1. **PROJENİN AMACI**

Günümüz teknolojilerini gelişmesiyle birlikte, önceden kullanılan ancak yavaş yavaş kullanılabilirliğini ve popülaritesini yitiren birçok araç ortaya çıkmıştır. Bunlardan biride gazetelerdir. Günümüzde gazetelerin yerini daha çok mobil ve internet haber uygulamaları yerini almıştır.

Mobil ve internet haber uygulamasının fazlalığı, bir haber sitesinden başka bir haber sitesine geçme, bir konudaki farklı görüş fikirlerini öğrenme kullanıcıların farklı haber sitesine yönlendiren etmenlerdendir.

Bu projedeki amaç belli başlı haber kaynaklarını tek bir çatı altında buluşturarak, kullanıcıların tek bir site veya uygulama üzerinden farklı haber kaynaklarına ulaşabilmelerini sağlamaktadır.

Projedeki bir başka amaç ise kendi sosyal platformunu oluşturmaktır. Yani proje hem sosyal bir ağ ortamı sağlayacak hem de farklı haber sitelerindeki haberleri kullanıcıya sunacaktır.

* 1. **Projede düşünülen özellikler**
* Tek bir uygulama ile birden fazla kaynaktaki haber verilerine ulaşmak,
* Yayınlanan haberlere yorum yapılabilmesi,
* Farklı sosyal uygulamaların yine aynı uygulama içerisinde bulunulması,
* Haber linklerini hızlı paylaşma özelliği,
* Kayıtlı kullanıcıların arası mesaj atması,
* Kayıtlı kullanıcıların haberleri veya arz ettiği şeyleri paylaşabilme özelliği.

1. **YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ**

Geliştirilen her bir yazılım projesinin planlamasından başlayarak teslimatına kadar geçirmiş olduğu bütün aşamalara ve bu aşamalardan oluşan döngüye, Yazılım Geliştirme Yaşam Döngüsü denir.

Bir yazılım projesinin geliştirilmesi, sadece kodlamadan oluşmamaktadır. Basitçe bir proje geliştirilirken projenin planlama, analiz, tasarım, üretim ve test aşamaları yer almaktadır. Bu aşamalar bir kere gerçekleştirildikten sonra proje tamamlanmayabilir. Bu aşamaların bir döngü halinde düşünülmesi gerekmektedir. Proje tamamlandıktan sonra gelecek istekler, hata düzeltmeleri, projeye eklenecek yeni modüller vs. konular için bu süreç devam etmektedir. Bu döngüye yazılım geliştirme yaşam döngüsü adı verilmektedir.

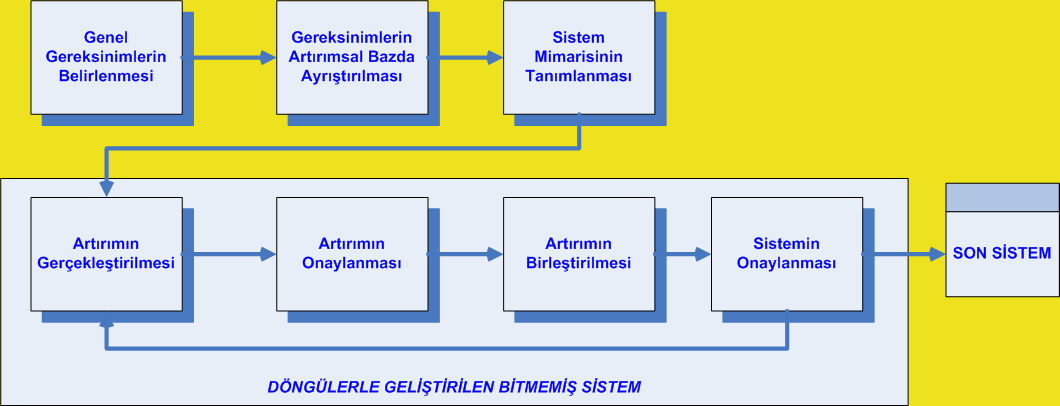
Bizim oluşturacağımız projede bu adımları Artırımsal Süreç Modeli kullanarak gerçekleştireceğiz.

