**Flutter Access Projesi - Uygulama Raporu**

Flutter Uygulama Geliştirme Projesi - Ara Sınav Uygulama Ödevi

Proje Konusu: Servise Gelen Araçlar Takip Sistemi

Projemizin amacı:

Bu projenin temel amacı, servise gelen araçların kayıt altına alınması, görüntülenmesi ve yönetilmesini sağlayan bir takip sistemi geliştirmektir. Gerçek hayattaki küçük ve orta ölçekli oto servis işletmelerinin dijital dönüşüm sürecine katkı sunmak hedeflenmiştir.

Projede, kullanıcı dostu bir mobil uygulama arayüzü ile araç bilgileri sisteme kolayca eklenebilir, mevcut araç kayıtları görüntülenebilir ve istenmeyen kayıtlar silinebilir. Tüm bu işlemler, Flutter ile geliştirilen mobil uygulama aracılığıyla gerçekleştirilirken, veriler bir Microsoft Access veritabanında saklanır. Veritabanına erişim ve işlem yapılması ise Python Flask web servisi üzerinden sağlanır.

📘 Bu Projeyi Tamamlayan Ne Öğrenir?

Bu proje bittikten sonra Bu projeyi başarıyla tamamlayan biri, yazılım geliştirme sürecine dair uçtan uca deneyim kazanmış olur. Mobil uygulama geliştirmede kullanılan **Flutter** teknolojisini temel seviyede öğrenmiş; sayfa yapıları, kullanıcıdan veri alma, butonlarla işlem tetikleme gibi konularda pratiğe sahip olmuştur. Ayrıca **backend geliştirme** konusunda da önemli bir adım atmış; **Python ve Flask** kullanarak veri alışverişi sağlayan bir web servis oluşturmayı öğrenmiştir. Proje kapsamında veritabanı bağlantısı kurarak gerçek bir **Microsoft Access** veritabanıyla çalışmış ve veri ekleme, listeleme, silme gibi temel işlemleri uygulamalı olarak yapmıştır. Bunun yanında bir frontend (Flutter) ile bir backend (Flask) arasında bağlantı kurarak **API üzerinden veri alışverişi** yapmayı öğrenmiştir. Tüm bu süreç boyunca proje oluşturma, dosya organizasyonu, kütüphane kurulumu, terminal komutları kullanımı gibi yazılımcıların sıkça karşılaştığı pratik konularda da ciddi bir deneyim edinmiştir. Bu proje sayesinde, yazılım geliştirmenin hem arayüz hem de sunucu tarafını görerek yazılım dünyasına sağlam bir adım atar.

Projenin Yapım Aşaması:

🔧 Gerekli Kurulumlar ve Amaçları:

Öncelikle aşağıda vereceğim bağlantıyla flutter in kendi sitesine gidin

(<https://flutter.dev/>)

Daha sonra ;

**1.Git Kurulumu**

📝 **Ne için kullanılır?**  
Kod versiyon kontrol sistemi olan Git, projeleri yedeklemek, yönetmek ve ekip içi işbirliği yapmak için kullanılır. Ayrıca Flutter SDK kurulurken arkada Git kullanılır, bu yüzden gereklidir.

🔧 **Kurulum Adımları:**

1. Git’in resmi web sitesine gidin:  
   👉 <https://git-scm.com/>
2. Windows için olan kurulumu indirin ve standart ayarlarla ilerleyerek tamamlayın.
3. Kurulum tamamlandıktan sonra terminal açıp şu komutu yazın:

git –version

Bu komutun ardından Git versiyon bilgisi çıkıyorsa, kurulum başarıyla tamamlanmış demektir

**2️.Android Studio Kurulumu**

📝 **Ne için kullanılır?**  
Flutter uygulamalarını Android cihaz veya emülatör üzerinde test etmek için gerekli olan **Android SDK**, **AVD Manager** ve diğer bileşenler Android Studio ile birlikte gelir.

🔧 **Kurulum Adımları:**

1. Android Studio’nun resmi sitesine gidin:  
   👉 <https://developer.android.com/studio>
2. İşletim sisteminize uygun olan kurulumu başlatın.

Burada önemli husus hiçbir ayarına dokunmadan hepsini next seçeneğini tıklayıp yolumuza devam etmek

3.Visual Studio Code (VS Code) Kurulumu

📝 Ne İçin Kullanılır?

Flutter projeleri için kod yazmak ve geliştirme ortamını yönetmek için Visual Studio Code (VS Code) kullanılır. VS Code, hafif, hızlı ve özelleştirilebilir bir editördür. Ayrıca, Flutter için birçok eklenti sunarak geliştirme sürecini kolaylaştırır.

🔧 Kurulum Adımları:

1. Visual Studio Code’un Resmi Sitesine Gidin: 👉 [Visual Studio Code İndir](https://code.visualstudio.com/)
2. İşletim Sisteminize Uygun Olan Kurulumu Başlatın:
   * Açılan sayfada Windows, Mac veya Linux işletim sisteminize uygun olan kurulum dosyasını indirin.
3. Kurulum Adımları:
   * İndirilen kurulum dosyasını çalıştırın.
   * Kurulum sihirbazı ile gelen adımları takip ederek kurulumu başlatın.
   * Kurulum sırasında herhangi bir ayar yapmanıza gerek yoktur, tüm varsayılan seçeneklerle Next butonuna basarak kurulumu tamamlayabilirsiniz.

**4. Flutter SDK Kurulumu (VS Code Üzerinden)**

📝 **Neden Önemlidir?**  
Flutter SDK, Flutter uygulamalarını derleyip çalıştırmak için gereken temel yapı taşlarını içerir. Yani uygulamanın motorudur. Bu olmadan Flutter projesi oluşturulamaz ya da çalıştırılamaz.

🧭 **Kurulum İçin İki Yol Sunulur:**  
Kurulum sırasında karşımıza iki seçenek çıkar:  
✅ **Use VS Code to install Flutter SDK** (Otomatik Kurulum)  
🔽 **Download and install manually** (Manuel Kurulum)

🔧 **Biz Hangi Yolu Seçtik?**  
✨ Hem Visual Studio Code kullandığımız için,  
✨ Hem de bu yol çok daha pratik ve hızlı olduğu için,  
ilk seçeneği yani:  
✅ **“Use VS Code to install Flutter SDK”**  
seçeneğini tercih ettik.

