**T.C.**

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Proje Dokümantasyonu**

**Tezkont**

**Proje Ekibi**

**180541063-Emirhan ALTUNEL**

**180541073-Murat Can TUNÇEL**

**170541029-Osman Nuri YAĞAR**

**180541014-Özgür YALIM**

**180541028-Yusuf AKBAŞ**

**Aralık – 2020**

|  |
| --- |
|  |
| 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 Yönetim Planı   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. Bilgi Sistemleri/Nesneler      5. Veri Modeli      6. Veri Sözlüğü      7. İşlevlerin Sıradüzeni      8. Başarım Gerekleri   3. **Arayüz (Modül) Gerekleri**      1. Yazılım Arayüzü      2. Kullanıcı Arayüzü      3. İletişim Arayüzü      4. Yönetim Arayüzü   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. Varsayımlar ve Kısıtlamalar      3. Sistem Mimarisi      4. Dış Arabirimler         1. Kullanıcı Arabirimleri         2. Veri Arabirimleri         3. Diğer Sistemlerle Arabirimler      5. Veri Modeli      6. Testler      7. Performans   2. **Veri Tasarımı**      1. Tablo tanımları      2. Tablo- İlişki Şemaları      3. Veri Tanımları      4. Değer Kümesi Tanımları   3. **Süreç Tasarımı**      1. Genel Tasarım      2. Modüller         1. XXX Modülü            1. İşlev            2. Kullanıcı Arabirimi            3. Modül Tanımı            4. Modül iç Tasarımı         2. YYY Modülü      3. Kullanıcı Profilleri      4. Entegrasyon ve Test Gereksinimleri   4. **Ortak Alt Sistemlerin Tasarımı**      1. Ortak Alt Sistemler      2. Modüller arası Ortak Veriler      3. Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri      4. Güvenlik Altsistemi      5. Veri Dağıtım Altsistemi      6. Yedekleme ve Arşivleme İşlemleri |
| 1. **SİSTEM GERÇEKLEŞTİRİMİ** |
| * 1. Giriş   2. Yazılım Geliştirme Ortamları      1. Programlama Dilleri      2. Veri Tabanı Yönetim Sistemleri         1. VTYS Kullanımının Ek Yararları         2. Veri Modelleri         3. Şemalar         4. VTYS Mimarisi         5. Veritabanı Dilleri ve Arabirimleri         6. Veri Tabanı Sistem Ortamı         7. VTYS'nin Sınıflandırılması         8. Hazır Program Kütüphane Dosyaları         9. CASE Araç ve Ortamları   3. Kodlama Stili      1. Açıklama Satırları      2. Kod Biçimlemesi      3. Anlamlı İsimlendirme      4. Yapısal Programlama Yapıları   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      2. Gözden Geçirme Sırasında Kullanılacak Sorular         1. Öbek Arayüzü         2. Giriş Açıklamaları         3. Veri Kullanımı         4. Öbeğin Düzenlenişi         5. 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**

**1.1 Projenin Amacı**

Dünya üzerinde kabul edilen lisansüstü tez kurallarına göre kullanıcın hazırladığı tezin bu kurallar çerçevesinde doğruluğunu kontrol etmek.

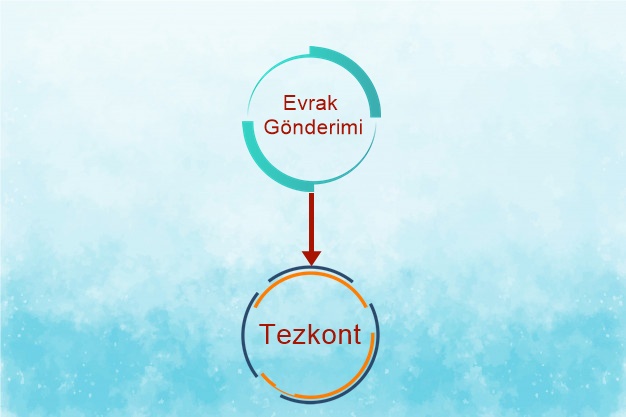
**1.2 Projenin Kapsamı**

Tez kontrolü yapan kurum ve kuruluşlar, akademisyenler ve lisans üstü öğrencilerin kullanacağı bu masaüstü uygulaması Giriş, Proje Planı, Sistem Çözümleme, Sistem Tasarımı, Sistem Gerçekleştirimi, Doğrulama ve Geçerleme, Bakım, Sonuç ve Kaynakların doğru yazıldığının doğruluğu kısımlarını kontrol eder.

**2. Proje Planı**

**2.1 Giriş**

Kullanıcı tarafından programa verilen tez dosyasını belirlenen kurallara göre inceleyip kullanıcıya geri dönüş sağlar.

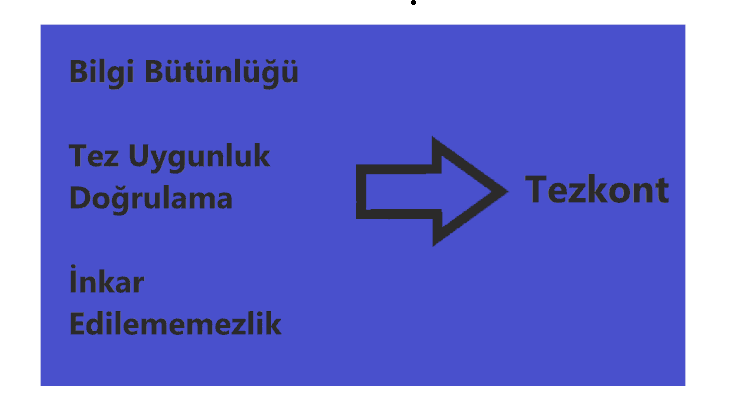


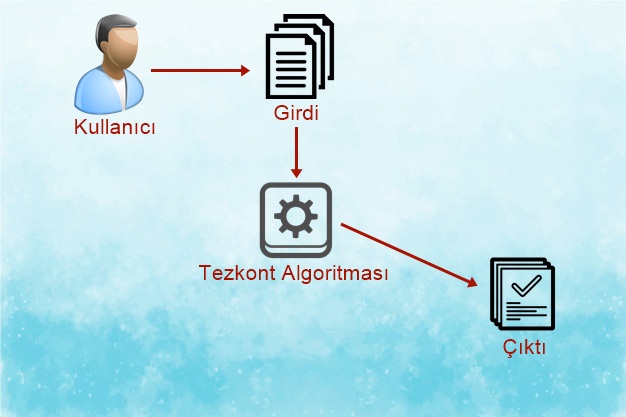
**2.2 Projenin Plan Kapsamı**

Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin güvenilirliğinden yola çıkıldı. Tezkont, sunulan tezlerin tez kurallarına uygun, kaynak gösterilmeyen bilgilerin elektronik ortamda kontrolünü sağlayıp bilgi hırsızlığını önleyen bununla beraber vakit kaybı olmadan ve daha kolay kullanıma olanak sağlar.



