İSTİNYE ÜNİVERSİTESİ İSTANBUL

Yazılım Proje Yönetimi Dersi Proje Raporu

Mühendislik ve Doğa Bilimleri Fakültesi

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Proje Adı: Green Points

SEÇİL ŞENER
ŞEVVAL KAPÇAK
CEM BOZKURT
EDA MENEKŞEYURT - 200701056
KÜBRA AKPUNAR - 200701092

İÇİNDEKİLER

Proje Konusu

- Geri dönüşüm, kullanılmış ve tekrar kullanılmayacak olan maddelerin işlenmesi ve tüketici ile buluşmasına verilen isimdir. Geri dönüşüm enerji tasarrufu sağlanması, doğal kaynakların ve çevrenin korunması gibi konularda oldukça önemlidir.
- Biz bu projede görüntü işleme ve nesne tespiti ile geri dönüşüm uygulaması yaptık. Kullanıcıdan alınan fotoğraflar görüntü işleme ile kağıt, metal, plastik gibi kategorilere ayrıldı.
- Kullanıcının fotoğraf yükleyebilmesi için kullanıcı dostu bir uygulama yapıldı.
- Kullanıcının çeşitli kategoriler için farklı puanlar kazanabileceği bir sistem inşa edildi.
- Uygulamada kullanıcıların kazanacağı puan miktarına göre ödüllendirme yapılmakta, bu sayede insanlar geri dönüşüme teşvik edilmektedir.

Aşağıda projenin bölümleri başlıklara ayrılmış ve detaylı açıklanmıştır.

1) Görüntü İşleme

Projemizin görüntü işleme ile geri dönüşüm sınıflandırılması bölümü Python programlama dili ile yazılmıştır. Python görüntü işleme için kapsamlı kütüphane desteği sunan, makine öğrenmesi ve derin öğrenme için entegre edilmiş kütüphanelerle çalışma ortamı sunan, kolay kullanım ve çapraz platform desteği sunan bir programlama dilidir. Projenin görüntü işleme bölümü için en efektif programlama dilinin Python olabileceği öngörülmüştür.
Bu kısımda amaç, bir görüntünün geri dönüşüm için sınıflandırılmasıdır. Cam, metal, batarya, kağıt, plastik, biyolojik olmak üzere 6 farklı kategori için sınıflandırma yapılmıştır. Bunun için "https://www.kaggle.com/datasets/mostafaabla/garbage-classification" veri seti kullanılmıştır. Önce gereksiz kategoriler temizlenmiş, daha sonra karmaşıklık yaratabilecek görüntüler silinmiştir. Beyaz-yeşil-kahverengi cam olarak ayrılan 3 kategori "cam" kategorisi adı altında birleştirilmiş, overfitting olmaması adına veri sayısı diğer kategorilerle benzer sayıya indirgenmiştir. Daha sonra veri seti ile model eğitilmiş ve performansı test edilmiştir.

MODELİN EĞİTİLMESİ:

```
# Verilerin yuklenmesi ve islenmesi
veri_seti = 'geridonusum_train'

class_names = os.listdir(veri_seti)
class_paths = [os.path.join(veri_seti, cls) for cls in class_names]

images = []
labels = []
for i, cls_path in enumerate(class_paths):
  for img_name in os.listdir(cls_path):
    img_path = os.path.join(cls_path, img_name)
    try:
      img = cv2.imread(img_path)
      if img is None:
        print(f"Resim yüklenemedi: {img_path}")
        continue
      img = cv2.resize(img, (128, 128))
      images.append(img)
      labels.append(i)
    except Exception as e:
```

```
        print(f"Hata: {e} - Dosya: {img_path}")

images = np.array(images)
labels = np.array(labels)
```

Burada veri seti içeri aktarılmış ve sınıf isimleri alınmıştır. Daha sonra veri setinin içerisinde gezinerek görüntü dosyaları okunmuştur, eğer görüntü dosyaları okunamazsa gerekli hata mesajlarının ekrana yazdırılması sağlanmıştır. Görüntüler 128x128 piksel boyutunda yeniden boyutlandırılır, görüntü ve etiket bilgisi saklanır.

```
# Veri setinin egitim ve test seti olarak bolunmesi
train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size=0.2,
random_state=42)

# CNN modelinin olusturulmasi
model = models.Sequential([
  layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
  layers.MaxPooling2D((2, 2)),
  layers.Conv2D(64, (3, 3), activation='relu'),
  layers.MaxPooling2D((2, 2)),
  layers.Conv2D(128, (3, 3), activation='relu'),
  layers.MaxPooling2D((2, 2)),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dropout(0.5),
  layers.Dense(len(class_names), activation='softmax')
])

# Modelin compile edilmesi
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Modelin egitilmesi
history = model.fit(train_images, train_labels, epochs=10, batch_size=32,
validation_data=(test_images, test_labels))

# Modelin kaydedilmesi
model.save("geridonusum_modeli.h5")
```

Veri seti %80 eğitim ve %20 test olmak üzere bölünmüştür. CNN modeli oluşturulmuş ve compile edilmiştir. Modelin performansını değerlendirmek için metrik olarak doğruluk (accuracy) belirterek model derlenmiştir. Model eğitilmiş ve eğitim sürecindeki bilgileri içeren “history” nesnesi tutulmuştur. Model daha sonraki kullanımlarda tekrar eğitilmeye ihtiyaç duyulmaması adına “geridonusum_modeli.h5” ismi ile kaydedilmiştir.

```
# labels.txt dosyasına class isimlerini yazdırma
with open('labels.txt', 'w') as f:
  for class_name in class_names:
      f.write(f"{class_name}\n")

# loss ve accuracy degerlerinin goruntulenmesi
train_loss = history.history['loss']
train_acc = history.history['accuracy']
epochs = range(1, len(train_loss) + 1)

plt.figure(figsize=(8, 4))
```

```
plt.plot(epochs, train_loss, 'b', label='Eğitim Kaybı')
plt.plot(epochs, train_acc, 'r', label='Eğitim Doğruluğu')
plt.title('Eğitim Kaybı ve Doğruluğu')
plt.xlabel('Epochs')
plt.ylabel('Loss / Accuracy')
plt.legend()

plt.show()
```

Mobil uygulamayla entegre edilmesi için class isimleri label.txt dosyasına yazdırılmıştır. Modelin eğitimi sırasında kaybı ve doğruluğu izlemek için grafik oluşturulmuştur. Bu grafik, her bir epoch için eğitim kaybını ve doğruluğunu göstermektedir. Grafik boyutu, eksen etiketleri ve başlık gibi grafik bileşenlerinin düzeni ayarlanmıştır. Eğitim kaybı için mavi, eğitim doğruluğu için kırmızı renkler kullanarak, grafikteki farklı eğrileri ayırt edilebilmesi ve modelin performansının görselleştirilmesi sağlanmıştır.

