Kimlik Doğrulama Projesi

için

Yazılım Gereksinimleri

Versiyon <1.0>

Hazırlayanlar

**Grup İsmi: CROSSFİRE**

|  |  |  |
| --- | --- | --- |
| Mert DEMİR | 14542516 | mert.demir96@hotmail.com |
| Mustafa ARSLAN | 15541552 | mustafaarslan1001@gmail.com |
| Emre R. AYDIN | 15542509 | emramzn@gmail.com |
| Mert Can AKTAŞ | 16541530 | mertcanaktas6@gmail.com |
| Hakan YILMAZ | 15541536 | yilmaz076@gmail.com |
| Özgenur DAŞLI | 15541502 | nrdsl97@gmail.com |
| Nisa Nur SÖYLEMEZ | 16541520 | Nsoylemezz.98@gmail.com |

|  |  |
| --- | --- |
| Ögretmen: | Fatih ÖZKAYNAK |
| Bölüm: | Yazılım Mühendisliği |
| Öğretim Asistanı: | Gamze Pelin AKSOY |
| Tarih: | 04/11/2019 |

İçindekiler

Revizyon iii

1 GİRİŞ 1

1.1 Proje Amacı 1

1.2 Proje Kapsamı 1

1.3 Hedef Kitle ve Döküman İçeriği 1

1.4 Tanımlamalar ve Kısaltmalar 1

1.5 Belge Kuralları 1

1.6 Referanslar ve Teşekkürler 1

2 Kullanılan Geliştirme araçları ve ortamları 2

2.1 Genel Bilgi 2

2.2 JavaScript ve Vue.Js 2

2.3 Node.Js 2

2.4 HTML, CSS ve Bootstrap 2

2.5 Visual Studio Code 2

2.6 GİT VE GİTHUB 2

2.7 Varsayımlar ve Bağımlılıklar 3

3 ÖZEL Gereksinimler 4

3.1 Dış Arayüz Gereksinimleri 5

3.1.1 Kullanıcı Arayüzleri 5

3.1.2 Donanım Arayüzleri 5

3.1.3 Yazılım Arayüzleri 5

3.1.4 İletişim Arayüzleri 6

3.2 Fonksiyonel Gereksinimler 6

3.3 Hareket Gereksinimleri 6

4 DİĞER FONKSİYONEL GEREKSİNİMLER 7

4.1 Performans Gereksinimleri 7

4.2 Güvenlik ve Güvenlik Gereksinimleri 7

4.3 Yazılım Kalite ÖZellikleri 7

4.3.1 Güvenilirlik 7

4.3.2 Sürdürülebilirlik 7

4.3.3 Kullanılabilirlik 7

4.4.4 Taşınabilirlik 7

5 yAZILIM GELİŞTİRME MODELİ 8

5.1 scrum 8

Grup AKTİVİTELERİ 9

Revizyon

| Versiyon | Başlıca Yazar(lar) | Versiyonun Açıklaması | Tamamlanma Tarihi |
| --- | --- | --- | --- |
| 1.0 | Nisa SÖYLEMEZ  Özgenur DAŞLI Emre AYDIN  Mert DEMİR  Mertcan AKTAŞ  Mustafa ARSLAN  Hakan YILMAZ | Kimlik Doğrulama Projesi için gerekli olan gereksinimler yazıldı. | 06/11/2019 |

# 

# Giriş

## Proje Amacı

Öğrencilerin bir birleri yerine yoklama kağıdın da imza atmalarını engellemek. Böylelikle imza kontrolü tam olarak sağlanılacak.

## Proje Kapsamı

İlk olarak yazılım mühendisliği güncel konuları dersinde uygulanacak olup daha sonra tüm bölümler ve diğer fakültelerde uygulamaya geçilecek.

## Hedef Kitle ve Döküman İçeriği

Hedef kitle öğrenciler olup bir ömğrencinin başka bir öğrenci yerine imza atmasını engellemek amacıyla bu uygulama yapılmıştır. Dökümanda ağırlıklı olarak bu konulara değinilecektir.

## Tanımlamalar ve Kısaltmalar

Öğrenci: Derse gelip yoklamaya imza atan kişi.

Öğretmen:Yoklama uygulamasını yöneticek olan kişi.

## Belge Kuralları

Genel olarak bu belge IEEE formatlama gereksinimlerini takip eder. Dökümanda Arial font 11 veya 12 boyutları başlıklar için ise Arial font 14 boyutları kullanılmıştır. Yorum satırları italik yazılmıştır. Kelimeler arası tek boşluk kullanılmıştır.

