AKILLI TELEFONLAR İLE NESNE TESPİTİ PROJESİ

Hüseyin Yılmaz
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
ylmzhsyn98@gmail.com

Özet

Bu projede, React Native framework'ünü kullanarak o anda kamerayla çektiğimiz veya galeriden hazır aldığımız bir resmi, bulut servislerinden birine yollamamız, orada resimde bulunan nesneler tespit edildikten sonra kareler içine almamız, karelerin üstüne de nesne isimlerini yazı ile bastırmamız, işlenmiş resmin linkini mobil uygulamaya döndürdükten sonra ekranda göstermemiz istenmiştir.

Giriş

Çalışmada amaç, bulut servislerini, API kullanımını ve React Native kullanarak akıllı telefonlar için mobil uygulama tasarımını öğretmektir. Projenin frontend kısmında React Native kullanımı zorunlu tutulmuştur. Backend'de ise herhangi bir bulut servisi veya programlama dili kısıtlamasına gidilmemiştir. API, Google App Engine'e yüklenmiş ve Python'ın Flask kütüphanesiyle yazılmıştır. Resimler ise Google Cloud Storage'da saklanmıştır.

Bizden hem emülatör hem de fiziksel cihazlarda çalışacak bir uygulama tasarlamamız istenmiştir. Uygulama ekrana geldiğinde kamerayla yeni bir resim çekilmesi veya galeriden hazır bir resmin seçilmesi için seçenek gelmektedir. Resim seçimi yapıldıktan sonra, seçilen resim ekrana bastırılmaktadır.

Daha sonra resim onaylandıktan sonra, resmin bilgileri Google App Engine'de bulunan API'a gönderilmektedir. API, öncelikle resmi decode etmekte, daha sonra da Google Cloud Vision'da bulunan nesne tanıma API'ına göndermektedir. Bu API'dan, resimde bulunan tüm nesneler ve konumları dönmektedir.

Nesne bilgileri alındıktan sonra resimdeki tüm nesneler, gelen koordinatlara göre kutu içine alınmakta ve tüm kutuların üstüne nesnenin ismi yazdırılmaktadır. Tüm nesnelerin bilgisi bastırıldıktan sonra resmin son hali Google Cloud Storage Bucket'ına yüklenmektedir. Yüklenen dosya herkese açık hale getirildikten sonra dosyanın linki ve tespit edilen nesne sayısı React Native uygulamasına döndürülmektedir.

Daha önce uygulamada gösterilen işlenmemiş resim, API'dan gelen bu yeni resim ile değiştirilmekte ve tespit edilen nesne sayısı da yazdırılmaktadır. Bir resim işlendikten sonra üstüste binmeleri önlemek adına onay butonu kaldırılmakta ve yeni resim seçildiğinde tekrar gözükmektedir.

Ayrıca uygulamada dil değiştirme butonu da bulunmaktadır. Uygulama açılırken telefon dili Türkçe ise uygulama Türkçe açılmakta, başka herhangi bir dil ise veya dil tespit edilemezse İngilizce olarak açılmaktadır.

Yöntem

Frontend yazılırken NodeJS'in son uzun dönem destekli sürümü olan 14 versiyonu kullanıldı. Componentler yazılırken sınıf yerine fonksiyonlar kullanıldı. Böylece state değişkenleri de useState hook'u kullanılarak tanımlandı.

Ana component'teki "loading" değişkeni, uygulamada zaman gerektiren bir işlemin devrede olup olmadığını belirlemektedir. Bu değişken true olduğu zamanlarda ekranın tamamını kaplayan bir yükleme ekranı görünmekte, işlem bitince de yok olmaktadır. Bu yükleme ekranı, Animated.View component'i ile yazılmıştır, böylece ekranda görünürken bir anda değil, belirli süre içerisinde belirmesi sağlanmıştır.

Resim seçim işlemleri için react-native-image-picker modülü kullanıldı. Bu modül hem kameradan hem de galeriden resim seçimini desteklemektedir. İki işlem için ayrı ayrı fonksiyon yazmak yerine tek bir fonksiyon yazıldı ve uygun seçim fonksiyonu da

bu fonksiyona parametre olarak gönderildi. Seçilen resmin adresi, base64 kodlu stringi, genişliği ve yüksekliği kullanılmaktadır. Uygulama çalıştığında resim genişliği olarak ekranın %90'ı alınmaktadır. Oranı korumak için de her resim seçimi sonrasında yükseklik olarak resim yüksekliğinin, resim genişliğinin alan genişliğine bölümüne bölümü alınmaktadır. Böylece tüm resimler aynı genişlikte gösterilmekte, yükseklik oranlanmaktadır. Seçim sırasında hata olması durumunda da hata ekrana bastırılmaktadır.