MODELİN TEST EDİLMESİ:

```
# Modelin yuklenmesi
model = tf.keras.models.load_model('geridonusum_modeli.h5')

# Tahmin edilmesi istenen goruntuler
resimler = 'geridonusum_veriseti'

class_names = os.listdir(resimler)
class_paths = [os.path.join(resimler, cls) for cls in class_names]

images = []
labels = []
for i, cls_path in enumerate(class_paths):
  for img_name in os.listdir(cls_path):
    img_path = os.path.join(cls_path, img_name)
    try:
      img = cv2.imread(img_path)
      if img is None:
        print(f"Resim yüklenemedi: {img_path}")
        continue
      img = cv2.resize(img, (128, 128))
      images.append(img)
      labels.append(i)
    except Exception as e:
      print(f"Hata: {e} - Dosya: {img_path}")

images = np.array(images)
labels = np.array(labels)
```

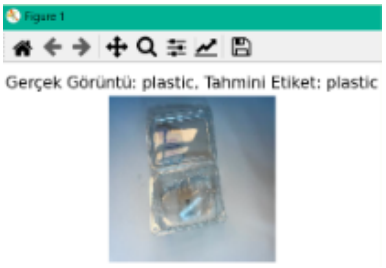
Daha önce eğitilip kaydedilmiş model yüklenmiştir. Tahmin edilmesi istenen görüntülerin klasörü bir değişkende tutulmuştur. Daha sonra veri setinin içerisinde gezinerek görüntü dosyaları okunmuştur, eğer görüntü dosyaları okunamazsa gerekli hata mesajlarının ekrana yazdırılması sağlanmıştır. Görüntüler 128x128 piksel boyutunda yeniden boyutlandırılır, görüntü ve etiket bilgisi saklanır.

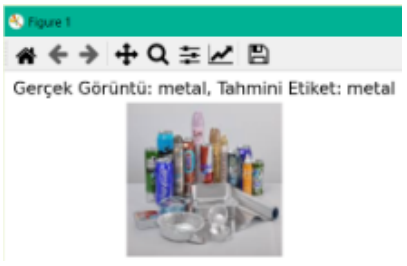
```
train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size=0.2,
random_state=42)

# 50 adet ornek gosterilmesi
num_samples_to_display = 50

# Rastgele orneklerin indekslerinin secilmesi
random_indexes = np.random.choice(len(test_images), num_samples_to_display, replace=False)
```

Veri seti %80 eğitim ve %20 test olmak üzere bölünmüştür. Göstermek istenen rastgele örneklerin sayısı 50 olarak belirlenmiştir. np.random.choice(len(test_images), num_samples_to_display, replace=False) kısmı test setinden rastgele örneklerin gösterilmesi için rastgele indekslerin seçilmesini sağlamıştır. replace=False ile aynı örneğin birden fazla kez seçilmesi önlenmiştir.

```
dogru = 0
for i, idx in enumerate(random_indexes):
  sample_image = test_images[idx]
  sample_label = test_labels[idx]

  prediction = model.predict(np.expand_dims(sample_image, axis=0))
  predicted_class = np.argmax(prediction)

  # Gercek ve tahmin etiketlerinin yazdirilmasi
  print(f"Örnek {i + 1}:")
  print("Gerçek Etiket:", class_names[sample_label])

  # Görüntüleri ekrana bastırma
  plt.figure(figsize=(4, 2))


  plt.subplot(1, 2, 1)
  plt.imshow(sample_image)
  plt.title('Gerçek Görüntü: {}, Tahmini Etiket: {}'.format(class_names[sample_label],
class_names[predicted_class]))
  if class_names[sample_label] == class_names[predicted_class]:
     dogru += 1
  plt.axis('off')

  plt.tight_layout()
  plt.show()

print("dogru sayisi= ", dogru)
```

Doğru tahmin sayısını tutmak için doğru = 0 değişkeni yazılmıştır. Yazılan for döngüsü ile rastgele indexler seçilmiştir. Rastgele seçilen indeks ile test görüntüleri ve etiketleri alınmıştır. prediction = model.predict(np.expand_dims(sample_image, axis=0)) ile modelin örneği tahmin etmesi için kullanılır. np.expand_dims fonksiyonu, örneği bir diziye dönüştürülmüş ve axis=0 parametresiyle diziye bir boyut eklenmiştir bunun yapılmasının sebebi predict fonksiyonun tek bir örnek beklemesidir. predicted_class = np.argmax(prediction) ile tahmin edilen sınıf belirlenmiştir. np.argmax fonksiyonu ile modelin çıktısındaki en yüksek olasılığa sahip sınıfı belirlenmiştir.Gerçek ve tahmin edilen etiketler ekrana yazdırılmıştır. Ayrıca, gerçek ve tahmin edilen etiketlerle görüntü gösterilmiştir.Eğer gerçek etiket ile tahmin edilen etiket aynıysa, dogru sayacı bir artırılmıştır. Tüm örneklerin üzerinden döngü tamamlandıktan sonra, doğru tahmin edilen örneklerin sayısı ekrana yazdırılmıştır.





2) Mobil Uygulama Geliştirme

Mobil uygulamamız Flutter ile geliştirilmiştir. Flutter'ın tercih edilmesindeki sebepler şunlardır; Flutter, tek bir kod tabanı kullanarak iOS ve Android gibi farklı platformlarda çalışabilen çapraz platform bir geliştirme çerçevesidir. Bu, geliştirme sürecini hızlandırır ve maliyetleri düşürür. Ayrıca hızlı prototipleme ve geliştirme süreçleri için öndegelen çözümlerdendir.
Zengin bir widget kütüphanesine sahiptir ve Firebase gibi popüler bulut tabanlı hizmetlerle kolayca entegre edilebilir.
Firebase, uygulama için oturum açma, gerçek zamanlı veritabanı, analitikler, bildirimler ve diğer birçok özellik sunar. Uygulamanın Firebase ile entegrasyonu, kullanıcı verilerinin güvenli bir şekilde saklanmasına ve yönetilmesine yardımcı olur.
Bu sebeplerle Flutter kullanımının, projenin gereksinimlerini en iyi şekilde karşılayacağı ve başarılı bir sonuç elde edilmesine olanak tanıyacağı öngörülmüştür.

Mobil uygulamamızın asıl amacı kullanıcıları geri dönüşüme teşvik etmektir. Bunu da kullanıcıları ödüllendirme yöntemiyle sağladığı düşünülmektedir.

Kullanıcılar mobil uygulamamız sayesinde geri dönüştüreceği malzemeyi elle seçerek ya da kameradan taratıp görüntü işleme ile tespiti yapıldıktan sonra sepete ekleyerek puan kazanmaktadırlar. Bu puanlar geri dönüştürülen malzemenin cinsine göre değişmektedir. Örneğin pil 20 puan iken kağıt 10 puandır. Toplam 6 çeşit ürün kategorisi vardır. Bunlar; *plastic, paper, metal, glass, biological, battery* şeklindedir.