🚀 **Kurulum Adımları:**

1. **VS Code'u açın.**
2. **Sol taraftaki 🧩 Extensions (Eklentiler) simgesine tıklayın.**
3. **Flutter eklentisini kurun.** (Bu işlem sırasında Dart eklentisi de otomatik olarak kurulur.)
4. **Kurulum tamamlandığında, VS Code ekranının üst kısmında aşağıdaki gibi bir uyarı görebilirsiniz:**  
   ⚠️ **Flutter SDK not found**  
   Bu durumda:  
   🟢 **"Download SDK"** butonuna tıklayın.  
   VS Code, sizin yerinize SDK’yı indirip, kurulumunu yapacaktır.
5. **Flutter SDK'nın kurulum dizinini belirtmek önemlidir.**  
   Flutter SDK'nızı yüklerken, önerilen dizin olarak **C:\dev** klasörünü seçmeniz önerilir. Burada dikkat edilmesi gereken nokta:  
   📌 **Yolda boşluk veya Türkçe karakter kullanmamanız gerekmektedir.** Bu, olası hataların önüne geçmek için önemlidir.
6. **Kurulum tamamlandığında, "Add SDK to PATH" seçeneğine tıklayın.**  
   Burada, Flutter SDK'nın kurulduğu dizini **PATH**'e ekleyerek Flutter'ı sistemde her yerden erişilebilir hale getirebilirsiniz.

Kurulum başarıyla tamamlandığında, artık Flutter projeleri oluşturup kendi projenizi geliştirmeye başlayabilirsiniz!

💡 **Ekstra Not:**  
Eğer bu otomatik kurulum yolu herhangi bir sebeple başarısız olursa,  
ikinci seçeneği (manuel kurulum) kullanarak SDK'yı Flutter Resmi Sitesi üzerinden indirip, kurabilirsiniz.

5️. Visual Studio Kurulumu (Windows Uygulama Geliştirme İçin)

📝 Neden Önemlidir?  
Windows platformunda uygulama geliştirmek istiyorsanız, Visual Studio vazgeçilmez bir araçtır. Özellikle Flutter ile masaüstü uygulamaları geliştirmek için Visual Studio, C++ desteğiyle Windows masaüstü geliştirme için gerekli olan SDK'lar ve araçları sağlayan güçlü bir IDE'dir. Flutter projenizin Windows üzerinde çalışması için Visual Studio kurulumu şarttır.

🚀 Kurulum Adımları

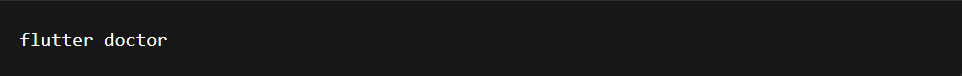
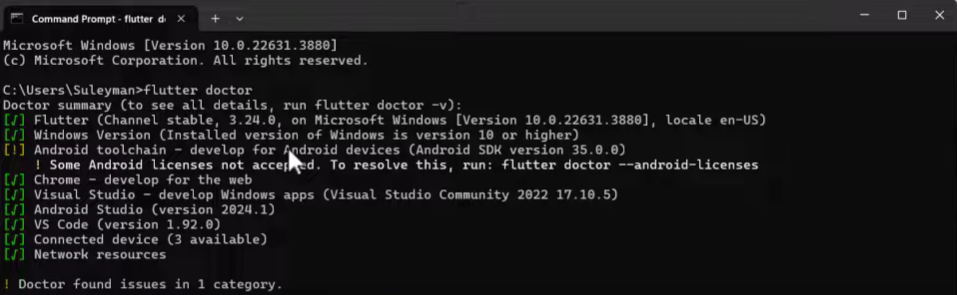
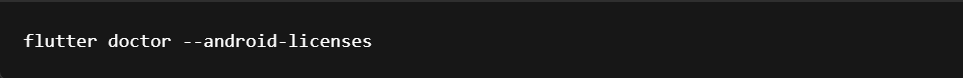
1. Visual Studio Resmi Sitesine Gitmek İçin:  
   İlk adım olarak, Visual Studio'nun en güncel sürümünü indirmek için [resmi siteyi](https://visualstudio.microsoft.com/) ziyaret edin.  
   👉 Buradan Visual Studio'nun en son sürümünü kolayca edinebilirsiniz.
2. Topluluk (Community) Sürümünü İndirmek:  
   İndirilecek sürüm Topluluk (Community) sürümüdür. Bu sürüm tamamen ücretsiz olup, kişisel kullanım ve açık kaynak projeleri için uygundur.  
   👉 "İndir" butonuna tıklayın ve indirme işlemini başlatın.
3. Kurulum Dosyasını Çalıştırın:  
   İndirilen Visual Studio kurulum dosyasını çalıştırarak kuruluma başlayın. Bu işlem sırasında kurulum dosyası, gerekli bileşenleri internet üzerinden indirecektir.
4. Kurulumda Önemli Seçenekleri Seçin:  
   Kurulum ekranında, "Desktop development with C++" seçeneğini mutlaka işaretleyin. Bu, Windows masaüstü uygulama geliştirmek için gerekli olan tüm araçları içerecektir.  
   🔔 Uyarı: Eğer C++ desteğini işaretlemezseniz, Windows uygulamaları geliştiremeyebilirsiniz. Bu nedenle bu seçeneği işaretlemeyi unutmayın!
5. Kurulumun Tamamlanması:  
   Kurulum işlemi tamamlandığında, "Finish" butonuna tıklayarak kurulum işlemini sonlandırabilirsiniz. Bu aşamadan sonra, Visual Studio'yu açarak yeni projeler oluşturabilir veya mevcut projelerinize devam edebilirsiniz.
6. Kurulum Sonrası Yapılacaklar:
   * Visual Studio'yu Başlatın: Programı ilk açtığınızda, ana ekran üzerinden yeni projeler oluşturabilir ya da mevcut projeleri açabilirsiniz.
   * C++ ve Gerekli Araçlar: Windows için geliştirme yapabilmek için gerekli olan C++ araçları ve diğer kütüphaneler otomatik olarak kurulacaktır.

💡 Ekstra İpucu:  
Visual Studio, sadece Windows uygulamaları değil, aynı zamanda Flutter ile masaüstü uygulamaları geliştirebilmek için ihtiyacınız olan tüm altyapıyı sağlar. C++ desteği olmadan Windows platformunda uygulama geliştiremeyeceğiniz için, bu kurulum aşaması oldukça kritik bir adımdır. Visual Studio, güçlü hata ayıklama özellikleri, zengin eklenti desteği ve çok yönlü geliştirici araçlarıyla Flutter geliştirme sürecinizi çok daha verimli hale getirir. Flutter SDK ile uyumlu çalışarak, Windows üzerinde Flutter projelerinizi rahatça geliştirebilirsiniz.

