

**KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ**

**MİMARLIK VE MÜHENDİSLİK FAKÜLTESİ**

Bilgisayar Mühendisliği

BİL306 - Yazılım Mühendisliği Dersi

Proje Uygulamaları ve Dokümantasyonu

**Proje Yürütücüleri**Mehtap ÖKLÜ

HAZİRAN –2021

| **1. GİRİŞ** | 5 |
| --- | --- |
| 1.1 Projenin Amacı | 5 |
| 1.2 Projenin Özeti & Kapsamı | 5 |
| 1.3 Tanımlama ve Kısaltma | 5 |
|  |  |
| **2. PROJE PLANI** | 5 |
| 2.1 Giriş | 5 |
| 2.2 Projenin Plan Kapsamı | 6 |
| 2.3 Proje Zaman-İş Planı | 8 |
| 2.4 Proje Ekip Yapısı | 8 |
| 2.5 Önerilen Sistemin Teknik Tanımları | 9 |
| 2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları | 10 |
| 2.7 Proje Standartları, Yöntem ve Metodolojiler | 10 |
| 2.8 Kalite Sağlama Planı | 13 |
| 2.9 Konfigürasyon Yönetim Planı | 15 |
| 2.10 Kaynak Yönetim Planı | 15 |
| 2.11 Sistem Eğitimi | 15 |
| 2.12 Test Planı | 16 |
| 2.13 Bakım Planı | 16 |
|  |  |
| **3. SİSTEM ÇÖZÜMLEME** | 17 |
| 3.1 Mevcut Sistem İncelemesi | 17 |
| 3.1.1 İşlevsel Model | 17 |
| 3.1.2 Veri Modeli | 18 |
| 3.1.3 Var Olan Yazılım/Donanım Kaynakları | 19 |
| 3.1.4 Var Olan Sistemin Değerlendirilmesi | 22 |
| 3.2 Gereksenen Sistemin Mantıksal Modeli | 23 |
| 3.2.1 Giriş | 23 |
| 3.2.2 İşlevsel Model | 23 |
| 3.2.3 Genel Bakış | 25 |
| 3.2.4 Bilgi Sistemleri/Nesneler | 25 |
| 3.2.5 Veri Modeli | 27 |
| 3.2.6 Veri Sözlüğü | 28 |
| 3.2.7 İşlevlerin Sıradüzeni | 29 |
| 3.2.8 Başarım Gerekleri | 30 |
| 3.3 Arayüz (Modül) Gerekleri | 30 |
| 3.3.1 Yazılım Arayüzü | 30 |
| 3.3.2 Kullanıcı Arayüzü | 30 |
| 3.3.3 İletişim Arayüzü | 31 |
| 3.4 Belgeleme Gerekleri | 32 |
| 3.4.1 Geliştirme Sürecinin Belgelenmesi | 32 |
| 3.4.2 Sıkça Sorulan Sorular | 32 |
|  |  |
| **4. SİSTEM TASARIMI** | 32 |
| 4.1 Genel Tasarım Bilgileri | 32 |
| 4.1.1 Genel Sistem Tanımı | 32 |
| 4.1.2 Kısıtlamalar | 34 |
| 4.1.3 Sistem Mimarisi | 35 |
| 4.1.4 Dış Arabirimler | 36 |
| 4.1.4.1 Kullanıcı Arabirimleri | 36 |
| 4.1.4.2 Veri Arabirimleri | 36 |
| 4.1.5 Veri Modeli | 37 |
| 4.1.6 Testler | 38 |
| 4.1.7 Performans | 38 |
| 4.2 Veri Tasarımı | 38 |
| 4.2.1 Tablo Tanımları | 38 |
| 4.2.2 Tablo - İlişki Şemaları | 39 |
| 4.2.3 Veri Tanımları | 39 |
| 4.2.4 Değer Kümesi Tanımları | 39 |
| 4.3 Süreç Tasarımı | 40 |
| 4.3.1 Genel Tasarım | 40 |
| 4.3.2 Kullanıcı Arayüz Girişi | 41 |
| 4.3.2.1 Kullanıcı1 & Admin2 Girişi | 41 |
| 4.3.2.2 Kullanıcı2 Girişi | 41 |
| 4.3.3 Kullanıcı Profilleri | 42 |
| 4.3.4 Entegrasyon ve Test Gereksinimleri | 42 |
| 4.4 Ortak Alt Sistemlerin Tasarımı | 43 |
| 4.4.1 Ortak Alt Sistemler | 43 |
| 4.4.2 Modüller Arası Ortak Veriler | 43 |
| 4.4.3 Güvenlik Alt Sistemi | 43 |
| 4.4.4 Veri Dağıtım Alt Sistemi | 43 |
| 4.4.5 Yedekleme ve Arşivleme İşlemleri | 43 |
|  |  |
| **5. SİSTEM GERÇEKLEŞTİRİMİ** | 44 |
| 5.1.Giriş | 44 |
| 5.2.Yazılım Geliştirme Ortamları | 44 |
| 5.2.1 Programlama Dilleri | 45 |
| 5.2.2 Veri Tabanı Yönetim Sistemleri | 45 |
| 5.2.2.1 VTYS Kullanımının Ek Yararları | 46 |
| 5.2.2.2 Veri Modelleri | 46 |
| 5.2.2.3 Şemalar | 48 |
| 5.2.2.4 VTYS Mimarisi | 48 |
| 5.2.2.5 Veritabanı Dilleri ve Arabirimleri | 49 |
| 5.2.2.6 Veri Tabanı Sistem Ortamı | 49 |
| 5.2.2.7 VTYS'nin Sınıflandırılması | 49 |
| 5.2.2.8 Hazır Program Kütüphane Dosyaları | 49 |
| 5.2.2.9 CASE Araç ve Ortamları | 49 |
| 5.3. Kodlama Stili | 49 |
| 5.3.1 Açıklama Satırları | 50 |
| 5.3.2 Kod Biçimlemesi | 50 |
| 5.3.3 Anlamlı İsimlendirme | 50 |
| 5.3.4 Yapısal Programlama Yapıları | 50 |
| 5.4. Program Karmaşıklığı | 50 |
| 5.5. Olağan Dışı Durum Çözümleme | 51 |
| 5.5.1 Olağandışı Durum Tanımları | 51 |
| 5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları | 51 |
| 5.6. Kod Gözden Geçirme | 52 |
| 5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi | 52 |
| 5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular | 52 |
| 5.6.2.1 Öbek Arayüzü | 52 |
| 5.6.2.2 Giriş Açıklamaları | 52 |
| 5.6.2.3 Veri Kullanımı | 53 |
| 5.6.2.4 Öbeğin Düzenlenişi | 53 |
| 5.6.2.5 Sunuş | 53 |
|  |  |
| **6. DOĞRULAMA VE GEÇERLEME** | 54 |
| 6.1 Giriş | 54 |
| 6.2 Sınama Kavramları | 54 |
| 6.3 Doğrulama ve Geçerleme Yaşam Döngüsü | 55 |
| 6.4 Sınama Yöntemleri | 55 |
| 6.4.1 Beyaz Kutu Sınaması | 55 |
| 6.4.2 Temel Yollar Sınaması | 56 |
| 6.5 Sınama ve Bütünleştirme Stratejileri | 56 |
| 6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme | 57 |
| 6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme | 57 |
| 6.6 Sınama Planlaması | 57 |
| 6.7 Sınama Belirtimleri | 58 |
| 6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri | 58 |
|  |  |
| **7. BAKIM** | 60 |
| 7.1 Giriş | 60 |
| 7.2 Kurulum | 61 |
| 7.3 Yerinde Destek Organizasyonu | 61 |
| 7.4 Yazılım Bakımı | 61 |
| 7.4.1 Tanım | 61 |
| 7.4.2 Bakım Süreç Modeli | 62 |
|  |  |
| **8. SONUÇ** | 66 |
| **9. KAYNAKLAR** | 66 |

1. **GİRİŞ**

**1.1 Projenin Amacı**

Alerjisi olan insanların çok fazla sıra beklemeden ve canları acımadan sadece enzimler yardımı ile neye alerjisini olduğunu bulma.

**1.2 Projenin Özeti& Kapsamı**

İnsanlar alerji testleri için oldukça çok vakit harcıyor. Onun yanında vücutlarına birden çok iğne batırılmak zorunda kalıyor. Hem zaman tasarrufu yapmak için hem de daha az acıyla test yaptırmak için bir proje.

Projede kişi veya kişiler hastaneye alerji testi başvurusu için öncelikle bilgilerini verir, kayıt yaptırır. Daha sonra ilgili yere gider. Test yapılacak kişinin ağzına tıbbi bir kâğıt yerleştirilir. Bekleme süresinden sonra kâğıt ilgili cihaza yerleştirilir. Cihaz enzimleri ayrıştırır, kişinin neye alerjisi olduğunu karşılaştırma yolu ile bulur. Sonucu doktora gönderir. Doktor sisteme giriş yapar, kontrolü sağlar ve bu sonuçlar doğrultusunda gerekli tedavilere başlar.Ayrıca hasta daha sonra sistemin web sitesi üzerinden sonuçlarını tekrar görüntüleyebilir.

Kapsamı:

* Özel ve devlet hastaneleri
* Sağlık ocakları
* Aile hekimleri
* Tıp merkezleri

**1.3 Tanımlama & Kısaltma**

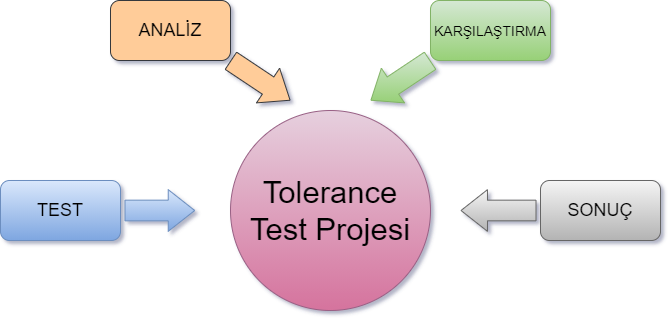
* **TT Machine:** Tolerance Test, tıbbi kâğıt kontrol makinesi

1. **PROJE PLANI**

**2.1 Giriş**

Hastanın ağzından alınan tükürük içerisindeki enzimlere yapılan testler sonucunda ortaya bir veri çıkar. Bu veri veritabanına işlenir ve veritabanındaki veriler ile karşılaştırılır. Çıkan sonuca göre hastanın neye alerjisi olduğu bulunmuş olunur.

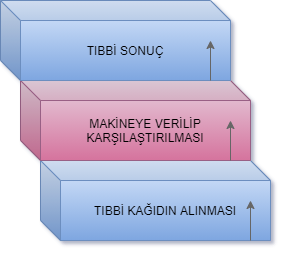
Şekil 2.1 – Projenin Genel Yapısı



**2.2 Projenin Plan Kapsamı**

Projenin plan kapsamında genel olarak alışılageldik alerji testi sisteminin hastaya verdiği rahatsızlıktan yola çıkıldı. Hem zamandan tasarruf edip hem de hastaya en az düzeyde rahatsızlık yaşatarak alerji testi yapılmasını sağlar. Bu sistem ile hastanın bilgileri bilgisayar ortamında tutulduğu için daha sonra koyulacak teşhislerde de kolaylık sağlar.

Şekil 2.2 – Projenin Genel İşleyişi

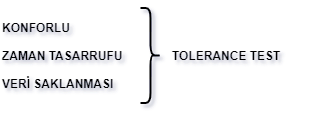


**TOLERANCE TEST NEDEN GEREKLİ?**

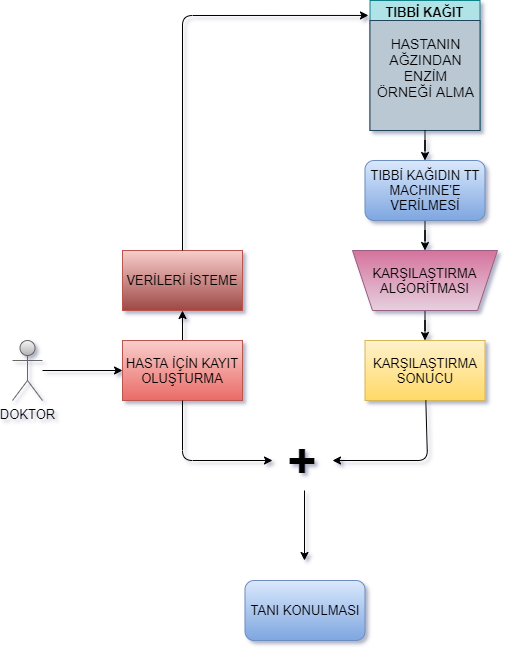
* Alışılageldik alerji testlerine göre daha konforlu olduğu için
* Kişi veya kişilere zaman tasarrufu sağladığı için,
* Yapılan testlerin sonuçlarının kaybolmasını önlediği için,

Bu projenin başta hastane olmak üzere diğer tıp merkezlerinde de kullanılmasını gerekli kılmıştır.

Şekil 2.3 – Tolerance Test Avantajları



Şekil 2.4 – Tolerance Test Alerji Testi Nasıl Yapılır



**TOLERANCE TEST ALTYAPISI NASIL ÇALIŞIR?**

* Hasta alerji testi yaptırmak için gerekli sağlık kuruşuna başvuru yapar
* Hastanın bilgileri alınır ve sistemde yeni bir kayıt oluşturulur.
* Eğer hastanın daha önceden bir kaydı varsa mevcut kayıt güncellenir
* Hastanın ağzından tükürük örneği alınır ve test cihazına yerleştirilir.
* Test cihazı hastanın neye alerjisi olduğunu bulur ve sisteme kaydeder.
* Başvurulan sağlık çalışanı tedaviye başlamak için sonuçları hastaya bildirir.

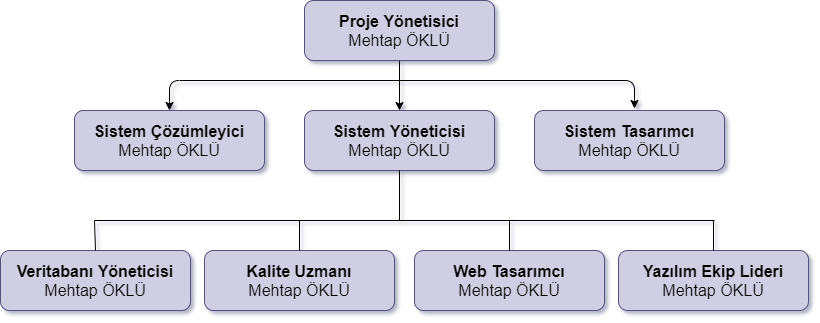
**2.3 Projenin Zaman-İş Planı**

| **İŞ - ZAMAN ÇİZELGESİ** | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ZAMAN  İŞ | **1. Hafta** | **2. Hafta** | **3. Hafta** | **4. Hafta** | **5. Hafta** | **6. Hafta** | **7. Hafta** | **8. Hafta** | **9. Hafta** | **10. Hafta** |
| **Proje Teklifi** | C:\Users\SAMED\Desktop\tik2.png |  |  |  |  |  |  |  |  |  |
| **Proje Planı** |  | C:\Users\SAMED\Desktop\Untitled-2.png | C:\Users\SAMED\Desktop\tik2.png |  |  |  |  |  |  |  |
| **Analiz** |  |  | C:\Users\SAMED\Desktop\Untitled-2.png | C:\Users\SAMED\Desktop\tik2.png |  |  |  |  |  |  |
| **Sistem Çözümleme** |  |  |  | C:\Users\SAMED\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled-3.png | C:\Users\SAMED\Desktop\Untitled-4.png | C:\Users\SAMED\Desktop\tik2.png |  |  |  |  |
| **Arayüz Tasarımı** |  |  |  |  |  | C:\Users\SAMED\Desktop\Untitled-2.png | C:\Users\SAMED\Desktop\tik2.png |  |  |  |
| **Gerçekleştirim** |  |  |  |  |  |  | C:\Users\SAMED\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled-3.png | C:\Users\SAMED\Desktop\Untitled-4.png | C:\Users\SAMED\Desktop\tik2.png |  |
| **Test** |  |  |  |  |  |  |  |  | C:\Users\SAMED\Desktop\tik2.png |  |
| **Sunum** |  |  |  |  |  |  |  |  |  | C:\Users\SAMED\Desktop\tik2.png |

Şekil 2.5 – Proje İş-Zaman Çizelgesi

**2.4 Proje Ekip Yapısı**

Şekil 2.6 – Proje Ekibi



Şekil 2.7 – Proje Ekip Görevleri



**2.5 Önerilen Sistemin Teknik Tanımları**

**Kullanılan Teknoloji**

* SHA-256 şifreleme yöntemi kullanılır.
* Şifreli verilerin saklandığı veritabanı
* TT Machine

Tıbbi enzim kâğıdı

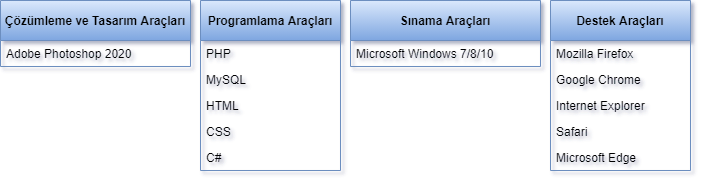
* Steril olarak tükürük enzimlerini korur.
* Tek kullanımlıktır.
* Test cihazına sokulmaya uygundur.

