**MOBİL FOTOĞRAF DÜZENLEME**

Onur GÜLER

*Bilgisayar Mühendisliği Bölümü*

*Kocaeli Üniversitesi*

*170202031*

[45onurguler45@gmail.com](mailto:45onurguler45@gmail.com)

**1.Özet**

Bu projenin amacı android tabanlı bir mobil cihazda, uygulama da kullanıcı istediği resmi veya fotoğrafı google cloud, amazon veya IBM’in sağladığı cloud desteği kullanılarak depolama ve resim de sıkıştırma veya nesne tanımlama yapılması isteniyor.

**2.Giriş**

Mobil uygulamayı açan bir kişi ilk önce üst tarafta bulunan galeriden fotoğraf veya resim seç butonu ya da kamerayı kullanarak yeni bir fotoğraf çek butonuna basmalı. Bastığı butona göre galeriden bir resim seçmesi veya açılan kamera ile bir tane fotoğraf çekmesi isteniyor. Daha sonra resmi seçtikten ya da fotoğrafı çektikten sonra resmi ekranda 2 butonun altında göster. Daha sonra sayı ile yetmiş beş yazan yere sıkıştırma değerini girip sıkıştır yazan butona basılırsa firebase storageye resmin başına girilen değeri de yazarak yükle. Storageye resim yüklendiğinde firebase fonksiyonlarında olan strorageye resim yüklenmesi halinde çalışan fonksiyonu çalıştır. Çalışan fonksiyon resmin başındaki değeri alır ve isminden çekilen değere göre resmi sıkıştırır, resmi sıkıştırdan sonra farklı bir resim olarak storageye geri yükler. Daha sonra uygulamada kullanıcı sıkıştırılan resmi göster butonuna basarsa textboxa girilen değerdeki kadar sıkıştırılanresmi storageden çek ve uygulamadaki image

view da göster. Nesne tanımla butonuna basılırsa Ml kite resimi gönder ve mesajbox olarak tanımlanan kısımda 4 taneye kadar algılanan öğelerin isimlerini göster.

**3. Temel Bilgiler**

Proje gelişiminde;

Tümleşik geliştirme ortamı olarak “Android Studio”, dil olarak ise “Java” kullanılmıştır. Cloud tarafında ise Google cloud kullanılmıştır.

**4. Sorunlar ve Yapılan Araştırmalar**

Android Studio da uygulama arayüzü tasarımı için schemas kütüphanesinde ki ImageView, Button, Edittext, text view gibi araçların kullanımı, cloud tarafında ise googlenin sağladığı firebase, firebase strorage, cloud functions ve fonksiyonda kullanılan resim manipüle kütüphanesi imagemagick’i anlamak için kaynakçada belirtilen linklerden yardım alınmıştır.

**Firebase kurulumu ve projeye ekleme;**

Firebase’i başta kurarken bağlantıyı yanlış yaptığım için uygulama çökmeye başladı daha sonra gradle(app) klasörüne gerekli kütüphanelerin eklenmesi gerektiğini öğrendim. Bağlantı sorunum böylelikle çözülmüş oldu.

**Storage’ye dosya yükleme;**

Storage’ye ilk önce galeriden bir fotğraf ya da resim seçilerek göndemeyi denedim. Tanımladığım Intent değişkenine seçtiğim dosyayı atadım. Daha sonra onActivityResult ile Uri değişkenine Intent değişkeninin datasını getData fonksiyonu ile atadım. Bu seçilen dosyayı storageye atmak için Uri değişkeninin yolunu getLastPathSegment fonksiyonu ile dosyanın yolunu alıp putFile ile storageye yükledim. Burada bir sorun yaşamadım ama kamer ile bir fotoğraf çektiğimde uygulama hata veriyordu. Bu hatanın kamer ile fotoğraf çekildiğinde Intent data değişkenini Uri değişkeninde atadıktan sonra resimin yolunu döndüren getLast PathSegment fonksiyonu çalıştırıldığında dönen değerin ‘null’ olduğunu fark ettim. Bunu da global bir Uri değişkeni tutarak daha ilk başta kameradan fotoğraf çekildikten sonra MediaStore Images Media External contenturi sayesinde global tanımladığım uri değişkenine attım. Kamera ile işlem yapıldığında Intent data değişkenini Uri atamadım ve direk global olarak tuttuğum Uri değişkenini kullandım ve sağlıklı bir şekilde çekilen fotoğrafın dosya adresine erişebildim ve böylelikle storageye resmi gönderebildim.