Bu teknolojinin avantajları;

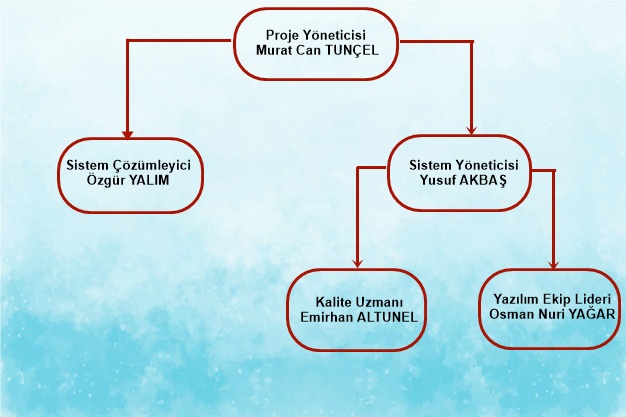




**2.3 Proje Zaman-İş Planı**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Zaman \ İş | 1.Hafta | 2.Hafta | 3.Hafta | 4.Hafta | 5.Hafta |
| Proje Planı | **+** |  |  |  |  |
| Analiz | **+** | **+** |  |  |  |
| Çözümleme |  | **+** | **+** |  |  |
| Tasarım |  |  | **+** |  |  |
| Gerçekleştirim |  |  |  | **+** | **+** |
| Bakım |  |  |  |  | **+** |

**2.4 Proje Ekip Yapısı**



**2.5 Önerilen Sistemin Teknik Tanımları**

-> Derin Metin Okuma

Tezin söz dizimi olarak doğruluğu kontrol edilir.

-> Metin Okuma

Tezin metnini okur.

**2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları**

Programlama Araçları

-> Visual Studio Code

Sınama Araçları

-> Windows 10

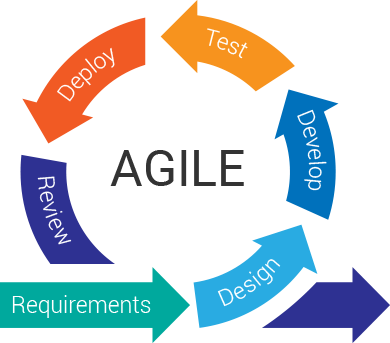
**2.7 Proje Standartları, Yöntem ve Metodolojiler**

Proje de Çevik (Agile) Model kullanılmıştır. Proje de Çevik Model kullanılmasının nedeni:

-> Analiz, tasarım ve test etme süreçlerinin ne kadar zaman alacağının önceden bilinememesi.

-> Proje ekibi olarak birlikte çalışmak, hiyerarşiye önem vermemek, sağlam iletişim kurmak gibi özelliklere sahip olunduğundan

-> Projenin parçalarının önce tasarlanıp ardından hemen geliştirilmesinin gerekmesi ve önceden ne yapılacağını, detaylı yol haritasını ve tasarımını tahmin etmenin çok güç olması



**2.8 Kalite Sağlama Planı**

1.Ekonomi: Maliyeti online kaynaklardan elde edilecek veriler ve mevcut işlem bilgisayarlarından oluşacağından dolayı oldukça hesaplı olacaktır.

2.Tamlık: Projede herhangi bir hata, açık bulunmamalı tüm kodlar ve textler düzgün ve çalışır durumdadır

3.Yeniden Kullanılabilirlik: Kontrolcü her koşulda tekrardan konfigüre edilip devamlı kullanılabilir hale getirilebilir.

4.Etkinlik: Kullanıcı sadece ön tarafı göreceğinden sadece girdi çıktıları etkin bir şekilde kullanacaktır.

5.Bütünlük: Mühendis programın arka tarafını yönetebilirken kullanıcı sadece ön tarafla ilgilenecek ve program bütün halde ilerleyecektir.

6.Güvenilirlik: Kontrolcü tüm güvenlik önlemlerinin içinde saklanacaktır.

7.Modülerlik: Kontrolcü sadece iki modülden oluşacaktır. Bunlar kod modülü ve kullanıcı modülüdür.

8.Belgeleme: Bu belgeden de anlaşılacağı üzere tam anlamıyla sistemin özeti olacak bu doküman oluşturulmuştur.

9.Kullanılabilirlik: Kullanılabilirlik açısından oldukça basit tasarlanmış, sade renklerle oluşturulmuştur.

10.Temizlik: Gerek arka tarafta gerek ön tarafta kalabalık durumlardan kaçınılarak en mümkün temizlik sağlanmıştır.

11.Değiştirilebilirlik: Mühendis istediği kod parçalarıyla oynayarak değiştirilebilirliği sağlayacaktır.

12.Esneklik: Proje farklı platformlarda çalışacağından gayet esnektir.

13.Genellik: Kontrolcü her yeterli sistemde kullanılabileceğinden geneldir.

14.Sınanabilirlik: Projenin geliştiricileri tarafından pilot kullanıcılarla sınanacaktır.

15.Taşınabilirlik: Kontrolcü internet üzerinden dağıtılabileceği için taşınabilirlik problemi olmayacaktır.

16.Birlikte Çalışabilirlik: Kontrolcünün çalıştığı bilgisayarlar birbirinden bağımsız olacağı için birlikte çalışabilirlik durumu söz konusu değildir.



**2.9 Kaynak Yönetim Planı**

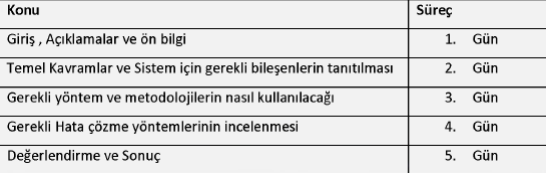
Mevcut bir kaynağımız olmadığından kaynak olarak sadece bu proje dokümantasyonu mevcuttur.

Bunun yanında Fırat Üniversitesi Tez ve Seminer Yazma Aracından MS WORD Tez Şablonu v1.3 baz alınarak mevcut sistem geliştirilmiştir.

**2.10 Eğitim Planı**

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

• Python



* Proje teslimi sonrası sistemi kullanacak kişilere proje kullanım olarak kılavuz yayınlanacaktır. Sistem kullanımı oldukça basit olduğundan seminere gerek duyulmamaktadır. Kılavuz içerisinde;

1. Sisteme genel bakış
2. Sisteme dosyanın nasıl yükleneceği
3. Sistemin kısıtları açıklanarak nasıl kullanıldığına dair bilgiler verilmektedir.

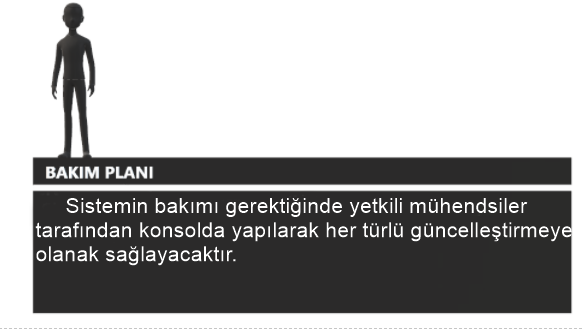
**2.11 Test Planı**

Proje test ekipleri ve görevleri şu şekildedir;

* Çeşitli platformlarda sistem uygunluk testi yapmak,
* Tezkont’un güncel tez kurallarını takip edip etmediğini kontrol etmek.

