**T.C.**

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Proje Dokümantasyonu**

**(Tez Hazırlama Programı)**

**Proje Ekibi**

**Hamza OBAN 175542006**

**Aydın YAĞIZ 175542001**

**Alim Ahmet AKSAKAL 170542002**

**Muhammed Raşit YILMAZ 180541022**

**Hasan ÇAKMAKÇI 170542019**

**Tarih – Versiyon**

**08.01.2021-1**

|  |
| --- |
| **1 GİRİŞ** |
| * 1. Projenin Amacı   2. Projenin Kapsamı   3. Tanımlamalar ve Kısaltmalar |
| **2 PROJE PLANI** |
| * 1. Giriş   2. Projenin Plan Kapsamı   3. Proje Zaman-İş Planı   4. Proje Ekip Yapısı   5. Kullanılan Özel Geliştirme Araçları   6. Test Planı   7. Bakım Planı   8. Projede Kullanılan Yazılım/Donanım Araçlar |
| **3 SİSTEM ÇÖZÜMLEME** |
| * 1. **Mevcut Sistem İncelemesi**      1. İşlevsel Model      2. Use-Case Modeli      3. Varolan Yazılım/Donanım Kaynakları      4. Varolan Sistemin Değerlendirilmesi   2. **Arayüz (Modül) Gerekleri**      1. Yazılım Arayüzü      2. Kullanıcı Arayüzü      3. İletişim Arayüzü      4. Yönetim Arayüzü |
| **4 SİSTEM TASARIMI** |
| * 1. **Genel Tasarım Bilgileri**       1. Genel Sistem Tanımı      2. Varsayımlar ve Kısıtlamalar      3. Veri Modeli      4. Testler      5. Performans   2. **Süreç Tasarımı**      1. Modüller         1. İstatistik Modülü            1. İşlev            2. Modül Tanımı      2. Kullanıcı Profilleri      3. Entegrasyon ve Test Gereksinimleri |
| **5 SİSTEM GERÇEKLEŞTİRİMİ** |
| * 1. Giriş   2. Yazılım Geliştirme Ortamları      1. Programlama Dilleri   3. Kodlama Stili      1. Açıklama Satırları      2. Kod Biçimlemesi      3. Anlamlı İsimlendirme      4. Yapısal Programlama Yapıları   4. Program Karmaşıklığı   5. Olağan Dışı Durum Çözümleme      1. Olağandışı Durum Tanımları   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 |
| **6 DOĞRULAMA VE GEÇERLEME** |
| * 1. Giriş   2. Sınama Kavramları   3. Doğrulama ve Geçerleme Yaşam Döngüsü   4. Sınama Yöntemleri      1. Beyaz Kutu Sınaması      2. Kara Kutu Sınaması      3. 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 |
| **7 BAKIM** |
| * 1. Giriş   2. Kurulum   3. Yerinde Destek Organizasyonu   4. Yazılım Bakımı      1. Tanım      2. Bakım Süreç Modeli |
| **8 SONUÇ** |
| **9 KAYNAKLAR** |

1. **Giriş**
   1. **Projenin Amacı**

Yazılı metinlerin Türkçe dil bilgisi kuralarına uygun olmasını sağlama ve metinlerin akıcılığını ve okunabilirliğini artırıcı öneriler sunmaktır. Türkçe metinlerin biçimsel olarak incelenmesi, yazım yanlışlarının bulunması, cümlelerin belirlenmesi, anlam çıkarımı yapılması ve bu yollarla yazının iyileştirilerek okunabilirliğini artırıcı öneriler sunulması amaçlanmıştır.

* 1. **Projenin Kapsamı**

Proje kapsamında uygulama alanı olarak Üniversiteler baz alınmıştır. Bu sayede proje kapsamında;

* Özel Üniversiteleri
* Devlet Üniversiteleri
  1. **Tanımlamalar Ve Kısaltmalar**

Tanımlamalar burda olucak…

1. **PROJE PLANI**
   1. **Giriş**

Tez hazırlama programımızda projemiz için yazılım dili olarak c# programlama dili ile analizi test etmek için c# ‘ın form ekranı ile bir arayüz hazırladık. Metnin kelime sayısı, cümle sayısı, karakter sayısı gibi istatistiklerini bulur. Gereksinim olarak ağırlıklı olarak projemizde IO ve iTextSharp sınıflarından yararlanılmıştır.

* 1. **Projenin Plan Kapsamı**

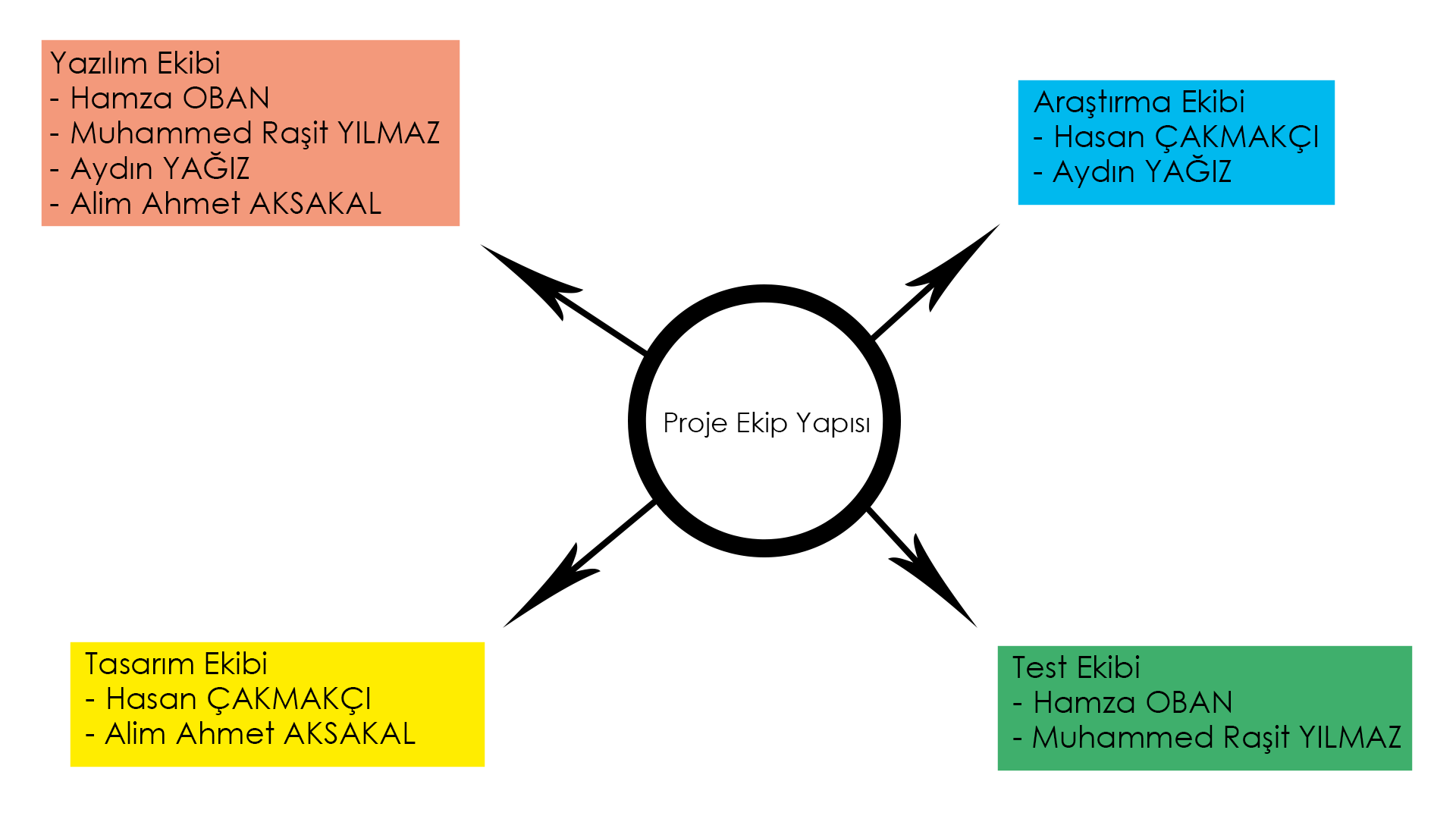
Projemizin kapsamı cümle analizi, yazım denetimi, istatistik çıkarma gibi özellikleri bulunmaktadır. Yazım yanlışlarını, noktalama işareti hatalı kullanımlarını kontrol eder, yazım yanlışı yapılmış kelimeler için doğru olabilecek kelime fikri önerir, Türkçe olmayan kelimeler yerine Türkçe kelime önerisi yapar, metnin okunabilirliğinin ve akıcılığının artırılmasını sağlamak amacıyla metin içindeki cümleleri analiz ederek kullanıcıya istatistik sunar ve uzun yazılmış birleşik cümlelerin bölünebilmesini önerir/sağlar.

