

**FIRAT ÜNİVERSİTESİ**

TEKNOLOJİ FAKÜLTESİ

Yazılım Mühendisliği Bölümü

**YMH319 – Programlama Dilleri**

**Proje Uygulaması ve Dokümantasyonu**

**Kodlama Eğitim Sistemi**

170541612 Badour alkhateb

**Ocak – 2021**

|  |
| --- |
| **1. GİRİŞ** |
| **1.1** Önsöz  **1.2** Projenin Amacı  **1.3** Projenin Kapsamı |
| **2. PROJE PLANI** |
| **2.1** Giriş  **2.2** Projenin Plan Kapsamı  **2.3** Proje Zaman-İş Planı  **2.4** Proje Ekip Yapısı  **2.5** Önerilen Sistemin Teknik Tanımları  **2.6** Kullanılan Özel Geliştirme Araçları ve Ortamları  **2.7** Proje Standartları, Yöntem ve Metodolojiler  **2.8** Kalite Sağlama Planı  **2.9** Konfigürasyon Yönetim Planı  **2.10** Kaynak Yönetim Planı  **2.11** Eğitim Planı  **2.12** Test Planı  **2.13** Bakım Planı |
| **3. SİSTEM ÇÖZÜMLEME** |
| **3.1 Mevcut Sistem İncelemesi**  3.1.1 Örgüt Yapısı  3.1.2 İşlevsel Model  3.1.3 Veri Modeli  3.1.4 Var Olan Yazılım/Donanım Kaynakları  3.1.5 Var Olan Sistemin Değerlendirilmesi  **3.2 Gereksenen Sistemin Mantıksal Modeli**  3.2.1 Giriş  3.2.2 İşlevsel Model  3.2.3 Genel Bakış  3.2.3.1 İşleyiş Diyagramı  3.2.3.2 Sınıf Diyagramı  3.2.4 Başarım Gerekleri  **3.3 Arayüz (Modül) Gerekleri**  3.3.1 Yazılım Arayüzü  3.3.2 Kullanıcı Arayüzü  **3.4 Belgeleme Gerekleri**  3.4.1 Geliştirme Sürecinin Belgelenmesi  3.4.2 Eğitim Belgeleri  3.4.3 Kullanıcı El Kitapları |
| **4. SİSTEM TASARIMI** |
| **4.1 Genel Tasarım Bilgileri**  4.1.1 Genel Sistem Tanımı  4.1.2 Varsayımlar ve Kısıtlamalar  4.1.3 Sistem Mimarisi  4.1.4 Dış Arabirimler  4.1.4.1 Kullanıcı Arabirimleri  4.1.4.2 Veri Arabirimleri  4.1.5 Testler  4.1.6 Performans  **4.2 Veri Tasarımı**  4.2.1 Tablo tanımları  4.2.2 Tablo- İlişki Şemaları  4.2.3 Veri Tanımları  4.2.4 Değer Kümesi Tanımları  **4.3 Süreç Tasarımı**  4.3.1 Genel Tasarım  4.3.2 Modüller  4.3.2.1 Yönetici Modülü  4.3.2.1.1 İşlev  4.3.2.1.2 Kullanıcı Arabirimi  4.3.2.1.3 Modül Tanımı  4.3.2.1.4 Modül İç Tasarımı  4.3.2.2 Kullanıcı Modülü  4.3.2.2.1 İşlev  4.3.2.2.2 Kullanıcı Arabirimi |
| 4.3.2.2.3 Modül Tanımı  4.3.2.2.4 Modül İç Tasarımı  4.3.3 Kullanıcı Profilleri  4.3.4 Entegrasyon ve Test Gereksinimleri  **4.4 Ortak Alt Sistemlerin Tasarımı**  4.4.1 Ortak Alt Sistemler  4.4.2 Modüller arası Ortak Veriler  4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri  4.4.4 Güvenlik Alt Sistemi  4.4.5 Veri Dağıtım Alt Sistemi  4.4.6 Yedekleme ve Arşivleme İşlemleri |
| **5. SİSTEM GERÇEKLEŞTİRİMİ** |
| **5.1.** Giriş  **5.2.** Yazılım Geliştirme Ortamları  5.2.1 Programlama Dilleri  5.2.2 Tasarım Araçları  **5.4.** Kodlama Stili  5.4.1 Açıklama Satırları  5.4.2 Kod Biçimlemesi  5.4.3 Anlamlı İsimlendirme  5.4.4 Yapısal Programlama Yapıları  **5.5.** Program Karmaşıklığı  5.5.1 Örnek Dfa Tasarımı Ön sayfalar  5.5.2 Örnek Dfa Tasarımı  5.5.3 Örnek Dfa Tasarımı  **5.6.** Olağan Dışı Durum Çözümleme  5.6.1 Olağandışı Durum Tanımları  5.6.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları  **5.7.** Kod Gözden Geçirme  5.7.1 Gözden Geçirme Sürecinin Düzenlenmesi  5.7.2 Gözden Geçirme Sırasında Kullanılacak Sorular  5.7.2.1 Öbek Arayüzü  5.7.2.2 Giriş Açıklamaları  5.7.2.3 Veri Kullanımı  5.7.2.4 Öbeğin Düzenlenişi  5.7.2.5 Sunuş |
| **6. DOĞRULAMA VE GEÇERLEME** |
| 6.1. Giriş  6.2. Sınama Kavramları  6.3. Doğrulama ve Geçerleme Yaşam Döngüsü  6.4. Sınama Yöntemleri  6.4.1 Beyaz Kutu Sınaması  6.4.2 Temel Yollar Sınaması  6.5.Sınama ve Bütünleştirme Stratejileri  6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme  6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme  6.6. Sınama Planlaması  6.7. Sınama Belirtimleri  6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri |
| **7. BAKIM** |
| 7.1 Giriş  7.2 Kurulum  7.3 Yerinde Destek Organizasyonu  7.4 Yazılım Bakımı  7.4.1 Tanım  7.4.2 Bakım Süreç Modeli |
| **8.** SONUÇ |
| **9.** KAYNAKLAR |

|  |
| --- |
| **1. GİRİŞ** |

* 1. **Önsöz**

Bu projede yer alan sistem, programlama eğitimi için muhteşem bir başlangıç sayılabilir. Özellikle de yaş grubuna bakılmaksızın olarak gerçekleşecektir. Projeye olan çalışmalarım ekip olmaksızın bir çalışma olduğu için çok fazla zamanımı aldı diyebilirim. Özellikle de projenin içerik eklenmesi ve durmadan hataları test edilip analize çıkartılması benim en çok takıldığım kısımlardı. Ama bu projenin sonucunda bir ürün olarak çıkacağına dair inancım var diyebilirim. Benim anlatacağım ve yorumlayabileceğim kısım bu kadardır. Şimdi de projenin ana içeriklerini ve maddeler halinde sistemlerini size anlatmaya geçebiliriz.

* 1. **Projenin Amacı**

Projenin amacı programlama diline olan merakı veya algoritmaya olan merakı için küçük çaplı bir eğitim aracıdır.

**Projenin Kapsamı**

Programlama dili öğrenmek isteyenlere yönelik bir projedir.

|  |
| --- |
| **2. PROJE PLANI** |

**2.1 Giriş**

Projeye başlayabilmek için ilk önce temel adımlarını kurulması gerekiyor. İlk yapılacak adımlara başlangıç olarak zaman ve maliyeti tahmin bir yaklaşımda kurabilmek.

AYARLANMAMIŞ İŞLEV NOKTASI PUANININ (AİP) BULUNMASI

AİP = 1 \* W(1) + 2 \* W(2) + 3 \* W(3) + (4) \* W(4) + 5 \* W(5)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ölçüm Parametresi** | Ağırlık Faktörü (W) | | | | | | | Parametre  Değeri |
| **Sayı** | **Yalın** | | **Ortalama** | | **Karmaşık** | |
| **(1)Kullanıcı Girdi sayısı** | - | | 3 | | 4 | | 6 |  |
| **(2)Kullanıcı Çıktı sayısı** | 8 | | 4 | | 5 | | 7 | 40 |
| **(3)Kullanıcı Sorgu Sayısı** | - | | 3 | | 4 | | 6 |  |
| **(4)Kütük Sayısı** | 4 | | 7 | | 10 | | 15 | 28 |
| **(5)Dışsal Arayüz Sayısı** | - | | 5 | | 7 | | 10 |  |
| TOPLAM |  | |  | |  | |  | **68** |
|  |  |  |  |  |  |  |  |  |

TEKNİK KARMAŞIKLIK FAKTÖRÜNÜN (TKF) BULUNMASI

|  |  |  |
| --- | --- | --- |
| 1. | Sistem güvenilir yedekleme ve kurtarma gerektiriyor mu? | 1 |
| 2. | Veri iletişimi gerekiyor mu? | 3 |
| 3. | Dağıtık fonksiyon var mı? | 4 |
| 4. | Performans kritik mi? | 2 |
| 5. | Sistem çok kullanılan bir işletim ortamında mı çalışacak? | 1 |
| 6. | Sistem on-line veri girişi gerektiriyor mu? | 0 |
| 7. | On-line veri giriş işlemlerinin birden fazla ekran ya da işlem üzerinden olmasını mı gerektiriyor? | 4 |
| 8. | Ana dosyalar on-line mı güncelleniyor? | 0 |
| 9. | Girdiler, çıktılar, dosyalar ve sorgular karmaşık mı? | 5 |
| 10. | Kod yeniden kullanabilir olarak mı tasarlanmış? | 5 |
| 11. | İç süreç karmaşık mı? | 5 |
| 12. | Dönüşüm ve kurulum tasarım içerisinde mi? | 1 |
| 13. | Uygulama değişik kuruluşlarda birden fazla kurulum gerektirecek şekilde mi tasarlanmış? | 1 |
| 14. | Uygulama kullanıcı tarafından kolaylıkla kullanmayı ve değiştirmek üzere mi tasarlanmış? | 2 |
|  | **0:** Hiçbir Etkisi Yok **1:** Çok Az etkisi var **2**: Etkisi Var **3**: Ortalama Etkisi Var  **4**: Önemli Etkisi Var **5**: Mutlaka Olmalı, Kaçınılamaz | 34 |

İŞLEV NOKTASI PUANININ (UFP) BULUNMASI

İN = AİN x (0.65 x 0.01 x TKF)

İN= 68\*(0,65\*0,01\*34) İN = **15,028**

SATIR SAYISININ BULUNMASI

Proje kodlamasında kullanılacak olan dil python olup, satır sayısı kestirim katsayısı 60 dur. Satır Sayısı = İN \* 30 = 15,028 \* 60 **= 786**

ETKİN MALİYET MODELİ: COCOMO

|  |  |  |
| --- | --- | --- |
| Ayrık Projeler | Yarı Gömülü Projeler | Gömülü Projeler |
| İş Gücü K = 2.4\*S1,05  Zaman T = 2.5\*K0,38 | İş Gücü K = 3,0\*S1,12  Zaman T = 2.5\*K0,35 | İş Gücü K = 3,6\*S1,20  Zaman T = 2.5\*K0,32 |
| S: 1000 türünden satır sayısı | | |

Projemiz, küçük ölçekli bir proje olduğundan Temel Model alınacaktır. Modelin uygulamasında kullanılan formüller aşağıdaki gibidir.