**Cloud functions ile resim sıkıştırma;**

Storage’ye bir resim veya fotoğrafı yükledikten sonra bu resim üzerinde sıkıştırma işlemi yapmak için önce storageye dosya yüklendiğinde tetiklenen bir fonksiyon olup olmadığını araştırdım ve github’daki(kaynakçada belirtildi) simple-functions’ı keşfettim. Burada thumbnails diye bir fonksiyon buldum ve bu fonksiyon storage’ye resim yüklendiğinde tetiklenip bu resmin boyutunu thumbnail olacak şekilde değiştiriyordu. Ama sıkıştırma yapmıyordu. Bu işlemi convert’in içinde ‘-thumbnails’,’200\*200’ şeklinde tanımlanmıştı. Bir gün boyunca imagemagick kütüphanesi ile resim manipülasyon işlemlerine baktım ve deneye deneye sıkıştırma işlemini ‘-quality’,’x’(x int değeri sıkıştırma değeri) x ne kadar küçük olursa o kadar fazla resmi veya fotoğrafı sıkıştırıyordu. Resmi sıkıştırdıktan sonra ise storage’ye resmi farklı bir isimde geri yükledim. Dosya yüklü olarak storage’de gözüküyordu ancak firebase storage kendisi otomatik erişim anahtarı oluşturmuyordu ve dosyanın kendisi görüntülenemiyordu. Storage yi açıp sıkıştırılan resme tıklayıp sol altta create accses token diye bir seçenek var ona bastığımda bir token oluşturuluyor ve görüntülenebiliyordu. Ama bunu yapmak saçmaydı, bir müddet araştırma yaptıktan sonra rastgele erişim anahtarını oluşturamadım. Cloud’da gezinirken erişim anahtarı diye bir şey gördüm. Bu anahtarı yazdığım fonksiyonda metadata değişkenindeki firebaseStorageDownloadTokens adlı metadata bilgisine aktardım. Bunu yaptıktan sonra her sıkıştırılan resmin anahtarı aynı oldu ama görüntülenememe sorunu ortadan kalktı.

# Java’da Değişkenler ve Veri Türleri

**ArrayList:**

Tıpkı Arra*y* gibi index yapısını kullanmaktadır. Ekleme, silme, sıralama, arama, ters çevirme gibi işlemler yapılabilir. ArrayList*leri* Array’den daha önde tutan dinamik bir yapısının olmasıdır. Array’lerde tanımlanma aşamasında dizinin boyutu belirtilmelidir, daha sonra program içinde dizinin boyutunu arttırmak zahmetli bir iştir. (Sürekli düzenli veya düzensiz bir şekilde arttırmak gerektirdiğinizi düşünürsek)**ArrayList** dinamik bir yapıya sahip olduğu için tanımlanırken boyutunun belirtilmesine gerek yoktur, çalışma anında dahi liste içine eleman eklenebilir, silinebilir, istenilen indise eleman eklenerek öteleme yapılabilir. Array’lerde bu mümkün değildir.

**ArrayList** listesinin elemanlarına erişebilmek için tıpkı Arraylerde olduğu gibi index numarası kullanılır. Ayrıca Arrayler sadece tanımlandıkları türde veri saklayabilirken, ArrayListler farklı türdeki verileri bir liste içinde barındırabilir.Dinamik olarak genişleyebilen tek boyutlu dizilere ArrayList denir.ArrayList kullanmak için kodumuzun en yukarısına

**import java.util.\*;**

eklememiz gerekmektedir.

**ArrayList dizi\_adi=new ArrayList();**

şeklinde kullanılır.