* 1. **Proje Zaman-İş Planı**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Yazılım Yaşam Döngüsü Adımları | Haftalar   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| Planlama | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **X** | **X** |  |  |  |  |  |  |  |  | |
| Çözümleme | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **X** | **X** |  |  |  |  |  |  |  | |
| Tasarım | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **X** | **X** | **X** |  |  |  |  |  | |
| Gerçekleştirim | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  | **X** | **X** | **X** | **X** |  |  |  | |
| Test | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | **X** | **X** |  | |
| Bakım | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  | **X** | |



* 1. **Proje Ekip Yapısı**

****

* 1. **Kullanılan Özel Geliştirme Araçları**

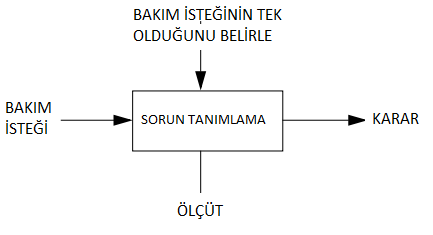
Tez Kontrol programı için kullanılan araçlar ve kütüphanerler;

* iTextSharp.5.5.13.2
* AngleSharp.0.9.11
* DevExpress
* Style Kütüphanesi
* System.IO
* System.Collections
* System.Linq
  1. **Test Planı**

Test planları öncelikle fikirlerin toparlanmasına ve güncellemelerin yönetilmesine yardımcı olur ve iletişim kurmaya yarar. Bu nedenle iyi bir test planı:

* Tez kurallarının kontrolleri
* Tez istatistiklerinin doğruluğu,
* Tez analizlerinin doğruluğu,
* Kelime kontrolleri,
* Kopya İçerik Kontrolü,
* Test için gerekli yazılım/donanımı,
* Zaman planı gibi bilgileri içermelidir,
* Kısa ve amaca odaklanmış olmalıdır.
  1. **Bakım Planı**

Temel yazılım bakım sorunları hem yönetimsel hem de tekniktir. Temel yönetim konuları şunlardır: müşteri öncelikleriyle uyum, kullanışlılık, hangi kuruluşun bakım yaptığı, maliyetleri tahmin etme. Temel teknik konular şunlardır: sınırlı anlayış, tez analizi , test etme, sürdürülebilirlik ölçümü. Bakım planları öncelikle fikirlerin toparlanmasına ve güncellemeler sonucu daha iyi sonuçlar üretebilmesi için sistemin onarılması ve geliştirilmesinde yardımcı olur ve iletişim kurmaya yarar. Bu nedenle iyi bir bakım planı:

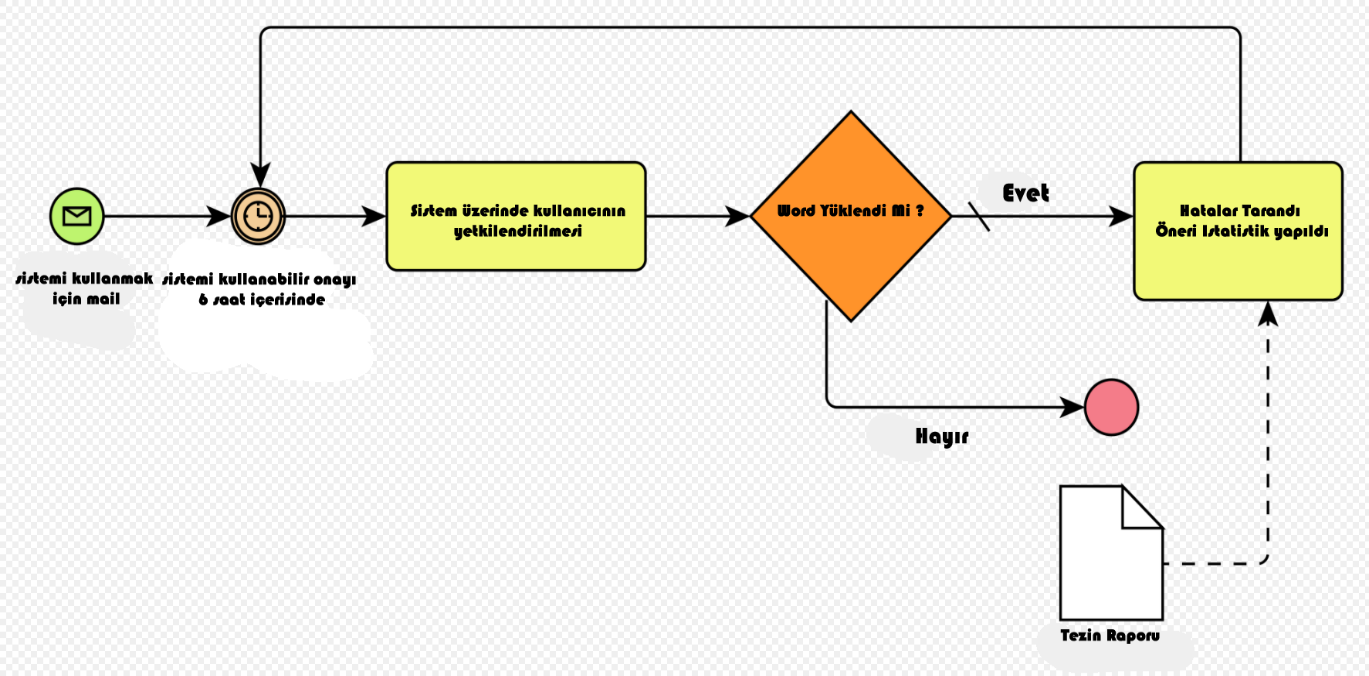


* 1. **Proje Kullanılan Yazılım/Donanım Araçları**

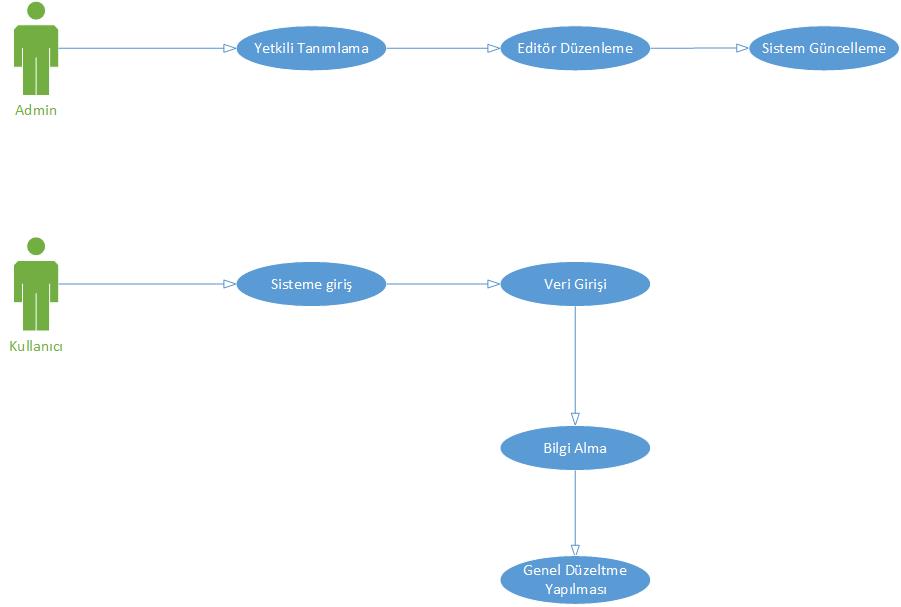
Projede geliştirilmesinde seçilen yazılım dili olarak c# belirlenmiş aynı zamanda c# tasarımı geliştirilmiş arayüz ile programa arayüz tasarımı sağlanacaktır. Donanım aracı olarak projede bir ihtiyaca gerek duyulmamıştır.