“Ayrık Proje” sınıfına giren projemiz için kullanacağımız COCOMO Formülü şu şekilde ortaya çıkmaktadır:

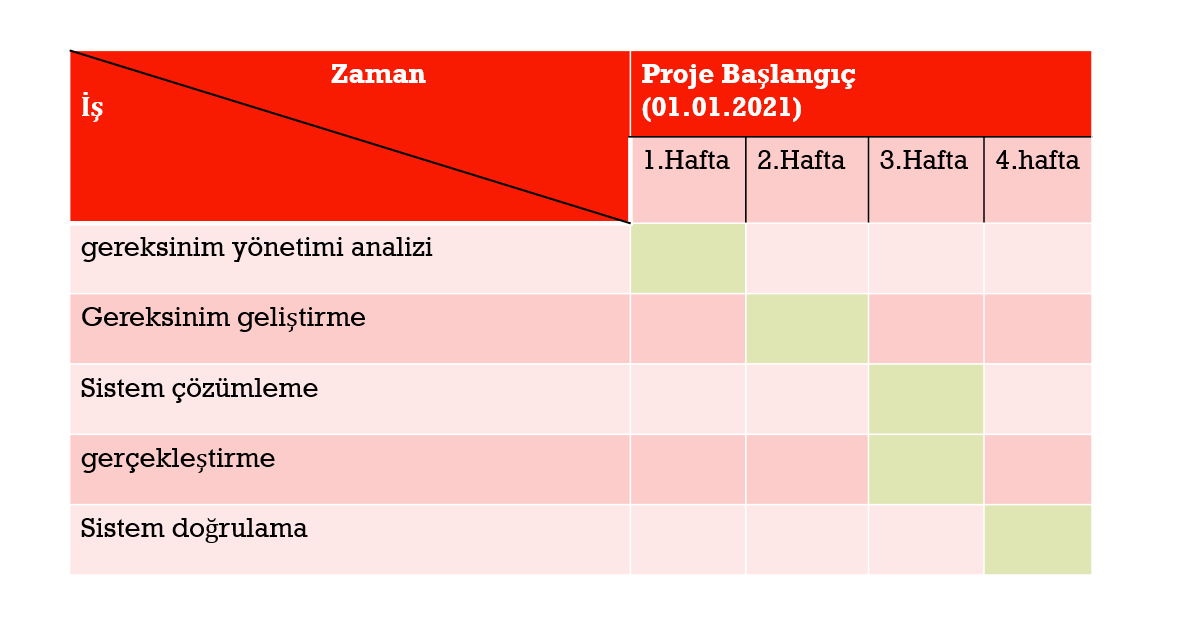
İş Gücü K = 2.4 \* S1,05 = 2.4 \* 0.51,05 = 1.16 (Aylık Kişi Başı İş Gücü)

Zaman T = 2.5 \* K0,38 = 2.5 \* 1.160.38= 2.9 = 3 Ay

**2.2 Projenin Plan Kapsamı**

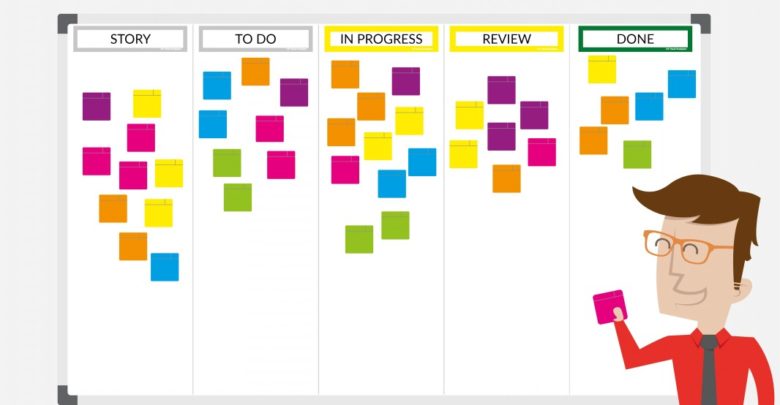
Bu yazılımın planı; yazılımın oluşturulması için gereken insan kaynaklarını, donanım ve yazılım kaynaklarını, yazılımın maliyet hesaplamalarını, mevcut sistemin incelenmesini ve önerilen sistemin gerektirdiği fonksiyonları kapsamaktadır.

**2.3 Yazılım Zaman-İş Planı**

****

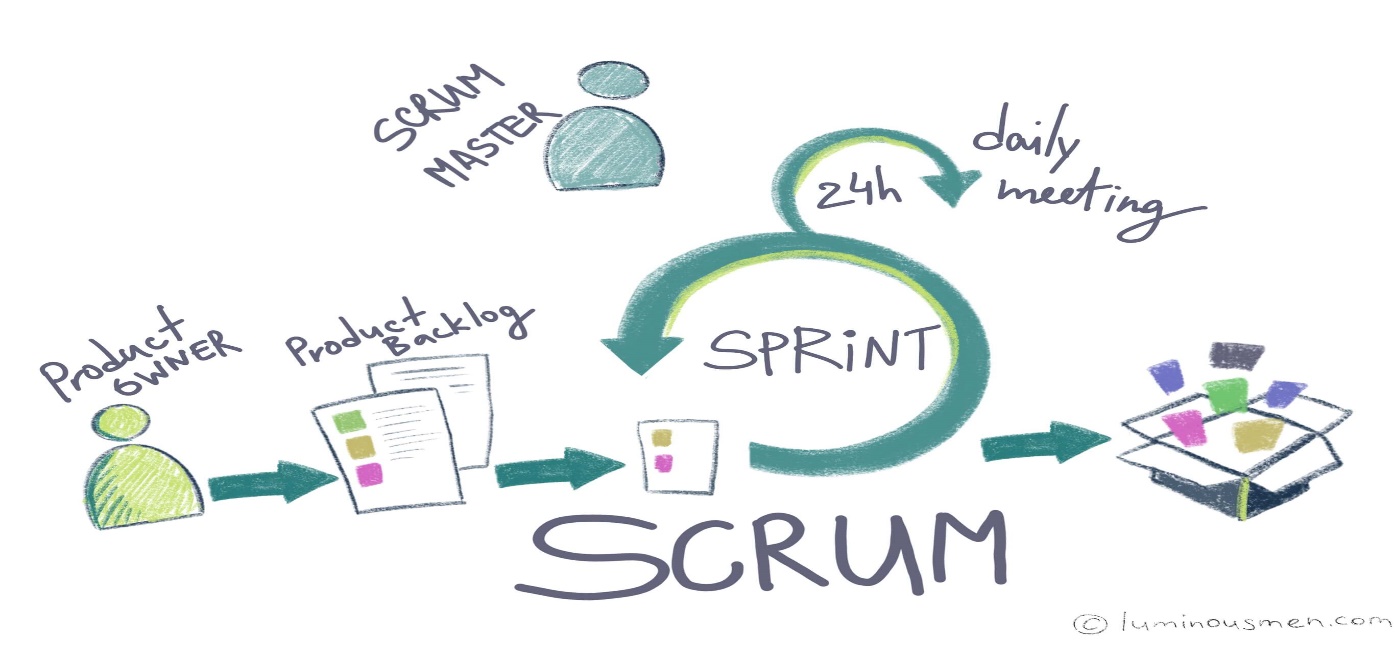
**Şekil 2.1 Yazılım Zaman-İş Planı**

**2.4 Yazılım Ekip Yapısı**



Kynk:001

Bizler zaman ve maliyet konusunda çok önemsediğimizi ve bu ikiliden nasıl fayda sağlanacağı hakkında araştırmalar yapıp dururuz aslında. Bir projenin temelini veya tasarımını çizmeden kodlamaya başlamak veya herhangi bir adımdan atlayarak işimizi hızlı bir şekilde bitirme çabasında oluyoruz. Bu bizim şu an bir hata payı göstermese bile zamanla bu hatanın bize büyük sorunlar yaratabilecektir. Bu sefer attığımız o sorumsuzca adımlarımızdan dolayı hem zaman hem de maliyet konusunda zarara uğramış oluruz. Buna dair iyi bir adımları çizmek ve bu çizdiğimiz adımları iyi bir şekilde emin olmak ve iyi bir şekilde tanımlamamız gerek.



Kynk: 002

Agile modeli proje yönetimi, yazılım geliştirme sürecinde karşılaşılan problemleri çözmek üzere, tekrarlanan yazılım geliştirme modeli taban alınarak geliştirilmiş, sık aralıklarla parça parça yazılım teslimatını ve değişikliği teşvik eden bir yazılım geliştirme modelidir.