**2.12 Bakım Planı**

Projenin bakım planında tez yazım kurallarında bir değişiklik olduğu zaman bu değişiklikler bakım planı ile yapılır.



**2.13 Projede Kullanılan Yazılım/Donanım Araçlar**

* Visual Studio Code
* Python
* Gereksinimleri karşılayacak bilgisayar

**3. Sistem Çözümleme**

**3.1 Mevcut Sistem İncelemesi**

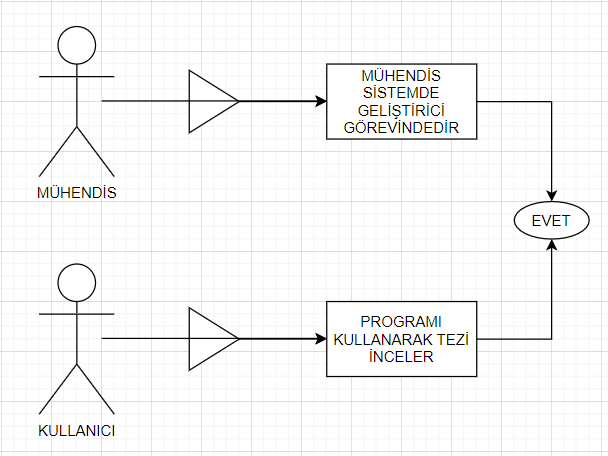
Sistem incelemesi olarak yapılacak projenin alanında az örneklerinin olması ve kapalı kaynak kodlu yazılmaları sistemin incelemesini oldukça zorlaştıracaktır. GemBox gibi kaynaklar incelenmiş sisteme nasıl katkı yapılacağı araştırılmıştır.

**3.1.1 Örgüt Yapısı**

Örgüt yapısı olarak Fırat Üniversitesi Yazılım Mühendisliği 3.Sınıf

öğrencilerinin ortak çalışmasıyla oluşan bir örgüt yapısı vardır.

**3.1.2 İşlevsel Model(Dfa Gelecek)**



**3.1.3 Var olan Yazılım/Donanım Kaynakları**

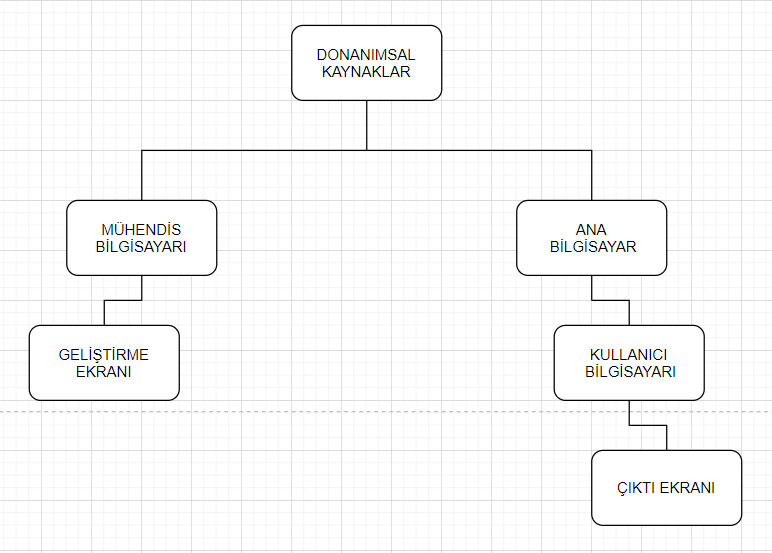
Yazılım Kaynakları

->Python

->Ms Word

Donanım Kaynakları

-> Kullanıcı Bilgisayarı



**3.1.4 Var olan S--istemin Değerlendirilmesi**

Herhangi bir ücret talep edilmeden kullanıcı, dosyasını girip dosyanın

kurallara uygun olup olmadığını kontrol edebilecek. Bu gibi avantajlar sunarak

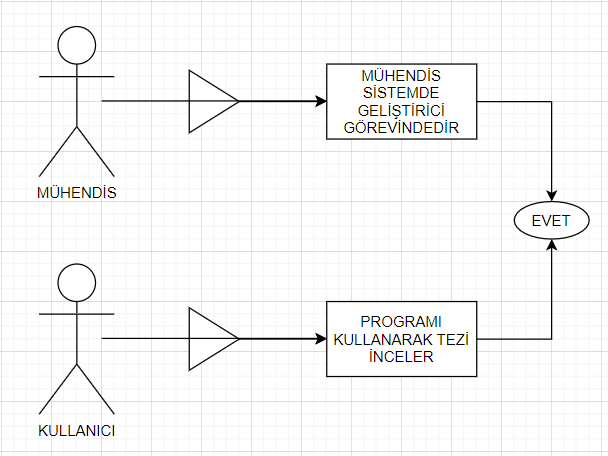
kullanıcın ücretsiz bir şekilde sistemden yararlanılması sağlanacak.

**3.2 Gereksenen Sistemin Mantıksal Modeli**

**3.2.1 Giriş**

Mevcut sistemler incelendiğinde kullanıcıların ilgili programlara oldukça yüksek fiyata ve sınırlı bir erişebiliyorlar. Mevcut sistemden daha performanslı derin metin okuma kullanarak ücretsiz bir şekilde erişime sunmak bize kalıyor.

**3.2.2 İşlevsel Model(Dfa gelecek)**



**3.2.3 Genel Bakış**

Genel hatlarıyla sistemi inceleyecek olursak mevcut intihal yönetim sisteminde mevcut olan tüm olaylar burada da var. Bunun yanı sıra olayın akış şekli USE-CASE diyagramında mevcuttur. Yani yüklenen dosya, veri, evrak hiçbir şekilde başkası tarafından erişilemez, alınamaz, görülemez ve işlem yapılamaz tamamen kullanıcının arayüzde yaptığı işlemlerin tümü orayla sınırlı kalıp hiçbir yerde veri tutulmaz.

**3.2.4 Bilgi Sistemleri/Nesneler**

KULLANICI: Yapının tek nesnesidir. Tez uygunluk gibi birçok istemde bulunur.

ARŞİV: Dosyaların elektronik veya herhangi bir ortamda saklanması olası değildir. Kullanıcı ihtiyaç duyduğu işlemleri yaptıktan sonra oturum sonlanınca veriler silinir.

**3.2.5 Veri Sözlüğü**

path: \*Girilen Tez Dosyanın İsmi\*

sayac: \*Dosyada ki tez satırları üzerinde ki işlemler\*

doc: \*dosya ismini "docx" kütüphanesinin entegresi\*

allText: \*Tez dosyasının içerikleri\*

pastext: \*Tez dosyasının içeriklerini belirli parametrelere göre ayrıştırılıp ham veri elde edilmesi\*

kaynakca\_indis: \*Düzenlenmiş dosyada kaynakça indis numarası\*

kaynakca\_DegerList: \*Kaynakça içerisindeki hataların liste olarak dönderilmesi\*

ArananKelimeList: \*Kelime arama fonksiyonu kullanıldığında ilgili aramanın satırlarının saklandığı değişken\*

Onsoz: \*Uygulamanın önsöz kısmını tutacak değişken\*

IndısEk: \*Kaynakça dışında ek bölümün başlangıç satırını saklayan değişken\*

**3.2.6 İşlevlerin Sıradüzeni(Dfa Gelecek)**

Kullanıcının kontrol etmek istediği doküman programa yüklenir ve program yaptığı analizler sonucunda çıktıyı ekrana yazdırır.