Test makinesi

* Karşılaştırma yapabilmesi için tıbbi kâğıtta yeterli enzim olup olmadığını kontrol eder.
* Yeterli enzim varsa analiz eder ve karşılaştırma yapar.
* Karşılaştırmaya göre tanı koyar.
* Bu işlemler bütününü sisteme kaydeder.

**2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları**

Şekil 2.8 – Kullanılan Araç ve Ortamlar



**2.7 Proje Standartları, Yöntem ve Metodolojiler**

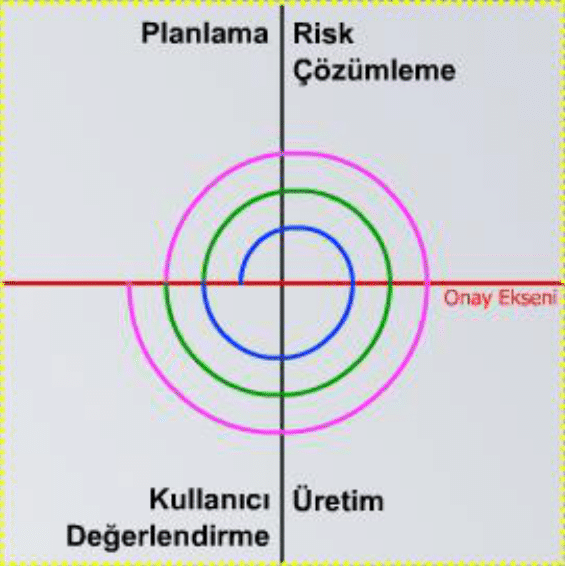
Spiral döngüyü diğer modellerden ayıran özellik risk analizinin ön planda olması ve prototip oluşturulmasıdır. Risk analizi ön planda olduğu için hataları erken giderme imkânı tanıyabilir. Spiralin başladığı ilk çeyrek içinde ilk istekler toplanır ve buna göre proje planlaması yapılır. İkinci çeyrekte, ilk tanımlanan isteklere göre risk çözümlemesi yapılır. Üçüncü çeyrekte, risk çözümlemesi sonunda ortaya çıkan isteklerin tanımlanmasındaki belirsizlikleri ortadan kaldırmak için prototipleme yöntemi kullanılır. Gerekirse benzetimveya diğer modelleme kullanılarak isteklerin daha sağlıklı tanımlanması sağlanır. Dördüncü çeyrekte, hastalar, ortaya çıkan ilk ürünü inceleyerek değerlendirme yapar, önerilerde bulunur. Bu şekilde tanımlanan ilk döngü bir sonraki döngü için bir girdi oluşturur.

| **Aşama** | **Kullanılan Yöntem & Araçlar** | **Ne İçin Kullanıldığı** | **Çıktı** |
| --- | --- | --- | --- |
| **Planlama** | * Veri Akış Şemaları | Süreç İnceleme | Proje Planı |
| * Süreç Belirtimleri | Kaynak Kestirimi |
| * Görüşme | Proje Yönetimi |
| * Maliyet Kestirim Yöntemleri |  |
| * Proje Yönetim Araçları |  |
| **Çözümleme** | * Süreç Belirtimleri | Süreç Çözümleme | Sistem Çözümleme Raporu |
| * Veri Akış Şemaları | Veri Çözümleme |
| * Görüşme |  |
| * Nesne-İlişki Şemaları |  |
| * Veri Sözlüğü |  |
| **Çözümlemeden Tasarıma Geçiş** | * Akışa Dayalı Çözümleme | Başlangıç Tasarım | Başlangıç Tasarım Raporu |
| * Süreç Belirtimlerinin Program Tasarım Diline Dönüştürülmesi | Ayrıntılı Tasarım |
| * Nesne-İlişki Şemalarının Veri Tablolarına Dönüştürülmesi | Başlangıç Veri |
|  | Tasarım |
| **Tasarım** | * Yapısal Şemalar | Genel Tasarım | Sistem Tasarım Raporu |
| * Program Tasarım Dili | Ayrıntılı Tasarım |
| * Veritabanı Tabloları | Veri Tasarımı |
| * Veri Sözlüğü |  |

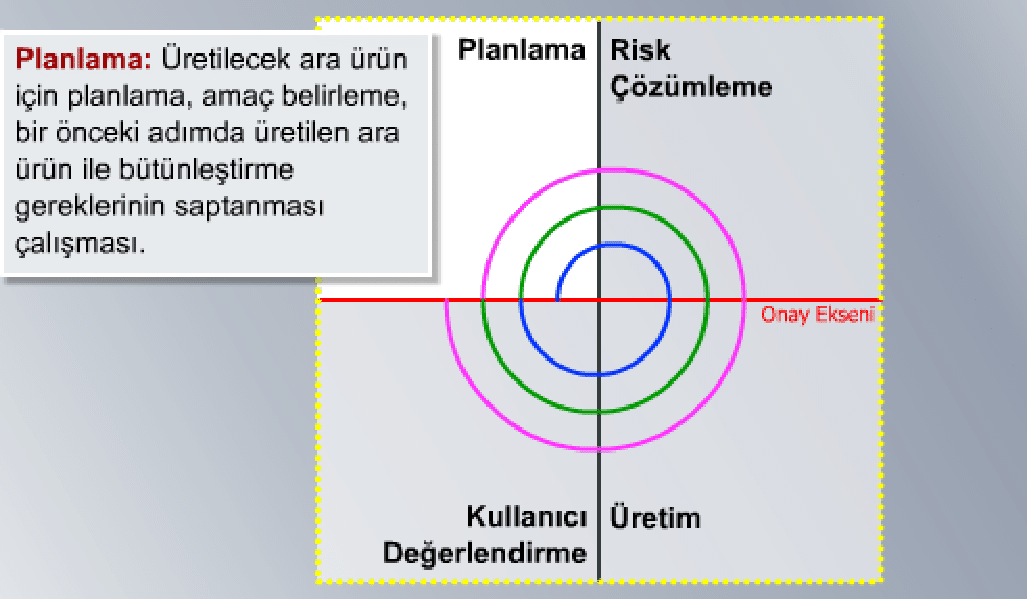
Şekil 2.9 – Proje Aşamaları

Proje standartları yukarda belirtildiği gibidir. Bunun yanı sıra kullanılan sistem modeli ise Spiral (helezonik) modeldir.

Şekil 2.10 – Proje Gelişim Modeli

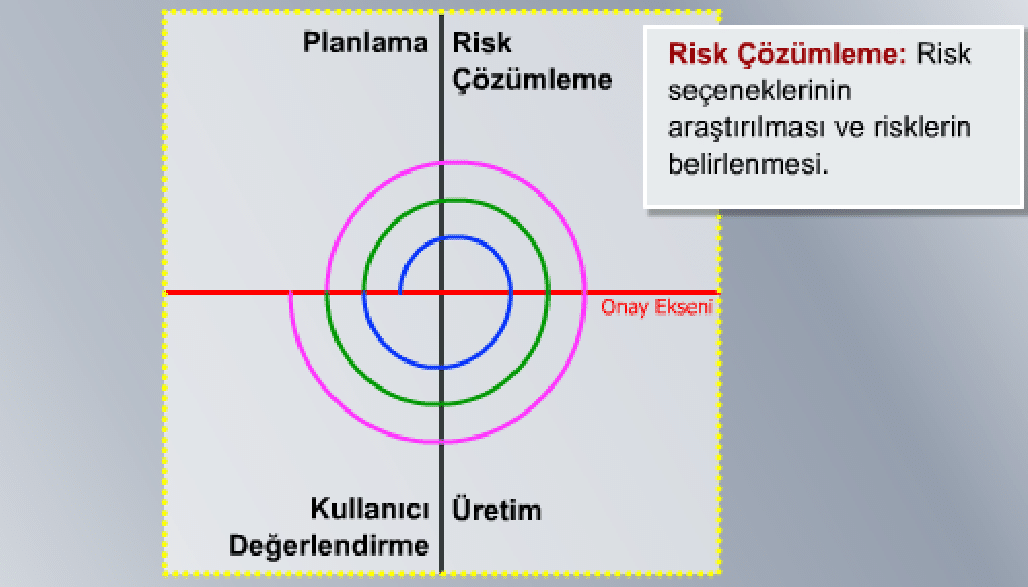


Şekil 2.11 – Planlama Aşaması

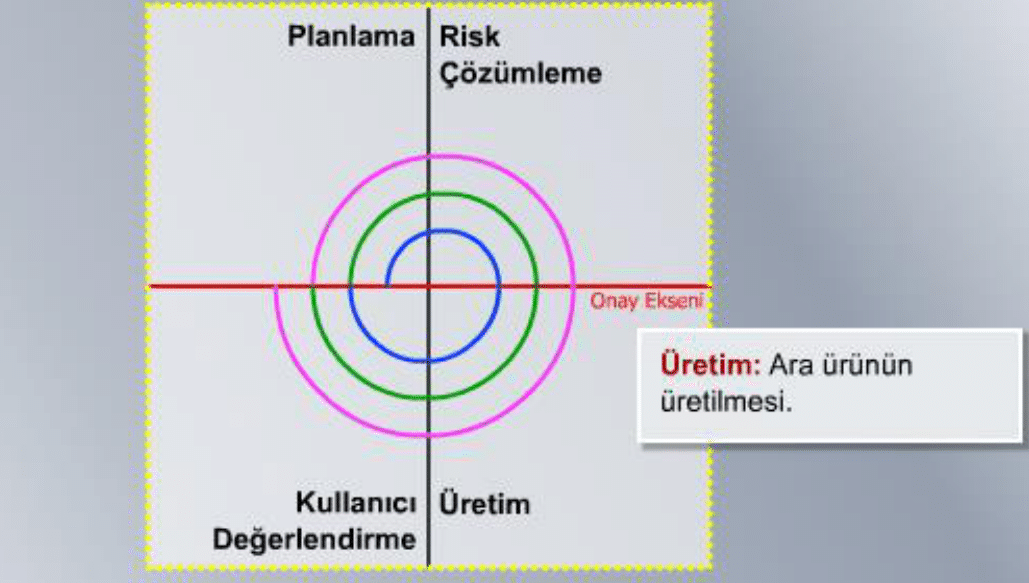


Tablo ve resimlerde belirttiğimiz aşamalar yazılım geliştirmenin önemli unsurlarındandır. Kullandığımız geliştirme modeli Spiral (helezonik) model olup kullanmamızın amacı döngü şeklinde kontrol edip hataları en aza indirgemektir.

Şekil 2.12 – Proje Risk Çözümleme Aşaması

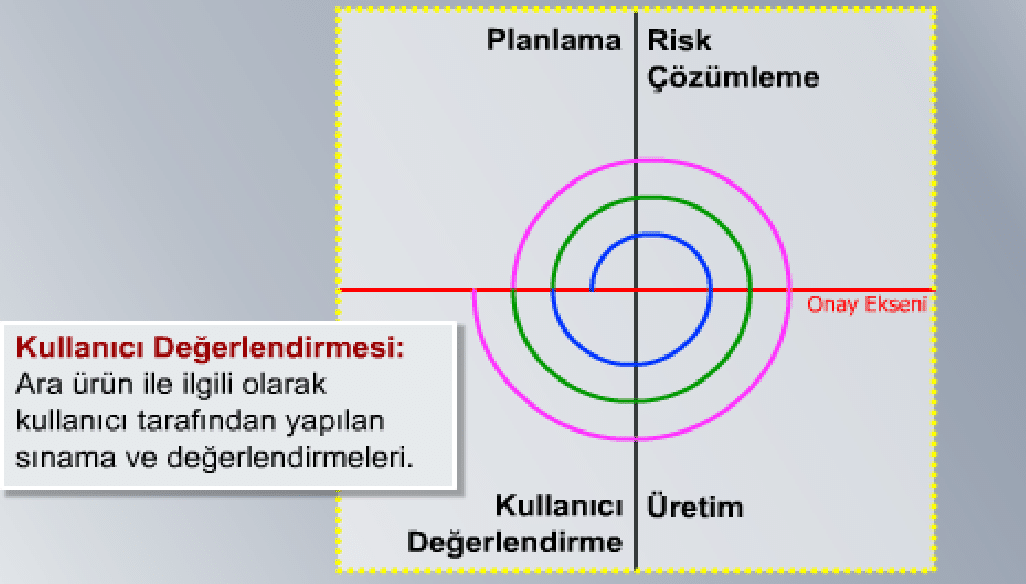


Şekil 2.13 – Proje Üretim Aşaması



Projenin üretimi oldukça uzun zaman alır. Proje tasarımına yer verilir ve projenin büyük çoğunluğu bitmiş olur. Bu aşamada ara ürün üretilir ve proje bu aşamayla birlikte büyük bir yol kat etmiştir.

Şekil 2.14 – Kullanıcı Değerlendirme Aşaması



| **EKİP YAPISI ZAMAN-İŞ PLANI** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
|  | **TEMMUZ** | | **AĞUSTOS** | | **EYLÜL** | |
| **SİSTEM ÇÖZÜMLEYİCİ** | **C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif** | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif |  |  |  |  |
| **SİSTEM YÖNETİCİSİ** | **C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif** | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif |
| **SİSTEM TASARIMCISI** |  |  | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif | C:\Users\SAMED\Desktop\3e490deba927ac580110aa5b2fc2b4e7_400x400.gif |

Şekil 2.15 – Ekip Yapısı Zaman-İş Planı

**2.8 Kalite Sağlama Planı**

Şekil 2.16 – Kalite Sağlama Planı



**1. Belgeleme:** Sunduğumuz belgeden anlaşılacağı üzere sistem hakkında bilgi tam anlamıyla bu dokümanda oluşturulmuştur.

**2. Birlikte Çalışabilirlik:** Projede çalışma süresini en çok arttıracak olan şey veri karşılaştırılması… Sistemde tüm veriler eş zamanlı olarak saklanır.

**3. Bütünlük:** Her kuruluşta ayrıca görevlendirilecek olan yetkili personel sistemin tüm kısımlarına hâkim olacak ve program bir bütün halinde çalışacaktır.

**4. Değiştirilebilirlik:** Otomasyonun veritabanına hâkim olan, yetkilendirilmiş personel sistem üzerinde değişiklik yapabilecek.

**5. Ekonomi:** Yazılımın maliyeti pahalı olabilir ancak kullanıcı taraflı bakarsak zaman tasarrufundan dolayı gayet uydundur.

