ISPARTA UYGULAMALI BİLİMLER ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ YAZILIM MÜHENDİSLİĞİ GERÇEKLEŞTİRİM RAPORU

2023

|  |  |  |  |
| --- | --- | --- | --- |
| **2022-2023 YAZILIM MÜHENDİSLİĞİ DERSİ PROJE EKİPLERİ** | | | |
| **Proje** | **Ekipler** | **Ekip Sorumlusu** | **Takım Adı** |
| **1-Şirket Muhasebe-Harcama Yönetim Sistemi**  **İş Analitiği**  **Veri analizi programları ve raporlama**  **Bütçe yönetimi** | **2012721033 HÜSEYİN KARDEŞ**  **2012721001 HACER ÇİFTÇİ**  **2012721017 RAFET KARACA**  **2012729005 HAMZA GÜLEN** | **2012721033 HÜSEYİN KARDEŞ** | **SMHYS-Takımı** |
| **2- Şirket İnsan Kaynakları Yönetim Sistemi**  **(Kurumsal Kaynak Planlama-ERP Çözümleri)** | **2012721004 ENES CAN ENGİNSOY**  **2012721019 İREM EBRAR BOZKURT**  **2012721034 ATAKAN BERK YILMAZ**  **2012721313 ABDULLAH ESMAIL** | **2012721004 ENES CAN ENGİNSOY** | **SIKYS-Takımı** |
| **3-Şirket Elektronik Bilgi Yönetim Sistemi EBYS**  **Dosya arşivleme Sistemi** | **2012721002 KUDRET CAN YALIM**  **2012721014 BİLGE GÖKKAYA**  **2012721036 YASEMİN BULUT**  **2012729003 SELMAN KOYAN** | **2012721002 KUDRET CAN YALIM** | **SEBYS-Takımı** |
| **4-Şirket Üretim Yönetim Sistemi** | **2012721007 FURKAN ÇAĞRI ERKOCA**  **2012721018 ÖMER SOYDİNÇ**  **2012721030 SERKAN ÖZDEMİR**  **2012729004 BARIŞ YILDIRIM** | **2012721007 FURKAN ÇAĞRI ERKOCA** | **SUYS-Takımı** |
| **5-Şirket İç Denetim ve Kalite Yönetim Sistemi** | **2012721010 MUHAMMED ALİ ARI**  **2012721026 MELİH KILIÇ**  **2112721060 OSMAN CEYLAN**  **2012721037 FATMA YILMAZ** | **2012721010 MUHAMMED ALİ ARI** | **SIDKYS-Takımı** |
| **6-Şirket E-ticaret, Sipariş Yönetimi ve Pazarlama Yönetim sistemi** | **2012721005 EDANUR ÇALIŞKAN**  **2012721015 YİĞİT ŞİMŞEK**  **2012729002 BARAN UĞUR ÖZDEMİR**  **2012721032 AYŞE PELİN ULUSOY** | **2012721015 YİĞİT ŞİMŞEK** | **SETPYS-Takımı** |
| **7-Şirket Toplantı Yönetim Sistemi** | **2012721009 NURSENA KÜÇÜKASLAN**  **2012721046 FATİH ALTUN**  **2012729008 SERDAR YÜKSEK**  **2012721025 HELİN AKAR** | **2012721009 NURSENA KÜÇÜKASLAN** | **STOPYS-Takımı** |
| **8-Teknik Servis Planlama ve Yönetim Sistemi** | **2012721022 BUSE CANSU YİĞİT**  **2012721035 YİĞİT LEBLEBİCİER**  **2012721011 UMUT KAHRAMAN**  **2012721016 SEBAHATTİN SERHAT TURGUT** | **2012721022 BUSE CANSU YİĞİT** | **STSPYS-Takımı** |
| **9-Şirket Depo Yönetim Sistemi** | **2012721020 BATUHAN KOCAGÖZ**  **2012721012 SABAHİTTİN ALTAN ASLAN**  **2012721029 ERKAN ÇEVİKGEDEY**  **2012721013 MUSTAFA AKILLI** | **2012721020 BATUHAN KOCAGÖZ** | **SDEYS-Takımı** |

**İÇİNDEKİLER**

[Yazılım Özellikleri 7](#_Toc135302443)

[1 Kullanılan Programlama Dili 7](#_Toc135302444)

[2 Genel bilgiler 7](#_Toc135302445)

[2a Proje Özeti 7](#_Toc135302446)

[2b Risk Planı 7](#_Toc135302447)

[2c Kullanılan toollar 7](#_Toc135302448)

[2d Takım üyeleri Görev Dağılım Tablosu /Sorumluluk Matrisi 8](#_Toc135302449)

[2e Proje Yönetimi ve İş Planı 8](#_Toc135302450)

[2f Work Breakdown Chart /Gantt Chart/Pert Chart 9](#_Toc135302451)

[2g Çalışma Planı Tablosu 11](#_Toc135302452)

[3 Sürümler 11](#_Toc135302453)

[4 Kullanılan Yazılım Mimarisi 11](#_Toc135302454)

[5 Uygulama bileşenleri 11](#_Toc135302455)

[5a Ön Yüz Bileşeni 11](#_Toc135302456)

[5b Arka Yüz Bileşeni 12](#_Toc135302457)

[5c Bileşen Şeması 13](#_Toc135302458)

[5d Çevrimdışı Bileşenler / Komut Dosyaları 14](#_Toc135302459)

[6 Teknoloji seçimi 14](#_Toc135302460)

[6a İlişkisel Veri tabanı Yönetim Sistemi (RDBMS) 14](#_Toc135302461)

[6b Java/C# Program Ortamı 14](#_Toc135302462)

[6c Uygulama Çerçevesi (Framework) 14](#_Toc135302463)

[6d Uygulama/Web Sunucu 15](#_Toc135302464)

[6e Nesne-İlişkisel Haritalama (ORM) Çerçevesi (Entity Framework, Hibernate, JPA) 15](#_Toc135302465)

[6f Veritabanı Sürümü Kontrol Çerçevesi 16](#_Toc135302466)

[6g İstemci tarafı teknolojiler 16](#_Toc135302467)

[6h MVVM Çerçevesi 17](#_Toc135302468)

[6i Derleme Araçları 18](#_Toc135302469)

[6j Geliştirme platformu 18](#_Toc135302470)

[6k Adlandırma Kuralları 19](#_Toc135302471)

[6l Çoklu dil desteği 19](#_Toc135302472)

[6m Teknoloji diagramı 19](#_Toc135302473)

[6n Teknoloji referansı 20](#_Toc135302474)

[7 Güvenlik Araçları 21](#_Toc135302475)

[7a Güvenlik İlkeleri 21](#_Toc135302476)

[7b Ön Yüz Güvenlik Bileşenleri 23](#_Toc135302477)

[7c Arka Yüz Güvenlik Bileşenleri 24](#_Toc135302478)

[8 Güvenlik Mimarisi Yaklaşımı 25](#_Toc135302479)

[9 Proje İzleme ve Kontrol Mekanizmaları 26](#_Toc135302480)

[Sistemdeki Class Tanımları (Class description) 26](#_Toc135302481)

[10 Class Tanımı 26](#_Toc135302482)

[10a Class Genel Özellikleri 26](#_Toc135302483)

[10b Class Özellikleri 26](#_Toc135302484)

[10c Metod Tanımları 26](#_Toc135302485)

[10d Signature/API/Interface/Web Servisleri Tanımları 27](#_Toc135302486)

[Programın Genel Özellikleri- Alt sistemler/Bileşen ve Modüller 27](#_Toc135302487)

[11 Programın Çalıştırılması 27](#_Toc135302488)

[11a Kullanıcı Fonksiyonları 27](#_Toc135302489)

[11b Sunucu Fonksiyonları 28](#_Toc135302490)

[11c Yönetici Fonksiyonları 28](#_Toc135302491)

[12 Menü Bölümü 28](#_Toc135302492)

[13 Rapor Tasarımları 28](#_Toc135302493)

[14 Modüller ve Ekran Tanıtımları 29](#_Toc135302494)

[15 Kullanıcı Girişi 29](#_Toc135302495)

[16 Kullanılan Kısayollar 30](#_Toc135302496)

[17 Programdan Çıkış 30](#_Toc135302497)

[18 Yazılımın Ölçülmesi COCOMO Modeli 31](#_Toc135302498)

[19 Bakım, Sonuç ve Öneriler 33](#_Toc135302499)

[20 Ekler 33](#_Toc135302500)

**Bu Doküman Nasıl Kullanılır**

**Bu belge, belirli bir yazılım mühendisliği projesine uyacak şekilde kopyalanabilen ve düzenlenebilen örnek bir şablon olarak tasarlanmıştır. Bu belgedeki başlıklardan eğer ihtiyacınız yoksa başlığı tamamen siliniz. Doküman Times New Roman 12 punto ile hazırlanmıştır. Şekiller ve Tablolara resim-şekil yazısı eklenmelidir. Ekleme yapıldıktan sonra şekiller dizini ve tablolar dizini Alanı Güncelleştir yapılarak güncellenmelidir. Başlıkların yapısı bozulmadan içindekiler dizini Alanı Güncelleştir yapılarak dokümanın içeriği güncellenir. Raporu hazırlarken dokümanın açıklamaları silinmelidir.** Örnekler sadece bilgi amaçlıdır. Dokümana buradaki örnekler ve açıklamalar konulmayacaktır.