1. **PROJEDE KULLANILAN YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ**
   1. **Artırımsal Süreç Modeli**

Sistemi tek seferde teslim etmek yerine, geliştirme ve teslim parçalara bölünür. Her teslim beklenen işlevselliğin bir parçasını karşılar. Bir parçanın geliştirmesi başladığında, gereksinimleri dondurulur. Olası değişiklikler sonraki teslimlerde ele alınır.

**3.1.1 Modelin üstün özellikleri:**

* Proje birçok döngünü gerçekleştirilmesi ile ilerler.
* Müşteriye sunulan ara yazılımlar ile yeni bir fonksiyon eklenebilir veya var olan bir özellik kaldırılabilir.
* Her bir ara yazılımda kullanıcının istekleri daha düzgün tahlil edilmiş olur ve yerine getirilmiş olunur.
* Kullanıcılar, yazılım geliştirilirken, yazılımı kullanıldığından daha az direnç ile karşılaşılır.
* Yazılımın birbiriyle ilgisiz parçaları bağımsız geliştirildiğinden daha hızlı yazılım geliştirilmiş olur.
* Oluşan bu ara yazılımların başarısı, nihai yazılımların başarısını da arttırır.

****

Şekil 3.1 Artırımsal süreç modeli şeması

**3.2 Modele Göre Projenin Gerçekleştirilmesi**

**3.2.1.Planlama**

Proje mobil ve internet uygulaması şeklinde iki farklı platformda gerçekleştirilecektir.

Uygulamanın mobil ayağında kullanılacak teknolojiler ve yapılar şu şekilde olacaktır:

* BaseAdapter
* WebView API
* SQLite Database
* Fragment yapısı

Projenin mobilde çalışabilmesi için internet kullanımına ihtiyaç vardır. Bu ihtiyacı karşılamak için “android.permission.INTERNET” ve “android.permission.ACCESS\_NETWORK\_STATE” izinlerini istemek gerekiyor.

Mobil uygulama için kullanılan SQLite3 veri tabanı mobil hafızaya kaydedilerek tutulmaktadır.

Projenin internet kısmı ise PHP dili ile oluşturulacak. Proje inovatif bir yapıda olacağı için MVC projesi şeklinde gerçekleştirilecek.

**Model Yapısı:**

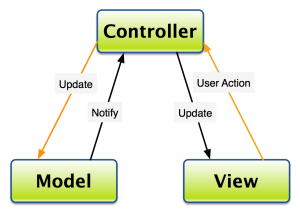
Uygulamada kullanılan veriler ve verilerin işlenme mantığının saklandığı kısım olacaktır. Genel yapısı itibari ile veri tabanı kısmıdır

**View Yapısı:**

View yapısı MVC’de genel olarak arayüzü ifade ettiği için css ve ajax kullanımının yoğun olacaktır.

**Controller Yapısı:**

Model ile View arasında işlemleri yapan kısımdır. Burdai arka planda çalışacak kodlar bulunacaktır.



İnternet uygulaması için kullanılacak veri tabanı ise firebase ile MySQL olacaktır.

MySQL ile sayfa bilgileri ile kullanıcı bilgileri tutulacaktır. Firebase ile ise sayfaya yapılan yorumlar ve paylaşımlar tutulacaktır.

**3.2.2 Yazılımın Ekibi**

Projenin yazılım ekibi için;

* 1 Proje Yöneticisi,
* 1 Analist,
* 2 Programcı,
* 1 Tasarımcı
* 1 Veri Tabanı Uzmanı

Olmak üzere toplamda 6 kişi görev alacaktır. Görev dağılımı ise şu şekilde olacaktır.