**3.2.7 Başarım Gerekleri**

Hali hazırda mevcut sistemler incelendi ve var olan sistemin eksiklerinden yola çıkılarak, sistemin başarısı için

-Tepki süresinin minimum olması

-Kullanım kolaylığı

-Anlaşılabilirlik

-Sistemin sonuç üretim doğrulukları ana gereklilikler olarak belirlenmiştir.

**3.3 Arayüz(Modül) Gerekleri**

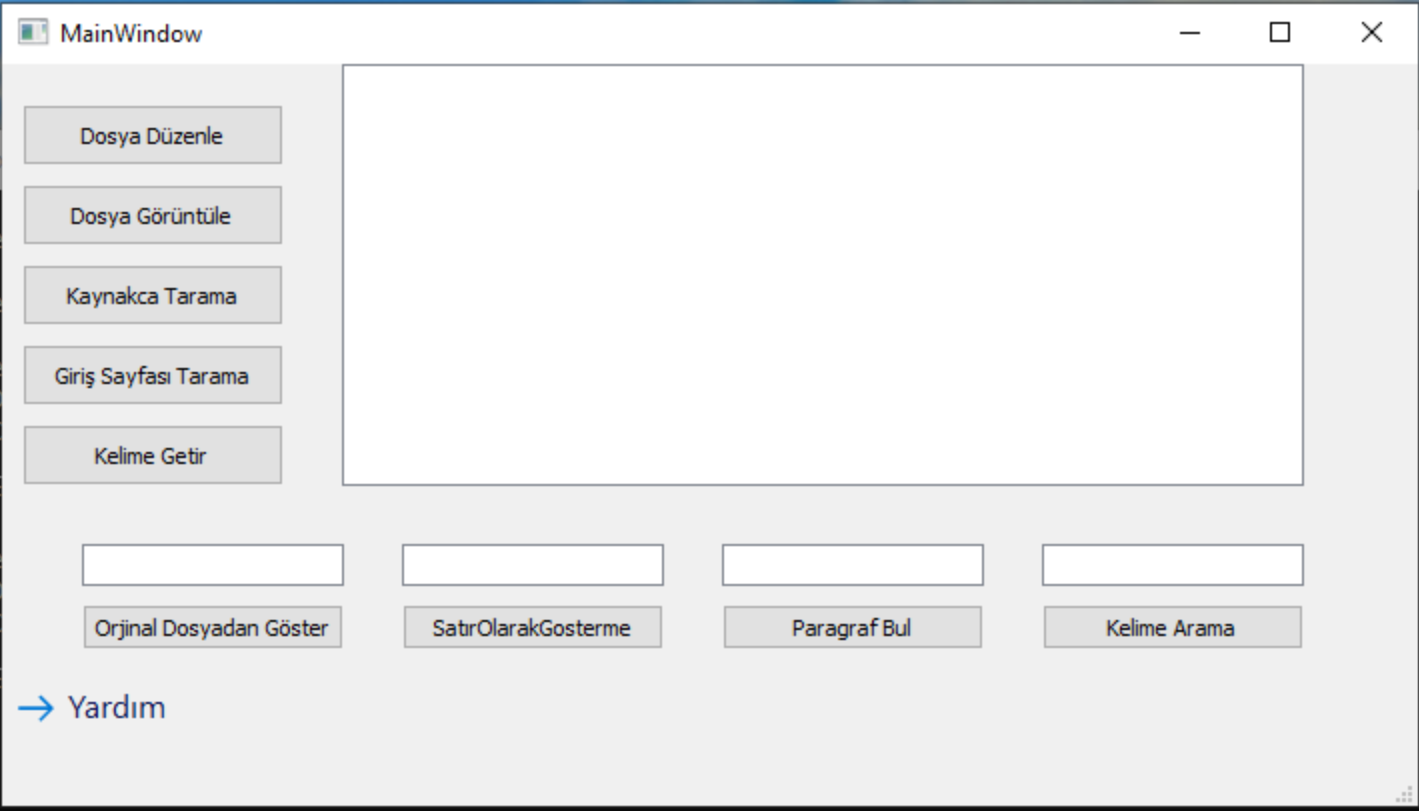
**3.3.1 Yazılım Arayüzü**

Projenin çalışması esnasında böyle bir açık verilmemesine özen gösterildi.

**3.3.2 Kullanıcı Arayüzü**

Kullanıcı arayüzü tasarlanırken sade ve düz bir arayüzü tasarlanılmasına

özen gösterildi.



**3.3.3 İletişim Arayüzü**

İletişim arayüzü bulunmamaktadır.

**3.3.4 Yönetim Arayüzü**

Yönetim arayüzü bulunmamaktadır.

**3.4 Belgeleme Gerekleri**

**3.4.1 Geliştirme Sürecinin Belgelenmesi**

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

**3.4.2 Eğitim Belgeleri**

Mevcut bir belgemiz bulunmamaktadır.

**3.4.3 Kullanıcı El Kitapları**

Bu kısma projenin en son safhasında kullanıcılara verilecek kullanım kılavuzu hazırlanacak. Yani proje sonunda rahat ve kolay kullanımdan dolayı bir eğitim semineri verilmeyecektir.

**4. Sistem Tasarımı**

**4.1 Genel Tasarım Bilgileri**

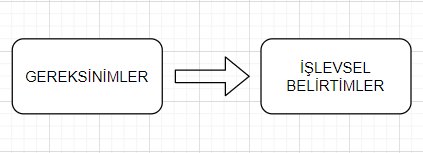
**4.1.1 Genel Sistem Tanımı**

-> Gereksinimler

Kullanıcı sisteme girerek analiz işlemini gerçekleştirecektir.

-> İşlevsel Belirtimler

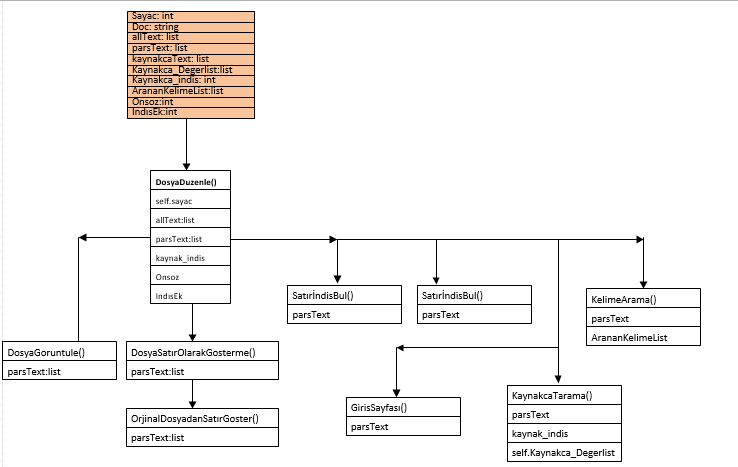
Sistemin ne yapacağı sorusuna cevap verelim. Sistem gelen belgeyi, sistemde tanımlı olan tez kurallarına göre inceleyip bunun sonucunu kullanıcıya verme üzerine kurulacak.