1. **Sistem Çözümleme**
   1. **Mevcut Sistemin İncelenmesi**
      1. İşlevsel Model

İşlev Modülü şekilde ki gibi işlenmektedir. Sistemi kullanmak isteyen kullanıcı mail atarak ulaşacaktır sistemden faydalanmak isteyen kullanıcıya 6 saat içerisinde olumlu geri dönüş yapılarak sistem üzerinde yetki verilir. Kullanıcı yüklediği dosyanın analizini hatalarını önerileri yüklediği Word dosyası üzerinden görebilir.



* + 1. Use Case Modeli



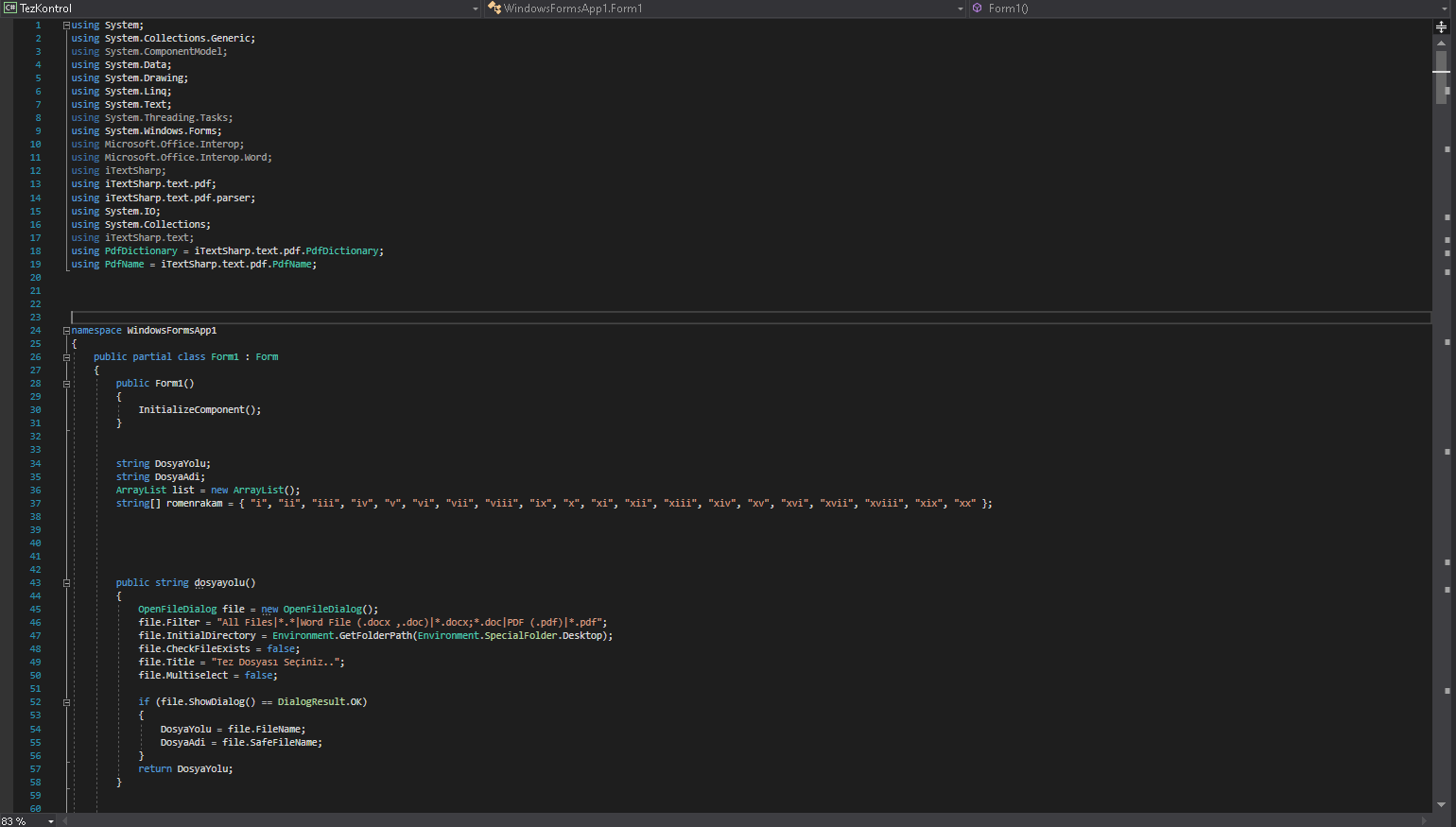
* + 1. Varolan Yazılım/Donanım Kaynaklar

Programın işlenebilmesi için var olan yazılım donanım kaynakları;  
**Yazılım** : c# yazılım dili ve arayüz modüller,Visio, Microsoft Office programları.