Proje Yöneticisi:

Projeye atanan proje yöneticisi bütünlüğünü sağlamak ve projenin en önemli 3 ayağı olan kapsam, zaman ve maliyet bilgi alanlarını devamlı olarak izlemeli ve buradaki değişimleri rapor etmelidir.

Analist:

Geliştirilebilir proje için kullanıcı isteklerini belgelemeli, gerekli olan iş mimarisini tasarlamalı ve geliştirme kontrolünü ve kalitesini sağlamalıdır.

Programcı:

Mobil ve Web olmak üzere iki farklı programcı çalışacaktır. Mobil programcı, mobil uygulama geliştirmede yetkin olmalı, gerekli bağlantıları sağlayabilmeli ve algoritmayı oluşturabilmelidir. Aynı durum web programcı için de geçerlidir. Web programlamada özellikle PHP yetkin olmalıdır.

**3.2.3 Proje Maliyeti**

**3.2.3 Analiz**

Günümüzde akıllı telefon ve internet kullanımı her geçen gün daha da artmaktadır. Bu verileri rakamlarla ifade etmek gerekirse, Digital’in 2016 raporuna göre;



* 7.395 Milyar olan Dünya nüfusundan internete bağlanan insan sayısı 3.419 milyar.
* Dünyada, sosyal medyayı aktif olarak kullanan kişi sayısı 2.307 milyar.
* Dünyada mobil cihaz kullanıcısı sayısı 3.790 milyar.
* Mobil cihaz üzerinden sosyal medyayı kullananların sayısı ise 1.968 milyar.

Digital’in Türkiye için düzenlediği rapora göre ise



* 79.14 Milyon insanın yaşadığı Türkiye’de, internete bağlanan kullanıcı sayısı 46,3 milyon.
* 46.3 Milyon internet kullanıcısından sosyal medyayı aktif olarak kullananların sayısı 42 milyon.
* 42 Milyon sosyal medya kullanıcısının 36 milyonu mobil cihaz üzerinden bağlanıyor sosyal medyaya.
* İnternet kullanıcılarının %77’si her gün online oluyor, %16’sı ise haftada en az bir kez internete bağlanıyor.
* Web trafiğinin %51’i diz üstü ve masaüstü bilgisayarlardan, %45’i mobil cihazlardan ve geri kalanı ise tabletler üzerinden gerçekleşiyor.

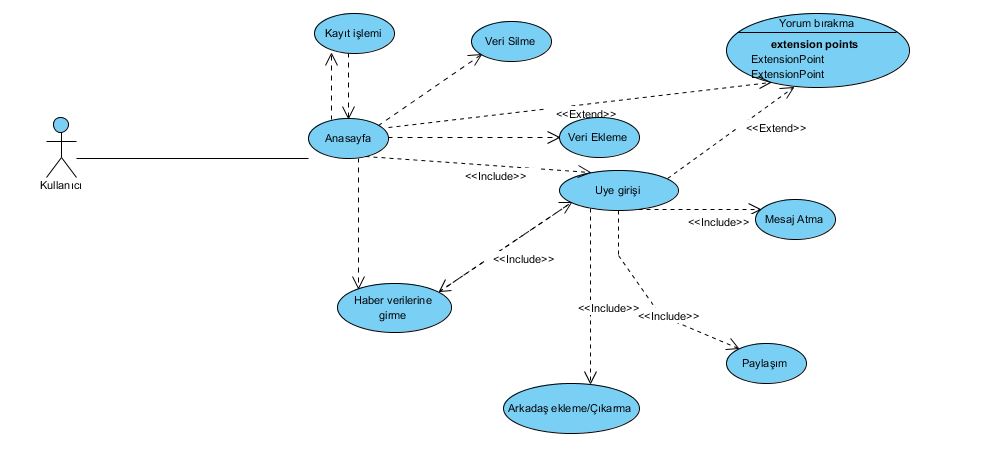
Ayrıca 2013 Ocak Ayı verilerine göre Türkiye’ de en çok takip edilen ilk 100 haber sitelerinin toplam ziyaretçi sayısı 100 bini aşmaktadır.