## Referanslar ve Teşekkürler

Erhan Sarıdoğan – Yazılım Mühendisliği Temelleri

[www.draw.io](http://www.draw.io)

# Genel Açıklama

## Genel Bilgi

**Programlama Dilleri:**

1. Front-end: JavaSCRİPT, VueJS, HTML, CSS
2. Back-end: NodeJS

**Tasarım:** Bootstrap

**Düzenleyici:** Visual Studio Code

**Mock Data:** Test edilen sistemin yapması beklenen işlemlerin yapılıp yapılmadığını doğrulamak için kullanılır.

## JavaScript ve Vue.Js

JavaScript, web tarayıcılarında yaygın kullanılan betik dil olduğundan projenin web tarafı için en uygun geliştirme aracıdır. Framework’ü olan VueJS, kolay, anlaşılabilir yapısı ile proje sonrasında değişiklik yapma avantajı sağlıyor.

## Node.Js

NodeJS, hem back-end hem de front-end’te aynı dil kullanılmasını sağlıyor. Bu projede JS kullanıldığı için NodeJS en uygun geliştirme aracıdır.

## HTML, CSS ve Bootstrap

HTML, web sayfası oluşturmak için kullanılan en temel betik dildir. CSS, CSS, HTML etiketlerimizin görsel olarak biçimlendirilmesini sağlar. Bootstrap, web sayfalarında, HTML ve CSS tabanlı tasarım şablonlarını içerdiğinden tasarım aşamasında kullanılır.

## Visual Studio Code

VS Code, hata ayıklama özellikleriyle, gelişmiş web uygulamaları üstünde kodları düzenlemeye, yeniden tanımlamaya ve optimize etmeye yarar. JS ve NodeJS dillerini destekliyor.

**2.6 Git ve GitHub**

Git, yazdığımız projeleri ve uygulamaları internet üzerinde tutmamızı ve yönetmemizi sağlayan versiyon kontrol sistemidir. **GitHub**, sürüm kontrol sistemi olarak Git kullanan yazılım projeleri için bir depolama servisidir. Proje boyunca yapılan değişiklikler için kullanılıyor.

## 2.7 Varsayımlar ve Bağımlılıklar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ölçüm parametresi | Sayı | Ağırlık | Toplam | |
| Kullanıcı Girdi Sayısı | 3 | 4 | 12 | |
| Kullanıcı Çıktı Sayısı | 4 | 5 | 20 | |
| Kullanıcı Sorgu Sayısı | 4 | 4 | 16 | |
| Kütük Sayısı | 6 | 10 | 60 | |
| Dışsal Arayüz Sayısı | 2 | 7 | 14 | |
| Ana işlev Nokta Sayısı |  | | 122 | |
| Teknik karmaşıklık Sorusu | | | | Puan |
| Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu ? | | | | 5 |
| Veri iletişimi gerekiyor mu? | | | | 5 |
| Dağıtık işlem işlevleri var mı? | | | | 4 |
| Performans kritik mi? | | | | 5 |
| Sistem mevcut ve ağır yükü olan bir iletişim ortamında mı çalışacak? | | | | 4 |
| Sistem, çevrimiçi veri girişi gerektiriyor mu? | | | | 4 |
| Çevrimiçi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu? | | | | 3 |
| Ana kütükler çevrimiçi olarak mı güncelleniyor? | | | | 5 |
| Girdiler, çıktılar, kütükler yada sorgular karmaşık mı? | | | | 4 |
| İçsel işlemler karmaşık mı? | | | | 4 |
| Tasarlanacak kod yeniden yazılabilir mi olacak? | | | | 4 |
| Dönüştürme ve kurulum, tasarımda dikkate alınacak mı? | | | | 3 |
| Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor? | | | | 5 |
| Tasarlanan uygulama kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak? | | | | 4 |
| Toplam | | | | 59 |

**0**: Hiçbir etkisi yok.  
**1**: Çok az etkisi var.  
**2**: Etkisi var.  
**3**: Ortalama etkisi var. **4**: Önemli etkisi var.  
**5**: Mutlaka önemli, kaçınılmaz.