Seçim işleminden sonra ekranda bir onay butonu gözüküyor. Bu butona tıklandığında backend'de çalışan API'a istek gönderiliyor. Bu API, saf resim verisi yerine, resim ve videoların string formatında aktarımını sağlayan base64 şifrelemesinde veri kabul etmektedir. Bu veri zaten react-native-image-picker modülünden geldiği için ayrı bir işlem gerekmedi. Sunucu bağlantısı kurulmadan önce yükleme ekranı ekrana yansıtılmaktadır.

API, Python'ın Flask kütüphanesinde yazıldı. "/" yoluna gelen post isteklerini takip etmektedir. İstek yapıldığı zaman öncelikle Python'da yüklü olarak gelen base64 kütüphanesini kullanarak şifreli veriyi decode etmektedir. Daha sonra da bu decode edilmiş veriyi, BytesIO objesine dönüştürmektedir.

Görsel üzerinde düzenlemeler yapmak için pillow kütüphanesi tercih edildi. Bu kütüphane hem resmin genişlik, yükseklik, piksel başına renk bilgisi gibi bilgileri sağlamanın yanı sıra, görsel üzerine yazı yazma, şekil çizdirme, renk formatı değiştirme gibi işlemleri desteklemektedir.

Buffer'da bulunan görsel bu kütüphane ile açıldıktan sonra görselin boyut bilgileri hafızaya alınıyor. Sıra görselin nesne bilgilerini almaya geldiğinde ise devreye Google Cloud Vision servisi devreye girmektedir. Program zaten Google bünyesinde çalıştığı için güvenlik bilgilerinin ayrıyeten belirtilmesine gerek kalmamıştır.

Bu API, birden fazla resim formatını desteklemektedir. Bu sayede ek bir dönüşüm yapmadan base64'ten decode edilmiş veri yollanabildi. Bu API'dan gelen koordinatlar, pikselleri belirtmemekte, bunun yerine yüzdeye göre bilgi vermektedir. Bu yüzden koordinat hesabı yaparken görsel yükseklik ve genişliğinin gelen koordinatla çarpılması yöntemi kullanıldı.

Sırasıyla kutucuğun sol üst, sağ üst, sağ alt ve sol alt koordinatları döndürülmektedir. Bu koordinatlar yardımıyla önce kutucuk koordinatları hesaplandı, daha sonra da yazı koordinatları. Yazı koordinatı hesaplanırken Y koordinatı olarak üst kenarın, resim yüksekliğinin 16'ya bölümü kadar fazlası alındı. Yazı boyutu olarak da görsel yüksekliğinin 14'e bölümü alındı, böylece resim boyutu ne olursa olsun yazıların okunacak boyutta olması sağlandı.

Kutucuk ve yazı renkleri belirlenirken öncelikle yazının başlangıç koordinatındaki rengin RGB bilgileri alınmaktadır. Bu RGB verisi, Relative Luminance denen bir hesaplama yöntemine tabi tutulmakta ve koordinatın rengi siyaha daha yakınsa renk olarak beyaz belirlenmekte, beyaza daha yakınsa da siyah olarak belirlenmektedir. Böylece kutucuk ve yazıların bir nebze daha kolay anlaşılması amaçlandı.

Pillow kütüphanesinde dikdörtgen çizdirilirken sırasıyla sol üst köşe ve sağ alt köşe koordinatları girilmektedir, bu da Google API'ından dönen verideki 0. ve 2. koordinatlara denk gelmektedir.

Tüm koordinatlara göre işlem yapıldıktan sonra Google Cloud Storage hizmetine bağlanılmaktadır. Uygulamamız Google App Engine'de çalıştığı için yine güvenlik bilgisi girişine gerek kalmamıştır.