**Donanım** Araçları : monitör (ekran), kasa(notebook), modem, mause (fare), klavyedir.

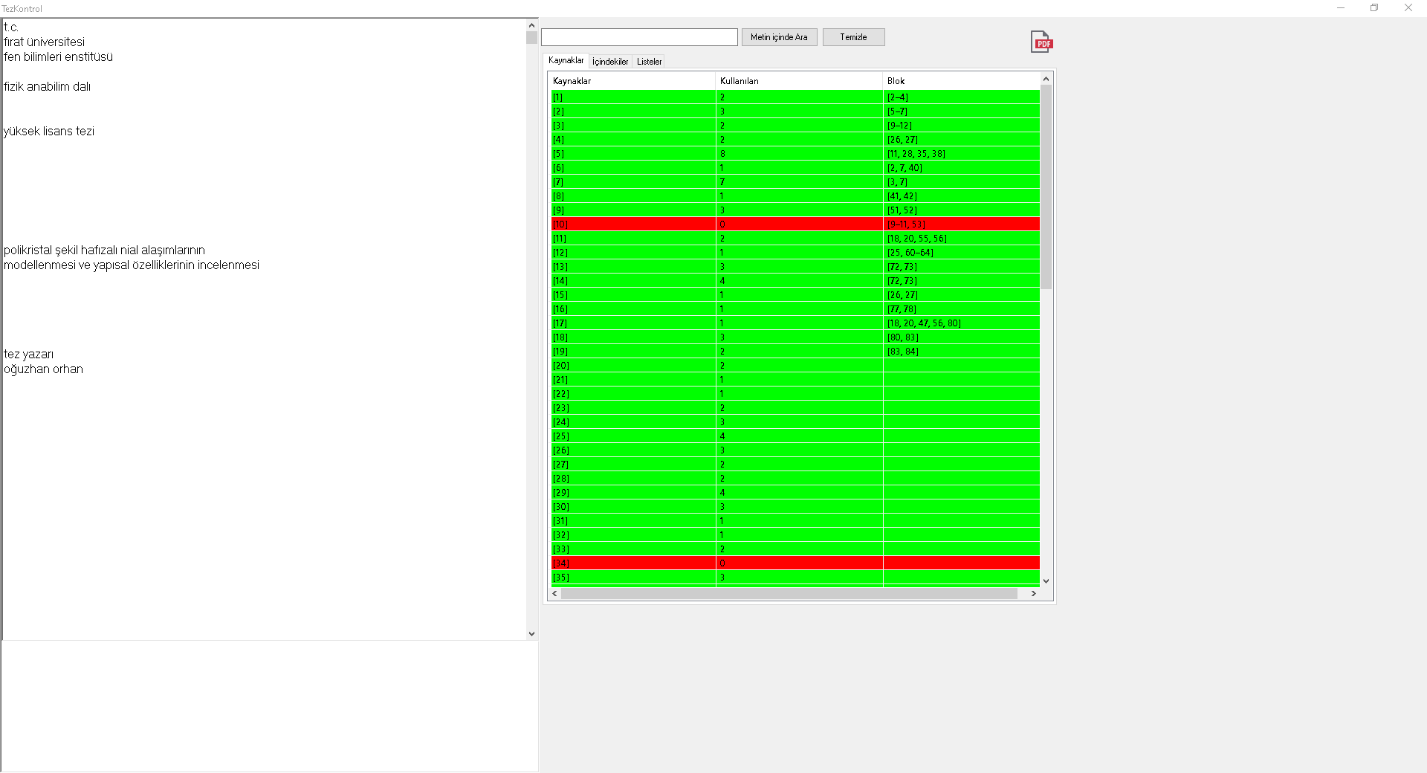
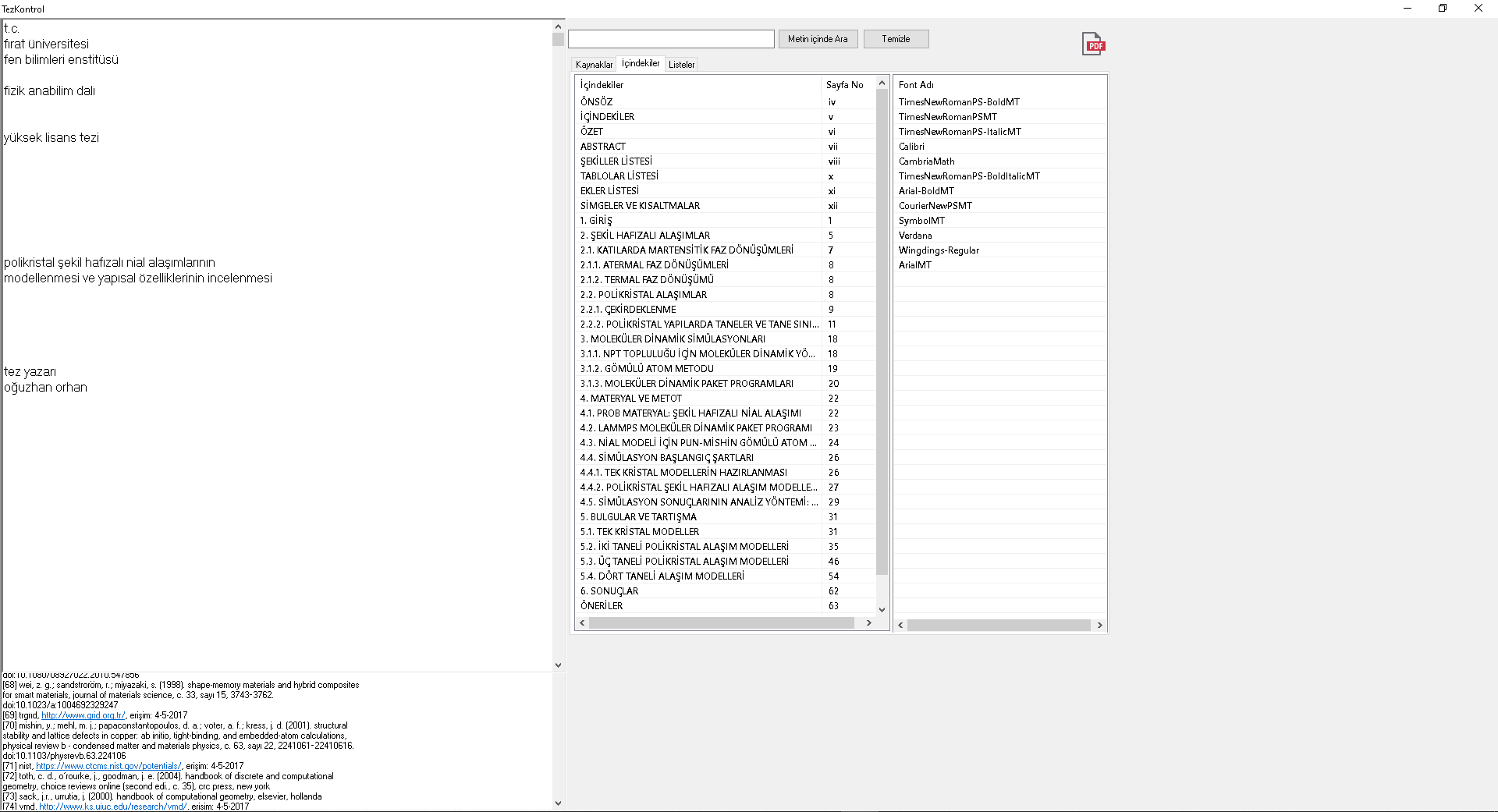
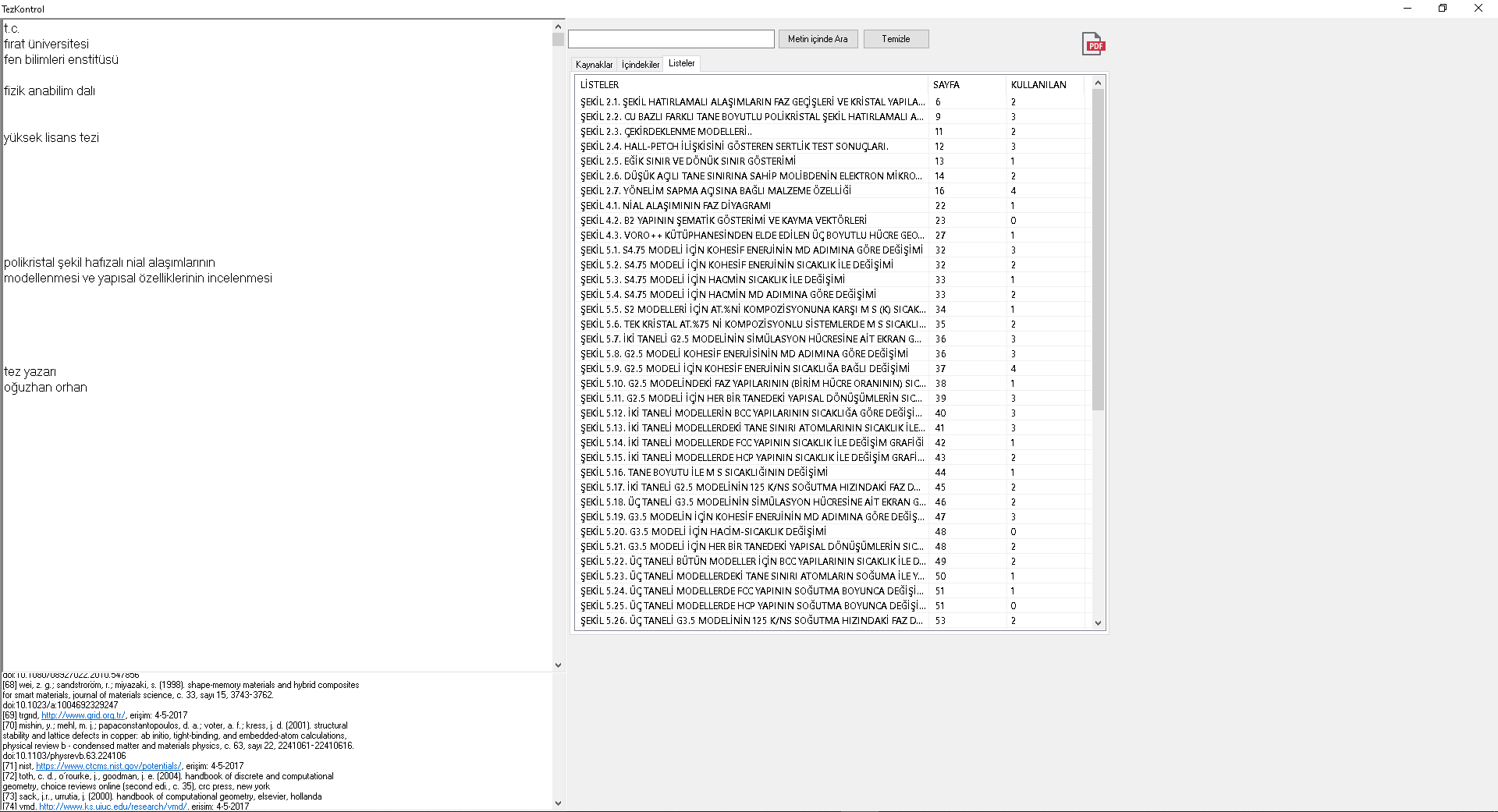
* + 1. Varolan Sistemin Değerlendirilmesi
  1. **Arayüz (Modül) Gerekleri**
     1. Yazılım Arayüzü

Projenin çalışması esnasında böyle bir açık verilmemesine özen gösterildi. Olabildiğince kelime haznesi yüksek tutularak her türlü hata ayıklamasına yer verilmeye çalışıldı. Gerekli olan her türlü değişiklik seurce kodları üzerinden yapılıp tekrar derlenecek.