Mobil uygulamamızı daha detaylı incelersek;

KAYDOL - GİRİŞ YAP SAYFASI:

Bu sayfada kullanıcılar, uygulamaya giriş yaparak geri dönüşüm sürecine katılabilirler. Uygulama, kullanıcı adı ve parolayı doğrulayarak kullanıcıları ana sayfaya yönlendirir.Firebase Realtime Database kullanılarak kullanıcı bilgileri doğrulanır ve giriş yapılır.

```
class SignInPage extends StatefulWidget {

  const SignInPage({Key? key}) : super(key: key);
```

```
@override

_SignInPageState createState() => _SignInPageState();}
```

Burada, SignInPage adlı bir Stateful widget sınıfı oluşturuyoruz. Bu, sayfanın durumunu yönetmek için gerekli olan değişikliklerin yapılmasını sağlar. İçerisinde aşağıda tanımlanan widget'ları içerir.

MaterialApp ve Scaffold Widget'ları: MaterialApp genellikle bir Flutter uygulamasının en dışındaki widget'tır ve uygulamanın temel widget ağacını oluşturur. Scaffold widget'ı ise genellikle uygulamanın ana yapılandırma öğesidir ve temel bir sayfa düzeni sağlar.

Scaffold'un backgroundColor Özelliği: Scaffold'un arka plan rengini beyaza ayarlar.

SingleChildScrollView ve Column: SingleChildScrollView, içeriği dikey olarak kaydırılabilen bir widget'tır. İçindeki Column widget'ı, dikey olarak sıralanmış bir dizi widget'ı içerir.

HeaderWidget: HeaderWidget, uygulamanın başlık kısmını oluşturan ve geri dönüşümlü bir widget'tır. Burada, _headerHeight değeri ve bir FontAwesome ikonu kullanılarak başlık oluşturulur.

SafeArea: SafeArea, içindeki widget'ların, ekranın üst ve alt kısmındaki çentikler, kamera çıkıntıları veya başka sistem UI unsurları tarafından engellenmesini önler.

Text Widget'ları: Başlık ve alt başlık metinlerini içerir.

Form Widget'ı: Kullanıcı adı ve parola girişi için bir form alanı oluşturur.

TextField Widget'ları: Kullanıcı adı ve parola girişi için metin alanlarıdır.

ElevatedButton Widget'ı: "Sign In" butonunu oluşturur ve bu butona basıldığında bir profil sayfasına yönlendirme gerçekleşir.

Text.rich Widget'ı: Metin bağlantısı içeren bir metin paragrafı oluşturur. "Create an account" metni bir bağlantıya dönüştürülür ve tıklanabilir hale getirilir. Bu bağlantıya tıklandığında kayıt sayfasına yönlendirilir.

```
Future<String?> _authUser(LoginData data) async {

 debugPrint('Name: ${data.name}, Password: ${data.password}');

 DatabaseReference usersRef = FirebaseDatabase.instance.ref().child("users");

 DatabaseReference conversionsRef = FirebaseDatabase.instance.ref().child("conversions");

 return usersRef.get().then((DataSnapshot snapshot) {

  if (snapshot.exists) {

   Map usersData = snapshot.value as Map;

   if (checkCredentials(data.name, data.password, usersData)) {

    // Kullanıcı doğrulandı, şimdi fotoğrafı dönüşüm tablosuna kaydedelim

    String userId = data.name.split('@')[0]; // Kullanıcı adından benzersiz bir ID oluştur

    // Kullanıcının giriş yaptığı tarihi al
```

```
      DateTime now = DateTime.now();
      String timestamp = now.toIso8601String();
      // Dönüşüm tablosuna fotoğrafı eklemek için bir referans oluştur
      DatabaseReference newConversionRef = conversionsRef.push();
      // Yeni bir dönüşüm nesnesi oluştur
      Map<String, dynamic> newConversion = {
        "userId": userId,
        "timestamp": timestamp,
        "photoUrl": "", // Fotoğraf henüz yüklenmedi, bu alan boş };
      // Dönüşüm nesnesini Firebase Realtime Database'e ekle
      newConversionRef.set(newConversion);
      // Profil sayfasına yönlendir
      Navigator.of(context).pushReplacement(MaterialPageRoute(
        builder: (context) => ProfilePage(email: data.name),));
      return null;
    } else {
      return 'Kullanıcı Bulunamadı';}}});}
```

Kullanıcı girişi doğrulama işlemi bu metodda gerçekleşir. Firebase Realtime Database'den kullanıcı bilgileri alınır ve girilen bilgilerle karşılaştırılır.Eğer doğrulama başarılı ise, kullanıcının profil sayfasına yönlendirilir ve dönüşüm tablosuna kullanıcı verileri kaydedilir.

PROFİL SAYFASI:

Kullanıcılar, bu sayfada kayıtlı bilgilerini değiştirebilir ve uygulama ayarlarına bu sayfadan erişebilir.

```
class ProfilePage extends StatelessWidget {
  final String email;
  ProfilePage({required this.email});
```

ProfilePage sınıfı, bir e-posta parametresi alarak oluşturulur. Bu, kullanıcının profiline erişimi sağlar.

```
String getUserNameFromEmail(String email) {
  return email.split('@')[0];}
```

Bu metod, kullanıcının e-posta adresinden kullanıcı adını ayıklar. E-posta adresi @ işaretine göre ayrılır ve kullanıcı adı döndürülür.

```
@override

Widget build(BuildContext context) {

 String userName = getUserNameFromEmail(email);
```

build metodu, kullanıcı arayüzünü oluşturur. E-posta adresinden kullanıcı adı alınır ve yerel bir değişkene atanır.

```
return Scaffold(

 appBar: AppBar(

  title: Text('Profil'),

  backgroundColor: Colors.green,

  leading: IconButton(

   icon: Icon(Icons.arrow_back),

   onPressed: () {

    Navigator.pushReplacement(

     context,

     MaterialPageRoute(builder: (context) => ucuncuSayfa()),);},),),
```

Scaffold widget'ı, temel yapının oluşturulduğu yerdir. AppBar ögesi, sayfa başlığını ve geri gitme işlevselliğini içerir.

```
body: SingleChildScrollView(

 child: Column(

  children: [

   Stack(

    alignment: Alignment.center,

    children: [

     Container(

      height: 150,

      decoration: BoxDecoration(

       color: Colors.green,

       borderRadius: BorderRadius.vertical(
```

```
            bottom: Radius.circular(30),),),),
        Positioned(
          top: 75,
          child: CircleAvatar(
            radius: 50,
            backgroundImage: NetworkImage(
              'https://via.placeholder.com/150',),),),],),
```

SingleChildScrollView, ekranın kaydırılabilir olmasını sağlar. Column widget'ı, dikey bir düzen oluşturur ve çocukları olan bileşenlerin düzenlenmesini sağlar. Profil fotoğrafı ve bilgileri bir Stack içinde düzenlenir.

```
SizedBox(height: 60),
Text(
  userName,
  style: TextStyle(
    fontSize: 22,
    fontWeight: FontWeight.bold,),),
SizedBox(height: 10),
Text(
  email,
  style: TextStyle(
    fontSize: 16,
    color: Colors.grey[600],),),
```