**2.5 Önerilen Sistemin Teknik Tanımları**

|  |  |  |
| --- | --- | --- |
| **Donanım Türü** | **Özellikler** | **Kullanım Amacı** |
| Dizüstü Bilgisayar | Intel i7 işlemci 3.5 GHz, 8 GB RAM | Programlama |
| **Yazılım Türü** | **Özellikler** | **Kullanım Amacı** |
| Vs Code | Python derlenmesi | Sistem kodlama |
| Windows 10 | İşletim Sistemi | Program çalıştırma |

**Tablo 2.5.1. Önerilen Sistemin Donanım ve Yazılım Tanımları**

**2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları**

* Elektronik Belge Okuma Araçları
* Programlama Araçları
* Python 3.6

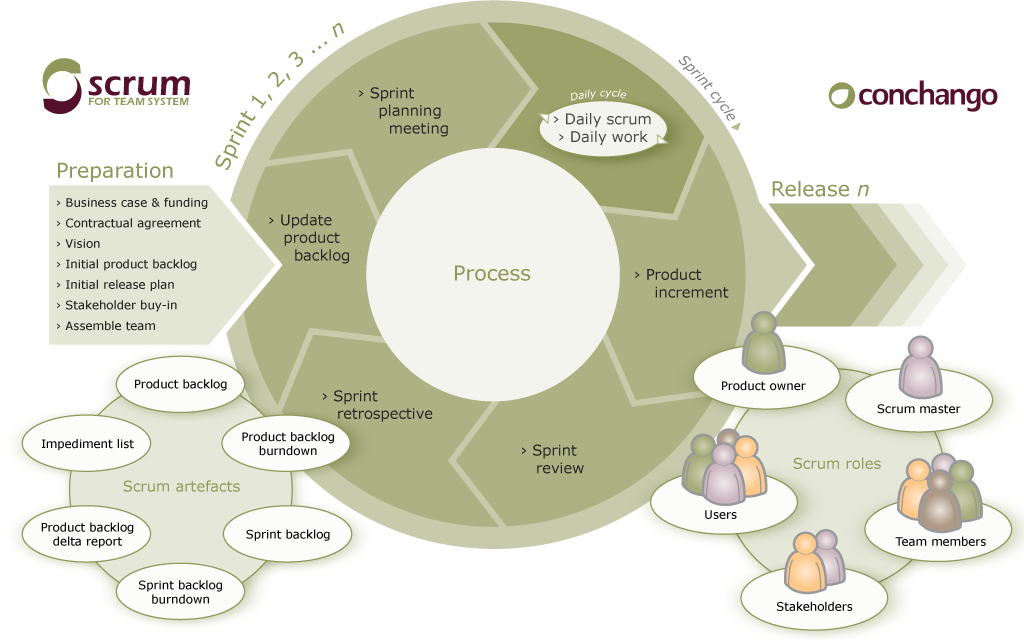
-İşletim Sistemleri

-Windows 10

**2.7 Proje Standartları, Yöntem ve Metodolojiler**

Uygulamanın geliştirilmesinde kullanılan yazılım süreç modeli Çevik (Agile) Modeldir.

Bu modele göre, iş sahibi (Product Owner), işi belirler. Daha sonra her biri alanında uzman 3 ila 7 personelden oluşan ve Scrum adı verilen ekipler oluşturulur. Bu ekiplerin yazılım geliştirme süreçleri, Sprint adı verilen 1 ila 5 haftalık zaman dilimlerinde olur. Scrum ekipleri, ilk önce bir Sprint Plan Toplantısı yapar ve o Sprint zamanı içinde neleri yapacaklarını kararlaştırır. Daha sonra bunları Scrum Master’ın (Ekip Lideri) direktiflerine uyarak gerçekleştirir. O Sprint zaman dilimindeki iş bittikten sonra da incelemeler yapılır ve tekrar planlamaya geçilir.



Kynk:003

**Şekil 2.2 Scrum ve Agile**

**2.8 Kalite Sağlama Planı**

****

**Şekil 2.3 Kalite Sağlama Planı**

1. **Tamlık:** Yazılımdaki fonksiyonların, butonların düzgün bir şekilde çalışmasıdır.
2. **Yeniden Kullanılabilirlik:** Yazılımın gerektiğinde tekrar düzenlenip kullanılabilmesidir.
3. **Güvenilirlik:** Yazılımdaki gerekli güvenlik önlemlerinin alınmasıdır.
4. **Modülerlik:** Yazılımın küçük parçalardan oluşması ve her bir parçanın bütün yapıyı bozmadan değiştirilebilmesinin sağlanmasıdır.
5. **Genellik:** Yazılımın ülke genelinde kullanılabilmesidir.
6. **Taşınabilirlik:** Microsoft işletim sistemi olan cihazlarda çalışır
7. **Kullanılabilirlik:** Yazılımın her insan tarafından kullanılabilmesidir.

**2.9 Konfigürasyon Yönetim Planı**

Sistemin ileride kullanıcının yeni isteklerini karşılayamaması veya sistemin yapısındaki bazı bileşenlerin değişmesi sonucunda sistem güncelliğini kaybedecektir. Bu gibi durumlar için yapılandırma planı hazırlanmıştır.

Yapılandırma planının uygulanmasını gerektiren bazı durumlar aşağıdaki gibidir:

* Kullanılan yazılım veya donanımın güncelliğini yitirmesi,
* Sistemde herhangi bir istenmeyen durum hali

|  |  |  |  |
| --- | --- | --- | --- |
| **Konfigüre Gerektiren Durum** | **İlgili Kişi** | **Süre** | **Sonuç** |
| **Standart kurulumunda mevcut olmayan bir dil paketi ihtiyacı** | BadourAlkhateb | 2 gün | Yeni bir dil paketi eklenerek, sistem kullanıcı gereksinimini karşılayacak şekilde Konfigüre  Edilecektir. |
| **Standart sürüm yükseltme ihtiyacı** | BadourAlkhateb | 5 gün | Sistemin sabitlendiği için gereken sürüm  Güncellemesi yapılacaktır. |
| **Tasarım gerekliliği** | BadourAlkhateb | 3 gün | Kullanıcıdan gelen tema isteği, beklentiyi karşılayacak şekilde kodlanıp mevcut sisteme  Yüklenecektir. |

**2.10 Kaynak Yönetim Planı**

Sistemin hızlı çalışabilmesi için uygun ve hızlı bir algoritma kurabilmesi gerekir. Yeni güncel algoritmaları kurulumunu gerçekleştirerek kullanıcılara yeniden dağıtımını gerçekleştirilmesi gerekir.

**2.11 Eğitim Planı**

Bu eğitimi planı proje teslimi gerçekleştirdikten sonra ek olarak, kısa ve anlamlı bir şekilde hazırlanmış bir kullanım kılavuzu vermemiz gerekir.

**2.12 Test Planı**

Scrum ekiplerindeki test mühendislerinin görevleri şunlardır:

* Projede oluşturulan parçalı proje gereksinimlerini karşılamasını sağlamak,
* Sistemdeki hız sorunlarının çözümü,
* Sistemin en uygun ve buna bağlı olarak performans artışını sağlamak.

**2.13 Bakım Planı**

Kullanıcılara yönelik özel konferanslar sergilenmesi gerekiyor. Sergilemeden sonra dokuman olarak hazırladığımız bir kullanım kılavuzu olarak da ayrıyeten verebiliriz.

|  |
| --- |
| **3. SİSTEM ÇÖZÜMLEME** |

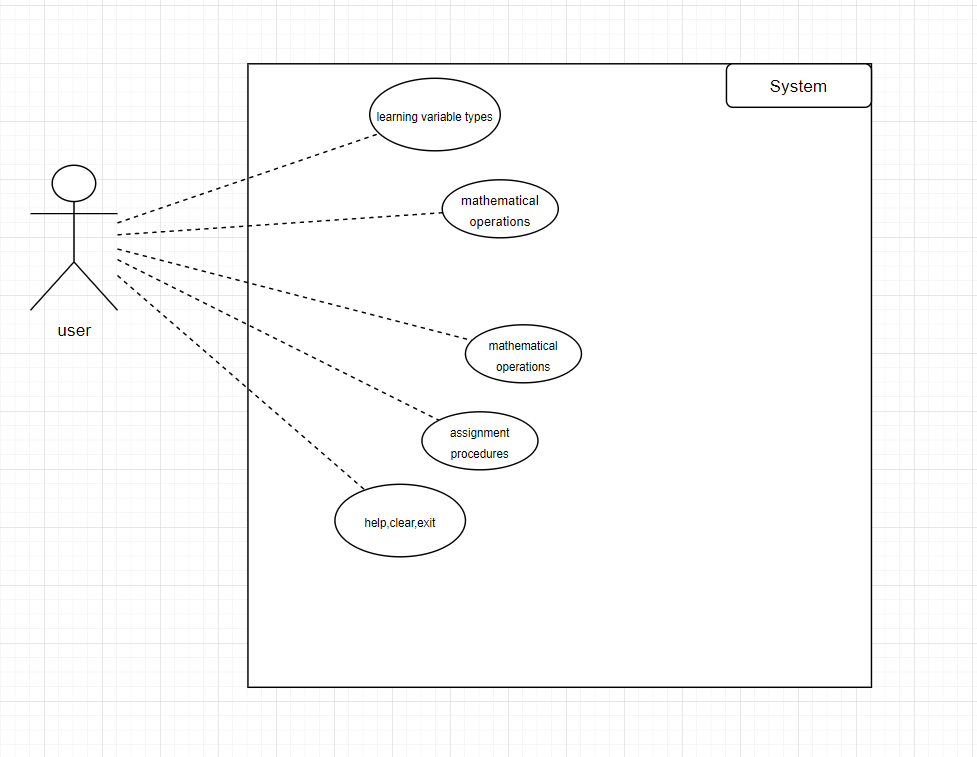
**3.1 Mevcut Sistem İncelemesi**

Kullandığım vs code ide yapısını incelendi ve not defteri(note pad) kullanış yapısını incelendi.

**3.1.1 Örgüt Yapısı**

Örgüt yapısı olarak şirket bünyesindeki bilgi işlem personellerinin, yazılım geliştirme bölümündeki çalışanların ve bilgi güvenliği bölümündeki personellerinin ortak çalışmalarıyla oluşan bir örgüt yapısı vardır.

**3.1.2 İşlevsel Model**

****

**Şekil 3.1.2.1 Sistemi ile ilişkisini gösteren Use Case Diyagramı**

Sistem gereksinimleri için müşterilerden gelen örnek senaryolar şunlardır:

1- Ben matematik işlemlerini kolaylıkla sorhulayabilirim.

2- Sistem: Kullanıcının girdileri benim fonksiyonumdaki kurallarına uyumlu olduğu sürece yanıtlarım.