**Ödevin hangi aşamada olduğu her hafta takip edilecektir. Ödev ilan edilen ÖDEV takvimine göre teslim edilecektir.**

**ŞEKİLLER DİZİNİ**

[Şekil 1 Work BreakDown Chart 1 9](#_Toc135302501)

[Şekil 2 Work BreakDown Chart 2 9](#_Toc135302502)

[Şekil 3 Örnek Bileşen Şeması 13](#_Toc135302503)

[Şekil 4 Teknoloji Diyagramı 20](#_Toc135302504)

[Şekil 5 Örnek Menü Tasarımı 28](#_Toc135302505)

[Şekil 6 Örnek Ekran Tasarımı 29](#_Toc135302506)

[Şekil 7 Örnek Giriş Ekranı 29](#_Toc135302507)

[Şekil 8 Örnek Kısayol Tanımı 30](#_Toc135302508)

[Şekil 9 Örnek COCOMO Modeli 32](#_Toc135302509)

**TABLOLAR DİZİNİ**

[Tablo 1 Rol Tanımları 8](#_Toc135302510)

[Tablo 2 İş Planı 8](#_Toc135302511)

[Tablo 3 Gant Chart 10](#_Toc135302512)

[Tablo 4 Pert Tablosu 10](#_Toc135302513)

[Tablo 5 Sürüm Çizelgesi 11](#_Toc135302514)

[Tablo 6 Veritabanı Sunucu 14](#_Toc135302515)

[Tablo 7 Uygulama Sunucu 15](#_Toc135302516)

# Yazılım Özellikleri

## Kullanılan Programlama Dili

**İçerik**

Kullanılan programlama dilinin özellikleri ve neden tercih edildiği anlatılacaktır.

## Genel bilgiler

### Proje Özeti

**İçerik**

Projenin şu ana kadar yapılan işlemleri ile ilgili kısa bir özet yazılacaktır. Gereksinim raporunda ve Tasarım raporunda yapılanlar kısaca açıklanacaktır.

### Risk Planı

**İçerik**

Sistem tasarımı ve önerilen stratejiler ile ilgili riskleri tanımlayınız.

Risk analizi sonuçlarının mimari üzerinde bir etkisi varsa, burada her bir risk analizi çıktısı için mimarideki riski azaltmak için neler yapıldığını açıklayın.

Seçenekleri açıklamak için gerekirse risk azaltmadan önceki mimari ve risk azaltmadan sonraki mimari gibi diyagramları kullanın.

### Kullanılan toollar

**İçerik**

Kullanılan toollar anlatılacaktır.

**Örnek**

**Versiyon Kontrol Tolları**

* Subversion (SVN)
* Concurrent Version System (CVS)
* GIT
* Perforce
* Microsoft Visual Source Safe (VSS)

**Yazılım Tasarımı**

* UML tools: Visual paradigm UML, Smart Draw, Omondo,...
* Standard office tools: OpenOffice Writer, Microsoft Visio, Microsoft Word,...
* Full development environments with modelling capabilities: Eclipse, Sparx Systems Enterprise Architect (Windows), Microsoft Visual Studio, ...

**İzlenebilirlik Toolu**

Hedefler:

* Her gereksinimin kaynağından uygulama ve teste (doğrulama) ayrıştırma yoluyla bağlantıyı sürdürmek.
* Tüm gereksinimlerin karşılanmasını ve yalnızca gerekli olanın geliştirilmesini sağlamak.
* Gereksinimlerin, tasarımın veya diğer yapılandırılmış öğe değişikliklerinin etki değerlendirmelerini yaparken kullanışlıdır.

### Takım üyeleri Görev Dağılım Tablosu /Sorumluluk Matrisi

**İçerik**

Projenizdeki gerçekleştirim kısmındaki takım üyeleri özellikleri ve görev dağılımlarını yapınız.

**Örnek**

Tablo 1 Rol Tanımları

|  |  |  |  |
| --- | --- | --- | --- |
|  | Rol | Kısaltma | Sorumluluk |
| 1. | Analist | AN | Gereksinimleri ortaya çıkaran, belirleyen ve analiz eden bilgi ve deneyim.  Kullanıcı arayüzleri tasarlama ve ergonomik kriterler konusunda bilgi.  Revizyon teknikleri bilgisi ve yazılım geliştirme ve bakımı konusunda deneyim.  Düzenleme teknikleri hakkında bilgi. |
| 2. | Tasarımcı | DES | Yazılım bileşenleri ve mimari tasarım konusunda bilgi ve deneyim.  Revizyon teknikleri bilgisi ve yazılım geliştirme ve bakımı konusunda deneyim.  Düzenleme teknikleri hakkında bilgi.  Entegrasyon ve sistem testlerinin planlanması ve gerçekleştirilmesi konusunda bilgi ve deneyim. |
| 3. | Teknik Sorumlu | TL | Yazılım geliştirme ve bakım konusunda bilgi ve deneyim. |

### Proje Yönetimi ve İş Planı

**İçerik**

Proje yönetimi ve iş planı anlatılacaktır.

**Örnek**

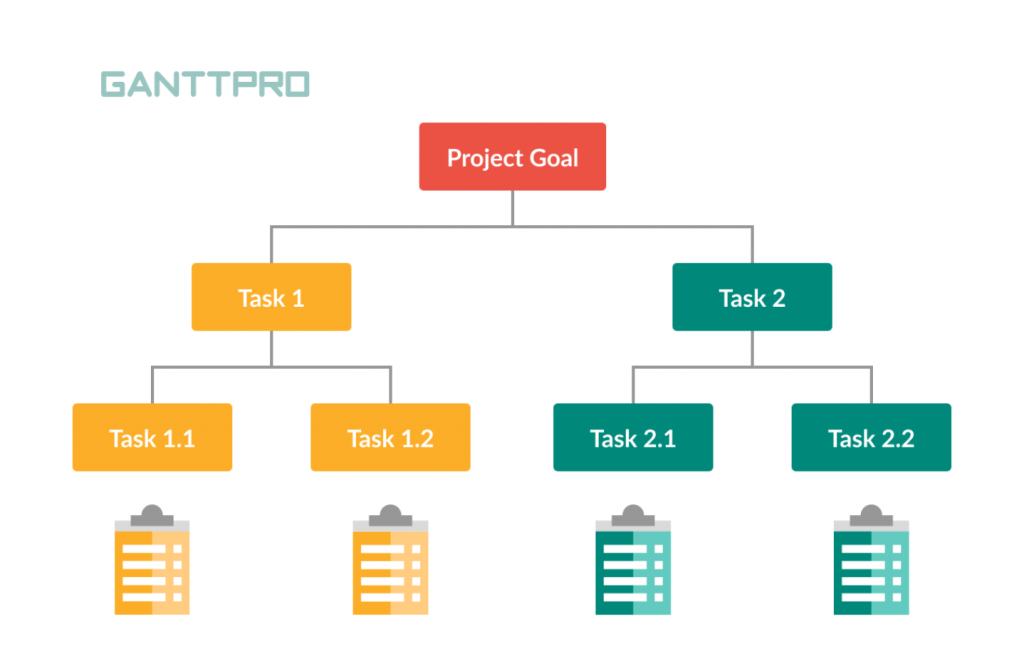
Tablo 2 İş Planı

|  |  |
| --- | --- |
| **İş Listesi** | **Roller** |
| İş Tanımlama | **TL, AN, DES** |
| Gereksinim Tanımlama | **AN, DES** |
| Yazılım Tasarım Dokümanı Hazırlama | **AN, DES** |
| Yazılım Tasarımı | **AN, DES** |
| … | **DES** |
| … | **DES, AN** |
| … | **DES** |
| … | **TL** |

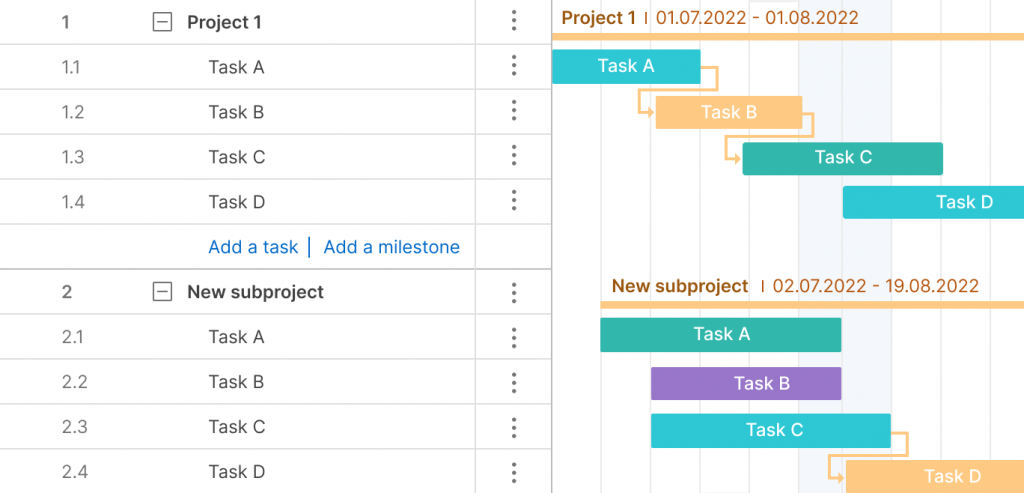
### Work Breakdown Chart /Gantt Chart/Pert Chart

Bir İş Dağılımı (Work Breakdown) Yapısı genellikle ana proje hedefini, belirli departmanların görevlerini bütçe, gerekli kaynaklar ve görevden sorumlu kişiler vb. dahil olmak üzere ayrıntılarla üretmeleri için daha küçük ve yönetilebilir parçalara (iş paketleri) böler. Burada teslim edilebilir, doğrulanabilir herhangi bir şey anlamına gelir ve bir hedef hizmeti yürütmek için benzersiz ürün veya kapasite. Bir iş paketi, ekip üyeleri tarafından belirli bir görevi tamamlamak için kullanılan bir iş kırılım yapısının en alt seviyesindeki bir çıktıdır.