Kullanıcı adı ve e-posta metinleri, uygun stillerle Text bileşeni içinde gösterilir.

```
SizedBox(height: 20),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 20.0),
  child: Card(
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(15),),
```

```
elevation: 5,
child: Padding(
  padding: const EdgeInsets.all(20.0),
  child: Column(
    children: [
 ListTile(
        leading: Icon(Icons.favorite, color: Colors.red),
        title: Text('Favorilerim'),
        trailing: Icon(Icons.arrow_forward_ios),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => FavoritePage()),);}, ),
      Divider(),
      ListTile(
        leading: Icon(Icons.settings, color: Colors.blue),
        title: Text('Ayarlar'),
        trailing: Icon(Icons.arrow_forward_ios),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => SettingsPage()),); },),
      Divider(),
      ListTile(
        leading: Icon(Icons.info, color: Colors.green),
        title: Text('Hakkında'),
        trailing: Icon(Icons.arrow_forward_ios),
        onTap: () {
```

```
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => AboutPage()),             );  },  ), ],),), ),),
```

Bir kart içinde, kullanıcıya favori ürünlerine, ayarlarına ve uygulama hakkında bilgiye erişim sağlayan ListTile'lar bulunur.

```
SizedBox(height: 20),
ElevatedButton.icon(
 onPressed: () {
   Navigator.pushReplacement(
     context,
     MaterialPageRoute(builder: (context) => SignInPage()),);},
 icon: Icon(Icons.logout),
 label: Text('Çıkış Yap'),
 style: ElevatedButton.styleFrom(
  backgroundColor: Colors.lightGreen,
  padding: EdgeInsets.symmetric(horizontal: 30, vertical: 10),
  textStyle: TextStyle(
    fontSize: 18,),),),
```

Son olarak, çıkış yapma işlevselliğini sağlayan bir düğme bulunur. Bu düğme, ElevatedButton widget'ı ile oluşturulur

ANA SAYFA:

Kullanıcıların uygulamanın belirli sayfalarına yönlendirildiği, giriş yaptıktan sonra profil sayfasından sonra açılan sayfa burasıdır. Burda diğer sayfaları temsil eden ikonlar barda bulunmaktadır. Tıklandığında kullanıcı ilgili sayfalara yönlendirilmektedir.

```
class ucuncuSayfa extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Color(0xFFA5D6A7),
    appBar: AppBar(
```

```
  title: const Text('Green Points'),

  leading: IconButton(

   icon: Icon(Icons.arrow_back),

   onPressed: () {

    Navigator.pushReplacement(

     context,

     MaterialPageRoute(builder: (context) => SignInPage()),

    );
```

ucuncuSayfa adlı bir StatelessWidget sınıfı tanımlanır. Bu sınıfın build yöntemi, sayfanın nasıl görüneceğini ve nasıl davranacağını belirler.

```
  floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

  floatingActionButton: FloatingActionButton(

   child: const Icon(Icons.favorite_outlined,),

   backgroundColor: Colors.green,

   foregroundColor: Colors.white,

   onPressed: () {

    Navigator.push(context, MaterialPageRoute(builder: (context) => HomePage()),);

   },),
```

Ortada yerleştirilmiş bir floating action button (FAB) tanımlanır. Bu düğmeye basıldığında, kullanıcı HomePage adlı sayfaya yönlendirilir. Burda HomePage olarak tanımlanan sınıf aslında "Seç-Dönüştür" sayfasının sınıfıdır.

```
  bottomNavigationBar: BottomAppBar(
   shape: CircularNotchedRectangle(),
   notchMargin: 4.0,
   child: Row(
    mainAxisSize: MainAxisSize.max,
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: <Widget>[
     IconButton(
      icon: Icon(Icons.shopping_basket),
      onPressed: () {
       Navigator.push(context, MaterialPageRoute(builder: (context) => sepet()),);
      },
     ),
     IconButton(
      icon: Icon(Icons.add_a_photo_rounded),
      onPressed: () {
```

```
            Navigator.push(context, MaterialPageRoute(builder: (context) => Barkod()),);
          },
        ),
        IconButton(
          icon: Icon(Icons.newspaper_sharp),
          onPressed: () {
            Navigator.push(context, MaterialPageRoute(builder: (context) => Haber()),);
          },
        ),
        IconButton(
          icon: Icon(Icons.account_circle_rounded),
          onPressed: () {
            Navigator.push(context, MaterialPageRoute(builder: (context) => ikinciSayfa()),);
          },
```

Alt kısımda bir BottomAppBar tanımlanır. Bu çubuk, kullanıcıyı farklı sayfalara yönlendirebilecek dört adet simge içerir:

1. Icons.shopping_basket ile sepet sayfasına.
2. Icons.add_a_photo_rounded ile Barkod sayfasına.
3. Icons.newspaper_sharp ile Haber sayfasına.
4. Icons.account_circle_rounded ile ikinciSayfa sayfasına.

HABERLER SAYFASI:

Kullanıcılar, geri dönüşümle ilgili güncel haberleri bu sayfadan takip edebilir.

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

import 'package:boot1_project/models/article.dart';

import 'package:boot1_project/models/news.dart';

class NewsService {

 static NewsService _singleton = NewsService._internal();

 NewsService._internal();

 factory NewsService() {

  return _singleton;}

 static Future<List<Articles>> getNews() async {

  Uri url =
Uri.parse("https://newsapi.org/v2/everything?q=recycle&apiKey=3807ceba236246c5a2dd6b842a219
156");

  final response = await http.get(url);

  if (response.statusCode == 200 && response.body.isNotEmpty) {

   final responseJson = json.decode(response.body);

   News news = News.fromJson(responseJson);
```

```
    return news.articles;}

  return [];}}
```

- import ifadeleri, gerekli kütüphaneleri projeye dahil eder.
- NewsService sınıfı, haberleri almak için bir servisi temsil eder.
- _singleton değişkeni, sınıfın tekil bir örneğini tutar.
- _internal adında özel bir kurucu yöntem, sınıfın yalnızca içten oluşturulmasını sağlar.
- factory yöntemi, NewsService sınıfının örneklerini döndürür.
- getNews metodu, haberleri bir API'den alır ve Articles türünde bir liste döndürür.

```
import 'article.dart';

class News {

 String status = '';

 int totalResults = 0;

 List<Articles> articles = [];

 News({required this.status, required this.totalResults, required this.articles});

 News.fromJson(Map<String, dynamic> json) {

  status = json['status'] ?? '';

  totalResults = json['totalResults'] ?? 0;

  articles = [];

  if (json['articles'] != null) {

   json['articles'].forEach((v) {

    articles.add(Articles.fromJson(v));

   });}}

 Map<String, dynamic> toJson() {

  final Map<String, dynamic> data = new Map<String, dynamic>();

  data['status'] = this.status;

  data['totalResults'] = this.totalResults;

  if (this.articles != null) {

   data['articles'] = this.articles.map((v) => v.toJson()).toList();}

  return data;}}
```

- import ifadesi, Article sınıfını projeye dahil eder.
- News sınıfı, haber nesnesini temsil eder. Bu sınıf, haberlerin durumu, toplam sonuç sayısı ve makale listesi gibi özelliklere sahiptir.
- fromJson metodu, bir JSON nesnesini News nesnesine dönüştürür.
- toJson metodu, News nesnesini JSON formatına dönüştürür.

TARA - DÖNÜŞTÜR SAYFASI:

Kullanıcılar, barkod tarama veya QR tarama özelliği ile geri dönüştürecekleri malzemeleri hızlıca tanıtabilirler.

Uygulamadan fotoğraf çekerek veya galeriden mevcut fotoğrafını ekleyerek hangi geri dönüşüm malzemesi olduğunu kolayca tespit ettirebilirler.