* + 1. Kullanıcı Arayüzü

Projede kullanılan arayüzün sade ve anlaşılır olmasına özen gösterildi. Fontlar, renkler tamamen birbirleriyle uyumlu olmasına dikkat edildi. Kullanıcıların zorlanmadan uygulamayı kullanabilmesi amaçlanmıştır.



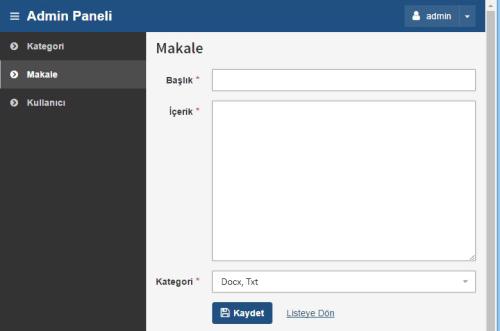
* + 1. İletişim Arayüzü

İletişim modülümüzde üniversiteler ile adminlerimiz arasında iletişimi sağlamak için sade ve kullanışlı bir arayüz yapılmıştır.



* + 1. Yönetim Arayüzü

Projenin %90 ı genelde yönetimsel arayüzlerden oluşacak sadece personelin göreceği ekran kullanıcı arayüzüne girecek dışındaki her yer yönetimsel arayüz olacak. Yöneticinin ekrana girdiğinde karşılaşacağı arayüzdür.İki arayüz vardır:Biri Kullanıcı Arayüzü diğeri ise yönetimsel arayüzdür.Yani bu arayüz admin tarafından sisteme girişlerde gerçekleşecektir.



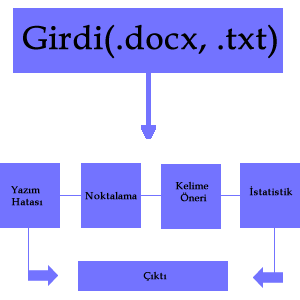
1. **Sistem Tasarımı**
   1. **Genel Tasarım Bilgileri**
      1. Genel Sistem Tanımı

Mevcut sistemde kullanıcılar Word, .txt .docx uzantılı dosyalar üzerideki kelime hataları cümle hataları gibi yazım yanlışı yapılmış kelimeler için doğru olabilecek kelime önerir. Noktalama hatalarını görebilecek yükledikleri dosyaları sistemde kayıtlı olarak tutabilecek aynı zamanda bu dosyaları istediği zaman görüntüleyebilecektir. Sistemin bir diğer özelliği ise kullanıcılar sınırsız denetleme yapabilecektir ve istatistik sonuçları gösterebilecektir;

* Kelime sayısı
* Benzersiz kelime sayısı
* Karakter Sayısı
* Benzersiz karakter Sayısı
* Paragraf Sayısı
* Noktalama Sayısı
* Cümle Sayısı
* Tez Özgünlük Kontrol
  + 1. Varsayımlar ve Kısıtlamalar

Proje kısıtlarında  varsayımları ve bağımlılıkları terimleri benzer gözükse de pratikte kafa karıştırıcı olabilirler. Göz önüne aldığımız birkaç temel noktadan bahsedelim;  
Proje kısıtları tıpkı müşteriler gibi paydaşlar tarafından da belirlenir.  
Yeni proje kısıtları proje planına uygun olmalıdır.  
Proje kısıtları belgelenmeli ve iyi iletişimle aktarılmalıdır. Proje kısıtlarında eksik iletişim, projede başarısızlığa sebep olabilir.  
Proje kısıtları belgelenirken ayrı bir kısımda belgelenmelidir.

* + 1. Veri Modeli



* + 1. Testler

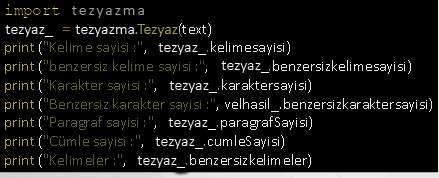
Genel olarak testlerimiz iki aşamada oluşturduk. Bunlar Alfa ve Beta aşamalarıdır;

Alfa Aşaması: Sistemin geliştirildiği yerde kullanıcıların gelerek katkıda bulunması sistemi test etmesi ile yapılacak.

Beta Aşaması: Kullanıcı, geliştirilen sistemi kendi yerleşkesinde, bir gözetmen eşliğinde yapılacak.

* + 1. Performans

Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecek Sistemin Tasarıma Uygunluk Performansı; Tasarımı oluşturulan sistemin hız ve doğruluk etkenleri incelenerek sisteme daha iyi bir performans sağlanması için geliştirmeler yapılacaktır.

* 1. **Süreç Tasarımı**
     1. Modüller
        1. İstatistik Modülü  
           

İstatistik modüle ile kullanıcı etkileşimini arttırmak hedeflenmiştir. İstatistik modülü yüklenen Word veya txt dosyalarındaki kuralları kontrol eder.

* + - * 1. İşlev

Metnin kelime sayısı cümle sayısı benzersiz kelime ve cümle sayısı gibi özellikleri kullanıcıya sunar

* + - * 1. Modül Tanımı

Modül tanımında kontroller ön planda tutulmuştur. Kullanıcının sistemde daha uzun süre kalması ve detaylarına müdahale edebileceği ekrana erişmesi için gereken bir modüldür. Kullanıcının girdiği veri ve metne göre modülün kontrol yapılarından geçip doğruluğu onaylanır. Kullanıcının hata olasılığını minimuma indirilir.

* + 1. Kullanıcı Profilleri

2 Çeşit kullanıcı profili vardır. Bunlar admin ve kullanıcılardır. Kullanıcılar sistemin özelliklerinden faydalanabilirken adminler ise sisteme güncelleme ve kelime ekleme gibi işlemler yapabilir sistemin çalışma prensipleri üzerinde değişiklik yapabilirler.

* + 1. Entegrasyon ve Test Gereksinimleri

Mevcut sistem entegre bir yapı kullanmamaktadır. Test, bir sistemin veya sistemin bileşenlerinin belirtilen gereklilikleri karşılayıp karşılamadığını öğrenmek amacıyla yapılan bir değerlendirme sürecidir. Bu süreç, yazılım geliştirme uzmanları tarafından başlayan ve son kullanıcıya kadar uzanan teknik seviyelerden oluşur. Test gereksinimi olarak sistemin

* Müşteriye sunulmadan önce ürün kalitesinden emin olmak,
* Yeniden çalışma (düzeltme) ve geliştirme masraflarını azaltmak,
* Geliştirme işleminin erken aşamalarında yanlışları saptayarak ileri aşamalara yayılmasını önlemek, böylece zaman ve maliyetten tasarruf sağlamak,
* Müşteri memnuniyetini arttırmak ve izleyen siparişler için zemin hazırlamak,

1. **Sistem Gerçekleştirimi**
   1. **Giriş**

Söz konusu ortamda belirli bir standartta geliştirilen programlar, gözden geçirilir, sınanır ve uygulamaya hazır hale getirilir.Üretilen kaynak kodların belirlenecek bir standartta üretilmesi yazılımın daha sonraki aşamalardaki bakımı açısından çok önemlidir.Tersi durumda kaynak kodların okunabilirliği, düzeltilebilirliği zorlaşır ve yazılımın işletimi süresince ortaya çıkabilecek sorunlar kolayca çözülemez

* 1. **Yazılım Geliştirme Ortamları**
     1. Programlama Dilleri

Gerçekleştirdiğimiz mevcut sistemde c# programlama dillerini kullandık. Kodlama yaparken önem verdiğimiz şeyler öğrenilmesinin kolaylığı yani basitlik ve aynı zamanda sistemi olabildiğince açıklayıcı (yorum satırları) olmasına dikkat ettik. Projeyi ekip halinde geliştirdiğimizden dolayı kodların kolay okunabilir, anlaşılabilir olmasına özen göstererek ortak bir platformda projeyi geliştirdik. Kodlar içerisindeki gerek değişkenler olsun gerek kullandığımız sistemin algoritmaları olsun karmaşıklık ve sistemin hızını en optimal şekilde kodlamaya çalıştık.