**Örnek**



Şekil 1 Work BreakDown Chart 1



Şekil 2 Work BreakDown Chart 2

Tablo 3 Gant Chart

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Onay | İşlem Adı | İşlem Süresi | Başlama | Bitiş |
| 1 |  | Proje Tanımı  Proje Araştırma ve Hazırlıklar  Proje Kapsamının Belirlenmesi | 2 gün  1 gün | 01.05.2023  04.05.2023 | 03.05.2023  05.05.2023 |
| 2 |  |  |  |  |  |
| … |  |  |  |  |  |

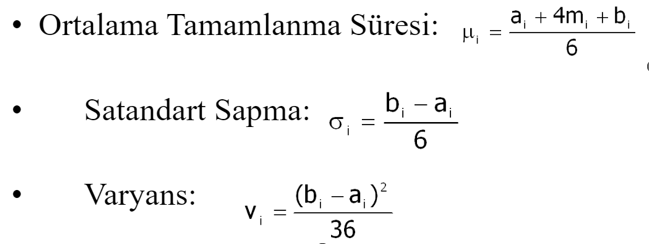
PERT

Pert yaklaşımında faaliyet süreleri üç süre tahminli rassal değişken olarak alınır.

ai: faaliyeti gerçekleştirmek için gereken iyimser süre

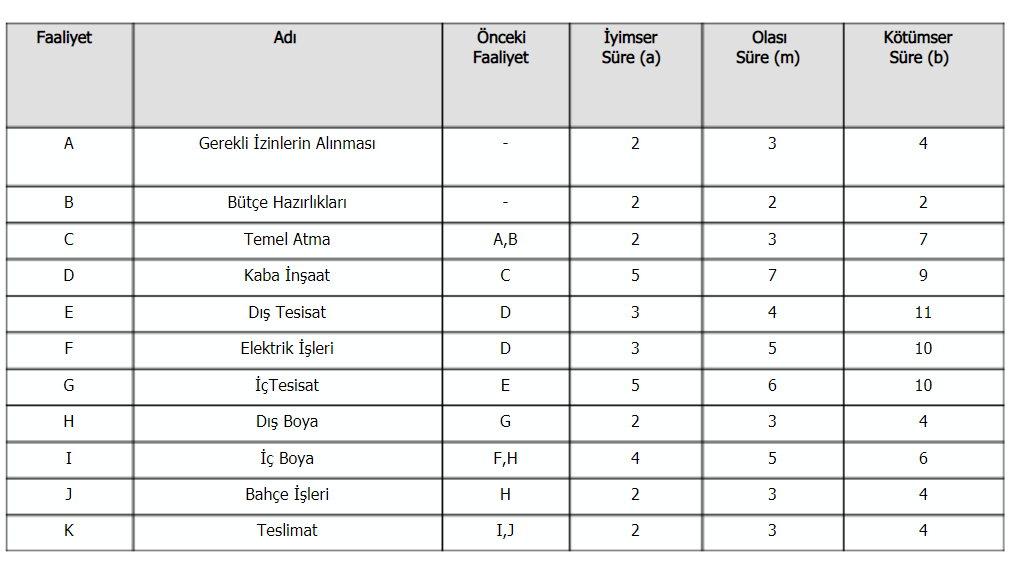
mi: faaliyeti gerçekleştirmek için gereken olası süre

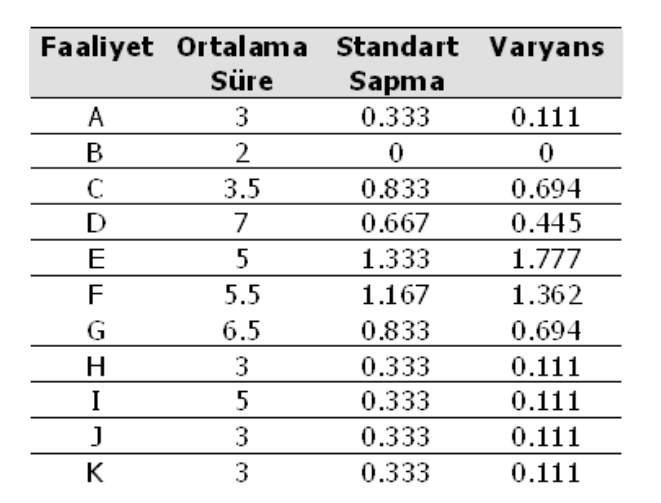
bi: faaliyeti gerçekleştirme k için gereken kötümser düre

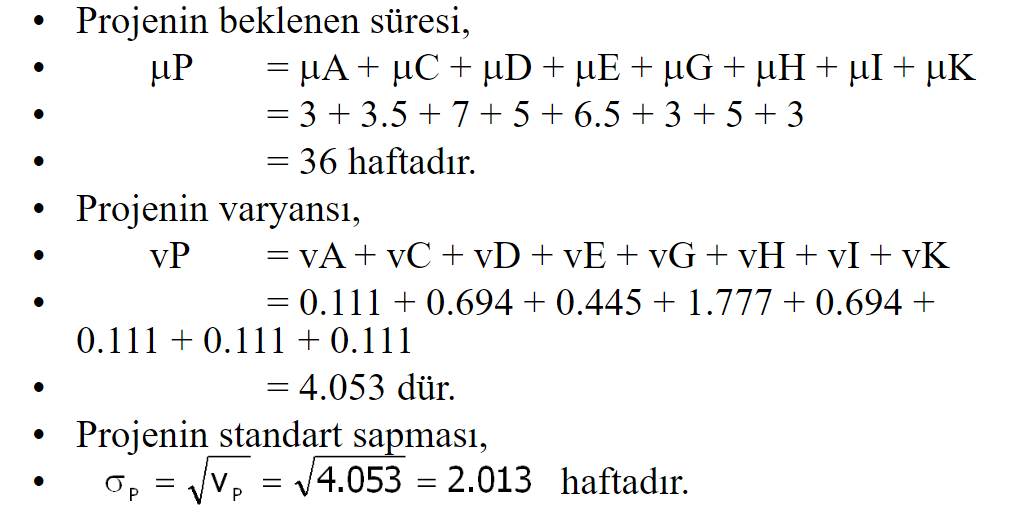


Örnek

Tablo 4 Pert Tablosu







### Çalışma Planı Tablosu

## Sürümler

Tablo 5 Sürüm Çizelgesi

|  |  |  |  |
| --- | --- | --- | --- |
| **Sürüm** | **Sürüm Adı** | **Değişiklik Nedenleri** | **Tarih** |
| *1.0* | *…* | *…* | ../../20…. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Kullanılan Yazılım Mimarisi

**İçerik**

Projede kullanılan mimari yapı tanımlanacaktır.

**Örnek**

Mimari, her katmanın belirli görevlerden sorumlu olması gereken Katmanlı Mimari yaklaşımını izleyecektir. Kabul edilen yaklaşım, mimarinin inşa edilme şekline esneklik sağlar, örn. diğer katmanlar üzerinde minimum etkiye sahip bir katmanda değişiklikler. Küçükten orta tarafa bir uygulama için, bir Bileşen yaklaşımı (çoğunlukla birçok uygulamada kullanılması gereken yeniden kullanılabilir bileşenleri hedefleyen) veya bir Mikro Hizmetler yaklaşımı yerine Katmanlı bir yaklaşımla gitmek daha uygun görünmektedir. Tipik katmanlar Sunum, Hizmet, Kalıcılık, Veri olacaktır.

<https://www.yazilimkodlama.com/c-2/cok-katmanli-yazilim-mimarisi-ornegi-c/>

<https://www.dcs.gla.ac.uk/~quintin/courses/db3/N-Tier.pdf>

<https://dbis.dmi.unibas.ch/publications/2009/db-encyclopedia-multi-tier-architecture/db-encyclopedia-multi-tier-architecture.pdf>

## Uygulama bileşenleri

### Ön Yüz Bileşeni

Projedeki kullanıcı arayüz bileşenleri tanımları yapılacaktır.

**Örnek**

Bu bileşen:

* Sunum katmanını uygular
* Bir Kullanıcı ara yüzünü açığa çıkararak uygulamaya bir giriş noktası sağlar
* MVVM tasarım modeline uygun olarak tasarlanmıştır
* Aşağıdaki mantıksal modülleri içerir:
* Görünüm modülleri: Html şablonları, CSS ve farklı kullanıcı arabirimi kontrollerini temsil eden komut dosyalarından oluşur
* Denetleyici modülleri: arka uçtan alınan verileri ve Görünüm modüllerini birbirine yapıştıran denetleyicilerden oluşur. Denetleyici, görünüm modelini başlatır ve görünümün model değişikliklerine nasıl tepki vermesi gerektiğini tanımlar ve bunun tersi de geçerlidir. Denetleyicinin temel sorumluluklarından biri, ön uç doğrulamaları yapmaktır.
* Ön uç iletişim modülleri: arka uç hizmetleri ile etkileşime izin veren ve Denetleyici modülleri tarafından kullanılabilen bir dizi modül.
* Güvenlik modülleri: görevi aşağıdaki gibi ön uç güvenlik özelliklerini uygulamak olan farklı modüllerden oluşur:
  + Siteler arası komut dosyasını (XSS) önleme
  + Siteler arası istek sahteciliğini önleme (CSRF veya XSRF)
  + Kullanıcı girdisini temizleme (komut dosyası yerleştirmeyi önlemek için)

### Arka Yüz Bileşeni

**İçerik**

Projedeki kullanılan arka yüz bileşenleri tanımları yapılacaktır.