```
void main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp();
 runApp(Barkod());}
```

void main() async: Uygulamanın ana giriş noktasıdır.

WidgetsFlutterBinding.ensureInitialized(): Flutter framework'ünü başlatır ve gerekli binding işlemlerini yapar.

await Firebase.initializeApp(): Firebase'in başlatılmasını sağlar.

runApp(Barkod()): Barkod widget'ını çalıştırır ve uygulamanın ana widget'ı olarak ayarlar.

```
class Barkod extends StatefulWidget {
 @override
 _BarkodState createState() => _BarkodState();}
```

Barkod: StatefulWidget'i genişleten bir sınıf. Durum yönetimi gerektiren bir widget.

_BarkodState createState() => _BarkodState(): Widget'ın durumunu oluşturur ve döner.

```
class _BarkodState extends State<Barkod> {
 String _scanBarcode = 'Unknown';
 XFile? photoPath;
 File? imageFile;
 String label = '';
 double confidence = 0.0;
@override
```

```
void initState() {
  super.initState();
  _tfLiteInit();}
```

- _BarkodState: Barkod widget'ının durum sınıfı.
- String _scanBarcode = 'Unknown': Tarama sonucunu tutmak için bir değişken.
- XFile? photoPath: Seçilen fotoğrafın yolunu tutmak için.
- File? imageFile: Dosya tipi fotoğrafı tutmak için.
- String label = '': Etiket bilgisini tutmak için
- double confidence = 0.0 : Doğruluk değerini tutmak için

```
  DatabaseReference conversionsRef = FirebaseDatabase.instance.ref('conversions');
  await conversionsRef.push().set({
    'userId': user.uid,
    'email': user.email,
    'timestamp': DateTime.now().toIso8601String(),
    'photoUrl': photoUrl,});
  print('Fotoğraf başarıyla yüklendi ve kaydedildi: $photoUrl');
 } catch (e) {
  print('Fotoğraf yükleme ve kaydetme sırasında hata: $e');}}
```

- uploadPhoto(XFile photo): Seçilen fotoğrafı Firebase Storage'a yükler ve ilgili bilgileri Firebase Realtime Database'e kaydeder.
- User? user = FirebaseAuth.instance.currentUser;: Mevcut kullanıcıyı alır.
- String filePath = 'user_photos/${user.email}/${DateTime.now().toIso8601String()}.jpg';: Fotoğrafın depolanacağı dosya yolunu oluşturur.
- Reference ref = FirebaseStorage.instance.ref().child(filePath);: Firebase Storage'da referans oluşturur.
- UploadTask uploadTask = ref.putFile(File(photo.path));: Fotoğrafı yükleme işlemini başlatır.
- TaskSnapshot taskSnapshot = await uploadTask.whenComplete(() => null);: Yükleme tamamlanana kadar bekler.
- String photoUrl = await taskSnapshot.ref.getDownloadURL();: Yüklenen fotoğrafın URL'sini alır.
- DatabaseReference databaseRef = FirebaseDatabase.instance.ref('users/${user.uid}/photos');: Kullanıcı fotoğrafları için bir referans oluşturur.
- await databaseRef.push().set({ ... }): Fotoğraf bilgilerini Firebase Realtime Database'e kaydeder.
- print('Fotoğraf başarıyla yüklendi ve kaydedildi: $photoUrl');: Yükleme ve kaydetme işleminin başarılı olduğunu konsola yazar.
- catch (e) { ... }: Herhangi bir hata durumunda hatayı yakalar ve konsola yazdırır.

```
photoPath != null

          ? Column(

         children: [

          Container(

            constraints: BoxConstraints(

              maxHeight: 300, // set a max height for the image),

            child: Image.file(File(photoPath!.path)),),

          Text(

            label,

            style: TextStyle(

              fontSize: 18,

              fontWeight: FontWeight.bold,

            ),),

          Text(

            "Doğruluk Oranı ${confidence.toStringAsFixed(0)}%",

            style: TextStyle(

              fontSize: 18,),),],)

          : Container(), ],), ),),),);}
```

- photoPath != null: photoPath null değilse, yani bir fotoğraf seçilmiş veya çekilmişse, alttaki Column widget'ı oluşturulur.
- Column(children: [...]): Bir sütun widget'ı oluşturur. Bu widget, içinde belirtilen widget'ları dikey olarak hizalar.
    - Container(constraints: BoxConstraints(maxHeight: 300), child: Image.file(File(photoPath!.path))):
        - Container: İçine aldığı widget'ın belirli kısıtlamalarla görüntülenmesini sağlar.
        - constraints: BoxConstraints(maxHeight: 300): Container'ın maksimum yüksekliğini 300 piksel ile sınırlar.
        - child: Image.file(File(photoPath!.path)): photoPath ile belirtilen dosyadan bir görüntü oluşturur ve bunu container'ın içine yerleştirir.
    - Text(label, style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)):
        - Text(label): label değişkeninin değerini metin olarak gösterir.
        - style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold): Metni 18 piksel boyutunda ve kalın olarak stilize eder.
    - Text("Doğruluk Oranı ${confidence.toStringAsFixed(0)}%", style: TextStyle(fontSize: 18)):

        - Text("Doğruluk Oranı ${confidence.toStringAsFixed(0)}%"): confidence değişkeninin yüzde formatında gösterilmesini sağlar. toStringAsFixed(0) ile confidence virgülden sonra 0 basamak gösterilir.
        - style: TextStyle(fontSize: 18): Metni 18 piksel boyutunda stilize eder.
- : Container(): Eğer photoPath null ise boş bir Container döndürülür.

Bu kod parçacığında, kullanıcı bir fotoğraf çektiğinde veya galeriden seçtiğinde, bu fotoğraf photoPath değişkenine atanır. Eğer bir fotoğraf mevcutsa, bu fotoğraf ekranda gösterilir ve tahmin edilen kategori ve doğruluk oranı görüntülenir.

Kodun tamamı, kullanıcının QR veya barkod taraması yapmasına, fotoğraf çekmesine veya galeriye erişmesine, bu fotoğrafı TensorFlow Lite modeli ile analiz etmesine ve sonuçları göstermesine olanak tanır. Firebase entegrasyonu ile fotoğraflar ve ilgili veriler saklanır. Analiz sonuçları ile ilgili bilgiler kullanıcıya gösterilir ve kullanıcı isterse ürünü sepete ekleyebilir.

SEÇ - DÖNÜŞTÜR SAYFASI:

Kullanıcılar, uygulamada tanımlı olan (plastic, paper,metal, glass, biological, battery) gibi geri dönüşüm malzemelerini doğrudan seçerek sepetlerine ekleyebilirler.

