

ALLFACE ETKİNLİK ÖNERİ UYGULAMASI

HÜSEYİN ÇAĞLAR

ÖZET

ALLFACE, duygu durumu analizi ve kişiselleştirilmiş etkinlik öneri sistemi üzerine geliştirilmiş kapsamlı bir mobil uygulamadır. Bu yenilikçi platform, kullanıcıların anlık duygu durumlarını analiz eden bir model kullanarak, yaş, cinsiyet ve meslek gibi faktörlere dayalı olarak etkinlik önerileri sunar. Kullanıcılar, ana sayfada özgün gönderiler oluşturup paylaşım yapabilir ve önerilen etkinlikleri onayladıklarında bu etkinlikleri de toplulukla paylaşabilirler. Ayrıca, kullanıcı deneyimini sürekli iyileştirmek amacıyla, detaylı geri bildirimlerin toplanabildiği özel bir geri bildirim sayfası da uygulamaya entegre edilmiştir. ALLFACE, sadece bireylerin sosyal etkileşimlerini desteklemekle kalmaz, aynı zamanda duygusal farkındalıklarını artırarak daha bilinçli bir sosyal katılımı teşvik eder.

İÇİNDEKİLER

ÖZET	i
İÇİNDEKİLER	ii
ŞEKİLLER.....	iv
GİRİŞ.....	1
BİRİNCİ BÖLÜM.....	2
KULLANICI GİRİŞİ VE KAYDI.....	2
GİRİŞ SAYFASI.....	2
KAYIT OL SAYFASI.....	6
İKİNCİ BÖLÜM.....	11
ANA SAYFA	11
Sayfa Özellikleri.....	12
Kullanıcı Deneyimi	15
KAMERA SAYFASI.....	16
Sayfa Özellikleri.....	17
Kullanıcı Deneyimi	20
DOĞRULAMA SAYFASI	21
Sayfa Özellikleri.....	22
Kullanıcı Deneyimi	24
ÖNERİ SAYFASI.....	25
Sayfa Özellikleri.....	26
Kullanıcı Deneyimi	28
AKTİVİTE SAYFASI	29
Sayfa Özellikleri.....	30
Kullanıcı Deneyimi	31
ONAYLANAN AKTİVİTELER SAYFASI.....	32
Sayfa Özellikleri.....	33
Kullanıcı Deneyimi	36
PROFİL SAYFASI	37
Sayfa Özellikleri.....	39
Kullanıcı Deneyimi	41
AYARLAR SAYFASI.....	42
GERİ BİLDİRİM SAYFASI.....	44
Sayfa Özellikleri.....	45

Kullanıcı Deneyimi	48
ÜÇÜNCÜ BÖLÜM.....	49
ETKİNLİK ÖNERİ MODELİ.....	49
Kullanılan Teknolojiler.....	49
Özellikler	50
Örnek Kullanım.....	54
DUYGU ANALİZİ MODELİ	56
Kullanılan Teknolojiler.....	56
DÖRDÜNCÜ BÖLÜM	65
FIREBASE	65
KAYNAKÇA.....	67

ŞEKİLLER

Şekil 1 Giriş Sayfası	2
Şekil 2 Giriş İşlemi	3
Şekil 3 Tema Değişikliği	4
Şekil 4 Kayıt Ol Sayfası	6
Şekil 5 Şifre Eşleşme Kontrolü ve Kullanıcı Kaydı	8
Şekil 6 Kullanıcı Bilgilerini Firestore'a Kaydetme	9
Şekil 7 Ana Sayfa	11
Şekil 8 Mesaj Paylaşma ve Firebase Kaydı	13
Şekil 9 Gönderi Akışı ve Verilerin Gösterilmesi	14
Şekil 10 Kamera Sayfası	16
Şekil 11 Kamera Başlatma	18
Şekil 12 TensorFlow Lite Model Yükleme	18
Şekil 13 Modeli Çalıştırma ve Duygu Analizi	19
Şekil 14 Fotoğraf Çekme ve Kaydetme	20
Şekil 15 Doğrulama Sayfası	21
Şekil 16 Firebase Storage'a Fotoğraf Yükleme Fonksiyonu	23
Şekil 17 Duygu Klasörü Belirleme	23
Şekil 18 Öneri Sayfası	25
Şekil 19 Firebase Kullanıcı Bilgilerini Çekme	26
Şekil 20 HTTP POST İsteği ile Öneri Alma	27
Şekil 21 Duygu Durumuna Göre Öneri Getirme	27
Şekil 22 Aktivite Sayfası	29
Şekil 23 Etkinlik Kaydetme veya Güncelleme	30
Şekil 24 Etkinlik Açıklamalarını Sağlayan Fonksiyon	31
Şekil 25 Buton ile Onaylama ve Yönlendirme	31
Şekil 26 Onaylanan Aktiviteler Sayfası	32
Şekil 27 Firestore Akışı	33
Şekil 28 Veri Listeleme	34
Şekil 29 Gönderi Paylaşma Diyaloğu	35
Şekil 30 Firestore'a Veri Kaydı	36
Şekil 31 Profil Sayfası	37
Şekil 32 Profil Düzenleme	38
Şekil 33 Meslek Seçimi (SearchableDropdown)	39
Şekil 34 Gönderi Düzenleme ve Silme	40
Şekil 35 Ayarlar Sayfası	42
Şekil 36 Tema Değişimi	43
Şekil 37 Arka Plan	43
Şekil 38 Geri Bildirim Sayfası	44
Şekil 39 Geri Bildirim Gönderme	46
Şekil 40 Özel Snackbar Mesajı	47
Şekil 41 Model Eğitim Fonksiyonu	51
Şekil 42 Tahmin API'si	52
Şekil 43 Tüm Modelleri Eğitim	53
Şekil 44 En Çok Tercih Edilen Etkinlikler	53

Şekil 45 Model Kullanımı 1	54
Şekil 46 Model Kullanımı 2	55
Şekil 47 MobilNetV2	56
Şekil 48 Convolutional Neural Networks (CNN)	58
Şekil 49 ImageDataGenerator (Veri Artırma)	59
Şekil 50 Learning Rate Scheduler	60
Şekil 51 Callbacks (EarlyStopping ve ReduceLROnPlateau)	61
Şekil 52 Model Kaydetme ve Dönüştürme	62
Şekil 53 Sonuçların Görselleştirilmesi	63
Şekil 54 Model Sonuçları	63
Şekil 55 Model Değerlendirme	64
Şekil 56 Aktiviteler	65
Şekil 57 Geri Bildirim	65
Şekil 58 Paylaşılan Gönderiler	66
Şekil 59 Kullanıcılar	66

GİRİŞ

Dijital çağda bireylerin ruh hali ve sosyal etkileşimlerinin izlenmesi, kişiselleştirilmiş öneri sistemleri aracılığıyla daha anlamlı ve etkili bir hale gelmiştir. Bu bağlamda geliştirilen ALLFACE mobil uygulaması, kullanıcıların duygu durumlarını analiz ederek hem bireysel farkındalığı artırmakta hem de önerilen etkinliklerle kullanıcıları sosyal olarak aktif kılmaktadır. Gelişmiş duygu analizi modeli sayesinde, uygulama anlık duygusal verileri işleyerek etkin öneri mekanizmaları sunar. Bunun yanı sıra, kullanıcıların sosyal etkileşimlerini desteklemek için gönderi paylaşımı ve etkinlik onayı gibi özellikler geliştirilmiştir. Kullanıcı geri bildirimleri, platformun sürekli iyileştirilmesi için değerli veriler sağlamak amacıyla özel bir geri bildirim sayfasında toplanmaktadır.

ALLFACE projesi, uygulama geliştirme kısmında Flutter, duygu analizi ve öneri modellerinin geliştirilmesi için ise Python programlama dillerini kullanmaktadır. Flutter, modern ve kullanıcı dostu bir mobil arayüz sağlarken, Python'un veri analizi ve makine öğrenimi yetenekleri duygu analizi modelinin temelini oluşturur. Bu teknik yaklaşım, hem performans hem de doğruluk açısından güçlü bir uygulama deneyimi sunmaktadır. Bu rapor, ALLFACE platformunun teknik yapısını, işlevsel özelliklerini ve kullanıcı deneyimini detaylı bir şekilde açıklamayı amaçlamaktadır.

BİRİNCİ BÖLÜM

KULLANICI GİRİŞİ VE KAYDI

GİRİŞ SAYFASI

Giriş sayfası, kullanıcının giriş yapmasını sağlamak amacıyla bir login formu içerir. Kullanıcı email ve şifre bilgilerini girdikten sonra Firebase Authentication kullanılarak giriş işlemi gerçekleştirilir. Giriş işlemi sırasında, kullanıcıya işlem süresince bir yükleme göstergesi gösterilir. Sayfa teması karanlık ve aydınlık mod arasında geçiş yapabilecek şekilde tasarlanmıştır. Ayrıca, yeni kullanıcıların kayıt olmalarını sağlayan bir bağlantı da içerir.[4]



Şekil 1 Giriş Sayfası

Sayfa Özellikleri

Giriş Formu:

Kullanıcıdan email ve şifre bilgileri alınır. Bu bilgiler Firebase Authentication ile doğrulanır.

Firebase Authentication ile Giriş:

Kullanıcı email ve şifre bilgileriyle giriş yapar. Eğer giriş başarılı olursa, kullanıcı ana sayfaya yönlendirilir.

Tema Değişikliği:

Sağ üst köşedeki ikon butonu, kullanıcıya karanlık ve aydınlık mod arasında geçiş yapma imkanı verir.

Kaydolma Yönlendirmesi:

Kullanıcı, kaydolmak için başka bir sayfaya yönlendirilir.

```
void login() async {  
  showDialog(  
    context: context,  
    builder: (context) => const Center(  
      child: CircularProgressIndicator(),  
    )); // Center  
  
  try {  
    await FirebaseAuth.instance.signInWithEmailAndPassword(  
      email: emailController.text, password: passwordController.text);  
    if (context.mounted) Navigator.pop(context);  
  } on FirebaseAuthException catch (e) {  
    Navigator.pop(context);  
    displayMessageToUser(e.code, context);  
  }  
}
```

Şekil 2 Giriş İşlemi

Yükleme Göstergesi Gösterme:

showDialog ile bir yükleme göstergesi (CircularProgressIndicator) açılır. Bu, kullanıcı giriş işlemi yapılırken ekranda gösterilen bir yuvarlak dönen simgedir.

Firebase ile Giriş Yapma:

FirebaseAuth.instance.signInWithEmailAndPassword metodu, kullanıcının girdiği email ve şifre bilgilerini kullanarak Firebase Authentication ile giriş yapmaya çalışır.

Giriş Başarılıysa:

Eğer giriş başarılı olursa, Navigator.pop(context) ile yükleme göstergesi kapanır ve kullanıcıya başka bir işlem yapılmaz.

Hata Durumu:

Eğer giriş sırasında bir hata oluşursa (örneğin, yanlış email ya da şifre girilmişse), FirebaseAuthException yakalanır. Bu durumda yükleme göstergesi kapanır ve hata mesajı kullanıcıya gösterilir (displayMessageToUser(e.code, context)).

```
Positioned(  
  top: 30,  
  right: 10,  
  child: IconButton(  
    icon: Icon(  
      isDarkMode ? Icons.dark_mode : Icons.light_mode,  
      size: 24,  
    ), // Icon  
    onPressed: () {  
      Provider.of<ThemeProvider>(context, listen: false)  
        .toggleTheme();  
    },  
  ), // IconButton  
), // Positioned
```

Şekil 3 Tema Değişikliği

Positioned Widget:

Positioned widget'ı, butonu ekranın belirli bir yerine yerleştirir. Bu örnekte, buton 30 piksel yukarıda (top: 30) ve 10 piksel sağda (right: 10) konumlanır.[2]

IconButton:

IconButton, bir ikonun tıklanabilir hale gelmesini sağlar. Burada, butonun ikonu karanlık modda (Icons.dark_mode) veya aydınlık modda (Icons.light_mode) olacak şekilde ayarlanır. Hangi modda olduğuna göre ikon değişir (isDarkMode ? Icons.dark_mode : Icons.light_mode).[8]

Tema Değiştirme:

onPressed fonksiyonu, butona tıklanıldığında çağrılır. Bu fonksiyon, Provider.of<ThemeProvider>(context, listen: false).toggleTheme() ile tema değişimini tetikler. toggleTheme() fonksiyonu, uygulamanın tema modunu (karanlık veya aydınlık) değiştirir.[5]

Kullanıcı Deneyimi

Uygulama, kullanıcılara modern ve kullanıcı dostu bir giriş ekranı sunar. Giriş sayfası, kullanıcıların hesaplarına kolayca giriş yapabilmesi için tasarlanmıştır. Kullanıcı deneyimini daha da iyileştirmek için yapılan bazı önemli tasarım unsurları şunlardır:

Giriş Formu ve Kullanıcı Dostu Tasarım:

Giriş ekranı, kullanıcıların mail adreslerini ve şifrelerini girmeleri için iki farklı metin kutusu (email ve password) içerir. Bu kutular, kullanıcıların doğru veriyi girmelerini sağlamak amacıyla açıkça etiketlenmiştir.

