Yazılım Testi ve Test Süreçleri

Yazılım Testi, kullanıcı gereksinimlerinin karşılanıp karşılanmadığı ve sistemin işlevselliğinin kontrol edildiği aktiviteler olarak tanımlanır.

**Açık Kutu Testi:[[1]](#endnote-2)**

Türkçe’de açık kutu testi olarak geçebilecek olan bu test, yapısal bir test demektir (structural testing). Bir bileşenin ya da bir iç mekanizması ile ilgili yapılan testtir. Uygulamanın temel kodunu test olarak alır ve iç yapısını kurmaya çalışır.

**Avantajları:**

Testi yapan kaynak kod elinde olduğu için uygulamanın etkin şekilde test edilmesi hangi veri türleri yardımcı olabileceği de daha kolay bulabilmektedir.

Kaynak Kodun optimize edilmesi yardımcı olmaktadır. Algoritmaların doğru tanımlanmış olması ve yazılmış olması, yazılım doğruluğunun belirlenmesini sağlar.

**Dezavantajları:**

Test için deneyimli ve yetenekli bir testçiye bir ihtiyaç vardır.

Sorun oluşturan gizli kusurların tespit edilmesi oldukça zor olabilir.

***Birçok farklı alt konusu olmakla beraber başlıcaları ve dikkat edilecek olanlar verilmiştir.[[2]](#endnote-3)***

***1)Kod Kapsamı:*** *Herhangi bir yazılım geliştirme şirketinin nihai hedefi, iyi kalitede yazılım geliştirmektir. Bu amaca ulaşmak için, yazılımın kapsamlı bir şekilde test edilmesi gerekir. Bütün bunların mantıklı bir şekilde işlenmesi ve saklanması ve bunun ar-gesi proje için oldukça faydalı bir süreçtir.*

***2)Segment Kapsamı***

***3)Veri Akış Testi (VAT):*** *Bu yaklaşımda, olası her hesaplama yoluyla belirli değişkenleri izlersiniz, böylece kod aracılığıyla ara yollar kümesini tanımlarsınız. VAT bağımlılıkları yansıtma eğilimindedir, ancak esas olarak veri işleme dizileri aracılığıyladır. Kısacası, her bir veri değişkeni izlenir ve kullanımı doğrulanır. Bu yaklaşım, kullanılan ancak başlatılmayan veya bildirilen ancak kullanılmayan değişkenler gibi hataları ortaya çıkarma eğilimindedir.*

***4)Yol testi:*** *Yol testi, koddaki tüm olası yolların tanımlandığı ve kapsandığı yerdir. Bu işin tamamlanması ciddi bir süre gerektirir.*

***5****)****Döngü Testi:*** *Bu stratejiler, tek döngülerin, birleştirilmiş döngülerin ve iç içe döngülerin test edilmesiyle ilgilidir. Bağımsız ve bağımlı kod döngüleri ve değerleri bu yaklaşımla test edilir.*

***Projede kullanılması öngörülen programlar ve kullanılma biçimleri:***

*#1 Veracode: Java vb. birkaç dilde birçok yazılımcının kullandığı profesyonel bir kod yönetim uygulamasıdır. Bu uygulamada bizzat projenin için bulunan testçimiz kod satılarını okumakta zorluk çekmeyecek daha verimi yüksek programlar için kullanacağız.*

**Kapalı Kutu Testi**

Kapalı kutu testi fonksiyonellik testidir. Bu test yönetimi, kod veya tasarımla ilgilenmez, işlevsellik ve fonksiyonel ihtiyaçlara göre test yapılır. Kapalı kutu testi yapılırken bu testçi girişlerin nasıl ve nerede yapıldığı bilmeden girdi sağlayarak ve çıktılarını inceleyerek sistemin kullanıcı ön yüzü ile ilgilenir. Bu genel kullanıcı deneyimi açısından önemlidir.

**Avantajları**

* Büyük kod parçaları için çok uygundur.
* Yük testi, kullanılabilirlik testi gibi sadece kapalı kutu testi ile test sınıflarının çoğunu gerçekleştirmemizi sağlar.
* Açık kutu testlerine göre daha az kaynak gerektirir.