**Örnek**

Bu bileşen:

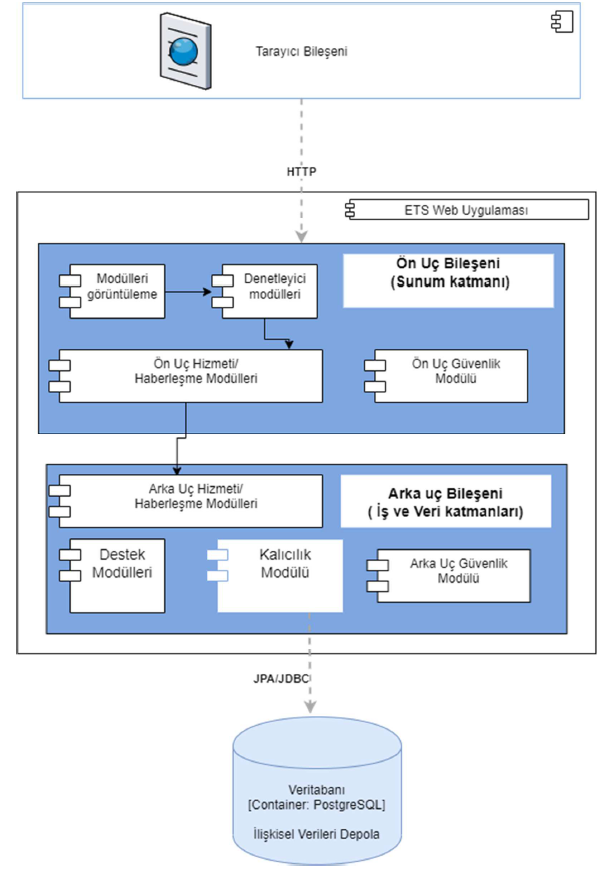
* İş ve Kalıcılık katmanlarını uygular
* Ön uç bileşeninden kullanılabilen bir RESTFul Web Hizmetleri API'sini ortaya çıkarır
* RDBMS'de depolanan verilere özel erişime sahiptir
* Aşağıdaki mantıksal modülleri içerir:
  + Arka uç iletişim modülleri: Ön uç bileşeninden çağrılabilen bir dizi RESTful Web hizmeti uç noktası ve işleminden oluşur
  + Destek modülleri: Gereksinimlere göre yararlı destekleme işlemleri, doğrulamalar, veri değerlendirmeleri / hesaplamaları uygulayan bir dizi modül o Kalıcılık modülü: RDBMS ile verilerin depolanması ve geri alınması için soyut bir şekilde arabirim oluşturmaktan sorumlu bir dizi modül. Bu modül, veri bütünlüğünü ve RDBMS ile birlikte çalışabilirliği sağlar Yazılım Tasarım Dökümanı.docx
  + Güvenlik modülleri: görevi aşağıdakiler gibi arka uç güvenlik özelliklerini uygulamak olan farklı modüllerden oluşur:
    - Kimliği doğrulanmamış kullanıcıların sisteme erişimini önleme
    - Kaynaklara yetkisiz erişimi önleme
    - SQL yerleştirme ve Verilere maruz kalma gibi güvenlik tehditlerini yönetme

### Bileşen Şeması

**İçerik**

Projenizin bileşen şemasını çiziniz.

**Örnek**



Şekil 3 Örnek Bileşen Şeması

### Çevrimdışı Bileşenler / Komut Dosyaları

**İçerik**

Komut dosyaları veya projede kullanılan çevrimdışı bileşenleri belirtiniz.

## Teknoloji seçimi

### İlişkisel Veri tabanı Yönetim Sistemi (RDBMS)

**İçerik**

Bir RDBMS, İlişkisel verilerin tüm depolanması ve geri alınması için tercih edilen sistemdir.

**Örnek**

Kullanılan Veritabanı Sürümü

Veritabanının Çalıştığı İşletim Sistemi:

Önerilen İşlemci Hızı ve Sayısı:

Önerilen RAM Kapasitesi:

Önerilen Veritabanı Dosya Sistemi Sürücü Kapasitesi:

Tablo 6 Veritabanı Sunucu

|  |  |  |  |
| --- | --- | --- | --- |
|  | Development | Test | Live |
| Server |  |  |  |
| Physical / Virtual |  |  |  |
| Shared / Dedicated |  |  |  |
| CPU cores |  |  |  |
| Memory |  |  |  |
| OS |  |  |  |
| Instance |  |  |  |
| Database version |  |  |  |
| Dependencies | e.g. Java, packages |  |  |

### Java/C# Program Ortamı

**İçerik**

Java veya C# program ortamını tanıtınız. Kullanılan IDE nin özelliklerini kısaca yazınız.

### Uygulama Çerçevesi (Framework)

**İçerik**

Kullanılan .NET Frameworkü, Spring Framworkü, Spring Boot gibi frameworklerin sürümü ve özellikleri anlatılacaktır.

**Örnek**

Spring Framework, modern Java tabanlı kurumsal uygulamalar için her tür dağıtım platformunda kapsamlı bir programlama ve yapılandırma modeli sağlar.

Spring Boot, Java geliştiricilerine otomatik olarak yapılandırılabilen, üretim düzeyinde bir Spring uygulamasıyla başlamaları için bir platform sağlayan açık kaynaklı bir mikro çerçevedir.

Özellikler şunları içerir:

Bağımsız spring uygulamaları oluşturma

• Undertow'u doğrudan gömülü yapma (WAR dosyalarını dağıtmaya gerek yoktur)

• Spring ve 3. taraf kitaplıklarını mümkün olduğunda otomatik olarak yapılandırma

• Metrikler, sağlık kontrolleri ve harici konfigürasyon gibi üretime hazır özellikler sağlama

• Kesinlikle kod üretilmez ve XML yapılandırması gerekmez

### Uygulama/Web Sunucu

**İçerik**

Projede kullanılan örnek uygulama web sunucusu ve uygulaması hakkında bilgi verilecektir.

**Örnek**

Internet Information Services(IIS) esnek yapılı ve genel kullanım için Microsoft tarafından Windows sistemlerde çalışmak üzere geliştirilmiş bir web sunucudur. Internet Information Services özellik açısından son derece zengindir. IIS özellikle [ASP.NET](http://asp.net/) ile yazılmış web uygulamalarının ve internet sitelerinin barındırılmasında tercih edilmektedir. Ayrıca FTP sunucusu, WCF servislerinin barındırılması ve PHP gibi diğer platformlarda geliştirilen web uygulamalarının barındırılmasında da tercih edilmektedir.

Tablo 7 Uygulama Sunucu

|  |  |  |  |
| --- | --- | --- | --- |
|  | Development | Test | Live |
| Servers |  |  |  |
| Physical / Virtual |  |  |  |
| Shared / Dedicated |  |  |  |
| CPU cores |  |  |  |
| Memory |  |  |  |
| OS |  |  |  |
| Software and versions |  |  |  |
| Dependencies | e.g. SMTP, Cosign |  |  |

### Nesne-İlişkisel Haritalama (ORM) Çerçevesi (Entity Framework, Hibernate, JPA)

**İçerik**

Projedeki ORM yapısı anlatılacaktır**.**

**Örnek 1**

Seçilen ORM, Spring Data JPA'dır.

Daha büyük Spring Data ailesinin bir parçası olan Spring Data JPA, JPA tabanlı depoların kolayca uygulanmasını kolaylaştırır. Bu modül, JPA tabanlı veri erişim katmanları için gelişmiş destekle ilgilenir. Veri erişim teknolojilerini kullanan Spring destekli uygulamalar oluşturmayı kolaylaştırır.

**Örnek 2**

Seçilen ORM, Entity Frameworktür.

Entity Framework(EF) Ado.Net altyapısını kullanmaktadır. İçerisinde UnitOfWork design pattern’nini barındırmaktadır.  UnitOfWork yapılan her değişkliğin anlık olarak veritabanına yansıması yerine, bu işlemlerin toplu halde tek bir kanal üzerinden gerçekleşmesini sağlar.

….

### Veritabanı Sürümü Kontrol Çerçevesi

Oracle/SQL Server/MySQL…. vb ürününün bulunacağı Windows makinesinde çalışan TCP/IP bir gerekliliktir. İşletim sistemi olarak ………………..…… kullanılmıştır.

Veritabanı Yedekleme ve Kurtarma Senaryosu:

Veritabanı Performans Ölçüm Yöntemleri:

Veritabanı Performans Çözüm Yöntemleri:

Veritabanı Erişim ve Güvenlik Yöntemleri:

Örnek

Seçilen DB SCM Liquibase'dir.

Liquibase, veritabanı şema değişikliklerini izlemek, yönetmek ve uygulamak için açık kaynaklı, veritabanından bağımsız bir kitaplıktır. Scriptler farklı formatlarda (XML, JSON, SQL) tanımlanabilir ve Java, Maven, Ant ile çalıştırılabilir.

https://www.liquibase.com/

https://docs.liquibase.com/start/tutorials/mssql.html

Solarwinds, CA HARVEST SCM 13.0 vb Database Server Configuration Monitor kullanılabilir.

<https://www.solarwinds.com/database-performance-monitor>

### İstemci tarafı teknolojiler

**İçerik**

Projede kullanılan istemci taraflı mimariler anlatılacaktır

**Örnek**

Ön Uç Kullanıcı Arayüzü için kullanılan seçili istemci tarafı teknolojileri HTML5, Javascript ve CSS'dir.