Şifre alanı, şifreyi gizlemek için bir obscureText özelliğiyle korunur. Bu, güvenlik açısından önemli bir özelliktir.

Hızlı ve Kolay Giriş İmkânı:

Kullanıcılar, giriş yaptıktan sonra işlemin ne kadar sürdüğünü görebilmek için bir yükleme göstergesi (CircularProgressIndicator) ile bilgilendirilir. Bu, kullanıcıların işlem sırasında beklediklerinde daha iyi bir deneyim sağlar.

Eğer giriş işlemi başarılı olursa, kullanıcı doğrudan ana sayfaya yönlendirilir. Ancak bir hata durumu meydana gelirse (örneğin, yanlış şifre ya da email), hata mesajı kullanıcıya bildirilir ve kullanıcıyı yönlendiren uyarılar gösterilir.

Karanlık Mod ve Işık Modu Desteği:

Kullanıcıların tercihlerine göre uygulamanın tema rengini değiştirmelerini sağlayan bir özellik de eklenmiştir. Ekranın üst kısmında bulunan ikon, kullanıcıların karanlık mod ve ışık mod arasında geçiş yapabilmesini sağlar. Kullanıcı, modları değiştirdiğinde ekranın görünümü hemen değişir, böylece hem gece hem gündüz kullanımına uygun bir deneyim sunulur.

KAYIT OL SAYFASI

Kayıt ol sayfası, yeni kullanıcıların sisteme üye olabilmesi için gerekli bilgileri girmelerini sağlayan bir formdan oluşur. Kullanıcı adı, soyadı, e-posta adresi, şifre, şifre doğrulama, yaş, cinsiyet, meslek ve konum gibi bilgilerin alındığı alanlar vardır. Bu sayfa, kullanıcı dostu bir arayüzle kullanıcılara kolayca kayıt olma imkanı sunar. Kullanıcı, bilgilerini girdikten sonra "Kaydol" butonuna tıklayarak hesabını oluşturabilir. Eğer şifreler uyuşmazsa, bir hata mesajı gösterilir. Kayıt işlemi başarılı olduğunda kullanıcı Firestore'a kaydedilir. Ayrıca, varsa kullanıcıyı giriş yapma sayfasına yönlendiren bir seçenek de bulunur.

Şekil 4 Kayıt Ol Sayfası

Sayfa Özellikleri

Form Alanları:

Kullanıcı adı, soyadı, e-posta, şifre, şifre doğrulama, yaş ve konum gibi temel bilgilerin girilmesi için gerekli alanlar yer alır.

Meslek Seçimi:

Kullanıcılar, geniş bir meslek listesi arasından seçimini yapabilirler. Bu liste, kullanıcıların meslek bilgilerini seçmelerine olanak tanır.

Cinsiyet Seçimi:

Kullanıcılar, radyo butonları aracılığıyla cinsiyetlerini seçebilirler.

Form Doğrulama:

Şifre doğrulama işlemi yapılır, şifreler eşleşmezse kullanıcıya hata mesajı gösterilir.

Firebase Entegrasyonu:

Kullanıcı, bilgilerini girdikten sonra Firebase Authentication kullanılarak hesap oluşturulur. Bu işlem başarılı olursa, kullanıcı bilgileri Firestore veritabanına kaydedilir.[6]

Hata Mesajları:

Formda girilen bilgilerde bir hata varsa (örneğin şifre eşleşmiyorsa), kullanıcıya uygun bir hata mesajı gösterilir.

```

void registerUser() async {
  // Yükleniyor dairesi göster
  showDialog(
    context: context,
    builder: (context) => const Center(
      child: CircularProgressIndicator(),
    ), // Center
  );

  // Şifrelerin eşleştiğini kontrol et
  if (passwordConfirmController.text != passwordController.text) {
    Navigator.pop(context);
    displayMessageToUser("Şifre eşleşmedi!", context);
  } else {
    // Kullanıcı oluştur
    try {
      UserCredential? userCredential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
          email: emailController.text,
          password: passwordController.text,
        );

      // Kullanıcı dosyasını oluşturup Firestore'a ekle
      createUserDocument(userCredential);

      if (context.mounted) Navigator.pop(context);
    } on FirebaseAuthException catch (e) {

      Navigator.pop(context);
      displayMessageToUser(e.code, context);
    }
  }
}

```

Şekil 5 Şifre Eşleşme Kontrolü ve Kullanıcı Kaydı

Bu fonksiyon, kullanıcı şifrelerini kontrol eder, eşleşmiyorsa hata mesajı gösterir ve eşleşiyorsa kullanıcıyı Firebase'e kaydeder.

```
Future<void> createUserDocument(UserCredential? userCredential) async {  
  if (userCredential != null && userCredential.user != null) {  
    await FirebaseFirestore.instance  
      .collection("Users")  
      .doc(userCredential.user!.email)  
      .set({  
        'Email': userCredential.user!.email,  
        'Name': nameController.text,  
        'Surname': surnameController.text,  
        'Gender': genderController.text,  
        'Age': ageController.text,  
        'Occupation': selectedOccupation,  
        'Location': locationController.text,  
      });  
  }  
}
```

Şekil 6 Kullanıcı Bilgilerini Firestore'a Kaydetme

Kayıt işlemi başarılı olduktan sonra, kullanıcının bilgilerini Firestore'a kaydederiz. Bu işlemde kullanıcının adı, soyadı, cinsiyeti gibi bilgiler Firestore koleksiyonuna kaydedilir.[3]

Kullanıcı Deneyimi

Kayıt Formu:

Kullanıcı adı, soyadı, e-posta, şifre ve diğer kişisel bilgileri girerken her alanın açıklamaları ve uygun input türleri vardır (örneğin, yaş sadece sayılarla girilebilir).

Form, görsel olarak düzenli ve kullanıcı dostudur. Kullanıcıdan fazla bilgi almak için her alan uygun şekilde sıralanmıştır.

Şifre Kontrolü:

Kullanıcı şifreyi iki kez girmelidir. Eğer şifreler eşleşmezse, hata mesajı hemen gösterilir. Bu, kullanıcıyı bilgilendirir ve hataların erken fark edilmesini sağlar.

Dinamik Cinsiyet ve Meslek Seçimi:

Cinsiyet, LabeledRadio butonları ile basit bir şekilde seçilir. Meslek seçimi için SearchableDropdown bileşeni kullanılır, böylece kullanıcı listede kaybolmadan arama yaparak hızlıca bir meslek seçebilir.

Konum Seçimi:

Kullanıcı, konum bilgisini manuel olarak yazabilir ya da harita üzerinden seçebilir. Bu iki seçenek kullanıcılara esneklik sağlar.

Kaydol Butonu:

Kullanıcı "Kaydol" butonuna tıkladığında, kayıt işlemi başlar ve Firebase ile iletişim kurulur. Eğer şifre eşleşmesi sağlanmazsa kullanıcı bilgilendirilir. Firebase'e başarılı bir şekilde kayıt yapıldığında, kullanıcıya işlem başarıyla tamamlandığı bildirilir.[13]

Yükleniyor Göstergesi:

Kayıt işlemi sırasında bir CircularProgressIndicator kullanılarak işlem devam ettiği gösterilir. Bu, kullanıcıyı bilgilendirir ve uygulamanın işlem yaptığı hissini verir.

İKİNCİ BÖLÜM

ANA SAYFA

Bu sayfa, kullanıcı aktiviteleri paylaşımı yapmayı amaçlayan bir uygulama sayfasıdır. Kullanıcılar, ruh hallerini seçip, kişisel etkinliklerini ve mesajlarını paylaşabilirler. Ayrıca, bir gönderi akışını izleyebilirler. Sayfa, Firebase veritabanını kullanarak verileri depolar ve dinamik olarak kullanıcıların paylaştığı gönderileri gösterir.

Kullanıcı alttaki simgeye tıklayarak kamera sayfasına yönlendirilir.



Şekil 7 Ana Sayfa

Sayfa Özellikleri

Ruh Hali Seçimi:

Kullanıcılar, "Öfkeli", "Mutlu" veya "Üzgün" gibi ruh halleri arasından bir seçenek seçebilirler. Bu seçim, gönderiyi paylaşılan mesaja dahil edilir.

Kişisel Etkinlik Girişi:

Kullanıcılar, kişisel aktivitelerini paylaşmak için bir metin alanı kullanarak etkinliklerini girerler.

Mesaj Girişi:

Kullanıcılar, metin alanına bir mesaj girerler. Bu mesaj, paylaşılabilecek gönderiye dahil edilir.

Gönderi Butonu:

"Gönder" butonuna tıklayarak, kullanıcılar girilen mesajları, ruh halini ve kişisel etkinlik bilgilerini paylaşabilirler.

Firestore Entegrasyonu:

Kullanıcı gönderisi Firestore veritabanına kaydedilir. Firestore Authentication ile kullanıcı doğrulaması yapılır ve Firestore ile veriler saklanır.

Gönderi Akışı:

StreamBuilder kullanılarak, Firestore'dan gelen gönderiler gerçek zamanlı olarak gösterilir. Her gönderi, kullanıcı e-postası, mesaj, ruh hali ve etkinlik adı gibi bilgileri içerir.[7]

Kamera Sayfası:

Sayfanın alt kısmında, bir fotoğraf çekmeye yönlendiren bir kamera butonu bulunur.

```

void postMessage() async {
    String postMessage = newPostController.text.trim();
    String personalActivity = personalActivityController.text.trim();

    if (postMessage.isNotEmpty && selectedMood != null && user != null) {
        String? userEmail = user?.email;
        await database.addPost(postMessage, userEmail, selectedMood, personalActivity);

        // Alanları temizle
        newPostController.clear();
        personalActivityController.clear();
        setState(() {
            selectedMood = null;
        });
    }
}

```

Şekil 8 Mesaj Paylaşma ve Firebase Kaydı

Bu fonksiyon, kullanıcı tarafından girilen bilgileri kontrol eder ve gönderiyi Firebase veritabanına kaydeder. Ayrıca, metin alanlarını temizler ve ruh hali seçimini sıfırlar.

```

StreamBuilder<List<DocumentSnapshot>>(
  stream: database.getPostsStream(),
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(child: CircularProgressIndicator());
    }
    if (snapshot.hasData) {
      final combinedPosts = snapshot.data!;

      if (combinedPosts.isEmpty) {
        return Center(
          child: Text(
            AppLocalizations.of(context)!.noPostYet)); // Text // Center
      }
      return Expanded(
        child: ListView.builder(
          itemCount: combinedPosts.length,
          itemBuilder: (context, index) {
            final post = combinedPosts[index];
            String message = post['PostMessage'];
            String userEmail = post['UserEmail'];
            Timestamp timestamp = post['TimeStamp'];
            String actName = (post.data() as Map<String,
              dynamic>)['activityName'] ??
              '';
            String mood = (post.data()
              as Map<String, dynamic>)['mood'] ??
              '';

            return MyListTile(
              title: [
                "Aktivite: $actName",
                "Ruh Hali: $mood",
                message,
              ].join('\n'),
              subTitle: userEmail,
              time: timestamp.toDate(),
              actionType: 'publish',
            ); // MyListTile
          },
        ), // ListView.builder
      ); // Expanded
    }
  }
);

```

Şekil 9 Gönderi Akışı ve Verilerin Gösterilmesi

Bu kod, Firestore'dan gelen verileri StreamBuilder ile dinler ve her bir gönderiyi MyListTile widget'ı ile ekranda gösterir.

Kullanıcı Deneyimi

Kullanıcılar için kolay erişim:

Sayfa, kullanıcıların mesaj ve etkinlik bilgilerini kolayca girmelerini sağlar. Kullanıcılar ruh hallerini seçip, etkinliklerini ve mesajlarını yazarak gönderilerini paylaşabilirler.

Canlı Gönderi Akışı:

Sayfa, yeni gönderilerin hemen görünmesini sağlar. Firestore'dan gelen veriler anlık olarak görüntülenir, bu da kullanıcıların diğer kullanıcıların paylaşımlarını hemen görmelerini sağlar.