**6️.Android SDK Kurulumu (Android Studio Üzerinden)**

📝 **Neden Önemlidir?**  
**Android SDK**, **Flutter uygulamalarını Android cihazlarında test edebilmek ve çalıştırabilmek** için gereklidir. Bu SDK'yı **Android Studio** üzerinden kolayca kurarak, Android platformunda uygulama geliştirmeye hazır hale gelirsiniz. **Flutter**'ı Android cihazlarda test edebilmek için SDK'nın doğru şekilde kurulması çok önemlidir.

🚀 **Kurulum Adımları**

1. **Android Studio'yu Açın:**  
   İlk olarak **Android Studio** uygulamasını açıyoruz. Eğer daha önce kurulum yapılmadıysa, **Android Studio**'yu [buradan](https://developer.android.com/studio) indirip kurabilirsiniz.
2. **SDK Manager'ı Açın:**
   * **Android Studio** açıldıktan sonra, **"More Actions"** seçeneğine tıklıyoruz.
   * Ardından, açılan menüde **"SDK Manager"** seçeneğini seçiyoruz. Bu işlem, **Android SDK** ve gerekli araçları yönetmemizi sağlar.
3. **SDK Kurulumunu Başlatın:**  
   SDK Manager açıldığında, **Edit** butonuna tıklıyoruz ve **SDK Tools** sekmesine geçiyoruz. Burada, **Android SDK** için gerekli olan tüm bileşenlerin listesini görmemiz mümkün.
4. **Ekstra Araçları Seçin ve Kurulumu Başlatın:**
   * **"Extras"** seçeneğine tıklıyoruz ve ardından **"Android SDK Command-line Tools"** seçeneğini işaretliyoruz. Bu araç, komut satırından Android SDK işlemlerini yönetebilmemizi sağlar.
   * Son olarak, **"Next"** butonuna tıklayarak kurulumu başlatıyoruz.
5. **Kurulumun Tamamlanması:**
   * Kurulum işlemi tamamlandığında, **"Accept"** butonuna tıklıyoruz.
   * Ardından, **"Next"** butonuna tıklayarak devam ediyoruz.
   * Kurulumun son aşamasında ise **"Finish"** butonuna tıklayarak işlemi tamamlıyoruz.
6. **Kurulumun Kontrol Edilmesi:**
   * Kurulumun eksiksiz ve doğru şekilde yapıldığını kontrol etmek için **Flutter Doctor** komutunu çalıştırıyoruz.
   * Komut satırında aşağıdaki komutu yazıyoruz:
   * Bu komut, tüm Flutter kurulumunu kontrol eder ve olası eksiklikleri veya hataları bildirir.
7. **Lisans Kabulü ve Tam Kurulum:**
   * **Android SDK**'nın kurulumunda bazı lisansların kabul edilmediğine dair bir uyarı alabiliriz.
   * 
   * Bu durumu çözmek için, aşağıdaki komutu çalıştırıyoruz:
   * 
   * Komut çalıştırıldığında, her bir lisans için **"yes"** diyerek lisans sözleşmelerini kabul ediyoruz.
8. **Son Kontrol:**
   * Lisansları kabul ettikten sonra, tekrar **flutter doctor** komutunu çalıştırıyoruz.
   * Bu sefer, **Android Toolchain** kısmının yeşil tik işaretiyle tamamlandığını ve kurulumun eksiksiz olduğunu görüyoruz.
   * Eğer her şey yeşil tikli ise, **Android SDK** kurulumu başarılı bir şekilde tamamlanmış demektir.

💡 **Ekstra İpucu:**  
Bu adımlar, **Flutter** ile Android uygulamaları geliştirmek için **Android SDK**'nın doğru bir şekilde kurulduğundan emin olmanızı sağlar. Eğer herhangi bir hata ile karşılaşırsanız, kurulum aşamalarını tekrar gözden geçirerek eksiklikleri tamamlayabilirsiniz. Kurulum tamamlandığında, **Flutter** projelerinizi Android cihazlar üzerinde test edebilir ve geliştirme sürecinize devam edebilirsiniz.

1. **1. VERİTABANI KURULUMU  
   📂 Microsoft Access ile Veritabanı Oluşturma Adımları:**
2. **Microsoft Access Programını Açıyoruz:**  
   İlk olarak, Microsoft Access programını açıyoruz.
3. **"Boş Veritabanı" Seçeneğine Tıklıyoruz:**  
   Açılış ekranında "Boş Veritabanı" seçeneğini tıklayın.
4. **Veritabanına Anlamlı Bir İsim Veriyoruz:**  
   Veritabanınıza anlamlı bir isim veriyoruz. Örneğin:  
   Database2.accdb
5. **Veritabanını Kaydetmek İstediğimiz Klasörü Belirliyoruz:**  
   Bu projede, veritabanını şu dizine kaydediyoruz:  
   C:\Users\enest\flutterProjesi
6. **"Oluştur" Butonuna Basarak Veritabanını Oluşturuyoruz:**  
   Son olarak, "Oluştur" butonuna basarak veritabanımızı oluşturuyoruz.

**Yeni Tablo Oluşturma ve Sütunları Ekleme:**

1. **Varsayılan Tabloya "ServiseGelenAraclar" İsmini Veriyoruz:**  
   Oluşan veritabanı içinde varsayılan olarak "Tablo1" isminde bir tablo oluşur.  
   Bu tabloyu, **servisegelenaraclar** adında yeniden adlandırıyoruz.
2. **Tablo Görünümünü "Tasarım Görünümü" Moduna Getiriyoruz:**  
   Access’te üst menüden "Tasarım Görünümü" modunu seçiyoruz.
3. **Aşağıdaki Sütunları, Türleriyle Birlikte Ekliyoruz:**  
   Şimdi aşağıdaki sütunları, veri türleriyle birlikte ekliyoruz:

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

 Not**:** AracID sütunu, Access tarafından otomatik artan birincil anahtar (primary key) olarak ayarlanacaktır.

 Tabloyu **Kaydediyoruz:**  
Tüm sütunları eksiksiz ekledikten sonra tabloyu kaydediyoruz.

**2. BACKEND KURULUMU (Flask ile API Geliştirme)**

📝 **Neden Önemlidir?**  
Backend, mobil uygulamanızın işlevselliğini sağlayan ve verileri yöneten bir bileşendir. Uygulamanızın veritabanı ile etkileşimde bulunabilmesi, verileri doğru şekilde alıp gönderebilmesi için backend yapılandırması gereklidir. Flask, bu tip uygulamalarda hızlı ve etkili bir şekilde API geliştirmeyi sağlayan popüler bir Python framework'üdür. Python dili ve Flask kullanarak, mobil uygulamanız için sağlam bir backend kurabilirsiniz.