# Özel Gereksinimler

## Dış Arayüz Gereksinimleri

### Kullanıcı Arayüzleri



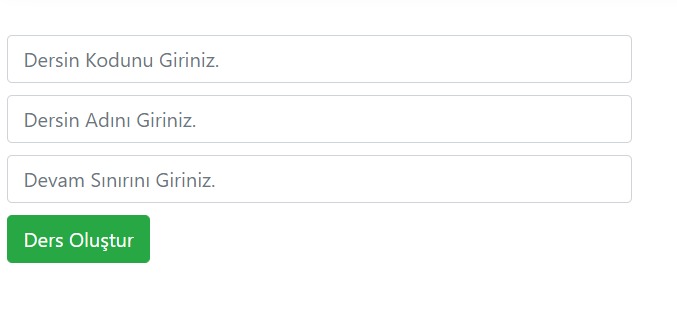


**Şekil 1:** Öğrenci Giriş Arayüzü

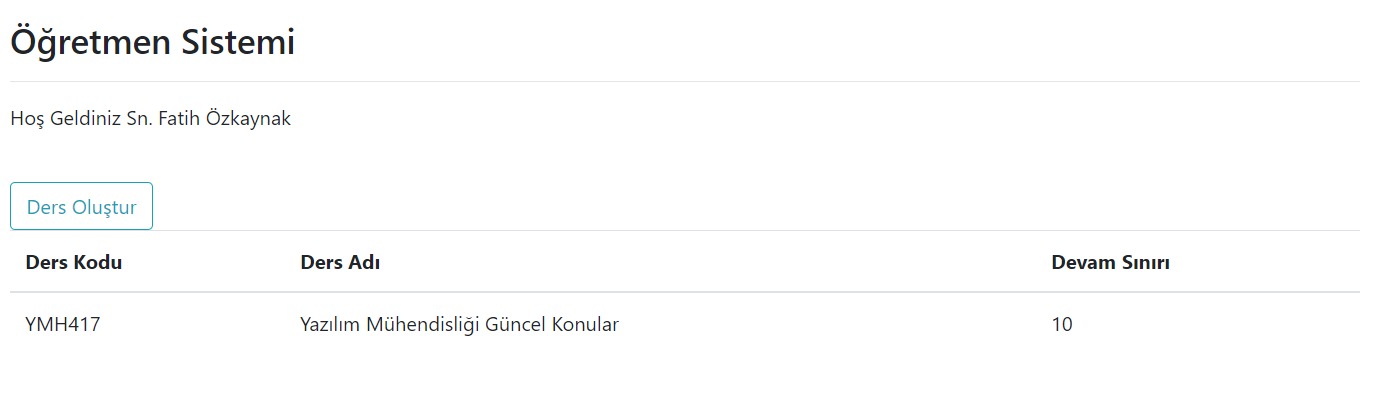


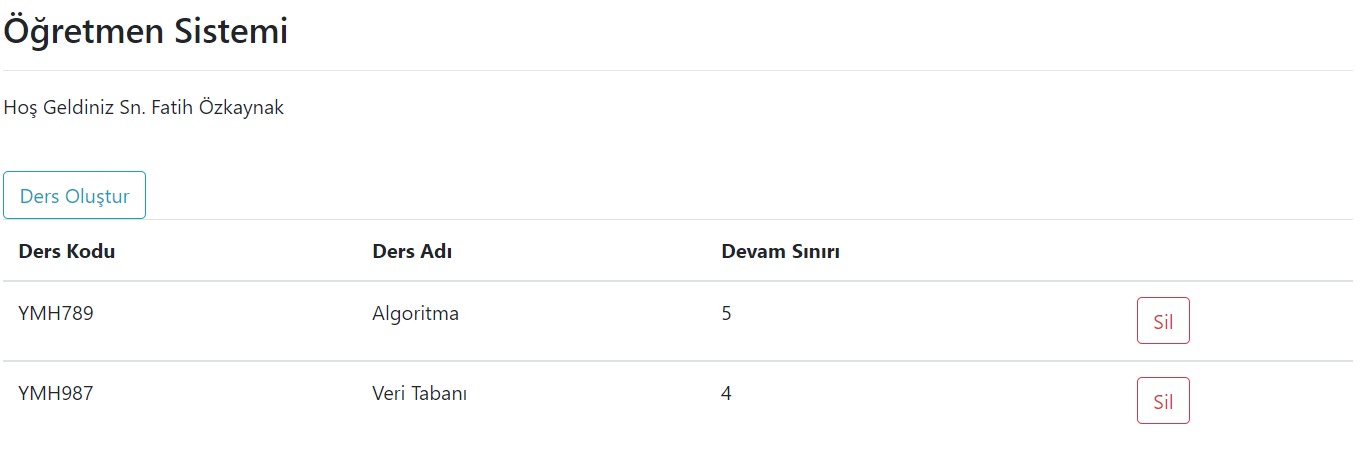
**Şekil 2:** Öğretmen Giriş Arayüzü

**Şekil 3:** Öğrenci Arayüzü



**Şekil 4:** Derse Giriş Arayüzü





**Şekil 5:** Öğretmen Arayüzü

### Donanım Arayüzleri

Ağ üzerinde veritabanlarımızı tutmak için özelleştirilmiş bir sunucuya ihtiyacımız olacaktır.İstemciler suncu üzerinde tutulan verilere, verilen yetkiler doğrultusunda erişip çeşitli sorgulamalar yapmak için kullanılıcaktır.