Dinamik Arayüz:

Kullanıcılar, form alanlarını doldurduktan sonra "Gönder" butonuna tıklayarak kolayca paylaşım yapabilirler. Eğer alanlar eksikse, kullanıcıya herhangi bir hata mesajı gösterilmez, ancak gönderi yapılmaz.

Etkileşimli Tasarım:

Sayfa tasarımı, kullanıcıların ruh halleri, etkinlikler ve mesajları kolayca girmelerine olanak tanır. Ayrıca, görsel tasarımın da kullanıcı deneyimini iyileştiren özellikler taşıdığı söylenebilir.

Kamera Özelliği:

Sayfa, kullanıcıları fotoğraf çekmeye yönlendiren bir kamera butonu içeriyor.

KAMERA SAYFASI

CameraPage, cihazın kamerasını kullanarak gerçek zamanlı duygu analizi yapar. Kullanıcının yüzünü analiz eder ve TensorFlow Lite makine öğrenmesi modeliyle "Mutlu", "Üzgün", "Öfkeli" duygularını tespit eder. Belirli bir duyguya 50 kez ulaşıldığında, otomatik olarak bir fotoğraf çeker ve kullanıcının doğrulama sayfasına yönlendirilmesini sağlar.



Şekil 10 Kamera Sayfası

Sayfa Özellikleri

Canlı Kamera Akışı:

Cihazın kamerasını kullanarak anlık görüntü akışı sağlar.

Duygu Analizi:

TensorFlow Lite modeli ile duyguları tespit eder.[14]

Progres Çemberi:

Kullanıcının belirli bir duyguya ulaşip ulaşmadığını gösterir.

Fotoğraf Çekme:

Belirli bir duyguya ulaşıldığında otomatik fotoğraf çekimi.

Kamera Değiştirme:

Ön ve arka kamera arasında geçiş yapma.

Navigasyon:

Fotoğraf çekildikten sonra kullanıcıyı doğrulama sayfasına (/verificationpage) yönlendirir.

State Management:

Duygu sayıları, kamera durumu ve model durumu gibi veriler state ile yönetilir.[1]

```

Future<CameraController?> loadCamera() async {
  cameraController = CameraController(
    cameras![selectedCamIdx],
    ResolutionPreset.max,
    enableAudio: false,
  );

  // Kamera kontrolörü başlatılıyor
  await cameraController!.initialize();
  isCameraInitialized = true;

  if (!isModelBusy) {
    cameraController!.startImageStream((imageStream) async {
      cameraImage = imageStream;
      runModel(cameraImage);
    });
  }
  return cameraController;
}

```

Şekil 11 Kamera Başlatma

Kamera başlatılır.

Canlı görüntü akışı açılır ve runModel() fonksiyonuna görüntüler gönderilir.

```

Future<void> loadModel() async {
  await Tflite.loadModel(
    model: "assets/model.tflite",
    labels: "assets/labels.txt",
  );
}

```

Şekil 12 TensorFlow Lite Model Yükleme

TensorFlow Lite modeli ve etiketler yüklenir.[15]


```

Future<void> runModel(input) async {
  if (cameraImage != null && cameraImage!.planes.isNotEmpty && !isModelBusy) {
    isModelBusy = true;
    try {
      // Dönüşümü düz pozisyonda ayarlıyoruz
      var predictions = await Tflite.runModelOnFrame(
        bytesList: cameraImage!.planes
          .map<Uint8List>((Plane plane) => convertPlaneToBytes(plane))
          .toList(),
        imageHeight: cameraImage!.height,
        imageWidth: cameraImage!.width,
        imageMean: 0,
        imageStd: 255,
        rotation: 0,
        numResults: 3,
        threshold: 0.1,
        asynch: true,
      );

      if (predictions != null && predictions.isNotEmpty) {
        for (var element in predictions) {
          setState(() {
            emotion = element['label'];
            emotionCounts[emotion!] = (emotionCounts[emotion] ?? 0) + 1;
            progress = (emotionCounts[emotion]! / 50.0)
              .clamp(0.0, 1.0);
          });
        }
        if (emotionCounts[emotion] != null && emotionCounts[emotion]! == 50) {
          await captureImage(); // 50. resim çekiliyor
          cameraController!.stopImageStream();
          Navigator.pushReplacementNamed(context, '/verificationpage',
            arguments: {
              "emotion": emotion,
              "capturedImageFile": capturedImageFile
            });
          await stopCameraAndModel();
        }
      }
      // Yüzde 100 olduğunda sayfaya geçiş
      if (progress == 1.0) {
        await captureImage();
        cameraController!.stopImageStream();
        Navigator.pushReplacementNamed(context, '/verificationpage',
          arguments: {
            "emotion": emotion,
            "capturedImageFile": capturedImageFile
          });
        await stopCameraAndModel();
      }
    }
  }
} catch (e) {
  debugPrint("Error running model: $e");
} finally {
  isModelBusy =
    false;
}

```

Şekil 13 Modeli Çalıştırma ve Duygu Analizi

Kamera akışından alınan görüntü, modele gönderilir.

Model, emotion sonucunu döndürür ve bu duygu sayacı güncellenir.

Kullanıcı aynı duyguya 50 kez ulaştığında fotoğraf çekilir ve yeni sayfaya yönlendirilir.

```
Future<void> captureImage() async {  
  final image = await cameraController!.takePicture();  
  setState(() {  
    capturedImageFile =  
      File(image.path);  
  });  
}
```

Şekil 14 Fotoğraf Çekme ve Kaydetme

Kamera kullanılarak fotoğraf çekilir ve dosya olarak kaydedilir.

Kullanıcı Deneyimi

Başlangıç:

Kullanıcı sayfayı açtığı anda kamera otomatik olarak başlar ve yüzünü algılar.

Duygu Analizi:

TensorFlow Lite modeli, kullanıcının yüz ifadesini analiz eder ve ekranda mevcut duyguyu gösterir.

İlerleme Göstergesi:

Kullanıcının belirli bir duyguya ulaşma ilerlemesi dairesel bir çubukta gösterilir.

Fotoğraf Çekimi:

Kullanıcı 50 kez aynı duyguyu gösterdiğinde otomatik olarak fotoğraf çekilir.

Navigasyon:

Fotoğraf çekildikten sonra kullanıcı '/verificationpage' sayfasına yönlendirilir.

Kontroller:

Kullanıcı kamera arasında geçiş yapabilir (ön/arka).

DOĞRULAMA SAYFASI

VerificationPage, kullanıcının algılanan duygusunu ve çekilen fotoğrafı gösterir. Kullanıcı bu sayfa üzerinden duygusunu onaylayabilir veya değiştirebilir. Ayrıca kullanıcı duygusunu onaylarsa fotoğraf Firebase Storage'a yüklenir ve kullanıcı ilgili duyguyla öneri sayfasına yönlendirilir.



Şekil 15 Doğrulama Sayfası

Sayfa Özellikleri

Başlık ve Geri Dönüş:

Sayfanın üst kısmında AppBar ile bir başlık bulunur.

Sol üst köşede bir geri dönüş butonu yer alır ve ana sayfaya yönlendirir.

Fotoğraf ve Duygu Gösterimi:

Kullanıcının çektiği fotoğraf ve algılanan duygu gösterilir.

Fotoğraf ve duygu bilgileri ModalRoute aracılığıyla önceki sayfadan alınır.

Duygu Onaylama Butonları:

Kullanıcının duygusunu onaylaması için 3 farklı buton: **Mutlu, Üzgün, Öfkeli.**

Kullanıcı butona tıkladığında Firebase Storage'a fotoğraf yüklenir ve ilgili öneri sayfasına yönlendirilir.

Geri Dön Butonu:

Kullanıcı duygu analizine geri dönmek isterse, bu buton ile kamera sayfasına geri döner.

```

Future<void> resmiFirebaseYukle(File resimDosyasi, String duygu) async {
    try {
        final String dosyaAdi = path.basename(resimDosyasi.path);
        final String duyguKlasoru = duyguKlasorunuGetir(duygu);
        final Reference ref =
            FirebaseStorage.instance.ref('$duyguKlasoru/$dosyaAdi');

        await ref.putFile(resimDosyasi);
        print('Resim Firebase Storage\'a başarıyla yüklendi: $dosyaAdi');
    } catch (e) {
        print('Resim yüklenemedi: $e');
    }
}

```

Şekil 16 Firebase Storage'a Fotoğraf Yükleme Fonksiyonu

```

// Algılanan duyguya göre klasör adını belirle
String duyguKlasorunuGetir(String duygu) {
    switch (duygu) {
        case 'Mutlu':
            return 'Mutlu';
        case 'Üzgün':
            return 'Üzgün';
        case 'Öfkeli':
            return 'Öfkeli';
        default:
            throw Exception('Tanımlanmayan duygu: $duygu');
    }
}

```

Şekil 17 Duygu Klasörü Belirleme

Kullanıcı Deneyimi

Anlaşılır Arayüz:

Kullanıcı dostu ve minimal bir arayüz.

Kolay Navigasyon:

Geri dönme ve duygu seçerek öneri sayfasına yönlendirme butonları mevcut.

Hızlı Geri Bildirim:

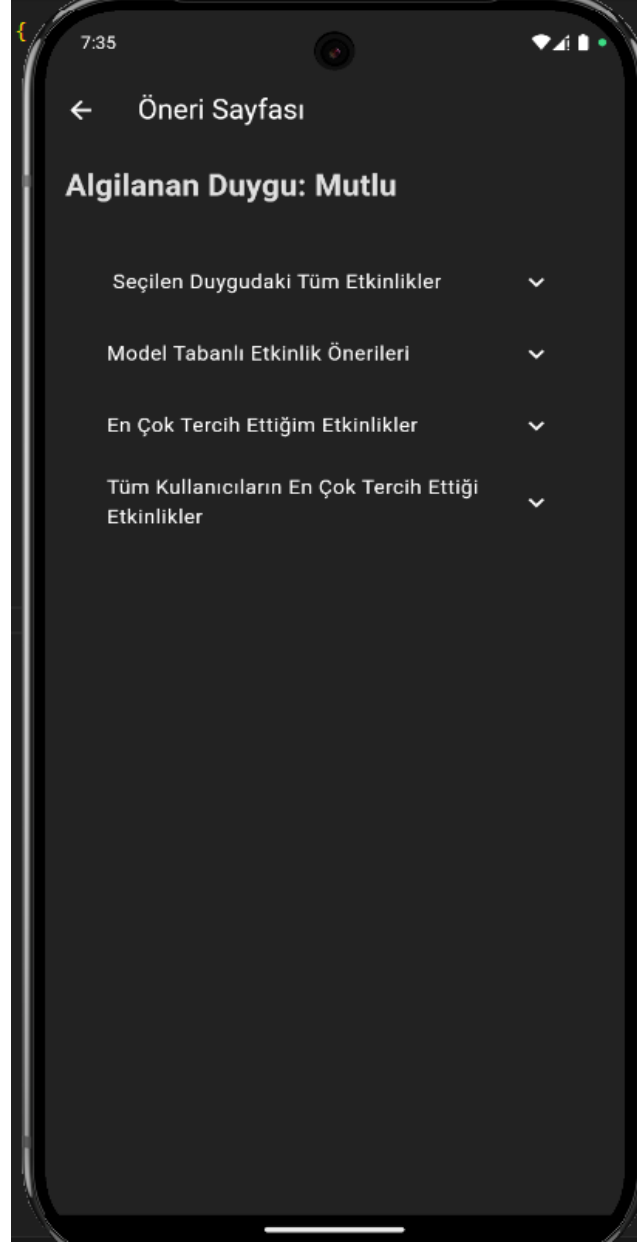
Duygu seçildiğinde hemen işlem gerçekleştirilir.

Görsel Destek:

Fotoğraf görüntülenerek kullanıcının daha iyi doğrulama yapabilmesi sağlanır.

ÖNERİ SAYFASI

SuggestionPage sayfası, bir kullanıcının duygu durumuna bağlı olarak öneriler sunar. Firebase ve HTTP istekleri aracılığıyla hem kullanıcının hem de genel veri tabanındaki verilere dayalı olarak öneriler getirir. Kullanıcının mevcut ruh hali, yaş, cinsiyet ve mesleği gibi faktörler önerileri etkiler.



Şekil 18 Öneri Sayfası

Sayfa Özellikleri

Firestore Entegrasyonu:

Kullanıcının verilerini FirebaseAuth ve Firestore aracılığıyla çeker.