Bu veriler ışığında haber sitelerini ve sosyal paylaşım sitelerini tek bir çatı altında birleştirmenin kar getiren bir durum olduğu ortaya çıkmaktadır.

**3.3.3 Tasarım**

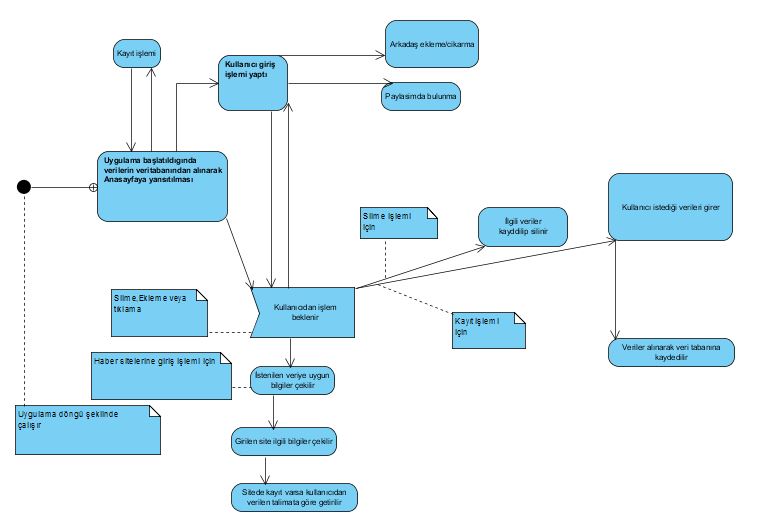
Bu aşamada aktöreler ve senaryoları ile projenin kodlamaya başlamadan önceki aşamalarından bahsedilecektir.

**Use Case Diyagramı:**

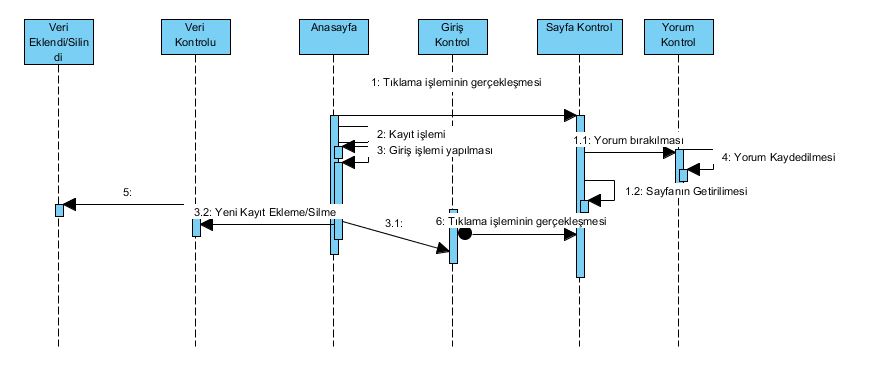


* Kullanıcı ana senaryoda giriş yapıp işleme devam edebilir.
* Giriş yapan kullanıcı arkadaş ekleme/çıkarma ve paylaşma yetkisine sahip olacak.
* Bireyse mesaj atma gibi özellikten yararlanacak.
* Giriş yapmadan devam eden kullanıcı sadece haber sayfalarına ulaşacak.