🚀 **Kurulum Adımları**

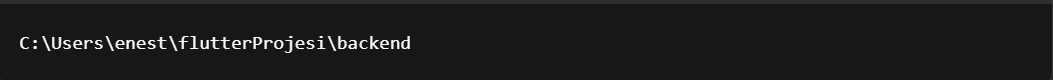
**1. Backend Klasörü Oluşturma 📁**

İlk olarak, projenin ana dizininde bir **backend** klasörü oluşturmak önemlidir. Bu klasör, uygulamanızın backend kodlarını ve veri tabanı dosyalarını düzenli tutar. Klasörlerin doğru bir şekilde yapılandırılması, projenin yönetilebilir olmasını sağlar.

**Backend Klasörünün Amacı 📂:**

* **Proje içinde farklı bileşenleri düzenli bir şekilde tutar.**
* **Backend kodlarının ve veritabanı dosyalarının bir arada bulunmasını sağlar.**

Klasör yolu:



**Klasör Yapısı 📁:**

1. **FlutterProjesi** klasörünün içine **backend** adında bir klasör oluşturun.
2. Bu klasörün içinde **app.py** adında bir dosya oluşturun ve gerekli backend kodlarını bu dosyaya yazın.

**Not**: Veritabanı dosyasının da **backend** klasöründe olması, tüm backend bileşenlerinin tek bir yerde düzenli olmasını sağlar.

Daha sonra vereceğim kodu app.py ye yapıştırın

app.py kodu:

from flask import Flask, request, jsonify

from flask\_cors import CORS

import pyodbc

app = Flask(\_\_name\_\_)

CORS(app)

conn\_str = r'DRIVER={Microsoft Access Driver *(*\**.*mdb, \**.*accdb*)*};DBQ=C:\\Users\\enest\\flutterProjesi\\Database2*.*accdb;'

conn = pyodbc.connect(conn\_str)

cursor = conn.cursor()

@app.route('/add\_car', *methods*=['POST'])

def add\_car():

    data = request.json

    cursor.execute("""

        INSERT INTO servisegelenaraclar

        (AracAdi, AracMarka, AracModel, UretimYili, Plaka)

        VALUES (?, ?, ?, ?, ?)

    """, data['AracAdi'], data['AracMarka'], data['AracModel'], data['UretimYili'], data['Plaka'])

    conn.commit()

    return jsonify({'message': 'Araç eklendi.'})

@app.route('/list\_cars', *methods*=['GET'])

def list\_cars():

    cursor.execute("SELECT \* FROM servisegelenaraclar")

    rows = cursor.fetchall()

    cars = []

    for row in rows:

        cars.append({

            "AracID": row.AracID,

            "AracAdi": row.AracAdi,

            "AracMarka": row.AracMarka,

            "AracModel": row.AracModel,

            "UretimYili": str(row.UretimYili),

            "Plaka": row.Plaka

        })

    return jsonify(cars)

@app.route('/delete\_car/<int:car\_id>', *methods*=['DELETE'])

def delete\_car(*car\_id*):

    cursor.execute("DELETE FROM servisegelenaraclar WHERE AracID = ?", *car\_id*)

    conn.commit()

    return jsonify({'message': 'Araç silindi.'})

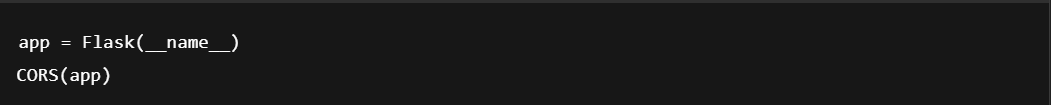
if \_\_name\_\_ == '\_\_main\_\_':

    app.run(*debug*=True)

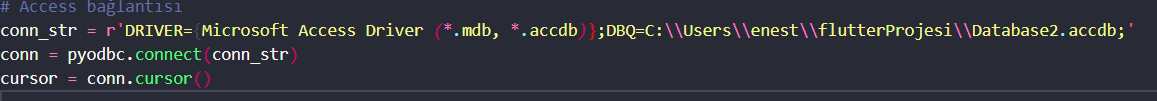
Flask Uygulamasının Genel Yapısı ve Açıklamaları 🖥️

Bu Python kodu, Flask framework'ü kullanarak basit bir web uygulaması oluşturur ve veritabanı işlemleri yapar. Bu uygulama, kullanıcıların araç verilerini eklemesine, listelemesine ve silmesine olanak tanır. Uygulama, Microsoft Access veritabanı kullanır ve veritabanı işlemleri için pyodbc modülünü kullanır. Aşağıda her bir bölüm açıklanmıştır:

**1. Flask Uygulamasını Başlatma ve CORS Ayarı** 🔧

****

2. Veritabanı Bağlantısı 🔗

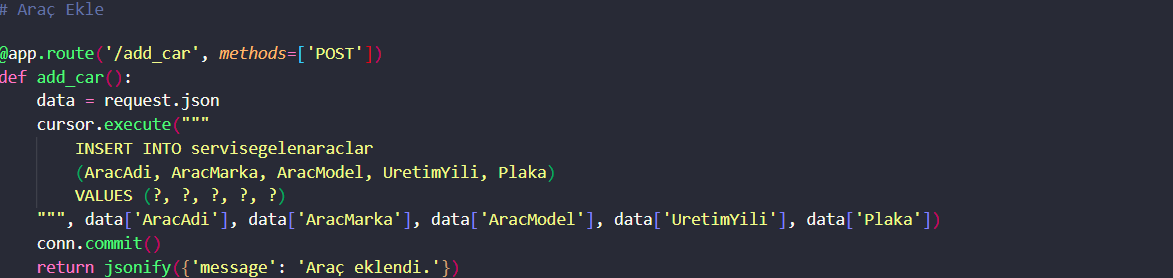
****

 conn\_str: Microsoft Access veri tabanına bağlanmak için kullanılan bağlantı dizesi (connection string). Bu bağlantı, veri tabanının tam yolunu belirtir.

 conn: Veri tabanı bağlantısı kurulur.

 cursor: Veri tabanı sorguları çalıştırmak için kullanılan bir cursor nesnesidir.

