T.C.
FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Proje Dokümantasyonu

(Kimlik Doğrulama)

Proje Ekibi

WEB SUNUCUSU	ANDROİD UGULAMASI	VERİ TABANI
Mert XXXXX	İhsan ALPTEKİN	Turgay XXX
Mustafa XXXX	Muhammed XXXX XXXXX	Cihat Can XXXX
Emre R. XXXX	Ömer XXXX XX	Betül XXX
Mert Can XXXX	Sedat XXXXX	Damla XXXX
Özgenur XXXX	Bektaş XXXXX	Selim XXXX
Nisa XXXXXX XXXX	Akın XXXX	Asiye XXXX
	Abdullah XXXXXX	Yasir XXXX

10/01/2020 – Versiyon 5

1. GİRİŞ

1.1 Projenin Amacı

1.2 Projenin Kapsamı

1.3 Tanımlamalar ve Kısaltmalar

2. PROJE PLANI

2.1 Giriş

2.2 Projenin Plan Kapsamı

2.3 Proje Zaman-İş Planı

2.4 Proje Ekip Yapısı

2.5 Önerilen Sistemin Teknik Tanımları

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

2.7 Proje Standartları, Yöntem ve Metodolojiler

2.8 Kalite Sağlama Planı

2.9 Test Planı

2.10 Bakım Planı

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

3.1.1 Örgüt Yapısı

3.1.2 Varolan Yazılım/Donanım Kaynakları

3.1.3 Varolan Sistemin Değerlendirilmesi

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

3.2.2 İşlevsel Model

3.2.3 Genel Bakış

3.2.4 Bilgi Sistemleri/Nesneler

3.2.5 Veri Modeli

3.2.6 İşlevlerin Sıradüzeni

3.2.7 Başarım Gerekleri

3.3 Arayüz (Modül) Gerekleri

3.3.1 Kullanıcı Arayüzü

3.3.2 Yönetim Arayüzü

3.4 Belgeleme Gerekleri

3.4.1 Geliştirme Sürecinin Belgelenmesi

3.4.2 Eğitim Belgeleri

3.4.3 Kullanıcı El Kitapları

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

4.1.2 Varsayımlar ve Kısıtlamalar

4.1.3 Sistem Mimarisi

4.1.4 Veri Modeli

4.1.5 Testler

4.2 Veri Tasarımı

4.2.1 Tablo tanımları

4.2.2 Tablo- İlişki Şemaları

4.2.3 Veri Tanımları

4.2.4 Değer Kümesi Tanımları

4.3 Süreç Tasarımı

4.3.1 Genel Tasarım

4.3.2 Modüller

4.3.2.1 Öğrenci Modülü

4.3.2.1.1 İşlev

4.3.2.1.2 Kullanıcı Arabirimi

4.3.2.1.3 Modül Tanımı

4.3.2.1.4 Modül iç Tasarımı

4.3.3 Kullanıcı Profilleri

4.3.4 Entegrasyon ve Test Gereksinimleri

4.4 Ortak Alt Sistemlerin Tasarımı

4.4.1 Ortak Alt Sistemler

4.4.2 Yetkilendirme Alt Sistemi

4.4.3 Güvenlik Altsistemi

4.4.4 Yedekleme ve Arşivleme İşlemleri

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1. Giriş

5.2. Yazılım Geliştirme Ortamları

5.2.1 Programlama Dilleri

5.2.2 Veri Tabanı Yönetim Sistemleri

5.2.2.1 VTYS Kullanımının Ek Yararları

5.2.2.2 Veri Modelleri

5.2.2.3 Şemalar

5.2.2.4 VTYS Mimarisi

5.2.2.5 Veritabanı Dilleri ve Arabirimleri

5.2.2.6 Veri Tabanı Sistem Ortamı

5.2.2.7 Hazır Program Kütüphane Dosyaları
5.2.2.8 CASE Araç ve Ortamları
5.3. Kodlama Stili
5.3.1 Açıklama Satırları
5.3.2 Kod Biçimlemesi
5.3.3 Anlamlı İsimlendirme
5.3.4 Yapısal Programlama Yapıları
5.4. Program Karmaşıklığı
5.4.1 Programın Çizge Biçimine Dönüştürülmesi
5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama
5.5. Olağan Dışı Durum Çözümleme
5.6. Kod Gözden Geçirme
5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi
5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular
5.6.2.1 Öbek Arayüzü
5.6.2.2 Giriş Açıklamaları
5.6.2.3 Veri Kullanımı
5.6.2.4 Öbeğin Düzenlenişi
5.6.2.5 Sunuş

6. DOĞRULAMA VE GEÇERLEME

6.1. Giriş
6.2. Sınama Kavramları
6.3. Doğrulama ve Geçerleme Yaşam Döngüsü
6.4. Sınama Yöntemleri
6.4.1 Beyaz Kutu Sınaması
6.4.2 Kara Kutu Sınaması
6.5. Sınama ve Bütünleştirme Stratejileri
6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme
6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme
6.6. Sınama Planlaması
6.7. Sınama Belirtimleri
6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri

7. **BAKIM**

7.1 Giriş
7.2 Yerinde Destek Organizasyonu
7.3 Yazılım Bakımı

8. SONUÇ

9. KAYNAKLAR

1. GİRİŞ

1.1. Projenin Amacı

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

1.2. Projenin Kapsamı

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

1.3. Tanımlamalar ve Kısaltmalar

Öğrenci: Derse gelip yoklamaya imza atan kişi.

Öğretmen: Yoklama uygulamasını yönetecek olan kişi.

2. PROJE PLANI

2.1 Giriş

Bu kapsamda öncelikle inceleme ve analiz yapılacak, bütün sistem bu inceleme ve analiz kısmına dayandırılacaktır. İnceleme kısmında projenin en ince detayına kadar müşteriden bilgi alınacak ve analiz edilecektir. Bundan sonra yapılabilirliğin hesabı yapılacak ve olumlu sonuçlanırsa projeye başlanacaktır.

Daha sonra izlenecek yolun belirlenmesi, kaynak ihtiyaçları ve gider tahmini, proje süresi ve yazılım geliştirmeye düşen görev ve işlemler analiz edilecek ve sonuçlandırılacaktır.

2.2 Projenin Plan Kapsamı

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 ya da 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.

İşlev Noktası Hesaplama

İN: İşlev Nokta Sayısı, AİN: Ana İşlev Nokta Sayısı, TKF: Teknik Karmaşıklık Faktörü

İN = AİN x (0.65 x 0.01 x TKF)

122 x (0.65 x 0.01 x 59)= 46,787

Tahmini oluşacak satır sayısı:

Satır Sayısı = İN *30

Satır Sayısı = 1.403,61

Etkin Maliyet Modeli – COCOMO

İş gücü (K) $K = a \times S^b$

Zaman (T) $T = c \times K^d$

a, b, c, d: her bir model için farklı olan katsayılar

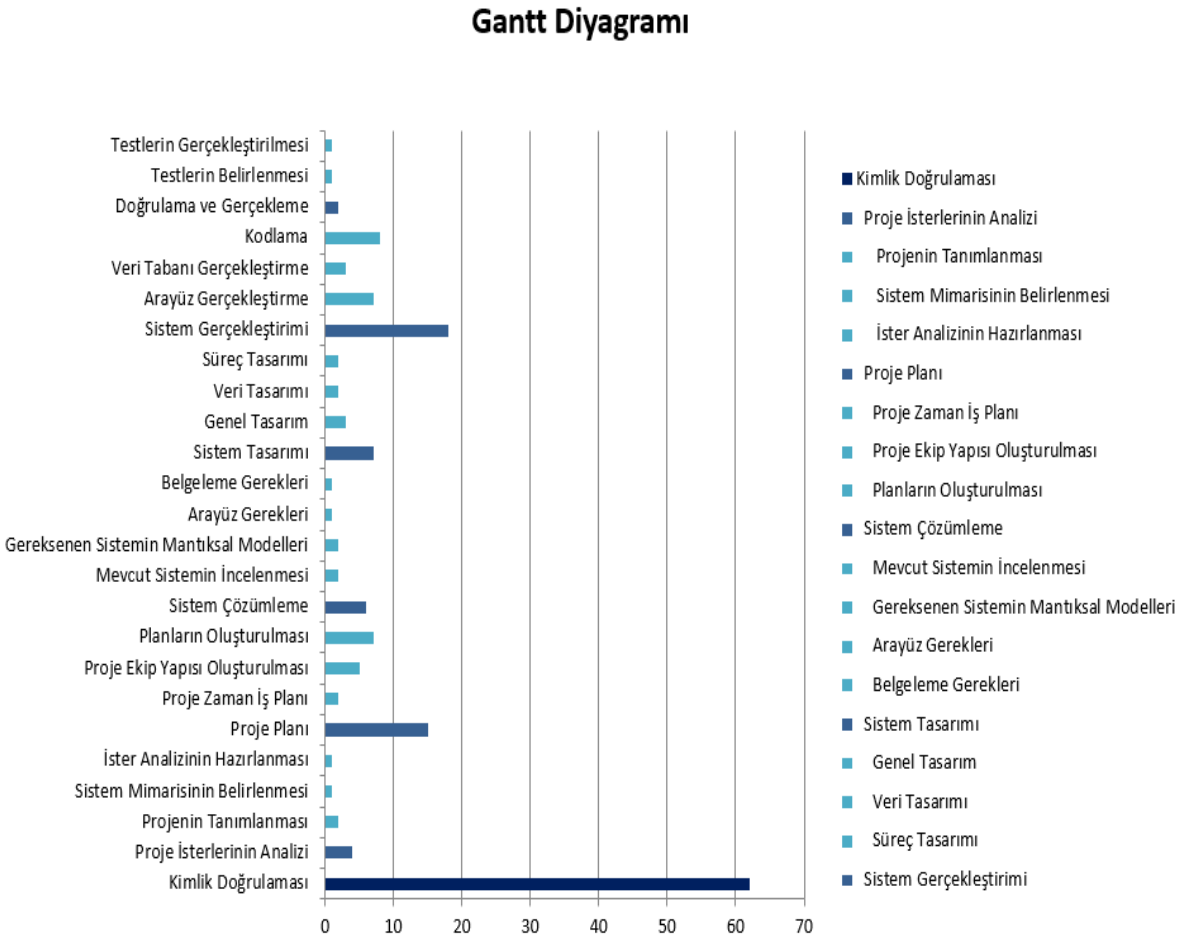
S = bin türünden satır sayısı

İş gücü (K) K = 2,4 x 4^1.05 = 10,28 İş Gücü

Zaman (T) T = 1,12 x 10^0,38

=2,68 ay

2.3 Proje Zaman-İş Planı



2.4 Proje Ekip Yapısı

- Product Owner: Product Owner temel olarak geliştirilecek ya da mevcut olan ürünü sahiplenen, yöneten kişidir. Ürünün içerisinde bulunduğu pazardaki stratejik konumunu değerlendirerek rakipleri ile kıyaslandığında sahip olduğu artıları ve eksileri bağlı olduğu yönetim ile görüşerek projeyi şekillendirir. Product Owner aynı zamanda yazılım geliştirme ekibinin hangi işleri yapacağını yapılacak işin önceliği ve değerine göre belirleyen kişidir.
- 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.
- Front-end Developer: Back-end kısmı olarak veri tabanında kullanıcı datalarına sahip olmak güzel; ancak diğer yandan web sitenizin görüntüsünü iyi yapan şey ne? İşte bu noktada Front-end devreye giriyor. Ziyaret edilen her sayfa için güzel bir görünüm ve iyi bir his yaratıyor. Başka bir deyişle direkt etkileşime geçtiğiniz ve gözünüzle görebildiğiniz her sayfa için renk, fon ve diğer tüm stilleri oluşturuyor. Web sitesi kullanıcılarına marka kimliğini yansıtıp duygusal deneyim yaşatıyor.Front-End için ise; HTML, CSS ve JavaScript programlama dilleri kullanılıyor. Bu programlama dilleri web sitesinin size yansıyan görünüşüne odaklanıyor.
- Back-end Developer: Tanımlandığı kelime öbeğinden de anlaşılacağı üzere, bir web projesinin son kullanıcının görmediği çekirdek yazılım kısmına Back-End ve bu mimariyi

kodlayan yazılımcıya ise Back-End Developer, yani yazılım geliştirici denir. Back-End altyapısı için kullanılan yazılım dilleri; genelleme yaparak aşina olduğumuz PHP ve ASP.net, veritabanı mimarisi için ise MYSQL ve MSSQL'dir. Mimaride kullanılan bu bileşenler ile örneğin bir CMS (Content Management System – İçerik Yönetim Sistemi) oluşturmak ve bir web sitesinde kullanıcı dostu bir admin panel vasıtası ile her bir sabit – statik alanı dinamik olarak yönetilebilir kılmak mümkündür.

- Sistem Çözümleyici: Bilgi işlem sistemlerini kuran ve yeni bilgi toplayan, sistemlerin kurulmaları ve çalışmaları için gerekli yöntemleri tanımlayan, kurulumlarını yapan, denetleyen ve gelişmeleri için önerilerde bulunan nitelikli kişi.
- Sistem Tasarımcı: Sistem çözümleyicinin tanımladığı gereksinimleri mantıksal, ekonomik ve pratik sistem tasarımlarına dönüştürerek ilgili programların yazılabilmesi için gerekli ayrıntılı spesifikasyonları hazırlayan kişidir.
- Sistem Yöneticisi: Sistem yöneticisi, projenin ihtiyaçlarını analiz ederek bilgisayar sistemlerini tasarlama, kurma, destekleme, geliştirme, sürekliliğini ve güvenliğini sağlama işini yapar.
- Veri Tabanı Yöneticisi: Veri tabanı sistemlerinin kurulması, konfigürasyonun yapılması, tasarlanması, sorgulanması ve güvenliğinin sağlanması işlemlerini üstlenmiştir.

2.5 Önerilen Sistemin Teknik Tanımları

Sistem, Back-end kısmı Javascript, Front-end kısmı VueJs ile, mobil kısmı Androin ile geliştirilecek. Bu uygulama öğrenciler ve öğretmen tarafından kullanılacak ve kullanılabilirlik açısından olabildiğince basit tasarlanacaklardır.

Öğretmenin öğrencileri takip yapabilmesini sağlayan ve denetlenebilen bir web program geliştirilecektir.

