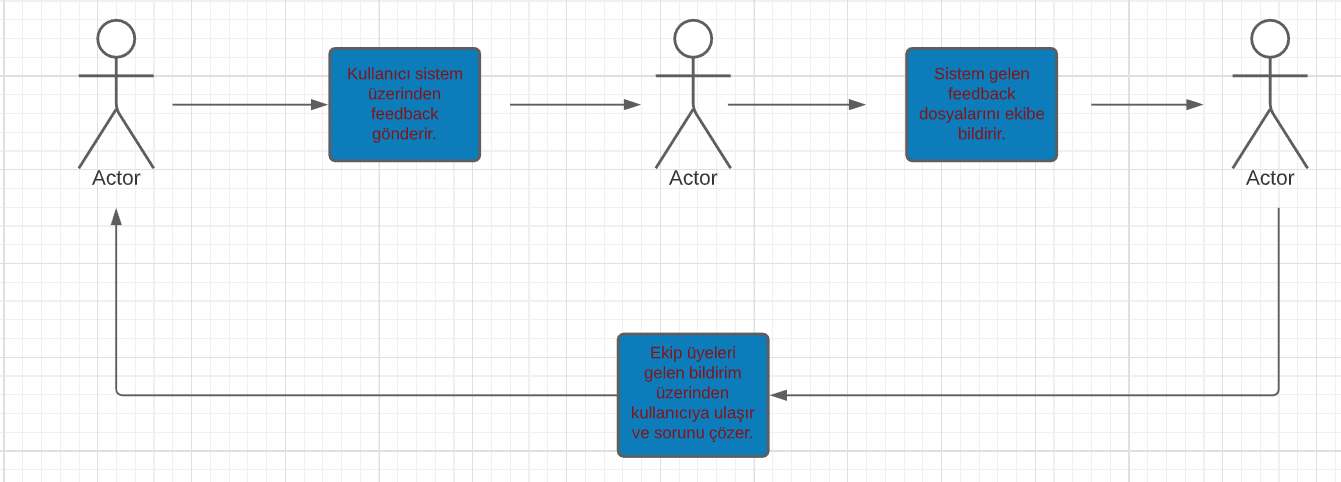
Kullanıcının yaşayabileceği herhangi bir sıkıntıda ekibimiz tarafından çevrimiçi olarak destek sağlanacaktır.

**7.4 YAZILIM BAKIMI**

**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://docs.python.org/3/howto/logging.html>

<http://muhammetbaykara.com/wp-content/uploads/2017/04/YMT114-7.Hafta-Ger%C3%A7ekle%C5%9Ftirim.pdf>

<https://askingedik.net/2013/05/10/code-style/>

<https://bilgisayarnot.blogspot.com/2020/05/program-karmasikligi-nedir.html>

<https://gokremtekir.com/category/proje-kaynak-yonetimi/>

<https://python-istihza.yazbel.com/donguler.html>