**6. Esneklik:** Proje internet üzerinden çalışacağı ve mobil uyumlu tasarıma sahip olduğu için yeterince esnektir.

**7. Etkinlik:** Kullanıcı, kendisine sunulan her alana hâkim olduğu için; sistemi etkin bir biçimde kullanabilecektir.

**8.Genellik:** Proje teşhis koyulabilecek her kuruluşta kullanılabileceği için geneldir. Sadece Türkiye sınırlı değil globalde kullanılabilir.

**9.Güvenilirlik:** Sistem için gerekli güvenlik önlemleri alınmıştır.

**10. Kullanılabilirlik:** Karmaşık olmayan arayüzü ile her bilgi düzeyindeki insana hitap eder.

**11. Modülerlik:** Yetkili personel ve hastaların kendine özel arayüzleri vardır.

**12. Sınanabilirlik:** Projedeki pilot bölge olan sağlık kuruluşu sınanabilirliğin bir göstergesidir.

**13. Tamlık:** Projede herhangi bir açık yoktur ve programda bulunan tüm butonlar, textler vs. çalışır ve tam durumdadır.

**14. Taşınabilirlik:** Kullanıcı taraflı olan arayüze internet bağlantısı olan her cihazdan erişilebilir.

**15.Temizlik:** Sade arayüzü ve birbirine yakın renkler kullanılarak oluşturulmuş renk paleti sayesinde oldukça temiz bir arayüze sahiptir.

**16.Yeniden Kullanılabilirlik:** Sistem her koşulda tekrardan düzenlenip kullanılabilir.

**2.9 Konfigürasyon Yönetim Planı**

Sistem ileriki zamanlarda yeni alerjilerin oluşması durumunu teşhis edemeyeceği için güncelliğini korumak adına basit bir yapay zekâ içermektedir. Bu sistemin içeriği;

* Yetkili sağlık personeli tarafından fark edilen hatalı teşhisler sisteme bildirilir.
* Sistem, bildirilen hatalı teşhisleri inceler.
* Yeterli sayıda ve yüksek oranda bildirilen aynı tipteki teşhisler tespit edilir.
* Tespit edilen teşhisler, sisteme eklenmesi adına ilgili uzmanlara inceletilir.

**2.10 Kaynak Yönetim Planı**

* Bu projede yeni bir süreç iyileştirilmesi kullanılıyor mu?
* Eğer yeterli sayıda hata bulunmuyorsa veya kalite beklenenden daha iyi görünüyorsa, kalite gerçekten daha iyi olabilir.
* İnceleme toplantısında çözüm bulmak için vakit kaybı oluyor mu?
* İnceleyecek kişi veya kişiler hazırlık yapıyor mu?
* Kalite beklendiği gibi mi?
* Proje elemanları yeni bir eğitim aldı mı?
* Yeni bir araç kullanılıyor mu?
* Yeterli hata bulunuyor mu?
* Yeterli vakit harcanıyor mu?

**2.11 Sistem Eğitimi**

Projede kazanılacak en önemli yetkinliklerden birisi de eğitimdir. Kullanılacak dillerin arayüz editör ve programların kullanımına hâkim olunamaması durumunda proje başarılı bir şekilde sonlandırılmaz. Bu yüzden proje için bazı eğitimlerin alınması gereklidir. Bu eğitimler;

* Adobe Photoshop 2020 kullanım eğitimi
* MySQL dil eğitimi
* C# dil eğitimi
* PHP dil eğitimi
* HTML dil eğitimi
* CSS dil eğitimi

| KONU | SÜREÇ |
| --- | --- |
| Adobe Photoshop 2020 Arayüzü | 1. Gün |
| C# & MySQL Bağlantıları | 2. Gün |
| PHP & MySQL Bağlantıları | 3. Gün |
| HTML & CSS Kullanımı | 4. Gün |

Şekil 2.17 – Eğitim Planı

Proje teslimi sonrası sistemi kullanacak olan sağlık personellerine ve yetkili personellere proje kullanımı için 3 saatlik bir seminer verilecektir. Bu seminerin içeriği şu şekilde olacaktır;

* Sisteme genel bakış
* Sistemde sağlık personellerinin yetkisi
* Sistemin kendi personellerinin yetkisi
* Sistemin genel işleyişi ve genel kısıtları açıklanarak nasıl kullanıldığına dair bilgiler vermektir.

**2.12 Test Planı**

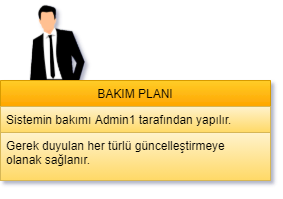
Proje test ekipleri ve görevleri şu şekildedir;

* Seçilen pilot hastanedeki bir kısım sağlık personeli ve sağlık personellerinden seçilecek olan sistem yöneticisi görev yapacaktır.
* Sistem yöneticisi süreci gözlemleyecek ve gereken kısımlarda yardımcı olacaktır.
* Bu ekipler test aşamasında hastadan sadece tıbbi kâğıt aracılığı ile tükürüğünü alıyor ve test yapma işlemini gerçekleştiriyor.

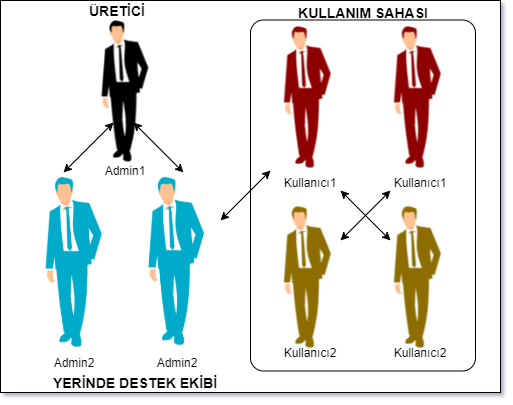
**2.13 Bakım Planı**

Yetkili bilgisayar ve yetkili kişilerin sisteme eklenmesi veya çıkartılması durumları; veritabanının kurulu olduğu sistemin güncellenmesi, bakımı ve yükseltilmesi durumları bakım planına dâhil olup bu adımda yapılacaktır.

Şekil 2.18 – Proje Bakım Planı



Şekil 2.19 – Kullanım & Üretim Sahası

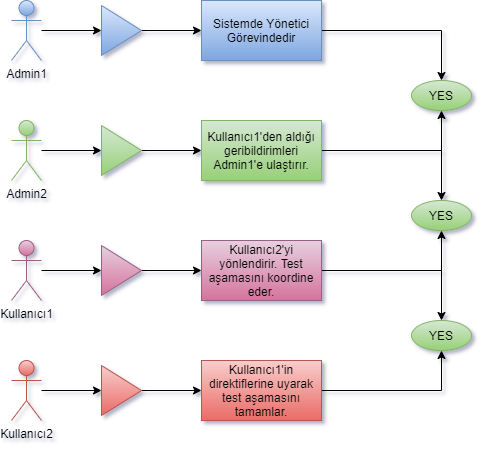


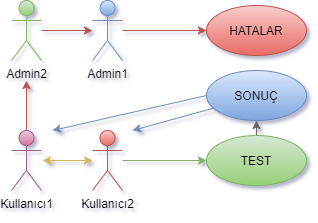
**3.SİSTEM ÇÖZÜMLEME**

**3.1 Mevcut Sistem İncelemesi**

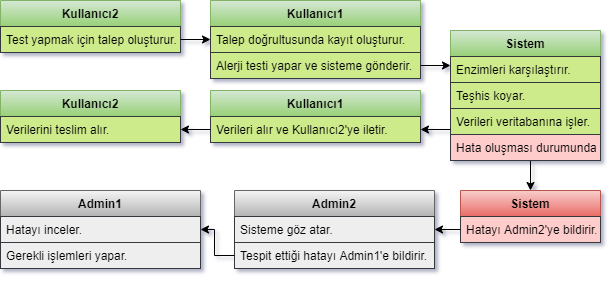
Elimizde bulunan sistem tüm hastanelerde mevcut olduğu için sistemin incelenmesi zor olmayacaktır. Ayrıca yeni sistemde oluşabilecek hataları gidermek adına mevcut sistemden yararlanılabilir.

**3.1.1 İşlevsel Model**

****Şekil 3.1 – Use-Case Diyagramı

Şekil 3.2 – Basit Bir Use-Case

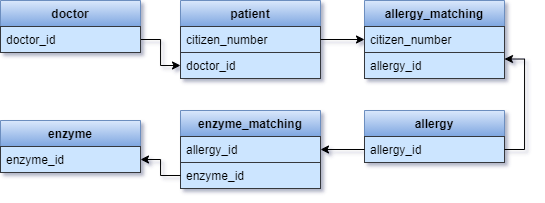
Şekil 3.3 – Temel Görevlendirme



**3.1.2 Veri Modeli**

Kullanacağımız sistemin veritabanı ilişkisel veri modelinde bulunan tablo ilişkileri aşağıdaki gibidir;

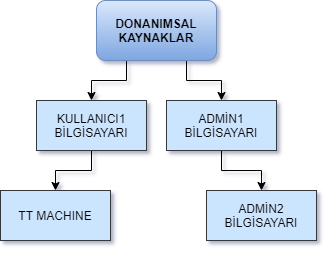
Şekil 3.4 – Sistem Veri Modeli



**3.1.3 Varolan Yazılım - Donanım Kaynakları**

* TT Machine

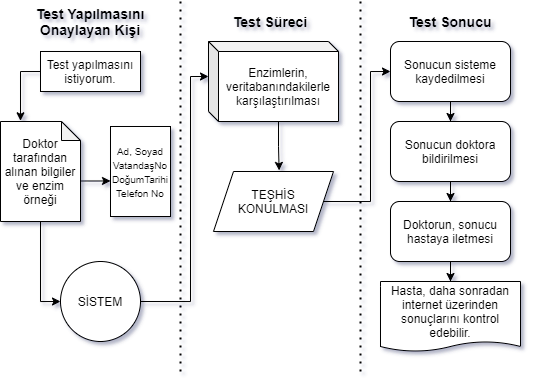
Şekil 3.5 – Donanımsal Kaynaklar



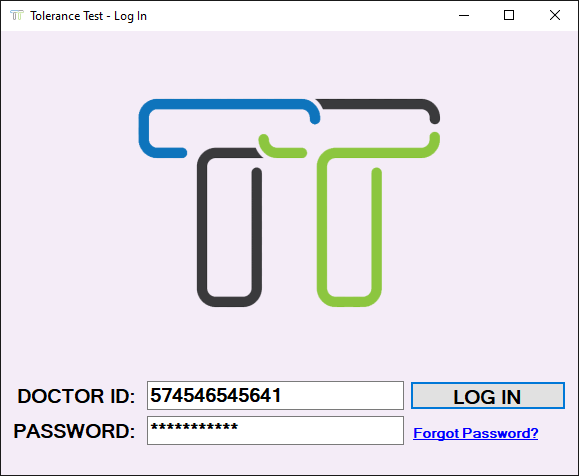
Şekil 3.6 – Test ve Sonuç Aşaması



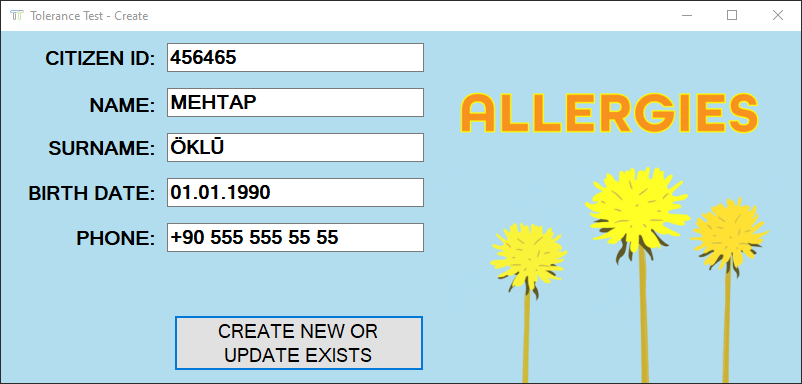
Şekil 3.7 – Test ve Sonuç Aşaması



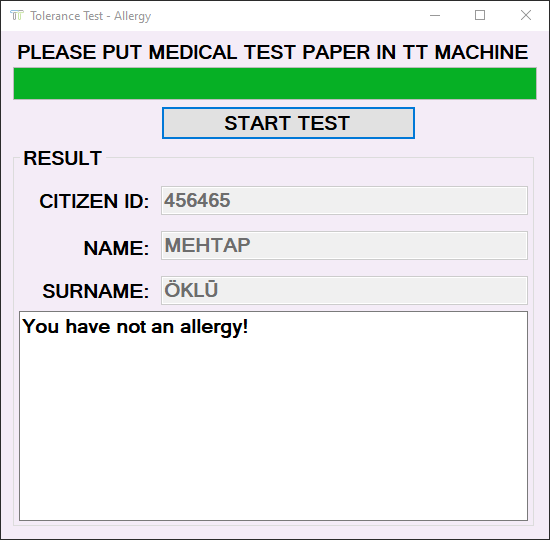
Şekil 3.8 – Tolerance Test Doktor Giriş Ekranı

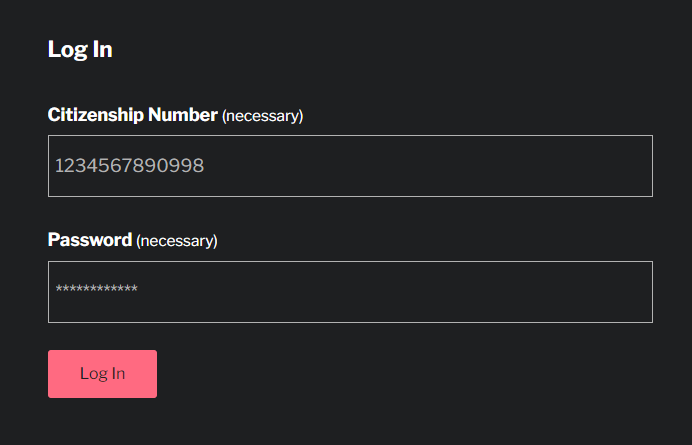


Şekil 3.9 – Tolerance Test Hasta Kayıt Ekranı



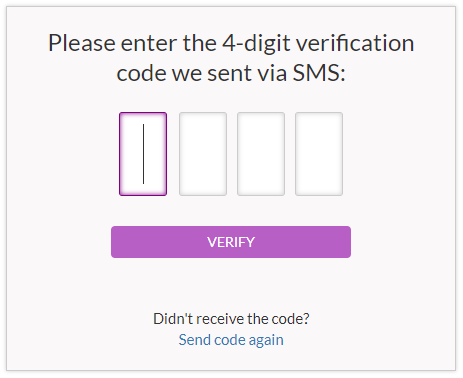
Şekil 3.10 – Tolerance Test Sonuç Ekranı



****Şekil 3.11 – Tolerance Test Hasta Sonuç Ekranı

**3.1.4 Varolan Sistemin Değerlendirilmesi**

Daha önce belirttiğimiz gibi hastalar kendi sonuçlarını uygulama üzerinden tekrar görüntüleyebilecektir. Hasta daha sonra kendi sonucunu görmek istediğinde cep telefonuna gelen SMS ile sistemde güvenlik doğrulaması yapabilecektir.

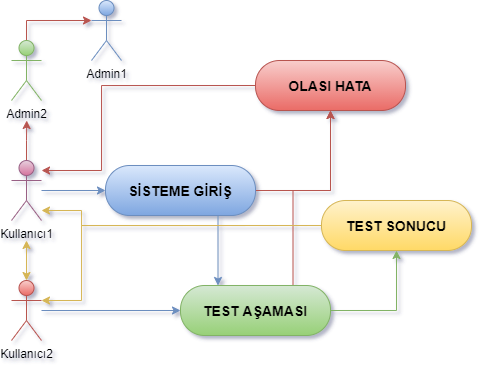
****Şekil 3.12 – Tolerance Test Hasta Sonuç Ekranı (Doğrulama)

**3.2 Gereksenen Sistemin Mantıksal Modeli**

**3.2.1 Giriş**

Mevcut sistem incelendiğinde verilerin kaybolması daha sonra tekrar erişilememesi gibi eksikler söz konusu. Yeni sistemde bu eksikleri gidermek bize düşüyor.