**String:** Metinsel veri türüdür.

|  |  |
| --- | --- |
| 1  2  3 | String deneme00 = "dedendeneme";      String deneme01 = "1234";      String deneme02 = '9Cdeneme'; |

Bir değişkeni string olarak tanımlayabilmek için = operatöründen sonra tırnak işareti içerisinde değişkenin değerini yazmalıyız. Yukarıdaki örnekte deneme00 adlı bir değişken tanımladık ve değerini zinzinzibidi şeklinde string olarak belirledik. deneme01 değişkeni de string'dir. Çünkü değeri sayısal olmasına rağmen tırnak işaretleri içinde yazdık. Tek tırnak ('') işareti içerisine yazılan değerler de string olarak algılanır.

**Boolean:** Mantıksal veri türüdür.

|  |  |
| --- | --- |
| 1  2 | int deneme05 = true;      int deneme06 = false; |
|  |  |

Değişkenin değeri ya true (doğru) ya da false (yanlış) olabilir.

## Intent ve Uri:

Intent’in **action** parametresi mevcut. Bu özelliği etkileşime (Activity) birden fazla faaliyet (eylem) yapma imkanı sunuyor. Tabii bir faaliyet (action) yapmak için nesneye ihtiyacımız vardır. Bunun için **Intent**’te bir parametre (**data**-veri) mevcut. **Data**’ya değeri atamak için [setData (Uri data)](https://developer.android.com/reference/android/content/Intent.html#setData%28android.net.Uri%29) metodunu kullanabiliriz. Bu metoda giriş olarak [Uri](https://developer.android.com/reference/android/net/Uri.html) nesneyi gönderiyoruz. **Uri** – nesne bir karakter diziyi alıp bölüyor ve bölen parçaları içeriyor. **Uri**’nin amacı soyut veya fiziksel kaynağı belirlemek. En basit örnek olarak web sayfası kullanalım. “<http://developer.android.com/reference/android/net/Uri.html>”

Bu adresi birkaç parçaya bölebiliriz:

* Scheme – http
* Scheme-specific part – *//*developer.android.com/ref/android/net/Uri.html
* Path – */*reference/android/net/Uri.html

**5. Kazanımlar**

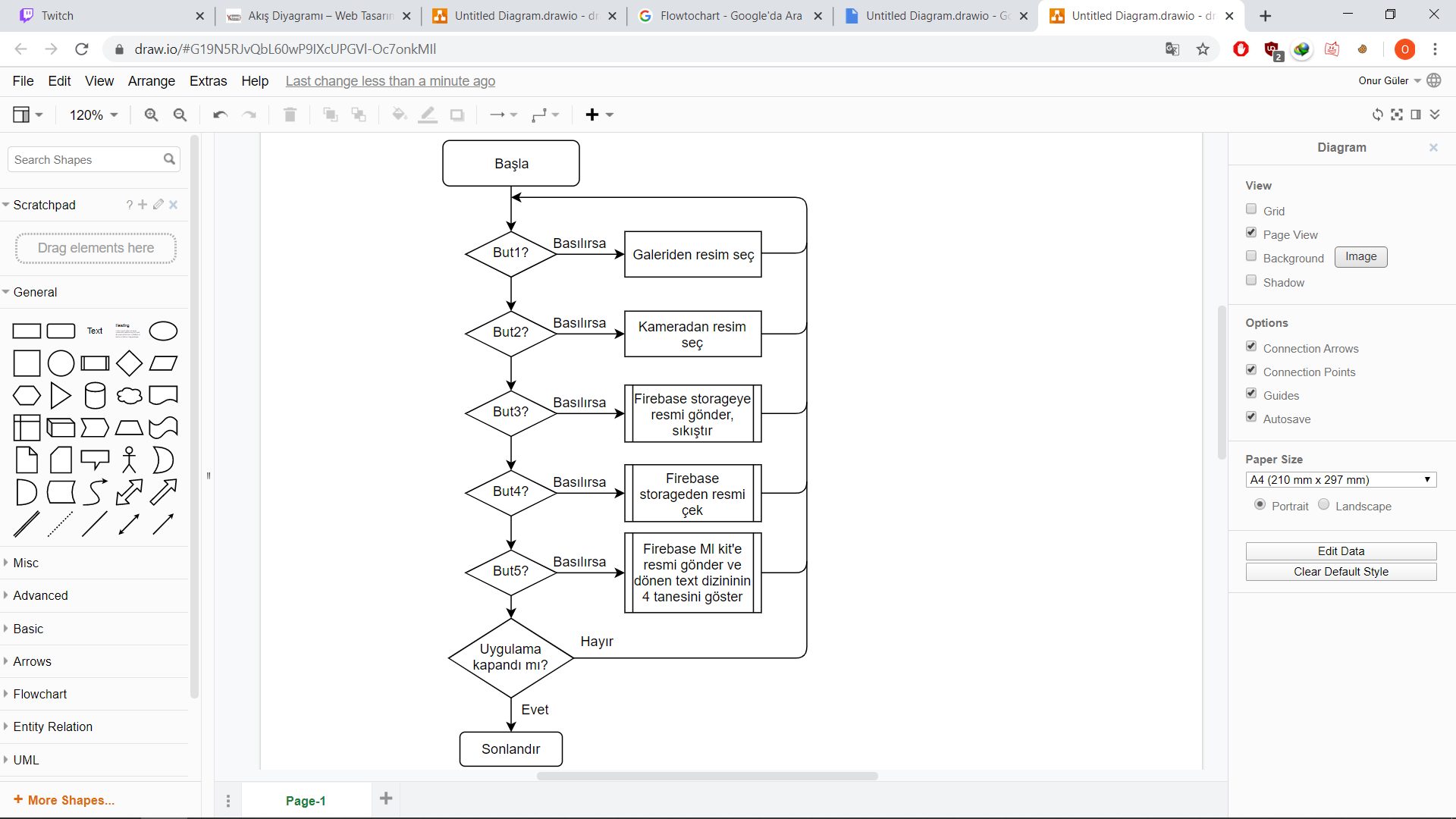
**Android Studio ile Mobil Programlama**