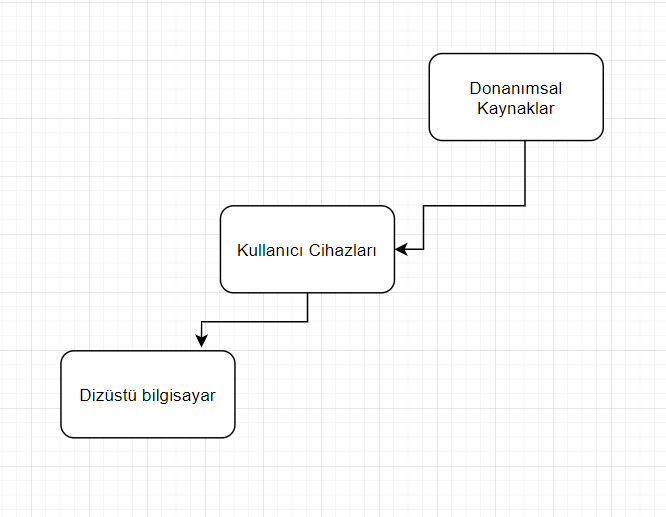
**3.1.4 Var Olan Yazılım/Donanım Kaynakları**

Donanım Kaynakları

* SDD
* Ram
* Dizüstü Bilgisayarlar

Yazılım Kaynakları

* Microsoft Office
* Vs Code
* Windows Server / Windows 10 Pro



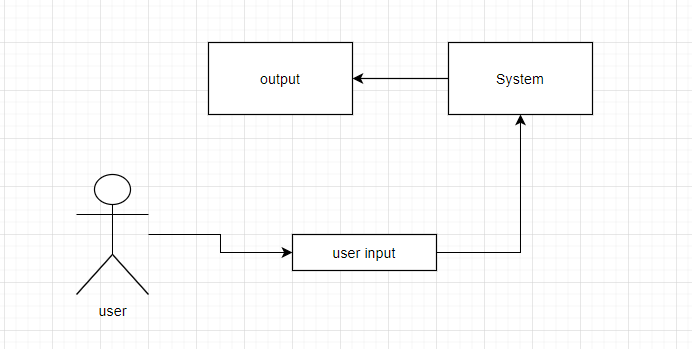
**Şekil 3.1.4.1 Donanımsal Kaynaklar**

**3.2 Gereksenen Sistemin Mantıksal Modeli**

**3.2.1 Giriş**

Bu sisteme göre herkes tarafından kullanılabilir özellikle de kod öğrenmeye yönelik bir çalışma olduğu için, öğrenmek isteyen bireylere özgü bir yapıdır diyebiliriz.

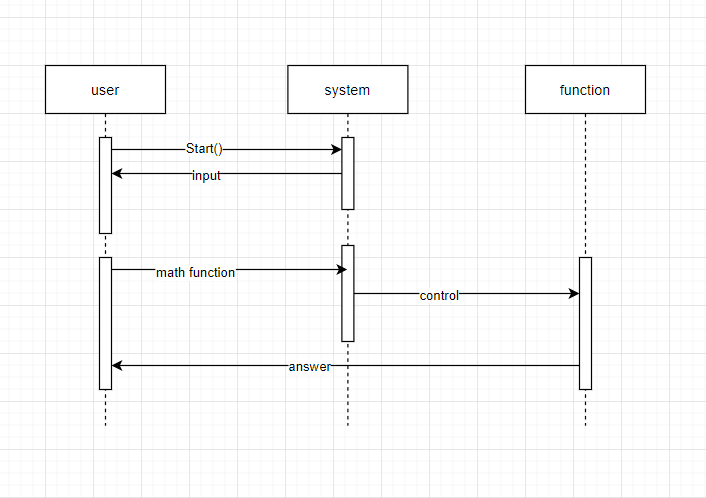
**3.2.2 İşlevsel Model**

**Şekil 3.2.2.1 Kullanıcı için UML Diyagramı**

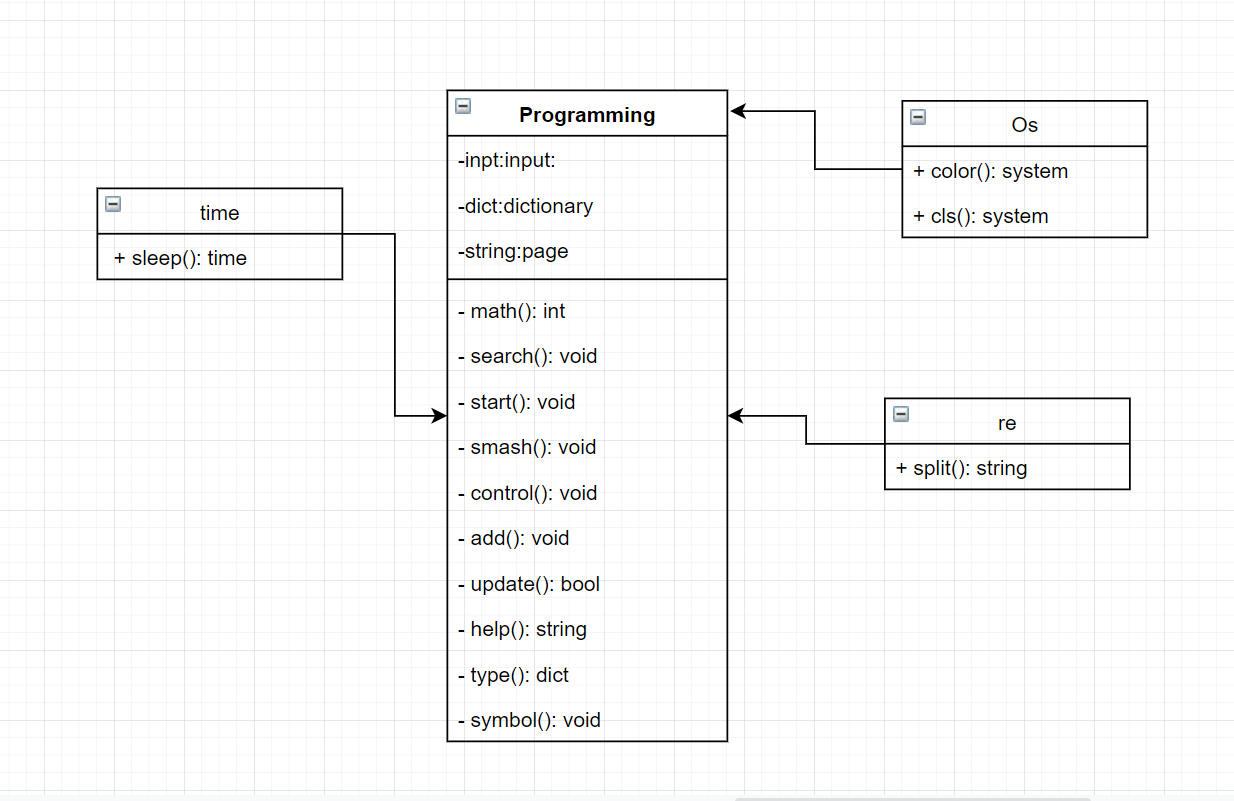
**3.2.3 Genel Bakış**

Genel olarak bakarsak aslında sistemin çok geniş bir çeşitliliğe sahip olmadığını, daha basit ve daha anlaşılır bir şekilde tasarlanmış ve kullanıcılara daha iyi bir şekilde sunabilen bir sistem olduğunu görüyoruz.

İşleme bakılırsa kullanıcıların girdilerini analiz ayrıştırma işlemlerini gerçekleştirerek sistemde bulunana fonksiyonlara dağıtım sağlanmaktadır. Fonksiyonda bulunan özellikler niteliğinde kullanıcıların isteklerini karşılayabilmektedir.

****

**Şekil 3.2.3.1 İşleyiş Diyagramı**

****

**Şekil 3.2.3.2 Sınıf Diyagramı**

**3.2.4 Başarım Gerekleri**

Mevcut sistem incelenmiş ve mevcut sistemin eksiklerinden yola çıkılarak, sistemin başarımı için:

* + - Sistemin sonuç üretim doğrulukları
    - Tepki sürelerinin en aza indirilmesi
    - Maliyetin en aza indirilmesi
    - Hata risklerinin en aza indirilmesi
    - Kullanım kolaylığı
    - Anlaşılabilirlik

Temel gereklilikler olarak tespit edilmiştir.

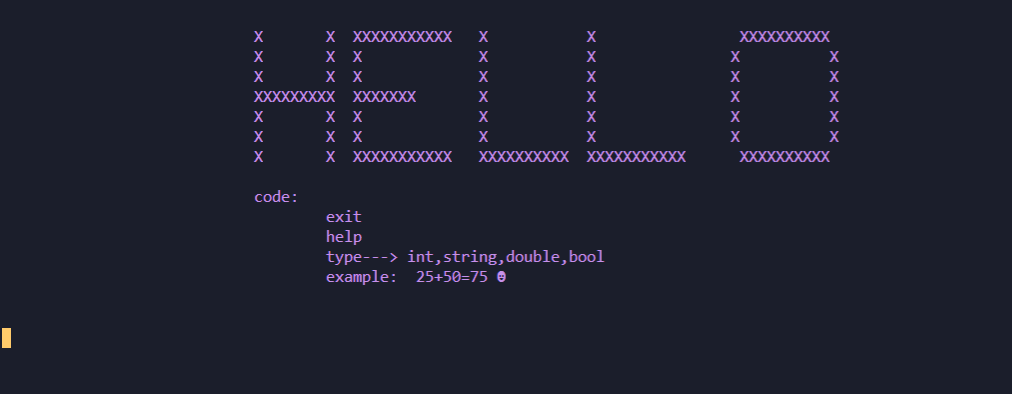
**3.3 Arayüz(Modül) Gerekleri**

**3.3.1 Yazılım Arayüzü**

Proje, çalışma süresince değişime gerek duyulmayacak şekilde tasarlanmıştır.Bir hata meydana gelmesi halinde projenin kaynak kodlarında değişiklik yapılarak hata giderilecektir.

**3.3.2 Kullanıcı Arayüzü:**

Projede kullanıcının sistemi daha rahat anlaması ve kullanabilmesi için arayüz tasarımı yapılırken bütün özellikler ve işlemler bir arada tutularak kullanıcıya sunulmuştur.

****

**Şekil 3.3.2.1 Kullanıcı Arayüzü**

**3.4 Belgeleme Gerekleri:**

**3.4.1 Geliştirme Sürecinin Belgelenmesi:**

Geliştirme sürecini belgelemek ileriye dönük olarak projeye katılan geliştirme ekibinin ve kullanıcıların projeyi daha iyi anlaması açısından etkin rol oynamaktadır. Projenin geliştirilme sürecinde ileriye dönük olarak tasarımsal hataların tespitinde önemli bir yer almaktadır. Bunun yanı sıra proje ekibinin projeye olan hâkimiyeti açısından oldukça önemlidir.

**3.4.2 Eğitim Belgeleri:**

Gerekli bilgi ve dokumnatasyonlar sitede belirlenere bilgileri oradan da ulaşılabilir bir biçimde kullanıcılara desteğini sağlamaktadır.