**Activity Diyagram:**

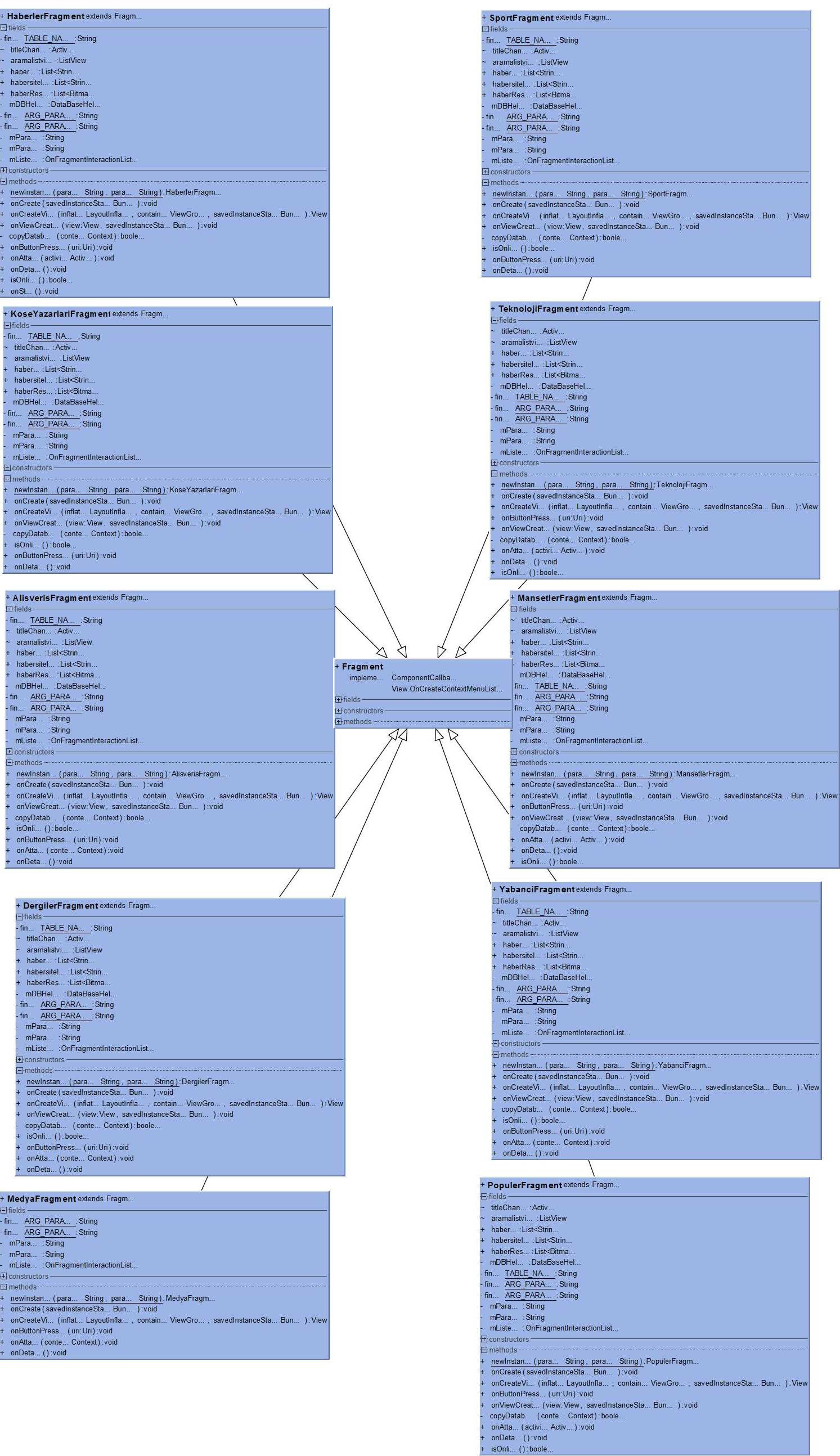
