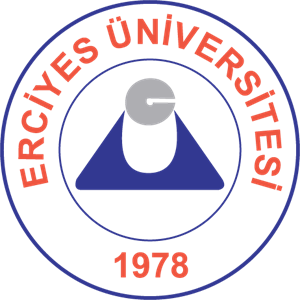
****

**ERCİYES ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

MOBILE APPLICATION DEVELOPMENT DERSİ

DR. ÖĞR. ÜYESİ FEHİM KÖYLÜ

RAPOR

1030510576 ŞEYMA GİZEM SİVRİ

Nisan, 2025

**Assignment for the Mobile Application Development Course**

**RandomUser API’den İsim/Soyisim Çekip Profil Kartı Oluşturma**

**( Dart Asenkron Programlama)**

Bu ödev, Dart diliyle asenkron programlama kullanılarak RandomUser API’den veri çekilmesi ve verilerin bir profil kartı üzerinde gösterilmesi amacıyla hazırlanmıştır. Çalışma, Erciyes Üniversitesi Bilgisayar Mühendisliği Bölümü öğrencisi 1030510576 numaralı Şeyma Gizem Sivri tarafından, Mobile Application Development dersi kapsamında, Öğr. Gör. Dr. Fehim Köylü danışmanlığında gerçekleştirilmiştir. (<https://github.com/seymagizem/Dart_Asenkron>)

Bu proje, Dart programlama dili ve Flutter framework'ü kullanılarak geliştirilmiştir. Projenin temel amacı, internet üzerinden**RandomUser API** aracılığıyla rastgele bir kullanıcı bilgisi çekmek ve bu bilgileri şık bir profil kartı üzerinde görsel olarak sunmaktır. Uygulama, **asenkron programlama, HTTP istekleri, durum yönetimi**ve**responsive kullanıcı arayüzü tasarımı**gibi mobil uygulama geliştirme açısından önemli konuları kapsamaktadır.

**Kullanılan Kütüphaneler**

Bu hedefleri gerçekleştirmek için aşağıdaki kütüphaneler kullanılmıştır:

* **Flutter Material**: Flutter'ın temel tasarım bileşenlerini içerir. material.dart paketi, uygulamanın tüm UI öğeleri için kullanılmıştır.
* **http**: REST API üzerinden veri çekmek için kullanılır. http kütüphanesi ile GET isteği yapılmıştır.
* **dart:convert**: API'den gelen JSON verilerini Dart nesnelerine çevirmek için kullanılır.

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

Bu kod bloğunda Flutter uygulamasının temel yapı taşı olan MaterialApp widget'ı oluşturulmuştur. title parametresi ile uygulama başlığı belirlenmiş, ThemeData kullanılarak genel arka plan rengi açık mor (Scaffold için) ve AppBar arka planı beyaz olarak ayarlanmıştır; ayrıca modern ve akıcı tasarım sunan Material 3 standartları etkinleştirilmiştir. home parametresi ile uygulamanın başlangıç ekranı olarak RandomUserPage atanmış, debugShowCheckedModeBanner: false diyerek geliştirici modunda çıkan DEBUG etiketi gizlenmiştir. Bu yapı sayesinde uygulama genelinde tutarlı bir tema ve düzenli bir görünüm sağlanmıştır. Bu temel iskelet kurulduktan sonra, kullanıcı verilerinin yönetilmesi aşamasına geçilmiştir:

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

\_RandomUserPageState sınıfı Flutter uygulamasında kullanıcı arayüzünün durumunu (state) yönetir ve iki temel değişken içerir:

* userData çekilen kullanıcı verilerini saklamak için tanımlanmış bir Map türündedir,
* isLoading ise veri çekme sürecinde yüklenme animasyonu gösterilmesini sağlar.

initState() fonksiyonu widget ekrana ilk kez yüklendiğinde çalışarak otomatik şekilde fetchRandomUser() fonksiyonunu çağırır ve uygulama açılır açılmaz bir kullanıcı verisinin getirilmesini sağlar.  fetchRandomUser() fonksiyonu asenkron çalışır; http.get() metoduyla https://randomuser.me/api/ adresinden veri çeker, gelen yanıtı JSON formatında ayrıştırır ve başarı durumunda userData değişkenine atayıp arayüzü setState() fonksiyonu ile günceller; hata oluşursa bir Exception fırlatır. Bu yapı, modern Flutter uygulamalarında gerçek zamanlı veri çekimi ve kullanıcı arayüzü güncellemesi için kullanılan standart bir yöntemdir.