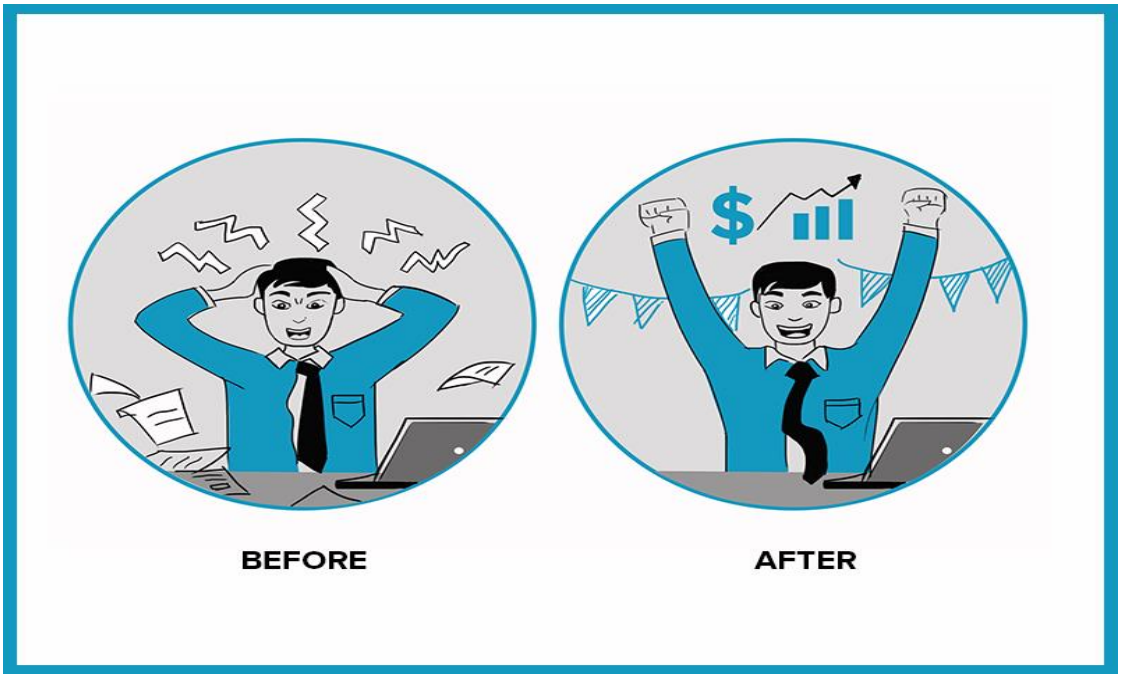
2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

İşletim Sistemleri	Programlama Dilleri	VTYS	Derleyiciler
• İOS • Windows • Android	• JavaScript • VueJs • Android • Java	• MySQL • MockData • (PL/SQL)	• Netbeans • Eclipse • Visual Studio Code

2.7 Proje Standartları, Yöntem ve Metodolojiler

Yazılım geliştirme sürecinin yapısını ve adımlarının uygulanış biçimini, seçilen yazılım geliştirme modeli belirlemektedir. Dikkat edilmesi gereken nokta eldeki ürüne ve sürece uygun modelin seçilebilmesidir. Bu projede, kişisel bilgi yönetim yazılımı olarak bir takip uygulaması oluşturmak için yazılım geliştirme modellerinden biri olan Scrum model kullanılmıştır.

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.

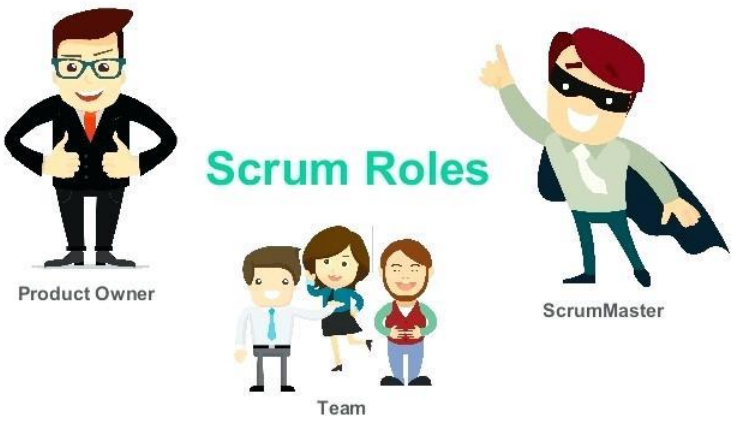


SCRUM SÜRECİ

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.

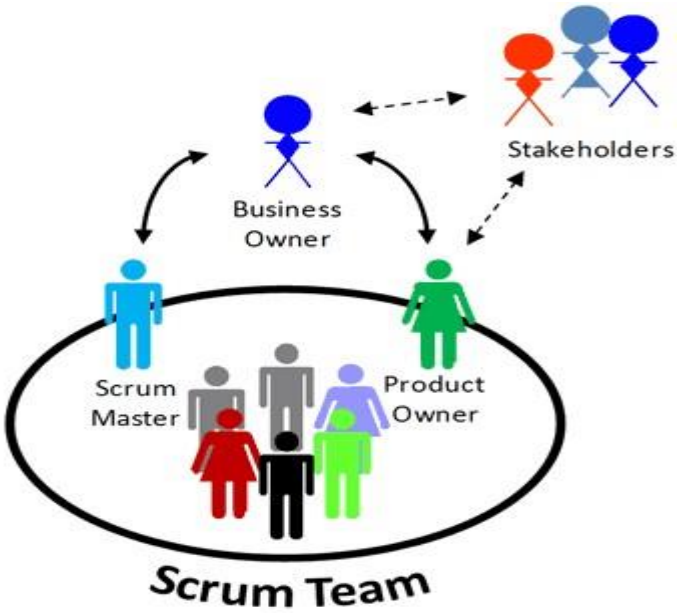
Scrum'da 3 rol vardır;

- Product Owner
- Scrum Master



- Team Member

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

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.

SPRİNT

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

SER 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.

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.

2.8 Kalite Sağlama Planı

Buna rağmen, yazılım kalitesi basit bir yolla tanımlanması mümkün olmayan karmaşık bir kavramdır. “Klasik olarak, kalite kavramı, üretilen ürünün belirtimlerini karşılaması gerektiğini ortaya koyar (Crosby, 1979).”

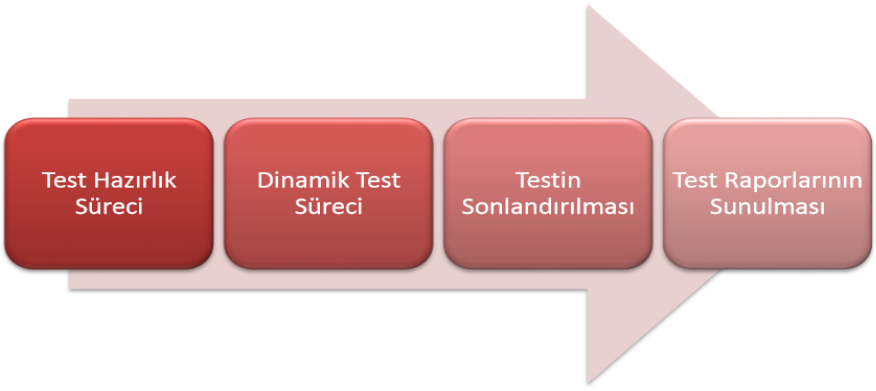
Kalite sağlama üzerine geliştirilmiş belirtimler gerçek dünyadaki çoğu ürün için uygulanabilse de yazılım için istenilen seviyeye ulaşamamış, tam olarak kaliteyi ölçmekte yararlı olamamışlardır.

Bu projede yazılım kalite yönetimi üç temel davranış ile yapılandırılacaktır:

Kalite Sağlama Planı	*Kalite Güvence:* Yüksek kaliteye sahip yazılıma götürecek kurumsal yordam ve standartlar çatısının ortaya konması.
	Kalite Planlama: Bu çatıdan uygun standart ve yordamların seçimi ve bunların projeye uyarlanması.
	Kalite Kontrol: Proje kalite standart ve yordamlarının yazılım geliştirme takımı tarafından takip edildiğini garanti eden süreçlerin tanımlanması ve belgelenmesi.

2.9 Test Planı

Scrum modelde analiz, kodlama, test ve kullanım birbirini izler. Yani her bir prototip kullanıcıya verilmeden her seferinde test edilir. Bu testleri Yazılım Test Ekibi yapacaktır. Kodlamanın bitmesi beklenmeden test hazırlık sürecine geçilecektir. Projenin test süreci;



Test Planı

- Test Hazırlık Süreci

Testin sağlıklı geçebilmesi için önceden planlanmış bir test planına ihtiyaç vardır. Bu süreçte bu test planı hazırlanacaktır. Test hazırlık sürecinde şu standartlara uyulacaktır:

- ✓ Öncelikle test edilecek projenin analiz ve teknik tasarım aşamaları ile ilgili dökümanlar, test ekibi tarafından incelenecektir.
- ✓ Yazılım içinde test edilecek ve edilmeyecek modüller belirlenecektir.
- ✓ Risk analizi yapılır ve yapılan değerlendirmeye göre dinamik test aşamasında uygulanacak olan test teknikleri ve metodları belirlenir.
- ✓ Dinamik testin uygulanacağı ortamlar ve bu ortamların ihtiyaçları belirlenip, uygun şartlar sağlanacaktır.
- ✓ Test ekibi içinde görev paylaşımı ve zaman planlaması yapılır.

- ✓ Testin sonlandırma kriterleri belirlenir.
- ✓ Bir programa belirli girdiler (input) verildiğinde hangi çıkışların (output) ne şekilde alınması gerektiğini bildiren test case senoryaları belirlenir.
- ✓ Dinamik testin hangi adımlarla ve ne şekilde uygulanacağının belirtildiği test planı hazırlanır.

- Dinamik Test Süreci

Bu süreç kodlama çalışmalarının bitmesine yakın bir dönemde başlayacaktır. Bulunan tüm hatalar çözülmeden ve testin sonlandırma kriterleri sağlanmadan sona ermez. Test edilecek yazılımın türüne göre, dinamik olarak uygulanacak test teknikleri ve bu tekniklerin uygulanma metotları farklılık gösterebilir.

- Testin Sonlandırılması

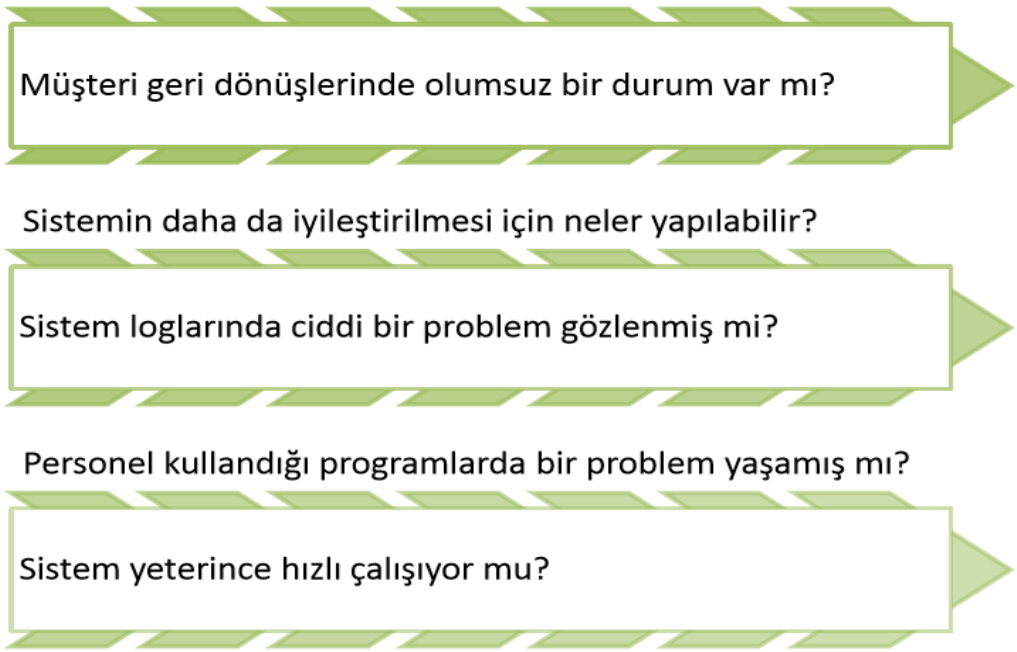
Yapılan testler sonucunda bulunan hatalar düzeltildikten sonra test sonlandırma kriterleri (test hazırlık süreci) kontrol edilir. Eğer tüm kriterlerin kabul edilebilir düzeyde sağlandığı tespit edilirse test sonlandırılır. Testin sonlandırılmasının ardından uygulama müşteri testine açılır (Kullanıcı Kabul Testi). Müşterilerin bulduğu hatalar veya değiştirilmesi istenilen noktalar gözden geçirilerek tekrar test ekibinin kontrolüne sunulur. Bu kontrolden çıkan uygulama ürün aşamasına geçer ve böylelikle yazılım test süreci sona erdirilerek, yazılım geliştirme sürecinin son basamağına geçilmiş olunur.

- Test Raporlarının Sunulması

Bu süreçte testler yapıldıktan sonra raporlanıp yetkili kişiye verilecektir.

2.10 Bakım Planı

Sistem, ilk 1 sene her ay bakıma alınacaktır. 1 seneden sonra ciddi sorun çıkarabilecek hatalar düzeltileceği için 3 ayda 1 bakım yapılacaktır. Bakım yapılırken şu şartlar aranacaktır:



Aylık Bakım Planı

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

Projenin gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve düzgün bir şekilde, terimsel olarak açıklanması bu bölümde yapılacaktır.

3.1.1 Örgüt Yapısı

3.1.2 Varolan Yazılım/Donanım Kaynakları

Microsoft Project: MS Project, Microsoft tarafından geliştirilen ve satılan, proje yöneticilerine plan oluşturma, kaynakların görevlere atanması, aşama takibi, bütçe yönetimi ve iş yükü analizi gibi konularda yardımcı olması amacıyla tasarlanmış bir proje yönetimi yazılımıdır. Bu projede Gantt Diyagramı çizilmesinde faydalanılmıştır.

Microsoft Word: Dünyanın en popüler metin kontrol uygulamasıdır. Bu projede dokümantasyonun hazırlanmasında faydalanılmıştır. 3.1.4.3 ArgoUML

ArgoUML: Ücretsiz olarak UML diyagramlarını çizmeye, modellemeye yarayan kullanımı kolay bir UML çizim programıdır. Bu projede diyagramların çiziminde faydalanılmıştır.

Adobe Photoshop CS5: Photoshop, piksel tabanlı görüntü, resim ve fotoğraf düzenlemede bir tek biçim olan, Adobe Sistem'in sayısal fotoğraf işleme yazılımıdır. Bu projede belirli resimlerin çizilmesinde faydalanılmıştır.

Microsoft Visio: Kapsam diyagramı ve örgüt yapısı çiziminde yararlanılmıştır.

SmartDraw: Use Case Diyagramları hazırlanırken faydalanılmıştır.

3.1.3 Varolan Sistemin Değerlendirilmesi

Şuan da yapılan projeler arasında bizim projemizin maliyet en az olandır. Projemizin öne çıkan bir başka özelliği kullanımı oldukça basit ve anlaşılabilir olmasıdır.