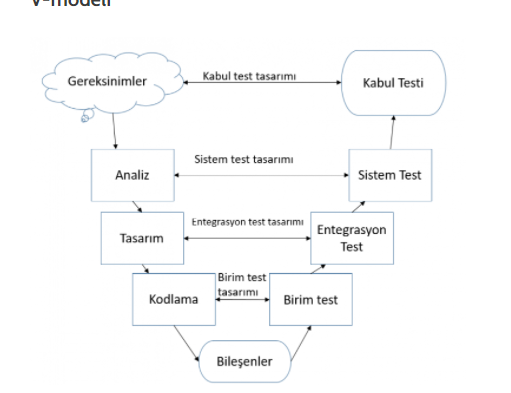
**Dezavantajları:**

* Yanlışlıkla veya es-kaza birkaç hatanın doğru sonuç üretmesi durumunda buradaki hataları kapalı kutu testi kolayca tespit edemez.
* Test case/test durumu kapsamına alınmayan bazı kod satırlarının test edilmemsi durumu oluşabilir.
* Sadece sınırlı sayıda test senaryosu gerçekleştirilebilir.
* Test case/test durumlarının tasarlanması zordur.
* Ayrı bir gray box testing kullanılan bir yöntemdir fakat bu projede buna gerek duyulmamıştır.

***Projede kullanılması öngörülen programlar ve kullanılma biçimleri:***

Bunun için bir program gereki olmamakla beraber frontend(ön yüzü) şeklinde kullanımda test edileceğinden proje çıkış sürecine göre şekillenecektir.

## **Yazılım Geliştirme Süreci**



V-model (yazılım geliştirme) şelale (waterfall) modelinin gelişmiş hali olarak düşünülebilecek bir yazılım geliştirme süreci sunar. Doğrusal bir yönde ilerlemek yerine, süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve tipik V şeklini oluşturur. V-Model geliştirme yaşam çevriminin her bir evresi arasındaki ilişkileri gösterir. Yatay ve dikey açılar zaman veya projenin tamamlanabilirliğini ve soyut seviyeyi gösterir.

***Projede kullanılması için neden seçildi?***

*V modeli genel olarak birkaç farklı testten geçtiği için sürekli kullanımda ve kullanıcı haklarını koruma açısından maksimum güvenlikli uygulama tasarlamak zorundayız. Bunun için test edilecek uygulamanın genel modele tam uyması ve çalışması gerekecektir.*

**Birim Testi** **(Unit Testing) :** Dinamik test sürecinin ilk aşaması olmakla beraber, hataların erken bulunup düzeltilebilmesi açısından da bu sürecin en önemli aşamasını oluşturur. Mikro ölçekte yapılan bu testte, özel fonksiyonlar veya kod modülleri (fonksiyonlar, veri yapıları, nesneler vb.) test edilir. Bu test, test uzmanlarınca değil programcılar tarafından yapılır ve program kodunun ayrıntıları ile içsel tasarım biçiminin bilinmesi gerekir. Uygulama kodu çok iyi tasarlanmış bir mimaride değilse oldukça zor bir testtir.

***Birim testi proje için ne ifade etmekte?***

*Bizim alanımızda sürekli olarak (real-time) çalışan bir sınıflandırma ve tanıma algoritmaları temelli olduğundan bütün komponentlerin tam manası ile birbirine uyumlu olması gerekmektedir. Bu sınıflandırma temeli olarak database (veri tabanı) kullanılacağından sürekli bağlantı kaçınılmazdır.*

**Entegrasyon Testi;**

Unit test’e göre daha yavaş çalışır, SQL’e ya da network’e bağlanabilir, network’ten belli bir veri getirebilir, SQL’den veri getirebilir, validasyonunu sağlayabilir, her bir entegrasyon test altındaki sınıfları, fonksiyonlar ve altındaki network, io işlemlerini yapar ve unit test’e göre daha yavaştır.

**Bigbang Testi:** En yaygın kullanılan entegrasyon test tipidir. Geliştirilmiş tüm modüller bir araya getirilerek yapılan testtir. Hızlı ve kolay bir şekilde birbirleri ile beraber çalıştıklarında anlam ifade eden modüllerin doğruluğunu sağlar fakat birim başı metot doğruluğunun gözden kaçınılması olasıdır.

**Top-Down Integration Testi:**

Bu entegrasyon testindeki amaç ise modüller arası geçiş yapılırken hatalı olan modülün kolay bir şekilde bulunabilmesini sağlamaktır. Test işlemi yukarıdan aşağı doğru gerçekleşmektedir ve her birinin test işleminden başarılı bir şekilde geçerek ilerlemesi gerekmektedir. Her bir modül testleri stub olarak adlandırılmaktadır. Modül ağacının son bacaklarında ise her bir stub kendi içerisinde test edilerek test işlemi sonuçlandırılır.