**4.1.2 Varsayımlar ve Kısıtlamalar**

Sistemde herhangi bir şekilde varsayım veya kısıtlamalar bulunmamaktadır.

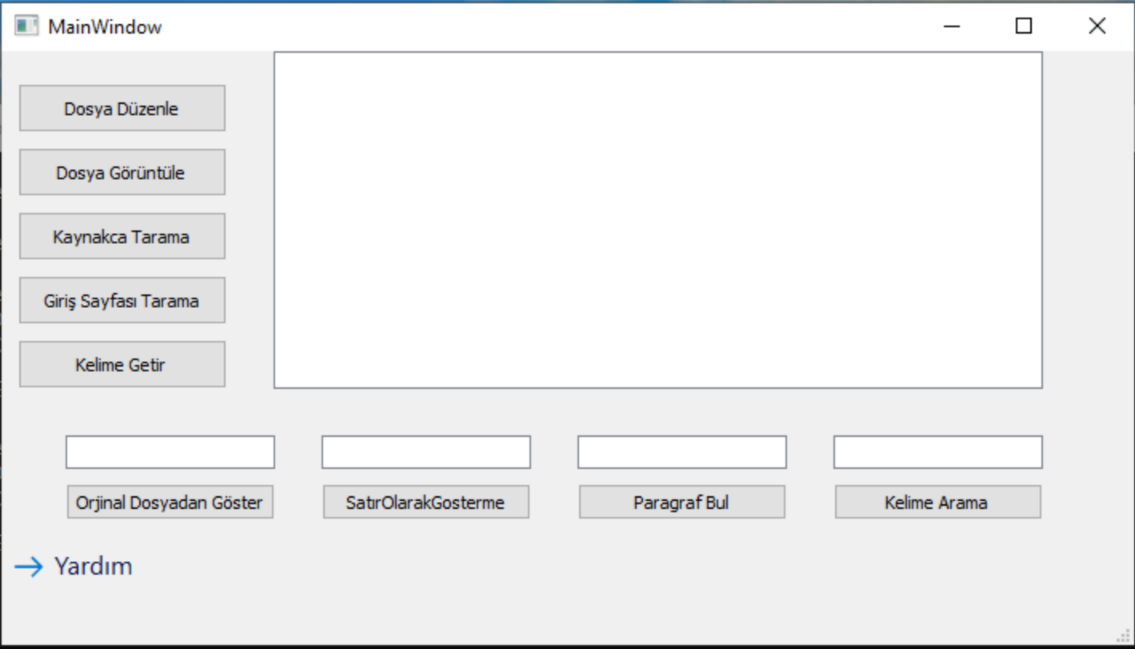
**4.1.3 Sistem Mimarisi**



**4.1.4 Dış Arabirimler**

**4.1.4.1 Kullanıcı Arabirimleri**

Kullanıcı arabirimi olarak tek bir kullanıcı arabirimi vardır. Kullanıcı dosyasını girerek dosya kontrolünü sağlamış olacaktır.



**4.1.4.2 Diğer Sistemlerle Arabirimler**

Şuan tam bilgi sahibi olunmadığından bu arabirim kullanılmayacaktır.

**4.1.5 Testler**

Bir pilot bölge seçilmeksizin sistem, genel hatlarıyla geliştiren ekip tarafından test edilip kullanıcıya sunulacaktır.

**4.1.6 Performans**

Sistemin performansını etkileyen faktörler eğer varsa tespit edilip test edildikten sonra test verileri değerlendirilecek.

**4.3 Süreç Tasarımı**

**4.3.1 Genel Tasarım**

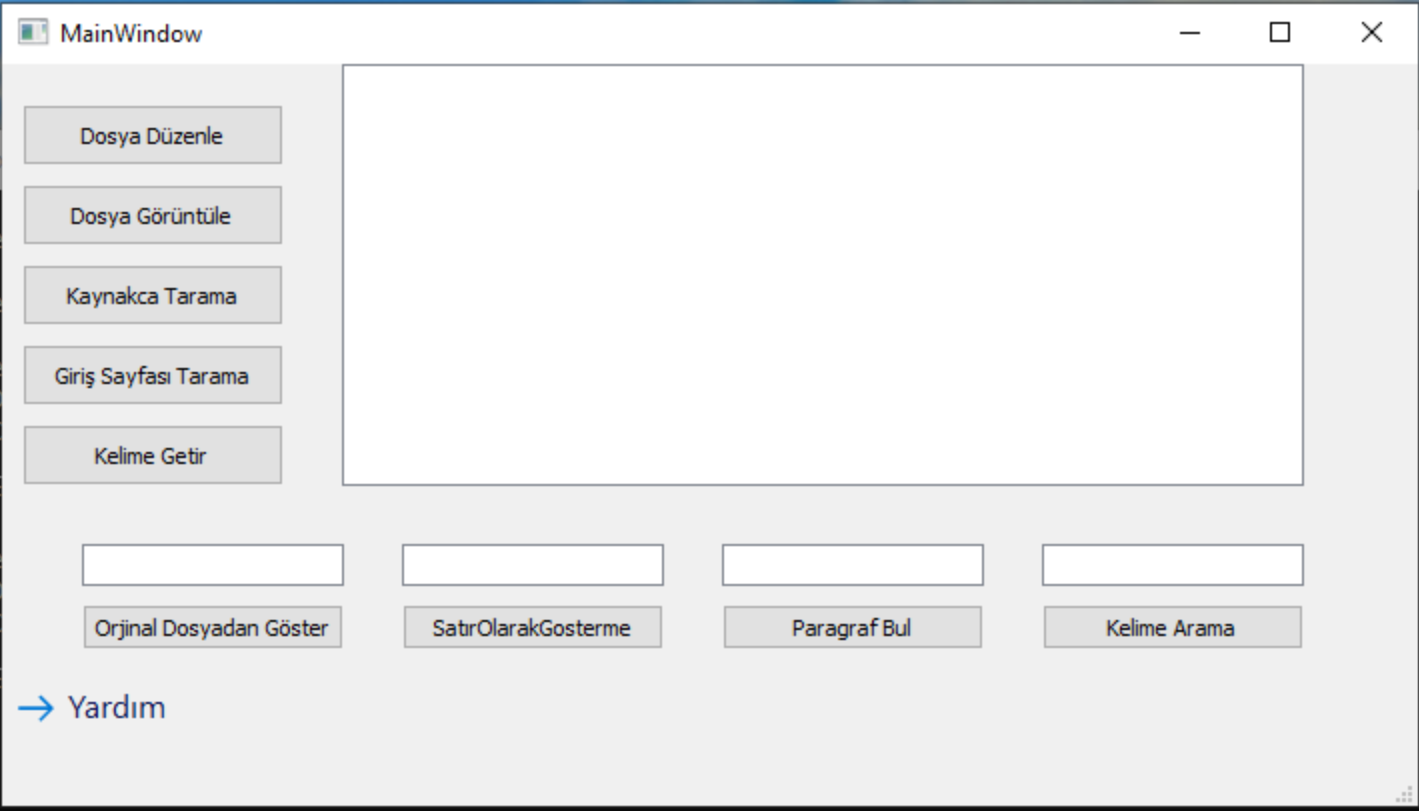
Genel olarak tasarımda sadece kullanıcı arayüzü oluşturduk.