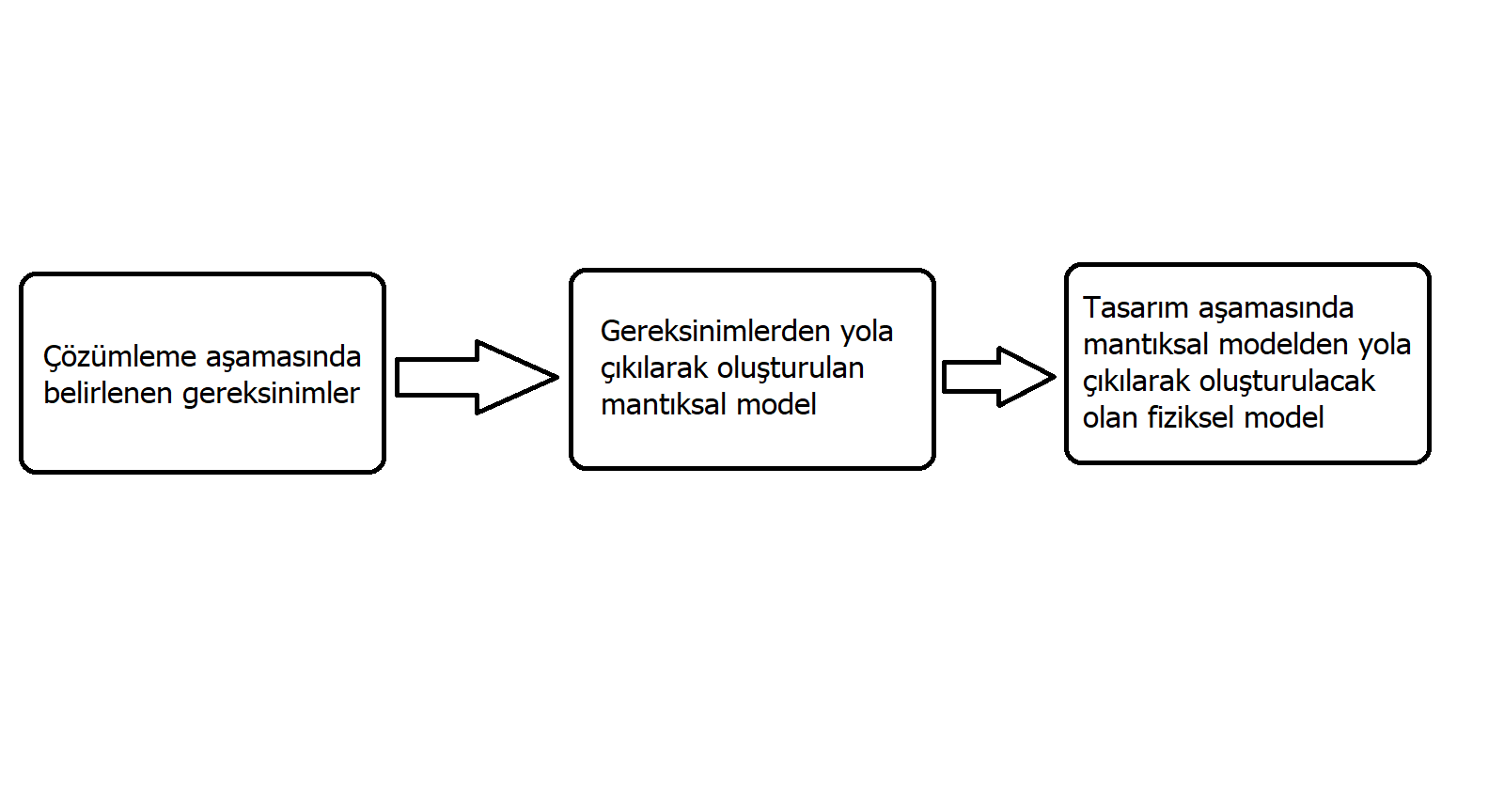
**3.4.3 Kullanıcı El Kitapları:**

Müşteriden gelen sürekli değişen istekler neticesinde ön bir kullanıcı el kitabı yayınlamak pek mümkün olmadığı için kullanıcı el kitabı oluşturmak için projenin tamamlanması beklenmektedir.

|  |
| --- |
| **4. SİSTEM TASARIMI** |

**4.1 Genel Tasarım Bilgileri**

**4.1.1 Genel Sistem Tanımı**

****

**Şekil 4.1.1.1. Genel Sistem Tanımı**

* Gereksinimler

Yapılan anketler sonucu müşterilerden gelen User Story’ler aracılığıyla gereksinimler belirlenmiştir.

* İşlevsel Belirtimler

Bu kısımda sistemin işlevleri belirtilecektir. Bu sistem nasıl çalışacak, ne yapacak gibi sorulara cevap verilecektir.

* Tasarım

Tasarım aşamasında olacaklar grafiksel olarak belirtilmiştir.

1. İlk önce bir süreç tasarımı olacak ve adımlar aşağıdaki gibi olacaktır.



**Şekil 4.1.1.2 Tasarım Aşamaları-1**



**Şekil 4.1.1.3 Tasarım Aşamaları-2**

2.Süreç tasarımı bittikten sonra sıra arayüz tasarımına gelmiş ve arayüz tasarımı aşağıda göründüğü adımlarla gerçekleştirilmiştir.



**Şekil 4.1.1.4 Tasarım Aşamaları-3**

3.Arayüz tasarımı bittikten sonra sıra yapısal tasarıma gelmiştir. Yapısal tasarımda izlenen yollar aşağıdaki gibi olmuştur.



**Şekil 4.1.1.5 Tasarım Aşamaları-4**

4.Yapısal tasarım bittikten sonra veri tasarımına geçilmiş ve şu adımlar izlenmiştir.



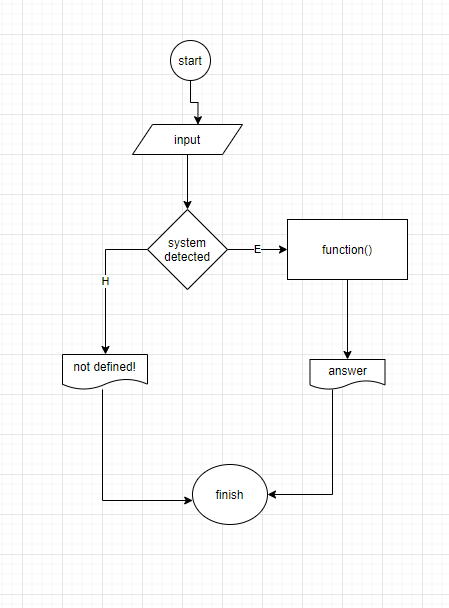
**Şekil 4.1.1.6 Tasarım Aşamaları-5**

**4.1.2 Varsayımlar ve Kısıtlamalar**

Sistemdeki başlıca kısıtlamalar şunlardır:

* Gerekli dokumantasyonları incelemiş kullanıcılar kullanabilir
* Az miktarda bilgisayar bilgisine sahip kullanıcılar kullanabilir

**4.1.3 Sistem Mimarisi**

****

**Şekil 4.1.3.1 Sistem Akış Diyagramı**

Sistemin mimarisinin akış diyagramı şeklinde verilmesinin temel nedeni sistemin işleyiş mantığının nasıl olduğu ve nasıl bir yol çizileceğinin bilinmesidir. Akış diyagramı sistemin temel mantığı hakkında fikir verecektir.

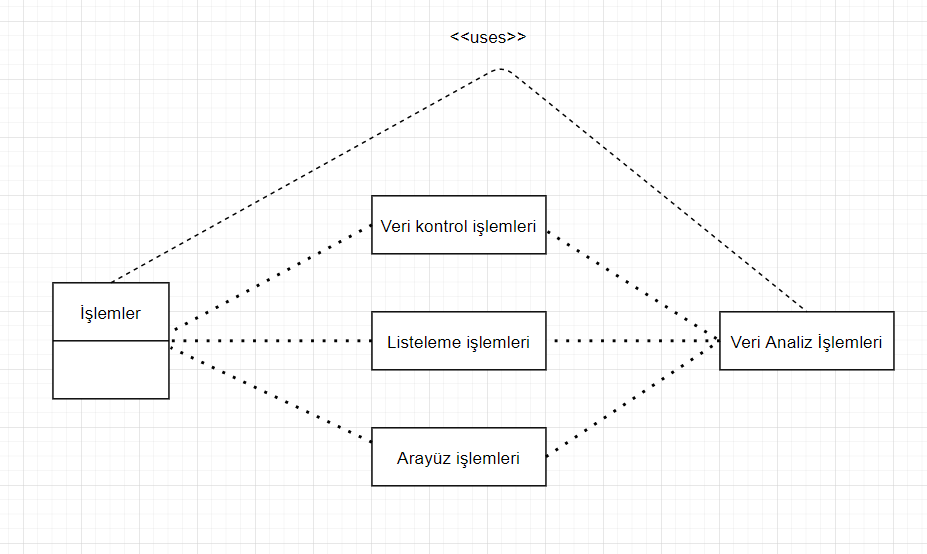
**4.1.4 Dış Arabirimler**

**4.1.4.1 Kullanıcı Arabirimleri**

Kullanıcı arabirimlerinin ilk başında sisteme giriş ekranı bulunacaktır ve yapması gereken tek şey istediği girdilerin karşılanmasıdır.

**4.1.4.2 Veri Arabirimleri**

Veri arabirimlerinde sistem Microsoft işletim sistemi üzerinde çalışacaktır. Kullanıcılara ait belgelendirme deposu uygulanacaktır.

****

**Şekil 4.1.5.2 Veri Modeli**

**4.1.5 Testler**

Sistem testleri şirket tarafından belirlenen pilot bölgelerde tek aşama halinde gerçekleştirilecektir.

**Alfa Aşaması**: Kullanıcılar, geliştirilen sistemi pilot bölgelerde, yetkili personel eşliğinde yapacaktır.

**4.1.6 Performans**

Test aşamasından sonra sistemin performansını etkileyen faktörlerden elde edilen test verileri değerlendirilecektir. Tasarımı yapılan sistemin sabitleştirilmiş ve işleyiş performansı değerlendirilecektir. Veri yapısının sistemle sabitleştirilmiş ve çalışma zamanındaki uyumluluk düzeyindeki performansı değerlendirilecektir.