### Yazılım Arayüzleri

**İş Sistemleri Planlama Araçları**

* Microsoft Visio
* SWOT Analizi
  + Yapmakta olduğumuz veya yapmayı planladığınız iş için sizin Güçlü, zayıf noktaları dışsal fırsatları ve dışsal tehditleri ortaya koyan analizdir

**Proje Yönetim Araçları**

* Microsoft Visio
* Provital Tracker

**Proje Çözümleme Ve Tasarım Araçları**

* Draw.io

**Programlama Araçları**

* Visual Studio
* Sql Server Management Studio

**Sınama Araçları**

* QAComplete
* QTest
* PractiTest

**Prototipleme ve Benzetim Araçları**

* Invision studio
* Sketch

**Destek Araçları**

* Teamviewer
* Alpemix

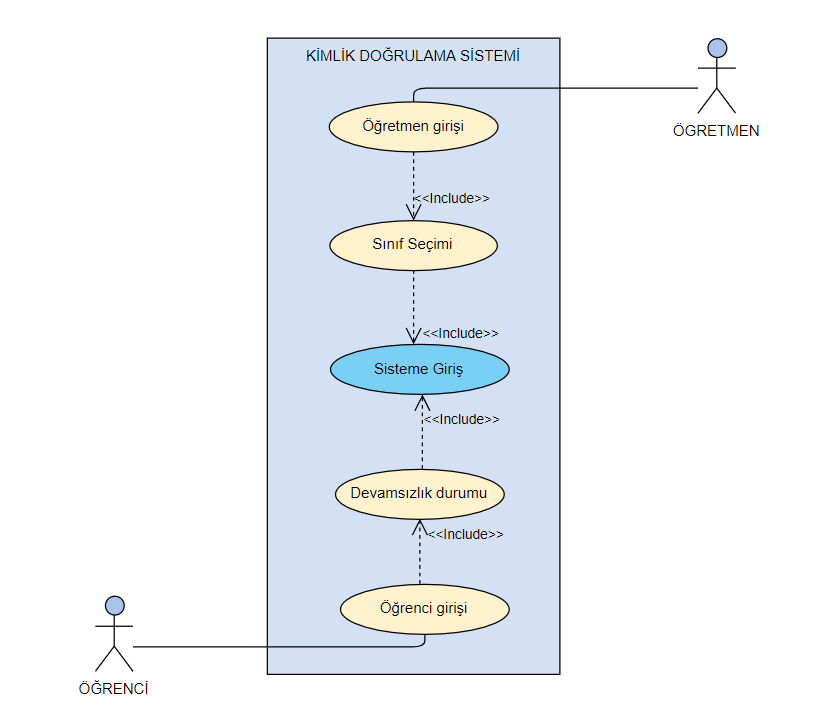
## Fonksiyonel Gereksinimler

Öğrencilerin bir birleri yerine imza atmaları engellenmesi sağlanılacak.

## Hareket Gereksinimleri

### Use Case Diyagramı

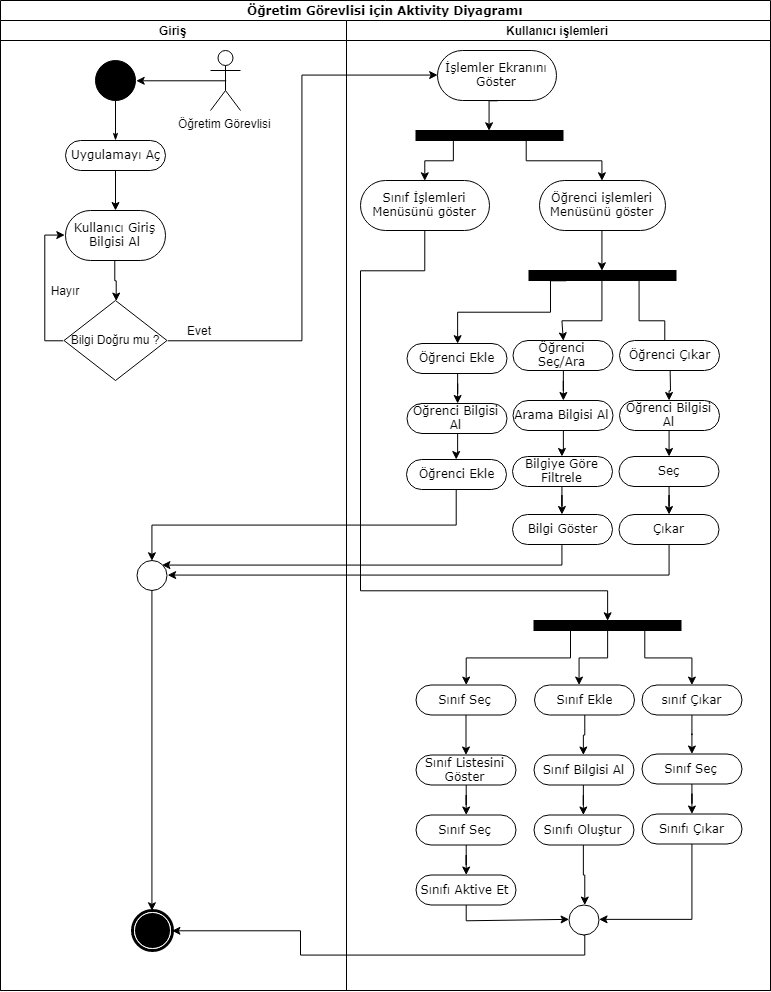
Use case, en basit şekliyle ifade etmek gerekirse, bize herhangi bir sistemin nasıl davrandığını ve çalıştığını gösterir. Use case’ler, bir kullanıcının spesifik bir amaca ulaşabilmek için yazılımsal bir sistem içerisinde atması gereken adımları tanımlamak için kullanılırlar.