* 1. **Kodlama Stili**

Kendimize has kodlama bicini kullandık herhangi bir hazır düzene bağlı kalmadık Bakım programcımıza da aynı stil üzerine eğitim verdik ve sorunları ortadan kaldırdık.

* + 1. Açıklama Satırları

Açıklama veya  yorum satırları programlama dilleri tarafından kod olarak görünmez ve yorumlanmazlar. Programlama dillerinde yorum satırı diğer bir ifadeyle açıklama satırı ekleme programcılar tarafından sıklıkla kullanılmaktadır.

* + 1. Kod Biçimlemesi

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

* + 1. Anlamlı İsimlendirme

Sistem kodlamasının genel yapısında kullanılan değişkenlerin veri tabanında karşılığı varsa önce “tabloadı\_islevadı\_sayısı” seklinde bir anlamlı isimlendirme yaptık.

* + 1. Yapısal Programlama Yapıları

Genel olarak 3 başlıkta incelersek: • Ardışık işlem yapıları: Bu tür yapılarda genellikle fonksiyon, altprogram ve buna benzer tekrarlı yapıları tek bir seferde çözdük. • Koşullu işlem yapıları: Bu yapıları ise neredeyse programın tamamında kullandık karşılaştırma yapılan her yerde bunlara yer verildi. • Döngü yapıları: Tıpkı ardışık işlemler gibi alt alta birkaç satır yazıcığımıza tek bir döngüyle bu sorunların üstesinden geldik.

* 1. **Program Karmaşıklığı**

Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür. Bu bölümde bu ölçüt anlatılmaktadır. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir. McCabe ölçütü, bir programda kullanılan "koşul" deyimlerinin program karmaşıklığını etkileyen en önemli unsur olduğu esasına dayanır.

* 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 Tanımları

Olağandışı gelişen durumlarda try-catch blokları devreye girecek ve program kırılmadan çalışmasına devam edebilecek şekilde tasarladık. 71 Farklı Olağandışı Durum Çözümleme Yaklaşımları Tüm olağan dışı durumlarda program kırılmadan hata mesajlarıyla tekrar başa dönecek şekilde tasarladık.

* 1. **Kod Gözden Geçirme**

Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeden okunabilir bir program yazamaz. Hiçbir yazı editörün onayını almadan basılamayacağı gibi hiçbir program da incelenmeden, gözden geçirilmeden işletime alınmamalıdır. Kod gözden geçirme ile program sınama işlemlerini birbirinden ayırmak gerekir. Program sınama, programın işletimi sırasında ortaya çıkabilecek yanlış ya da hataları yakalamak amacıyla yapılır. Kod gözden geçirme işlemi ise, programın kaynak kodu üzerinde yapılan bir incelemedir. Kod gözden geçirmelerinde program hatalarının %3-5 oranındaki kesimi yakalanabilmektedir. Eğer programı yazan kişi, yazdığı programın hemen sonra bir "kod inceleme" sürecine girdi olacağını bilerek program yazdığında daha etkin, az hatalı ve okunabilir programlar elde edilebilmektedir.

* + 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.

* + 1. 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.

* + - 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.

* + - 1. 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.

* + - 1. Veri Kullanımı

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

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

• Değişken adları,işlevlerini yansıtacak biçimde anlamlı mı?

• Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?

• Her değişken tek bir amaçla mı kullanılıyor?

• Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?

• Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı? Şeklinde oldu.

* + - 1. Öbeğin Düzenlenişi

• Modüller birleşimi uyumlumu?

• Modüller arası veri aktarımları sağlanıyor mu?

• Bütün modüller birleştiğinde sistem çalışıyor mu? Gözden geçirme sırasında referans alınacak sorular olacaktır.

* + - 1. 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?

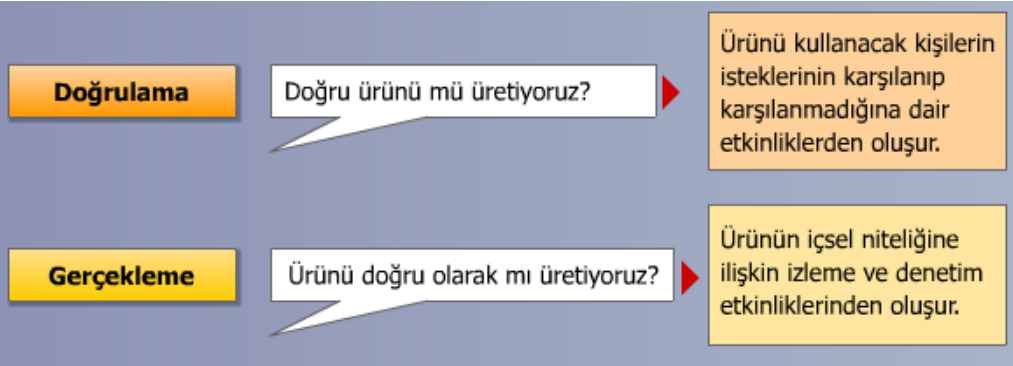
• Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

1. **Doğrulama ve Geçerleme**
   1. **Giriş**

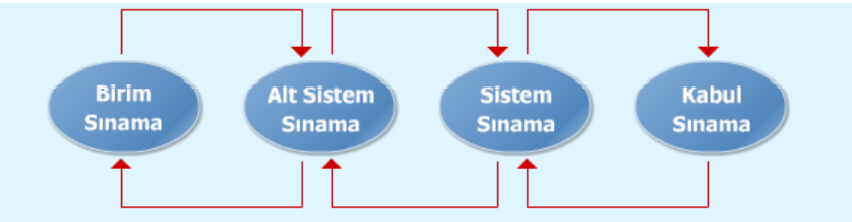
Geliştirilecek bilgi sistemi yazılımının doğrulanması ve geçerlenmesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler:

• 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.

* 1. **Sınama Kavramları**

****

Sınama ve Bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerinin seçilmesi gerekmektedir. Sınama işlemleri dört ana sınıfta incelenebilir:

• Birim sınama

• Alt-sistem sınama

• Sistem sınama

• Kabul sınaması

Birim sınama : Bağlı oldukları diğer sistem unsurlarından tümüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.

Alt-sistem sınama : Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar. ν Yine bağımsız olarak sınamaları yapılmalıdır. ν Bu aşamada en çok hata arayüzlerde bulunmaktadır. Bu yüzden arayüz hatalarına doğru yoğunlaşılmalıdır.

Sistem sınama : Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde çıkacak hatalar aranmaktadır.

• Ayrıca, belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

Kabul sınaması : Çalıştırılmadan önce sistemin son sınamasıdır.

Kabul Sınama : Çalıştırılmadan önce sistemin son sınamasıdır.

• Artık, yapay veriler yerine gerçek veriler kullanılır.

• Bu sınama türü alfa sınaması veya beta sınaması olarak ta bilinir.

* 1. **Doğrulama ve Geçerleme Yaşam Döngüsü**

Doğrulama ve Geçerleme Yaşam Döngüsü ν Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır. ν Planlama genellikle; ν alt-sistem, ν bütünleştirme, ν sistem ve ν kabul sınamalarının tasarımlarını içerir. ν Gerçekleştirim aşamasının sonunda ise söz konusu plan uygulanır.