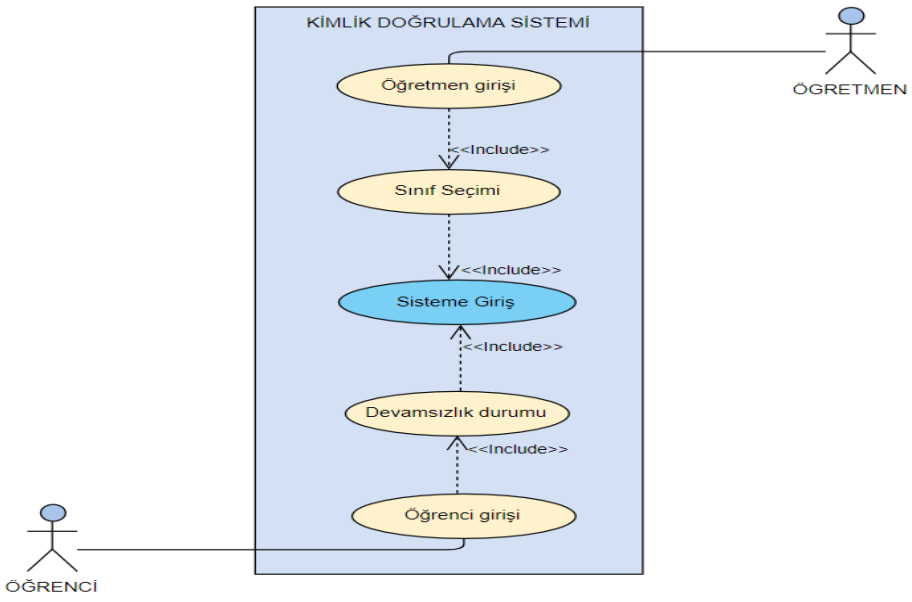
3.2 Gereksenen Sistemin Mantıksal Modeli

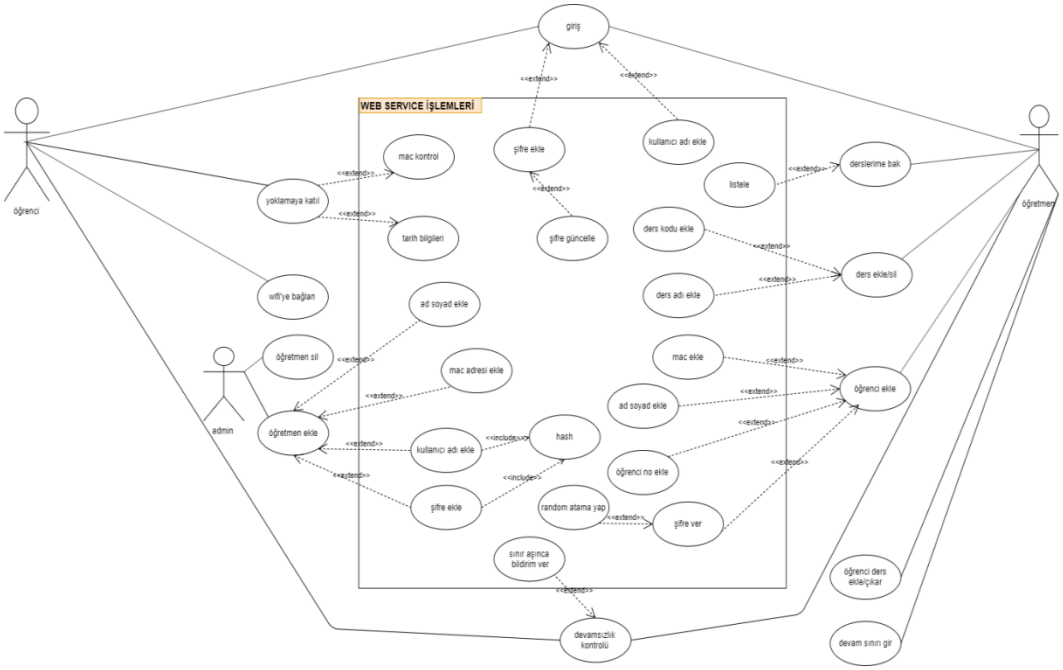
3.2.1 Giriş

Bu bölümde önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünde çözümleme yapılır. Bu model daha çok bilgi sistemini geliştirecek teknik personele yöneliktir. Mantıksal model olarak da tanımlanır.

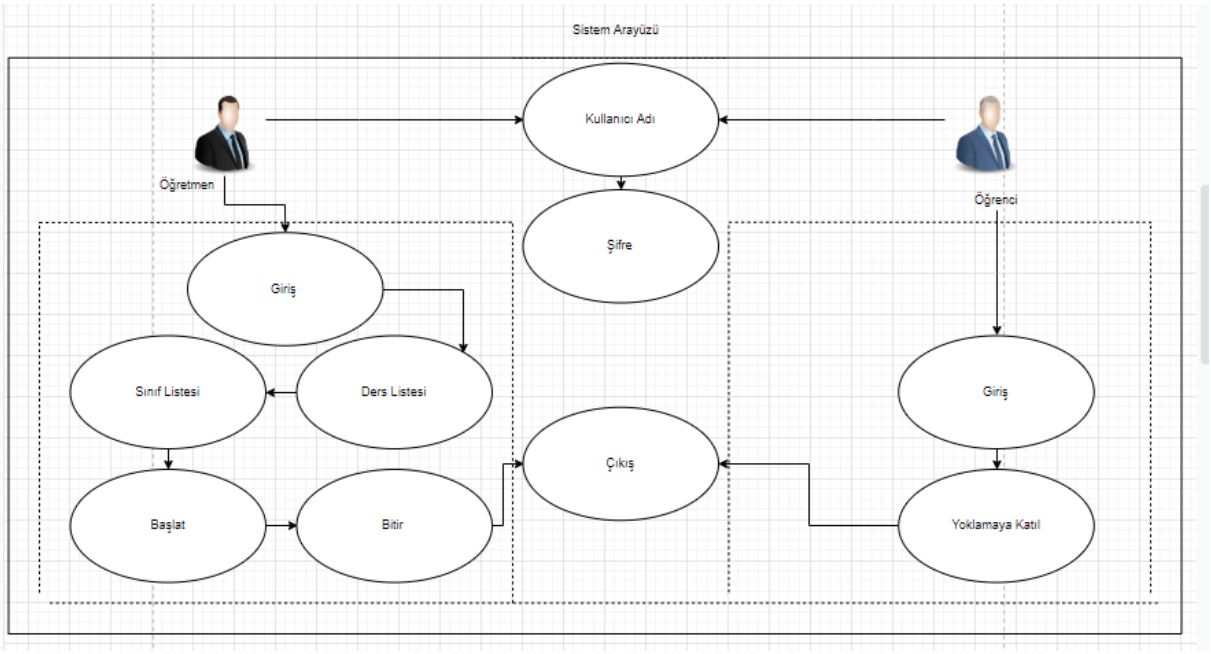
3.2.2 İşlevsel Model

3.2.2.1 İşlevsel Model (Web Servis)





3.2.2.2 İşlevsel Model (Mobil Uygulama)



3.2.2 Genel Bakış

Senaryo Adı: Ders Tanımlanması

Öğretmen: Sisteme giriş yaparak kod oluşturur ve sınıfı o koda atar.

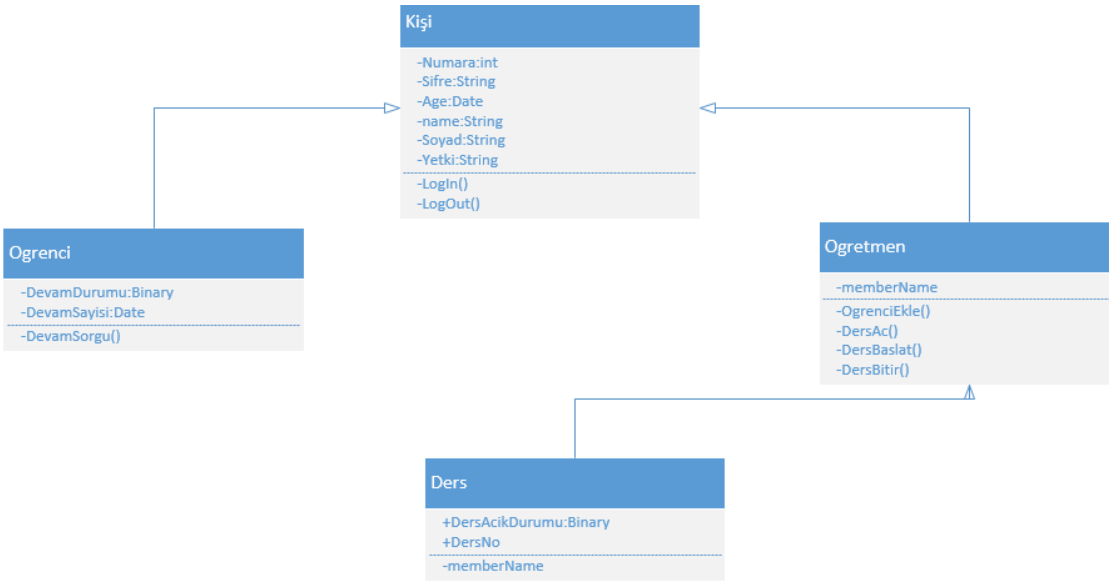
Öğrenci: Sisteme giriş yapar ve sınıfı öğrenir.

Senaryo Adı: Yoklamayı Başlatma

Öğretmen: Kendi telefonundan kişisel erişim noktasını açar.

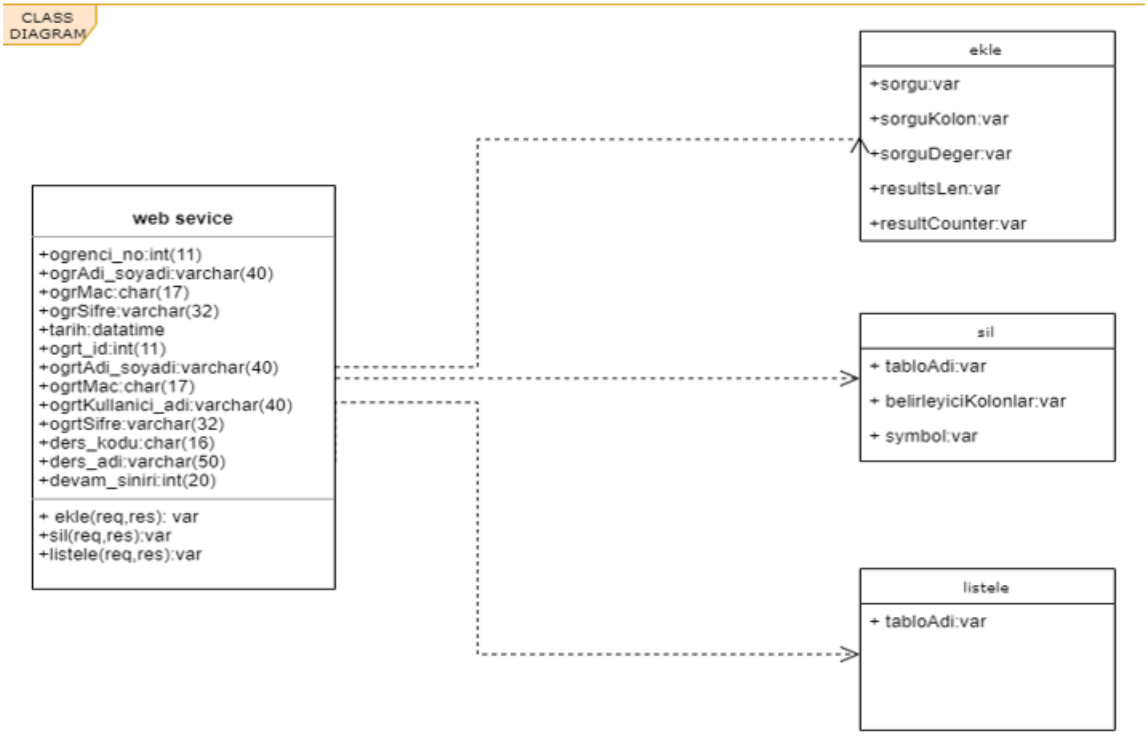
Öğrenci: Kendi telefonundan öğretmenin ağına katılır.

