IOS İLE MOBİL UYGULAMA GELİŞTİRME II
FİNAL RAPORU

GitHub Link: https://github.com/MetehanTRN/IOS_PROJECT

METEHAN TURAN

20291263

1. GİRİŞ

Bu proje, araç plakalarının otomatik olarak tanınmasını, veritabanı üzerinde kontrol edilmesini ve kullanıcıya görsel bildirimlerle bilgi verilmesini sağlayan, mobil tabanlı bir plaka tanıma sistemidir. Sistem hem kayıtlı plaka yönetimi hem de kara liste işlemlerini desteklemektedir. Plaka tanıma işlemleri Platerecognizer API üzerinden gerçekleştirilmekte olup, tüm veriler Firebase Firestore üzerinde tutulmaktadır.

2. KULLANILAN TEKNOLOJİLER

- React Native (mobil arayüz)
- Firebase (authentication ve Firestore veritabanı)
- Platerecognizer API (plaka tanıma)
- Node.js (otomatik dosya işlemleri)
- Expo (uygulama yönetimi)
- React Navigation (sayfalar arası geçiş)
- TypeScript (tip güvenliği ve okunabilirlik için)

3. PROJE DOSYA YAPISI

Projede assets, components, contexts, screens ve utils gibi klasör yapısı kullanılmıştır.

- **assets:** Test resimleri ve uygulama ikonları.
- **components:** AddPlateModal, EditPlatesModal, LogModal, BlackListModal gibi bileşenleri içerir.
- **contexts:** Uygulama teması gibi global verileri içerir.
- **screens:** AuthLoading, HomeScreen ve LoginScreen gibi ekranları içerir.
- **utils:** authStorage gibi yardımcı fonksiyonları içerir.

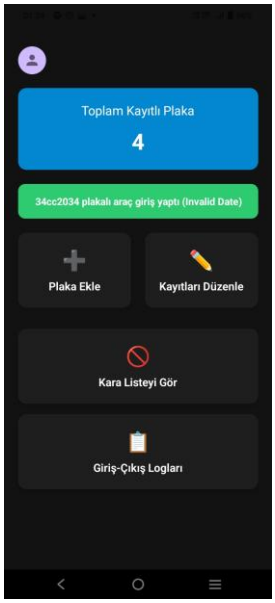
4. UYGULAMA AKIŞI

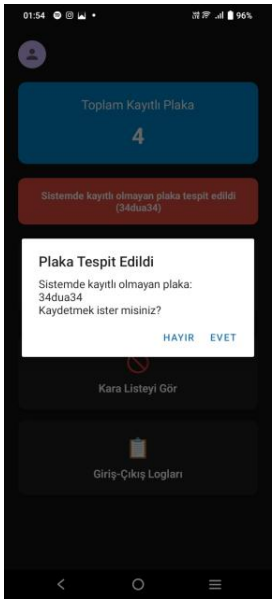
Uygulama açıldığında sistem otomatik olarak klasördeki tüm test resimlerini işler ve her plaka için şu adımları uygular:

- Plaka tanıma yapılır
- Kara liste kontrolü yapılır
- Kayıtlı plaka kontrolü yapılır
- Duruma göre kullanıcıya alert gösterilir veya log kaydı oluşturulur
- Gerekirse kayıtlı plaka modalı ya da kara liste modalı açılır