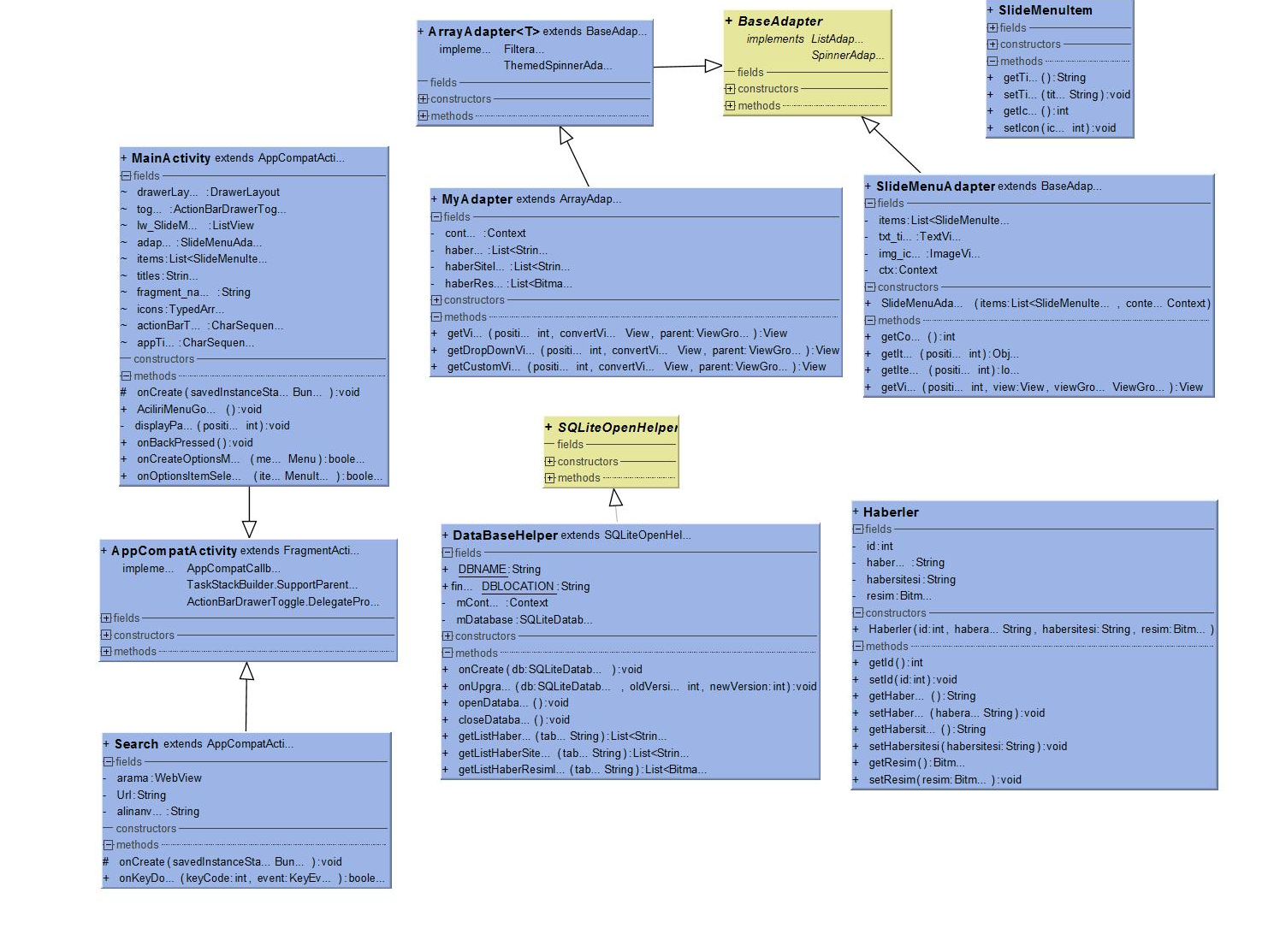