3.2.3 Bilgi Sistemleri/Nesneler

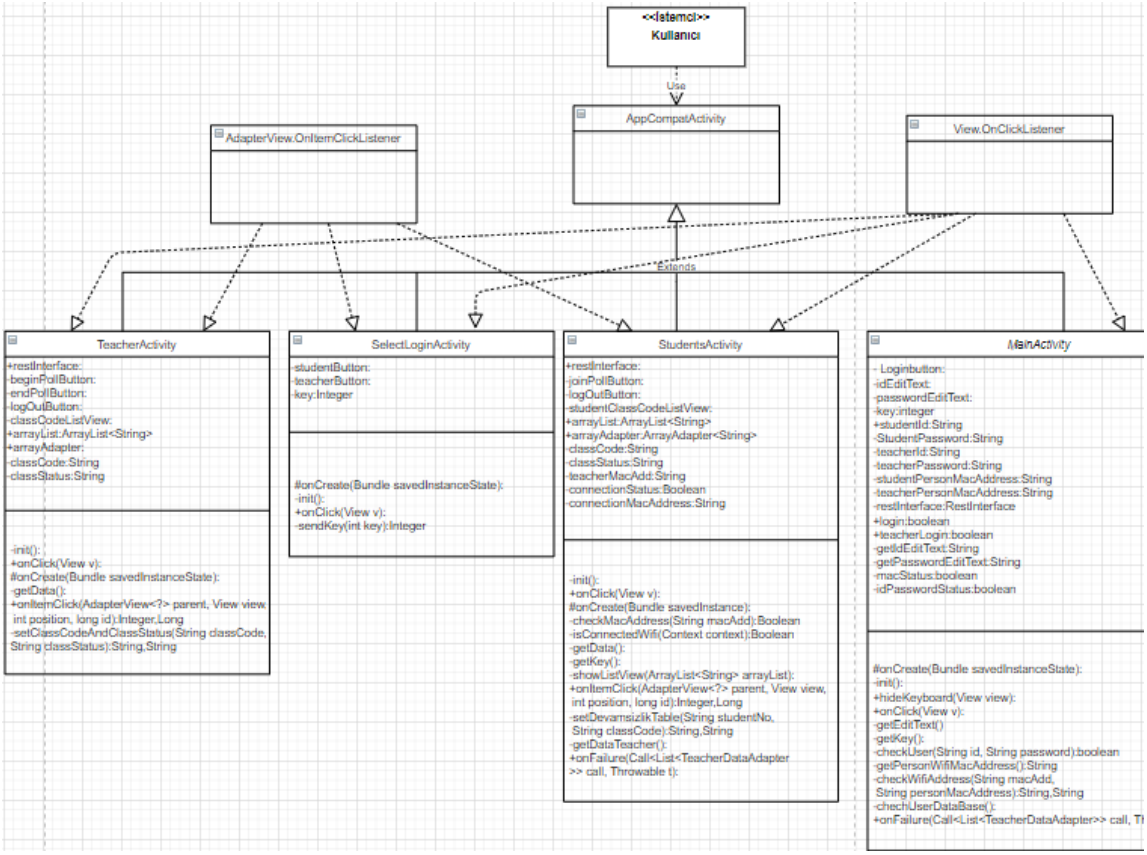


3.2.4 Veri Modeli

3.2.4.1 Veri Modeli (Web Servis)

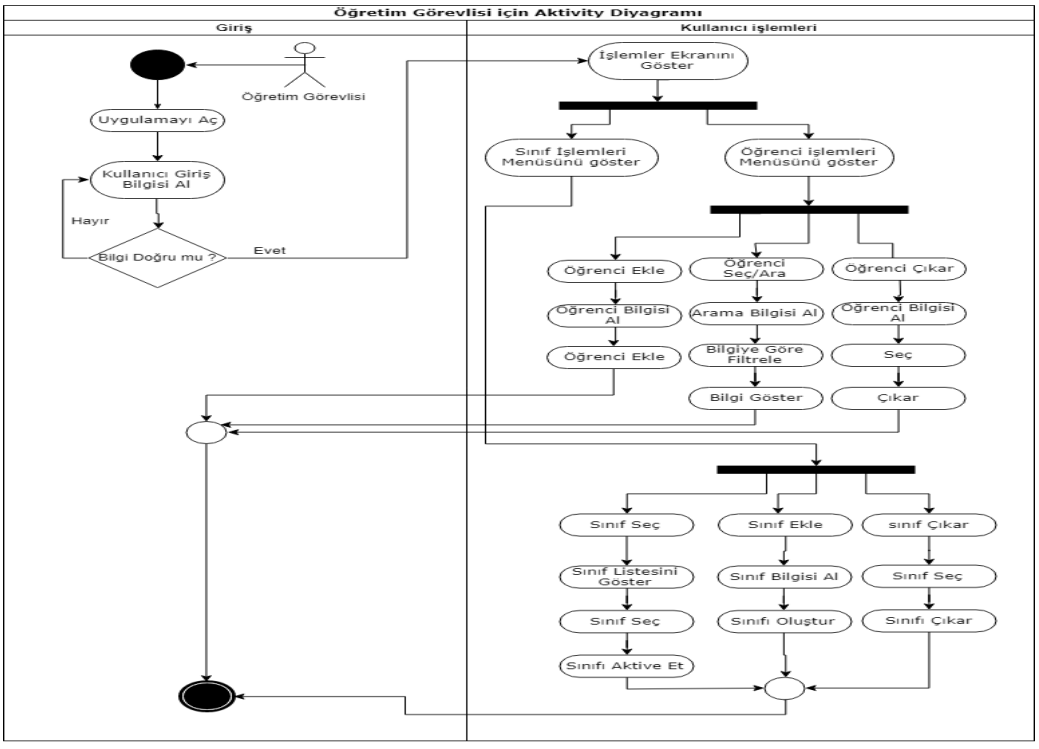


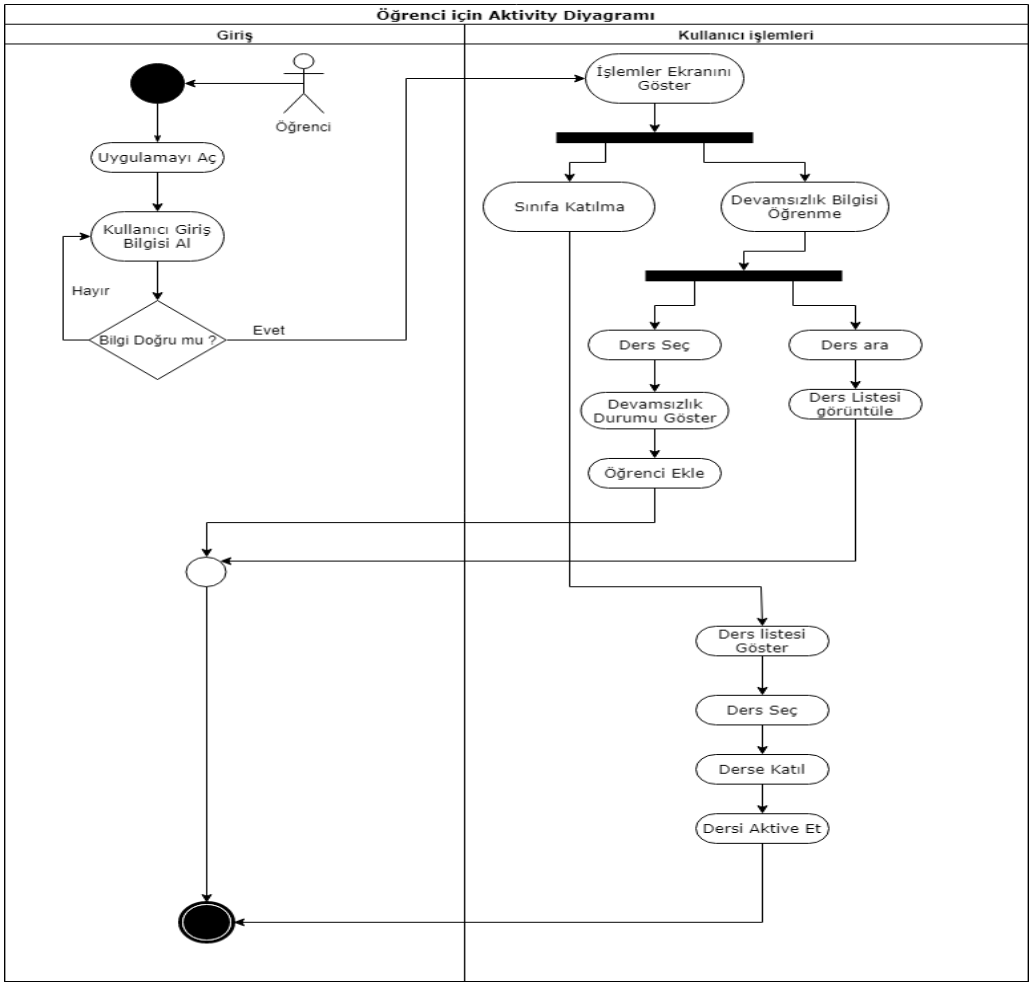
3.2.4.2 Veri Modeli (Mobil Uygulama)



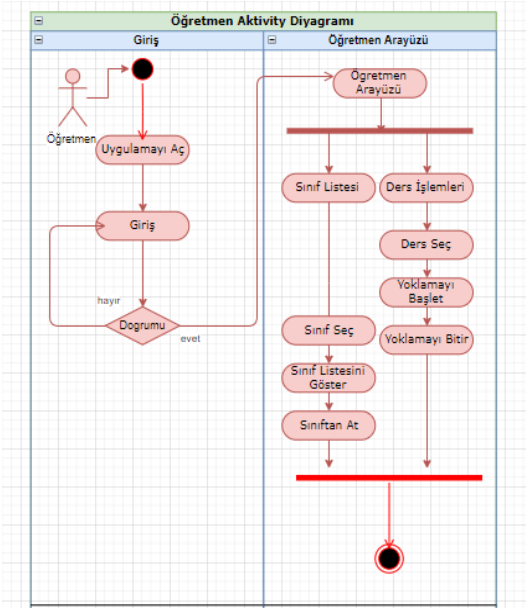
3.2.5 İşlevlerin Sıradüzeni

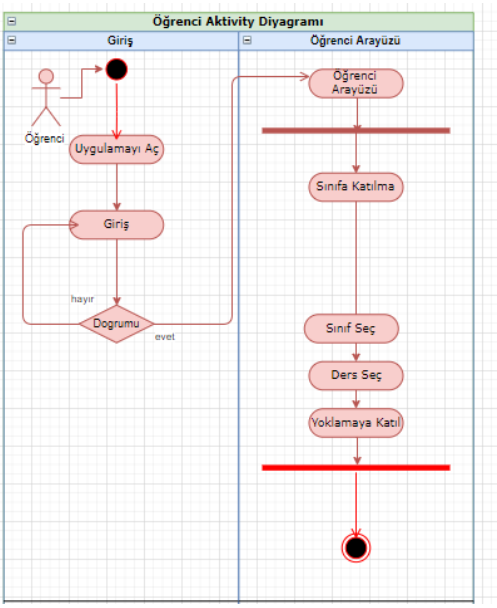
3.2.5.1 İşlevlerin Sıradüzeni (Web Servis)





3.2.5.2 İşlevlerin Sıradüzeni (Mobil Uygulama)



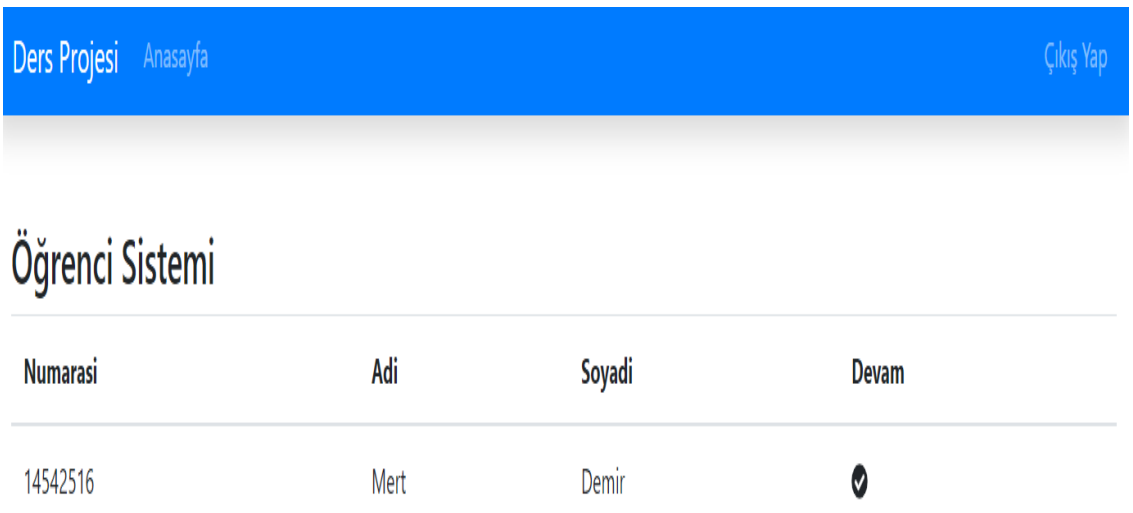


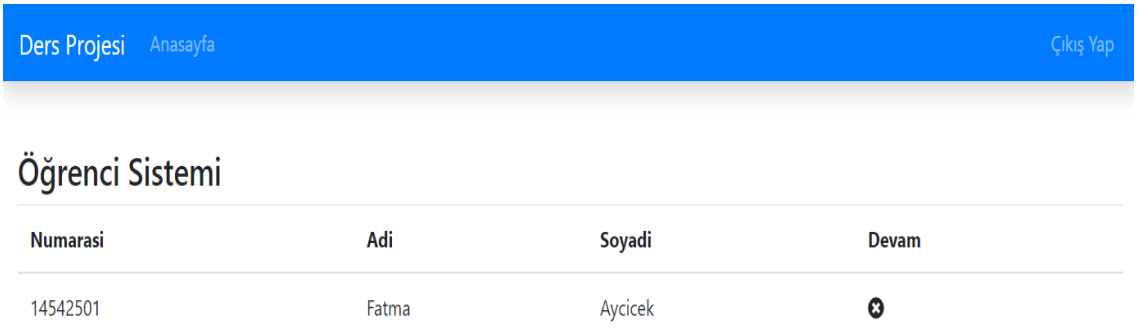
3.2.6 Başarım Gerekleri

- ❖ Sistem hızlı çalışmalı
- ❖ Ara yüz basit olmalı
- ❖ Personel ekip yapısı düzgün seçilmeli
- ❖ Personel eğitimi yapılmalı
- ❖ Güvenlik sorunları çıkmamalı
- ❖ Kullanıcılardan geri dönüş alınmalı

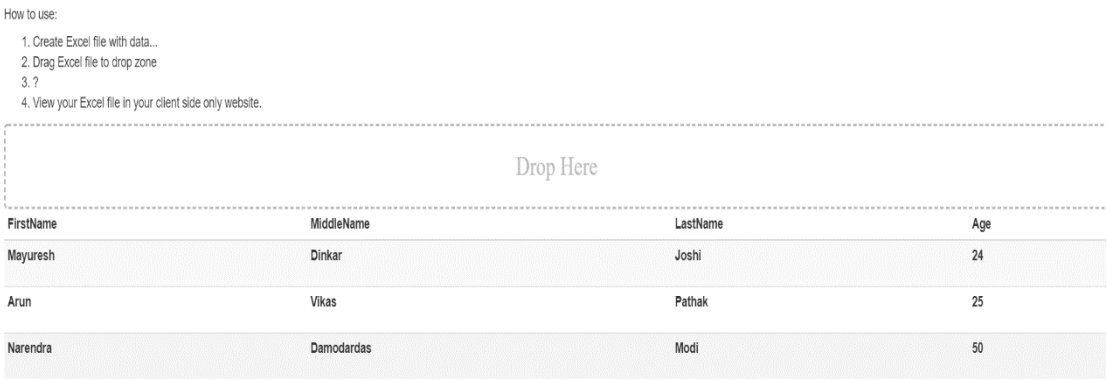
3.3 Arayüz (Modül) Gerekleri

3.3.4 Kullanıcı Arayüzü (Web Servis)

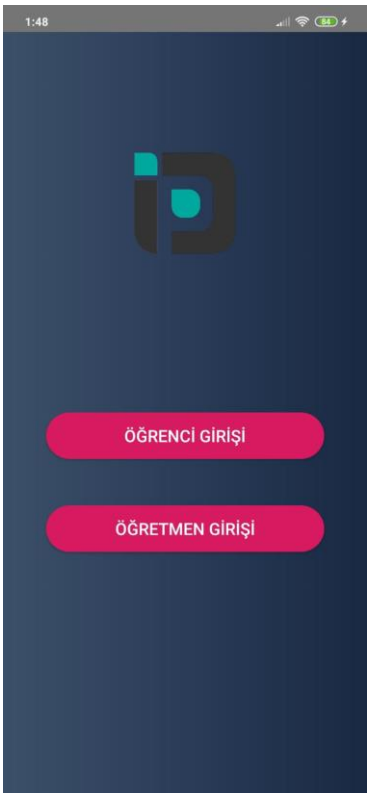


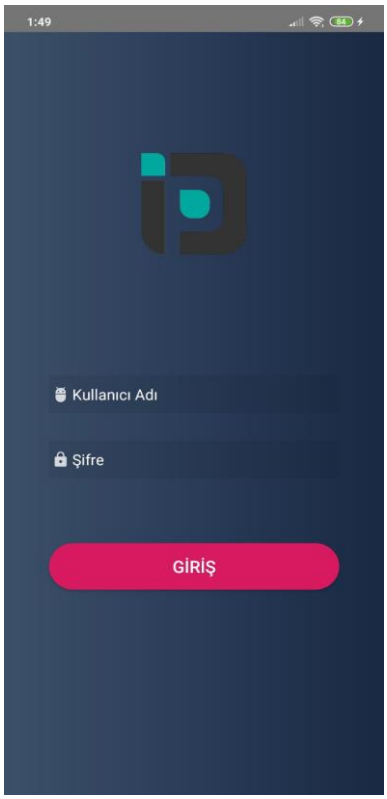


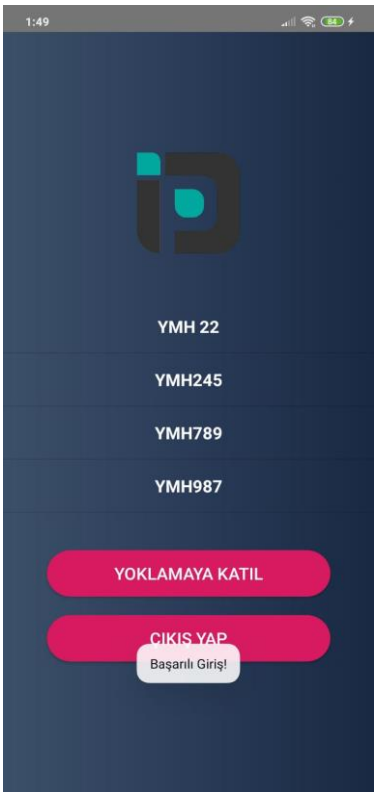
3.3.5 Yönetim Arayüzü (Web Servis)