**3. Araç Ekleme (POST)** 🚗➕



**4.Araç Listeleme (GET) 🚗📋**

**metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**5.Araç Silme (DELETE) 🚗❌**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**2. Python Sanal Ortamı (venv) Kurulumu 🌱**

Sanal ortam (venv), Python projelerinde kullanılan kütüphaneleri izole etmek için önemli bir adımdır. Bu, proje bağımlılıklarının sistem genelinde değil, yalnızca proje dizini içinde yönetilmesini sağlar. Böylece, başka projelerde farklı kütüphane sürümleri kullanılabilir ve bu projelerin birbirine etkisi engellenir. Aynı zamanda, sanal ortam oluşturmak, projenin taşınabilirliğini artırır.

**Neden sanal ortam kurmalıyız?**

* **Bağımlılık yönetimi:** Projeye özel kütüphaneler sadece sanal ortamda bulunur, böylece sistemdeki diğer projelerle karışmaz.
* **Taşınabilirlik:** Proje taşınırken veya başka bir sistemde çalıştırılırken aynı ortamı yaratmak çok daha kolaydır.
* **Sistem temizliği:** Sanal ortam sayesinde sistem genelinde gereksiz kütüphane yüklemesi yapmaktan kaçınırız.

**Sanal ortamı kurma adımları:**

* **Backend klasörüne gidin:**
* Terminale Şunu yazın:
* metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

  Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

✅ Aktifleştirdikten sonra terminalde (venv) ifadesi görünmelidir. Bu, sanal ortamın aktif olduğunu gösterir.

**⚠️ Not: Sanal Ortam Neden Aktif Edilmelidir?**

Sanal ortamı aktifleştirmezsek:

* Kütüphaneler sistem genelinde kurulur, bu da diğer projelerle çakışmalara yol açabilir.
* pip ile yüklenecek olan Flask, pyodbc, flask\_cors gibi modüller global sistem yerine **bu projeye özel** ortamda kurulmalıdır.
* Projeyi bir başkası çalıştırmak istediğinde, ortamda hangi kütüphanelerin gerektiği requirements.txt ile belirtilebilir. Bu da sadece sanal ortamda işe yarar.

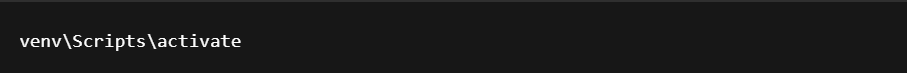
🧩 3. Gerekli Kütüphaneleri Kurma

Sanal ortam aktifleştirildikten sonra, terminale şu komutları yazarak gerekli paketleri yüklüyoruz:



💡 Bu komutlar sayesinde app.py dosyamız için gereken tüm bağımlılıklar yüklenmiş olur.

💡 Sanal Ortamı Aktif Etmek:

Eğer Windows kullanıyorsan ve terminalin backend klasörünün içindeyse, şu komutu yaz:

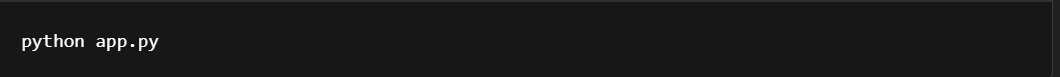
Başarılı bir şekilde aktif ettiğinde terminal satırının başında şöyle bir şey göreceksin:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

🚀 Son Adım: Sunucuyu Başlatmak:

Kütüphaneler başarıyla kurulduysa, artık Flask sunucumuzu başlatabiliriz. Bunun için:



Bu komutu terminale yazdıktan sonra aşağıdaki gibi bir çıktı alırsan, her şey doğru çalışıyor demektir:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu noktadan sonra:

* Tarayıcından bu adresi ziyaret edebilirsin.
* Flutter uygulamandan bu API’ye veri gönderebilir veya veri çekebilirsin.

Harika! 🚀 Şimdi Flutter kurulumuna geçiyoruz. Backend kısmını başarıyla tamamladık ve bağlantıyı kurmak için gerekli olan her şeyi yapılandırdık. Şimdi, **Flutter uygulamamızı** geliştirmek için gerekli kurulumları yapalım ve frontend tarafını oluşturmak için adımlarımıza devam edelim.

**🛠️** 2. Flutter Projesi Oluşturma

1.Flutter Projeyi Oluşturun:

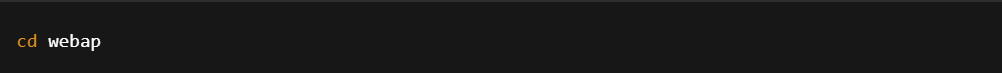
1.Flutter Komutuyla Proje Oluşturma: Visual Studio Code veya terminal kullanarak Flutter projesini oluşturmak için terminali açın ve aşağıdaki komutu yazın:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu komut, webap isminde yeni bir Flutter projesi oluşturacaktır.

2.**Proje Klasörüne Gidin**: Proje oluşturulduktan sonra, projenizin klasörüne geçiş yapmak için şu komutu kullanın:

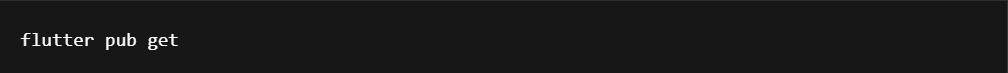


**🚀 Projeyi Çalıştırma:**

Proje oluşturduktan sonra, **Flutter** uygulamasını başlatmak için şu adımları izleyin:

1. **Flutter Dokümanlarını Güncelleyin**:

Terminal üzerinden bağımlılıkları güncellemek için şu komutu kullanın



flutter pub get komutu, Flutter projesinde **pubspec.yaml** dosyasında belirtilen bağımlılıkları (kütüphaneleri ve paketleri) indirir ve projeye ekler. Bu komut, gerekli olan tüm dış kütüphaneleri Flutter projenize yükleyerek çalışmaya hazır hale getirir. Yani, projede kullanılan tüm paketleri ve kütüphaneleri projenize entegre eder ve bunların doğru şekilde çalışmasını sağlar.

**📑 4. pubspec.yaml Dosyasını Düzenlemek**

Şimdi **pubspec.yaml** dosyasını açalım. Bu dosya, projede kullanılan bağımlılıkları içerir. Geliştirme sırasında bazı yeni bağımlılıkları eklemeniz gerekebilir.

Örneğin, **http** paketini kullanarak API ile iletişim kurmak için aşağıdaki gibi bir bağımlılık ekleyebilirsiniz.

1. **pubspec.yaml** Dosyasını Açın:

VS Code veya kullandığınız herhangi bir metin editörü ile **pubspec.yaml** dosyasını açın.