**3.2.2 İşlevsel Model**

****Şekil 3.13 – Use-Case Diyagramı

Durum İsmi Test İçin Başvuru  
 Aktör Başlatılıyor Kullanıcı2  
Aktör Katılımcı Kullanıcı1  
Giriş Durumu Kullanıcı2, Kullanıcı1’e Başvuru Yapmış  
Olayların Akışı **1.** Kullanıcı1, Kullanıcı2’den kişisel bilgilerini alır.  
 **2.**Kullanıcı1, bilgileri kullanarak kayıt oluşturur.  
Durum Çıkışı Test için başvuru yapılıp, başarılı sonuçlanmıştır.

Durum İsmi Test Aşaması  
 Aktör Başlatılıyor Kullanıcı1  
Aktör Katılımcı Kullanıcı2  
Giriş Durumu Kullanıcı1, Kullanıcı2’yiYönlendirir  
Olayların Akışı **1.** Kullanıcı1, Kullanıcı2’den enzim örneği alır.  
 **2.** Enzim örneği alınan kâğıt, TT Machine’e yerleştirilir.  
Durum Çıkışı Test, yapılmıştır.

Durum İsmi Sonuç Aşaması  
 Aktör Başlatılıyor Kullanıcı1  
Aktör Katılımcı Kullanıcı2  
Giriş Durumu Kullanıcı1, Sistemi Kontrol Eder  
Olayların Akışı **1.** Kullanıcı1, sistemi kontrol eder.  
 **2.** Sistemden çıkan sonuç, Kullanıcı2’ye bildirilir.  
Durum Çıkışı Test sonucu öğrenilmiştir.

Durum İsmi Test Sonrası Tekrar Sonuç Kontrolü  
 Aktör Başlatılıyor Kullanıcı2  
Aktör Katılımcı -  
Giriş Durumu Kullanıcı2, Sonuçlarını Kontrol Eder  
Olayların Akışı **1.** Kullanıcı2, daha sonradan sonuçlarını görmek ister.  
 **2.** Sistemin web arayüzünü kullanarak, sonuçlarına bakar.  
Durum Çıkışı Test sonuçları tekrar görüntülenmiştir.

Durum İsmi Olası Hata Durumu  
 Aktör Başlatılıyor Kullanıcı1  
Aktör Katılımcı Admin2 – Admin1  
Giriş Durumu Kullanıcı1, Sistemde Çıkan Hataları Adminlere Bildirir  
Olayların Akışı **1.** Kullanıcı1, sistemde çıkan hatayı Admin2’ye bildirir.  
 **2.** Admin2, hatayı Admin1’e bildirir.  
Durum Çıkışı Olası hata durumu tespit edildi.

Durum İsmi Hata Gidermesi  
 Aktör Başlatılıyor Admin1  
Aktör Katılımcı -  
Giriş Durumu Admin1, Hataları Giderir  
Olayların Akışı **1.** Admin1, hatayı analiz eder.  
 **2.** Gerekli işlemler dizisini gerçekleştirir.  
Durum Çıkışı Hata giderildi.

**3.2.3 Genel Bakış**

Sistemimize genel açıdan göz atarsak Admin1 ve Admin2, hatalara müdahale edemiyor. Hatalar Kullanıcı1 tarafından saptanır ve Admin2’ye bildirilir. Admin2 ise kendisine bildirilen hataları Admin1’e bildirir. Tüm bu hiyerarşik rapor sisteminin sonrasında Admin1, ilgili bilirkişilere danışır ve sistemdeki hatanın kaynağını düzeltir. Tüm bu olaylar akışının özeti, yukarıdaki Use-Case diyagramında yer almaktadır.

**3.2.4 Bilgi Sistemleri – Nesneler**

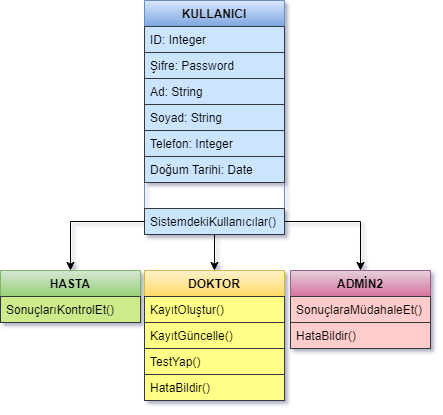
**Admin1:** Sistemde bulunan en yetkili kişidir. Admin2 ve Kullanıcı1’e yetki verir

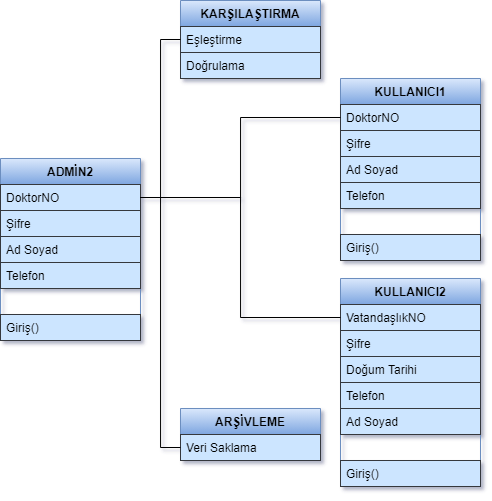
**Admin2:** Sistemde bulunan en yetkili ikinci kişidir. İlgili birimdeki hataları Admin1’e bildirir.

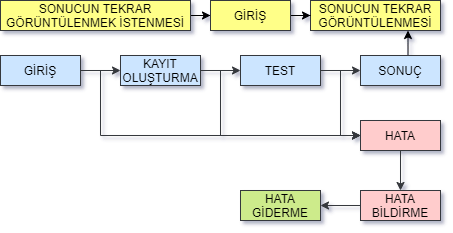
**Kullanıcı1:** Sistemi kullanacak olan doktordur. Sistemi kullanır aynı zamanda sistemde oluşan hataları gözlemler ve Admin2’ye bildirir.

**Kullanıcı2:** Sistem asıl olarak Kullanıcı2 için tasarlanmıştır. Kullanıcı2’ye özel olarak tasarlanmış olan arayüzü kullanır.

**Arşiv:** Verilerin elektronik ortamda saklanarak kaybolmasını önleyen stok nesnedir. Arka planda olduğu için Use-Case diyagramında ele almadık.

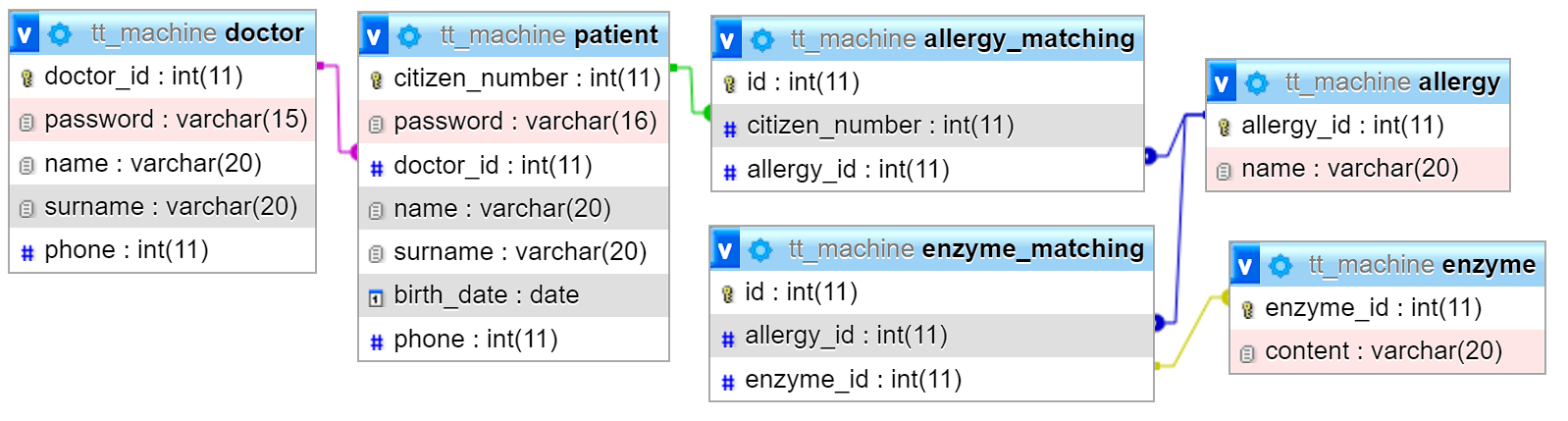
****Şekil 3.14 – Kullanıcılara Verilmiş Fonksiyon Tablosu

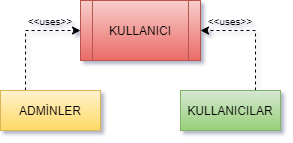
  
Şekil 3.15 – Veri Tabanı İlişkileri

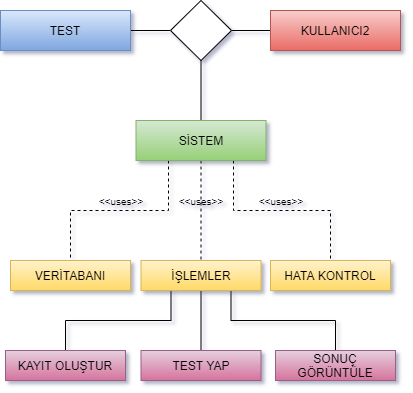
  
Şekil 3.16 – İşlem Aşamaları

**3.2.5 Veri Modeli**

Şekil 3.17 – Veri Tabanı Modeli



Şekil 3.18 – İşleyiş Diyagramı

  
Şekil 3.19 – Test & Kullanıcı

**3.2.6 Veri Sözlüğü**

Ana bilgilerin tanımlanması:

* **ID:** Integer
* **Ad Soyad:** String
* **Doğum Tarihi:** Date
* **Phone:** Integer

Alerji bilgilerinin tanımlanması:

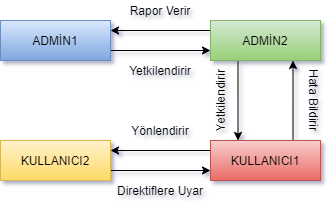
* **ID:** Integer
* **Alerji Adı:** String

Enzim bilgilerinin tanımlanması:

* **ID:** Integer
* **Enzim İçeriği:** String

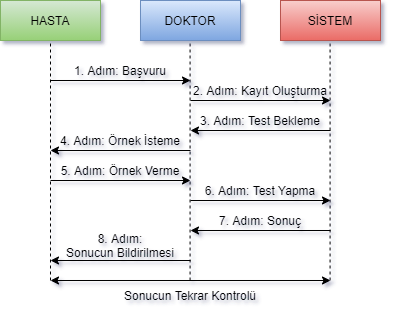
Alerji - Enzim ilişkilerinin tanımlanması:

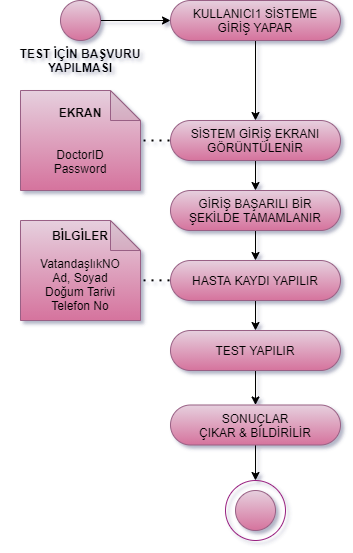
* **ID:** Integer
* **Alerji ID:** Integer (Alerji tablosundaki **ID** sütununa bağlıdır.)
* **Enzim ID:** Integer (Enzim tablosundaki **ID** sütununa bağlıdır.)

  
Şekil 3.20 – Hiyerarşik Düzen

**3.2.7 İşlevlerin Sıradüzeni**

Veriler Kullanıcı1 tarafından işlenir. Olası bir hata durumunda Kullanıcı1, Admin2’ye bilgi verir. Admin2 de Admin1’e gerekli bilgileri iletir. Admin1 ise yapılması gereken işlemleri yapar. Tüm bunların yanında Kullanıcı2, kendine özel olarak tasarlanmış arayüzü kullanır.

  
Şekil 3.21 –İşleyiş Diyagramı

****Şekil 3.22 –Aktivite Diyagramı

**3.2.8 Başarım Gerekleri**

Elimizde bulunan sistemler incelenmiş olup mevcut sistemin eksiklerini gözlemleyerek yeni sistemin başarısı için yapılması gerekenler:

* Teşhis koyma doğrulukları,
* Teşhis koyma sürecinde zaman tasarrufu,
* Hastanın konforu,
* Kullanım kolaylığı,
* Anlaşılabilirlik,
* Verilerin saklanması durumu…

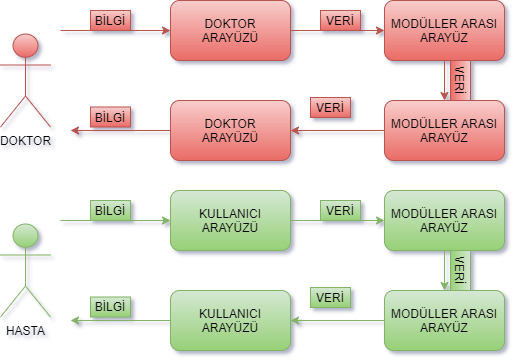
**3.3 Arayüz (Modül) Gerekleri**

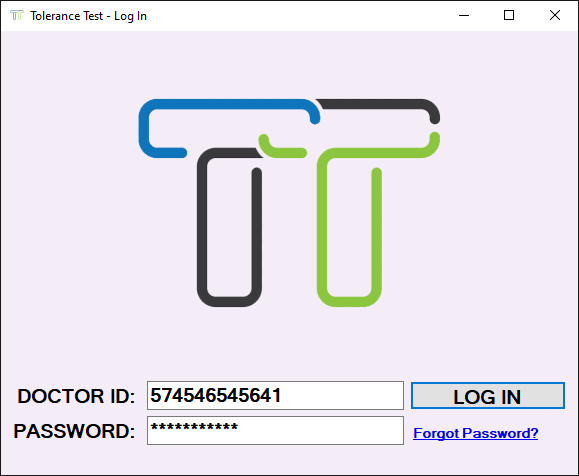
**3.3.1 Yazılım Arayüzü**

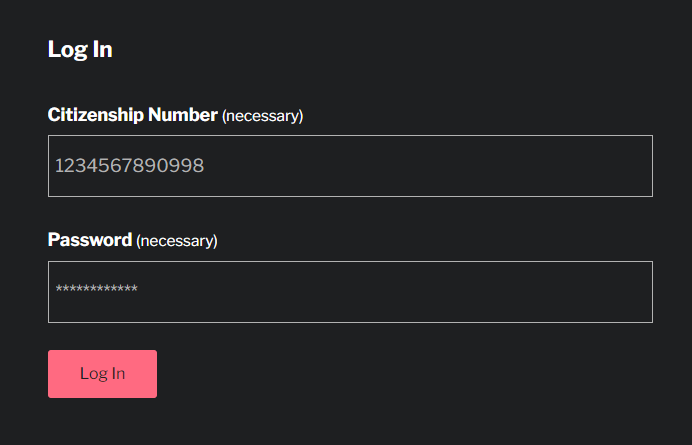
Projenin çalışma sırasında herhangi bir açık oluşmamasına dikkat edildi. Bunun için gerekebilecek her türlü değişiklik kaynak kodları üzerinden yapılıp tekrar derlenecek.