HTTP İstekleri:

Bir Flask API'ye http paketi ile POST istekleri gönderir.

Duygu Analizi:

Kullanıcının duygusuna göre öneriler sunar.

Veri Kaynakları:

Kullanıcının kendisinin en çok tercih ettiği aktiviteler, diğer kullanıcıların tercihi ve model tabanlı öneriler.

Durum Yönetimi:

setState yöntemi kullanılarak öneriler dinamik olarak güncellenir.

Yüklenme Durumu:

_isLoading boolean değişkeni ile sayfa yüklenme durumunu kontrol eder.

```
Future<DocumentSnapshot?> _getUserDocument() async {  
  try {  
    return await FirebaseFirestore.instance  
      .collection("Users")  
      .doc(FirebaseAuth.instance.currentUser!.email)  
      .get();  
  } catch (e) {  
    print('Error fetching user document: $e');  
    return null;  
  }  
}
```

Şekil 19 Firestore Kullanıcı Bilgilerini Çekme

Kullanıcının yaş, cinsiyet ve meslek bilgilerini Firestore'dan çeker.


```
Future<http.Response> _postRequest(
  String endpoint, Map<String, dynamic> body) async {
  return await http.post(
    Uri.parse(endpoint),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8'
    },
    body: jsonEncode(body),
  );
}
```

Şekil 20 HTTP POST İsteği ile Öneri Alma

Flask tabanlı etkinlik öneri modeline veri gönderir ve önerileri JSON formatında döner.

```
Future<void> _fetchActivitySuggestion() async {
  final args = _getRouteArguments();
  if (args == null) return;
  print(args['emotion']);

  final userDoc = await _getUserDocument();
  if (userDoc == null) return;

  int age;
  if (userDoc['Age'] is String) {
    age = int.parse(userDoc['Age']);
  } else {
    age = userDoc['Age'];
  }

  String gender = userDoc['Gender'];
  String job = userDoc['Occupation'];

  String moodType = args["emotion"].toLowerCase();

  try {
    final response =
      await _postRequest('http://$url:5000/predict/$moodType', {
        'yas': age,
        'meslek': job,
        'cinsiyet': gender,
        'mood': args["emotion"].toLowerCase(),
      });
  }
```

Şekil 21 Duygu Durumuna Göre Öneri Getirme

Kullanıcının belirttiği duyguya göre API'den öneriler çeker.

Kullanıcı Deneyimi

Basit ve Anlaşılır:

Sayfa, algılanan duyguyu ve önerileri temiz bir şekilde sunar.

Duygu Temelli Öneriler:

Kullanıcının ruh haline göre özel öneriler gösterir.

Dinamik Veri Güncellemesi:

Öneriler ve duygu durumu dinamik olarak güncellenir.

Yüklenme Göstergesi:

Sayfa yüklenirken bir CircularProgressIndicator gösterir.

Kişiselleştirme:

Kullanıcının önceki tercihleri ve genel kullanıcı tercihleri ayrı ayrı gösterilir.

AKTİVİTE SAYFASI

ActivityPage sayfası, seçilen aktivite önerisinin gösterildiği ve kullanıcının bu aktiviteyi onaylayabileceği bir sayfadır. Kullanıcı bu sayfa üzerinden önerilen aktivitenin detaylarını görebilir ve "Etkinliği Onayla" butonuna basarak etkinliği onaylayabilir.



Şekil 22 Aktivite Sayfası

Sayfa Özellikleri

Aktivite ve Duygu Gösterimi:

Sayfa, kullanıcının önerilen aktiviteyi ve bu aktivitenin önerildiği duygu durumunu gösterir.

Açıklama:

Aktiviteyle ilgili bilgilendirici bir metin sunar.

Onaylama:

Kullanıcı, "Etkinliği Onayla" butonuna basarak aktiviteyi Firebase Firestore'a kaydedebilir.

Navigasyon:

Aktivite onaylandıktan sonra, kullanıcı ApprovedActivitiesPage sayfasına yönlendirilir.

```
final querySnapshot = await userDoc
  .where('activityName', isEqualTo: suggestion)
  .where('mood', isEqualTo: mood)
  .get();

if (querySnapshot.docs.isEmpty) {
  await userDoc.add({
    'activityName': suggestion,
    'mood': mood,
    'approvalDate': DateTime.now().toIso8601String(),
    'count': 1,
  });
} else {
  final docId = querySnapshot.docs.single.id;
  final docRef = userDoc.doc(docId);

  await docRef.update({
    'count': FieldValue.increment(1),
    'lastApprovalDate': DateTime.now().toIso8601String(),
  });
}
```

Şekil 23 Etkinlik Kaydetme veya Güncelleme

Eğer önerilen aktivite daha önce onaylanmamışsa, Firestore'a yeni bir kayıt eklenir.

Eğer aktivite zaten varsa, mevcut kaydın count değeri artırılır ve lastApprovalDate güncellenir.

```
String getActivitySuggestion(String activity) {
    switch (activity) {
        case 'Yazma Terapisi':
            return 'Yazma terapisi, duygularınızı ifade etmenin ve zihinsel rahatlama sağlamanın harika bir yoludur.';
        case 'Meditasyon Yapmak':
            return 'Meditasyon yapmak, zihninizi sakinleştirir ve stres seviyenizi azaltır.';
        case 'Arkadaşlarla Buluşmak':
            return 'Arkadaşlarla bir araya gelmek, sosyal bağlarınızı güçlendirir ve keyifli anlar yaşamamanızı sağlar.';
        case 'Konsere Gitmek':
            return 'Canlı müzik dinlemek, enerjinizi artırır ve eğlenceli bir deneyim sunar.';
        case 'Kitap Okumak':
            return 'Kitap okumak, yeni dünyalara açılan kapılar sunar ve hayal gücünüzü geliştirir.';
    }
}
```

Şekil 24 Etkinlik Açıklamalarını Sağlayan Fonksiyon

Bu fonksiyon, her aktivite için özel bir açıklama döndürür.

```
MyButton(
    text: "Etkinliği Onayla",
    onTap: () => {
        _approveActivity(context),
        Navigator.pop(context),
        Navigator.pushReplacementNamed(
            context, '/approvedactivitiespage')
    },
) // MyButton
```

Şekil 25 Buton ile Onaylama ve Yönlendirme

Kullanıcı butona bastığında `_approveActivity` fonksiyonu çalışır ve kullanıcıyı `ApprovedActivitiesPage` sayfasına yönlendirir.

Kullanıcı Deneyimi

Basit ve Anlaşılır Arayüz:

Kullanıcıya önerilen aktivite ve kullanıcının ruh hali net bir şekilde sunulur.

Bilgilendirici İçerik:

Aktiviteyle ilgili açıklayıcı metin, kullanıcının aktivitenin faydalarını anlamasını sağlar.

Tek Tıklamayla Kaydetme:

Kullanıcı, Etkinliği Onayla butonuna tıklayarak aktiviteyi kolayca onaylayabilir.

Bildirim Geri Bildirimi:

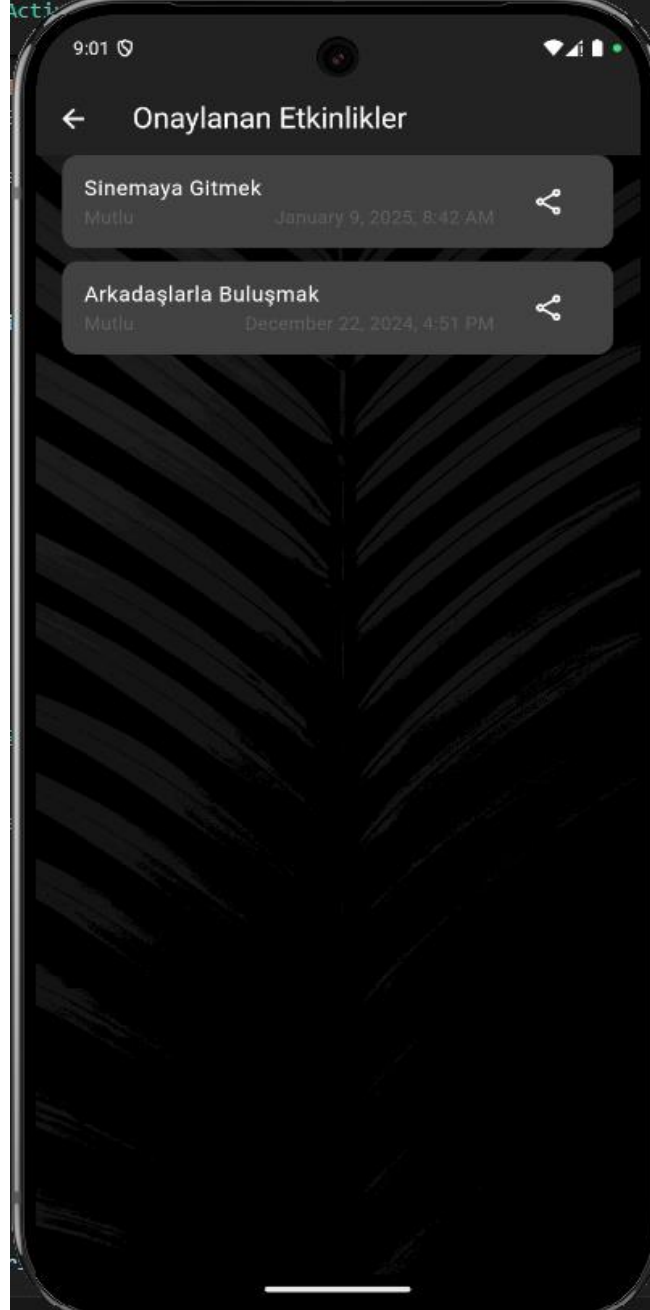
Aktivite başarıyla onaylandığında `SnackBar` ile bildirim gösterilir.

Otomatik Yönlendirme:

Aktivite onaylandıktan sonra kullanıcı, onaylanan aktiviteler sayfasına yönlendirilir.

ONAYLANAN AKTİVİTELER SAYFASI

ApprovedActivitiesPage, onaylanmış etkinlikleri listeleyen ve kullanıcıların bu etkinlikleri platformda paylaşmasını sağlayan bir Flutter sayfasıdır. Firebase Firestore üzerinden gerçek zamanlı veri akışı sağlar.



Şekil 26 Onaylanan Aktiviteler Sayfası

Sayfa Özellikleri

Gerçek Zamanlı Veri Akışı:

Firestore kullanılarak onaylanmış etkinlikler gerçek zamanlı olarak gösterilir.

Listeleme:

Onaylanan etkinlikler, ListView.builder widget'ı kullanılarak listelenir.

Paylaşım İmkanı:

Kullanıcı, listedeki etkinlikleri seçerek açıklama ekleyip paylaşabilir.

Veritabanı:

Firebase Firestore ile çalışır ve ApprovedActivities koleksiyonundan veri çeker.

```
stream: firestore
  .collection('Users')
  .doc(user?.email)
  .collection('ApprovedActivities')
  .snapshots(),
```

Şekil 27 Firestore Akışı

Kullanıcının onaylanan etkinliklerini gerçek zamanlı olarak dinler.

StreamBuilder widget'ı sayesinde veri değiştikçe otomatik güncellenir.

```

itemBuilder: (context, index) {
  final activity = approvedActivities[index];
  String activityName =
    activity['activityName'] ?? "Bilinmiyor";
  String mood = activity['mood'] ?? "Bilinmiyor";
  DateTime approvalDate =
    DateTime.parse(activity['approvalDate']);
  return MyListTile(
    title: activityName,
    subTitle: mood,
    time: approvalDate,
    onEdit: () => publishActivity(activityName, mood),
    actionType: 'publish',
  ); // MyListTile
},

```

Şekil 28 Veri Listeleme

Etkinlikler ListView.builder içinde gösterilir.