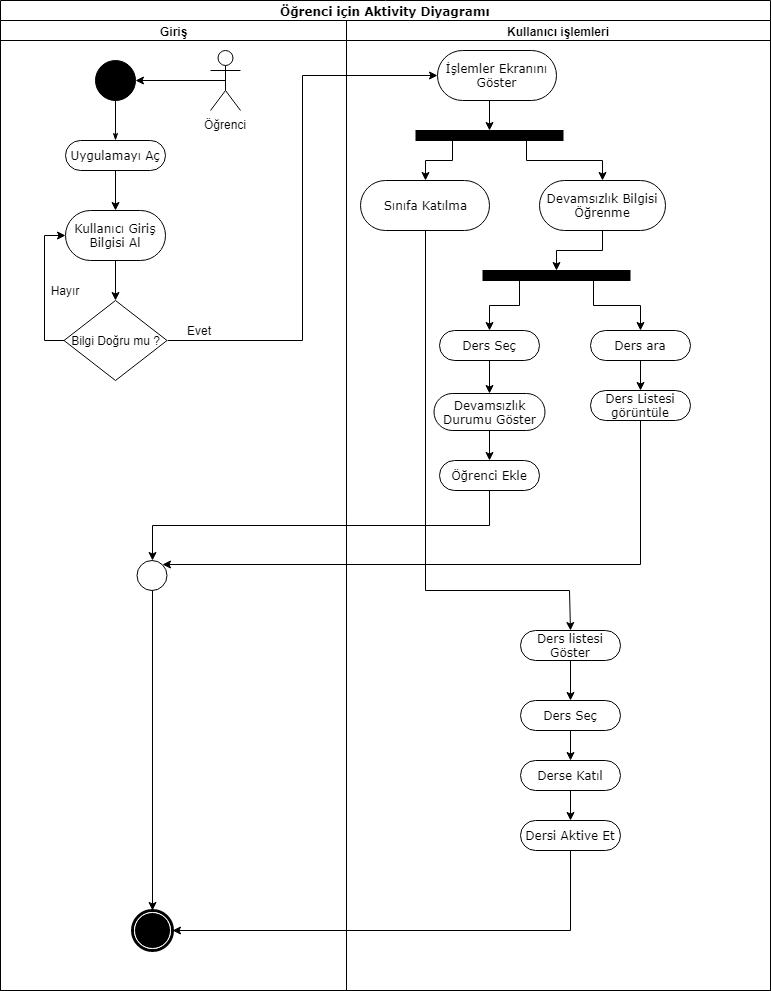
**Şekil 6:** Use-Case Diyagramı.

### Activity Diyagramı

Activity Diyagramları sistemin analistine, kullanıcısına ve mimarına kadar sistemin yapısı hakkında grafik tabanlı işleyişinin bilgisini sunar. Kimlik doğrulama projesinde iki farklı kullanıcı tipi bulunluğundan iki farklı activity diyagramı ile sistemin genel fonksiyonel işleyişi hakkında bilgi sunulmuştur.



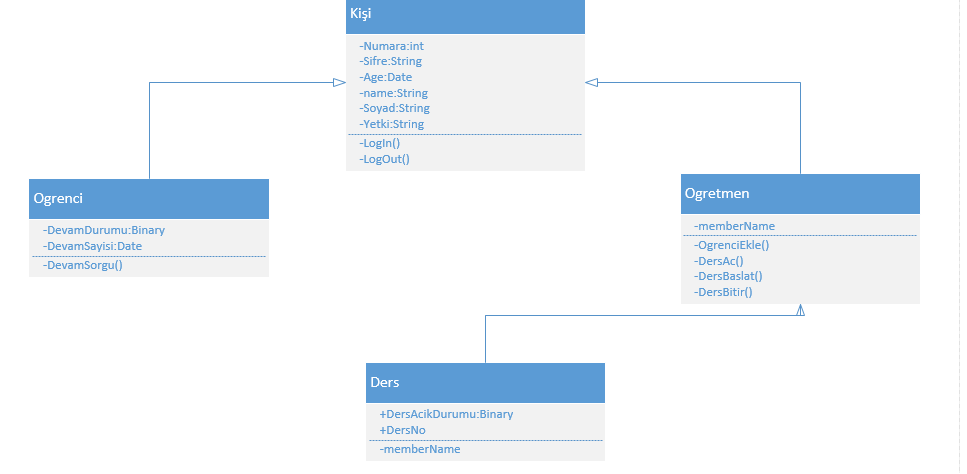
**Şekil 7:** Öğretim Görevlisi Activity Diyagramı.