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

Bu kod parçasında ise Flutter uygulamasında internet üzerinden veri çekme işlemi gerçekleştirilmiştir. http.get fonksiyonu kullanılarak https://randomuser.me/api/ adresine bir GET isteği gönderilmiş ve bu istek asenkron (await) şekilde beklenmiştir; yani uygulama, veri gelene kadar bu satırda durur ama arka planda çalışmaya devam eder.Gelen response nesnesinin statusCode değeri 200 ise, bu sunucudan başarılı bir yanıt alındığını gösterir; bu durumda response.bodyiçeriği json.decode ile ayrıştırılarak bir Dart nesnesine çevrilir. Ardından setState() fonksiyonu çağrılarak kullanıcı verileri (userData) güncellenir ve isLoading değişkeni false yapılarak yükleme animasyonunun durması sağlanır. Eğer statusCode 200 değilse, yani bir hata oluşmuşsa Exception fırlatılarak kullanıcı verisinin alınamadığı bildirilir. Bu yapı Flutter'da API'den veri çekimi sırasında doğru veri işleme ve hatalı durumları yönetme adına oldukça standart ve güvenli bir yaklaşımdır. Veri çekim işleminin ardından bu verilerin nasıl ekrana yansıtıldığına geçelim:

build metodu kullanıcı arayüzünü tanımlar ve her setState çağrıldığında çalışarak ekranı günceller. Bu metotta bir Scaffold widgetı kullanılarak sayfa iskeleti oluşturulmuş, üst kısmına özel tasarımlı bir AppBar eklenmiştir. Sayfanın gövdesi (body) bir SingleChildScrollView içinde organize edilmiş ve içerik dikey bir Column kullanılarak sıralanmıştır. Eğer APIden çekilen userData mevcutsa, kullanıcı bilgilerini gösteren modern tasarımlı bir Card yapısı görüntülenir; profil fotoğrafı (CircleAvatar), isim, ülke, yaş ve e-posta bilgileri kartın içinde düzenli bir şekilde sunulur. Kartın altında, "Yeni Kullanıcı Getir" yazılı bir buton (ElevatedButton.icon) bulunur; bu butona basıldığında API'den yeni bir kullanıcı verisi çekilir. Eğer veri çekme işlemi devam ediyorsa (isLoading true ise) kullanıcıya CircularProgressIndicator animasyonu ve "Yükleniyor, lütfen bekleyiniz..." metni ile yükleme durumu gösterilir. Bu yapı, hem veri çekimine bağlı dinamik bir içerik sunar hem de kullanıcı etkileşimini sürekli güncel tutar.

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

Bu kod bloğunda da kullanıcı arayüzüne bir buton (ElevatedButton.icon) eklenmiş ve bu buton sayesinde kullanıcı, ekrandaki verileri güncelleyebilir hale getirilmiştir. SizedBox widget'ları, butonun üstüne ve altına 20 birimlik boşluk bırakılarak düzenli bir görünüm sağlamak için kullanılmıştır. ElevatedButton.icon hem bir ikon (Icons.refresh) hem de bir yazı ("Yeni Kullanıcı Getir") içerir; butonun arka plan rengi cardColor (beyaz), yazı ve ikon rengi ise siyah olarak ayarlanmıştır. Kullanıcı bu butona bastığında onPressed olayında fetchRandomUser()fonksiyonu tetiklenir ve API'den yeni bir kullanıcı verisi çekilir. Ayrıca, isLoading değişkeni true olduğunda (yani veri çekme işlemi devam ederken) ekranda bir CircularProgressIndicator (yüklenme çarkı) ve altında "Yükleniyor, lütfen bekleyiniz..." yazısı gösterilerek kullanıcıya veri işleminin sürdüğü bilgisi verilir. Böylece kullanıcı etkileşimleri daha akıcı, bilgilendirici ve sezgisel bir hale getirilmiştir.

Bu proje ile Dart ve Flutter kullanılarak API'den veri çekme ve dinamik kullanıcı arayüzü oluşturma becerileri geliştirilmiştir. Yapılan çalışma, mobil uygulama geliştirme alanındaki teknik bilgilerimi pekiştirmiş ve ileriye dönük projeler için sağlam bir temel oluşturmuştur.

Son olarak uygulamadan ekran görüntüleri aşağıda sunulmuştur:

**metin, mobil telefon, mobil cihaz, İletişim Cihazı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. metin, mobil telefon, mobil cihaz, insan yüzü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. insan yüzü, metin, mobil telefon, mobil cihaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Şekil 1. Örnek kişi kartı yenileme ekranı**