#### **Bottom-Up Integration Test**:

Bu test yöntemi ise Unit Testler ile beraber ilerlemektedir. Alt tarafta bulunan tüm stublar, Unit Testlerden geçirilerek yukarıya doğru ilerlenir. Top-Down’da olduğu gibi yukarıya ilerlerken Unit Testler aracılığı ile her test başarılı olarak sonuçlanmalıdır. Tüm stublar için Unit Testler oluşturulduktan sonra bir üst seviyede hepsi bir ele alınarak test işlemi yapılır. Bu test tipindeki amaç ise stublardan başlayarak hataların en kısa sürede bulunabilmesidir.

#### **Sandwich/Hybrid Integration Test**:

Modüllerin bir kısmı Top-Down, bir diğer kısmı ise Bottom-Up tiplerini kullanılarak gerçekleştirilen test tipidir. Bu karma tipteki amaç ise bazı modülleri gruplara ayırabilirken diğer modülleri ise ayrı bir şekilde test edebilmektir. Genel hatları ile entegrasyon test tipleri bu şekildedir. Yukarıda da belirttiğim gibi en sık kullanılan entegrasyon test tipi genelde Big Bang Integration Test dir. Amacımız genelde yazdığımız modüllerin tutarlılığını sağlamak olduğu için bu test tipi bir çok zaman yeterli olmaktadır. Tabi bu durum birazda iş kurallarına göre de değişiklik göstermektedir. Bir diğer faydası ise entegrasyon işlemi sırasında oluşabilecek hataların önceden önüne de geçebilmektir.

***Entegrasyon testi proje için ne ifade etmekte ve hangisi kullanılacak?***

Sandwich model düşünülmekle beraber Botto up modeli de projemiz için oldukça uygundur. Entegrasyon testi ünite testinde yazılan kodların birbiri ile çalışması ve entegre olması kontrol edilmesi açısından çok fazla öneme sahiptir. Bütünlüklü olarak çalışan ve çalışmaya devam uygulamada uyumsuzluk sorununu bitirmeye çalışacağız.

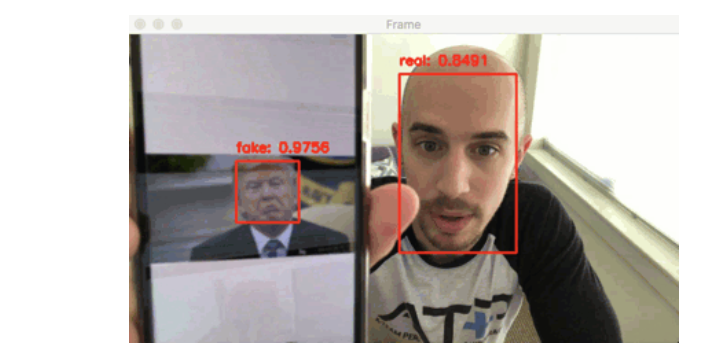
**Deneysel Modüller:**

***Yüz Blurlama Modülü:*** Yüz kişisel haklar nedeniyle düzgün ve paylaşılmayacak şekilde saklanılması gerektiğinden yüzü blur hale getirip o şekilde saklanacaktır[[3]](#endnote-4). Bu bize hem güvenlik açısından belli bir etki kazandırırken hem de algoritmaların genişliği ve öğreticiliği gibi farklı alandarda etkinlik ve yetkinlik katacaktır.

****

***Yüzde Maske Tanıma Modülü:*** Kişiler yüzlerinde maskeler ile[[4]](#endnote-5) bulunacaklarından bir geri dönüt alınması için bu modülün kullanılması ön görülmektedir. Dezavantaj olarak sadece kişilerin farklı çekilmiş fotoğrafları ile bir veri bulunması gerekmektedir ki bu da otomatik olarak kişi tarafından tanılabilir.



**Yaşam belirti Modülü: [[5]](#endnote-6)**Bu modül genel olarak kişilerin sahte fotoğraflar ile girmesini önleyecek. Genel toplum güvenliği açısından denenmesi muhtemel olan bir algoritmadır.

1. http://www.arakatman.com/yazilim-test-surecleri/ [↑](#endnote-ref-2)
2. https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/ [↑](#endnote-ref-3)
3. https://www.pyimagesearch.com/2020/04/06/blur-and-anonymize-faces-with-opencv-and-python/ [↑](#endnote-ref-4)
4. https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/ [↑](#endnote-ref-5)
5. https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/ [↑](#endnote-ref-6)