Java da daha önce projeler yazmıştım ama mobil ile ilgili bir proje ya da uygulama yapmamıştım. Android tabanlı bir mobil uygulama geliştirmeyi ve ara yüzünü tasarlayabilmeyi, butonlar ile galeriden veya kameradan fotoğraf seçip uygulamaya yükleyebilmeyi öğrendim.

**Google Cloud**

Google cloudun sağladığı firebase ve ml kit ile uygulama geliştirdim. Firebase’e gerçek zamanlı veri aktarımı ve veri görüntülemeyi, storageye dosya yükleyip dosya çekmeyi, storageye bir dosya yüklendiğinde cloud functionsı kullanarak bu resim dosyası üstünde değişiklikler yapmayı ve ml kiti uygulamaya dahile edip cloud tabanlı nesne tanımlamayı öğrendim.

**6.Tasarım**

**a) Akış Diyagramı**

**b) Yazılım Mimarisi**

Mobil uygulamayı çalıştırıldığında ilk açılan ekranda üstte 2 buton ,alt tarafta 3 buton olmak üzere 5 buton ve edittext karşımıza çıkıyor. Üst taraftaki soldaki butona tıklandığında galeriden veya dosya yöneticisinden bir resim veya fotoğrafı seçmesi isteniyor. Bir resim veya fotoğraf seçildikten sonra uygulama ekranı geri geliyor ve imageview değişkeni içerisinde seçilen resim veya fotoğraf gösteriliyor. Aynı şekilde sağ üstteki kameradan fotoğraf seç butonuna basıldığında ise karşımıza kamera çıkıyor ve bir fotoğraf çekmemiz isteniyor. Fotoğrafı çektikten sonra sağ alttaki tik işaretine tıklıyoruz ve galerideki gibi uygulama ekranı geri geliyor. Uygulama ekranımızdaki imageview da bu sefer çektiğimiz fotoğraf gözüküyor. Daha sonra iki butonun arasında bulunan edittextte sayı ile yetmiş beş yazıyor bu rakamı istediğimiz sıkıştırma oranına göre değiştiriyoruz. Sıkıştırma işlemi bir ile yüz arasında bir değer olmalı bir değeri girilirse sıkıştırma çok fazla olur, eğer 99’a yakın bir değer girilirse sıkıştırma oranı az olur ve göz ile gözükmesi zor olur. Resim seçildikten sonra ve istenen sıkıştırma değeri girildikten sonra sol taraftaki sıkıştır butonuna basılırsa seçili resmin dosya yolunu regex ile parçalayarak dosya ismini al edittext’te belirtilen integer değeri dosya ismini başına arasında ‘-’ olacak şekilde Google cloud storageye gönder. Daha sonra firebase cloud functionsta çalışmakta olan storageye bir dosya yüklenip yüklenmediğini kontrol eden fonksiyon, yeni yüklenen bu resmi imagemagick yardımı ile bu resmin kalitesini resmin doya ismindeki değer ile sıkıştırır. Resim sıkıştırıldıktan sonra ise tekrar storageye geri yükler. Bu esnada uygulamada 2-3 saniye sonra sol taraftaki resmi göster butonuna basıldığında sıkıştırılmış olan resmi ekrandaki imageviewda gösterir. En alttaki nesne tanımlama butonuna basılırsa firebase Ml kite ekrandaki resmi gönder ve cloud yardımıyla bu resimdeki nesneleri tanımlayıp ekranda mesaj olarak tanımlanan nesnelerin ilk dördünü ekranda göster.