* 1. **Sınama Yöntemleri**

• Her yazılım Mühendisliği ürünü iki yoldan sınanır:

• Kara kutu testi (Black-Box testing ): Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün testidir. Sistem şartnamesinin gerekleri incelenir.

• Beyaz Kutu Testi (White Box testing ): İç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır.

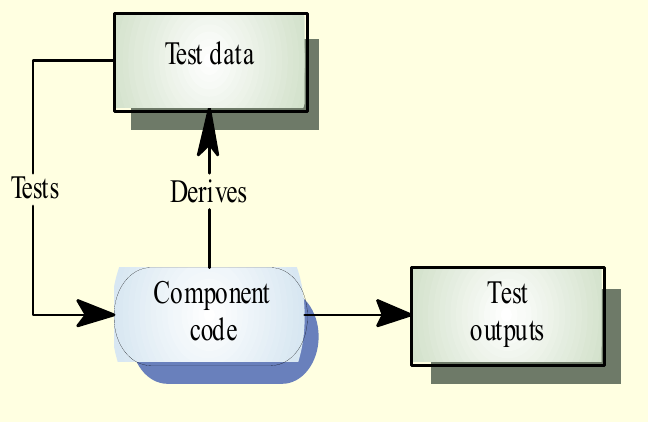
* + 1. Beyaz Kutu Sınaması

• Bütün bağımsız yolların en az bir kez sınanması gerekir.

• Bütün mantıksal karar noktalarında iki değişik karar için sınamalar yapılır.

• Bütün döngülerin sınır değerlerinde sınanması

• İç veri yapılarının denenmesi



class BinSearch {

public static void search ( int key, int [] elemArray, Result r )

{

int bottom = 0 ;

int top = elemArray.length - 1 ;

int mid ;

r.found = false ;

r.index = -1 ;

while ( bottom <= top )

{

mid = (top + bottom) / 2 ;

if (elemArray [mid] == key)

{

r.index = mid ;

r.found = true ;

return ;

} // if part

else

{ if (elemArray [mid] < key) bottom = mid + 1 ;

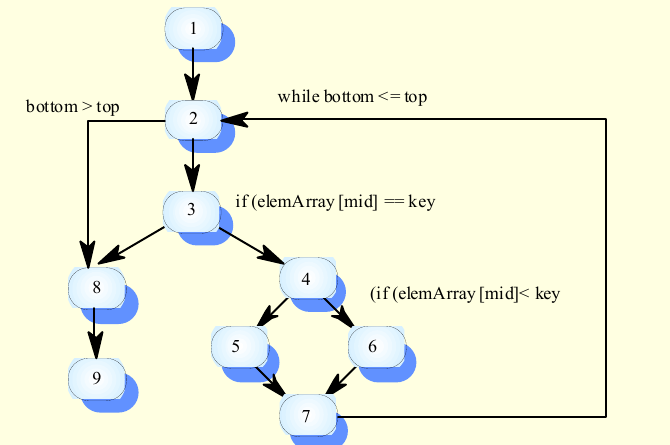
else

top = mid - 1 ;

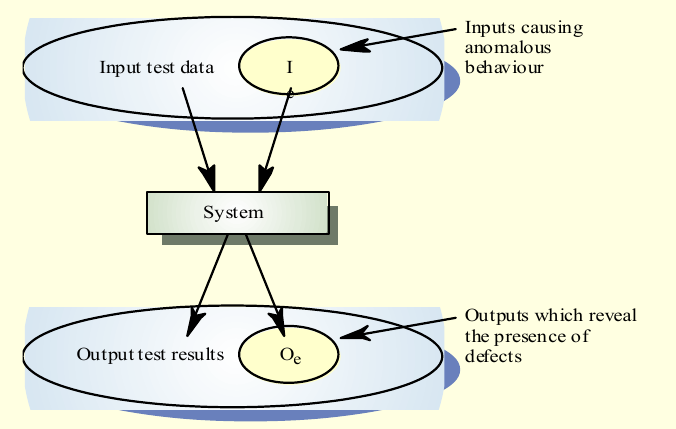
}

} //while loop

} // search } //BinSearch



* + 1. Kara Kutu Sınaması

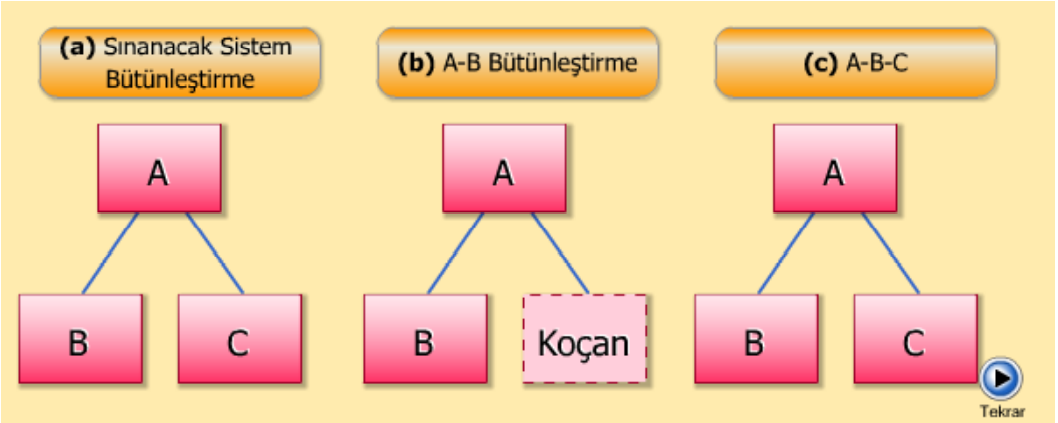


* 1. **Sınama ve Bütünleştirme Stratejileri**

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.

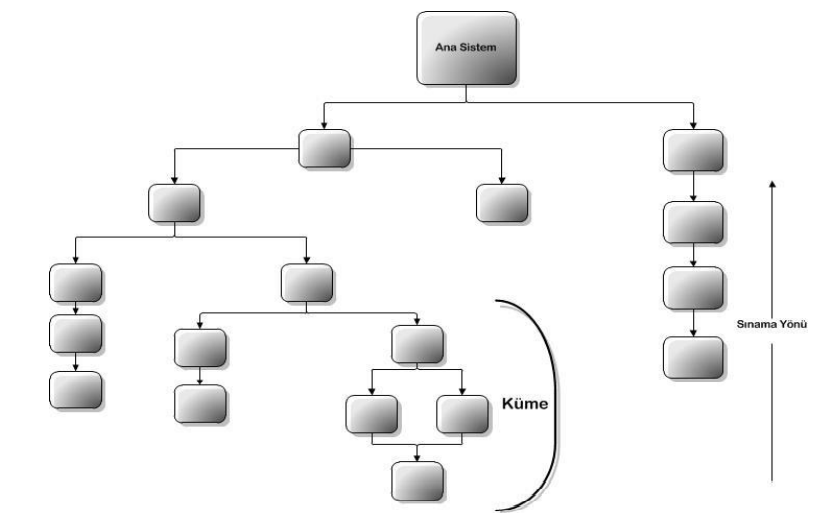
* + 1. Yukarıdan Aşağı Sınama ve Bütünleştirme

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.



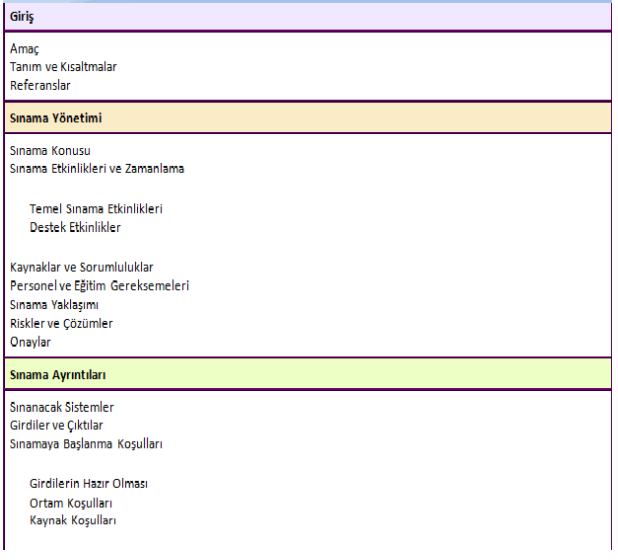
* + 1. Aşağıdan Yukarıya Sınama ve Bütünleştirme

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.