**3.3.2 Kullanıcı Arayüzü**

Karmaşık olmayan arayüzü ile her bilgi düzeyindeki insana hitap eder. Sade arayüzü ve birbirine yakın renkler kullanılarak oluşturulmuş renk paleti sayesinde oldukça temiz bir arayüze sahiptir.Kullanıcı1 ve Kullanıcı2’ye özel iki farklı arayüz mevcuttur.

  
Şekil 3.23 – İş Akışları

****  
Şekil 3.24 – Kullanıcı Arayüzü 1

  
Şekil 3.25 – Kullanıcı Arayüzü 2

**3.3.3 İletişim Arayüzü** Gerekli iletişim modülleri;

* TT Machine – Arayüz (Kullanıcı1)
* Tıbbi kâğıt TT Machine’e yerleştirildikten sonra analiz edilen enzimlerin arayüze gönderilmesi
* Arayüz (Kullanıcı1)– Database
* TT Machine’den gelen analiz sonuçlarının veritabanında işlenmesi
* Database – Arayüz (Kullanıcı2)
* Kullanıcı2’nin kendi verilerini görebileceği kısım

**3.4 Belgeleme Gerekleri**

**3.4.1 Geliştirme Sürecinin Belgelenmesi**

Geliştirme sürecinin dokümantasyonunun yapılması hem şimdiki süreç için hem de ileriye dönük geliştirme sürecinin ilerleme durumunu ve eksikliklerini genel anlamda görebilmek amacıyla yapılır. Bunun yanında projeye yeni Admin1 dâhil olması durumunda, kendisine sunulabilecek bir belge olması için dokümantasyon yöntemine başvuruldu.

**3.4.2 Sıkça Sorulan Sorular (SSS)**

Kullanıcı2 arayüzünde ve Kullanıcı1 panelinde yer alacak olan bilgilendirme metinleri, kullanıcıları yönlendirecektir.

**4. SİSTEM TASARIMI**

**4.1 Genel Tasarım Bilgileri**

**4.1.1 Genel Sistem Tanımı**

Şekil 4.1 – Genel Sistem Tanımı



* **Gereksinimler**

Sistemin kullanıldığı hastanelerde bir anket düzenlenecek olup çıkan sonuçlar doğrultusunda bir tasarım oluşturulacaktır.

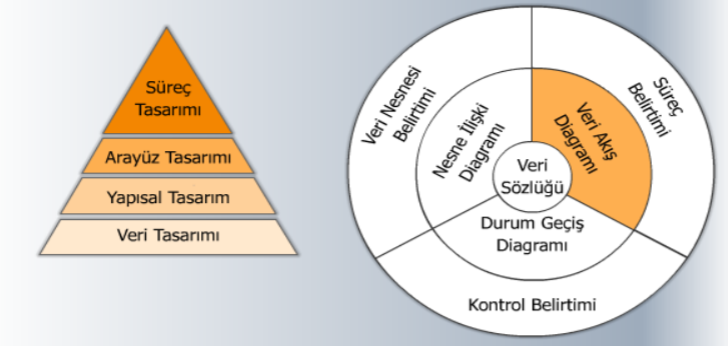
* **İşlevsel Belirtimler**

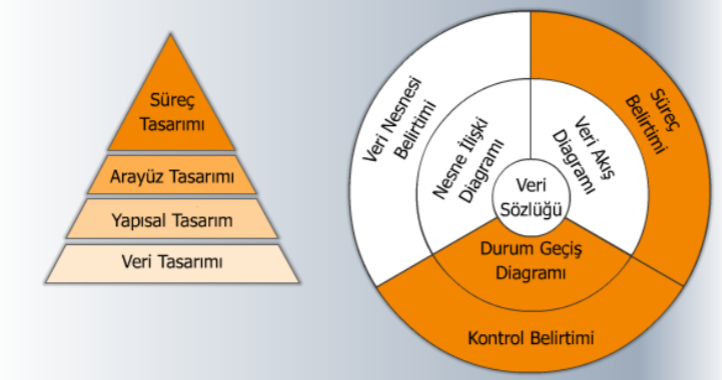
Sistemin ne olduğuna kısaca değinelim. Sistem Türkiye ve Dünya’da alerji testi yapılabilen hastane ve tıp merkezlerinde kullanılacaktır. Kişilerin konforları ve sonuçların çabucak çıkması açısından zaman tasarrufu sağlanmıştır. Kişiler eski sistemde canları acıyarak bir teste tabii tutulurken yeni olan sistemde kişinin ağzına tıbbi kâğıt yerleştirilir ve örnek alınır. Örneği analiz etmek için TT Machine’e koyarız. TT Machine enzimleri veritabanında karşılaştırarak sonucu ekrana yazdırır. Doktor sonuçlar doğrultusunda gerekli tedavilere başlar. Bunlara ek olarak test yaptıran kişiler daha sonra test sonuçlarını tekrar görüntülemek istediğinde kendileri için geliştirilen arayüzde sonuçlarına tekrar ulaşabilecekler.

* **Tasarım**

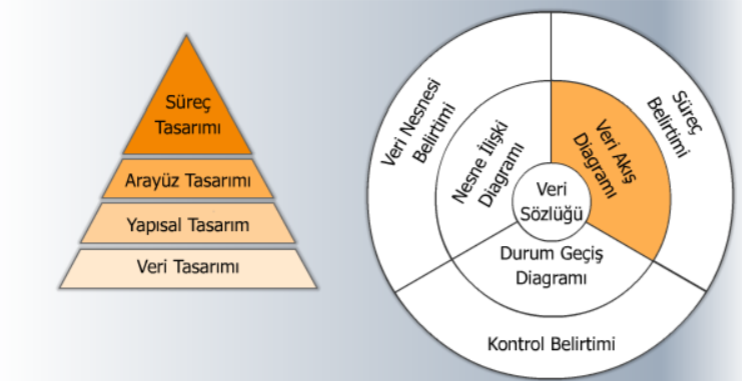
Aşamalı olarak tasarımda neler olacağını grafiksel olarak aşağıda aktardım.

1. Süreç tasarımı olacak olup adımları aşağıdaki gibidir.

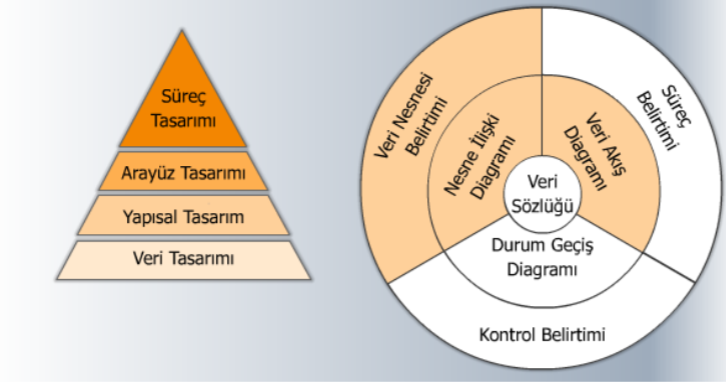




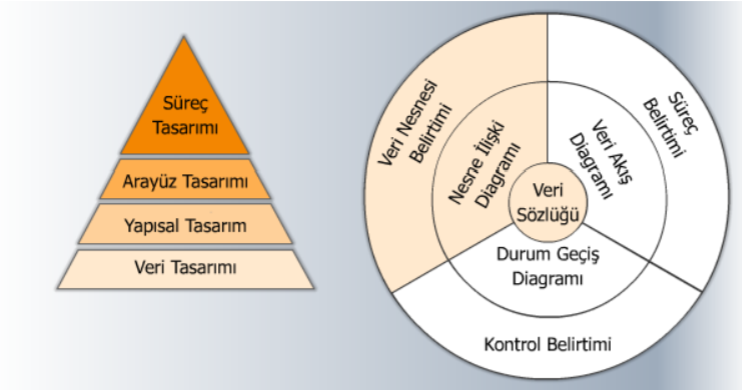
1. Süreç tasarımı bittikten sonra önceliğimiz arayüz tasarımı olacak ve adımları aşağıdaki gibi gerçekleştirilecektir.



1. Arayüz tasarımının da ardından sıra yapısal tasarıma geldi. Yapısal tasarımda izlenen yollar aşağıdaki gibi olmuştur.



1. Yapısal tasarımı da sonlandırdıktan sonra sıra veri tasarımına geldi ve izlediğimiz adımlar aşağıdaki gibidir.



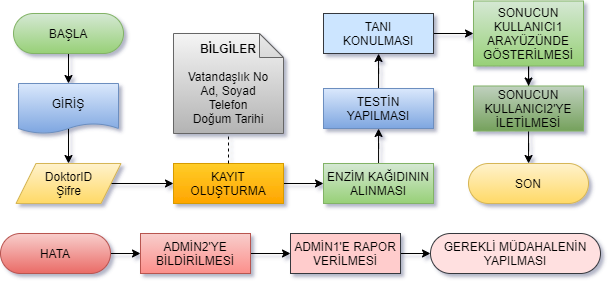
**4.1.2 Kısıtlamalar**

Sistemimizde bulunan kısıtlamalar aşağıdaki gibidir

* Hastaneye ait sistemin olması
* Daha önce bu konuda eğitilmiş olması
* Dosya içeriğinin yalnızca hastanede bulunan doktorlar ve hasta tarafından görüntülenebiliyor olması

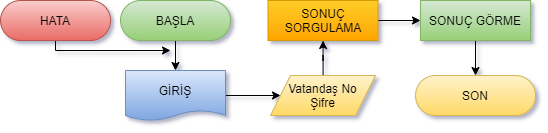
**4.1.3 Sistem Mimarisi**

Sistemin işleyiş mantığının anlatılabilmesi ve nasıl bir yol izlenebileceğinin aktarılabilmesi için sistem mimarisi akış diyagramı şeklinde verilmiştir.

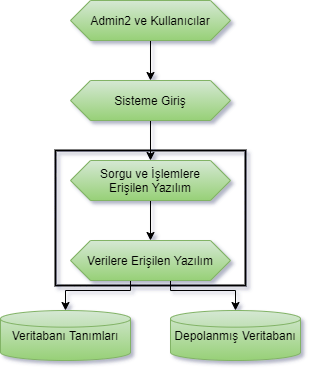


Şekil 4.2 – Genel Sistem Mimarisi

Şekil 4.3 – Kullanıcı2 Sistem Arayüz Mimarisi



Şekil 4.4 – Veritabanı Sistemi



**4.1.4 Dış Arabirimler**

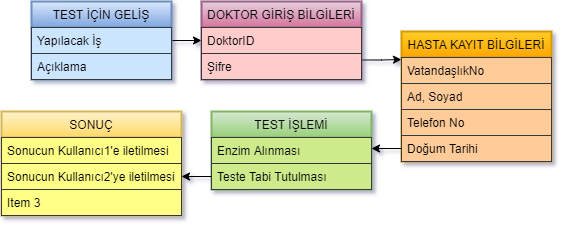
**4.1.4.1 Kullanıcı Arabirimleri**

Kullanıcı arabirimlerinin ilk başında sistem giriş ekranı bulunacak. Admin2 ve Kullanıcı1’in ortak kullandığı kendilerine ait giriş ekranı olacak. Aynı zamanda Kullanıcı2’nin sonuçlarını tekrar kontrol edebilmesi için kendine ait arayüz mevcuttur. Kullanıcılar bilgilerini girerek bu sisteme giriş yapabilecektir.

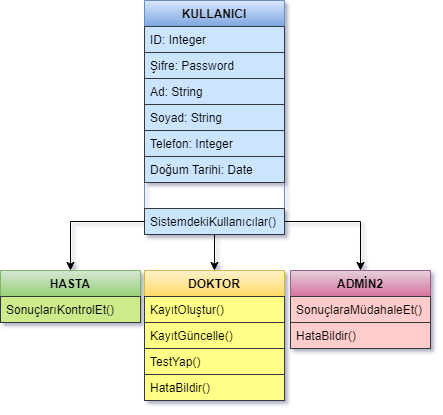
**4.1.4.2 Veri Arabirimleri**

Veri arabirimlerinde veritabanı ile bağlantı kurulurken PHP modülleri kullanılacaktır. Arayüz ve veritabanı eş zamanlı olarak iletişim halinde olacaktır. Bu sistemin kullanılmasından önce elde edilen veriler sisteme dâhil edilmeyecektir.

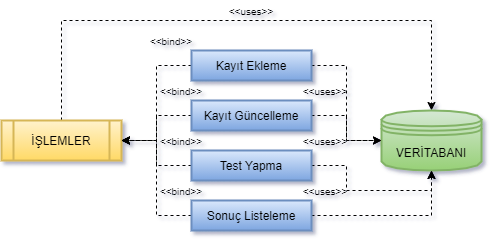
Şekil 4.5 – Genel Sistem Mimarisi



**4.1.5 Veri Modeli**

****Şekil 4.6 – Veritabanı Diyagramları

Şekil 4.7 – Veritabanı Diyagramları



**4.1.6 Testler**

Sistemimizin testini iki aşamada gerçekleştireceğiz. Daha önce belirttiğimiz gibi pilot bölge uygulaması yapılacaktır.

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

**Beta Aşaması:** Kullanıcı1’ler geliştirilen sistemi bulunduğu hastanede Admin2 gözetmeni eşliğinde test edecek.

**4.1.7 Performans**

Sistemin performansını etkileyebilecek her türlü etken değerlendirilecek.

* Veri yapısının değerlendirilmesi;
  + Verilerin arayüz üzerinden veritabanına veya veritabanından arayüze stabil bir şekilde iletebiliyor olması.
* Veritabanı yapısının değerlendirilmesi;
  + Tabloların lojik bir şekilde oluşturulması ve bağlantılarının kurulması
* Arayüz yapısının değerlendirilmesi;
  + Doktor arayüzünün sade ve kolay anlaşılabilir olması durumunda test işleminin daha kısa sürmesi
  + Hasta arayüzünün kalabalık olmaması ve arayüzdeki nesnelerin, işlevlerine belli ediyor olması, herhangi bir eğitim veya kılavuz gereksinimi duymadan kullanmayı sağlaması.

**4.2 Veri Tasarımı**

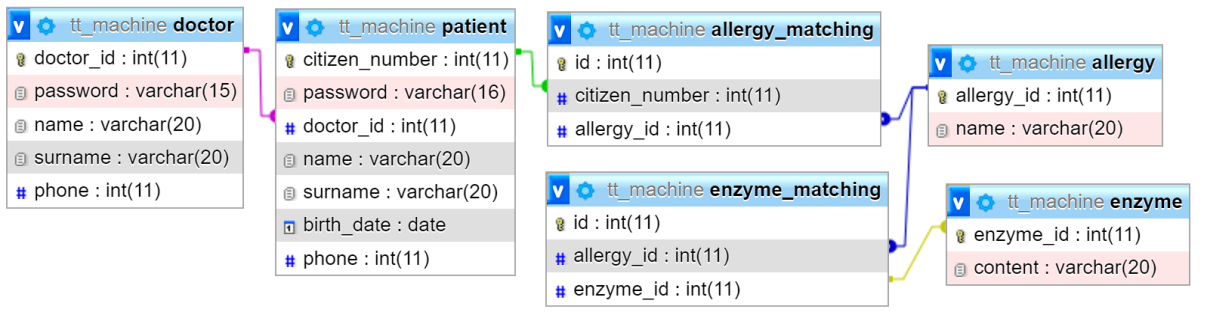
**4.2.1 Tablo Tanımları**

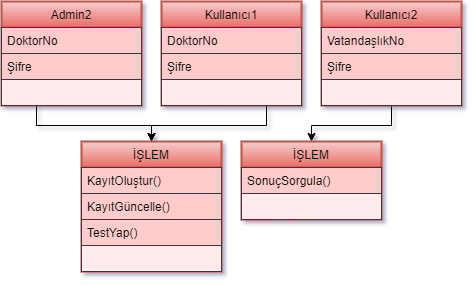
Sistem 6 tablodan oluşmaktadır. Veritabanımız daha sonradan değiştirilecek şekilde tasarlanmamıştır. Sistemi kullanacak olan Kullanıcı2 ve Kullanıcı1 farklı fonksiyonlara sahip olduğundan verileri farklı tablolarda tutulmuştur. Bunun yanı sıra Kullanıcı1 ve Admin2 neredeyse aynı fonksiyonlara sahip olduğundan, verileri aynı tabloda tutulmuştur.

| **KULLANICI TABLOSU** | |
| --- | --- |
| **Veri Adı** | **Veri Tipi** |
| citizen\_number | Integer |
| password | Integer |
| name | Date |
| surname | Time |
| birth\_date | Integer |
| phone | Boolean |

Şekil 4.8 – Kullanıcı Tablosu Veri Çeşitleri

**4.2.2 Tablo İlişki Şemaları**

****Şekil 4.9 – İlişki Şemaları

  
Şekil 4.10

**4.2.3 Veri Tanımları**

Veri tipi olarak Integer kullanılmasının amacı sayısal değerleri almak olurken string kullanılmasının amacı değerleri kelime olarak tutmaktır. Bunların dışında bir de date isimli bir veri tipimiz vardır. Bu veri tipi içerisinde tarih verisi barındırır. Bu veri tipini Kullanıcı2’nin doğum tarihini kaydetmek için kullanırız.