HTML5, World Wide Web'de içerik yapılandırmak ve sunmak için kullanılan bir biçimlendirme dilidir. HTML5, bir World Wide Web Consortium (W3C) önerisi olan HTML'nin beşinci ve son ana sürümüdür. Mevcut spesifikasyon HTML Yaşam Standardı olarak bilinir ve başlıca tarayıcı satıcıları (Apple, Google, Mozilla ve Microsoft), Web Köprü Metni Uygulama Teknolojisi

Çalışma Grubu (WHATWG) konsorsiyumu tarafından sürdürülür.

JavaScript, World Wide Web'in temel teknolojilerinden biridir. JavaScript, etkileşimli web sayfalarını etkinleştirir ve web uygulamalarının önemli bir parçasıdır.

CSS3, Basamaklı Stil Sayfaları dilinin en son evrimidir ve CSS2'yi genişletmeyi amaçlamaktadır.

Yuvarlatılmış köşeler, gölgeler, gradyanlar, geçişler veya animasyonlar gibi birçok yeni özellik ve eklemenin yanı sıra çoklu sütunlar, esnek kutu veya ızgara düzenleri gibi yeni düzenler getirmektedir.

**Örnek**

Blazor, [.NET](https://learn.microsoft.com/tr-tr/dotnet/standard/tour) ile etkileşimli istemci tarafı web kullanıcı arabirimi oluşturmaya yönelik bir çerçevedir:

* [JavaScript](https://www.javascript.com/) yerine [C#](https://learn.microsoft.com/tr-tr/dotnet/csharp/) kullanarak zengin etkileşimli URI'ler oluşturun.
* .NET'te yazılmış sunucu tarafı ve istemci tarafı uygulama mantığını paylaşın.
* Mobil tarayıcılar da dahil olmak üzere geniş tarayıcı desteği için kullanıcı arabirimini HTML ve CSS olarak işleyin.
* [Docker](https://learn.microsoft.com/tr-tr/dotnet/standard/microservices-architecture/container-docker-introduction/index) gibi modern barındırma platformlarıyla tümleştirme.
* .NET ve Blazor ile karma masaüstü ve mobil uygulamalar oluşturun.

İstemci tarafı web geliştirmesi için .NET kullanmak aşağıdaki avantajları sunar:

* JavaScript yerine C# dilinde kod yazma.
* [.NET kitaplıklarının](https://learn.microsoft.com/tr-tr/dotnet/standard/class-libraries) mevcut .NET ekosistemini kullanma.
* Uygulama mantığını sunucu ve istemci arasında paylaşma.
* .NET'in performansı, güvenilirliği ve güvenliğinden yararlanın.
* [Visual Studio](https://visualstudio.microsoft.com/) veya [Visual Studio Code](https://code.visualstudio.com/) gibi bir geliştirme ortamıyla Windows, Linux veya macOS'ta üretkenliğinizi koruyun.
* Kararlı, özellik bakımından zengin ve kullanımı kolay ortak bir dil, çerçeve ve araç kümesi üzerinde derleme yapın.

### MVVM Çerçevesi

**İçerik**

Kullanılan MVVM tanımlanacaktır.

**Örnek**

Mimari bir Model-görünüş-görünüm modeli (MVVM) mimari modelini izleyecektir. MVVM, grafik kullanıcı ara yüzünün geliştirilmesinin- bir biçimlendirme dili veya GUI kodu yoluyla- iş mantığının veya arka uç mantığının (veri modeli) geliştirilmesinden ayrılmasını kolaylaştırır. MVVM'nin görünüm modeli bir değer dönüştürücüsüdür, yani görünüm modeli, modeldeki veri nesnelerini nesnelerin kolayca yönetilebileceği ve sunulabileceği bir şekilde ortaya çıkarmaktan (dönüştürmek) sorumludur. Bu açıdan, görünüm modeli, görünümden daha modeldir ve görünümün görüntüleme mantığının tamamını olmasa da çoğunu idare eder. En tipik MVVM uygulamaları Angular'dır.

### Derleme Araçları

**İçerik**

Projede kullanılan derleme aracı anlatılacaktır.

**Örnek**

Java uygulaması için seçilen derleme aracı Maven'dir.

Apache Maven, bir yazılım proje yönetimi ve anlama aracıdır. Bir proje nesne modeli (POM) kavramına dayalı olarak Maven, bir projenin yapısını, raporlamasını ve dokümantasyonunu merkezi bir bilgi parçasından yönetebilir.

Ön uç için seçilen derleme aracı Webpack'tir.

Webpack, modern JavaScript uygulamaları için statik bir modül paketleyicidir. Webpack uygulamanızı işlerken dahili olarak projenizin ihtiyaç duyduğu her modülü eşleştiren ve bir veya daha fazla paket oluşturan bir bağımlılık grafiği oluşturur.

**Örnek**

Uygulama Azure DevOps ile yapılmıştır.

Uygulamaların planlanması, geliştirilmesi, teslimi ve işletilmesi sırasında DevOps uygulamalarını hayata geçirmek için Azure’daki uçtan uca çözümlerden yararlanılmıştır. Sürekli yazılım teslimi sağlamak ve müşterilere daha yüksek değer sunmak için DevOps teknolojilerini, kültürünü ve süreçlerini doğru şekilde bir araya getiren Azure projede kullanılmıştır.

### Geliştirme platformu

**İçerik**

Geliştirme platformunuzun özelliklerini yazınız.

**Örnek**

Kaliteli ürünler, uygulamalar ve hizmetler sunmak için ekibimizi, geliştirme/test ve üretim öncesi ortamlarını hızlı bir şekilde sağlama konusunda güçlendirmek için Azure DevTest Labs, GitHub Codespaces ve Windows Sanal Masaüstü kullanılmıştır. Azure DevTest Labs, GitHub Codespaces ve Windows Sanal Masaüstü gibi amaca yönelik, yönetilen geliştirici hizmetlerini kullanarak idare, maliyet denetimleri veya güvenlik konusunda ödün vermeden geliştirme/test ortamlarını, kiracıları ve abonelikleri kolayca yönetip ve iyileştirme imkanı elde edilmiştir.

**Örnek**

Geliştirmeyi önyüklemek için kullanılan seçilen geliştirme platformu JHipster'dır, ancak sürekli geliştirme için gerekli değildir.

JHipster, Spring Boot + Angular / React / Vue Web uygulamalarını ve Spring mikro hizmetlerini başlatmak, oluşturmak, geliştirmek ve dağıtmak için kullanılabilen bir geliştirme platformudur.

Sanal sunucular

Geliştirme araçları

DevOps araç zinciri

Gerektiğinde mesajlaşma hizmetleriyle entegrasyon, örn. Posta sunucusuyla

Ağ üzerinden kullanılabilirlik

İmzalı sertifikalar

### Adlandırma Kuralları

**İçerik**

Projede kullanılan adlandırma kuralları belirtilecektir.

**Örnek**

Bu Mimari bağlamında aşağıdaki adlandırma kuralları izlenecektir.

• Veri tabanı (RDBMS) nesnelerinin adlandırılması için İngilizce dili kullanılacaktır

• İngilizce dili, Uygulama Bileşenlerinin, Modüllerinin, Java/C# sınıflarının, örn.

• Otomatik adlandırma önerilmemektedir, yani nesnelerin İngilizce dilinde bazı anlamlı isimleri olmalıdır

• Hem Veri tabanı nesnelerinin adlandırılmasında hem de Uygulama bileşenlerinde ön eklerin kullanılması önerilir

### Çoklu dil desteği

**İçerik**

Projede çoklu dil desteği var mı varsa hangileri olduğu belirtilecektir.

**Örnek**

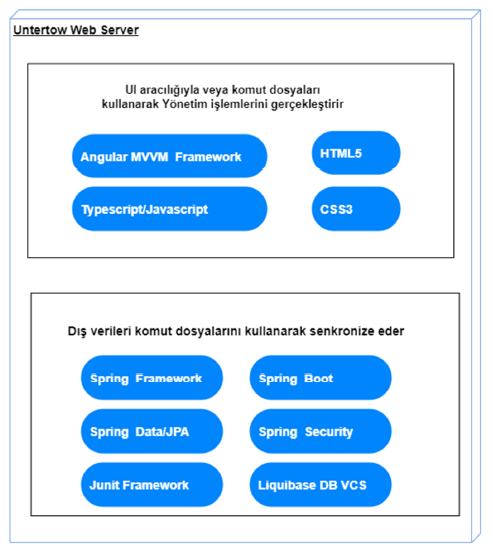
Uygulama kullanıcı ara yüzü İngilizce ve Türkçe olmak üzere 2 dilde mevcut olacaktır. Ek diller için destek sağlanacaktır.

### Teknoloji diagramı

**İçerik**

Projede kullanılacak en önemli teknolojiler aşağıdaki diyagramda gösterilmektedir.

**Örnek**



Şekil 4 Teknoloji Diyagramı

### Teknoloji referansı

**İçerik**

Proje yaparken kullandığınız web adresi, online kitap vb kaynakları yazınız.

**Örnek**

<https://learn.microsoft.com/tr-tr/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-7.0&tabs=visual-studio>

<https://azure.microsoft.com/tr-tr/free/>

<https://openjdk.java.net/>

<https://spring.io/projects/spring-boot>

<https://spring.io/projects/spring-data>

<https://spring.io/projects/spring-framework>

<http://undertow.io/>

<https://angular.io/>

<https://www.liquibase.org/>

<https://junit.org/junit5/>

<https://maven.apache.org/>

<https://webpack.js.org/>

<https://www.jhipster.tech/>

<https://www.btkakademi.gov.tr/>

<https://www.turcademy.com/tr>

## Güvenlik Araçları

### Güvenlik İlkeleri

Güvenlik mimarisi, bir uygulamanın normal mimarisinden ayrı değildir en basit sistemlerin bile mimarisinin doğuştan gelen bir yönüdür.