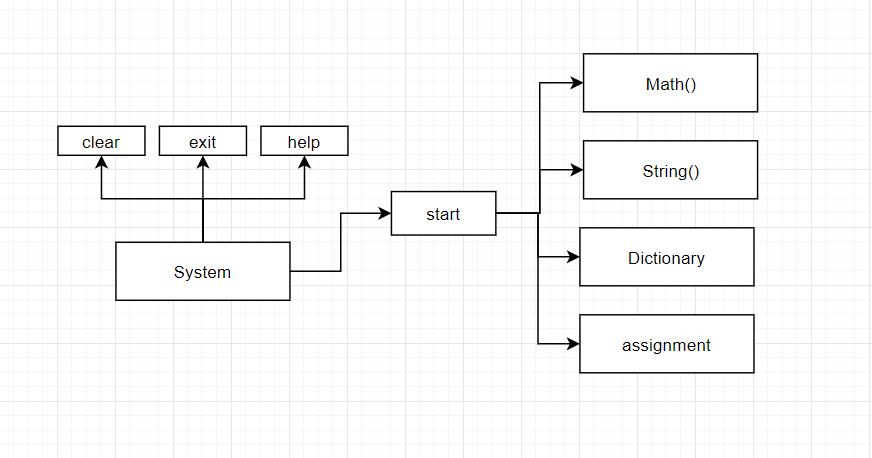
**4.2 Veri Tasarımı**

**4.2.1 Veri Tanımları**

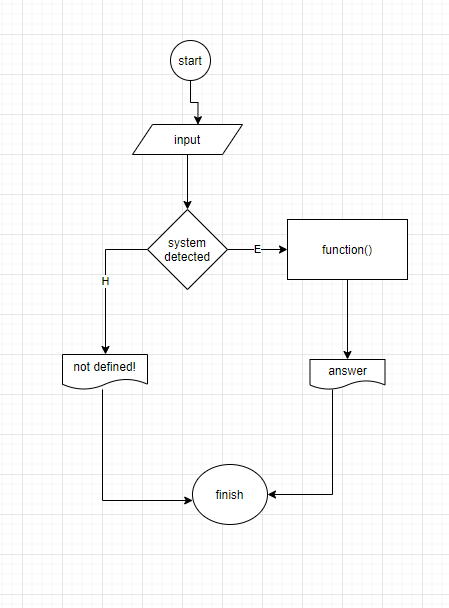
**4.2.2 Değer Kümesi Tanımları**

**Yetki:** Yetki kişinin bireyselliğine ya da kurumsallığına göre değişen bir değerdir**.**

**4.3 Süreç Tasarımı**



**Şekil 4.3.1.2 Modüller**

****

**Şekil 4.3.1.3 Yetkili Kullanıcı Tanımlama**

**4.3.2 Modüller**

**4.3.2.1 Giriş Modülü**

**4.3.2.1.1 İşlev**

Kullanıcının test analize erişmesi için aşması gereken bir parçadır.

**4.3.2.2 Kullanıcı Modülü**

Sistem içindeki tüm arabirimler aslında kendi içinde birer yönetim parçasıdır. Ara yüzler birbirinden kısmen farklıdır ama işlevler benzerdir.

**4.3.1 Kullanıcı Profilleri**

Kullanıcı: Yapının üçüncü nesnesidir.

**4.3.2 Entegrasyon ve Test Gereksinimleri**

Bu uygulama, mevcut sisteme bütünleşmiş halde çalışacaktır. Mevcut sistemde yapılabilen çoğu işlem masaüstünde uygulayabilecektir. Uygulamanın testi de pilot bölgelerde yetkili personeller tarafından yapılacaktır.

**4.4 Ortak Alt Sistemlerin Tasarımı**

**4.4.1 Ortak Alt Sistemler**

Kullanıcıların tüm ortak arayüzleri ve alt sistemleri kullanımları ortaktır.

**4.4.2 Modüller Arası Ortak Veriler**

Kullanıcıların modüler arası kullanımları aynı olduğu için ve sadece code sistemini uygulaması gerçekleştiği için kullanıcılara yönelik herhangi bir farklılık bulunmamaktadır.

**4.4.3 Güvenlik Alt Sistemi**

Yazılım sistemlerinin güvenilirliğe ilişkin nicelikleri, kullanıcıların gereksinimlerini karşılayacak şekilde ortaya koyma ve güvenilirliğin hesaplanmasına yönelik verileri toplama, istatistiksel tahmin etme, ölçütlerin tespiti, yazılıma ait mimari özelliklerin belirlenmesi, tasarım, geliştirme ve bunlara yönelik çalışma ortamının belirlenmesi ve modellenmesini kapsamaktadır.

1. Model seçme ve düzenlemeye yönelik faaliyetlerin temelinde uygun hedeflerin tespit edilmesi bulunmaktadır.

2. Hata ve aksaklıkların analiz edilmesi için uygun verilerin tanımlanması gerekmektedir. Örneğin, arıza veya hataları önemine göre sınıflandırmak, hatalar arası ortalama süreyi bulmak, hata nedenlerini araştırmak, hataları bulmaya yönelik test verilerine karar vermek vb.

3. Belirtilen hedeflere yönelik veriler modellenir.

4. Geçmişe yönelik veriler zaman bilgilerini de içerecek şekilde elde edilerek yazılım geliştirme sürecine dâhil edilir.

5. Yazılım geliştirme sürecinin modellenmesi, hata ile karşılaşılıp, test sürecine başlamak ve model doğrulama işlemlerini gerçekleştirmek gibi süreçler uygulanır.

6. Güvenilirlik tahmin modelinin seçilmesi sağlanır.

7. Güvenilirlik modeli tarafından kullanılacak olan parametreler tespit edilir.

8. Verilen bir noktayı kullanarak gelecekteki olası hatalar hakkında tahmin yapılır.

9. Tahmin edilen hata ve arıza oranları ile gerçekleşen değerler kıyaslanır.

**4.4.4 Veri Dağıtım Alt Sistemi**

Kullanıcılar sisteme girerken kendi bilgilerini giriş ekranına girdikten sonra sistem sunucudan gerekli verileri çekip karşılaştırdıktan sonra kullanıcılar sisteme giriş yapabileceklerdir. Kullanıcılar işlem yaptıktan sonra sistem yapılan işlemlere göre geri dönüş sağlayacaktır.

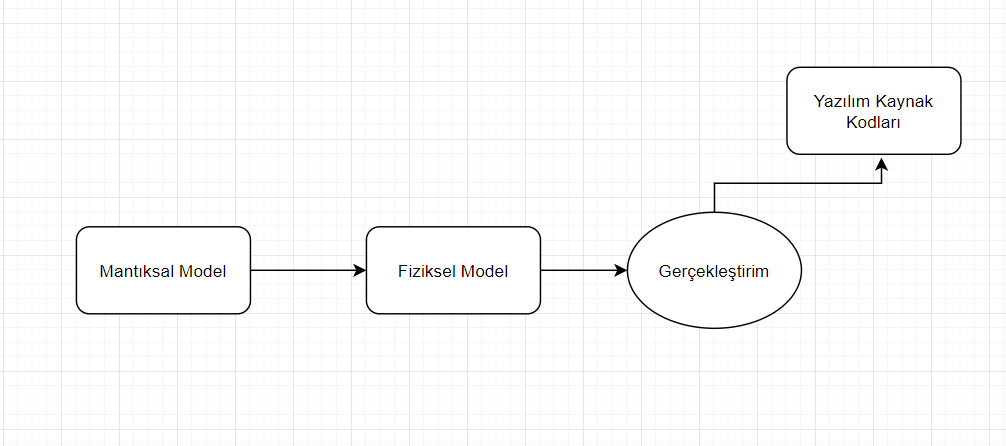
**4.4.5 Yedekleme ve Arşivleme İşlemleri**

Depolanan verilerin, herhangi bir nedenle zarar görmesi, sistemin çalışma süreçlerinde ciddi zararlar oluşturabilir. Yaşanabilecek bir felaket durumu sonrasında, depolanan verilerin geri yüklenememesi, sistemin sağlandığı kullanıcılara veya kurumlara çok ciddi zararlar verebilir. Bu nedenle sistemin çalışma süreçlerine bağlı olarak, yedekleme sistemleri kurulmalı ve yedekleme işlemleri günlük olarak takip edilmelidir. Yedekleme sistemlerinin kurulumu; yedeklenecek veri miktarı, yedekleme sıklığı, yedeklenen verinin zaman içerisinde değişme oranı ve maksimum veri kaybı gibi parametrelere bağlıdır. Sistemin birden fazla sunucusunun eş zamanlı yedekleme işlemini yapabilmesi, işletim sistemlerinin kayıt dosyalarını tam ve eş zamanlı olarak yedekleyebilmesi ve işletim sistemleri üzerinde çalışan veri tabanı uygulamasının yedeklerini sistem kapatılmadan alabilmesi gerekmektedir.

|  |
| --- |
|  |
| **5. SİSTEM GERÇEKLEŞTİRİMİ** |

**5.1 Giriş**

Gerçekleştirim bölümü, tasarım kısmında oluşturulan fiziksel modelin programlama dilleri ve araçları kullanılarak gerçeğe dönüştürülmesini içerir. Fiziksel modelin gerçekleştirimi için de en başta bu süreçlerin takibine yardımcı olacak bir yazılım geliştirme ortamı gerekmektedir.



**Şekil 5.1.1. Sistem Gerçekleştirim Modeli**

**5.2 Yazılım Geliştirme Ortamları**

Yazılım geliştirme ortamı, tasarım kısmında oluşturulan fiziksel modelin gerçeğe dönüştürülmesinde kullanılacak olan:

* Programlama Dilleri
* Tasarım Araçları

**5.2.1 Programlama Dilleri**

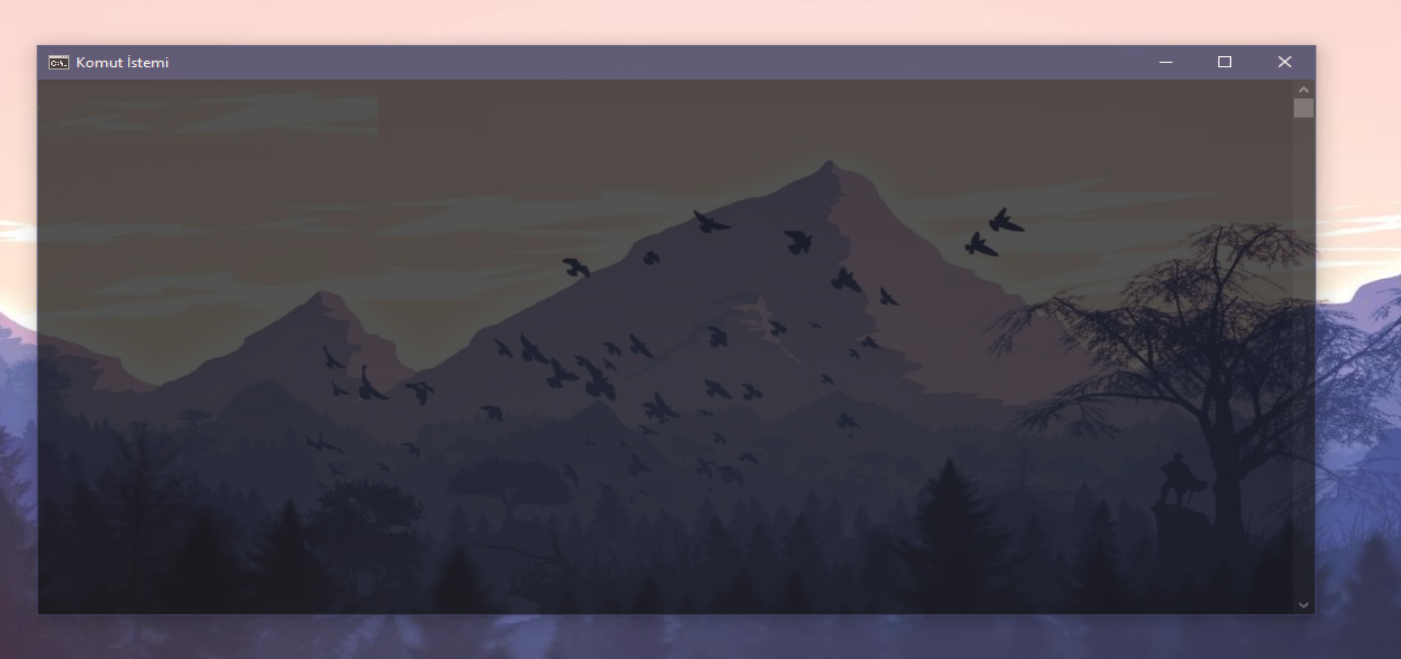
Bu sistemde kullanılacak programlama dili Python ‘dur. Hem nesne tabanlı oluşu, hem fonksiyonel programlama uyumluluğu hem de sistemin uyumlu kütüphaneleri bulunduğu için bu sistemde Python programlama dili seçildi.



**Şekil 5.2.1.1. Python Programlama Dili**

**5.2.2 Tasarım Araçları**

Tasarım aracı olarak tasarım konusunda çok fazla kodlamaya gerek duyulmadığı artı olarak hıza bağlı olabildiği için tasarım kısmı cmd panel üzerinden kodladık.



**Şekil 5.2.2.1. Cmd Panel**

**5.2.3.1 CASE Araç Ve Ortamları**



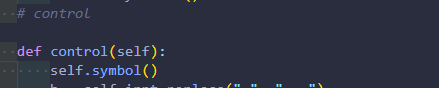
Sistemin UML diyagramlarını ve diğer diyagramları çizmek için Draw.İO kullanılmıştır.

**5.3 Kodlama Stili**

Sistemin kodlama işleminin daha kolay ve anlaşılabilir olması için sisteme özel belirli bir kodlama stili olması gerekmektedir. Kodlama stili 4 aşamadan oluşur.

**5.3.1 Açıklama Satırları**

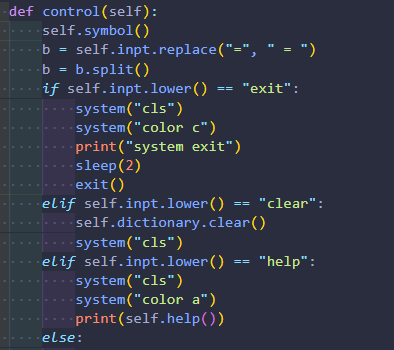
Program kod satırlarının arasına kodlarda düzenleme yapacak yetkili personellerin kullanması adına açıklama (yorum) satırları yerleştirilmiştir.



**5.3.2 Kod Biçimlemesi**

Programın okunabilirliğini artırmak amacıyla oluşturulmuş biçimlemedir.

Örnek:



**5.3.3 Anlamlı İsimlendirme**

Program kodlarındaki değişken isimleri Camel Case tipinde ve değişkenin yaptığı işi açıklayacak biçimde verilmiştir.



**5.3.4 Yapısal Programlama Yapıları**

Yapısal programlama, büyük çaptaki projelerin daha küçük parçalara ayrılarak kodlanması ve sorunlarının bu sayede daha hızlı giderilmesi yöntemidir. Bu yöntem, üç tür yapılanma kullanır:

* Ardışık Yürütme
* Şartlı Yürütme
* Döngü

**5.5 Olağan Dışı Durum Gözlemleme**

Olağan dışı durum, program çalışırken oluşabilecek istenmeyen durumlardır. Burada önemli olan, sistemin bu durumlar karşısında nasıl davranacağı ve kullanıcıyı nasıl yönlendireceğidir.

**5.5.1 Olağandışı Durum Tanımları**

* İşlemlerde zaman aşımı
* Bilgisayar bir anda kapatılması

**5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları**

Sistemi çalıştırdığınızda sizin onayınız ile verdiğiniz kodu analiz etmeye başlar. Eğer sistemdeki fonksiyonda istenilen kod bulunmuyorsa size hatalo bir girdi yaptığınızı belirtiyor. Eğer işleminiz sonlandırmak exit yazıp enter basmanız yeterlidir.

**5.6 Kod Gözden Geçirme**

Yazılan kodların düzgün, tutarlı ve hatasız bir şekilde çalışabilmesi için gözden geçirilmesi gerekmektedir. Gözden geçirilmezse oluşabilecek hatalardan doğacak sorunlar veri kaybına sebep olabilir.

**5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi**

Kod yazma işlemi tamamlandığında, kod gözden geçirilmeye hazır demektir. Bunun için gerekli ortam hazırlanır, yazım hataları giderilir, test senaryoları düzenlenir ve gözden geçirme işlemine başlanır.

**5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular**

Gözden geçirme sırasında sorulacak soruların bazıları şunlardır:

* Kullanıcı arayüzleri kullanıcının kolaylıkla kullanabileceği şekilde mi tasarlandı?
* Kullanıcılar rahat bir şekilde kullanabilecek mi?
* Hangi işletim sistemleri ile çalışabiliyor?
* Program olası bir hata durumunda ne tür bir davranış sergileyecek?

**5.6.2.1 Öbek Arayüzü**

Oluşturulan öbeklerin (kod bütünü) testi ve doğrulaması için sorulan belirli sorular vardır. Bunlar:

* Her öbek tek bir işlevsel amacı yerine getiriyor mudur?
* Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş midir?
* Öbek tek girişli ve tek çıkışlı mıdır?
* Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mudur?

**5.6.2.2 Giriş Açıklamaları**

Sistemin doğru anlaşılabilmesi için başta giriş açıklamaları oluşturmuştur. Bu açıklamaları test etmek ve doğrulamak için belli sorular sorulmuştur. Bunlar:

* Öbek, doğru biçimde giriş açıklama satırları içeriyor mudur?
* Giriş açıklama satırları, öbeğin amacını açıklıyor mudur?
* Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mudur?
* Giriş açıklama satırları, çıktıları ve hata iletilerini tanımlıyor mudur?
* Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mudur?

**5.6.2.3 Veri Kullanımı**

Oluşturulan veri kullanımlarının testi için belirli sorular sorulmuştur. Bunlar:

* İşlevsel olarak ilişkili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mıdır?
* Değişken adları, işlevlerini yansıtacak biçimde anlamlı mıdır?
* Değişkenlerin kullanımları arasındaki uzaklık anlamlı mıdır?
* Her değişken tek bir amaçla mı kullanılıyor?
* Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?
* Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mıdır?

**5.7.2.4 Öbeğin Düzenlenişi**

Oluşturulan modellerin düzenlenişi için belirli sorular sorulmuştur. Bunlar:

* Modüllerin birleşimi uyumlu mudur?
* Modüller arası veri aktarımları sağlanıyor mudur?
* Bütün parçalar birleştiğinde sistem çalışıyor mudur?

**5.6.2.5 Sunuş**

Sunuş kısmında ise şu sorular sorulmaktadır:

* Her satır, en fazla bir deyim mi içermektedir?
* Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mıdır?
* Koşullu deyimlerde kullanılan mantıksal işlemler yalın mıdır?
* Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mıdır?
* Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mıdır?
* Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mıdır?

|  |
| --- |
| **6.** **DOĞRULAMA VE GEÇERLEME** |

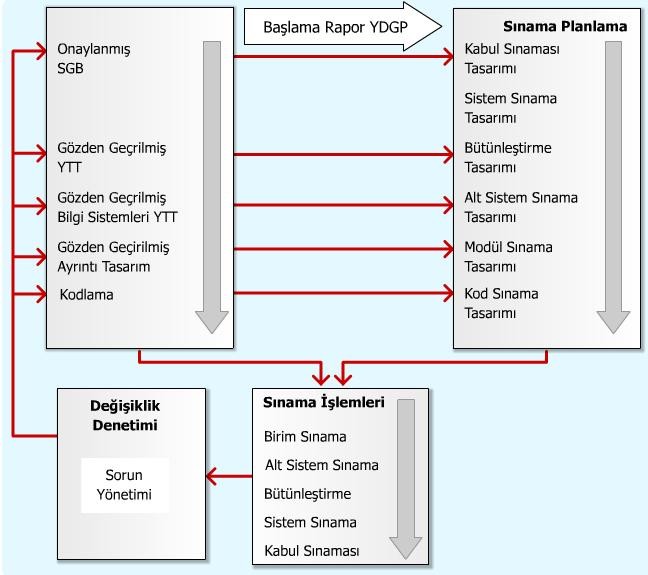
**6.1 Giriş**

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

* + - Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.
    - Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması. Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.
    - Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklemesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.