Her öğede MyListTile widget'ı ile kullanıcı dostu bir görünüm sağlanır.


```

await showDialog(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      title: const Text('Gönderi Paylaş'),
      content: TextField(
        onChanged: (value) {
          comment = value;
        },
        maxLines: null,
        keyboardType: TextInputType.multiline,
        decoration: InputDecoration(
          hintText: "Biraz aktiviteden bahsetsene",
          hintStyle:
            TextStyle(color: Theme.of(context).colorScheme.secondary),
        ), // InputDecoration
      ), // TextField
      actions: [
        TextButton(
          onPressed: () {
            Navigator.of(context).pop(); // Dialogdan çık
          },
          child: const Text('İptal'),
        ), // TextButton
        TextButton(
          onPressed: () {
            // Burada dialogu kapatmadan önce işlemi başlatıyoruz
            Navigator.of(context).pop(); // Dialogu kapat

            // Firestore'da veri kaydetme işlemini başlat
            publishToFirestore(activityName, mood, comment);
          },
          child: Text(
            'Paylaş',
            style: TextStyle(color: Theme.of(context).colorScheme.primary),
          ), // Text
        ), // TextButton
      ],
    ); // AlertDialog
  },
);

```

Şekil 29 Gönderi Paylaşma Diyaloğu

Kullanıcı bir etkinliği paylaşırken yorum ekleyebilmesi için bir AlertDialog penceresi açılır.[12]

Yorum boşsa paylaşım yapılmaz ve hata mesajı gösterilir.

```
await firestore.collection('PublishedActivities').add({
  'activityName': activityName,
  'mood': mood,
  'UserEmail': user!.email,
  'PostMessage': comment,
  'TimeStamp': Timestamp.now(),
});
```

Şekil 30 Firestore'a Veri Kaydı

Firestore koleksiyonuna veri eklenir.

Başarı ya da hata mesajı, ScaffoldMessenger aracılığıyla gösterilir.[11]

Kullanıcı Deneyimi

Kolay Kullanım:

Etkinliklerin görüntülenmesi ve paylaşımı için basit bir arayüz sunar.

Gerçek Zamanlı Güncellenme:

Etkinlikler eklenir eklenmez listede güncellenir.

Doğrulama:

Yorum boşsa hata mesajı gösterilir.

Etkileşim:

Kullanıcı, her etkinliğin yanında bulunan paylaş butonuna basarak paylaşım yapabilir.

PROFİL SAYFASI


ProfilPage, kullanıcıların profil bilgilerini görüntüleyip düzenleyebileceği, ayrıca sosyal medya benzeri özellikler sunarak gönderilerini yönetebileceği imkanı sağlar.



Şekil 31 Profil Sayfası

10:18

← Profilim



Adınız
Hüseyin

Soyadınız
Çağlar


Yaşınız
22

Cinsiyetiniz ☒ Erkek ☐ Kadın

Öğrenci ▼ ✕

Konumunuz
Koza Mh., Koza Mh., Esen

Konum bilgisine izin ver
Konuma izin verilmedi.

 Kaydet

Şekil 32 Profil Düzenleme

Sayfa Özellikleri

Profil Bilgileri Görüntüleme:

Kullanıcı, adını, soyadını, yaşını, cinsiyetini, mesleğini ve konumunu görüntüleyebilir.

Bu bilgiler Firebase Firestore veritabanından çekilir.

Profil Düzenleme:

Kullanıcı, profil bilgilerini değiştirebilir. Bilgiler düzenlenebilir alanlarda görüntülenir ve kullanıcı bu alanları düzenleyebilir.

Düzenleme işlemi TextField widget'ları ve bir SearchableDropdown aracılığıyla yapılır.

Meslek Seçimi:

Kullanıcı, bir arama kutusu ile meslek seçebilir. Bu özellik, SearchableDropdown widget'ı kullanılarak gerçekleştirilir. Kullanıcı, meslekler listesinden istediğini arayarak seçebilir.

Gönderi Düzenleme ve Silme:

Kullanıcı, daha önce yaptığı gönderileri düzenleyebilir veya silebilir.

Gönderi düzenleme, bir TextField aracılığıyla yapılır. Silme işlemi, kullanıcıya onay sormadan yapılmaz. Onay penceresi kullanılır.

```
SearchableDropdown.single(  
  items: occupations.map((occupation) {  
    return DropdownMenuItem<String>(  
      value: occupation,  
      child: Text(occupation),  
    ); // DropdownMenuItem  
  }).toList(),  
  value: selectedOccupation,  
  hint: "Mesleğiniz",  
  searchHint: "Mesleğinizi arayın",  
  onChanged: (value) {  
    setState(() {  
      selectedOccupation = value;  
    });  
  },  
  isExpanded: true,  
) // SearchableDropdown.single
```

Şekil 33 Meslek Seçimi (SearchableDropdown)

Meslek seçimi için SearchableDropdown kullanılır. Bu dropdown, kullanıcıların meslekleri arayarak hızlıca seçmesini sağlar

```
content: TextField(
  controller: editPostController,
  decoration: const InputDecoration(hintText: "Gönderini düzenle"),
), // TextField
```

```
// Silme onayı
showDialog(
  context: context,
  builder: (context) {
    return AlertDialog(
      title: const Text('Silmek İstediğinize Emin Misiniz?'),
      content: const Text('Bu gönderiyi silmek istiyorsunuz.'),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context); // Onay penceresini kapat
          },
          child: const Text('Hayır'),
        ), // TextButton
        TextButton(
          onPressed: () {
            // Gönderiyi sil
            final postRef = FirebaseFirestore.instance
              .collection('Posts')
              .doc(postId);
            final publishedActivitiesRef = FirebaseFirestore
              .instance
              .collection('PublishedActivities')
              .doc(
                postId); // "PublishedActivities" koleksiyonundan silmek için
```

Şekil 34 Gönderi Düzenleme ve Silme

Kullanıcı Deneyimi

Kullanıcı Bilgilerinin Düzenlenmesi:

Kullanıcılar, bilgilerini hızlıca görüntüleyip düzenleyebilirler. Bilgilerin metin alanlarında düzenlenebilir olması, kullanıcılara esneklik sağlar. TextField ve SearchableDropdown gibi widget'lar, kullanıcının girdiği bilgilere göre dinamik olarak güncellenir.

Meslek Seçimi Kolaylığı:

Meslekler, arama özelliği sayesinde kullanıcı tarafından hızlıca bulunabilir. Bu, uzun bir liste içinde kaybolmayı engeller ve kullanıcı dostu bir deneyim sağlar.

Gönderi Yönetimi:

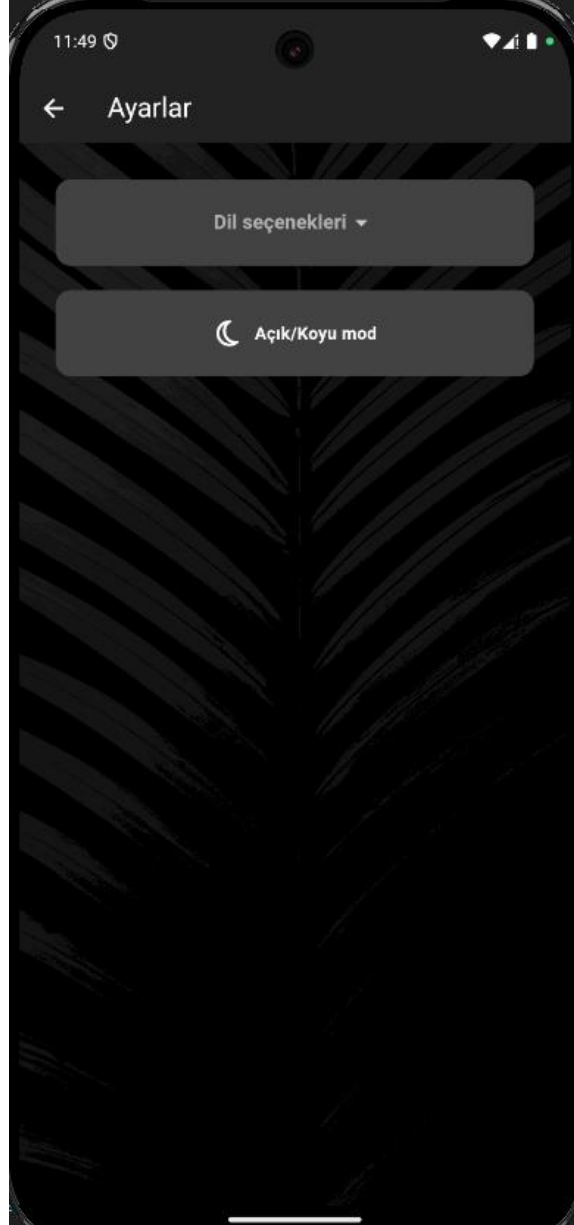
Gönderi düzenleme ve silme işlemleri çok basittir. TextField ile düzenleme, silme işlemi ise onay penceresi ile güvenli hale getirilmiştir.

Bilgi Akışı ve UI Akışı:

Sayfa tasarımı, kullanıcıların ihtiyaç duydukları bilgilere kolayca ulaşmalarını sağlar. Bilgiler düzenli bir şekilde ekranda yer alır ve düzenleme butonlarıyla kullanıcıya işlem yapma imkanı sunulur.

AYARLAR SAYFASI

SettingPage sayfasında kullanıcılar tema (karanlık/aydınlık) arasında geçiş yapma ayarını bu sayfa üzerinden kontrol edebilir.



Şekil 35 Ayarlar Sayfası


```

MyBotton(
  text: AppLocalizations.of(context)!.darkLight,
  onTap: () {
    Provider.of<ThemeProvider>(context, listen: false)
      .toggleTheme();
  },
  icon: Icon(
    isDarkMode
      ? FontAwesomeIcons.moon
      : FontAwesomeIcons.lightbulb,
    color: isDarkMode ? Colors.white : Colors.black,
  ), // Icon
), // MyBotton

```

Şekil 36 Tema Değişimi

MyBotton: Tema değiştirme için özel bir butondur.

Provider: Tema değişimini yönetmek için kullanılır. ThemeProvider sınıfı üzerinden tema kontrolü sağlanır.

FontAwesomeIcons: Tema durumuna göre ay ve ampul ikonları kullanılır.

```

body: BackgroundContainer(
  child: Center(
    child: Padding(
      padding: const EdgeInsets.all(30),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [

```

Şekil 37 Arka Plan

BackgroundContainer: Sayfaya özel bir arka plan eklemek için kullanılır.

Center: İçerikleri ortalamak için kullanılır.

Padding: İçeriklerin kenarlardan boşluklu görünmesini sağlar.

GERİ BİLDİRİM SAYFASI

FeedbackPage, kullanıcıların etkinlik öneri tavsiyesinde bulunmalarını sağlamak için tasarlanmıştır. Kullanıcı, kategori seçebilir, fikirlerini yazabilir ve gönder butonuna basarak verileri Firebase Firestore veritabanına kaydedebilir.

12:00

← Geri Bildirim

Lütfen geri bildirim kategorisini seçin:

✓ Mutlu Üzgün Öfkeli

Geri bildiriminizi buraya yazın:

Geri bildiriminizi buraya yazın...

Gönder

Şekil 38 Geri Bildirim Sayfası

Sayfa Özellikleri

Firestore:

Kullanıcı geri bildirimlerini Firestore veritabanına kaydeder.

Auth:

Kullanıcının mevcut bilgilerini (yaş, cinsiyet, meslek) çekmek için kullanılır.

Kullanıcı Bilgileri:

Kullanıcının yaşı, cinsiyeti ve mesleği otomatik olarak doldurulur.

Geri Bildirim Kategorileri:

Kullanıcı, Mutlu, Üzgün, Öfkeli kategorilerinden birini seçer.

Özelleştirilmiş Snackbar:

Geri bildirim başarılı veya hatalı olduğunda ekranın ortasında özel bir mesaj gösterilir.

Stateful Widget:

Kullanıcı etkileşimlerini ve verileri saklamak için StatefulWidget kullanılmıştır.[9]

```

// Firebase'e geri bildirim gönder
Future<void> _submitFeedback() async {
  if (_feedbackController.text.isNotEmpty) {
    try {
      String category = _selectedCategory ?? 'Mutlu';

      await FirebaseFirestore.instance.collection('Feedbacks').add({
        'Feedback': _feedbackController.text,
        'Category': category,
        'Age': _ageController.text,
        'Gender': _genderController.text,
        'Occupation': _occupationController.text,
        'Timestamp': FieldValue.serverTimestamp(),
      });

      // Başarılı mesajı göster
      await _showCustomSnackbar('Geri bildiriminiz alındı!');

      // Alanları temizle
      _feedbackController.clear();
      _ageController.clear();
      _genderController.clear();
      _occupationController.clear();

      setState(() {
        _selectedCategory = 'Mutlu';
      });

      await _fetchUserInfo();
    } catch (e) {
      await _showCustomSnackbar(
        'Geri bildirim gönderilirken bir hata oluştu.');
```

Şekil 39 Geri Bildirim Gönderme

Kullanıcının girdiği verileri Firebase Firestore'daki Feedbacks koleksiyonuna ekler.

```

Future<void> _showCustomSnackbar(String message) async {
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return Center(
        child: Container(
          padding: const EdgeInsets.symmetric(vertical: 16, horizontal: 24),
          decoration: BoxDecoration(
            color: Colors.black87,
            borderRadius: BorderRadius.circular(8),
          ), // BoxDecoration
          child: Text(
            message,
            style: const TextStyle(
              color: Colors.white,
              fontSize: 16,
              decoration: TextDecoration.none,
            ), // TextStyle
          ), // Text
        ), // Container
      ); // Center
    },
  );

  await Future.delayed(const Duration(seconds: 3));
  if (mounted) Navigator.of(context).pop();
}

```

Şekil 40 Özel Snackbar Mesajı

Kullanıcıya özel bir geri bildirim mesajı gösterir ve 3 saniye sonra otomatik olarak kapanır.