**4.3.2 Modüller**

**4.3.2.1 Giriş Modülü**

**4.3.2.1.1 Kullanıcı Arabirimi**



**4.3.2.1.2 Modül Tanımı**

Kullanıcı girerek dosyanın doğrululuğunu kontrol eder.

**4.3.3 Kullanıcı Profilleri**

KULLANICI: Yapının tek nesnesidir. Tez uygunluk gibi birçok istemde bulunur.

ARŞİV: Dosyaların elektronik veya herhangi bir ortamda saklanması olası değildir. Kullanıcı ihtiyaç duyduğu işlemleri yaptıktan sonra oturum sonlanınca veriler silinir.

**4.3.4 Entegrasyon ve Test Gereksinimleri**

Sistemimiz daha öncede belirttiğimiz gibi kabul görülen tez kurallarıyla entegrasyon halinde olması gerekir. Bunun yanı sıra test süresince geliştiren ekibin test sürecinin başında olmalı ve testi gözlemleyip notlar çıkartmalı.

**4.4 Ortak Alt Sistemlerin Tasarımı**

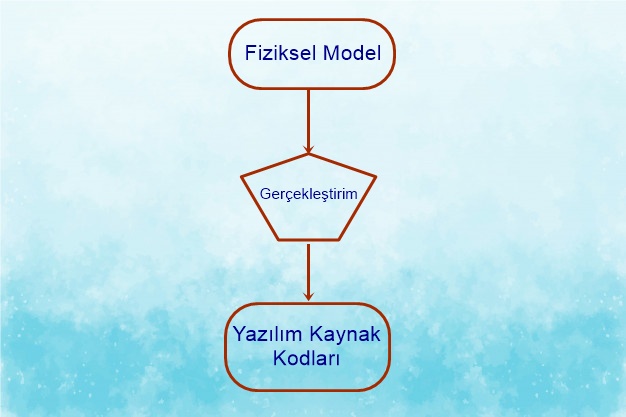
**4.4.1 Ortak Alt Sistemler**

Kullanılacak ortak bir alt sistem bulunmamaktadır.

**5. Sistem Gerçekleştirimi**

**5.1 Giriş**

Gerçekleştirim çalışması, fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmasını içerir. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.



**5.2 Yazılım Geliştirme Ortamları**

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

**->** Programlama Dili

**->** Hazır Program Kitapçıkları

CASE Araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

**5.2.1 Programlama Dilleri**

Sistemde kullanılan programlama dili Pyhton’dur.

**5.3 Kodlama Stili**

Nesne Tabanlı Programlama stiline göre bir kodlama yapıldı.

**5.3.1 Açıklama Satırları**

Gerekli yerlere açıklama satırları yazıldı.

**5.3.2 Kod Biçimlemesi**

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

**5.3.3 Anlamlı İsimlendirme**

Sistem kodlamasının genel yapısında kullanılan belli değişkenlerin isim tanımlanması yapılırken anlamlı isimlendirme yapıldı.

**5.3.4 Yapısal Programlama Yapıları**

**->** Ardışık işlem yapıları: Bu tür yapılarda genellikle fonksiyon,

altprogram ve buna benzer tekrarlı yapıları tek bir seferde

çözdük.

**->** Koşullu işlem yapıları: Bu yapıları ise neredeyse

programın tamamında kullandık karşılaştırma yapılan her

yerde bunlara yer verildi.

**->** Döngü yapıları: Tıpkı ardışık işlemler gibi alt alta birkaç satır

yazıcığımıza tek bir döngüyle bu sorunların üstesinden geldik.

**5.4 Olağan Dışı Durum Çözümleme**

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

**5.4.1 Olağandışı Durum Tanımları**

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

**5.4.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları**

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

**5.5 Kod Gözden Geçirme**

**5.5.1 Gözden Geçirme Sürecinin Düzenlenmesi**

Gözden geçirme sürecinin temel özellikleri;

**->** Hataların bulunması, ancak düzeltilmemesi hedeflenir,

**->** Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyici kullanılmasıdır. Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımı yapacak ekipten seçilmesinde yarar vardır.

**->** Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır. biçiminde özetlenebilir. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir.

**5.5.2 Gözden Geçirme Sırasında Kullanılacak Sorular**

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

**5.5.2.1 Öbek Arayüzü**

Oluşturduğumuz öbekleri test etmek için belli sorular sorduk bu sorular:

**->** Her öbek tek bir işlevsel amacı yerine getiriyor mu?

**->** Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?

**->** Öbek tek giriş ve tek çıkışlı mı?

**->** Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu? şeklinde oldu.

**5.5.2.2 Giriş Açıklamaları**

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

**->** Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?

**->** Giriş açıklama satırları, öbeğin amacını açıklıyor mu?

**->** Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?

**->** Giriş açıklama satırları, çıktıları (parametre, kütük vb) ve hata iletilerini tanımlıyor mu?

**->** Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?

**->** Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?

**->** Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?

**->** Giriş açıklama satırları, Öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?

**->** Her paragrafı açıklayan kısa açıklamalar var mı?

şeklinde oldu.

**5.5.2.3 Veri Kullanımı**

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

**->** İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?

**->** Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı?

**->** Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?

**->** Her değişken tek bir amaçla mı kullanılıyor?

**->** Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?

**->** Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

**5.5.2.4 Öbeğin Düzenlenişi**

**->** Modüller birleşimi uyumlumu?

**->** Modüller arası veri aktarımları sağlanıyor mu?

**->** Bütün modüller birleştiğinde sistem çalışıyor mu?

Gözden geçirme sırasında referans alınacak sorular olacaktır.

**5.5.2.5 Sunuş**

Artık son kısma gelindiğinde ise şu sorular soruldu:

**->** Her satır, en fazla bir deyim içeriyor mu?

**->** Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?

**->** Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?

**->** Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?

**->** Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?

**->** Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

**6. Doğrulama ve Geçerleme**

**6.1 Giriş**

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

**->** Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.

**->** Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.

**->** Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.

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



**6.2 Sınama Kavramları**

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

**6.3 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.

**6.3.1 Beyaz Kutu Sınaması**

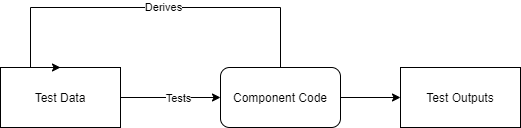
Denetimler arasında:

**->** Bütün bağımsız yolların en azından bir kere sınanması,

**->** Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,

**->** Bütün döngülerin sınır değerlerinde sınanması,

**->** İç veri yapılarının denenmesi yapıldı.



**6.3.2 Temel Yollar Sınaması**

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

**->** Eş değerlere bölme

**->** Uç değerler analizi

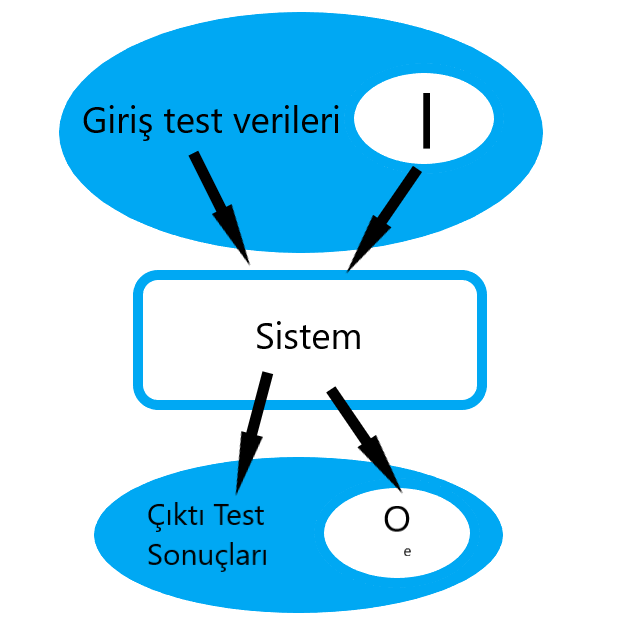
**->** Karar tablosu

**->** Sonlu durum makinesi

**->** Belgelenmiş özelliklere göre test

**->** Rastgele test

**->** Kullanım profili



**6.5 Sınama ve Bütünleştirme Stratejileri**

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

**6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme**

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

**6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme**

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

**6.6 Sınama Planlaması**

Bir tablo ile özetlemek gerekirse şu şekilde özetleyebiliriz.

Test raporu hazırlanırken şu özellikler mutlaka planda belirtilmelidir;

**Test planı kimliği:** Test planının adı veya belge numarası

**Giriş:** Test edilecek yazılımın elemanlarının genel tanıtım özetleri. Ayrıca bu plan kapsamı ve başvurulan belgeler. Kısaltmalar ve terim açıklamaları bu bölümde bildirilmelidir.

**Test edilecek sistem:** Sistemde bileşenleri sürüm sayıları olarak sıralar ve sistemin özelliklerini bileşenlerini ve nasıl kullanıldıkları açıklanmalıdır. Ayrıca sistemde test edilmeyecek parçalar belirtilmelidir.

**Test edilecek ana fonksiyonlar:** Sistemin test edilecek ana fonksiyonlarının kısa bir tanıtımı yapılmalıdır.

Test edilmeyecek ana fonksiyonlar: Sistemde test edilmeyecek fonksiyonları ve bunların neden test edilmedikleri açıklanacaktır.

**Geçti/Kaldı Kriterleri:** Bir test sonucunda sistemin geçmiş veya kalmış sayılacağını açıklanmalıdır.

**Test dokümanı:** Test süresince yapılan işlemleri alınan raporları elde edilen bilgileri rapor içinde sunulmalıdır. Sorumluluklar: Hangi kişilerin nelerden sorumlu olduğu ve test takım lideri bilgileri mutlaka raporda belirtilmelidir.

**Riskler ve Önlemler:** Test planında varsayılan ve olası yüksek riskli durumları belirtir ve bu durumların olması durumunda, etkilerinin en aza indirilebilmesi için alınması gereken 79 önlemleri açıklar.

**6.7 Sınama Belirtimleri**

Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir. Bu ayrıntılar temel olarak:

**->** Sınama türü, stratejisi(beyaz kutu vb.)

**->** Sınama verileri

**->** Sınama senaryoları

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 belirtimlerinin hazırlanmasındaki temel maç, etkin sınama yapılması için bir rehber oluşturmasıdır.

**6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

Bütün bu etkinlikleri bir hiyerarşi altında incelemek gerekirse:

**->** Planlama aşamasında genel planlama sınaması gerçekleştirilir. Bu olan tüm planların basit bir ön hazırlığı niteliğindedir.

**->** Çözümleme aşamasında sınama planı alt sistemler bazında ayrıntılandırılır.

**->** Tasarım aşamasında sınama plana detaylandırılır ve sınama belirtimleri oluşturulur. Bu oluşumlar daha sonra eğitim ve el kitabında kullanılır.

**->** Gerçekleştirim aşamasında teknik sınamalar yapılır sınama raporları hazırlanır ve elle tutulur ilk testler yapılır.

**->** Kurulum aşamasında sistemle ilgili son sınamalar yapılır ve sınama raporları hazırlanır.

**7. Bakım**

**7.1 Giriş**

Sistemin tasarımı bittikten sonra artık tez kuralları değiştikçe sistemin bakıma sokulması gerekir. Daha öncede belirttiğimiz gibi sistem tez kurallarına göre çalıştığından eğer herhangi bir değişiklik olursa bakım yapılması gerekmektedir.

**7.2 Kurulum**

Kurumlum oldukça basit şekilde tasarlanmıştır. Konuya en uzak kullanıcılar dahi herhangi bir zorlukla karşılaşmadan kurulumu sağlayabilecektir. Kurulum videosu eklenecektir.

**7.3 Yerinde Destek Organizasyonu**

Yerinde destek organizasyonu söz konusu olmayacaktır.

**7.4 Yazılım Bakımı**

**7.4.1 Tanım**

Düzeltici Bakım: Teorik olarak bir yazılımın tümüyle sınanabilmesi olası olsa bile, pratikte bu sağlanamaz. Bu nedenle çalışan yazılımda her an hata ile karşılaşma olasılığı vardır. Bu nedenle zaman zaman çalışan yazılımda ortaya çıkan hataların düzeltilmesi gerekir. Bu tür düzeltme çalışmaları “düzeltici bakım” olarak adlandırılır.

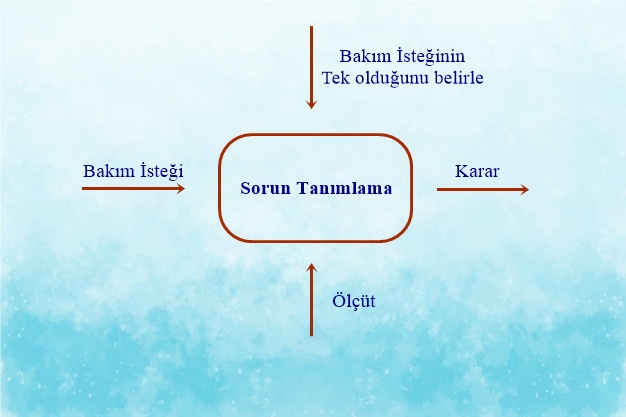
Uyarlayıcı Bakım: Uyguluma yazılımları, işletme ya da kuruluşların günlük yaşamlarında yaptıkları işleri bilgisayar ortamında yapmalarını sağlayan ağaçlardır. Her kuruluş ya da işletme canlı bir varlık düşünülebilir. Hiçbir işletme durağan değildir. Süreç içinde değişiklik gösterir. İşletme ya da kuruluşlarda değişiklik, yapılan işlerin yapılma tarzlarının değişmesi, yeni iş türlerinin ortaya çıkması biçiminde kendini gösterebilir. İşletme ya da kuruluşlarda yaşanan değişikliklerin, o kuruluşun işlerini bilgisayar yardımı ile yapmalarını sağlayan uygulama yazılımlarına da yansıtılması gerekir. Bu yansıtma işlemi “uyarlayıcı bakım” olarak adlandırılır.