**Sequence Diyagram:**

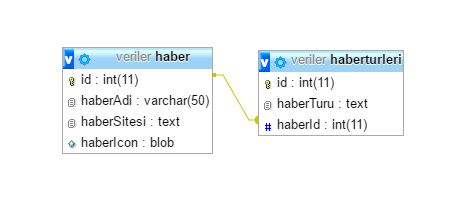
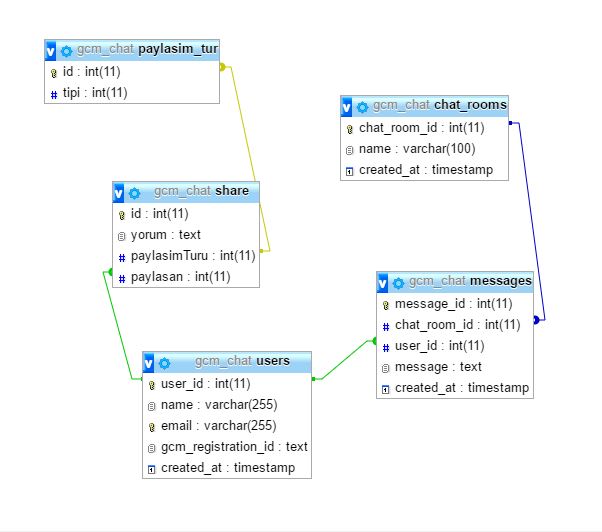
****

* Sistemin kontrol başlangıç noktası ana sayfaya giriş ile başlıyacak.
* Kayıt işlemi gerçekleştirildiğinde yeni bir intent yada sayfaya yönlendirilecek.
* Kayıt tamamlandığında ise tekrar anasayfaya yönlendirilecek.
* Kullanıcı isterse sisteme giriş yapıp devam edecek istemezse giriş yapmadan devam edecek.

**Class Diyagram:**



**Veri Tabanı:**



**3.3.4 Gerçekleştirim**

1. **Genel Bilgiler**

Tasarım aşamasını belirli bir olgunluğa erişmesiyle birlikte kodlama aşaması başlar. Kodlama süresince ve kodlama sonrasında yapılan en önemli aşama testtir. Erken test et yaklaşımı ile hareket edilmelidir. Analiz aşamasından itibaren test bakış açısına sahip olmamız hata yapma oranımızı ve maliyetleri (zaman, para vb.) düşürecektir.

Bu süreçte belirlenen,

* Yazılım geliştirme ortamı,
* Kodlama stili,
* Program karmaşıklığı,
* Veri tabanı

gibi aşamalar gerçekleştirim aşamasında önemli rol oynar.

1. **Yazılım Geliştirme Ortamı**

**PhpStorm :** Php de yazılım geliştirme ortamı için uygun bir IDE.

**DreamWeaver,Photoshop :** Sayfa tasarımı için kullanılacak programlardır.

**MySQL, SQLite3, Firebase :** Verilerin tutulacağı veri tabanları.

1. **Kodlama Stili**

Koda bakıldığı zaman ne yapılmak istendiğini kısa sürede anlayabilmek için yazılım geliştirmede kodlama stili önemlidir. Bunun olabilmesi için de uygulanabilecek en iyi yol kullandığımız dil, çalıştığımız platform gibi etkenleri de göz önüne alarak bir kodlama stili benimseme ve bu stili kararlı bir biçimde uygulamak olmalıdır.

Mümkün olduğunca gerekli yerlere yorum satırları eklenmeli, uygun tabları ve boşlukları bırakmalıyız.

1. **Program Karmaşıklığı**

Problem veya algoritmaların çözüme ulaşmadaki karmaşıklığını ölçmek için kullanırız.

Bu aşamada, her problem veya algoritmanın bir fonksiyon gibi düşünülebilmesi ve fonksiyonların karmaşıklığını sınıflandırma için kullanılan karmaşıklık sınıflarının, algoritmalar için uygulanması gerekmektedir.