**Şekil 8:** Öğrenci Activity Diyagramı.

### Class Diyagramı

Sınıf diyagramları projemizdeki sınıfların , özelliklerinin ve birbirleri arasaındaki ilişkilerin gösterilmesi için kullanılır. Diagramda Öğretmen , öğrenci ve ders sınıflarının bulunduğu özellikler ve fonksiyonları Şekil 7 de gösterilmiştir..



### Ekran AlıntısıEkran Alıntısı

**Şekil 9:** Class Diagramı

# Diğer Fonksiyonel Olmayan Gereksinimler

## Performans Gereksinimleri

İstenilen verileri en iyi çalışma zamanında kullanıcıya aktarılması veya gösterilmesini sağlamak asıl amaçtır. Bunun için veri tabanı erişim tepki süresi önem arz etmektedir.

Ağaç veri yapısı ile sistemin kayıtları göstermesi log(n) zamanda gerçekleşecektir.

## Güvenlik ve Güvenlik Gereksinimleri

Yazılım güvenilirliği, sistem veya bileşenlerinin, belirli bir ortamda, belirli bir zaman dilimi içinde kendilerinden beklenilen işlevleri yerine getirebilme olasılığı olarak tanımlanmaktadır.

Sisteme ya da bilgiye erişim bakım aşamasında belirli süreçler ile siber güvenlik uzmanı tarafından denetlecektir.

Sistem yedeklemesi günlük olarak yapılacaktır.

## Yazılım Kalite Özellikleri

### Güvenilirlik

Tarayıcı veya işletim sistemi hatasından sistem çökmez veya askıda kalamaz.

### Sürdürülebilirlik

Bütün kodlar tam olarak belgelenmelidir. Her işlev, ön ve sonlandırmalarla yorumlanacaktır.

Tüm program dosyaları yazarlık ve son değişiklik tarihi ile ilgili yorumları içermelidir. Son olarak, kod gelecekteki değişikliklere izin vermek için modüler olacaktır.

### Kullanılabilirlik

Kullanıcıların belirli bir eğitimi gerekmemektedir. Kullanıcı arayüzü anlaşılması kolay olacak ve operatör yazılımı hemen kullanmaya başlayabilmelidir.

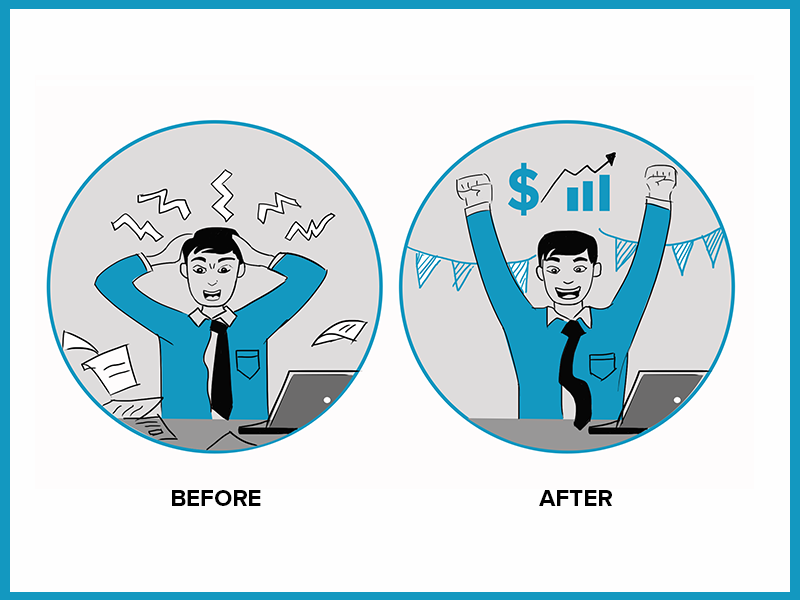
### Taşınabilirlik

Yazılım bir Microsoft Windows platformunda tarayıcı üzerinde çalışacak şekilde tasarlanacaktır. Windows XP'ye kadar ve Windows XP dahil olmak üzere herhangi bir Windows sürümünde kullanılmasına izin verecek şekilde taşınabilir olacaktır.