Bilgi güvenliği genellikle aşağıdaki sütunlara dayanır:

Gizlilik - yalnızca kullanıcıya izin verilen verilere erişime izin verme

Bütünlük - verilerin yetkisiz kullanıcılar tarafından tahrif edilmemesini veya değiştirilmemesini sağlama

Kullanılabilirlik - sistemlerin ve verilerin yetkili kullanıcılara ihtiyaç duyduklarında erişebilmelerini sağlama

Aşağıda, güvenlik özelliklerini şekillendirmek

Ekleme yüzey alanını en aza indirme: Bir uygulamaya eklenen her özellik, genel uygulamaya belirli bir miktar risk ekler. Güvenli geliştirmenin amacı, saldırı yüzey alanını azaltarak genel riski azaltmaktır. Örneğin, bir web uygulaması, bir arama işlevi ile çevrimiçi yardım uygular. Arama işlevi, SQL enjeksiyon saldırılarına karşı savunmasız olabilir. Yardım özelliği yetkili kullanıcılarla sınırlıysa, saldırı olasılığı azalır.

En az ayrıcalık: En az ayrıcalık ilkesi, hesapların iş süreçlerini gerçekleştirmek için gereken en az ayrıcalığa sahip olmasını önerir. Bu, kullanıcı haklarını, CPU sınırları gibi kaynak izinlerini, belleği, ağı ve dosya sistemi izinlerini kapsar.

Örneğin, bir ara yazılım sunucusu yalnızca ağa erişim, bir veri tabanı tablosuna okuma erişimi ve bir günlüğe yazma yeteneği gerektiriyorsa, bu, verilmesi gereken tüm izinleri açıklar.

Gizliliğe dayalı güvenlikten kaçınma: Belirsizlik yoluyla güvenlik, zayıf bir güvenlik kontrolüdür ve tek kontrol olduğunda neredeyse her zaman başarısız olur. Bu, sır saklamanın kötü bir fikir olduğu anlamına gelmez, basitçe anahtar sistemlerin güvenliğinin ayrıntıları gizli tutmaya bağlı olmaması gerektiği anlamına gelir. Örneğin, bir uygulamanın güvenliği, gizli tutulan kaynak kod bilgisine dayanmamalıdır.

Güvenliği basit tutma: Saldırı yüzey alanı ve basitlik el ele gider. Bazı yazılım mühendisliği hevesleri, aksi takdirde görece basit ve basit olan koda aşırı karmaşık yaklaşımları tercih eder. Geliştiriciler, daha basit bir yaklaşımın daha hızlı ve daha basit olacağı durumlarda, çift negatiflerin ve karmaşık mimarilerin kullanımından kaçınmalıdır.

En zayıf bağlantı: Bu güvenlik ilkesi, yazılımınızın hacker girişimlerine karşı dayanıklılığının, kod, hizmet veya ara yüz gibi en zayıf bileşenlerinin korunmasına büyük ölçüde bağlı olacağını belirtir.

Güvenlik Riskleri:

Enjeksiyon: SQL, NoSQL, OS ve LDAP enjeksiyonu gibi enjeksiyon kusurları, güvenilmeyen veriler bir komut veya sorgunun parçası olarak bir yorumlayıcıya gönderildiğinde ortaya çıkar. Saldırganın düşmanca verileri, yorumlayıcıyı istenmeyen komutları çalıştırması veya uygun yetkilendirme olmadan verilere erişmesi için kandırabilir.

Bozuk Kimlik Doğrulama: Kimlik doğrulama ve oturum yönetimiyle ilgili uygulama işlevleri genellikle yanlış uygulanır ve saldırganların parolaları, anahtarları veya oturum belirteçlerini tehlikeye atmasına veya diğer kullanıcıların kimliklerini geçici veya kalıcı olarak üstlenmek için diğer uygulama kusurlarından yararlanmasına olanak tanır.

Hassas Verilerin İfşası: Birçok web uygulaması ve API, finans, sağlık ve PII gibi hassas verileri düzgün bir şekilde korumaz. Saldırganlar, kredi kartı dolandırıcılığı, kimlik hırsızlığı veya diğer suçları işlemek için bu tür zayıf korunan verileri çalabilir veya değiştirebilir. Beklemede veya aktarım sırasında şifreleme gibi ekstra koruma olmadan hassas veriler tehlikeye atılabilir ve tarayıcıyla değiştirildiğinde özel önlemler gerektirir.

XML Harici Varlıklar (XXE): Birçok eski veya kötü yapılandırılmış XML işlemci, XML belgelerindeki harici varlık referanslarını değerlendirir. Harici varlıklar, dosya URI işleyicisini, dahili dosya paylaşımlarını, dahili port taramayı, uzaktan kod yürütmeyi ve hizmet reddi saldırılarını kullanarak dahili dosyaları ifşa etmek için kullanılabilir.

Bozuk Erişim Kontrolü: Kimliği doğrulanmış kullanıcıların ne yapmasına izin verildiğine ilişkin kısıtlamalar genellikle düzgün bir şekilde uygulanmaz. Saldırganlar, diğer kullanıcıların hesaplarına erişmek, hassas dosyaları görüntülemek, diğer kullanıcıların verilerini değiştirmek, erişim haklarını değiştirmek gibi yetkisiz işlevlere ve / veya verilere erişmek için bu kusurları kullanabilir.

Yanlış Güvenlik Yapılandırması: Yanlış güvenlik yapılandırması en yaygın görülen sorundur. Bu genellikle güvenli olmayan varsayılan yapılandırmaların, eksik veya geçici yapılandırmaların, açık bulut depolamanın, yanlış yapılandırılmış http üstbilgilerinin ve hassas bilgiler içeren ayrıntılı hata mesajlarının bir sonucudur. Tüm işletim sistemlerinin, çerçevelerin, kitaplıkların ve uygulamaların güvenli bir şekilde yapılandırılması değil, aynı zamanda zamanında yamalanması / yükseltilmesi gerekir.

Siteler Arası Komut Dosyası XSS: XSS kusurları, bir uygulama yeni bir web sayfasında uygun doğrulama veya çıkış olmadan güvenilmeyen veriler içerdiğinde veya HTML veya JavaScript oluşturabilen bir tarayıcı API'si kullanarak kullanıcı tarafından sağlanan verilerle mevcut bir web sayfasını güncellediğinde ortaya çıkar. XSS, saldırganların kurbanın tarayıcısında kullanıcı oturumlarını ele geçirebilecek, web sitelerini tahrif edebilecek veya kullanıcıyı kötü amaçlı sitelere yeniden yönlendirebilecek komut dosyaları yürütmesine olanak tanır.

Güvensiz Serileştirme: Güvensiz serileştirme genellikle uzaktan kod yürütülmesine yol açar. Seri durumdan çıkarma kusurları uzaktan kod yürütülmesine yol açmasa bile, bunlar yeniden oynatma saldırıları, enjeksiyon saldırıları ve ayrıcalık yükseltme saldırıları dahil olmak üzere saldırıları gerçekleştirmek için kullanılabilir.

Bilinen Güvenlik Açıklarına Sahip Bileşenleri Kullanma: Kitaplıklar, çerçeveler ve diğer yazılım modülleri gibi bileşenler, uygulama ile aynı ayrıcalıklarla çalışır. Savunmasız bir bileşenden yararlanılırsa, bu tür bir saldırı ciddi veri kaybını veya sunucunun ele geçirilmesini kolaylaştırabilir. Güvenlik açıkları olduğu bilinen bileşenleri kullanan uygulamalar ve API'ler, uygulama savunmalarını zayıflatabilir ve çeşitli saldırılara ve etkilere olanak sağlayabilir.

Yetersiz Günlük Kaydı ve İzleme: Olay yanıtıyla eksik veya etkisiz entegrasyonla birlikte yetersiz günlük kaydı ve izleme, saldırganların sistemlere daha fazla saldırmasına, kalıcılığı sürdürmesine, daha fazla sisteme dönmesine ve verileri kurcalamasına, çıkarmasına veya yok etmesine olanak tanır. Çoğu ihlal araştırması, bir ihlali tespit etme süresinin 200 günden fazla olduğunu ve genellikle dahili süreçler veya izleme yerine harici taraflarca tespit edildiğini göstermektedir.

### Ön Yüz Güvenlik Bileşenleri

Angular (Açısal), yaygın web uygulaması güvenlik açıklarına ve siteler arası komut dosyası çalıştırma saldırıları gibi saldırılara karşı yerleşik korumalara sahiptir.

Siteler arası komut dosyasını (XSS) önleme

Angular, siteler arası komut dosyası çalıştırma güvenlik modeli sağlar: XSS hatalarını sistematik olarak engellemek için Angular, tüm değerleri varsayılan olarak güvenilmez olarak değerlendirir. Özellik, öznitelik, stil, sınıf bağlama veya enterpolasyon yoluyla bir şablondan DOM'a bir değer eklendiğinde, Angular güvenilir olmayan değerleri temizler ve bunlardan kaçar.

Güvenilir olmayan değerleri DomSanitizer.sanitize yöntemi ve uygun SecurityContext ile sterilize etmek için yerleşik Açısal temizleme işlevleri kullanılabilir. Bu işlev aynı zamanda güvenilir olarak işaretlenmiş değerleri de kabul eder.