1. dependencies dosyasını şu şekilde değiştirin

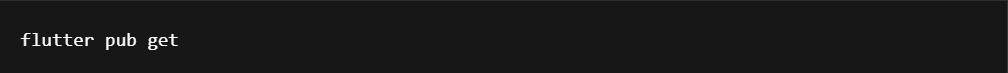
dependencies:

  flutter:

    sdk: flutter

  http: ^0.13.0

3.Sonra tekrardan üstte çalıştırdığımız;



Komutunu yazın.

Bu işlem, yeni eklediğiniz bağımlılıkları yükler ve projenizi günceller.

**🚀 Herşey Mükemmel Bir Sıkıntı Yok Hadi Başlatalım**

**📦 5. Flutter Projesini Çalıştırmak**

Tüm bağımlılıkları yükledikten ve projenizi güncelledikten sonra, Flutter projenizi çalıştırmaya hazırsınız. Terminalde şu komutu kullanarak projenizi çalıştırabilirsiniz ama ondan önce terminali ikiye bölün ve ol terminalde venv i aktif edin

Daha Sonra:



Bu komut, uygulamanızı çalıştıracak ve emülatör veya bağlı cihazda uygulamanızın açılmasını sağlayacaktır.

Ve karşınızda şöyle bir ekran var:

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu adımdan sonra 2 yani CHROME üzerinden göster demeniz gerekiyor

🎉 **Ve İşte İlk Flutter Uygulamanız! Hayırlı Olsun! 🚀** 🎉

🛠️ Flutter ile ilk adımlarınızı başarıyla attınız ve projenizi çalıştırdınız! Bu, sadece bir başlangıç. Artık Flutter dünyasına girişi tamamladınız ve yolculuğunuzda yeni uygulamalar geliştirmek için hazır durumdasınız.

İlk **Flutter uygulamanız**ı başarıyla başlattınız. Şimdi dikkat edin, sağ altta **+ butonunu** göreceksiniz. Her tıkladığınızda ekranda **artmaya başlayan sayılar** ile etkileşim kurarak uygulamanın dinamikliğini gözlemleyeceksiniz. Fakat biz burayı değiştiricez ve size birazdan söyleyeceğim işlemleri yapacağız

🛠️  **Lib Dosyasına 3 Dart Dosyası Ekleyin 📂**

Şimdi projemizde düzenlemeler yapmaya devam ediyoruz. Flutter uygulamanızı organize etmek adına **lib** klasörünün içine **3 yeni Dart dosyası** ekleyeceğiz. Bu dosyalar, uygulamanızın farklı işlevlerini düzenlemek için gerekli olacak.

**Adım Adım Dosya Oluşturma:**

1. **Lib Klasörüne Gitmek:**
   * VS Code'da veya terminalde projenizin **lib** klasörüne gidin.
2. **Yeni Dart Dosyalarını Oluşturun:**
   * **lib** klasörüne sağ tıklayın ve **New File** (Yeni Dosya) seçeneğine tıklayın.
   * Ardından aşağıdaki dosya isimlerini sırasıyla ekleyin:
     1. **arac\_ekle.dart** – Bu dosya araç ekleme işlemleri için kullanılacak.
     2. **arac\_sil.dart** – Bu dosya araç silme işlemlerini gerçekleştirecek.
     3. **arac\_listele.dart** – Bu dosya araçları listeleme işlemi yapacak.

**Dosya İsimleri:**

* **arac\_ekle.dart 🚗**
* **arac\_sil.dart 🗑️**
* **arac\_listele.dart 📋**
* **main.dart 🎯**

şimdi main.dartımızın içindeki kodları silin ve vereceğim kodları tekrardan main.dart dosyasına yapıştırın:

import 'package:flutter/material.dart';

import 'arac\_ekle.dart';

import 'arac\_listele.dart';

import 'arac\_sil.dart';

void main() {

  runApp(MaterialApp(

*title*: 'Servise Gelen Araçlar',

*debugShowCheckedModeBanner*: false,

*theme*: ThemeData(*primarySwatch*: Colors.blue),

*home*: AnaSayfa(),

  ));

}

class AnaSayfa extends StatelessWidget {

*@*override

  Widget build(BuildContext *context*) {

    return Scaffold(

*backgroundColor*: Colors.blue[50],

*appBar*: AppBar(

*title*: Text(

          'Araç Servis Takip Sistemi',

*style*: TextStyle(

*fontSize*: 30,

*fontWeight*: FontWeight.bold,

*color*: Colors.white,

*letterSpacing*: 1.2,

          ),

        ),

*centerTitle*: true,

*backgroundColor*: Colors.blue,

*elevation*: 5,

      ),

*body*: Padding(

*padding*: const EdgeInsets.all(16.0),

*child*: Column(

*crossAxisAlignment*: *CrossAxisAlignment*.stretch,

*children*: [

            const SizedBox(*height*: 40),

            ElevatedButton.icon(

*icon*: Icon(Icons.add, *size*: 50),

*label*: Text(

                'Araç Ekle',

*style*: TextStyle(

*fontSize*: 24,

*fontWeight*: FontWeight.bold,

*color*: Colors.white,

                ),

              ),

*style*: ElevatedButton.styleFrom(

*padding*: EdgeInsets.symmetric(*vertical*: 20),

*backgroundColor*: Colors.blue,

*shape*: RoundedRectangleBorder(

*borderRadius*: BorderRadius.circular(16),

                ),

*elevation*: 5,

              ),

*onPressed*: () {

                Navigator.push(*context*, MaterialPageRoute(*builder*: (*\_*) => AracEkle()));

              },

            ),

            const SizedBox(*height*: 20),

            ElevatedButton.icon(

*icon*: Icon(Icons.list, *size*: 50),

*label*: Text(

                'Araç Listele',

*style*: TextStyle(

*fontSize*: 24,

*fontWeight*: FontWeight.bold,

*color*: Colors.white,

                ),

              ),

*style*: ElevatedButton.styleFrom(

*padding*: EdgeInsets.symmetric(*vertical*: 20),

*backgroundColor*: Colors.green,

*shape*: RoundedRectangleBorder(

*borderRadius*: BorderRadius.circular(16),

                ),

*elevation*: 5,

              ),

*onPressed*: () {

                Navigator.push(*context*, MaterialPageRoute(*builder*: (*\_*) => AracListele()));

              },

            ),

            const SizedBox(*height*: 20),

            ElevatedButton.icon(

*icon*: Icon(Icons.delete, *size*: 50),

*label*: Text(

                'Araç Sil',

*style*: TextStyle(

*fontSize*: 24,

*fontWeight*: FontWeight.bold,

*color*: Colors.white,

                ),

              ),

*style*: ElevatedButton.styleFrom(

*padding*: EdgeInsets.symmetric(*vertical*: 20),

*backgroundColor*: Colors.red,

*shape*: RoundedRectangleBorder(

*borderRadius*: BorderRadius.circular(16),

                ),

*elevation*: 5,

              ),

*onPressed*: () {

                Navigator.push(*context*, MaterialPageRoute(*builder*: (*\_*) => AracSil()));

              },

            ),

            const Spacer(),

            Center(

*child*: Text(

                '© Faruk Tazeoğlu',

*style*: TextStyle(

*fontSize*: 14,

*fontStyle*: *FontStyle*.italic,

*color*: Colors.grey[700],

                ),

              ),

            ),

          ],

        ),

      ),

    );

  }

}

Ardından arac\_ekle.dart a vereceğim kodları yapıştırın.