Yazılım Geliştirme Modeli

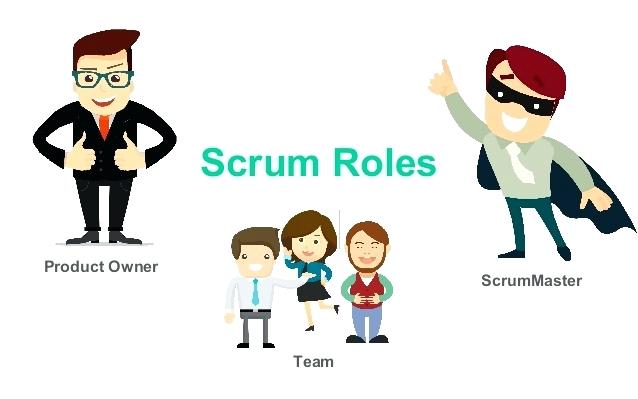
## Scrum

Scrum sürekli bir değişime ve gelişime adapte olmaya imkan veren bir Framwork’tür. Günlük toplantılarla iletişim halinde olan, iş birliği ile takımı organize eden, teşvik eden ve cesaretlendiren bir temele sahiptir.



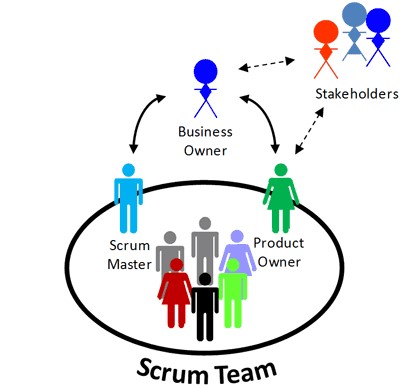
**5.1.1 SCRUM SÜRECİ**

Scrum süreci, müşterinin ihtiyaçları veya durumla alakalı istekleri doğrultusunda ekstra bir fikirlerinin olabileceğine dayanarak şekillenir. Scrum, ani fikir değişikliklerine hızlı ve etkili bir şekilde cevap geliştiren, sorunlara çözüm üreten takımın işini optimize etmeye odaklanır.

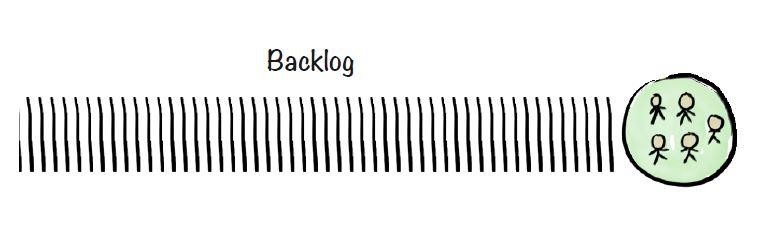
**Scrum’da 3 rol vardır;**

* + - Product Owner
    - Scrum Master
    - Team Member

**5.1.2 SCRUM TAKIMI**

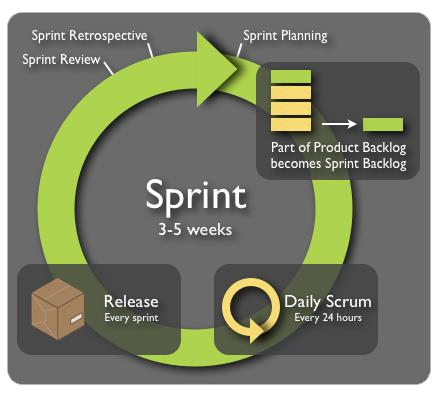
Ürün Sahibi, Geliştirme Ekibi ve Scrum Master’dan oluşur. Takım kendi kendini örgütler. Böylece kendi içerisinde uyum içinde olan takımlar daha başarılı sonuçlar alırlar. Scrum takım modeli esneklik, yaratıcılık ve verimliliği optimize etmek için tasarlanmıştır

**5.1.3 BACKLOG**



* Müşteriden ve son kullanıcıdan gelen gereksinimleri içerir.
* "Ne yapacağız" sorusunun yanıtını içerir.
* Herkese açık ve herkes tarafından müdahale edilebilir
* Risk, iş değeri, zaman gibi kavramlara göre ürün sahibi tarafından sıralandırılır.
* User Story'lerden oluşur.

**5.1.4 SPRİNT**



* Belirli bir süreye sahiptir.
* Sonunda ortada değeri olan bir çıktı olmalıdır.
* Toplantılarla içerik belirlenir.
* Sprint süresi boyunca her gün toplantılar yapılır

**5.1.5 USER STORY**

Müşteri, son kullanıcı veya ürün sahibi için değerli olan ve anlam ifade eden genellikle fonksiyonel özelliklerin belirtildiği ifadelerdir. Her bir user story farklı bir boyuttadır. Somut olarak bakarsak, bir projedeki her bir gereksinim için gereken iş gücü ve zaman aynı değildir. Bu sebeple ürün backlogları sprintlere bölünürken, user storylerin boyut ve öncelikleri göz önünde bulundurulur.