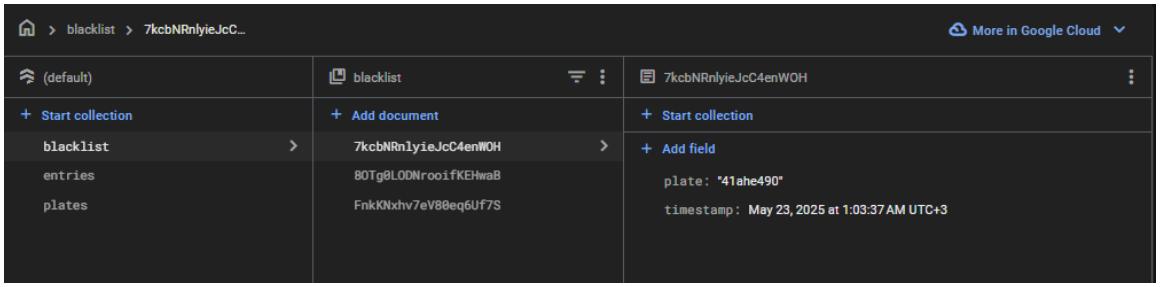
5. UYGULAMAYA EKLENEN ANA ÖZELLİKLER

1. **Otomatik Resim Tanıma**
   Uygulama açıldığında, 'assets/test-plates/' klasöründeki tüm resimler otomatik olarak tanınır. Her resim `recognizeSamplePlates()` fonksiyonunda base64'e çevrilerek Platerecognizer API ile analiz edilir. Aynı resimlerin tekrar tekrar işlenmemesi için `processedImages` adlı Set yapısı kullanılmıştır.

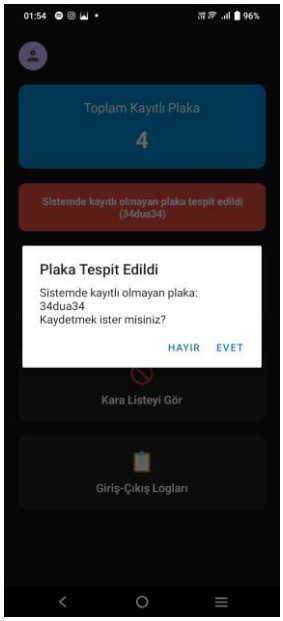




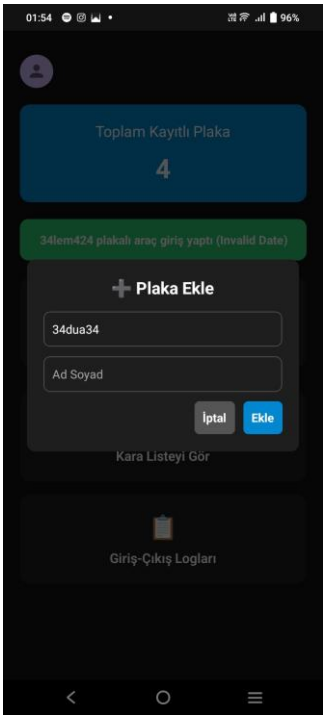
2. **Firebase Firestore Veritabanı Bağlantısı**
   Tüm veriler Firebase Firestore üzerinde saklanır. 'plates', 'entries' ve 'blacklist' adında üç koleksiyon kullanılır. Her plaka tanındığında sistem bu koleksiyonlarda gerekli sorgulamaları yapar ve veri yazma işlemleri gerçekleştirir.



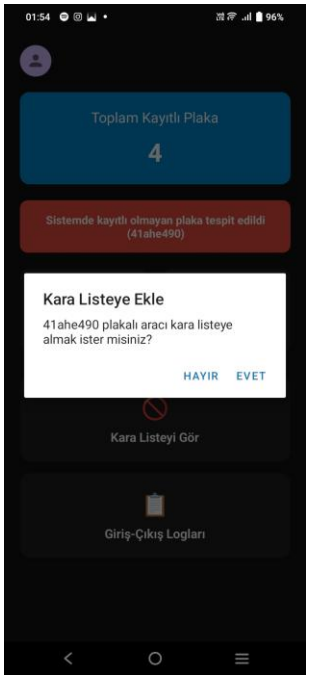
3. **Kayıtlı Olmayan Plaka İçin Kullanıcıdan Onay İsteme**
   Tanınan plaka sistemde kayıtlı değilse kullanıcıya alert ile bu plakayı kaydetmek isteyip istemediği sorulur. Bu işlem, kullanıcı kontrolünü merkeze alır ve hatalı veri kaydını engeller.



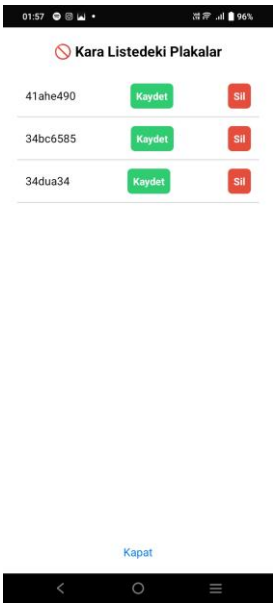
4. **AddPlateModal ile Plaka Kayıt Sistemi**
   Kullanıcı 'Evet' dediğinde AddPlateModal bileşeni açılır ve plakayı kaydetmesi sağlanır. Modalın içindeki plaka alanı otomatik olarak önceden tanınan plaka ile doldurulur (`prefillPlate`).