Kullanıcı Deneyimi

Kolay Kullanım:

Kullanıcı dostu bir arayüzle basit geri bildirim verme imkanı.

Kategori Seçimi:

Kullanıcı, ChoiceChip bileşenleri sayesinde kategori seçebiliyor.

Otomatik Bilgi Doldurma:

Kullanıcı bilgileri Firebase'den otomatik çekilip dolduruluyor.

Anlık Bildirim:

Geri bildirim başarıyla gönderildiğinde veya hata oluştuğunda ekranda anlık uyarı mesajı gösteriliyor.

Düzenli ve Temiz Arayüz:

Basit ve anlaşılır bir form yapısı mevcut.

ÜÇÜNCÜ BÖLÜM

ETKİNLİK ÖNERİ MODELİ

Bu model, kullanıcının ruh hali, yaşı, cinsiyeti ve mesleği gibi bilgileri kullanarak etkinlik önerileri sunar. Kullanıcıların geçmiş etkinlik seçimlerini ve ruh halleriyle olan ilişkilerini analiz ederek önerilerde bulunur.

Kullanılan Teknolojiler

Python:

Ana programlama dili.

Flask:

API sunucusu için kullanılmıştır.

Firebase Firestore:

Kullanıcı verilerinin saklandığı NoSQL veritabanı.

Scikit-Learn:

Makine öğrenmesi algoritmaları için (Random Forest Classifier).

Pandas & NumPy:

Veri işleme ve manipülasyonu için.

Joblib:

Makine öğrenmesi modellerini kaydetmek ve yüklemek için.

Firebase Admin SDK:

Firebase veritabanına erişim için.

Özellikler

Ruh Hali Temelli Tahmin:

Kullanıcının ruh haline (Mutlu, Üzgün, Öfkeli) göre öneriler sunar.

Makine Öğrenmesi:

Random Forest modeli kullanarak etkinlik tahminleri yapar.

Veritabanı Entegrasyonu:

Kullanıcı verilerini ve önerileri Firebase Firestore'dan çeker.

Web API:

Flask ile RESTful API olarak çalışır.

Eğitim ve Tahmin:

Kullanıcı verileriyle modeli eğitip yeni tahminlerde bulunabilir.


```

def train_model_for_mood(mood_type):
    users_ref = db.collection('Users')
    all_data = []

    for user_doc in users_ref.stream():
        user_data = user_doc.to_dict()
        activities_ref = user_doc.reference.collection('ApprovedActivities')
        for activity_doc in activities_ref.stream():
            activity_data = activity_doc.to_dict()

            mood = normalize_mood(activity_data.get('mood', ''))
            if mood == mood_type:
                print(f"Veri Eklendi: {activity_data}")
                all_data.append({
                    "yas": user_data.get('Age'),
                    "meslek": user_data.get('Occupation'),
                    "cinsiyet": user_data.get('Gender'),
                    "mood": mood,
                    "activity": activity_data.get('activityName')
                })

    if not all_data:
        print(f"Uyar: {mood_type} için veri bulunamadı!")
        return None

    df = pd.DataFrame(all_data)
    le_mood, le_meslek, le_cinsiyet, le_activity = LabelEncoder(), LabelEncoder(), LabelEncoder(), LabelEncoder()

    df['mood'] = le_mood.fit_transform(df['mood'])
    df['meslek'] = le_meslek.fit_transform(df['meslek'])
    df['cinsiyet'] = le_cinsiyet.fit_transform(df['cinsiyet'])
    df['activity'] = le_activity.fit_transform(df['activity'])

    X = df[['yas', 'meslek', 'cinsiyet', 'mood']]
    y = df['activity']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestClassifier(n_estimators=100)
    model.fit(X_train, y_train)

    accuracy = model.score(X_test, y_test)
    print(f"{mood_type} Model Doğruluk Oranı: {accuracy:.2f}")

    joblib.dump(model, f'activity_model_{mood_type}.pkl')
    joblib.dump(le_meslek, f'le_meslek_{mood_type}.pkl')
    joblib.dump(le_cinsiyet, f'le_cinsiyet_{mood_type}.pkl')
    joblib.dump(le_activity, f'le_activity_{mood_type}.pkl')

    return f"{mood_type} modeli başarıyla kaydedildi."

```

Şekil 41 Model Eğitim Fonksiyonu

Amaç: Belirtilen ruh hali için Random Forest Classifier modelini eğitir ve kaydeder.

Veri Kaynağı: Firebase Firestore'dan kullanıcıların onayladığı aktiviteler çekilir.

Özellikler: Yaş, cinsiyet, meslek, ruh hali.

Model: Random Forest Classifier.

Kaydetme: Eğitim tamamlandıktan sonra model .pkl dosyası olarak kaydedilir.

```

@app.route('/predict/<mood_type>', methods=['POST'])
def predict(mood_type):
    """
    Belirli bir mood türü için tahmin yapar.
    """
    data = request.json
    yas = data.get('yas')
    meslek = data.get('meslek')
    cinsiyet = data.get('cinsiyet')

    if not all([yas, meslek, cinsiyet]):
        return jsonify({"error": "Eksik veri girişi."}), 400

    if mood_type not in ['mutlu', 'uzgun', 'ofkeli']:
        return jsonify({"error": "Geçersiz mood türü."}), 400

    model = joblib.load(f'activity_model_{mood_type}.pkl')
    le_meslek = joblib.load(f'le_meslek_{mood_type}.pkl')
    le_cinsiyet = joblib.load(f'le_cinsiyet_{mood_type}.pkl')
    le_activity = joblib.load(f'le_activity_{mood_type}.pkl')

    features = [
        yas,
        le_meslek.transform([meslek])[0],
        le_cinsiyet.transform([cinsiyet])[0],
        0
    ]
    prediction = model.predict_proba([features])
    top_indices = prediction[0].argsort()[-6:][::-1]
    top_activities = le_activity.inverse_transform(top_indices)
    print(prediction)

    return jsonify({"predictions": top_activities.tolist()})

```

Şekil 42 Tahmin API'si

Amaç: Kullanıcının yaşı, mesleği, cinsiyeti ve ruh hali verilerine göre önerilen 6 etkinliği döndürür.

Kullanılan Teknoloji: RandomForestClassifier modeli, .pkl dosyalarıyla yüklenir.

Girdi: Kullanıcı bilgileri (yaş, cinsiyet, meslek, duygu durumu).

Çıktı: En çok önerilen 6 etkinlik.

```
@app.route('/train_all_models', methods=['GET'])
def train_all_models():
    results = []
    for mood in ['mutlu', 'uzgun', 'ofkeli']:
        result = train_model_for_mood(mood)
        if result:
            results.append({mood: result})
        else:
            results.append({mood: f"{mood} modeli için yeterli veri yok."})
    return jsonify(results)
```

Şekil 43 Tüm Modelleri Eğitme

Amaç: Tüm ruh halleri için (mutlu, üzgün, öfkeli) modelleri eğitir ve kaydeder.

Çağrılan Fonksiyon: train_model_for_mood fonksiyonu gelir.

```
def get_most_frequent_activity_all_users(mood):
    """
    Tüm kullanıcılar için belirli bir duygusal durum için en çok tercih edilen 7 aktiviteyi döner.
    """
    users_ref = db.collection('Users')
    activity_counts = {}

    # Tüm kullanıcıların aktivitelerini çek
    users = users_ref.stream()

    for user in users:
        approved_activities_ref = user.reference.collection('ApprovedActivities')
        docs = approved_activities_ref.stream()

        for doc in docs:
            activity_data = doc.to_dict()
            if activity_data['mood'].lower() == mood:
                activity_name = activity_data['activityName']
                count = activity_data['count']
                activity_counts[activity_name] = activity_counts.get(activity_name, 0) + count

    if not activity_counts:
        return []

    # Count'a göre azalan sırada ilk 7 aktivitenin adını döndür
    most_frequent_activities = sorted(activity_counts, key=activity_counts.get, reverse=True)[:7]

    return most_frequent_activities
```

Şekil 44 En Çok Tercih Edilen Etkinlikler

Amaç: Tüm kullanıcılar arasında en çok tercih edilen etkinlikleri döndürür.

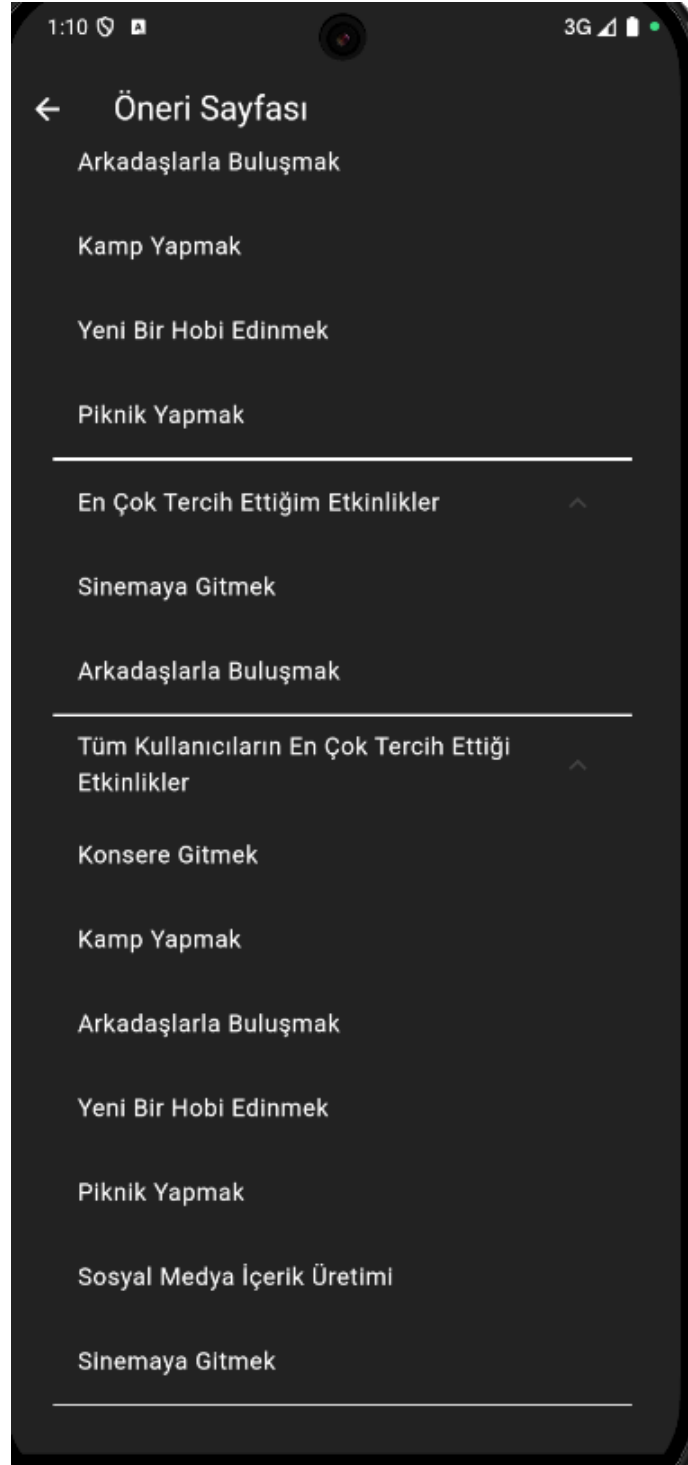
Veri Kaynağı: ApprovedActivities koleksiyonundan veri çeker.

Dönüş: En popüler 7 etkinliği listeler.

Örnek Kullanım



Şekil 45 Model Kullanımı 1



Şekil 46 Model Kullanımı 2

DUYGU ANALİZİ MODELİ

Duygu analizi modelini oluşturmak için **transfer learning** yaklaşımı kullanılır. **MobileNetV2** modelini temel alarak, bir **konvolüsyonel sinir ağı (CNN)** oluşturuluyor. Model, verilen duygu türüne (mutlu, üzgün, öfkeli) göre görüntüleri sınıflandıracak şekilde eğitilmiştir.