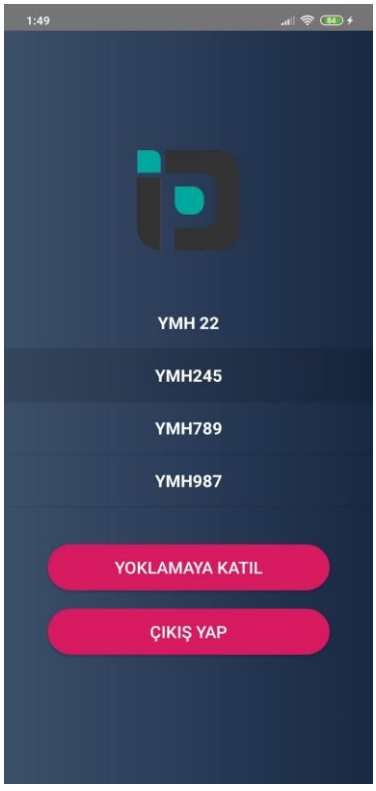


3.3.6 Kullanıcı Arayüzü (Mobil Uygulama)

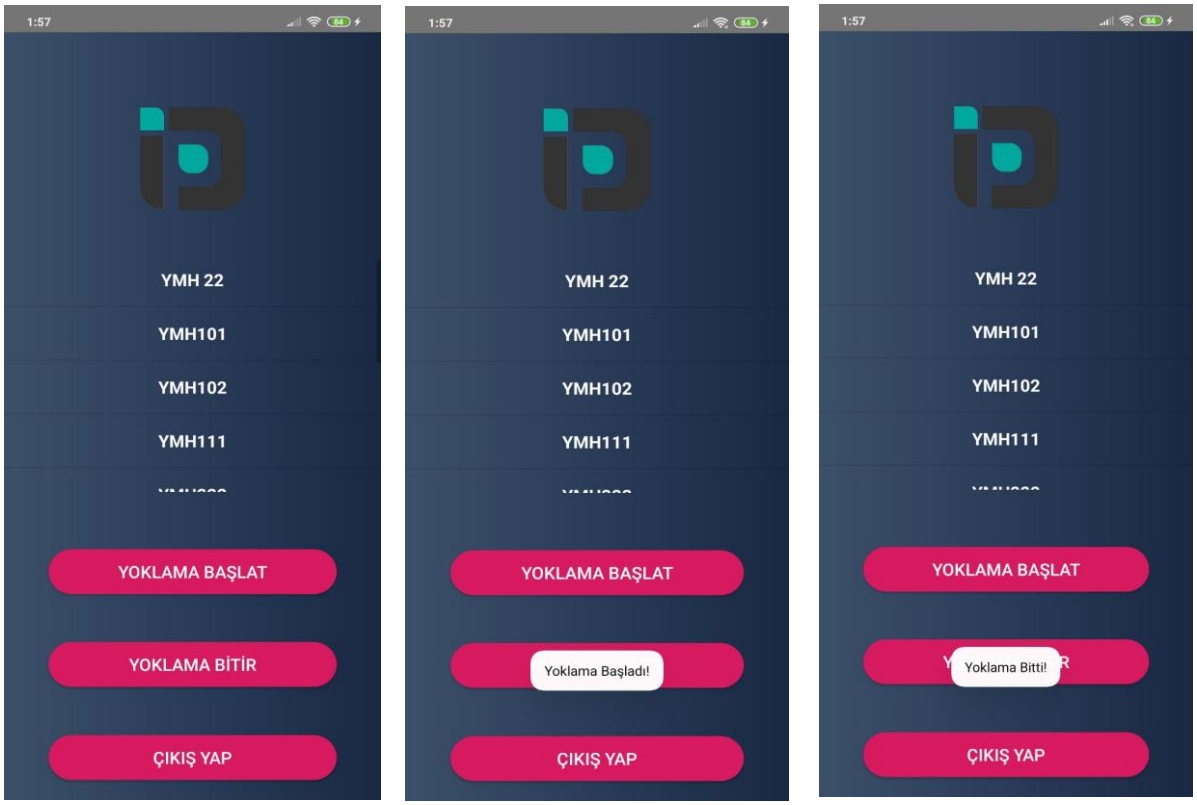








3.3.7 Yönetim Arayüzü (Mobil Uygulama)



3.4 Belgeleme Gerekleri

3.4.4 Geliştirme Sürecinin Belgelenmesi

Belgeleme Microsoft Word ile yapılmaktadır. Bu rapor projenin tüm ayrıntılarını içermektedir. Belge içeriği aşağıda listelenen 8 ana konudan oluşmaktadır.

- Giriş
- Proje Planı
- Sistem Çözümleme
- Sistem Tasarımı
- Sistem Gerçekleştirimi
- Doğrulama ve Geçerleme
- Bakım
- Sonuç

3.4.5 Eğitim Belgeleri

Çalışan personele sistemin bir parçası olabilmesi için eğitim verilmesi gerekmektedir. Bu eğitimin içeriği; sistemin işleyişi, personelin sistemdeki yeri, kullanacağı program hakkında bilgi verilmesi olacaktır.

3.4.6 Kullanıcı El Kitapları

Kullanıcı el kitabında sistemin tanıtımı, amacı ve kullanacağı uygulama hakkında bilgiler olacaktır.

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

Gerekenler:

Mobil uygulamalar(Android, iOS)

Web sitesi

Mobil uygulamalardan ve web sitesinden ulaşılabilecek bir sunucu

Her öğretmen ve öğrencinin kendine ait bir üyeliği olacak.

Kişisel bilgiler yer alacak.

Bu üyelikte önceden eklenebilen bir bakiye sistemi olacak.(Müşteri ödeme yaparken direk bakiyesinden düşebilecek).

Müşterinin uygulamada bir aracın doluluğunu görebilecek.

Müşteri uygulamadan belediye otobüslerinin ne kadar dolu olup olmadığını görebilecek.

Belediye araçları hatlar şeklinde ayrılacak. Örneğin 1.Hat Abdullah Paşa, 2.Hat Hilal Kent vb.

Mobil uygulamalar, Kod oluşturmaya dayalı olacak.

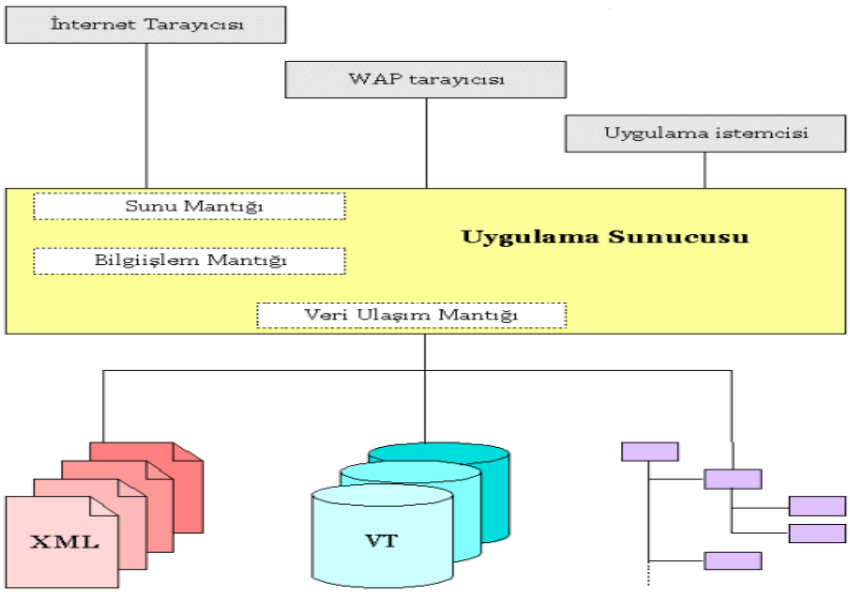
Bu uygulamalar veritabanıyla bağlantılı olacak. Müşteri belediye otobüsü içinde bulunan cihaza telefondaki Kodları oluşturarak öğrenci bilgilerini, devam durumlarını görebilecek.

4.1.2 Varsayımlar ve Kısıtlamalar

Öğrenci: Derse gelip yoklamaya imza atan kişi.

Öğretmen: Yoklama uygulamasını yönetecek olan kişi.

4.1.3 Sistem Mimarisi



5 katmandan oluşmaktadır;

İstemci Katı (Client Tier): Bu kat, geliştirilen uygulamaya ya da sisteme bağlanan diğer uygulamalar ya da cihazlardan oluşur. Örneğin: İnternet tarayıcısı, Java applet, WAP telefon...

Sunuş Katı (Presentation Tier): Bu kat, sistemin istemcileri için gerekli olan her türlü sunuş mantığını içinde bulundurur. Uygulamaya bağlanan istemcilerin taleplerini kaydeder, gerekli iş matığının uygulanmasını sağlar, talebin işlenmesi sonucu ortaya çıkan veriyi sunulur hale getirip istemciye cevap yollar. J2EE'yi oluşturan teknolojilerden ikisi JSP (JavaServer Pages) ve Java Servlet bu katta bulunur.

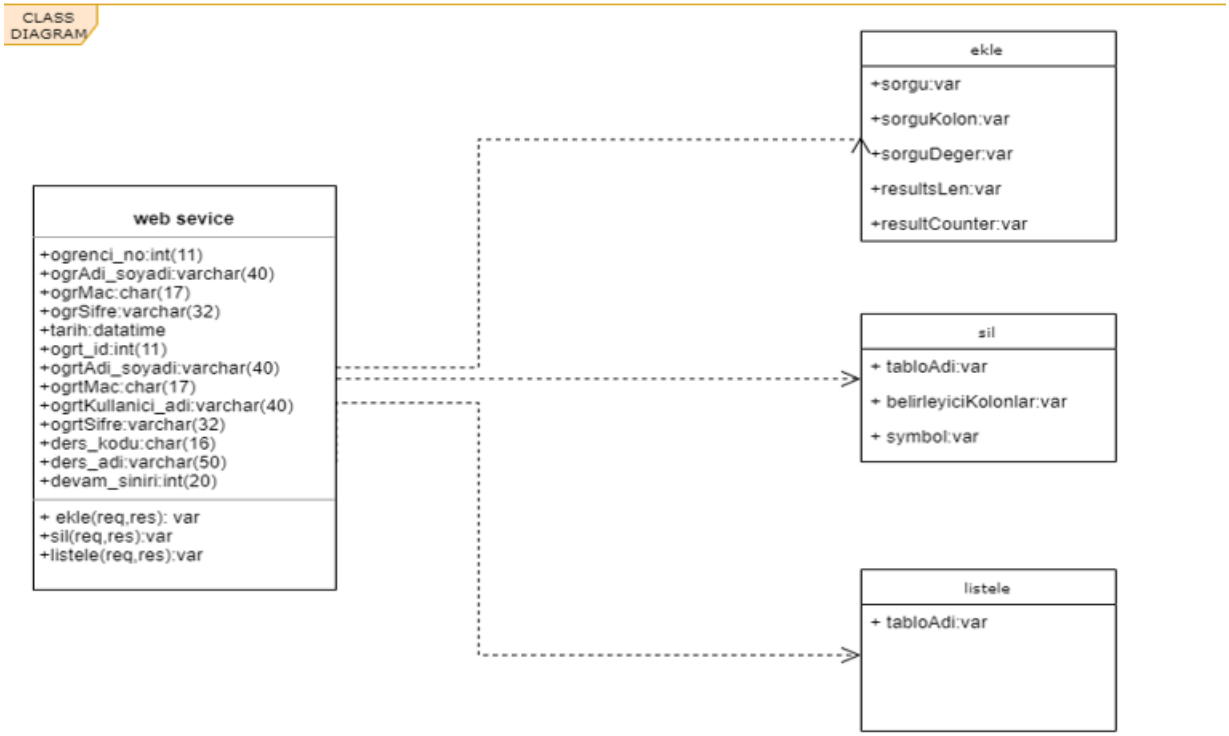
Uygulama ya da İş Katı (Application/Business Tier): Uygulamanın hedef aldığı ve gereklerini tatmin etmek için geliştirildiği işe dayalı tüm bilgiişlem bu katta toplanır. Bu görev, J2EE'yi oluşturan bir diğer teknoloji olan EJB (Enterprise JavaBeans)'ler tarafından sağlanır.

Entegrasyon Katı (Integration Tier): Bu kat, uygulamanın görevini yerine getirmesi için gerekli olan sistem dışı yazılımlara, sistemlere ya da veri tabanlarına bağlantıları sağlamakla yükümlüdür. J2EE uygulamalarının bu bölümleri, genelde, JDBC (Java DataBase Connectivity), J2EE Connector ya da bağlantı kurulan yazılımlara özel arayüzleri kullanırlar.