# 7. Kaynakça

Firebase storageye resim yükleyip çekmek:

\* <https://firebase.google.com/docs/storage/android/start>

\* <https://www.youtube.com/watch?v=Mhn6n-16kuA>

\* [https://www.youtube.com/watch?v=jTfMSSPTkZw& t=1276s](https://www.youtube.com/watch?v=jTfMSSPTkZw&%20%20%20%20%20t=1276s)

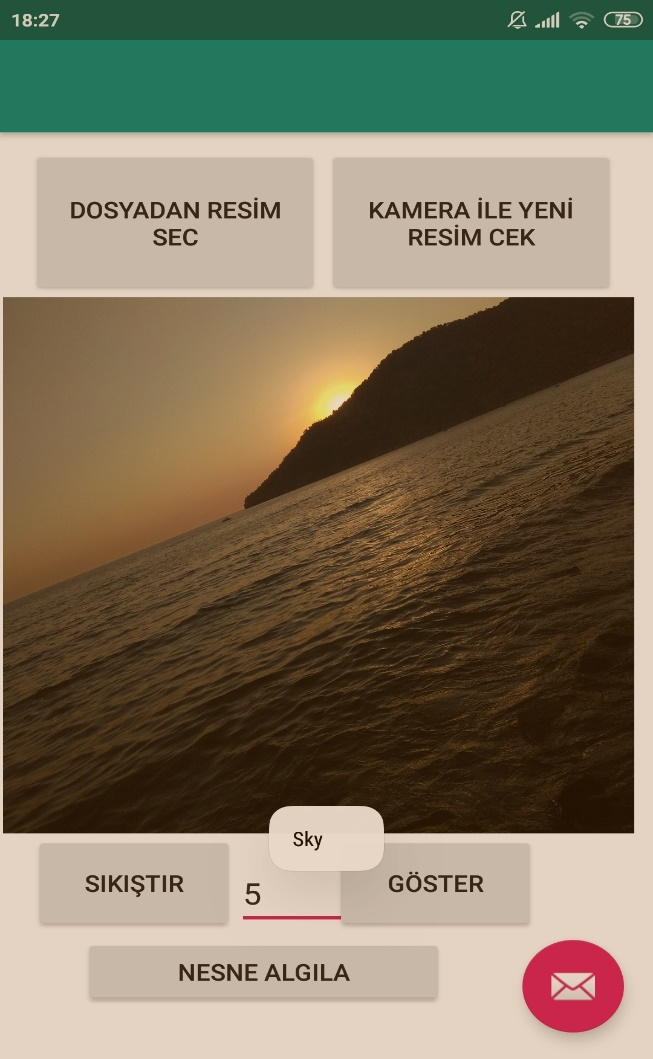
Firebase functions kullanımı:

\* <https://www.youtube.com/watch?v=DYfP-UIKxH0&t=493s>

\* <https://github.com/firebase/functions-samples/tree/master/quickstarts/thumbnails>

Firebase Ml kit kullanımı:

<https://firebase.google.com/docs/ml-kit/android/label-images>

**8. Deneysel Sonuçlar**