**4.2.4 Değer Kümesi Tanımları**

**Karşılaştırma:**Hastadan alınan enzimlerin veritabanındakilerle karşılaştırılıp eşleşen veri arama aşamasıdır.

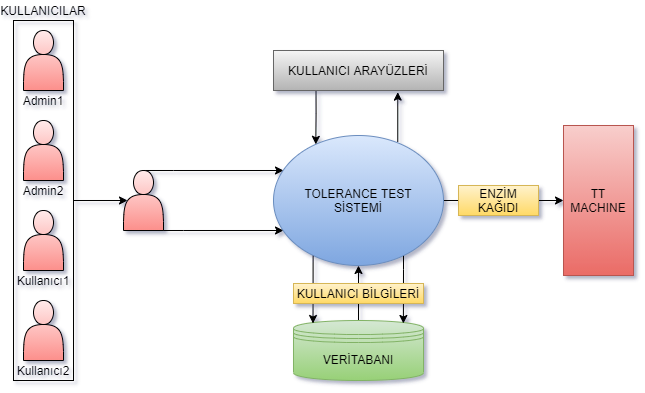
**Arşivleme:** Kendinden önce bitmiş olan tüm aşamaların veritabanı üzerine kaydedilmesi işlemidir.

**4.3 Süreç Tasarımı**

**4.3.1 Genel Tasarım**

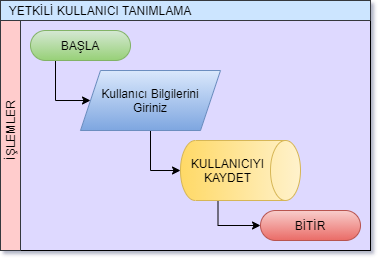
Tasarımda öncelik veritabanını oluşturmaya verildi. Daha sonra giriş modülü ardından yönetici modülleri ve en sonda kullanıcı arayüzü oluşturuldu.

Şekil 4.11



Şekil 4.12 – Giriş Sayfası Modülleri



  
Şekil 4.13 – Yetkili Kullanıcı Tanımlama

**4.3.2 Kullanıcı Arayüz Girişi**

**4.3.2.1 Kullanıcı1 Girişi ve Admin2 Girişi**

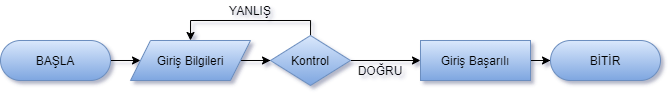
Kullanıcı1 ve Admin2’nin sisteme erişebilmesi için giriş yapması gerekmektedir. Giriş işlemini bu modül üzerinden yapar.

* **SqlConnection connect:** Veritabanı için bağlantı nesnesi
* **string connectionString:** Veritabanının bulunduğu yol
* **string userName:** Veritabanına bağlantı için kullanıcı adı
* **string password:** Veritabanına bağlantı için şifre

**4.3.2.2 Kullanıcı2 Girişi**

Kullanıcı2’nin sisteme erişebilmesi için giriş yapması gerekmektedir. Giriş işlemini bu modül üzerinden yapar.

* **mysql\_connect():** Veritabanı için bağlantı nesnesi
* **string $connectionString:** Veritabanının bulunduğu yol
* **string $userName:** Veritabanına bağlantı için kullanıcı adı
* **string $password:** Veritabanına bağlantı için şifre

  
Şekil 4.14

| **METOD ADI** | **TANIM** |
| --- | --- |
| Giriş() | Sisteme girişi sağlar |
| Çıkış() | Sistemden çıkışı sağlar |
| KayıtEkle() | Hasta kaydı oluşturmayı sağlar |
| KayıtGüncelle() | Mevcut hastanın kaydını günceller |
| TestYap() | Hastaya test yapmayı sağlar |

Şekil 4.15 – Metotlar

**4.3.3 Kullanıcı Profilleri**

**Admin1:** Sistemde bulunan en yetkili kişidir. Admin2 ve Kullanıcı1’e yetki verir

**Admin2:** Sistemde bulunan en yetkili ikinci kişidir. İlgili birimdeki hataları Admin1’e bildirir.

**Kullanıcı1:** Sistemi kullanacak olan doktordur. Sistemi kullanır aynı zamanda sistemde oluşan hataları gözlemler ve Admin2’ye bildirir.

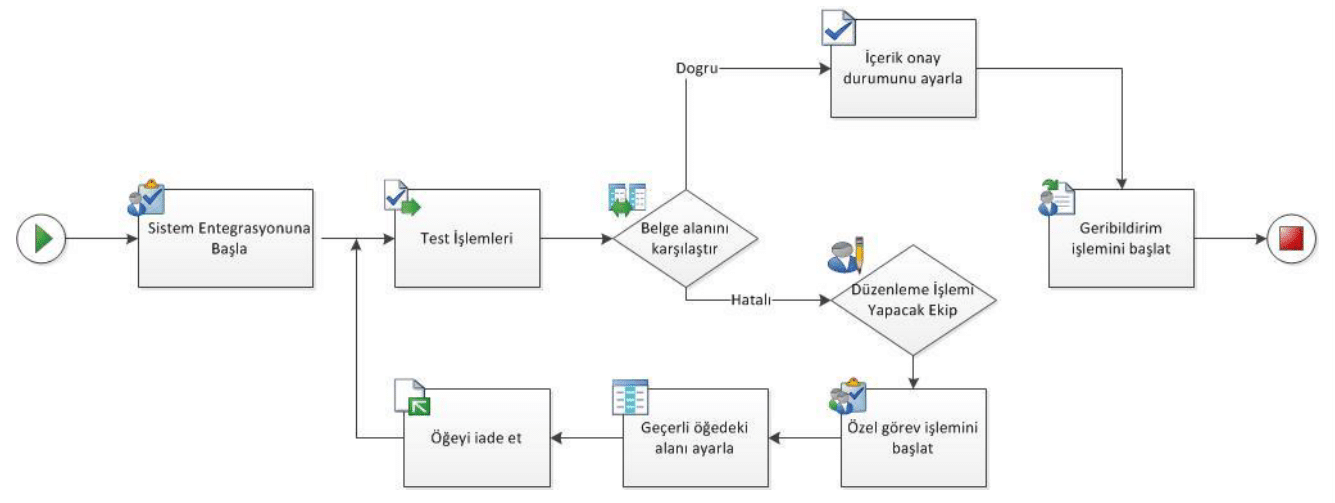
**Kullanıcı2:** Sistem asıl olarak Kullanıcı2 için tasarlanmıştır. Kullanıcı2’ye özel olarak tasarlanmış olan arayüzü kullanır.

**Arşiv:** Verilerin elektronik ortamda saklanarak kaybolmasını önleyen stok nesnedir. Arka planda olduğu için Use-Case diyagramında ele almadık.

**4.3.4 Entegrasyon ve Test Gereksinimleri**

Daha önce de belirttiğimiz gibi mevcut herhangi bir veritabanıyla bağlantımız olmayacaktır. Pilot bölgede yapılacak olan test aşamasından Admin2 ve Kullanıcı1’in yanı sıra Admin1 de yer alacaktır. Bu sayede Admin1 kâğıt üzerindeki verilerdense sistemi sahada gözlemlemiş olacaktır. Böylece sistem üzerinde gerçekleştirilecek olan bakım ve güncelleme çalışmaları daha verimli hale gelecektir.

Şekil 4.16



**4.4 Ortak Alt Sistemlerin Tasarımı**

**4.4.1 Ortak Alt Sistemler**

Kullanılacak tek ortak sistem Tolerance Test’in ana veritabanıdır. Her hasta ve doktor için uniqe bir ID kullanılacaktır.

**4.4.2 Modüller Arası Ortak Veriler**

Modüller arasında ilişki mevcuttur. Bunlar: doctor\_id, citizen\_number, allergy\_id, enzyme\_id.

**4.4.3 Güvenlik Alt Sistemi**

Sistemimizin güvenlikle alakalı gereksinimleri karşılayabilmesi adına MySQL veritabanı ve SHA-256 şifreleme tipi kullanılmıştır. Veri aktarımı sırasında, verinin aktarılacağı her cihazın yetkili olup olmadığı tek tek kontrol edilecektir.

* Olası hata ve arızalarla ilgili “Acil Durum Planı(ADP)” oluşturulması.
* Herhangi bir hata ve arıza olmaması durumlarında da kontrollere devam edilmesi…
* Proje geliştirmesinin her aşamasının ayrı ayrı test sürecine tabii tutulması…
* Yalnızca güncel değil geçmişe yönelik verilerin de incelenerek olası hataların önüne geçilmesi.

**4.4.4 Veri Dağıtım Alt Sistemi**

Sistemimiz veri çıkışından çok veri girişi almaktadır. Sisteme alınan veriler kullanıcıların hassas kişisel bilgilerini içermektedir. Bu yüzden hem Kullanıcı1 hem de Kullanıcı2 tarafında kullanılan arayüzler yüksek doğruluk oranlarına sahip güvenlik alt yapısına sahiptir.

**4.4.5 Yedekleme ve Arşivleme İşlemleri**

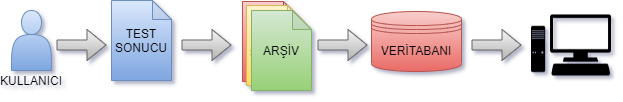
Elde ettiğimiz verilerin herhangi bir nedenden dolayı zarar görmesi, sistemin çalışma süreçlerinde zararlar oluşturabilir. Veriler geri yüklenemeyebilir.

Bu nedenle sistemin çalışma saatleri incelenir ve yoğunluğun en az olduğu saat bakım ve yedekleme saati olarak seçilir. Sistemin çalışma yoğunluğuna göre; günlük, haftalık veya aylık olarak bakım periyodu seçilir.

Sistem birden fazla sunucuya sahipse eş zamanlı yedekleme ve bakım yapılabilmesi için sunucuların birbirleriyle eşleştirilmesi ve sistem kapatılmadan yedekleme yapılması gerekmektedir.

Yoğunluğun az olduğu saatin seçilmesinin nedeni sistemin kapatılacak olması değildir. Bakım ve yedekleme esnasında ortaya çıkacak olan yavaşlama sonucunda hem yapılacak işlemlerin hem de bakım ve yedeklemenin aksamaması için bakım saatleri yoğunluğun en az olduğu saatler olarak seçildi.

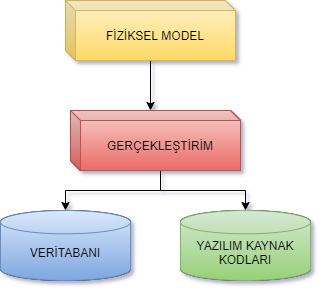
Şekil 4.17



**5. SİSTEM GERÇEKLEŞTİRİMİ**

**5.1 Giriş**

Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir. Yazılım geliştirmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.



Şekil 5.1 Sistem Gerçekleştirme Modeli

**5.2 Yazılım Geliştirme Ortamları**

Tasarım sonunda üretilmiş olan fiziksel modelin, ilgili ortamda çalışabilmesi için yazılım geliştirme ortamı gereklidir. Bunlar;

* Programlama dili,
* Veritabanı yönetim sistemi,
* Hazır program çıktıları

Gerekli olan CASE araçları belirlenmiş olup yazılım geliştirme ortamı hazırlanmıştır.

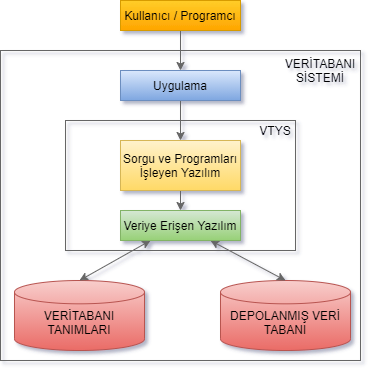
**5.2.1 Programlama Dilleri**

Sistemin geliştirme aşamasında kullanılan programlama dillerini bu dokümanın önceki kısımlarında belirtmiştik. Ancak bu dilleri neden seçtiğimizi söylememiştik. Öncelikli olarak Flash kullanmak istedik ancak geçtiğimiz zamanlarda Flash Player desteği tüm tarayıcılardan çekildi. Bu yüzden Flash yerine JavaScript eklentileri kullanabileceğimiz ve görselleştirme adına HTML – CSS kullanabileceğimiz PHP dilini seçtik. Ayrıca PHP dili rahat dokümantasyon bulunabilecek bir dildir ve veritabanı bağlantıları kolaylıkla yapılabilmektedir.

**5.2.2 Veritabanı Yönetim Sistemleri**

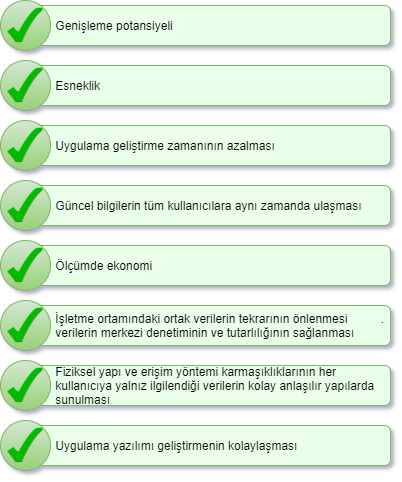
PHP veritabanı ile bağlantı kurup veriye erişen kodlama dilidir.

Veritabanı yönetiminde kullandığımız hiyerarşiyi gösterecek olursak;

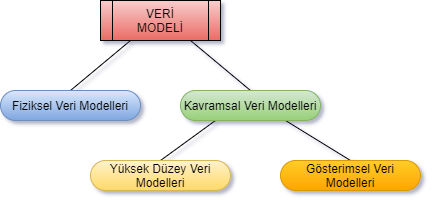


Şekil 5.2 - Veritabanı Sistemi

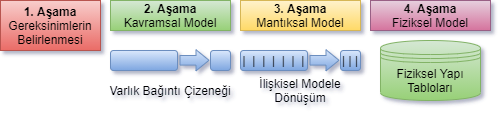
**5.2.2.1 VTYS Kullanımının Ek Yararları**

****  
Şekil 5.3 – VTYS’nin Ek yararları

**5.2.2.2 Veri Modelleri**

****  
Şekil 5.4 - Veri Modeli

MySql içerisindeki tablo alanlarında tutulacak olan verilerimizin birbiriyle ilişki içerisinde olduğunu fiziksel veri modeli sayesinde söyleyebiliriz.