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

class AracEkle extends StatefulWidget {

*@*override

  \_AracEkleState createState() => \_AracEkleState();

}

class \_AracEkleState extends State<AracEkle> {

  final \_formKey = GlobalKey<FormState>();

  final aracAdiController = TextEditingController();

  final aracMarkaController = TextEditingController();

  final aracModelController = TextEditingController();

  final uretimYiliController = TextEditingController();

  final plakaController = TextEditingController();

  Future<void> \_aracEkle() async {

    final response = await http.post(

      Uri.parse('http://127.0.0.1:5000/add\_car'),

*headers*: {'Content-Type': 'application/json'},

*body*: jsonEncode({

        'AracAdi': aracAdiController.text,

        'AracMarka': aracMarkaController.text,

        'AracModel': aracModelController.text,

        'UretimYili': uretimYiliController.text,

        'Plaka': plakaController.text,

      }),

    );

    if (response.statusCode == 200) {

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(*content*: Text("Araç başarıyla eklendi")));

      aracAdiController.clear();

      aracMarkaController.clear();

      aracModelController.clear();

      uretimYiliController.clear();

      plakaController.clear();

    } else {

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(*content*: Text("Bir hata oluştu: ${*response*.*body*}")));

    }

  }

*@*override

  Widget build(BuildContext *context*) {

    return Scaffold(

*appBar*: AppBar(

*title*: Text("Araç Ekle"),

*centerTitle*: true,

      ),

*body*: Padding(

*padding*: const EdgeInsets.all(16.0),

*child*: Form(

*key*: \_formKey,

*child*: ListView(

*children*: [

              TextFormField(

*controller*: aracAdiController,

*decoration*: InputDecoration(

*labelText*: "Araç Adı",

*border*: OutlineInputBorder(*borderRadius*: BorderRadius.circular(12)),

*filled*: true,

*fillColor*: Colors.blue[50],

                ),

*validator*: (*value*) {

                  if (*value* == null || *value*.isEmpty) {

                    return 'Lütfen araç adını girin';

                  }

                  return null;

                },

              ),

              SizedBox(*height*: 16),

              TextFormField(

*controller*: aracMarkaController,

*decoration*: InputDecoration(

*labelText*: "Araç Markası",

*border*: OutlineInputBorder(*borderRadius*: BorderRadius.circular(12)),

*filled*: true,

*fillColor*: Colors.blue[50],

                ),

*validator*: (*value*) {

                  if (*value* == null || *value*.isEmpty) {

                    return 'Lütfen araç markasını girin';

                  }

                  return null;

                },

              ),

              SizedBox(*height*: 16),

              TextFormField(

*controller*: aracModelController,

*decoration*: InputDecoration(

*labelText*: "Araç Modeli",

*border*: OutlineInputBorder(*borderRadius*: BorderRadius.circular(12)),

*filled*: true,

*fillColor*: Colors.blue[50],

                ),

*validator*: (*value*) {

                  if (*value* == null || *value*.isEmpty) {

                    return 'Lütfen araç modelini girin';

                  }

                  return null;

                },

              ),

              SizedBox(*height*: 16),

              TextFormField(

*controller*: uretimYiliController,

*decoration*: InputDecoration(

*labelText*: "Üretim Yılı (yyyy-mm-dd)",

*border*: OutlineInputBorder(*borderRadius*: BorderRadius.circular(12)),

*filled*: true,

*fillColor*: Colors.blue[50],

                ),

*validator*: (*value*) {

                  if (*value* == null || *value*.isEmpty) {

                    return 'Lütfen üretim yılını girin';

                  }

                  return null;

                },

              ),

              SizedBox(*height*: 16),

              TextFormField(

*controller*: plakaController,

*decoration*: InputDecoration(

*labelText*: "Plaka",

*border*: OutlineInputBorder(*borderRadius*: BorderRadius.circular(12)),

*filled*: true,

*fillColor*: Colors.blue[50],

                ),

*validator*: (*value*) {

                  if (*value* == null || *value*.isEmpty) {

                    return 'Lütfen plakayı girin';

                  }

                  return null;

                },

              ),

              SizedBox(*height*: 20),

              ElevatedButton(

*onPressed*: () {

                  if (\_formKey.currentState!.validate()) {

                    \_aracEkle();

                  }

                },

*style*: ElevatedButton.styleFrom(

*padding*: EdgeInsets.symmetric(*vertical*: 16),

*backgroundColor*: Colors.blue,

*shape*: RoundedRectangleBorder(

*borderRadius*: BorderRadius.circular(12),

                  ),

                ),

*child*: Text("Kaydet", *style*: TextStyle(*fontSize*: 18)),

              ),

            ],

          ),

        ),

      ),

    );

  }

}

Sonra aşşağıda vereceğim kodları arac\_sil.dart a yapıştırın:

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

class AracSil extends StatefulWidget {

*@*override

  \_AracSilState createState() => \_AracSilState();

}

class \_AracSilState extends State<AracSil> {

  List<*dynamic*> araclar = [];

*@*override

  void initState() {

*super*.initState();

    \_getAraclar();

  }

  // Araçları veritabanından çekme fonksiyonu

  Future<void> \_getAraclar() async {

    final response = await http.get(Uri.parse('http://127.0.0.1:5000/list\_cars'));

    if (response.statusCode == 200) {

      setState(() {

        araclar = json.decode(response.body);

      });

    } else {

      throw Exception('Veritabanından araçlar çekilemedi');