```
class HomePage extends StatelessWidget {

 List<Product> _products = [

  Product(

    id: 1,

    title: "Plastik Şişe",

    price: 30.0,

    imgUrl: "https://img.icons8.com/officel/160/plastic.png",

    qty: 1),

  Product(

    id: 2,

    title: "Kağıt",

    price: 10.0,

    imgUrl:
"https://img.icons8.com/external-icongeek26-linear-colour-icongeek26/128/external-papers-kindergarten-icongeek26-linear-colour-icongeek26.png",

    qty: 1),

  Product(

    id: 3,

    title: "Pil",
```

```
      price: 20.0,
      imgUrl:
"https://img.icons8.com/external-phatplus-lineal-color-phatplus/128/external-battery-ecology-system-phatplus-lineal-color-phatplus-2.png",
      qty: 1),
  Product(
      id: 4,
      title: "Teneke Kutu",
      price: 35.0,
      imgUrl: "https://img.icons8.com/doodle/192/tin-can--v1.png",
      qty: 1),
  Product(
      id: 5,
      title: "Cam",
      price: 30.0,
      imgUrl: "https://img.icons8.com/plasticine/200/fragile.png",
      qty: 1),
  Product(
      id: 6,
      title: "Biyolojik Atık",
      price: 25.0,
      imgUrl:
"https://img.icons8.com/external-fill-outline-pongsakorn-tan/128/external-plastic-ecology-and-pollution-fill-outline-pongsakorn-tan.png",
      qty: 1),];
```

Bu sınıf, _products adlı bir liste içerir. Bu liste, her biri Product nesnesi olan çeşitli ürünleri tanımlar.

```
@override
Widget build(BuildContext context) {
 return Scaffold(
   backgroundColor: Colors.indigo[50],
```

```
appBar: AppBar(

  backgroundColor: Colors.green,

  title: Text("Seç Dönüştür!"),

  actions: <Widget>[

    IconButton(

      icon: Icon(Icons.shopping_cart),

      onPressed: () {

        Navigator.push(context, MaterialPageRoute(builder: (context)=>CartPage()),); },)],),
```

build yöntemi, sayfanın kullanıcı arayüzünü oluşturur. Scaffold widget'ı ile sayfanın iskeleti tanımlanır.

```
body:
GridView.builder(
  padding: EdgeInsets.all(8.0),
  itemCount: _products.length,
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,
    mainAxisSpacing: 8,
    crossAxisSpacing: 8,
    childAspectRatio: 0.8
  ),
  itemBuilder: (context, index){
    return ScopedModelDescendant<CartModel>(
      builder: (context, child, model) {
        return Card(
          child: Column(
            children: <Widget>[
              Image.network(_products[index].imgUrl, height: 120, width: 120,),
              Text(_products[index].title, style: TextStyle(fontWeight: FontWeight.bold),),
              Text(_products[index].price.toString()+" puan"),
              ElevatedButton(
                child: Text(
                  "Ekle",
                  style: TextStyle(
                    color: Colors.green,
                  ),
                ),
                onPressed: () => model.addProduct(_products[index])
```

Ürünler, GridView ile iki sütunlu bir grid düzeninde görüntülenir. Her ürün bir kart içerisinde gösterilir.
AppBar: Üst kısımda bir uygulama çubuğu var. Bu çubuk, sayfanın başlığını ve bir alışveriş sepeti simgesini içerir. Sepet simgesine tıklandığında CartPage sayfasına yönlendirilir.
GridView: Ürünler iki sütunlu bir gridde görüntülenir. Her bir ürün bir kart içinde gösterilir ve ürün resmi, başlığı, puanı ve "Ekle" butonu içerir.
ScopedModel: ScopedModel kullanılarak ürünlerin sepete eklenmesi sağlanır. model.addProduct yöntemi ile ürünler sepete eklenir.

GERİ DÖNÜŞÜM SEPETİ SAYFASI:

Kullanıcılar, geri dönüştürecekleri malzemeleri sepetlerine ekleyerek puan kazanırlar. Bu puanlar geri dönüştürülen malzemenin cinsine göre değişmektedir. Örneğin pil 20 puan iken kağıt 10 puandır. Kazandıkları puanlar ve ödülleri bu sayfada gösterilir.

CartPage adlı bir StatefulWidget sınıfı tanımlanır ve _CartPageState adlı bir durumu (state) vardır. Diğer sayfalarda olduğu gibi _CartPageState sınıfının build yöntemi, sayfanın kullanıcı arayüzünü oluşturur.

```
  body: ScopedModel.of<CartModel>(context, rebuildOnChange: true)

    .cart

    .length == 0

    ? Center(

   child: Text("Dönüştürülecek ürün yok!"),)
```

Sayfa gövdesi, alışveriş sepetindeki ürünleri gösterir. Eğer sepet boşsa, merkezde bir mesaj gösterilir.

```
  : Container(
    padding: EdgeInsets.all(8.0),
    child: Column(children: <Widget>[
      Expanded(
        child: ListView.builder(
          itemCount: ScopedModel.of<CartModel>(context, rebuildOnChange: true).total,
          itemBuilder: (context, index) {
            return ScopedModelDescendant<CartModel>(
              builder: (context, child, model) {
                return ListTile(
                  title: Text(model.cart[index].title),
                  subtitle: Text(model.cart[index].qty.toString() +
                      " x " +
                      model.cart[index].price.toString() +
                      " = " +
                      (model.cart[index].qty * model.cart[index].price).toString()),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
                        icon: Icon(Icons.add),
                        onPressed: () {
                          model.updateProduct(model.cart[index], model.cart[index].qty + 1);
                        },
                      ),
                      IconButton(
                        icon: Icon(Icons.remove),
                        onPressed: () {
                          model.updateProduct(model.cart[index], model.cart[index].qty - 1); }, ),
```

Sepetteki ürünler, ListView.builder ile listelenir ve her bir ürün, başlık, miktar, birim fiyat ve toplam fiyat bilgilerini gösterir.