  
Şekil 5.5 - Şema Mimarisi

1. **Gereksinimlerin Belirlenmesi**

* Veri tipleri
* Veri grupları
* Veriler ile ilgili kurallar
* Veriler üzerinde yapılması gereken işlemler

1. **Kavramsal Model**

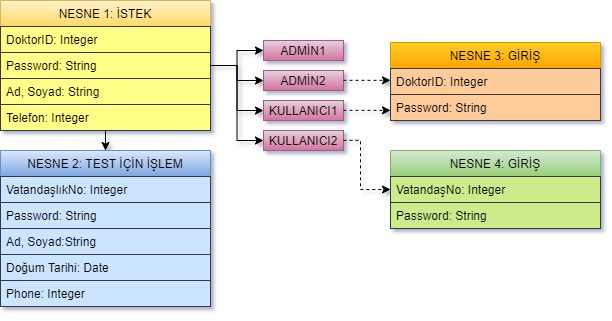
Alfa ve Beta aşamasında elde edilecek bilgiler ile gereksinimler analiz edilecek. Veriler ve grupların düzenlenmeleri yapılacak. Sonuç olarak elde edeceğimiz model grafiksel olarak gösterilecektir.

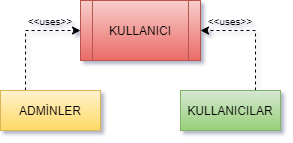
1. **Mantıksal Model**

Oluşturduğumuz veritabanını ilişkisel veritabanı modelinde tablolar ile ifade edebilmek için yapmamız gereken dönüşümü içerir.

1. **Fiziksel Model**

VTYS ile ilgili ilk temas burada kurulacak olup, sistemin fiziksel olarak kurulması sağlanır.





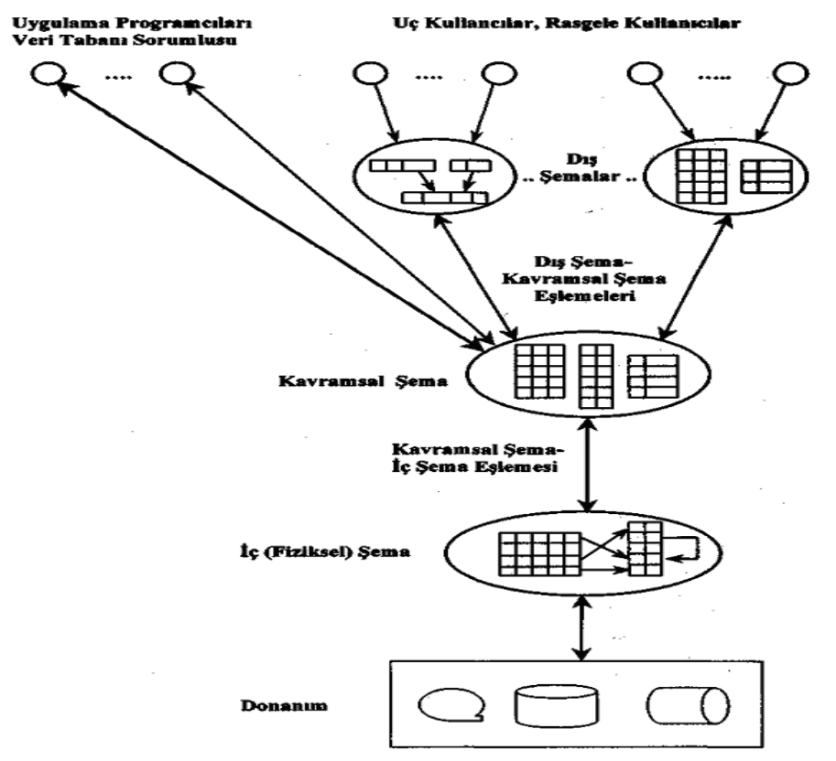
Şekil 5.6 - Kullanıcı Şemaları

**5.2.2.3 Şemalar**

Kullanılacak olan veri modelinde veritabanının kendisi ile veritabanının tanımlanması durumlarını ayırt edebilmek önemlidir. Veritabanı şeması planlama aşamasında oluşturulur ve daha sonradan büyük değişiklikler yapması beklenmez.

**5.2.2.4 VTYS Mimarisi**

* **İçsel Düzey:** Veritabanının fiziksel saklanma yapısını açıklar. Fiziksel veri modeli kullanır ve veritabanına erişim yolu ile veri saklamanın tüm detaylarını açıklar.
* **Kavramsal Düzey:** Kavramsal Düzey ise kavramsal şema içerir ve kullanıcılar için veritabanının yapısını açıklar. Gerçek fiziksel yapının detaylarını kullanıcıdan gizler, veri tipleri, varlıklar, ilişkiler, kullanıcı işlemleri ve sınırlamalar üzerine konsantre olmamızı sağlar. Daha yüksek seviyede veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.
* **Dışsal Düzey:** Bu düzey, dış şemalar veya kullanıcı görüşlerini içerir. Her dış şema veritabanının bir bölümünü açıklar ve her gruba kendi ilgilendiği görüşü sunarken, diğer bir gruptan ilgilenmediği görüşü saklar. Daha yüksek düzeyde veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

****Şekil 5.7 - Veri Tabanı Şemaları

**5.2.2.5 Veritabanı Dilleri ve Arabirimleri**

Sistemimizde veritabanı olarak SQL kullanıldı. Henüz yapım aşamasında olan sistemimizde MySQL tabanlı phpMyAdmin veritabanı yöneticisini kullandık.

| Veri Tanımlama Dili (VTD) | Kavramsal şemaları tanımlamak üzere veritabanı yönetici ve tasarımcı tarafından kullanılır. |
| --- | --- |
| Saklama Tanımlama Dili (STD) | İçsel şemayı tanımlamak için kullanılır. |
| Görüş Tanımlama Dili (GTD) | Görüş tanımlama dili kullanıcı görüşlerini tanımlamak ve kavramsal şemaya dönüştürmek amacıyla kullanılır. |
| Veri İşleme Dili (VİD) | Veri işleme dilidir. Veritabanı oluşturduktan sonra veritabanına veri eklemek, değiştirmek, silmek veya eklenmiş veriyi getirmek amacıyla kullanılır. |

Şekil 5.8 - Veri Tabanı Dilleri ve Arabirimleri

**5.2.2.6 Veritabanı Sistem Ortamı**

Veritabanı yöneticisi olarak phpMyAdmin kullandık. Tabloların oluşturulması, ilişkilendirilmesi, gerekli düzenlemelerin yapılması vb. işlevleri yerine getirdi.

**5.2.2.7 VTYS’nin Sınıflandırılması**

En fazla kullanılan veri modelleri ilişkisel, ağ, hiyerarşik, nesne-yönelimli ve kavramsal modellerdir. Bizim kullandığımız ise ilişkisel veri modeldir.

**5.2.2.8 Hazır Program Kütüphane Dosyaları**

Entegre olduğu için böyle bir şeye ihtiyaç duyulmadı.

**5.2.2.9 CASE Araç ve Ortamları**

* **MySQL:** phpMyAdmin
* **C#:** Visual Studio
* **PHP:** Visual Studio Code
* **Diğer eklenti diller:** Notepad**++ /** Visual Studio Code

**5.3 Kodlama Stili**

Bilinen algoritmaları kullanmak yerine kendi algoritmamızı yazdık. Bu sayede kendi ihtiyaçlarımız doğrultusunda daha avantajlı bir sisteme sahip olduk. Ayrıca bakım planlarında uygulanacak olan prosedürlere daha hâkim olduk.

**5.3.1 Açıklama Satırları**

Karmaşık her satır sonunda açıklama kodları eklendi.

**5.3.2 Kod Biçimlemesi**

Sistemi kodlarken en basit işlemine kadar tüm metotları ayırdık. Bu sayede her metot ve fonksiyonun ne iş yaptığı, kodun okunması aşamasında rahatça anlaşılabilir.

**5.3.3 Anlamlı İsimlendirme**

Kod biçimlemesi başlığında da belirttiğimiz gibi tüm metotlar ayrıldı. Ayrılan bu metotlar işlevlerini belirtecek şekilde isimlendirildi.

**5.3.4 Yapısal Programlama Yapıları**

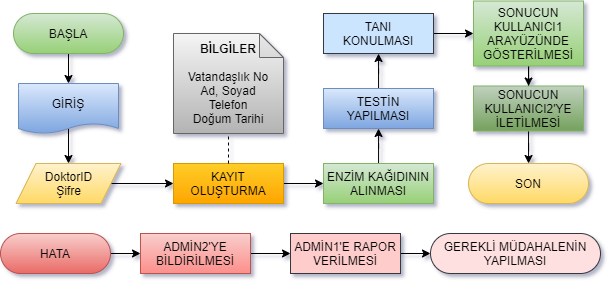
**Ardışık İşlem Aşamaları:** Gereken yerlerde kod parçacıklarını metotlar halinde ayırdığımızı belirtmiştik. İhtiyaç duyduğumuz farklı nesneler içinse sınıf yapısını kullandık.

**Koşullu İşlem Aşamaları:** Sistemimizin teşhis koyabilmesini sağlayan ana fonksiyonları karşılaştırma yöntemini kullanmaktadır. Bu yüzden projemizde koşullu işlem yapılarına bolca yer verilmiştir.

**Döngü Yapıları:**Sistemimizin karşılaştırma aşamasında birden fazla karşılaştırmaya yer verildiği için kod kalabalığını engellemek adına döngü yapılarına yer verilmiştir.

**5.4 Program Karmaşıklığı**

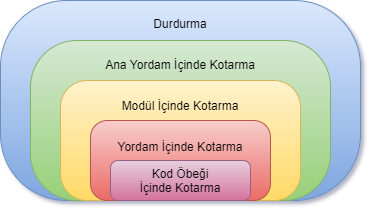
Programın karmaşıklığını ölçmek için McCabe karmaşıklık modelini kullandık. McCabe modeli programda kullanılan koşullu işlemlerin, program karmaşıklığına etki eden en önemli unsurlardan biri olduğunu savunur. V(G)=k-d+2p formülü ile hesaplanır.



Şekil 5.10 - Akış Diyagramı

**5.5 Olağandışı Durum Çözümleme**

Olağandışı durum, sistemin çalışması sırasında; geçersiz, hatalı, eksik ve yanlış veri oluşması durumunda sistemin beklenmedik şekilde sonlanmasıdır.



Şekil 5.11 - Olağan Dışı Halde Yapılacaklar

**5.5.1 Olağandışı Durum Tanımları**

Olası hata durumları tespit edilip kaynağı kaldırılmadan önceki aşamada sistemin beklenmedik şekilde sonlanmasını engellemek adına try-catch blokları kullanılmıştır.

**5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları**

Try-catch bloğu haricinde oluşacak hataların önüne geçmek adına hataya düşen sistemin her halükarda giriş ekranına yönlendirmesini sağladık.

**5.6 Kod Gözden Geçirme**

Projemizi geliştirme aşamasına başlamadan önce açık kaynak kodlu olan Git sisteminde projemize uygun şekilde yeni bir repository oluşturduk. Bu sayede projemizin her aşamasını ve versiyonunu ayrı ayrı inceleyip çıkarımlarda bulunabildik.

**5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi**

* Hataların fark edilmesi hedeflenir
* Bu aşamada hata kontrolü yapan ekibin küçük olması ve ekibe Admin1’in dâhil olmasında yarar vardır.
* Kalite ve bakım çalışmalarından ayrı tutulmaması gerekir. Fark edilen her hata kaybolmayacak şekilde saklanmalıdır.

**5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular**

Bir proje incelenirken örnek arayüzü ile ilgili soruların sorulması ve bu sorulara verilecek cevapların listelenmesi gerekir. Listelenen cevaplardan bazılarının cevabının olumsuz olması projenin reddedilmesine sebep olabilir.

**5.6.2.1 Öbek Arayüzü**

Öbekleri test etmek için belli sorular sorduk.

* Her öbek tek bir işlevsel amacı yerine getiriyor mu?
* Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?
* Öbek tek giriş ve çıkışlı mı?
* Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?

**5.6.2.2 Giriş Açıklamaları**

Giriş açıklamalarını test etmek için belli sorular sorduk.

* 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 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?

**5.6.2.3 Veri Kullanımı**

Veri kullanımlarını test etmek için belli sorular sorduk.

* 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?

**5.6.2.4 Öbeğin Düzenlenişi**

Referans alınacak sorular oluşturduk.

* Modüller birleşimi uyumlumu?
* Modüller arası veri aktarımları sağlanıyor mu?
* Bütün modüller birleştiğinde sistem çalışıyor mu?

**5.6.2.5 Sunuş**

Son kısımda sorulacak sorular şu şekildedir;

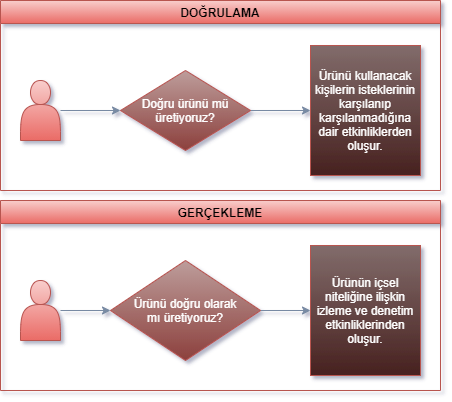
* Her satır, en fazla bir deyim içeriyor mu?
* Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?
* Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?
* Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?
*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

**6. DOĞRULAMA VE GEÇERLEME**

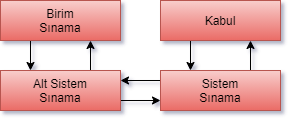
**6.1 Giriş**

Geliştirdiğimiz sistemin yazılımının doğrulanması ve geçerlenmesi, tüm geliştirme aşaması boyunca sürer. Bu aşamadaki etkinler aşağıdaki gibidir;

* Proje geçerlilik sürecinde oluşacak olan tüm çıktıların hatasız olması projeyi doğrular.
* Proje geliştirme aşamasında her modülün teknik yeterliliğinin incelenmesi ve gerekli çözümün uygulanması projeyi doğrular.
* Projenin her farklı aşamasında geliştirilen anahtar belirtimlerin öncekilerle karşılaştırılması projeyi doğrular.

  
Şekil 6.1 Doğrulama Geçerleme

**6.2. Sınama Kavramları**

****

Şekil 6.2

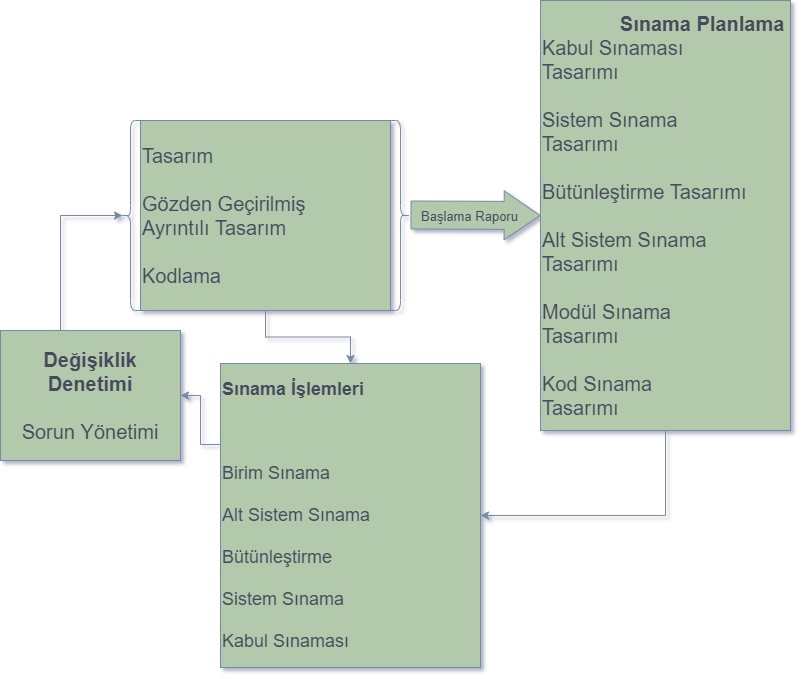
**Birim Sınama:** Sistemin kullanıcıları olan hasta ve doktor, kendi arasında iletişim sağladı ve sistemi kullandı.