**5.1.6 SCRUM MASTER**

Geliştirme Takımı’nı, Sprint içinde kurumda bulunan bütün elemanlar rahatsız edebilir, takım üyelerinin kendi aralarında ki anlaşmazlıklardan kaynaklı bir sorun oluşabilir. Bunun yanında dış etkenlerde takımı etkileyebilir. İş ortaklığı yapan firmalar, kurumdaki diğer Scrum takımları vb. sebeplerden ötürü geliştirme takımının motivasyonu bozulabilir. Bütün bu durumlarda Scrum Master araya girerek bu sorunlara el atan ilk kişi olur. Bu sorunlara göre Scrum Master takımı toplayabilir veya yönetim ile görüşerek bu sorunları çözmelerini isteyebilir.

Stand-up meeting (ayakta yapılan kısa toplantılar) adı verilen günlük toplantılar sırasında Scrum Master ekip üyelerine aşağıdaki üç soruyu sorar:

1. Dün ne yaptın?
2. Bugün ne yapacaksın/ ne yapmayı planlıyorsun?
3. Önünüzde planlarınızı gerçekleştirmenize sorun olacak engeller var mı?

Bu sorular yapılan işlerin tekrar gözden geçirilmesini, ilerlemeyi engelleyecek problemlerin belirlenmesini, takımın kısa sürede hedeflerine ulaşmasını ve hedefin başarılı olması için gerekenleri sorgulamayı amaçlar. Scrum metodunun asıl amacı ekipte ki bütün bireyler arasında etkileşimi arttırarak bir bütün halinde çalışmasını sağlamaktır.

****

**5.1.6.1 İYİ BİR SCRUM MASTER NELER YAPMALI ?**

1. İşleri kolaylaştırıcı olmalı,

* Takımın çalışmasını kolaylaştırmak. Scrum Master takıma ne yapacağını söylememeli fakat takım üyelerini yaptıkları işlerde desteklemelidir.
* Takıma Scrum ilkeleri çerçevesinde koçluk yapmak.
* Sprint zamanlarının belirlenmesine yardım etmek.
* Takımın sprint süresince nelerin yapılacağında hemfikir olmasında takıma yardımcı olmalıdır.
* Geliştirme sürecinin Scrum kurallarına uygun olarak gerçekleştiğinden emin olmalıdır.

1. Takımı kendi kendine organize olabilmesi motive etmeli,

* Scrum Master süreçte yardımcı olmalı ancak aşırı korumacı ya da kontrolcü olmamalıdır.

1. Sürekli gelişmelere açık olmalı,

* Değişiklikleri iyi olarak görüp ve sürekli değişime teşvik etmelidir.

1. Ürün ve müşteri ihtiyacları hakkında iyi derecede bilgiye sahip olmalıdır.



GRUP AKTIVITELERI

|  |  |
| --- | --- |
| **Yazarlar** | **Yaptığı Bölümler** |
| Mert DEMİR | 1. GİRİŞ    1. PROJE AMACI    2. PROJE KAPSAMI    3. HEDEF KITLE VE DÖKÜMAN İÇERIĞI    4. TANIMLAMALAR VE KISALTMALAR    5. BELGE KURALLARI    6. REFERANSLAR VE TEŞEKKÜRLER |
| Özgenur DAŞLI | 2. GENEL AÇIKLAMA  2.1. GENEL BİLGİ  2.2. JAVASCRIPT VE VUEJS  2.3. NODEJS  2.4. HTML, CSS VE BOOTSTRAP  2.5. VİSUAL STUDİO CODE  2.6. GİT VE GİTHUB  2.7. VARSAYIMLAR VE BAĞIMLILIKLAR |
| Emre AYDIN  Mustafa ARSLAN  Hakan YILMAZ | 3. ÖZEL GEREKSINIMLER  3.1. DIŞ ARAYÜZ GEREKSINIMLERI  3.1.1. KULLANICI ARAYÜZÜ  3.1.2. DONANIM ARAYÜZÜ  3.1.3. YAZILIM ARAYÜZÜ  3.2. FONKSIYONEL GEREKSINIMLER  3.3. HAREKET GEREKSINIMLERI  3.3.1. USE CASE DIYAGRAMI  3.3.2. ACTİVİTY DIYAGRAMI  3.3.3. CLASS DIYAGRAMI |
| Mertcan AKTAŞ | 4. DİĞER FONKSİYONEL OLMAYAN GEREKSINIMLER  4.1. PERFORMANS GEREKSİNİMLERİ  4.2. GÜVENLIK VE GÜVENLIK GEREKSINIMLERI  4.3. YAZILIM KALİTE ÖZELLİKLERİ  4.3.1. GÜVENİRLİLİK  4.3.2. SÜRDÜRÜLEBİLİRLİK  4.3.3. KULLABILABİLİRLİK  4.3.4. TAŞINABİLİRLİK |
| Nisa SÖYLEMEZ | 5. YAZILIM GELİŞTİRME MODELİ  5.1. SCRUM |