```
  : Container(
    padding: EdgeInsets.all(8.0),
```

```
    child: Column(children: <Widget>[
      Expanded(
        child: ListView.builder(
          itemCount: ScopedModel.of<CartModel>(context, rebuildOnChange: true).total,
          itemBuilder: (context, index) {
            return ScopedModelDescendant<CartModel>(
              builder: (context, child, model) {
                return ListTile(
                  title: Text(model.cart[index].title),
                  subtitle: Text(model.cart[index].qty.toString() +
                      " x " +
                      model.cart[index].price.toString() +
                      " = " +
                      (model.cart[index].qty * model.cart[index].price).toString()),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
                        icon: Icon(Icons.add),
                        onPressed: () {
                          model.updateProduct(model.cart[index], model.cart[index].qty + 1);
                        },
                      ),
                      IconButton(
                        icon: Icon(Icons.remove),
                        onPressed: () {
                          model.updateProduct(model.cart[index], model.cart[index].qty - 1);
                        },
```

Sepetin toplam puanı ve bir buton bulunur. Butona tıklandığında, kullanıcı sepet sayfasına yönlendirilir.

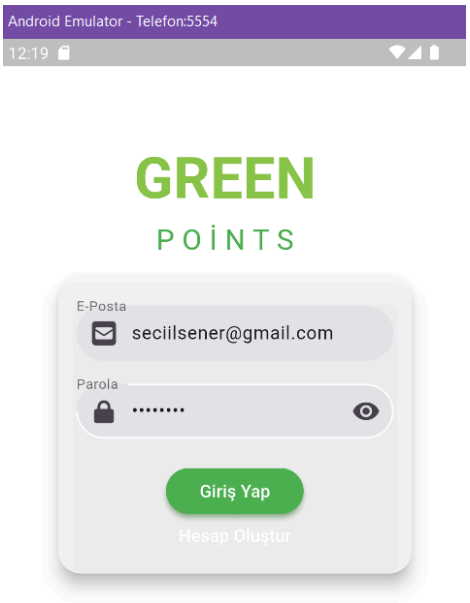
AppBar: Üst kısımda bir uygulama çubuğu var. Bu çubuk, sayfanın başlığını ve "Temizle" butonunu içerir.

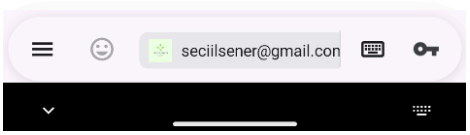
Ürün Listesi: Sepetteki ürünler bir liste halinde gösterilir. Her bir ürünün başlığı, miktarı, fiyatı ve toplam fiyatı listelenir.

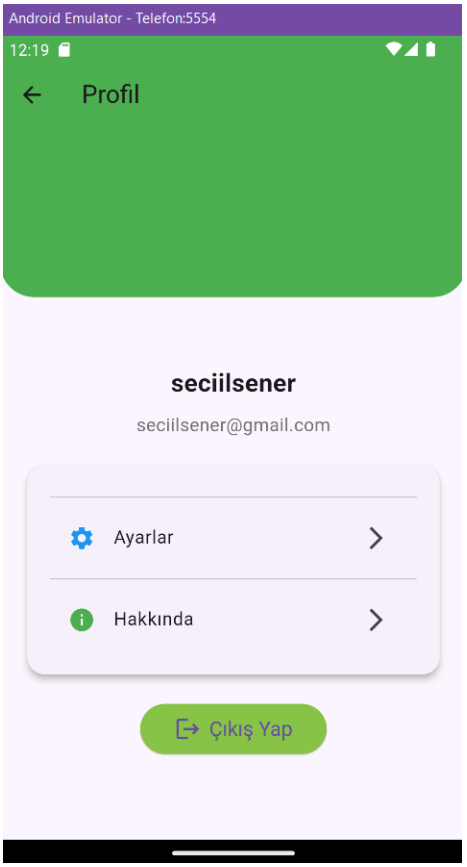
Toplam Değer: Sepetin toplam puanı gösterilir.

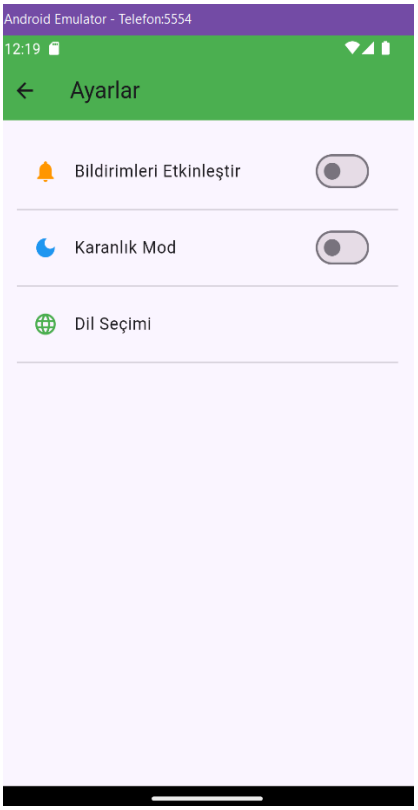
Sepete Git Butonu: Butona tıklandığında kullanıcı sepet sayfasına yönlendirilir.

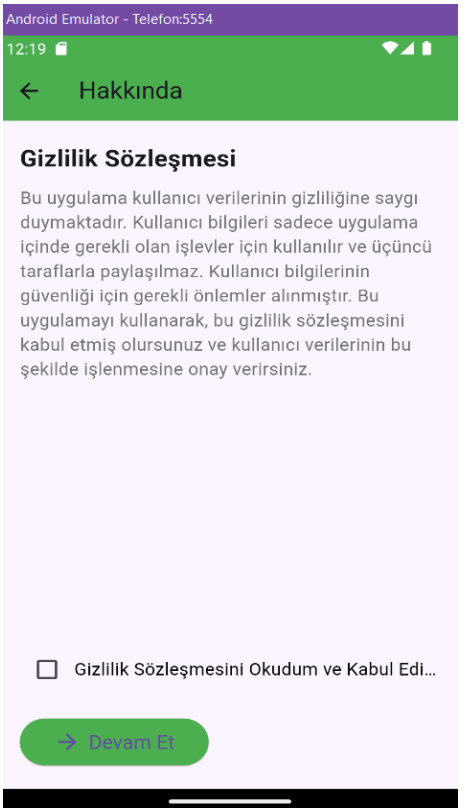
Proje Çıktıları

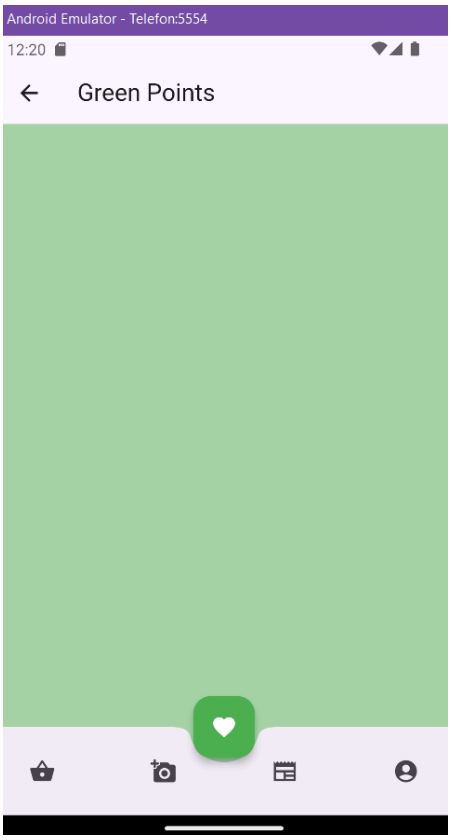








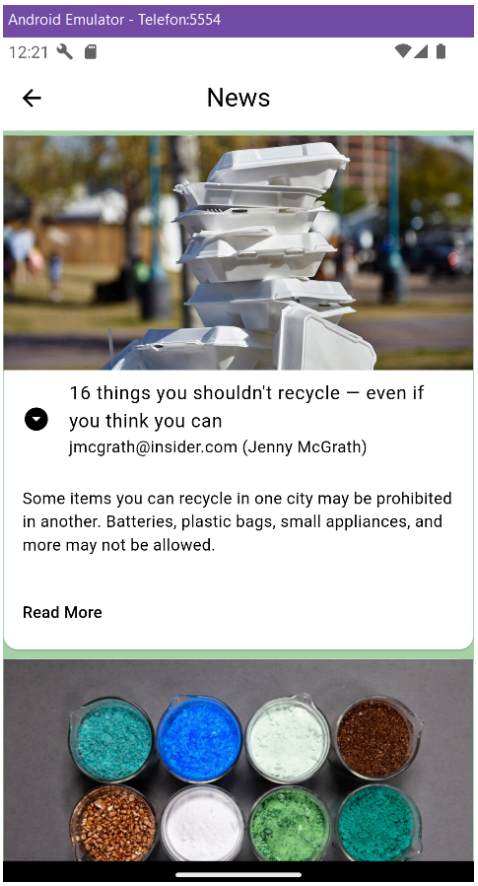


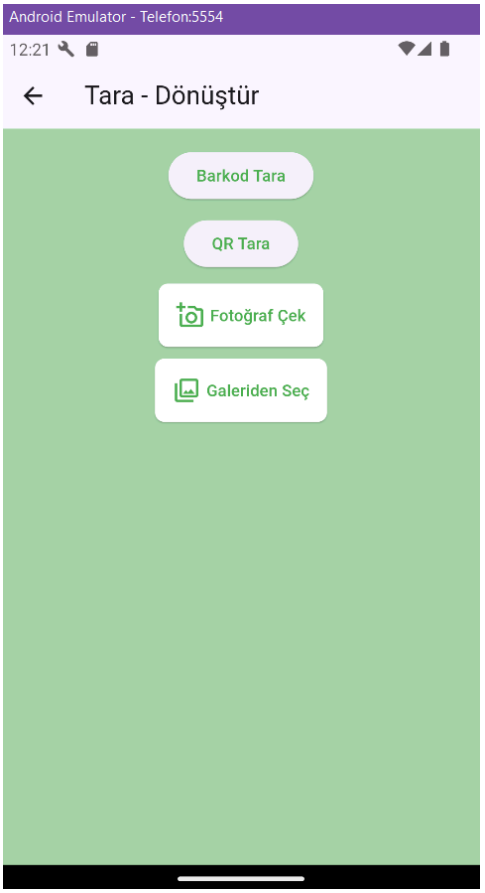

Android Emulator - Telefon:5554
12:20
Green Points

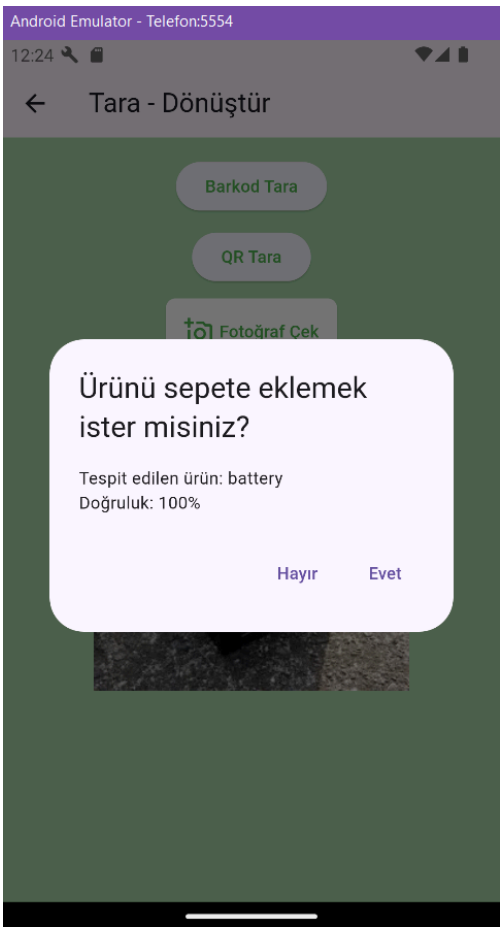

Android Emulator - Telefon:5554
12:20