**Alt Sistem Sınama:** Doktor ve yetkili sağlık personeli ortak modül kullanıp kendi arasında iletişim sağladı. Ortak arayüz kullanımında doğan hatalar tespit edildi ve giderildi.

**Sistem Sınama:** Sistemin test aşaması Admin1 tarafından incelendi, bulunan hatalar ve eksiklikler giderildi.

**Kabul Sınama:** Proje test aşamasını tamamladı. Pilot bölge haricindeki bölgelerde de kullanılması onaylandı.

**6.3 Doğrulama ve Geçerleme Yaşam Döngüsü**

****

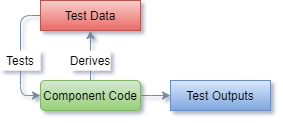
Şekil 6.3 - Yaşam Döngüsü

**6.4 Sınama Yöntemleri**

Sınama işlemi, sadece düzeltme görevi olarak bilinmemeli. Çünkü geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çabadır.

**6.4.1 Beyaz Kutu Sınaması**

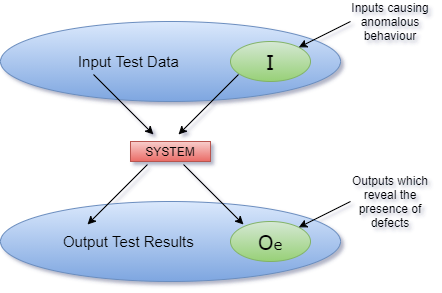
* Bütün bağımsız yolların en azından bir kere sınanması,
* Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
* Bütün döngülerin sınır değerlerinde sınanması,
* İç veri yapılarının denenmesi yapıldı.

****

Şekil 6.4 - Beyaz Kutu Sınaması

**6.4.2 Temel Yollar Sınaması**

* Eş değerlere bölme
* Uç değerler analizi
* Karar tablosu
* Sonlu durum makinesi
* Belgelenmiş özelliklere göre test
* Rastgele test
* Kullanım profili



Şekil 6.5 - Sınama Şekli

**6.5 Sınama ve Bütünleştirme Stratejileri**

Genellikle bütünleştirme ve sınama stratejileri birlikte değerlendirilir. Ancak bazı bütünleştirme stratejileri her sınama stratejisinin hedeflerini gözetmeyebilir. Örneğin yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır.

**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. Fakat bu sınama sistemi kullanılmadı.

**6.6 Sınama Planlaması**

Mutlaka planda olması gereken özellikler aşağıdaki gibidir

* **Test planı kimliği:** Test planının adı veya belge numarası
* **Giriş:**Test edilecek sistemin genel tanıtımı ve elemanlarının özeti. Ayrıca bu plandaki belgeler, kısaltmalar ve terim açıklamalarının belirtildiği kısım.
* **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 durumda olan fonksiyonları açıklar. Ayrıca bu fonksiyonların test edilememe sebeplerini de açıklar.
* **Geçti/Kaldı Kriterleri:** Bir test sonucunda sistemin geçmiş veya kalmış sayılacağını açıklamalıdır.
* **Test dokümanı:** Test süresince yapılan işlemleri alınan raporları elde edilen bilgileri rapor içinde sunulmalıdır.
* **Sorumluluklar:**Proje kapsamında çalışacak kişilerin hangi işten sorumlu olduğu takım lideri tarafından detaylı şekilde dokümante edilmelidir.
* **Riskler ve Önlemler:** Test planında varsayılan ve olası yüksek riskli durumları belirtir ve bu durumların olması durumunda, etkilerinin en aza indirilebilmesi için alınması gereken önlemleri açıklar.

**6.7 Sınama Belirtimleri**

Sınama işleminin nasıl yapılacağına ilişkin ayrıntıları ve sınama belirtimlerini içerir. Bu ayrıntılar temel olarak;

* Sınanacak programın modülleri ve modüllerinin isimleri
* Sınama türü, stratejisi
* Sınanacak veriler
* Sınama senaryoları

Projemizin karmaşıklığı yüksek olmadığı için otomatik sınama verisi üreten programlardan kaçındık ve verileri manüel olarak oluşturduk.

Sınama sırasında oluşacak senaryoları yenime sınama senaryosu üretebilmek için daha efektif bir şekilde hazırlanmalıdır. Çünkü bu verilerin hazırlanmasındaki temel amaç sınamayı etkin bir şekilde yapabilmektir.

Sınama işlemi sonrasında

* Sınamayı yapan,
* Sınama tarihi,
* Bulunan hatalar ve açıklamaları

gibi bilgiler eklenerek sınama raporları oluşturulur.

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

**6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

Bu etkinlikleri hiyerarşik olarak düzenlersek:

* Planlama aşamasında genel planlama sınaması yapılır.
* Çözümleme aşamasında sınama planı alt sistemler üzerinde çalışacak şekilde detaylandırılır.
* Tasarım aşamasında sınama planı detaylandırılır ve sınama belirtimleri oluşturulur. Oluşturulan bu belirtimler daha sonra eğitim aşamasında kullanılır.
* Gerçekleştirim aşamasında teknik sınamalar yapılır ve bu sınamaların raporları hazırlanır.
* Kurulum aşamasında sistemle ilgili son sınamalar yapılır ve bu sınamaların raporu hazırlanır.



Şekil 6.6 - Sınama Etkinlikleri

Sınama aşamasında bulunmuş olan her hata kaybolmayacak şekilde kaydedilir. Bu hatalara daha sonradan Admin1 tarafından müdahale edilir.

**Şekilsel Hatalar:** Arayüzdeki yazı ve şekillerin, kaybolması veya bozulması gibi durumlardır.

**Küçük Hatalar:** Sistemin beklenmeyen bir şekilde kapanmasını değil de ufak bir aksaklık yaşanmasına sebep olan hatalardır.

**Büyük Hatalar:** Sistemin beklenmeyen bir şekilde kapanmasına sebep olan hatalardır.

**Onulmaz Hatalar:** Veri kaybına yol açan ve sistemin her türlü çalışmasını engelleyen hatalardır.

| Gerçekleştirim ve Doğrulama-Zaman Diyagramı | | | |
| --- | --- | --- | --- |
|  | 7. Hafta | 8. Hafta | 9. Hafta |
| Gerçekleştirme | C:\Users\SAMED\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled-3.png | C:\Users\SAMED\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled-3.png | C:\Users\SAMED\Desktop\tik2.png |
| Doğrulama |  |  | C:\Users\SAMED\Desktop\tik2.png |

Şekil 6.7 - Gerçekleştirme Aşama ve Tarihleri Diyagramlar

**7. BAKIM**

**7.1 Giriş**

Yazılım bakımı, uluslararası standart belirleme organizasyonu olan IEEE (i triple e) tarafından belirli kriterlere ve aşamalara göre gerçekleştirilmektedir. Bakıma ilişkin standart IEEE 1219-1998 baz alınarak gerçekleştirilmektedir. IEEE tarafından sunulan bakım süreci şu adımları içermektedir:

1. Sorun tanımlama süreci
2. Çözümleme süreci
3. Tasarım süreci
4. Gerçekleştirim süreci
5. Sistem test süreci
6. Kabul test süreci
7. Kurulum süreci



Şekil 7.1 - Bakım Aşaması

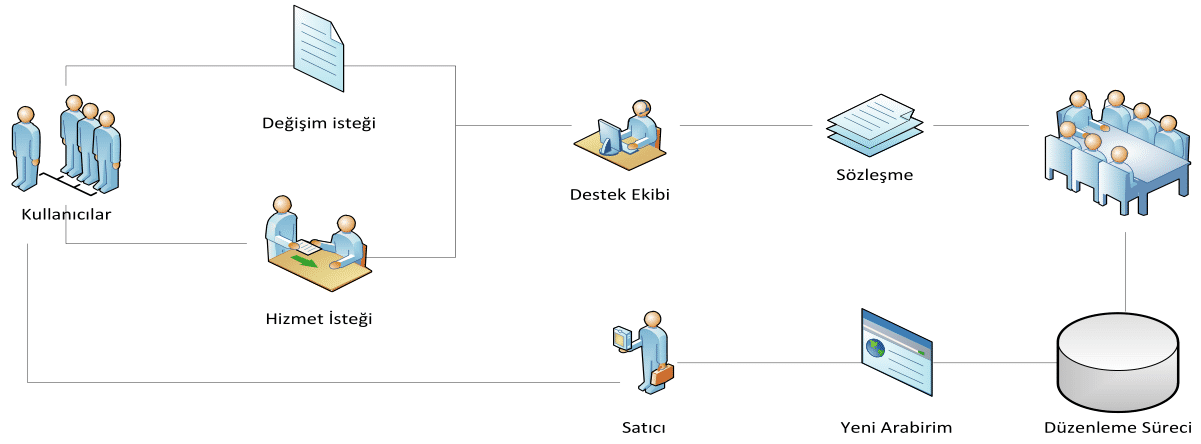
**7.2 Kurulum**

Kurulum süreci, geliştirilen ya da  değiştirilmiş yeni yazılım sürümünün, uygulama sahasına aktarılma işlemlerini içerir.

**7.3 Yerinde Destek Organizasyonu**

Sistemin kurulacağı her mekânda bir yetkili kişi atanacak(Admin2) ve sistemle ilgili detaylı eğitime tabii tutulacak. Herhangi bir hata çıkması durumunda bu yetkili kişi duruma müdahale edecek. Hatayı çözse de çözemese de proje yöneticisine (Admin1) bildirir.

**7.4 Yazılım Bakımı**

****

Şekil 7.2 - Bakım Aşaması

**7.4.1 Tanım**

**Düzeltici Bakım:**Teorik olarak bir yazılımın tümüyle sınanabilmesi olası olsa bile. pratikte bu sağlanamaz. Bu nedenle çalışan bir 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:** Uygulama yazılımları, işletme ya da kuruluşların günlük yaşamlarında yaptıkları işleri bilgisayar ortamında yapmalarını sağlayan araçlardır. Her kuruluş ya da işletme canlı bir varlık gibi düşünülebilir. Hiçbir işletme durağan değildir. Süreç içinde değişkenlik gösterir işletme ya da kuruluşlarda değişiklik, yapılan işlerin yapılma tarzının 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 tanımlanır.

**En İyileyici Bakım:** Zaman zaman uygulama yazılımlarının çalışma performanslarının iyileştirilmesi amacıyla yapılan çalışmalar "En İyileyici Bakım" olarak tanımlanır.

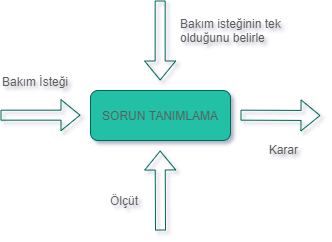
**7.4.2 Bakım Süreç Modeli**

****

Şekil 7.3 - Bakım Süreç Modeli

1. **Adım:** Sorun tanımlama süreci

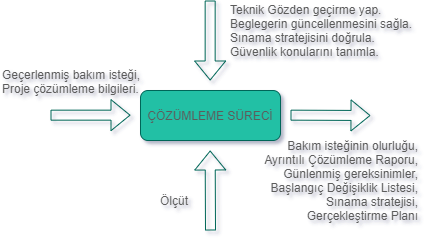
Bakım için ne yapılıyor ve sorun tam olarak ne bu duruma bakılır.



Şekil 7.4 - Sorun Tanımlama

1. **Adım:** Çözümleme süreci

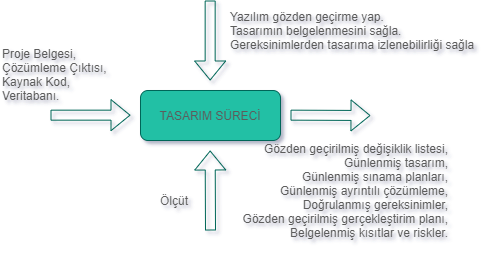
Karar doğrultusunda problemi kâğıt üzerinde çözme



Şekil 7.5 - Çözümleme Süreci

1. **Adım:** Tasarım süreci

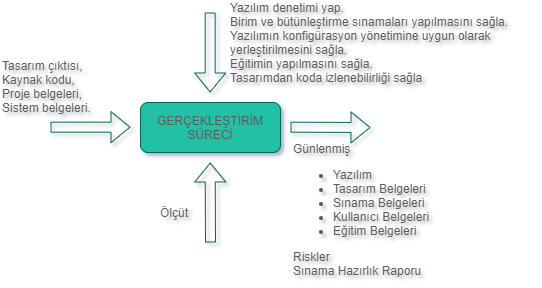
Sistemin çözümlenmesinden sonra sıra güncelleştirme işlemine geldi.



Şekil 7.6 - Tasarım Süreci

1. **Adım:** Gerçekleştirim süreci

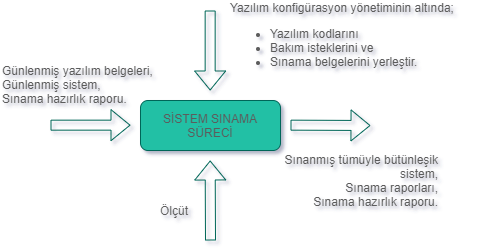
Sıra tasarımı yapılan sistemin gerçekleştirilmesine geldi.



Şekil 7.7 - Gerçekleştirim Süreci

1. **Adım:** Sistem test süreci

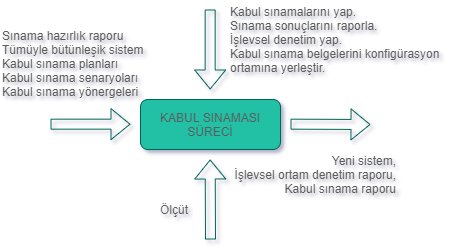
Tekrardan tasarlanan sistemin sürecini tekrar ele almak gerekiyor.



Şekil 7.8 - Sınama Süreci

1. **Adım:** Kabul test süreci

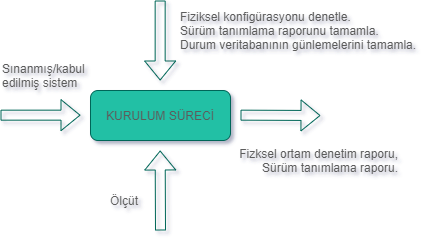
Sınadığımız sistemin müşteri ile birlikte sınanması.



Şekil 7.9 - Kabul Sınama Süreci

1. **Adım:**Kurulum süreci

Kabul test sürecini başarılı bir şekilde geçen sistemin kurulum aşamasına geçiyoruz.



Şekil 7.10 - Kurulum Süreci

**8. SONUÇ**

Sonuç olarak projemiz yürürlüğe girdiğinde ne gibi artıları sunduğunu bu raporda belirttik. Ayrıca basit bir kontrol mekanizması kullanarak sancılı bir test işlemini daha konforlu ve hızlı bir hale getirdi.

Hem sağlık çalışanları hem de hastalar için kolaylık sağlayan bu sistem sayesinde alerji testi yapmak artık çok daha kolay olacak. Sağlık çalışanları daha az uğraş verecek ve daha az zaman kaybedecek. Hastalar ise canları yanmadan ve sonuç verileri kaybolmadan alerji testi yapabilecek.

Bu süreçte desteklerini bizden esirgemeyen herkese teşekkürlerimi iletiyorum.

**9. KAYNAKLAR**

* <https://en.wikipedia.org/wiki/Spiral_model>
* <https://en.wikipedia.org/wiki/Software_development>
* <https://en.wikipedia.org/wiki/Function_model>
* <https://en.wikipedia.org/wiki/Data_modeling>
* <https://devdocs.io/css/>
* <https://devdocs.io/html/>
* <https://devdocs.io/javascript/>
* <https://www.php.net/docs.php>
* <https://www.phpmyadmin.net/docs/>
* <https://docs.microsoft.com/tr-tr/dotnet/csharp/>