Bağlantı kurulduktan sonra bulutta bir dosya oluşturulmaktadır. Dosya adı belirlenirken daha önce kullanılan bir isim olmaması için o anki tarih ve saatin yanına random bir sayı konulmakta ve uzantı olarak da "JPEG" belirlenmektedir. Daha önce elde ettiğimiz BytesIO objesinin string değerini alıp, bu değerle yükleme yapılıyor. Yükleme sonunda dosya yalnızca hesap sahibine görünür olduğu için, herkese açık yapma fonskiyonu çağırılıyor. En sonunda da dosya linkiyle birlikte nesne sayısı frontend'den gelen isteğe cevap olarak yollanıyor.

Tüm işlemler başarıyla tamamlandıysa durum kodu olarak 200, gelen istekte hata varsa 400, sunucu işlemlerinde sorun olduysa da 500 döndürülüyor. Ayrıca hata durumunda hatayı tanımlayan bir yazı da döndürülmektedir.

Sunucudan yanıt geldikten sonra da React Native uygulaması bir hata olup olmadığını bu durum koduna bakarak anlamaktadır. Eğer ki durum kodu 200 ise resmi ve nesne sayısını, hata varsa da gelen hata mesajını ekrana bastırmaktadır.

Tüm işlemlerin sonunda da yükleme ekranı kaldırılıyor ve resmin işlendiği bilgisi kaydedilip onay butonunun ekrandan silinmesi sağlanıyor.

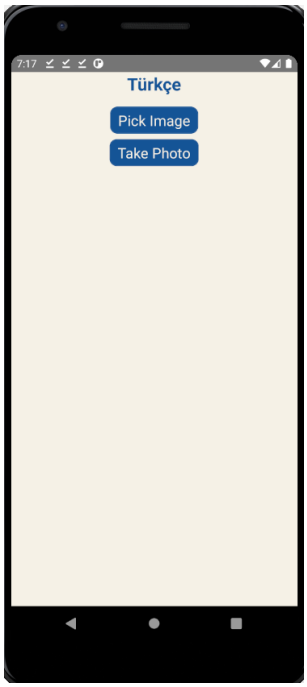
Duruma bağlı göstermeler için ternary operator'dan faydalanıldı.

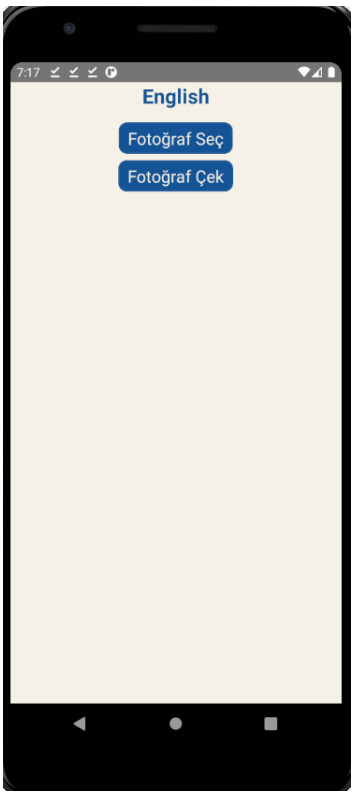
Sonuçlar

Projede belirtilen tüm isterler başarıyla tamamlanmıştır. Uygulama sorunsuz şekilde çalışmaktadır. Denenen birden fazla farklı biçimdeki görselde nesne tanımlamaları başarıyla yapılıp, hem görsel üzerinde gösterildiği hem de nesne sayısının yazdırıldığı görülmüştür.

Ayrıca herhangi bir hata olması durumunda da hem backend hem de frontend'de bu hatanın ekranda gösterildiği tespit edilmiştir. Backend, Google App Engine'de çalıştığı için kayıtların da düzgün tutulduğundan emin olunmuştur.