HTTP düzeyinde güvenlik açıkları

Angular (Açısal), iki yaygın HTTP güvenlik açığını, siteler arası istek sahteciliğini (CSRF veya XSRF) ve siteler arası komut dosyası dahil etmeyi (XSSI) önlemeye yardımcı olmak için yerleşik desteğe sahiptir. Bunların her ikisi de öncelikle sunucu tarafında azaltılmalıdır, ancak Angular (Açısal), istemci tarafında entegrasyonu kolaylaştırmak için yardımcılar sağlar.

### Arka Yüz Güvenlik Bileşenleri

Spring Security, güçlü ve son derece özelleştirilebilir bir kimlik doğrulama ve erişim denetimi çerçevesidir. Yay tabanlı uygulamaların güvenliğini sağlamak için fiili standarttır. Spring Security, Java uygulamalarına hem kimlik doğrulama hem de yetkilendirme sağlamaya odaklanan bir çerçevedir. Tüm Spring projeleri gibi, Spring Güvenlik'in gerçek gücü, özel gereksinimleri karşılamak için ne kadar kolay genişletilebileceğinde bulunur

Özellikler şunları içerir:

Kimlik Doğrulama: Kimlik doğrulama, belirli bir kaynağa erişmeye çalışanların kimliğini nasıl doğruladığımızdır. Kullanıcıların kimliğini doğrulamanın yaygın bir yolu, kullanıcının bir kullanıcı adı ve parola girmesini istemektir. Kimlik doğrulama yapıldığında kimliği biliriz ve yetkilendirme yapabiliriz.

Yetkilendirme veya Erişim Kontrolü: Yetkilendirme, bir kullanıcının yalnızca izne sahip olduğu kaynaklara erişebilmesini sağlamakla ilgilidir.

* Oturum sabitleme, tıklama korsanlığı, siteler arası istek sahteciliği vb. saldırılara karşı koruma
* Servlet API entegrasyonu

Doğrulama

Spring Güvenlik, kimlik doğrulama için kapsamlı destek sağlar. Bir kullanıcının kimliğini doğrulamanın en yaygın yollarından biri, bir kullanıcı adı ve parolayı doğrulamaktır. Bu nedenle Spring Güvenlik, bir kullanıcı adı ve parola ile kimlik doğrulaması için kapsamlı destek sağlar.

Aşağıdaki yerleşik mekanizmalar mevcuttur:

Form Girişi

Temel Kimlik Doğrulama

Özet Kimlik Doğrulaması

Depolama Mekanizmaları

Bir kullanıcı adı ve parolayı okumak için desteklenen mekanizmaların her biri, desteklenen herhangi bir depolama mekanizmasından yararlanabilir:

Bellek İçi Kimlik Doğrulamalı Basit Depolama

JDBC Kimlik Doğrulamalı İlişkisel Veritabanları

UserDetailsService ile özel veri depoları

LDAP Kimlik Doğrulamalı LDAP depolama

Spring güvenliği ayrıca Kimlik Doğrulama için bir dizi ek özelliği destekler:

Çoklu Parola şifreleme mekanizmaları

Beni hatırla / kalıcı oturum açma kimlik doğrulaması, yani oturumlar arasında bir müdürün kimliğini hatırlayabilme

Yetki

Spring Güvenlik içindeki gelişmiş yetkilendirme yetenekleri, popülerliğinin en zorlayıcı nedenlerinden birini temsil etmektedir.

Yetkililer, Spring güvenliğinde yetkilendirmeyi yöneten ve iş Rollerine karşılık gelen ana varlıktır. Spring güvenlik, bir dizi gelişmiş yetkilendirme özelliğini destekler:

İfade Tabanlı Erişim Kontrolü: Spring EL ifadelerini bir yetkilendirme mekanizması olarak kullanma yeteneği

Yöntem düzeyinde güvenlik: yalnızca hizmet düzeyinde değil, aynı zamanda yöntem düzeyinde de çok ayrıntılı erişim kontrolleri tanımlama yeteneği

İstismarlara karşı koruma

Spring Security, yaygın istismarlara karşı koruma sağlar. Mümkün olduğunda, koruma varsayılan olarak etkindir. Daha spesifik olarak, Spring Güvenlik şunlara karşı korur:

Spring, Siteler Arası İstek Sahteciliği (CSRF) saldırılarına karşı koruma sağlamak için kapsamlı destek sağlar

Spring Güvenlik, web uygulamalarının güvenliğini artırmak için kullanılabilen birçok HTTP / HTTPS yanıt başlığı için açık destek sağlar. Gerekirse, Spring Güvenlik özel başlıklar sağlayacak şekilde de yapılandırılabilir.

## Güvenlik Mimarisi Yaklaşımı

* Elimizdeki hususlar, endişeler ve araçlar göz önüne alındığında, bu Mimari, ETS Web uygulamasının:
* güvenli kimlik doğrulama özelliklerini destekler
* yetkilendirme özelliklerini destekler
* hem ön uç hem de arka uç bileşenlere yönelik yaygın istismarlara karşı güçlü karşı önlemlere sahiptir

Bu açıdan aşağıdaki Mimari kararlar alındı:

1. Ön uç bileşenini sabitlemek için yukarıda açıklanan açısal çerçeve araçları ve mekanizmaları kullanılacaktır

2. Spring Güvenlik çerçeve araçları ve mekanizmaları, arka uç bileşeninin güvenliğini sağlamak için kullanılacaktır. Daha spesifik olarak:

ETS uygulaması için seçilen kimlik doğrulama yöntemi şudur: ETS İlişkisel Veritabanına JDBC Kimlik Doğrulaması ile Form Oturum Açma

Seçilen yetkilendirme yöntemi: Verilen Yetkiler (Roller) tanımlanacak ve Uygulama düzeyinde, Hizmet düzeyinde ve Yöntem düzeyinde belirli Erişim Kontrol noktaları eklenecektir

· Yaygın saldırılara karşı koruma sağlamak için yerleşik Spring güvenlik bileşenleri yapılandırmada etkinleştirilecektir

3. 3. OWASP araçları, L1 yönergeleri ve karşı önlemler uygulama bağlamında kullanılacaktır:

* Güvenlik kusurları olan üçüncü taraf kitaplıkları belirlenecek ve kaldırılacaktır
* Yeterli izleme ve kayıt tutma
* Yaygın kod kötü uygulamaları tanımlanır ve düzeltilir

## Proje İzleme ve Kontrol Mekanizmaları

# Sistemdeki Class Tanımları (Class description)

## Class Tanımı

### Class Genel Özellikleri

Class Adı:

Amacı:

Kısıtları:

Kalıcı Kayıtlar:

### Class Özellikleri

1. Özellik Adı:

Özellik Tipi:

Özellik Tanımı:

Kısıtları:

2. Özellik Adı:

Özellik Tipi:

Özellik Tanımı:

Kısıtları:

3. Özellik Adı:

Özellik Tipi:

Özellik Tanımı:

Kısıtları:

……

### Metod Tanımları

Metod Adı:

Metod Dönüş Tipi:

Parametreler:

Geri Dönüş Değerleri:

Önkoşul:

Sonraki Koşul:

Okunana/Kullanılan Özellikler:

Çağırılan Diğer Metodlar:

İşleme Mantığı/Adımları:

Test Durumu 1:

Test Durumu 2:

….

### Signature/API/Interface/Web Servisleri Tanımları

**İçerik**

Kullanılan hazır Signature/API/Interface/Web Servisleri tanımlanacaktır.

**Örnek**

Mimari, REST Web Hizmetlerini / API'yi ortaya çıkaracaktır. Temsili durum aktarımı (REST), Web hizmetleri oluşturmak için kullanılacak bir dizi kısıtlamayı tanımlayan bir yazılım mimari stilidir. RESTful Web hizmetleri adı verilen REST mimari spesifikasyonuna uyan web hizmetleri, İnternet üzerindeki bilgisayar sistemleri arasında birlikte çalışabilirlik sağlar. RESTful Web hizmetleri, talepte bulunan sistemlerin, tek tip ve önceden tanımlanmış bir vatansız işlemler kümesi kullanarak Web kaynaklarının metinsel temsillerine erişmesine ve bunları değiştirmesine izin verir.

# Programın Genel Özellikleri- Alt sistemler/Bileşen ve Modüller

## Programın Çalıştırılması

### Kullanıcı Fonksiyonları

**İçerik**

Projedeki kullanıcı fonksiyonları açıklanacaktır.

Programı kullanacak kişilerin fonksiyonları tek tek tanımlanmalıdır.

Genel Kullanıcılar

Kasiyer/Sekreter vb.

Yönetici

Müşteri

### Sunucu Fonksiyonları

**İçerik**

Projedeki Sunucu fonksiyonları açıklanacaktır.

### Yönetici Fonksiyonları

**İçerik**

Projedeki yönetici fonksiyonları hazırlanacaktır.

## Menü Bölümü

**İçerik**

Programınızın menü tasarımı anlatılacaktır.