5. **Kara Listeye Ekleme ve Kontrol Sistemi**
   Kullanıcı 'Kaydetmek istemiyorum' dediğinde sistem bu plakayı kara listeye almayı önerir. Onay verilirse plaka sadece bir kez kara listeye eklenir. Firebase'de 'blacklist' koleksiyonuna kayıt yapılır.



6. **Kara Liste Ekranı (Silme ve Kaydetme Butonlarıyla)**
   Kara liste butonuna tıklanınca açılan modal, tüm kara listedeki plakaları listeler. Her plakanın yanında 'Kara Listeden Çıkar' ve 'Kayıtlı Plaka Olarak Ekle' butonları bulunur. Her işlem için kullanıcıdan tekrar onay istenir.



7. **Kara Listedeki Plakaları İşleme Dışı Bırakma**
   Plaka tanındığında, eğer o plaka daha önceden kara listeye alınmışsa, sistem bu plakayı otomatik olarak es geçer. Bu kontrol `blacklistedPlates.includes(plate)` ile yapılır. Böylece sistem daha hızlı ve hatasız çalışır.

8. **testImages.json ve imageMap.ts Otomatik Oluşturma Scripti**
   Klasöre manuel olarak görsel ekledikten sonra bu görsellerin otomatik olarak sisteme tanıtılması için bir Node.js script yazılmıştır. 'npm start' komutu ile hem JSON hem de `require` map dosyaları dinamik olarak oluşturulur.

```
JS generateImageAssets.js > ...
const fs = require('fs');
const path = require('path');

const imagesDir = path.join(__dirname, 'assets', 'test-plates');
const jsonOutput = path.join(imagesDir, 'testImages.json');
const mapOutput = path.join(__dirname, 'imageMap.ts');

fs.readdir(imagesDir, (err, files) => {
  if (err) {
    console.error('❌ Klasör okunamadı:', err);
    process.exit(1);
  }

  const imageFiles = files.filter((file) =>
    /\.(jpg|jpeg|png)$/i.test(file)
  );

  // ✅ 1. JSON dosyasını yaz
  fs.writeFileSync(jsonOutput, JSON.stringify(imageFiles, null, 2));
  console.log('✅ testImages.json oluşturuldu.');

  // ✅ 2. imageMap.ts dosyasını oluştur
  const importLines = imageFiles.map((file, index) =>
    `const img${index} = require('./assets/test-plates/${file}');`
  ).join('\n');

  const exportLines = `export const imageMap: Record<string, any> = {\n` +
    imageFiles.map((file, index) =>
      `  "${file}": img${index},`
    ).join('\n') +
    `\n};`;
```

```
TS imageMap.ts > ...
const img0 = require('./assets/test-plates/plate1.jpg');
const img1 = require('./assets/test-plates/plate2.jpg');
const img2 = require('./assets/test-plates/plate3.jpg');
const img3 = require('./assets/test-plates/plate4.jpg');
const img4 = require('./assets/test-plates/plate5.jpg');
const img5 = require('./assets/test-plates/plate6.jpeg');

export const imageMap: Record<string, any> = {
  "plate1.jpg": img0,
  "plate2.jpg": img1,
  "plate3.jpg": img2,
  "plate4.jpg": img3,
  "plate5.jpg": img4,
  "plate6.jpeg": img5,
};
```

9. **Aynı Görselin Tekrar İşlenmesini Engelleme**
   İşlenen görsellerin URI bilgisi `processedImages` adlı Set yapısında tutulur. Böylece döngüde aynı görsel ikinci kez işlenmez, API kullanımı ve sistem yükü azaltılmış olur.

SONUÇ:

Bu proje, plaka tanıma teknolojisinin mobil bir uygulama ile nasıl entegre edilebileceğini göstermektedir. Sistem, kullanıcı müdahalesi minimum olacak şekilde otomatik çalışmaktadır. Kullanıcı dostu arayüzü ve veri yönetimi özellikleriyle uygulama genişletilmeye uygundur. İlerleyen sürümlerde gerçek zamanlı kamera bağlantısı, admin paneli ve plaka istatistikleri gibi özellikler entegre edilebilir.