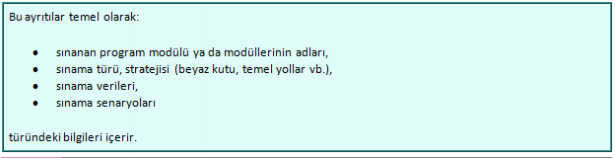
* 1. **Sınama Planlaması**

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.

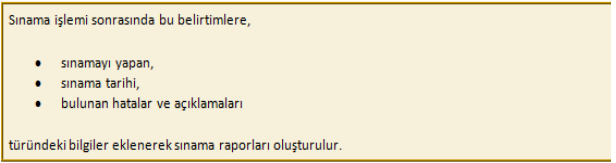


* 1. **Sınama Belirtimleri**

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



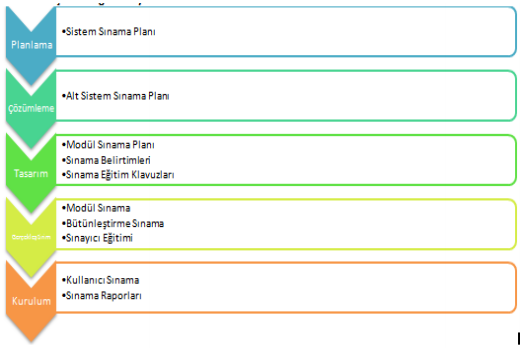
Sınama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir. Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtimlerinin hazırlanmasındaki temel maç, etkin sınama yapılması için bir rehber oluşturması.



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

* 1. **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.



Bu ayrıtılar temel olarak:

• Sınanan program modülü ya da modüllerinin adları,

• Sınama türü, stratejisi (beyaz kutu, temel yollar vb.),

• Sınama verileri,

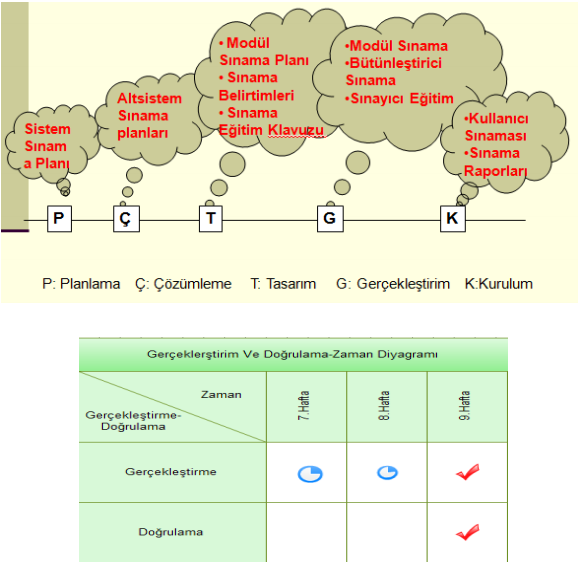
• Sınama senaryoları türündeki bilgileri içerir. Sınama sırasında bulunan her hata için, değişiklik kontrol sistemine (DKS), "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

• Onulmaz Hatalar: BT projesinin gidişini bir ya da birden fazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.

• Büyük Hatalar: Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.

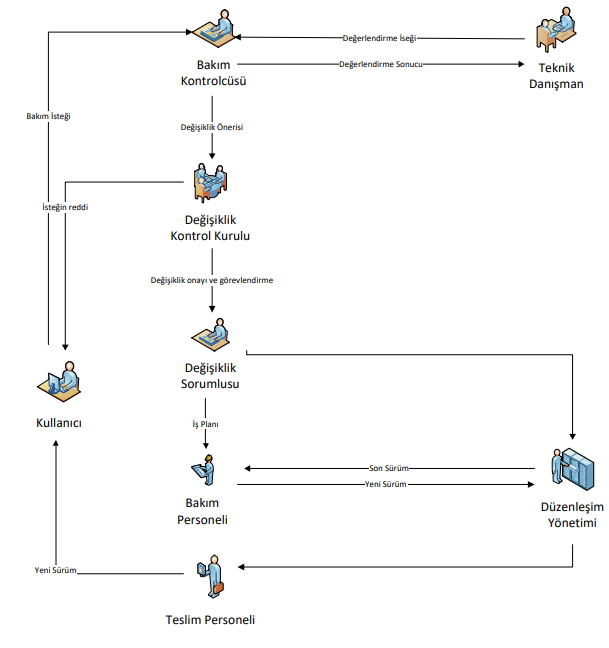
• Küçük Hatalar: Projeyi engellemeyen, ve giderilmesi az çaba gerektiren hatalardır.

• Şekilsel Hatalar: Heceleme hatası gibi önemsiz hatalardır.



1. **BAKIM**
   1. **Giriş**

Sistemin tasarımı bittikten sonra artık seçimden seçime sistemin bakıma sokulması gerekir daha öncede belirttiğimiz gibi sistem hassas ve hata kabul etmeyecek bir sistemden bahsediyoruz. Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı baz olarak alınmıştır.



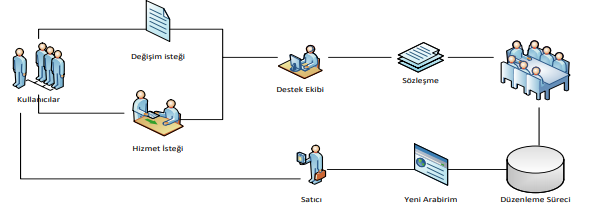
* 1. **Kurulum**

Sistem kurulumuna değinmek gerekirse devlet güvencesinde verilecek olan serverlara yüklenecek olan sistemimizde FTP arayüzü ile dosyaları servera aktaracağız ve internet explorer olan tüm cihazlarda çalışacak.

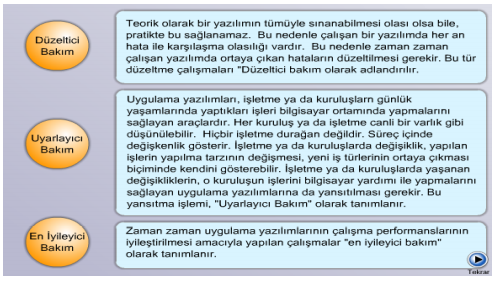
* 1. **Yerinde Destek Organizasyonu**

Bu konuyla ilgili pilot bölgede bizzat desteği ben vereceğim bunun yanında sistem canlandırılıp gerçeğe geçirilirse sistem tanımlaması kurulum için bölgelerde bayilik sistemi gibi alt kuruluşlara yetki verilecek eğer profesyonel destek istenirse yol uçak masrafını karşılamak şartıyla bölgeye yetkili gönderilip orada bir organizasyon yapılacaktır.

* 1. **Yazılım Bakımı**

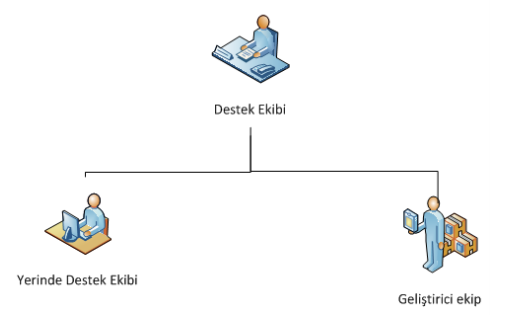


* + 1. Tanım



* + 1. 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.



1. **Sonuç**
   1. Sonuç Olarak Tez programımız yüklediğimiz .pdf, .docx uzantılı dosyalarda gerekli olan taramaları ve tezlerin kontrollerini sağlamaktadır.
2. **Kaynaklar**
   1. Kaynaklar

* <https://github.com/ahmetaksakall/TezKontrol>
* https://www.youtube.com/watch?v=-qEl2t8oxUw&feature=youtu.be&ab\_channel=Ra%C5%9FitY%C4%B1lmaz