Çıktılar

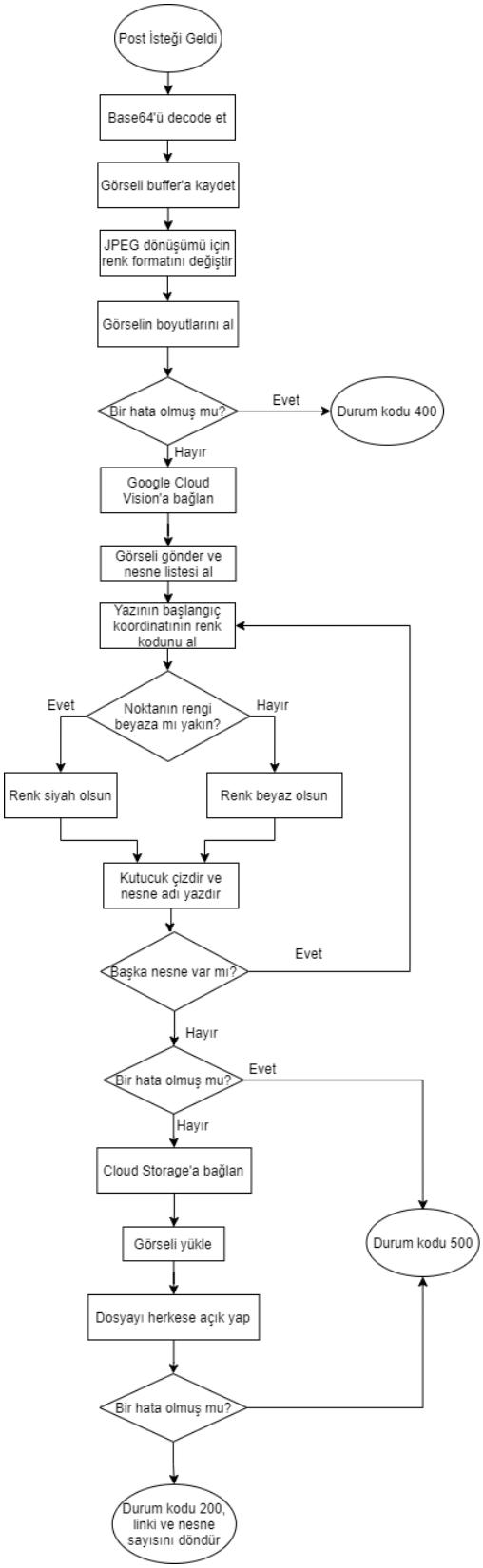





7:19
English
Fotoğraf Seç
Fotoğraf Çek
4 nesne tespit edildi.
SPACE. FOR ALL.
Person
Person
Person
BBC
DOCTOR WHO
COMING SOON


7:19
Türkçe
Pick Image
Take Photo
10 objects found.
BBC
DOCTOR WHO
Person
Person
Outerwear
Person
Pants
12
12

Şema


Post İsteği Geldi
Base64'ü decode et
Görseli buffer'a kaydet
JPEG dönüşümü için renk formatını değiştir
Görselin boyutlarını al
Bir hata olmuş mu?
Evet
Durum kodu 400
Hayır
Google Cloud Vision'a bağlan
Görseli gönder ve nesne listesi al
Yazının başlangıç koordinatının renk kodunu al
Noktanın rengi beyaza mı yakın?
Evet
Hayır
Renk siyah olsun
Renk beyaz olsun
Kutucuk çizdir ve nesne adı yazdır
Başka nesne var mı?
Evet
Hayır
Bir hata olmuş mu?
Evet
Hayır
Cloud Storage'a bağlan
Görseli yükle
Dosyayı herkese açık yap
Bir hata olmuş mu?
Durum kodu 500
Durum kodu 200, linki ve nesne sayısını döndür

Kaynakça

https://flask.palletsprojects.com/en/1.1.x/quickstart/

https://cloud.google.com/vision/docs/object-localizer

https://cloud.google.com/storage/docs/uploading-objects#storage-upload-object-code-sample

https://cloud.google.com/appengine/docs/standard/python3/quickstart

https://cloud.google.com/appengine/docs/standard/python3/building-app/writing-web-service

https://docs.python.org/3/library/base64.html

https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/

https://stackoverflow.com/questions/45122994/how-to-convert-base64-string-to-a-pil-image-object

https://stackoverflow.com/questions/43258461/convert-png-to-jpeg-using-pillow

https://pillow.readthedocs.io/en/stable/reference/ImageDraw.html

https://www.flaskapi.org/api-guide/status-codes/

https://en.wikipedia.org/wiki/Relative_luminance

https://stackoverflow.com/questions/596216/formula-to-determine-brightness-of-rgb-color

https://www.programiz.com/python-programming/datetime/current-datetime

https://reactjs.org/docs/hooks-state.html

https://stackoverflow.com/questions/33468746/whats-the-best-way-to-get-device-locale-in-react-native-ios

https://github.com/react-native-image-picker/react-native-image-picker

https://reactnative.dev/docs/animated

https://reactnative.dev/docs/animations

https://reactnative.dev/docs/pressable

https://reactjs.org/docs/hooks-reference.html