**Örnek**



Şekil 5 Örnek Menü Tasarımı

## Rapor Tasarımları

**İçerik**

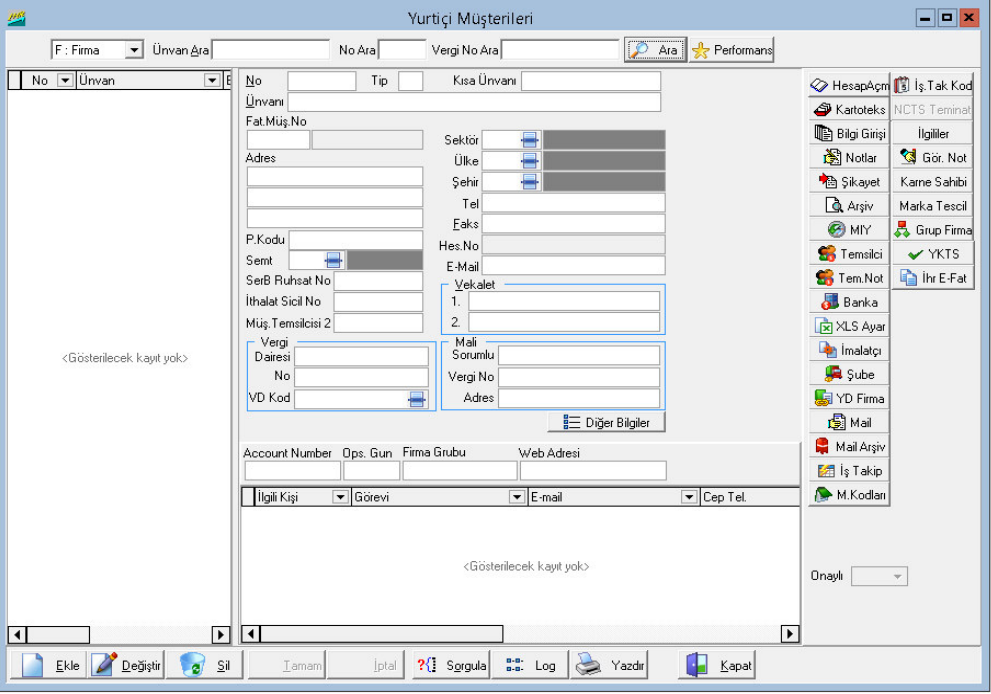
Projenizde kullanılan raporlama tooları ve raporların içeriği anlatılacaktır. Gerçekleştirim kısmında en az 2 rapor alınabilen bir sistem tasarımı yapınız.

## Modüller ve Ekran Tanıtımları

**İçerik**

Projenizdeki kullanılan modüller ve tüm ekranlar tanıtılacaktır.

**Örnek**



Şekil 6 Örnek Ekran Tasarımı

## Kullanıcı Girişi

**İçerik**

Kullanıcı girişini anlatınız.

**Örnek**



Şekil 7 Örnek Giriş Ekranı

## Kullanılan Kısayollar

**İçerik**

Kullanılan kısayollar anlatılacaktır.

**Örnek**

F4 Tuşu Ekran görüntüsü simgesi ile aynı işlevi görmektedir. Veri girişi işlemlerinde kolaylık ve hız sağlamak amacıyla klavyeden “F4” tuşuna basarak da veri girişi yapılan kutucuğa ilgili pencerenin açılması sağlanabilir.





Şekil 8 Örnek Kısayol Tanımı

## Programdan Çıkış

**İçerik**

Programdan Çıkış işlemleri anlatılmaktadır.

**Örnek**

“ …. Ekranı” sağ üst köşede bulunan işaretine basarak kapatabilirsiniz veya Alt+F4 bu şekilde masaüstünden programı ekrandan kaybolmasını sağlayabilir. Bu şekilde kapatılma sağlanır ise program arka planda çalışmaya devam edecektir

## Yazılımın Ölçülmesi COCOMO Modeli

**İçerik**

Kurumlar üç ana amaçla yazılımda ölçümü gündemlerine almaktadırlar:

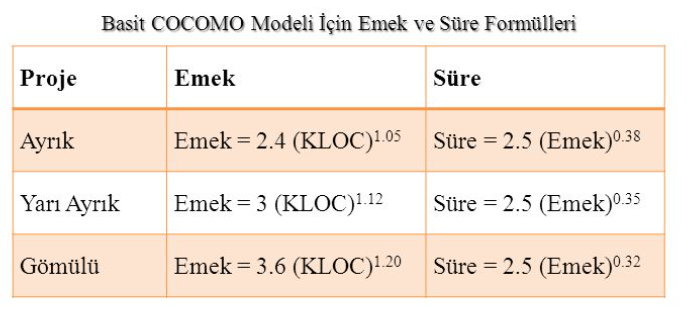
* Yazılım projesini anlamak ve modellemek,
* Yazılım projelerinin yönetilmesine yol göstermek,
* Yazılım süreç geliştirme ve iyileştirme çalışmalarını yön vermek.

Yazılımın ölçülebilmesi, harcanılan zaman, emek, proje büyüklüğü ve kalite gibi faktörlerin belirlenmesine olanak sağlamaktadır.

Beş Temel Yazılım Ölçütü

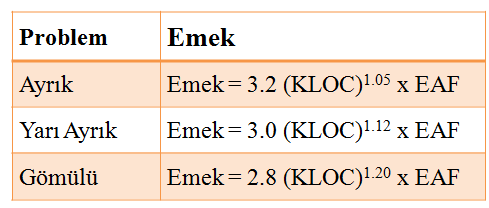
Büyüklük (Size): kodun satır sayısı.  
Emek (Effort): proje kapsamında ayda kaç kişi çalışacak.  
Maliyet (Cost): projenin ne kadara mal olacağı  
Zaman (Duration): proje kaç ayda tamamlanacak  
Kalite (Quality) : projenin kalitesi tespit edilen hata sayısından ölçülebilir.

Kullanılacak ayrıntı düzeyine göre Basit, Orta ve Detaylı olmak üzere üç temel model vardır. Basit COCOMO modeli, küçük-orta boy projeler için hızlı kestirim yapmak amacıyla kullanılır.



Orta COCOMO modeli sistemin (güvenilirlik, veri tabanı büyüklüğü, işletme ve kayıt sınırlandırmaları, personel özellikleri ve kullanılan yazılım araçları gibi) diğer özelliklerinin hesaba katılması amaçlanmıştır.

Orta COCOMO Modeli İçin Emek Formülleri



Detaylı COCOMO modeli projenin evrelerine bağlı olarak süreç içinde değişiklikleri hesaba katarak arada bir kestirim hesaplamasını önerir.

Detaylı COCOMO modeli, basit ve orta COCOMO modeline ek olarak iki özellik taşır. Aşama ile ilgili işgücü katsayıları: her aşama için (planlama, analiz, tasarım, geliştirme, test etme) farklı katsayılar, karmaşıklık belirler. Yazılım maliyet kestiriminde; Modül, Altsistem, Sistem sıra düzenini dikkate alır.

Ürün Özellikleri

RELY: Yazılımın güvenirliği.

DATA: Veritabanının büyüklüğü.

CPLX: Karmaşıklığı.

Bilgisayar Özellikleri

TIME: İşletim zamanı kısıtı.

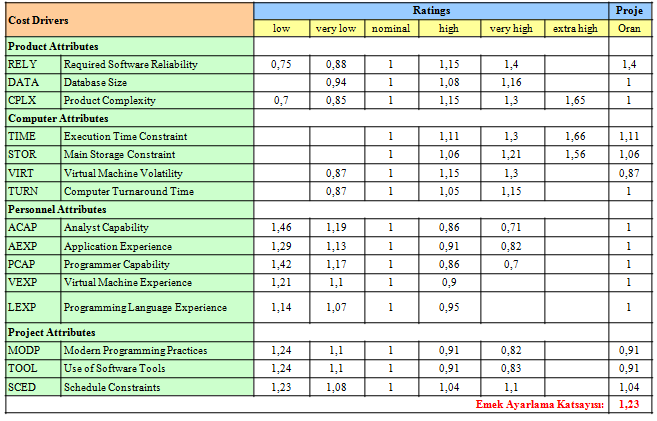
STOR: Ana bellek kısıtı

VIRT: Bilgisayar platform değişim olasılığı. Örn; bellek ve disk kapasitesi artırımı, CPU upgrade…

TURN: Bilgisayar iş geri dönüş zamanı. Örn; hata düzeltme süresi.

Personel Özellikleri  
ACAP: Analist yeteneği. Deneyim, birlikte çalışabilirlik.  
AEXP: Uygulama deneyimi. Proje ekibinin ortalama tecrübesi.  
PCAP: Programcı yeteneği.   
VEXP: Bilgisayar platformu deneyimi. Proje ekibinin geliştirilecek platformu tanıma oranı.  
LEXP: Programlama dili deneyimi.  
  
Proje Özellikleri  
MODP: Modern programlama teknikleri. Örn; Yapısal programlama, görsel programlama, yeniden kullanılabilirlik.  
TOOL: Yazılım geliştirme araçları kullanımı. Örn; CASE araçları, metin düzenleyiciler, ortam yönetim araçları  
SCED: Zaman kısıtı.

**Örnek**



Şekil 9 Örnek COCOMO Modeli

Emek = 3.0 x (KLOC)1.12 x EAF   
  
Emek = 3.0 x (7816)1.12  x 1,23 = 36,9 adam-ay  
  
Takvim= 2.5 x Emek 0,38= 2.5 x 36,90,38 = 9,84 ay (Geliştirme Zamanı)  
  
N = Emek  / Geliştirme Zamanı › (N: ortalama personel sayısı)  
  
N = 36,9 / 9,84 = 3,75 – 4 kişi

## Bakım, Sonuç ve Öneriler

**İçerik**

Proje bitiminde proje ile ilgili bakım (maintenance) ve güncelleme raporu önerilerinizi yazınız.

Değişim yönetimi nasıl ve kimler tarafından yapılacak belirtiniz.

## Ekler

**İçerik**

Projenizdeki ekleri belirtiniz.