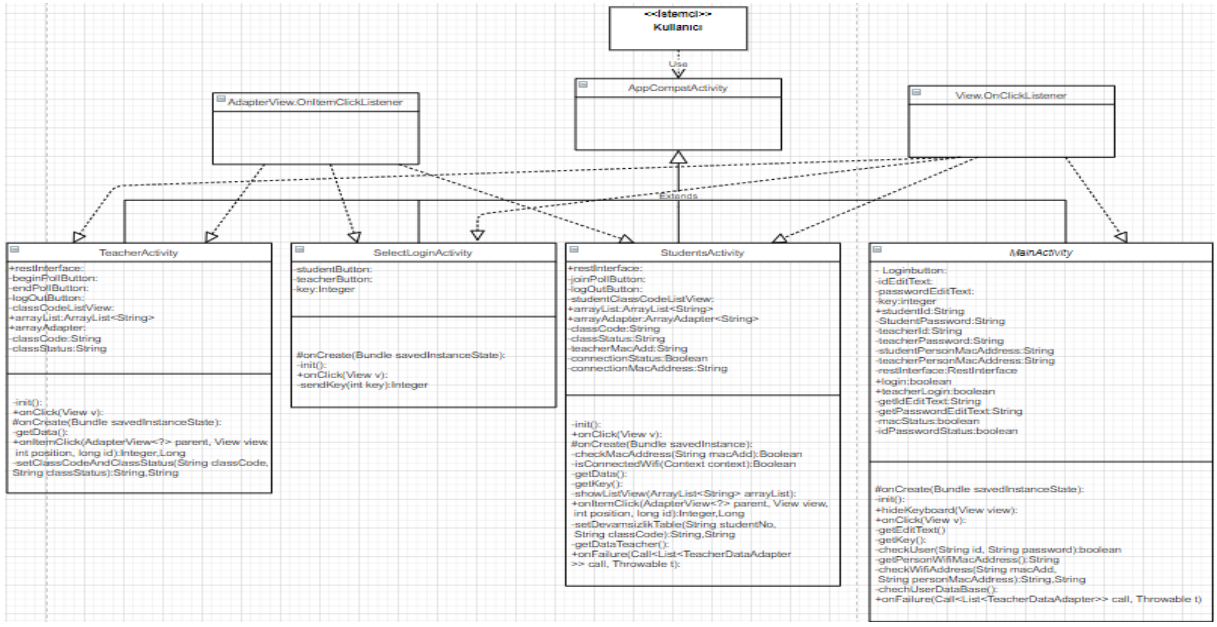
Kaynak Katı (Resource Tier): Bilgiişlem için gerekli veriler ve dış servisler bu katı oluşturur.

4.1.4 Veri Modeli

4.1.4.1 Veri Modeli (Web Servis)



4.1.4.2 Veri Modeli (Mobil Uygulama)



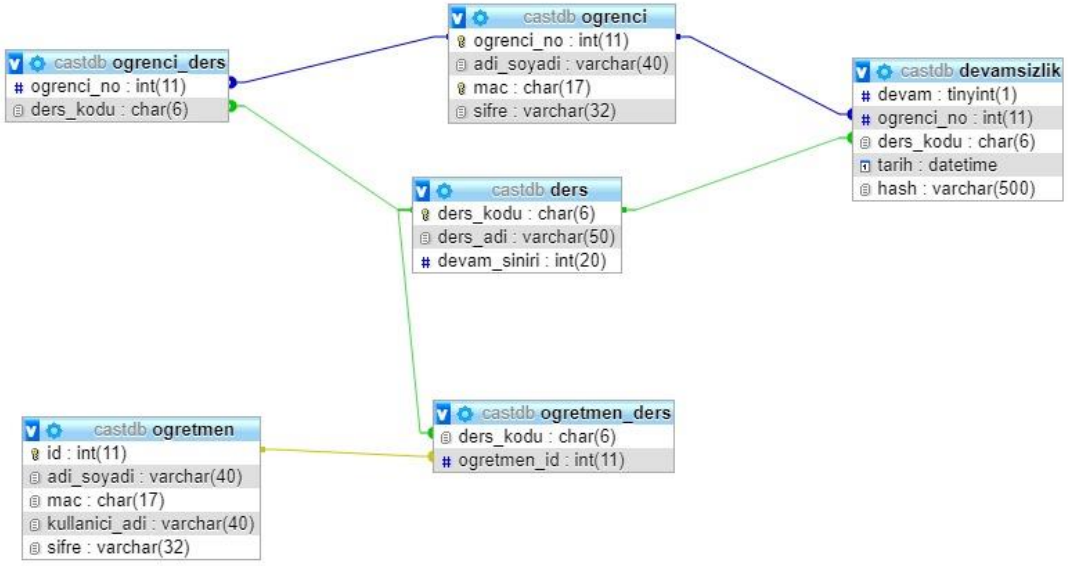
4.1.5 Testler

Yazılım Testi: Yazılım kodlama aşamasında programcı tarafından oluşabilecek hataları gidermek amacıyla acımasız bir şekilde yapılır.

Yeterlilik Testi: Yazılımın istenilen şekilde yapılıp yapılmadığını kontrol etmek amacıyla yapılır. Yani yazılım isterleri tam olarak karşılıyor mu sorusuna cevap olarak yapılır.

Sistem Testi: Yoğun veri akışı altında komple yükleme(load) testleri, normal olmayan koşullarda komple sistemin nasıl davranacağını görmek amacıyla germe(stres) testleri, istemli bir şekilde sistemi çökerterek sistemin nasıl davranacağını tespit etmek amacıyla geri kazanım(recovery) testleri, yazılımın geliştirilmesinde birimde yapay verilerle fabrika kabul testi, sistemin kullanılacağı yerde asıl verilerle kullanım hattı testleri, ve bundan sonra deneme testleri yapılır.

4.2 Veri Tasarımı



4.3 Süreç Tasarımı

4.3.4 Genel Tasarım

Hedef, yapı diyagramı çizmektir. Veri akış diyagramından yola çıkılarak işlem türlerinin bölgeleri tanımlanır ve bu bölgelere karşı düşecek yapısal öğeler ortaya çıkarılmış olur. İstenilen yapı diyagramı kontrol hiyerarşisini de göstermektedir.

4.3.5 Modüller

Kullanıcı Modülü: Kullanıcı olarak tanımlanan kişilerin ve yöneticinin erişebileceği ve kullanabileceği modül
Yönetici Modülü: Sadece yöneticinin ulaşabildiği tüm sistemin bağlı olduğu ve düzenleyebilinen modül
Öğrenci: Sistemi, ders bilgisi ve devam durumunu kontrol edecek kişinin modülü

4.3.6 Kullanıcı Profilleri

Sistemde 3 adet kullanıcı profili olacaktır.

- Öğrenci
- Yönetici
- Öğretmen

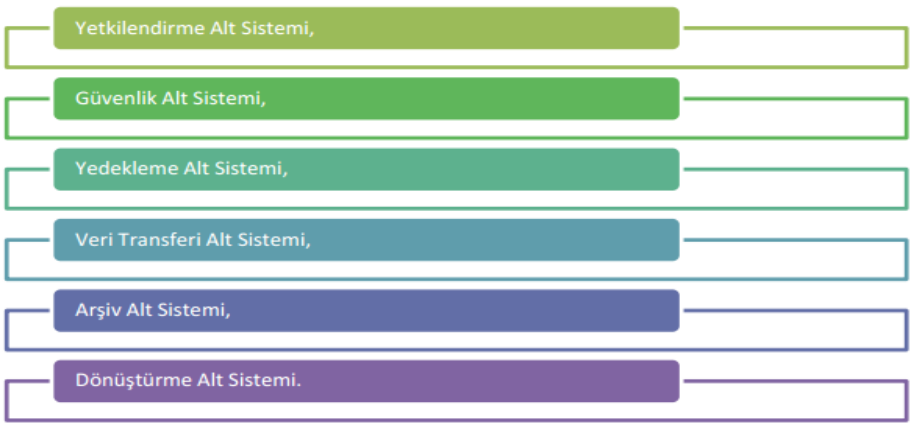
4.3.7 Entegrasyon ve Test Gereksinimleri

Sistemde 4 farklı program geliştirilecektir. Hepsinin ayrı ayrı ve birbirleriyle ilişkili şekilde test edilmesi gerekmektedir. Kullanıcı modülü için ayrı ayrı akıllı telefonlarda denenmelidir. Diğer programların tam verimli çalışıp çalışmadığı test ekibi tarafından detaylı bir şekilde test edilmelidir. Bütün modüller test edilip çalıştığına karar verildiğinde hepsinin veri tabanıyla kullanılarak test işlemi yapılmalıdır.

4.4 Ortak Alt Sistemlerin Tasarımı

4.4.4 Ortak Alt Sistemler

Herhangi bir bilgi sistemi tasarlanırken, hemen hemen tüm bilgi sistemlerinde ortak olarak bulunan bazı alt sistemlerin dikkate alınması gerekmektedir. Söz konusu alt sistemler:



4.4.5 Yetkilendirme Alt Sistemi

Özellikle kurumsal uygulamalarda farklı kullanıcıların kullanabilecekleri ve kullanamayacakları özellikleri ifade eder.

İşlev bazında yetkilendirme

Ekran bazında yetkilendirme

Ekran alanları bazında yetkilendirme

MySQL veri tabanına erişim konusunda yetkilendirme yapmaktadır.

4.4.6 Güvenlik Altsistemi

Yapılan bir işlemde, işlemi yapan kullanıcının izlerinin saklanması gerekmektedir. Bunlar LOG files(Sistem günlüğü) dosyalarında tutulmalıdır.

4.4.7 Yedekleme ve Arşivleme İşlemleri

Her bilgi sisteminin olağandışı durumlara hazırlıklı olmak amacıyla kullandıkları veri tabanı (sistem) yedekleme ve yedekten geri alma işlemlerinin olması gerekmektedir. Bu yüzden veri tabanı yedeklemesine önem verilmelidir. 4.4.1.4 Veri İletişim Alt Sistemi

Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleridir. Bu veri iletişimi 2 türlü sağlanmaktadır.

Çevrim içi veri iletimi (real time)

Çevirim dışı veri iletimi (disketler, teypler)

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. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

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

5.2 Yazılım Geliştirme Ortamları

5.2.4 Programlama Dilleri

Yazılım geliştirilirken pek çok farklı programlama dili kullanılmıştır. Bunlar;

- Java
- Android
- iOS
- JavaScript

Bu dilleri kısaca açıklamak gerekirse;

Java, Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu, nesneye yönelik, zeminden bağımsız, yüksek verimli, çok işlevli, yüksek seviye, adım adım işletilen (yorumlanan-interpreted) bir dildir.

Java, Sun Microsystems'den James Gosling tarafından geliştirilen bir programlama dilidir (Sun Microsystem'in şu anda Oracle Corporation ile bağlı ortaklığı bulunmaktadır) ve 1995 yılında Sun Microsystems'in çekirdek bileşeni olarak piyasaya sürülmüştür. Bu dil C ve C++'dan birçok sözdizim türetmesine rağmen bu türevler daha basit nesne modeli ve daha az düşük seviye olanaklar içerir. Java uygulamaları bilgisayar mimarisine bağlı olmadan herhangi bir Java Virtual Machine (JVM)'de çalışabilen tipik bytecode'dur (sınıf dosyası).

Android, Google, Open Handset Alliance ve özgür yazılım topluluğu tarafından geliştirilen, Linux tabanlı, mobil cihaz ve cep telefonları için geliştirilmekte olan, açık kaynak kodlu bir mobil işletim sistemidir.

Android, aygıtların fonksiyonelliğini genişleten uygulamalar yazan geniş bir geliştirici grubuna sahiptir. Android için halihazırda 250,000'den fazla uygulama bulunmaktadır. Google Play ise, Android işletim sistemi uygulamalarının çeşitli sitelerden indirilebilmesinin yanısıra, Google tarafından işletilen kurumsal uygulama mağazasıdır. Geliştiriciler, ilk olarak aygıtı, Google'ın Java kütüphanesi aracılığıyla kontrol ederek Java dilinde yazmışlardır.

iOS (eski adıyla iPhone OS) Apple'ın başlangıçta iPhone için geliştirdiği ancak daha sonra iPod Touch ve iPad'de de kullanılan mobil işletim sistemidir. Mac OS X'den türetilmiştir. iOS içinde 4

katman bulundurmaktadır: Core OS tabakası, Core Servisleri tabakası, Medya tabakası ve Cocoa Touch tabakası. Yazılım cihazın içinde 500 MB'lık bir alan kaplamaktadır.

JavaScript, 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.

5.2.5 Veri Tabanı Yönetim Sistemleri

Veri tabanı yönetim sistemi (**VTYS**, İngilizce: Database Management System, kısaca DBMS), veri tabanlarını tanımlamak, yaratmak, kullanmak, değiştirmek ve veri tabanı sistemleri ile ilgili her türlü işletimsel gereksinimleri karşılamak için tasarlanmış sistem ve yazılımdır.

5.2.5.2 VTYS Kullanımının Ek Yararları

Neden VTYS?

- Veri tutarlılığının sağlanması

Faydaları: Verilerin farklı tablolarda değişik değerler almasının önlenmesi

- Veri paylaşımının sağlanması

Faydaları: İnsan kaynaklarının ve donanımın verimli kullanılması

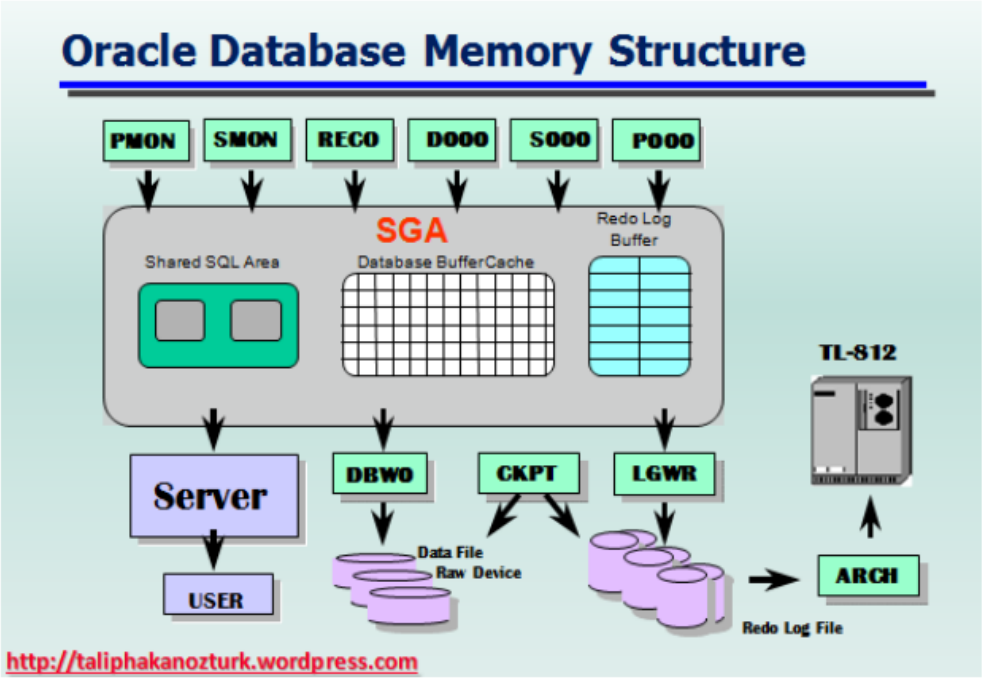
- Veri tekrarının azaltılması (data redundancy)

Faydaları: Donanım harcamalarının azalması, tutarsızlığın önlenmesi

- Verilerin güvenliğinin sağlanması (data security)

Faydaları: Yetki-sorumluluk bazında gerekli verileri görebilme, yetkisiz kişilerin sisteme girememesi, yedekleme hatadan kurtarma (recovery)

5.2.5.3 VTYS Mimarisi



Oracle Veri Tabanı Sistem Mimarisi

5.2.5.4 Veritabanı Dilleri ve Arabirimleri

MySQL, Windows, Unix ve OS/2 gibi sistemlerde kullanılan açık kaynak kodlu bir veritabanı sistemidir. Kullanıcıların ücretsiz şekilde faydalanabileceği bu sistemi ticari amaçlar için kullanmak isteyenler ise belirli bir ücret karşılığında hizmetten yararlanabiliyorlar. Örnek verecek olursak, bir wordpress blogu oluşturduğunuzda yarattığınız veritabanı sayesinde bloga eklediğiniz her türlü içerik güvenli bir şekilde burada saklanır. İçerik ekledikçe, kayıt işlemi de otomatikman devam eder.