    }

  }

  // Araç silme fonksiyonu

  Future<void> \_silArac(int *aracId*) async {

    final response = await http.delete(Uri.parse('http://127.0.0.1:5000/delete\_car/$*aracId*'));

    if (response.statusCode == 200) {

      // Başarılı silme işlemi sonrası veriyi güncelle

      \_getAraclar();

    } else {

      throw Exception('Araç silinemedi');

    }

  }

  // Onay penceresini açan fonksiyon

  void \_showDeleteConfirmationDialog(int *aracId*) {

    showDialog(

*context*: context,

*builder*: (BuildContext *context*) {

        return AlertDialog(

*title*: Text('Aracı Sil'),

*content*: Text('Aracı silmek istediğinizden emin misiniz?'),

*actions*: <Widget>[

            TextButton(

*onPressed*: () {

                \_silArac(*aracId*); // Araç silinsin

                Navigator.of(*context*).pop(); // Dialogu kapat

              },

*child*: Text('EVET'),

            ),

            TextButton(

*onPressed*: () {

                Navigator.of(*context*).pop(); // Dialogu kapat

              },

*child*: Text('HAYIR'),

            ),

          ],

        );

      },

    );

  }

*@*override

  Widget build(BuildContext *context*) {

    return Scaffold(

*appBar*: AppBar(

*title*: Text('Araç Sil'),

*backgroundColor*: Colors.red,

      ),

*body*: ListView.builder(

*itemCount*: araclar.length,

*itemBuilder*: (*context*, *index*) {

          final arac = araclar[*index*];

          return Card(

*margin*: EdgeInsets.all(8),

*elevation*: 5,

*child*: ListTile(

*title*: Text('${*arac*['AracAdi']} ${*arac*['AracMarka']}'),

*subtitle*: Text('${*arac*['AracModel']} - ${*arac*['UretimYili']}'),

*trailing*: Icon(Icons.delete\_forever, *color*: Colors.red),

*onTap*: () {

                // Araç üzerine tıklanınca onay penceresini göster

                \_showDeleteConfirmationDialog(arac['AracID']);

              },

            ),

          );

        },

      ),

    );

  }

}

Ve son olarak aşşağıda vereceğim kodları da arac\_listele.dart a yapıştırın  
import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

class AracListele extends StatefulWidget {

*@*override

  \_AracListeleState createState() => \_AracListeleState();

}

class \_AracListeleState extends State<AracListele> {

  List<*dynamic*> araclar = [];

*@*override

  void initState() {

*super*.initState();

    \_araclariGetir();

  }

  Future<void> \_araclariGetir() async {

    final response = await http.get(Uri.parse('http://127.0.0.1:5000/list\_cars'));

    if (response.statusCode == 200) {

      setState(() {

        araclar = jsonDecode(response.body);

      });

    } else {

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(*content*: Text("Araçlar alınırken hata oluştu")));

    }

  }

*@*override

  Widget build(BuildContext *context*) {

    return Scaffold(

*appBar*: AppBar(

*title*: Text("Araç Listele"),

*centerTitle*: true,

      ),

*body*: araclar.isEmpty

          ? Center(*child*: CircularProgressIndicator())

          : ListView.builder(

*itemCount*: araclar.length,

*itemBuilder*: (*context*, *index*) {

                return Card(

*margin*: EdgeInsets.symmetric(*vertical*: 10, *horizontal*: 16),

*elevation*: 5,

*shape*: RoundedRectangleBorder(

*borderRadius*: BorderRadius.circular(12),

                  ),

*child*: ListTile(

*title*: Text(araclar[*index*]['AracAdi'], *style*: TextStyle(*fontSize*: 20, *fontWeight*: FontWeight.bold)),

*subtitle*: Text('Marka: ${*araclar*[*index*]['AracMarka']} - Model: ${*araclar*[*index*]['AracModel']}'),

*trailing*: Icon(Icons.chevron\_right),

*onTap*: () {

                      // Araç detayına gitme işlevi

                    },

                  ),

                );

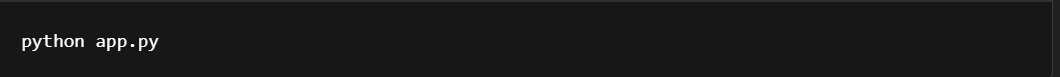
              },

            ),

    );

  }

}

**⚠️Not:** Açıkladığım şeyleri yaptıysanız geriye sadece run etmek kalıyor fakat bi şeye dikkat edin her seferinde soldaki terminalde   


Komutunu yazıp port almayı ihmal etmeyin

Sonra sağ taraftaki terminalde de tatbikî sırasıyla:

Komutunu yazıp çıkan seçeneği (2) seçmemiz gerekiyor

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.🏁Ve karşımızda şu ekran olacak

**Değerli Hocam,**

Bu proje sürecinde, **Flutter ile mobil uygulama geliştirme** konusunda edindiğimiz bilgi ve deneyimleri pekiştirdik. Başlangıçtan itibaren, her aşamayı titizlikle takip ederek, projemizi başarılı bir şekilde tamamladık.

İlk olarak, **Microsoft Access** ile veri tabanı oluşturmayı, ardından bu veri tabanını **Flask backend** ile entegrasyonunu sağladık. **Python**, **Flask** ve **pyodbc** kütüphanelerini kullanarak backend'in işleyişini gerçekleştirdik. Bu backend kısmı, veri tabanına veri ekleme, listeleme ve silme işlemlerini başarılı bir şekilde gerçekleştirebilmemizi sağladı.

Sonrasında, **Flutter** ile kullanıcı arayüzünü oluşturduk. Geliştirdiğimiz **Ana Sayfa** ekranı üzerinden araç ekleme, listeleme ve silme işlemlerini kolayca yönetebiliyoruz. Kod yapısını düzgün bir şekilde organize ederek, projeyi anlaşılır ve sürdürülebilir bir hale getirdik.

Projenin tüm adımlarını başarıyla geçerek uygulamayı çalışır hale getirdik. Geriye sadece, uygulamanın farklı cihazlarda sorunsuz çalışması ve test edilmesi kaldı.

Projeyi bu noktada başarıyla tamamladık. Bu süreç boyunca hem **backend** hem de **frontend** konularında önemli tecrübeler kazandım ve bu projenin yazılım geliştirme alanındaki bilgi birikimimi artırdığına inanıyorum.

Yardımlarınız ve rehberliğiniz için çok teşekkür ederim. Bu projede öğrendiğim bilgiler, kariyerimde bana yol gösterecek çok kıymetli deneyimler sağladı. Projeyi tamamlamış olmanın gururunu yaşıyorum.

Saygılarımla,  
Faruk Tazeoğlu