*Şekil 43Doğrulama ve Geçerlem**e*

**6.3 Kabul Doğrulama ve Geçerleme Yaşam Döngüsü**

**6.4 Sınama Yöntemleri**

**6.4.1 Beyaz Kutu Sınaması**

|  |  |  |
| --- | --- | --- |
| Sınama Noktası | Durum | Sınamayı Yapan |
| Bütün bağımsız yolların en azından bir kere sınanması, |  | BadourAlkhateb |
| İç veri yapılarının denenmesi |  |
| Bütün döngülerin sınır değerlerinde sınanması |  |
| Bütün mantıksal karar noktalarında iki  değişik karar için sınamaların yapılması, |  |

**6.5 Sınama Planlaması**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ZAMAN  İŞ | Proje Başlangıç  Tarihi:01/01/2021 | | |  |  |  |  |  |  |  | HAFTALAR | | | |  |  |
| **3. Hafta** | | | | | | **4.Hafta** | | | | | | | | |  |
| Sistem  Doğrulanması | **Birim Sınama** | | | | | **Alt Sistem Sınama** | | | | **Sistem Sınama** | | | **Kabul Sınama** | | |  |
| 1 | 2 | 3 | 4 | 5 | 6 | | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |
| 1-2 gün: iki\_dilli eklentisi 3 gün: resimli\_secenek ve tamamlama | | | | | iki\_dilli eklentisi ayarlar arayüzdeki  karakter sorunu | | | | Sistem yük, stres, performans testi uygulandı. | | | UAT (Kullanıcı Kabul Testi) yapıldı. Sistem  protipten çıkarıldı. | | |  |

**6.6 Sınama Belirtileri**

|  |  |  |
| --- | --- | --- |
| **YAZILIM SINAMA RAPORU** | | |
| **Yazılım Tanımı** |  |  |
| Programcı Ad- Soyad | BadourAlkhateb |
| Programlama  Dili | Python |
| Bilgisayar Sistemi | Intel i7 |
| İşletim Sistemi | Windows10 |
| Sınama Türü | Beyaz Kutu |
| **Yazılım Tasarımı** | Problemin Tanımlanması | Python ile kodlamasıyla birden fazla dil ayarları aktif etmesinden sonra bir sorunlar karşılaşırsa raporlanacaktır. |
| Problemin  Çözümü | Python, UTF8 Full destek için test.py dosyasında yapılan ayarlama ile sorun çözüldü. |
| **Sonuçların Testi** | | Test sonucu ilgili kod sunucudaki dosyaya eklendi ve sorun çözüldü. |

**Yaşama Döngüsü Boyunca Sınama Etkinlikleri**

PLANLAMA

Sistem Sınama

Planı

ÇÖZÜMLEME

TASARIM

GERÇEKLEME

KURULUM

Alt Sistem

Sınama Planı

Modül Sınama

Planı

Sınama Belirtimleri

Sınama Eğitim Kılavuzları

Modül Sınama

Bütünleştirme

Sınama

Sınayıcı Eğitim

Kullanıcı

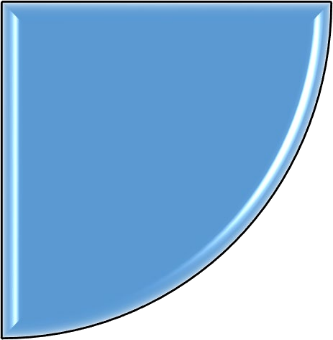
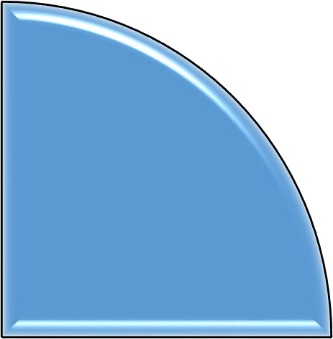
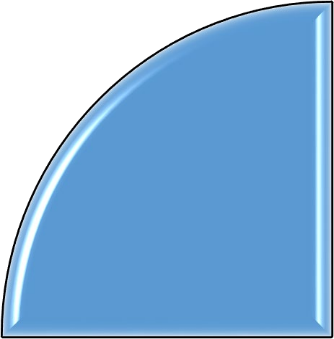
Sınama

Sınama Raporları

|  |
| --- |
| **7. BAKIM** |

**7.1 Giriş**

**Gelecekte yazılımın sürdürülebilirliği**



**Hata Bulma ve ve güvenirliği arttırma**

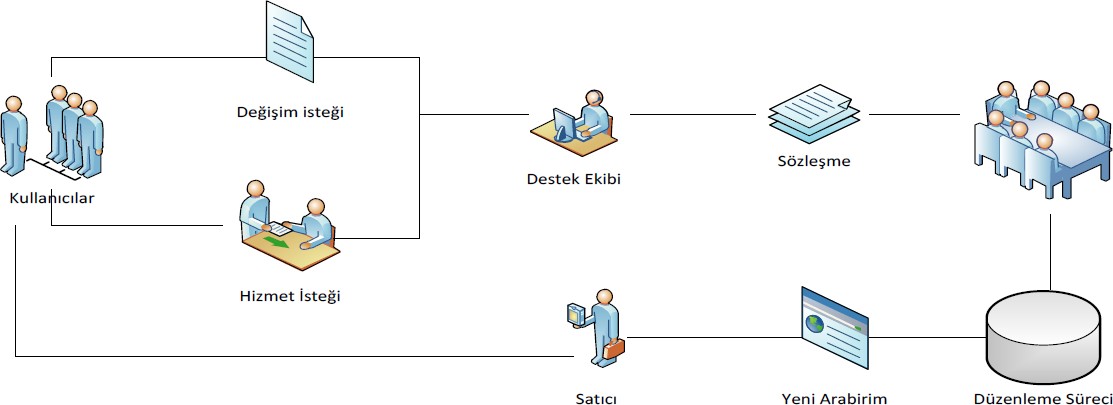
**ve düzeltme**

**Bakım Bakım**

**Uyumlaştırıcı Mükemmelleştirişi**

**Bakım Bakım**

**Yazılım değişikliği baş etme ve değiştirme Yazılım fonksiyonel zenginlik katma**

****

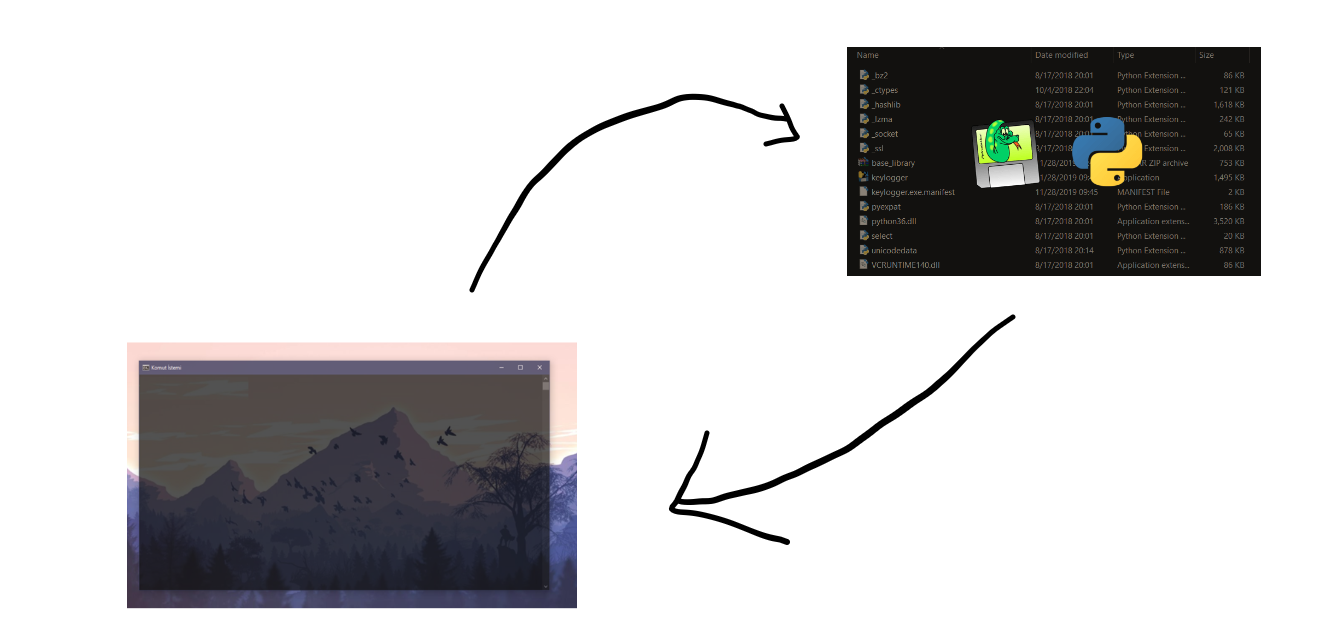
**•**Projemize ait parçaların “Düzeltici Bakımı” Doğrulama ve Geçerleme sürecinde gerçekleştirilen testlerle yapıldı. Diğer tür bakımlar ise; Python’ un sürüm yükseltilmesi ile gerçekleşmiş olacak. Bu konuda parçaların kodlamasında bakım ile ilgili yapmamız gereken ekstra bir işlem bulunmamaktadır.

•Modüllerin sağlıklı şekilde kurulumu, sürüm kontrolü, logging işlemleri tümüyle Python altyapısının sağladığı yönetim araçlarıyla gerçekleşmektedir.

**7.2 Kurulum**

Projenin kurulumu temel 2 safhadan oluşur:

* Standart Python kurulumu
* Proje parçalarının standart Windows üzerine kurulumu



*Şekil Projenin Kurulumu*

**7.3 Yerinde Destek Organizasyonu**

Proje Ekibi, tamamen üniversitemiz öğrencilerinden oluşmaktadır. Kurulumu müteakip, ihtiyaç duyulacak anlarda sisteme müdahale imkânı vardır. Uygulama alt yapısı, uzaktan destek imkânını da kolaylaştırmaktadır.

**7.4 Tanım**

Bir yazılımın bakımı(software maintenance), yazılımın geliştirilmesi tamamlandıktan sonraki süreçte, yazılımın ömrünü sağlıklı bir şekilde sürdürmesini sağlayan yaklaşımlar bütünüdür. Yazılım bakımı Yazılım Geliştirme Sürecinin(Software Development Process) son parçasını oluşturur. Müşteri gereksinimlerine göre oluşturulan yazılım çıktısı, değişen teknolojik araçlara ve müşteri gereksinimlerine göre sürekli bakımının yapılması, denetlenmesi gerekir. Nasıl bir arabanın yıllık bakımı yapılıyorsa, yazılım ürünün de belirli sürümler dâhilinde geliştirilmesi, iyileştirilmesi ve korunması yapılmalıdır.

**7.5 Bakım Süreç Modeli**

IEEE 1219-1998 Yazılım Bakım Standardı esaslarına göre sağlanacaktır

|  |
| --- |
| **8.** SONUÇ |

Artık bu uygulama tüm testlerden geçtikten sonra ortaya çıkmış oldu. Kullanıcılar rahat bir şekilde indirip kullanmaya başlayacaklar.

|  |
| --- |
| **9. KAYNAKLAR** |

\*kynk001:

<https://www.startupnedir.com/wp-content/uploads/2018/04/scrum-nedir-780x405.jpg>

\*kynk002:

<https://slideplayer.biz.tr/slide/12237711/72/images/14/%C3%87evik+%28Agile%29+Yaz%C4%B1l%C4%B1m+S%C3%BCre%C3%A7+Modelleri.jpg>

\*kynk003:

<https://sites.google.com/site/murathazer/ScrumOverviewDiagram.png>