Kullanılan Teknolojiler

TensorFlow ve Keras

TensorFlow, açık kaynaklı bir makine öğrenimi (ML) kütüphanesidir ve derin öğrenme (DL) modelleri oluşturmak, eğitmek ve dağıtmak için yaygın olarak kullanılır.

Keras, TensorFlow'un içinde yer alan yüksek seviyeli bir API'dir. Derin öğrenme modelleri oluşturmayı ve eğitmeyi kolaylaştıran soyutlamalar sağlar. Keras, modelin hızlı bir şekilde prototipini oluşturmak için kullanılır.[16]

Bu modelde, TensorFlow ve Keras kullanılarak derin öğrenme modeli oluşturulmuş ve eğitilmiştir.

MobileNetV2 (Transfer Learning)

MobileNetV2, Google tarafından geliştirilmiş, mobil cihazlarda çalışabilecek şekilde optimize edilmiş bir konvolüsyonel sinir ağı (CNN) modelidir. Çeşitli görsel tanıma görevleri için güçlü bir özellik çıkarıcıdır. Bu model, transfer öğrenme (transfer learning) tekniğiyle kullanılmıştır.[17]

Transfer Learning: Transfer öğrenme, zaten önceden eğitilmiş bir modelin (MobileNetV2 burada) özelliklerinin başka bir görevde kullanılmasını ifade eder. Bu model, ImageNet üzerinde önceden eğitilmiş ve daha sonra belirtilen özel görevde kullanılmak üzere yeniden eğitilmiştir.

```
base_model = MobileNetV2(input_shape=(img_size, img_size, 3), include_top=False, weights='imagenet')
base_model.trainable = True
for layer in base_model.layers[:100]:
    layer.trainable = False
```

Şekil 47 MobilNetV2

MobileNetV2 modeli, ImageNet üzerinde önceden eğitilmiş ve son katmanları çıkarılmıştır (include_top=False). İlk 100 katman donmuş (freeze) bırakılmıştır, böylece sadece son katmanlar yeniden eğitilir.

Transfer Learning (Aktarım Öğrenmesi) Kullanmak

İlk Katmanlar, Genel Özellikleri Öğrenmiştir: MobileNetV2, ImageNet üzerinde eğitilmiş bir modeldir ve bu eğitim sırasında model, temel görsel özellikleri (kenarlar, şekiller, desenler) öğrenmiştir. Bu özellikler genellikle birçok farklı görev için geçerlidir. Bu nedenle, modelin ilk katmanlarını yeniden eğitmeye gerek yoktur. Bu katmanları "donmuş" bırakmak, gereksiz hesaplama yükünden kaçınarak eğitim süresini kısaltır.

Verimli Öğrenme: Son katmanları yeniden eğitmek, modelin belirli bir probleme (örneğin, üç sınıflı bir sınıflandırma problemi) odaklanmasını sağlar. Bu, modelin daha verimli bir şekilde öğrenmesine ve daha iyi genelleme yapmasına olanak tanır.

Ağırlıkların Donması (Freezing)

Zaman ve Hesaplama Tasarrufu: İlk katmanlar (özellikle 100 katman) zaten geliştirilmiş özellikleri öğrenmişken, onları yeniden eğitmek gereksiz hesaplama maliyeti yaratır. Bu nedenle, bu katmanları "donmuş" bırakmak, modelin eğitim sürecini hızlandırır.

Özellik Çıkarmada Verimlilik: Bu yaklaşım, modelin sadece son katmanlarını yeniden eğiterek, yeni veri kümesindeki özellikleri tanımasını sağlar. Model, genel özellikleri öğrenmek yerine sadece hedef verinin özgül özelliklerini öğrenir.

Overfitting (Aşırı Öğrenme) Riskini Azaltma: Yeni bir problem üzerinde eğitim yaparken, ilk katmanların "donması" modeli genel özellikler üzerine odaklanmaya zorlar ve aşırı öğrenme riskini azaltır. Model, fazla özelleşmeden genel özellikleri kullanarak doğru tahminler yapar.

Daha Az Veri ile Eğitim

Transfer learning sayesinde, büyük miktarda etiketli veri gerekmemektedir. Çünkü model, ImageNet üzerinde çok geniş bir veri setiyle eğitildiği için, yeni görevde daha küçük veri setleriyle daha iyi sonuçlar verebilir. Bu sayede, hedef problem için sınırlı veriye sahip olsanız bile başarılı sonuçlar elde edebilirsiniz.

Convolutional Neural Networks (CNN)

CNN (Konvolüsyonel Sinir Ağları), görüntü tanıma, video analizi, doğal dil işleme gibi alanlarda yaygın olarak kullanılan derin öğrenme modelidir. Bu modelde, konvolüsyonel katmanlar (Conv2D) ve maksimum havuzlama (MaxPooling2D) katmanları görüntü verilerinden özellikler çıkarmak için kullanılmıştır.

Bu modelde, MobileNetV2'nin önceden eğitilmiş katmanları kullanılarak, daha fazla özelleştirilmiş katmanlar eklenmiştir: bir konvolüsyon katmanı, bir havuzlama katmanı ve birkaç tam bağlantılı (fully connected) katman.

```
# Model oluşturma
model = Sequential([
    base_model,
    Conv2D(256, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dropout(0.5),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
```

Şekil 48 Convolutional Neural Networks (CNN)

Base model (MobileNetV2): Önceden eğitilmiş bir model kullanmak, transfer öğrenme (transfer learning) ile genel özelliklerin öğrenilmesini sağlar, bu da daha hızlı ve verimli eğitim sunar.

Conv2D (256, (3, 3), activation='relu'): 256 filtre, daha fazla özellik çıkarımı sağlar. 3x3 filtre boyutu yerel özelliklerin öğrenilmesine olanak tanır. ReLU, doğrusal olmayan özellikler öğrenmeyi sağlar.

MaxPooling2D ((2, 2)): Bu, görsellerdeki en belirgin özellikleri çıkararak boyutları küçültür ve hesaplama gücünden tasarruf sağlar.

Flatten(): Konvolüsyon ve havuzlama katmanlarından çıkan verileri düzleştirir, böylece sınıflandırmaya geçilebilir.

Dropout (0.5): Modelin aşırı öğrenmesini engeller. Eğitim sırasında nöronları rastgele kapatarak genelleme gücünü artırır.

Dense (512, activation='relu'): 512 nöron, daha fazla özellik öğrenmesini sağlar. ReLU, doğrusal olmayan ilişkiler öğrenir.

Dense (num_classes, activation='softmax'): Softmax, sınıf olasılıklarını hesaplar ve en yüksek olasılığa sahip sınıfı tahmin eder.

Bu ayarlar, modelin etkili özellik çıkarımı yapmasını ve genelleme kapasitesini artırmasını sağlamak için seçilmiştir.

ImageDataGenerator (Veri Artırma)

ImageDataGenerator, TensorFlow/Keras tarafından sağlanan bir araçtır ve görüntü verileriyle veri artırma (data augmentation) yapmak için kullanılır. Bu modelde, eğitim verilerini çoğaltarak modelin daha iyi genelleştirme yapmasını sağlamak amacıyla birkaç dönüşüm yapılmıştır:[18]

```
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    shear_range=0.4,  
    zoom_range=0.4,  
    rotation_range=30,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    horizontal_flip=True  
)
```

Şekil 49 ImageDataGenerator (Veri Artırma)

rescale=1./255: Görsellerin değerleri [0, 1] aralığına getirilir, böylece daha iyi eğitim sağlanır.

shear_range=0.4, zoom_range=0.4, rotation_range=30: Görsellerin döndürülmesi, zoom yapılması ve kaydırılması modelin daha farklı varyasyonlar öğrenmesini sağlar.

width_shift_range=0.2, height_shift_range=0.2: Görsellerde yatay ve dikey kaydırma, daha fazla çeşitlilik sağlar.

horizontal_flip=True: Yatay çevirme, veri çeşitliliğini artırır ve modelin daha esnek olmasını sağlar.

Bu işlemler, modelin daha sağlam ve genelleyici öğrenmesini hedefler.

Learning Rate Scheduler

Learning Rate Scheduler, modelin eğitim sürecinde öğrenme oranını dinamik olarak değiştirmek için kullanılır. Bu modelde, LearningRateScheduler kullanılarak, eğitimde belirli bir sayıda epoch'tan sonra öğrenme oranı düşürülür. Bu, modelin daha hassas bir şekilde öğrenmesini sağlamak için faydalıdır.

Eğitim başlangıcında sabit bir öğrenme oranı kullanılır, ancak epoch sayısı arttıkça, öğrenme oranı %10 oranında düşürülür.

```
def scheduler(epoch, lr):  
    if epoch < 10:  
        return lr  
    elif lr > 5e-4:  
        return lr * 0.9  
    else:  
        return lr  
  
lr_scheduler = LearningRateScheduler(scheduler)
```

Şekil 50 Learning Rate Scheduler

Epoch < 10: İlk 10 epoch boyunca öğrenme oranı sabit tutulur, böylece model başlangıçta hızlıca öğrenir.

lr > 5e-4: Öğrenme oranı 5e-4'ün üzerine çıktığında, öğrenme oranı %10 azaltılır ($lr * 0.9$), bu sayede modelin daha stabil bir şekilde öğrenmesi sağlanır.

Else: Öğrenme oranı çok düşükse, daha fazla azaltılmadan kalır.

Bu ayarlamalar, modelin erken aşamalarda hızlı öğrenmesini, ilerleyen aşamalarda ise daha hassas ve kararlı bir şekilde öğrenmesini sağlar.

Callbacks (EarlyStopping ve ReduceLROnPlateau)

EarlyStopping, modelin doğrulama kaybı belirli bir süre boyunca iyileşmediyse eğitimin erken sonlandırılmasını sağlar. Bu, aşırı öğrenmeyi (overfitting) önlemeye yardımcı olur.[19]

ReduceLROnPlateau, modelin doğrulama kaybı iyileşmedikçe öğrenme oranını düşürür. Bu, eğitim sürecinin daha verimli hale gelmesini sağlar.

```
early_stopping = EarlyStopping(  
    monitor='val_loss',  
    patience=15,  
    restore_best_weights=True  
)  
  
reduce_lr = ReduceLROnPlateau(  
    monitor='val_loss',  
    factor=0.9,  
    patience=5,  
    min_lr=1e-3  
)
```

Şekil 51 Callbacks (EarlyStopping ve ReduceLROnPlateau)

EarlyStopping: Modelin doğrulama kaybı iyileşmediğinde, eğitim erken sonlandırılır. Bu, aşırı öğrenmeyi (overfitting) önler ve gereksiz eğitim sürelerini kısaltır. `patience=15` ile 15 epoch boyunca iyileşme görülmezse eğitim durdurulur.

ReduceLROnPlateau: Doğrulama kaybı iyileşmediğinde, öğrenme oranı azaltılır (`factor=0.9`). Bu, modelin daha hassas öğrenmesini sağlar. `patience=5` ile 5 epoch boyunca gelişme görülmezse, öğrenme oranı düşürülür. Minimum öğrenme oranı ise `min_lr=1e-3` ile sınırlandırılır.

Model Kaydetme ve Dönüştürme

Model eğitildikten sonra, **Keras formatında** (.keras uzantılı dosya) kaydedilir. Bu, modelin daha sonra yeniden kullanılabilir ve başka ortamlarda da çalıştırılabilir hale gelmesini sağlar.

Ayrıca, model **TensorFlow Lite** (.tflite) formatına dönüştürülür. **TensorFlow Lite**, mobil ve gömülü cihazlarda çalışan, optimize edilmiş bir model formatıdır. Bu sayede modelin performansı, kaynakları kısıtlı cihazlarda da yüksek olabilir.

```
model_save_path = 'trained_emotion_model2.keras'
model.save(model_save_path)
print(f"Model başarıyla kaydedildi: {model_save_path}")

try:
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    tflite_model = converter.convert()
    with open('model.tflite', 'wb') as f:
        f.write(tflite_model)
    print("Model başarıyla .tflite formatına dönüştürüldü.")
except Exception as e:
    print(f".tflite dönüşümünde hata oluştu: {e}")
```

Şekil 52 Model Kaydetme ve Dönüştürme

Matplotlib ile Sonuçların Görselleştirilmesi

Eğitim sırasında elde edilen doğruluk ve kayıp değerleri görselleştirilmiştir.