Daha çok '.php' uzantılı internet sayfalarında, hazır olarak yer alan scriptler üzerinden kullanıldığı için Unix sistemlerde daha yaygın şekilde kullanılır.

5.2.5.5 Veri Tabanı Sistem Ortamı

VTYS Mimarisi İçsel Yüzey, Kavramsal Yüzey, Dışsal Yüzey olarak ayrılır.

Kısaca bu yüzeylerde hedeflenen amaçlar;

5.2.5.6 VTYS'nin Sınıflandırılması

VTYS olarak ilişkisel veri modeli kullanılmıştır. İlişkisel veri tabanının kullanım nedeni birbiriyle alakalı aynı türden verilerin kullanılacağı için ayrıca projede kullanılacak en iyi alt yapı olduğu için ilişkisel veri tabanı seçilmiştir. Veri Tabanı Yöneticisinin Görevleri:

• Veriler üzerinde yapılacak uygulama gereksinim-lerini belirlemek, veri tabanı içeriğini oluşturmak, veri tabanı şemalarını (tabloları) tanımlamak.

• Bütünlük kısıtlamalarını (Primary Key, Foreign Key, Unique, Check, Not Null) belirleyip tanımlamak.

• Veri tabanı kullanıcılarını ve her kullanıcının hangi veriler üzerinde hangi işlemleri yapmaya yetkili olduğunu belirlemek; kullanıcı ve kullanım yetkilerini tanımlamak.

• Veri Tabanı Yönetim Sisteminin sunduğu seçenekler çerçevesinde, veri tabanının fiziksel yapısı ile ilgili parametreleri ve erişim yollarını (dizinleri) belirlemek ve tanımlamak.

• Yedekleme, yeniden başlatma ve kurtarma düzenlerini belirlemek.

• Veri tabanı sistemini sahiplenmek, işletimini izlemek, veri tabanının sürekli olarak kullanıma açık olmasını sağlamak.

• Gereksinimlerdeki değişiklikleri izlemek ve değişikliklere paralel olarak veri tabanı içeriği, şema tanımları, bütünlük kısıtlamaları, fiziksel yapı ile ilgili parametreler, erişim yolları, kullanıcılar ve kullanıcı yetkilerinde gerekli değişiklikleri oluşturmak ve tanımlamak.

• Veri tabanı bütünlük kısıtlamalarının yeterliliğini izlemek; bütünlük kısıtlamaları ile ilgili gerekli değişiklikleri oluşturmak ve tasarlamak.

• Veri tabanı kullanım istatistiklerini ve veri tabanı başarımını izlemek; varsa sorunları ve yetersizlikleri belirlemek ve gerekli her türlü önlemi almak.

5.2.5.7 CASE Araç ve Ortamları

Microsoft Project: Gantt Diyagramı çizilmesinde faydalanılmıştır.

Microsoft Word: Dokümantasyonun hazırlanmasında faydalanılmıştır.

ArgoUML: Diyagramların çiziminde faydalanılmıştır.

Adobe Photoshop CS5: belirli resimlerin çizilmesinde, düzenlenmesinde faydalanılmıştır.

Microsoft Visio: Kapsam diyagramı ve örgüt yapısı çiziminde yararlanılmıştır.

Smart Draw: Use Case Diyagramları hazırlanırken faydalanılmıştır.

5.3 Kodlama Stili

5.3.4 Açıklama Satırları

Bir yazılım geliştirirken kodların tekrar kullanılabilmesi, başkaları tarafından anlaşılabilmesi için kodlara açıklama satırları koyulur. Bunlar genel olarak

Bir paragraf şeklindeyse;

/*Açıklama

Açıklama

Açıklama

*/

Tek bir satır ise;

//Açıklama

Şeklinde ifade edilir.

5.3.5 Kod Biçimlemesi

Kod biçimlenmesi açıklama satırlarına olan ihtiyacı azaltır. Kod biçimlemesinde önemli olan az satır değil kodun okunabilirliğidir. Bu projede bu kriterler göze alınmalıdır.

5.3.6 Anlamlı İsimlendirme

Kodların okunabilirliğini ve anlaşılabilirliğini sağlayan önemli unsurlardan biri de kullanılan ve kullanıcı tarafından belirlenen belirteçlerin (Değişken adları, kütük adları, Veri tabanı tablo adları, işlev adları, yordam adları vb) anlamlı olarak isimlendirilmesidir.

5.3.7 Yapısal Programlama Yapıları

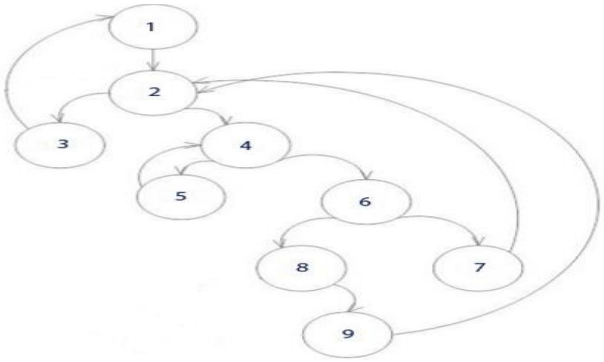
Program kodlarının, okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve perogram karmaşıklığının azaltılması amacıyla "yapısal programlama yapıları" kullanılarak yazılması önemlidir. Yapısal Programlama Yapıları, temelde, içinde "go to" deyimi bulunmayan, "tek giriş ve tek çıkışlı" öbeklerden oluşan yapılardır. Teorik olarak herhangi bir bilgisayar programının, yalnızca Yapısal Programlama Yapıları kullanılarak yazılabileceği kanıtlanmıştır.

Üç temel Yapısal Programlama Yapısı bulunmaktadır:

- Ardışıl işlem yapıları
- Koşullu işlem yapıları
- Döngü yapıları

5.4 Program Karmaşıklığı

5.4.4 Programın Çizge Biçimine Dönüştürülmesi



McCabe Çizgi Biçimi

5.4.5 McCabe Karmaşıklık Ölçütü Hesaplama

k = 12 Kenar sayısı

d = 9 Düğüm sayısı

p = 1 Bileşen sayısı

$$V(G)= k – d + 2p$$

$$V(G)=12 - 9 + 2 \times 1 = 5$$

5.5 Olağan Dışı Durum Çözümleme

Olağan dışı durumlar gerek kod yazım sürecinde gerekse testler sırasında gerçekleşebilir. Projede Scrum Model kullanıldığından dolayı her aşamada test yapılacağından olağan dışı durum anında çözülebilecektir.

5.6 Kod Gözden Geçirme

5.6.4 Gözden Geçirme Sürecinin Düzenlenmesi

Gözden geçirme sürecinin temel özellikleri;

Hataların bulunması, ancak düzeltilmemesi hedeflenir,

Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyici kullanılmasıdır. Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımı yapacak ekipten seçilmesinde yarar vardır.

Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir. Gözden Geçirme çalışmasının olası çıktıları biçiminde özetlenebilir. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir.

5.6.5 Gözden Geçirme Sırasında Kullanılacak Sorular

Yazılım geliştirme ekibinin geliştirdiği kodu gözden geçirmek için bir checklist kullanmak, bu sürecin bir parçasıdır ve uzmanlarca tavsiye edilir. Herhangi bir kod commit edilmeden önce aşağıdaki gibi bir liste ile check edilebilir:

1. Kod doğru bir şekilde build edildi mi? Kaynak kod derlendiğinde hata olmamalı. Kodda yapılan değişikliklerde uyarılar(warning) olmamalı.

2. Kod çalıştırıldığında beklendiği gibi davrandı mı?

3. Kodun sadece çalışırlığına bakılmamalı, kod tasarımı da göz önün de bulundurulmalıdır. Optimizasyon için öneriler takım olarak değerlendirilmelidir.

4. Gözden geçirilen kod anlaşılıyor mu? Gözden geçiricinin kodu anlaması gerekir. Eğer anlaşılmadıysa, gözden geçirme tamamlanmış olmaz veya kod iyi yorumlanabilmiş sayılmaz.

5. Geleneksel kodlama standartlarına uyuldu mu? Değişken isimlendirme, satır başı boşluklar, parantez stilleri vs. takip edilmeli.

6. Telif hakkı bilgisi ve uygun bir başlıkla başlayan kaynak dosya var mı? Her bir kaynak dosyası bu bilgilerle başlamalı, bütün kaynak dosyaları, fonksiyonelliğini anlatan bir dosya içermelidir.

7. Değişken deklarasyonlarına yorum satırları eklenmiş mi? Yorumlar, değişkenlerin görevlerini açıklaması gerekir. Özellikle her bir global değişkenin amacı ve neden global olarak tanımlandığı belirtilmelidir.

8. Sayısal verilerin birimleri açıkça belirtilmiş mi? Sayısal verilerin birimleri yorum satırı olarak belirtilmeli. Örneğin, eğer bir sayı uzunluğu temsil ediyorsa, metre mi feet mi olduğu gösterilmelidir.

9. Bütün fonksiyonlar, metotlar ve classlar dokümante edilmiş mi? Her bir fonksiyon, metot ve class'ın tanımlanmasının üstünde bir iki cümle ile açıklaması yer almalıdır. Amacı vurgulamalı ve tasarım gerekliliklerini işaret etmelidir.

10. Fonksiyonların kullandığı input ve output parametreleri açıkça tanımlandı mı?

11. Karmaşık algoritmalar ve kod optimizasyonları yeterli olacak şekilde açıklanmış mı? Karmaşık alanlar, algoritmalar ve kod optimizasyonları için yeterince yorum satırı eklenmelidir. Öyle ki, diğer geliştiriciler kodu anlayabilmeli ve kalınan yerden devam ettirebilmelidir.

12. Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar var mı? Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar olmalı. "Ölü Kod"lar çıkarılmalı. Eğer geçici bir kod bloğu ise neden tanımlandığı belirtilmeli.

13. Koddaki eksik işlevsellikler veya çözümlenmemiş sorunlar yorum satırlarında ifade edilmiş mi? Bu ifadeler eksikleri ve yapılacakları açıklamalıdır. Sonradan arandığında bulunabilmesi için de ayırıcı bir işaretleyici kullanılmalı, örneğin TODO.

14. Her zaman bir fonksiyonun döndürebileceği hatalar düzün bir şekilde handle edilmeli. Fonksiyonun üreteceği her bir sonuç düşünülmeli, her durum kontrol edilmeli ve kodun kalan kısmının yürütülmesini etkileyen hatalar yakalanmış olmalıdır.

15. Alınan hatalardan sonra tüm kaynaklar ve hafıza temizlenip serbest bırakılıyor mu? Bundan emin olunmalı. Bir hata meydana geldiğinde, dosya, soket ve veritabanı bağlantı objeleri gibi tüm objeler dispose edilmeli.

16. Exception'lar uygun bir şekilde yakalanıyor mu? Eğer fırlatılan exception kodun devamında kullanılıyorsa, bu fonksiyon devamında düzgün bir şekilde handle edilmeli veya yakalanmalı.

17. Tüm global değişkenler thread-safe olmalı. Eğer global değişkenlere birden fazla thread ile erişiliyorsa, kodun çalışmasını değiştirebilir veya sekronizasyon mekanizmasını engelleyebilir. Yine benzer bir şekilde olarak bir veya daha fazla thread ile erişilen objeler varsa, üyeler korunmalıdır.

18. Tespit edilen hata kodun başka yerlerini de etkiliyor mu, kontrol edilmeli? Hatanın tüm ekranlarda giderildiğinden emin olunmalı.

19. Kodun değişiklik yapılan yerlerinde, eski halini ve neden yapıldığını mutlaka açıklama olarak eklenmelidir.

20. Kodda yapılmış yorumlar değerlendirilmeli, eğer yorumun uygun/doğru olmadığı düşünülüyorsa geliştirici ile görüşülmeli ve konu tartışıldıktan sonra çözüme kavuşturulmalıdır.

5.6.5.2 Öbek Arayüzü

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

5.6.5.3 Giriş Açıklamaları

- 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 açıklamaları var mı?

5.6.5.4 Veri Kullanımı

- 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?

5.6.5.5 Öbeğin Düzenlenişi

- Algoritmalar istenen işlevleri karşılıyor mu?
- Ara yüzler genel tasarımla uyumlu mu?
- Mantıksal karmaşıklık anlamlı mı?
- Veri yapısı çözümleme çalışması sırasında elde edilen veri modeli ile uyumlu mu?
- Belirlenen tasarım standartlarına uyulmuş mu?
- Hata çözümleme tanımlanmış mı?
- Tasarım, kullanılacak programlama diline uygun mu?
- İşlerim sistemi ve programlama diline yönelik kısıtlar ya da özellikler kullanılmış mı?
- Bakım dikkate alınmış mı?

5.6.5.6 Sunuş

- 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

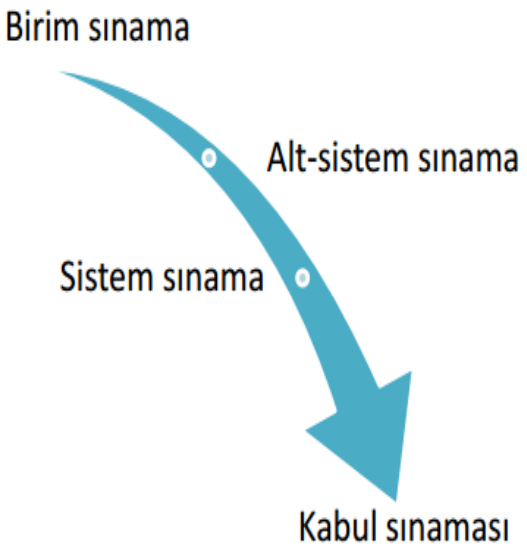
6.1 Giriş

Doğrulama, yazılımın yaşam döngüsü boyunca her aşamada bir önceki aşamadaki gereksinimlere uygunluğunu denetleme işlemidir. Geçerleme ise geliştirme işleminin sonunda yazılımın gereksinimlere uygunluğunu, yani kendinden beklenenleri karşılayıp karşılamadığını test etme işlemidir. D&G, yazılımın istenen görevleri doğru şekilde yerine getirip getirmediğini belirlemek, istenmeyen herhangi bir işlem yapmadığından emin olmak ve kalite ve güvenilirliğini ölçmek amacıyla yazılımı kapsamlı bir şekilde test eder.

6.2 Sınama Kavramları

Sınama ve Bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerinin seçilmesi gerekmektedir.

Sınama işlemleri dört ana sınıfta incelenebilir:



Birim Sınama:

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

Alt-Sistem Sınama:

Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar.