Karmaşıklık sınıfları şunlardır:

* Küçük-o
* Büyük-O
* Teta
* Büyük omega
* Küçük omega

Bu sınıflar, aynı fonksiyon için uygulanan ve bir fonksiyonun farklı özelliklerini anlatan sınıftır. Bu sınıflar, bir algoritma hakkında bilgi edinmemizi sağlar. Bu bilgileri ise şu şekilde sıralayabiliriz:

* Bir algoritmanın ölçeği,
* Bir algoritmanın zaman ve hafıza ihtiyacı

Bu sonuçlara, algoritmanın kullanıldığı giriş bilgisine göre ulaşılır. Bir algoritmanın işlendiği veri miktarına göre ne kadar ve ne kadar yer gerektiğini gösterir.

1. **Kodu Gözden Geçirme**

Kodu yazdıktan sonra release etmeden ve bir kod deposuna koymadan önce bir veya birden çok kişi tarafından değerlendirilmelidir. Bu, kod kalitesini artırır, olası buglardan arındırır ve proje hakkında diğer takım üyelerinin de bilgi sahibi olmasını sağlar.

Yazılım geliştirme ekibinin geliştirdiği kodu gözden geçirmek için bir kontrol listesi kullanmalıdır. Bu süreci bir parçasıdır. Herhangi bir kod commit edilmeden önce aşağıdaki liste ile kontrol edilmelidir.

**Check List**

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.
21. **Sınama Stratejileri**

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

Sınama işlemleri dört ana sınıfta incelenebilir:

* + - Birim sınama
    - Alt-sistem sınama
    - Sistem sınama
    - Kabul sınaması

1. **Birim Sınama**

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

1. **Alt-Sistem Sınama**

Alt-sistem modüllerin bütünleştirilmeleri ile ortaya çıkar. Yine bağımsız olarak sınamalrı yapılmalıdır. Bu aşamada en çok hata ara yüzde bulunmaktadır. Bu yüzden ara yüz hatalarına doğru yoğunlaşılmalıdır.

1. **Sistem Sınaması**

Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde çıkacak hatalar aranmaktadır. Ayrıca, belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

1. **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.

1. **Alfa ve Beta Sınaması**

Alfa Sınamada; sistemin geliştirildiği yerde kullanıcıların gelerek katkıda bulunması sistemi test etmesi amaçlanmaktadır.

Beta Sınamasında; kullanıcı, geliştirilen sistemi kendi yerleşkesinde, bir gözetmen eşliğinde yapar.

Sınamalar, hatalardan kurtulmanın bir güvencesi değildir. Hatalardan bütünüyle

aranıldığı gibi bir kanı elde edilmemelidir. Ne kadar hata sıklığına erişildiğinde sınama işlemlerinin durdurulacağına, maliyet ve kalite arasında yapılacak bir en iyileşme çalışması ile ulaşılır. Yazılımın kritiklik düzeyine göre sınamaya ayrılan süre ve çaba artar.

1. **Sınama Yöntemleri**
   1. **Kara kutu testi (Black-Box testing ):** Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün testidir. Sistem şartnamesinin gerekleri incelenir.
   2. **Beyaz Kutu Testi (White Box testing ):** İç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır.

**f) Sınama ve Bütünleştirme Stratejileri**

**a. Yukarıdan-Aşağıya 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 arayüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.

İki temel yaklaşımı 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.



**b.Aşağıdan-Yukarıya Bütünleştirme**

Öncekiyöntemin 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.

****

Bu proje için uygun görülen sınama stratejisi aşağıdan-yukarıya sınamadır. İlk önce al birimlerin sınanması ve daha sonra üst birimlere çıkılması projenin verimliliği açısından daha iyi olacağına karar verildi.

**3.3.5 Teslim ve Bakım**

Tüm test aşamaları tamamlandıktan sonra yazılım ürünün sahaya teslim edilebilir bir versiyonu çıkartılır ve teslim aşaması gerçekleştirilir.

Son kullanıcılar için bir kullanım tanıtımı veya bir bilgilendirme dokümanı oluşturmalıdır.