Matplotlib kütüphanesi kullanılarak:

Eğitim ve doğrulama doğruluğu (accuracy) grafiği

Eğitim ve doğrulama kaybı (loss) grafiği

Bu görselleştirmeler, modelin eğitim sürecinin izlenmesine ve iyileştirilmesi gereken alanların belirlenmesine yardımcı olur.

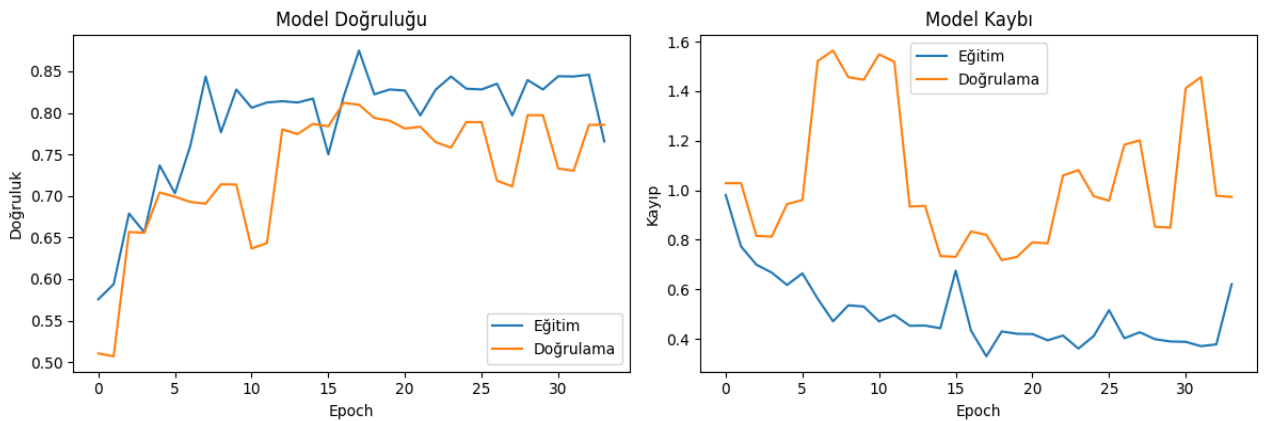
```
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Eğitim')
plt.plot(history.history['val_accuracy'], label='Doğrulama')
plt.title('Model Doğruluğu')
plt.xlabel('Epoch')
plt.ylabel('Doğruluk')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Eğitim')
plt.plot(history.history['val_loss'], label='Doğrulama')
plt.title('Model Kaybı')
plt.xlabel('Epoch')
plt.ylabel('Kayıp')
plt.legend()

plt.tight_layout()
plt.show()
```

Şekil 53 Sonuçların Görselleştirilmesi



Şekil 54 Model Sonuçları

Model Değerlendirme (Test)

Modelin son değerlendirmesi, **test verisi** üzerinde yapılmıştır. Eğitim sonrası, modelin test doğruluğu ve kaybı hesaplanarak modelin genel performansı ölçülmüştür.

```
try:
    test_loss, test_accuracy = model.evaluate(test_generator)
    print(f"Test kaybı: {test_loss:.4f}, Test doğruluğu: {test_accuracy:.4f}")
except Exception as e:
    print(f"Test sırasında hata oluştu: {e}")
```

Şekil 55 Model Değerlendirme

Eğitim Sonuçları:

Doğruluk (Accuracy): 73.96%

Kayıp (Loss): 0.7384 Bu, modelin eğitim setinde %73.96 doğruluk sağladığını ve kaybının yaklaşık 0.7384 olduğunu gösterir. Bu, modelin eğitim sürecinde iyi bir performans sergilediğini gösterir.

Test Sonuçları:

Test Doğruluğu: 79.91%

Test Kaybı: 0.5527 Test setindeki doğruluk oranı %79.91 ve kayıp 0.5527'dir. Bu, modelin doğruluğunun test verilerinde arttığını ve kaybın azaldığını gösteriyor, bu da modelin genelleme kapasitesinin iyi olduğunu ve test verileri üzerinde de başarı sağladığını ifade eder.

Genel Değerlendirme:

Model, eğitim ve test verileri arasında iyi bir denge kurmuş gibi görünüyor. Eğitim kaybı biraz daha yüksekken, test kaybı daha düşük ve doğruluk oranı artmış. Bu, modelin overfitting (aşırı uyum) yapmadığını ve doğru bir şekilde genelleme yapabildiğini gösteriyor.

DÖRDÜNCÜ BÖLÜM

FİREBASE

Activities > mutlu			More in Google Cloud
(default)	Activities	mutlu	
+ Start collection	+ Add document	+ Start collection	
Activities >	mutlu >	+ Add field	
Feedbacks	öfkeli	activity1: "Aile ile Vakit Geçirmek"	
Posts	üzgün	activity10: "DIY Projeleriyle Uğraşmak"	
Users		activity11: "Maker Atölyelerine Katılmak"	
		activity12: "Teknoloji Podcast'leri Dinlemek"	
		activity13: "Sağlık Ve Wellness Etkinliklerine Katılmak"	
		activity14: "Kütüphaneye Gitmek"	
		activity15: "Sinemaya Gitmek"	
		activity16: "Sanat Etkinliklerine Katılmak"	
		activity2: "Arkadaşlarla Buluşmak"	
		activity3: "Sosyal Medya İçerik Üretimi"	
		activity4: "Yoga Yapmak"	
		activity5: "Sergi Gezme"	
		activity7: "Kamp Yapmak"	
		activity8: "Konsere Gitmek"	
		activity9: "Piknik Yapmak"	

Şekil 56 Aktiviteler

Tüm aktiviteler Firebase’de duygu durumlarına göre sınıflandırılarak tutuluyor.

Feedbacks > ckuzTnLrXXZr5x			More in Google Cloud
(default)	Feedbacks	ckuzTnLrXXZr5xFpbKku	
+ Start collection	+ Add document	+ Start collection	
Activities	6BjnV4F5Efz9JghhQRL4	+ Add field	
Feedbacks >	ckuzTnLrXXZr5xFpbKku >	Age: "22"	
Posts		Category: "Mutlu"	
Users		Feedback: "Mutluyken yemek yapmayı çok severim "	
		Gender: "Erkek"	
		Occupation: "Öğrenci"	
		Timestamp: January 9, 2025 at 3:10:41 PM UTC+3	

Şekil 57 Geri Bildirim

Kullanıcılardan alınan geri bildirimler gönderen kullanıcının bilgileriyle beraber tutuluyor.

Home > Posts > Qs7Xroyx5reYW. More in Google Cloud		
<div>(default)</div> <div> <div>+ Start collection</div> <div>Activities</div> <div>Feedbacks</div> <div>Posts</div> <div>Users</div> </div>	<div>Posts</div> <div> <div>+ Add document</div> <div>Qs7Xroyx5reYW7MWGtd</div> </div>	<div>Qs7Xroyx5reYW7MWGtd</div> <div> <div>+ Start collection</div> <div>+ Add field</div> <div>PostMessage: "mutluyken kahve içirim."</div> <div>TimeStamp: November 23, 2024 at 8:46:41 PM UTC+3</div> <div>UserEmail: "hus@gmail.com"</div> <div>activityName: "kahve içmek"</div> <div>mood: "Mutlu"</div> </div>

Şekil 58 Paylaşılan Gönderiler

Kullanıcıların paylaştıkları gönderiler bu şekilde tutuluyor.

Home > Users > hus@gmail.com > ApprovedActiviti... > ei9npflkImyvWR. More in Google Cloud		
<div>hus@gmail.com</div> <div> <div>+ Start collection</div> <div>ApprovedActivities</div> </div> <div> <div>+ Add field</div> <div>Age: 22 (number)</div> <div>Email: "hus@gmail.com" Edit field</div> <div>Gender: "Erkek"</div> <div>Location: "Koza Mh., Koza Mh., Esenyurt, 34538"</div> <div>Name: "Hüseyin"</div> <div>Occupation: "Finans Çalışanı"</div> <div>Surname: "Çağlar"</div> <div>user_id: "hus@gmail.com"</div> </div>	<div>ApprovedActivities</div> <div> <div>+ Add document</div> <div>ZxxuYBdw8m7Vpb8Jb0sK</div> <div>ei9npflkImyvWRVVBfrP</div> </div>	<div>ei9npflkImyvWRVVBfrP</div> <div> <div>+ Start collection</div> <div>+ Add field</div> <div>activityName: "Arkadaşlarla Buluşmak"</div> <div>approvalDate: "2024-12-22T16:51:01.717501"</div> <div>count: 1</div> <div>mood: "Mutlu"</div> </div>

Şekil 59 Kullanıcılar

Kullanıcılar kayıt edildiği koleksiyonda onayladıkları aktiviteler de saklanılıyor.

KAYNAKÇA

Kullanılan veri setleri Kaynakları:

Arkhanzada, A. (2025). *Facial emotions classification - AffectNet dataset*. Kaggle. Erişim adresi: <https://www.kaggle.com/code/arkhanzada/facial-emotions-classification-affectnet-dataset/input?select=anger>

Yasserhessein, (2025). *Emotion recognition with ResNet50*. Kaggle. Erişim adresi: <https://www.kaggle.com/code/yasserhessein/emotion-recognition-with-resnet50/input>

Flutter ve Firebase ile ilgili Kaynaklar:

- 1) Flutter. (n.d.). *Flutter documentation - State management*. Flutter.dev. Erişim adresi: <https://flutter.dev/docs/get-started/flutter-for/ios-devs>
- 2) Flutter. (n.d.). *Flutter widgets*. Flutter.dev. Erişim adresi: <https://flutter.dev/docs/development/ui/widgets>
- 3) Firebase. (n.d.). *Firestore usage with Flutter*. Firebase Documentation. Erişim adresi: <https://firebase.flutter.dev/docs/firestore/usage>
- 4) Firebase. (n.d.). *Authentication in Flutter*. Firebase Documentation. Erişim adresi: <https://firebase.flutter.dev/docs/auth/usage>
- 5) Flutter. (n.d.). *Provider package*. pub.dev. Erişim adresi: <https://pub.dev/packages/provider>
- 6) Firebase. (n.d.). *Cloud Firestore for Flutter*. Firebase Documentation. Erişim adresi: <https://firebase.flutter.dev/docs/firestore/usage>

Flutter ile İlgili Sınıflar ve Widget'lar:

- 7) Flutter. (n.d.). *StreamBuilder class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/widgets/StreamBuilder-class.html>
- 8) Flutter. (n.d.). *FloatingActionButton class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/material/FloatingActionButton-class.html>
- 9) Flutter. (n.d.). *StatefulWidget class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html>
- 10) Flutter. (n.d.). *TextField class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/material/TextField-class.html>
- 11) Flutter. (n.d.). *Scaffold class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/material/Scaffold-class.html>
- 12) Flutter. (n.d.). *AlertDialog class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/material/AlertDialog-class.html>
- 13) Flutter. (n.d.). *TextButton class*. Flutter API. Erişim adresi:
<https://api.flutter.dev/flutter/material/TextButton-class.html>

TensorFlow ve Keras İle İlgili Kaynaklar:

- 14) TensorFlow. (2025). *TensorFlow documentation*. Erişim adresi: <https://www.tensorflow.org>
- 15) TensorFlow. (2025). *TensorFlow Lite: Machine learning for mobile and embedded devices*. Erişim adresi: <https://www.tensorflow.org/lite>
- 16) Chollet, F. (2015). *Keras*. Erişim adresi: <https://keras.io>
- 17) Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4510-4520). IEEE. <https://doi.org/10.1109/CVPR.2018.00474>
- 18) Keras. (n.d.). *ImageDataGenerator class*. Erişim adresi: <https://keras.io/api/preprocessing/image/>
- 19) Keras. (n.d.). *EarlyStopping class*. Erişim adresi: https://keras.io/api/callbacks/early_stopping/

Diğer Kaynaklar:

- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95. <https://doi.org/10.1109/MCSE.2007.55>
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362. <https://doi.org/10.1038/s41586-020-2649-2>
- Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.
- Firebase. (2025). *Firebase Admin SDK*. Erişim adresi: <https://firebase.google.com/docs/admin/setup>
- McKinney, W. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (pp. 51-56). <https://doi.org/10.25080/Majora-92bf1922-00a>
- Sanke, M., Furtado, S., Naik, S., Nevagi, S., & Bidikar, V. (2023). Emotion based movie recommendation system using deep learning. *International Journal of Computer Applications*, 185(20), 49-54. <https://doi.org/10.5120/ijca2023922924>