Yine bağımsız olarak sınamaları yapılmalıdır.

Bu aşamada en çok hata arayüzlerde bulunmaktadır. Bu yüzden arayüz hatalarına doğru yoğunlaşılmalıdır.

Sistem Sınaması:

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

Ayrıca, belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

Kabul Sınaması:

Çalıştırılmadan önce sistemin son sınamasıdır.

Artık, yapay veriler yerine gerçek veriler kullanılır.

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

6.3 Doğrulama ve Geçerleme Yaşam Döngüsü

Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır.

Planlama genellikle; Alt-sistem, bütünleştirme, sistem ve kabul sınamalarının tasarımlarını içerir. Gerçekleştirim aşamasının sonunda ise söz konusu plan uygulanır.

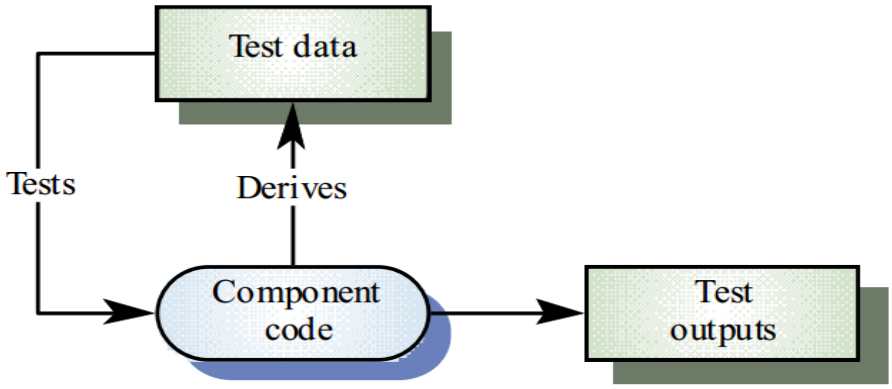
6.4 Sınama Yöntemleri

Her yazılım Mühendisliği ürünü iki yoldan sınanır:

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

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

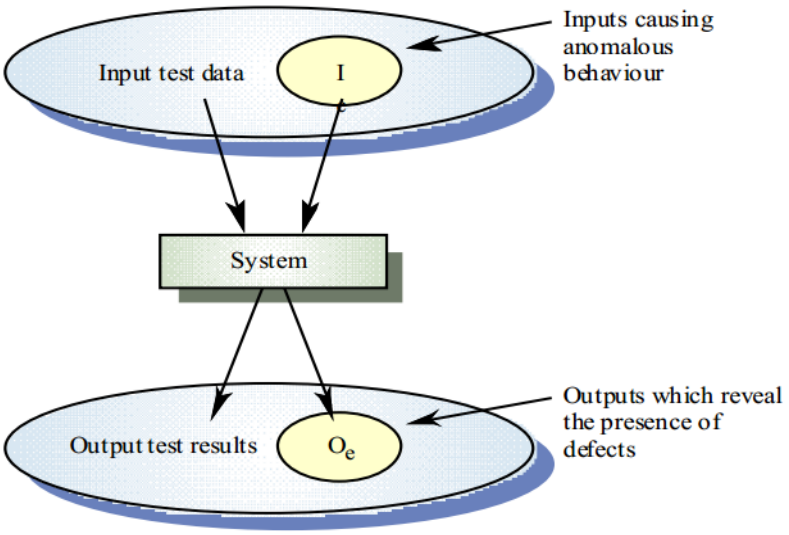
6.4.4 Beyaz Kutu Sınaması



Beyaz Kutu Sınaması

- Bütün bağımsız yolların en az bir kez sınanması gerekir.
- Bütün mantıksal karar noktalarında iki değişik karar için sınamalar yapılır.
- Bütün döngülerin sınır değerlerinde sınanması
- İç veri yapılarının denenmesi

6.4.5 Kara Kutu Sınaması



Kara Kutu Sınaması

Ürünlerin test edilmesi sırasında kullanılan en ilkel test metodudur. Bir takım test senaryolarının seçilip, yazılım kodundan bağımsız olarak takip edilmesi temeline dayanmaktadır. Bu yüzden ürünlerin fonksiyonel durumları ve inputlara verdikleri tepkilerin gözlenmesi uygulamada kullanılan kara kutu testlerinin kapsamını oluşturmaktadır. Bu noktada, yazılımın kodunda

yapılan herhangi bir değişiklik veya data yapısındaki uyarlamalar kara kutu testleriyle kontrol edilen özellikler değildir.

Test edilecek olan uygulamanın kodu hiç dikkate alınmadan, sadece girdilerin ve çıktıların incelenmesi ile gerçekleştirilen test metodudur. 5 ayrı tekniği bilinir. Denklik sınıfı test tekniği, test verileri gruplanır. Gruplar içinde testler yapılır.

1. Uç nokta test tekniği, hataların genelde sınırlarda çıktığı varsayılarak sınır değerlerinde test yapılır.

2. Karar tablosu test tekniği, çok fazla test yapılması gereken uygulamalarda verilerin matrix haline getirilerek test edilmesi test edilmesidir.

3. Sistem durumu test tekniği, farklı durum geçişleri yer alan sistemlerin testleridir.

4. İş senaryosu test tekniği, use case dokümanlarının kullanıldığı test tekniğidir.

6.5 Sınama ve Bütünleştirme Stratejileri

6.5.4 Yukarıdan Aşağı Sınama ve Bütünleştirme

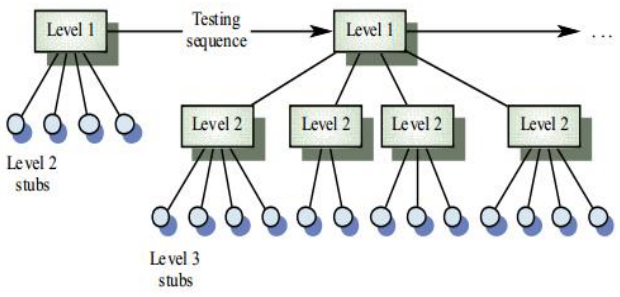
Yukarıdan-aşağıya bütünleştirmede önce sistemin üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeylere ilgili modülleri takılarak sınanması söz konusudur.

En üst noktadaki bileşen sınandıktan sonra alt düzeye geçilmelidir.

Alt bileşenler henüz hazırlanmamışlardır. Bu sebeple Koçanlar kullanılır. Koçan: Bir alt bileşenin, üst bileşen ile ara yüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.

İki temel yaklaşım vardır:

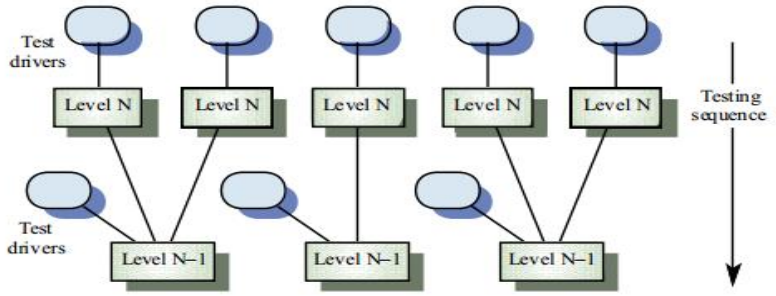
- *Düzey Öncelikli Bütünleştirme:* En üst düzeyden başlanır ve aynı düzeydeki birimler bütünleştirilir.
- *Derinlik Öncelikli Bütünleştirme:* En üst düzeyden başlanır ve her dal soldan sağa olmak üzere ele alınır.



Yukarıdan Aşağıya Sınama

6.5.5 Aşağıdan Yukarıya Sınama ve Bütünleştirme

Önceki yöntemin tersine uygulama yapılır. Önce en alt düzeydeki işçi birimler sınanır ve bir üst düzey ile sınanması gerektiğinde bu düzey bir sürücü ile temsil edilir. Bu kez kodlama, bütünleştirme ve sınama, aşağı düzeylerden yukarı düzeylere doğru gelişir.



Aşağıdan Yukarıya Sınama

6.6 Sınama Planlaması

Her sınama planı, sınama etkinliklerinin sınırlarını, yaklaşımını, kaynaklarını ve zamanlamasını tanımlar. Plan neyin sınanacağını, neyin sınanmayacağını, sorumlu kişileri ve riskleri göstermektedir. Sınama planları, sınama belirtimlerini içerir.

6.7 Sınama Belirtimleri

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

Bu ayrıtılar temel olarak:

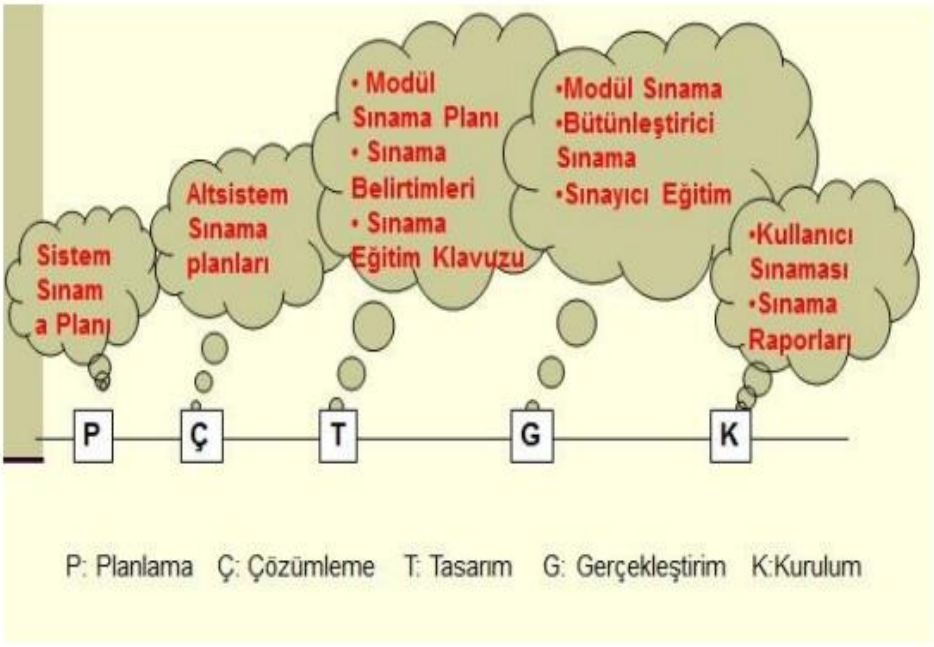
- ✓ Sınanan program modülü ya da modüllerinin adları,
- ✓ Sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
- ✓ Sınama verileri,
- ✓ Sınama senaryoları türündeki bilgileri içerir.

Sınama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir.

Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtimlerinin hazırlanmasındaki temel maç, etkin sınama yapılması için bir rehber oluşturmasıdır. Sınama işlemi sonrasında bu belirtimlere,

- ✓ Sınamayı yapan,
- ✓ Sınama tarihi,
- ✓ Bulunan hatalar ve açıklamaları türündeki bilgiler eklenerek sınama raporları oluşturulur.

6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri



Sınama Etkinlikleri

7 BAKIM

7.1 Giriş

Yazılımın dağıtılması ve kullanıma başlanmasından sonra yazılımda yapılacak değişiklikler yazılımın bakımı (software maintenance) olarak adlandırılır. Bu değişiklikler basit kodlama hatalarının düzeltilmesi (bug-fixes) şeklinde olabileceği gibi tasarımdan kaynaklanan hataların giderilmesi gibi daha kapsamlı değişiklikler şeklinde de olabilir. Yazılımın bakımı aslında yazılımın evrimleşmesidir. Yazılımın yaşamına devam edebilmesi için gerekli değişikliklerin uygulanmasıdır.

7.2 Yerinde Destek Organizasyonu

Yerinde destek ekibi, kullanıcı alanında yerleşik olarak bulunan gerekli sayıda elemandan oluşan bir ekiptir.

Bu ekibin temel görevleri:

- Kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak,
- Giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasındaki uygulama yazılımı destek ekibine iletmek,
- Kullanıcıya işbaşında uygulama eğitimi vermek,
- Kullanıcı sınama günlüklerini toplamak
- Yapılan tüm işlemleri konfigürasyon veri tabanına kaydetmek biçimindedir.

7.3 Yazılım Bakımı

Yazılım bakımı yapılırken şu programlardan faydalanılacaktır.

Atlassian Jira Talep Takip (Issue Tracking) Yazılımı – Değişiklik taleplerinin girilerek yapılabilirlik analizinin başlatılması, değişiklik yönetim süreçlerinin izlenmesi ve proje ekibi üzerindeki görevlerin takibi için kullanılacak web tabanlı yazılım aracıdır.

Atlassian Confluence (Wiki) Yazılımı – Değişiklik ve Yapılandırma yönetim süreçlerindeki tüm dokümantasyonların hazırlanması, saklanması ve erişilmesi için kullanılacak web tabanlı yazılım aracıdır.

Atlassian Fisheye + Crucible Yazılımı – Kaynak kod deposu üzerinde gezinmek, kaynak kod dosyalarını görüntülemek ve sürümleri arasındaki değişiklikleri izlemek için Fisheye, proje ekibindeki yazılım geliştiricilerin yapacakları değişiklikleri gözden geçirmek, yorumlamak ve gerekirse yönlendirmek amacı ile Crucible yazılımı kullanılmaktadır. Her iki yazılım da web tabanlıdır.

8 SONUÇ

Sonuç olarak yaptığımız bu uygulama sayesinde öğretmen derse gelmeden kod oluşturup sınıf listesini o koda atayabilecek buda öğretmene büyük kolaylık sağlayacak. Yoklama kâğıdının yerini alacak olan bu sistem sayesinde sınıfta ki öğrenciler gelmeyen diğer öğrencilerin yerine imza atamayacak. Öğrenciler sisteme girdiklerinde devam durumunu da görebileceklerdir.

9 KAYNAKLAR

http://www.bilgisite.com/yonetim/kaliteyonetim/yonetim_09.html

http://univera-ng.blogspot.com

http://www.bidb.itu.edu.tr/?d=1064

http://www.godoro.com/Divisions/Ehil/Mahzen/Java/J2EEEmpire/txt/html/document_SystemArchitectural.html

http://www.yarbis.yildiz.edu.tr/

http://tr.wikipedia.org/wiki/Ana_Sayfa

http://www.tutev.org.tr

http://taliphakanozturk.wordpress.com

http://www.yazilimprojesi.com

http://technet.microsoft.com

http://w3.gazi.edu.tr/~nyalcin/CASE.pdf

http://cse.cbu.edu.tr

http://www.kriptarium.com/ymt.html

https://bidb.itu.edu.tr/

http://koddit.com/yazilim

http://www.869tr.com/rfidteknolojisi.html

http://www.google.com.tr

ARİFOĞLU, Ali; Yazılım Mühendisliğine Giriş

KALIPSIZ, OYA; Yazılım Mühendisliği

ÇÖLKESEN, Rıfat, “Veri Yapıları ve Algoritmalar.

SARIDOĞAN, Dr. Erhan. ,” Yazılım Mühendisliği Temelleri