**7.4.2 Bakım Süreç Modeli**

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

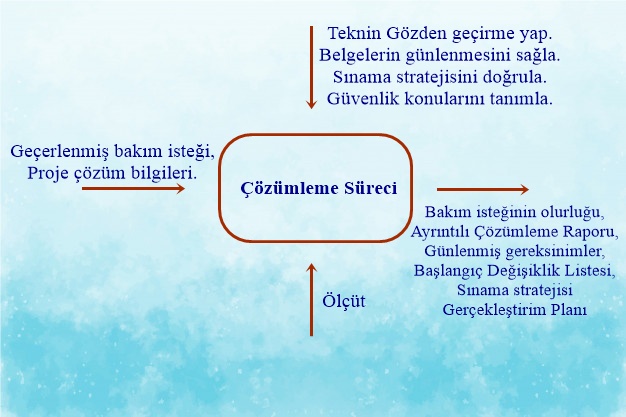
**1. Adım: Sorunu Tanımlama Süreci**

İlk önce bakım ne için yapılıyor sorun ne buna bir bakalım.



**2. Adım: Çözümleme Süreci**

Sorun tanımlamadan çıkan karar doğrultusunda problemi kâğıt üzerinde çözelim



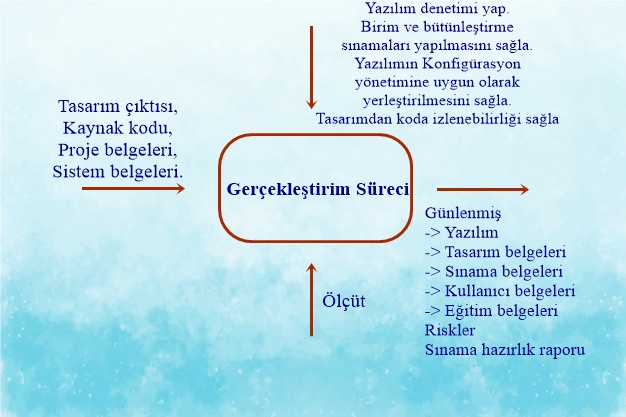
**3. Adım: Tasarım Süreci**

Çözümlenen sistem sonucunda tasarımı güncelleştirmeye geldi sıra.



**4. Adım: Gerçekleştirim Süreci**

Tasarımı yapılan sistemin gerçekleştirmesine sıra geldi



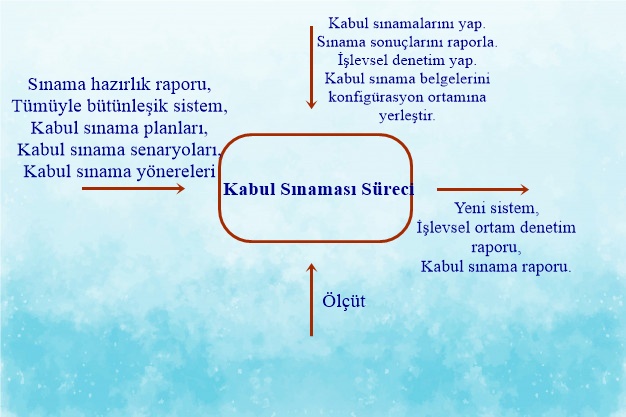
**5. Adım: Sistem Sınama Süreci**

Artık tekrardan tasarlanan sistemin sınama sürecini tekrar ele almak gerekiyor.



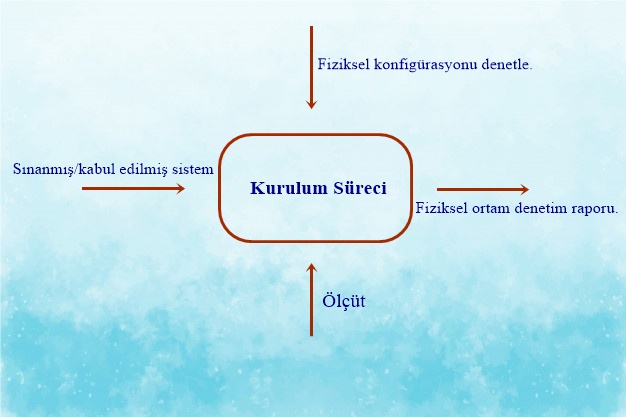
**6. Adım: Kabul Sınaması Süreci**

Kendi içimizde sınadığımız sistemi birde müşteri karşısında sınıyoruz.



**7. Adım: Kurulum Süreci**

Kabul sınamasını geçen sistemimiz artık tekrardan kurulum aşamasına geçiyor.



**8. Sonuç**

Sonuç olarak sistem hayata geçirildiği zaman neler değişeceği gözler önüne serdik. Bunun yanı sıra basit olan bu sistemle ek masraflar ortadan kalkacak.

Kullanıcı, çok karmaşık olmayan ve ücretsiz olacak bu sistem sayesinde artık eskisi kadar bir maliyet ödememiş olacak ve herkes bu sisteme minnettar kalacak.

**9. Kaynaklar**

[**https://e-bergi.com/y/cevik-modelleme-ve-cevik-yazilim-gelistirme/**](https://e-bergi.com/y/cevik-modelleme-ve-cevik-yazilim-gelistirme/)

[**http://web.firat.edu.tr/mbaykara/ORNEKPROJEDOKUMANI.pdf**](http://web.firat.edu.tr/mbaykara/ORNEKPROJEDOKUMANI.pdf)

[**http://fbe.firat.edu.tr/sites/fbe.firat.edu.tr/files/FBE%20Tez%20Yazım%20Ver1.3.pdf**](http://fbe.firat.edu.tr/sites/fbe.firat.edu.tr/files/FBE%20Tez%20Yazım%20Ver1.3.pdf)

[**https://www.freepik.com/free-vector/watercolor-background\_6229778.htm#page=1&query=blue%20background&position=1**](https://www.freepik.com/free-vector/watercolor-background_6229778.htm#page=1&query=blue%20background&position=1)

[**https://softgainz.com/Development-Methodology.html**](https://softgainz.com/Development-Methodology.html)

[**https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.omurorman.com%2F&psig=AOvVaw2N2UlHaX\_2fR1a2Fg0MV9P&ust=1609876640952000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNiio4KIg-4CFQAAAAAdAAAAABAJ**](https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.omurorman.com%2F&psig=AOvVaw2N2UlHaX_2fR1a2Fg0MV9P&ust=1609876640952000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNiio4KIg-4CFQAAAAAdAAAAABAJ)

**https://www.clipartmax.com/max/m2H7H7G6G6G6H7A0/**

[**https://iconarchive.com/show/people-icons-by-aha-soft/user-icon.html**](https://iconarchive.com/show/people-icons-by-aha-soft/user-icon.html)

[**https://www.cleanpng.com/png-computer-icons-document-clip-art-4168246/**](https://www.cleanpng.com/png-computer-icons-document-clip-art-4168246/)

**https://www.subpng.com/png-wypw3k/download.html**

**https://www.cleanpng.com/png-computer-icons-operating-systems-symbol-linux-3986000/download-png.html**