Seç Dönüştür!
Plastik Şişe
30.0 puan
Ekle
Kağıt
10.0 puan
Ekle
Pil
20.0 puan
Ekle
Teneke Kutu
35.0 puan
Ekle


Android Emulator - Telefon:5554
12:20
Hazinem
Temizle
Plastik Şişe
1 x 30.0 = 30.0
Kağıt
1 x 10.0 = 10.0
Pil
2 x 20.0 = 40.0
Toplam: 80.0 puan
Tıkla Sepetine Git!


Android Emulator - Telefon:5554
12:20
Sepet
Toplam: 80.0 puan
Topladığınız puanlar sayesinde sokak hayvanlarına yiyecek bağışlandı!


Android Emulator - Telefon:5554
12:21
News
16 things you shouldn't recycle — even if you think you can
jmcgrath@insider.com (Jenny McGrath)
Some items you can recycle in one city may be prohibited in another. Batteries, plastic bags, small appliances, and more may not be allowed.
Read More


Android Emulator - Telefon:5554
12:21
Tara - Dönüştür
Barkod Tara
QR Tara
Fotoğraf Çek
Galeriden Seç


Android Emulator - Telefon:5554
12:24
Tara - Dönüştür
Barkod Tara
QR Tara
Fotoğraf Çek
Ürünü sepete eklemek ister misiniz?
Tespit edilen ürün: battery
Doğruluk: 100%
Hayır
Evet


Android Emulator - Telefon:5554
12:24
Hazinem
Temizle
Pil
1 x 20.0 = 20.0
Toplam: 20.0 puan
Tıkla Sepetine Git!

Proje Aşamaları

Proje Grubu Görev Dağılımı:

Cem Bozkurt: Proje Yöneticisi

Seçil Şener : Developer

Eda Menekşeyurt : Developer

Kübra Akpunar: Developer

Şevval Kapçak: Developer

Ürün İsmi: Green Points

Ürün Açıklaması:

Ortaya çıkarılacak ürünle birlikte, toplumu geri dönüşüme teşvik etmek, hem kullanıcılar hem de üretici açısından çevre temizliği ve geri dönüşüm mantığı gözetilerek karşılıklı kar elde edilebilecek, kullanışlı yaygın ve kolay bir sanal zemin oluşturulması hedeflenmiştir.

Ürün Özellikleri:

Ürün, kullanıcının mail adresi ve şifresini girerek uygulamaya giriş yapması ve geri dönüştüreceği atığı sisteme girmesi ile başlamaktadır. Daha sonra kullanıcının geri dönüştüreceği ürüne göre puan toplaması ve bu puanların bir haznede tutulması hedeflenmiştir. Ardından yeterli puana ulaşıldığında kullanıcı onayıyla birlikte bu puanların kullanıcı adına fidan dikiminde ya da sokak hayvanlarına mama bağışı yapılmasında kullanılması hedeflenmiştir.

Hedef Kitle:

Hedef kitle; geri dönüşüm bilincinde olan ve mobil olarak ürüne ulaşabilecek herkes olarak belirlenmiştir. Her sprint 2 haftayı kapsamaktadır.

SPRİNT 1

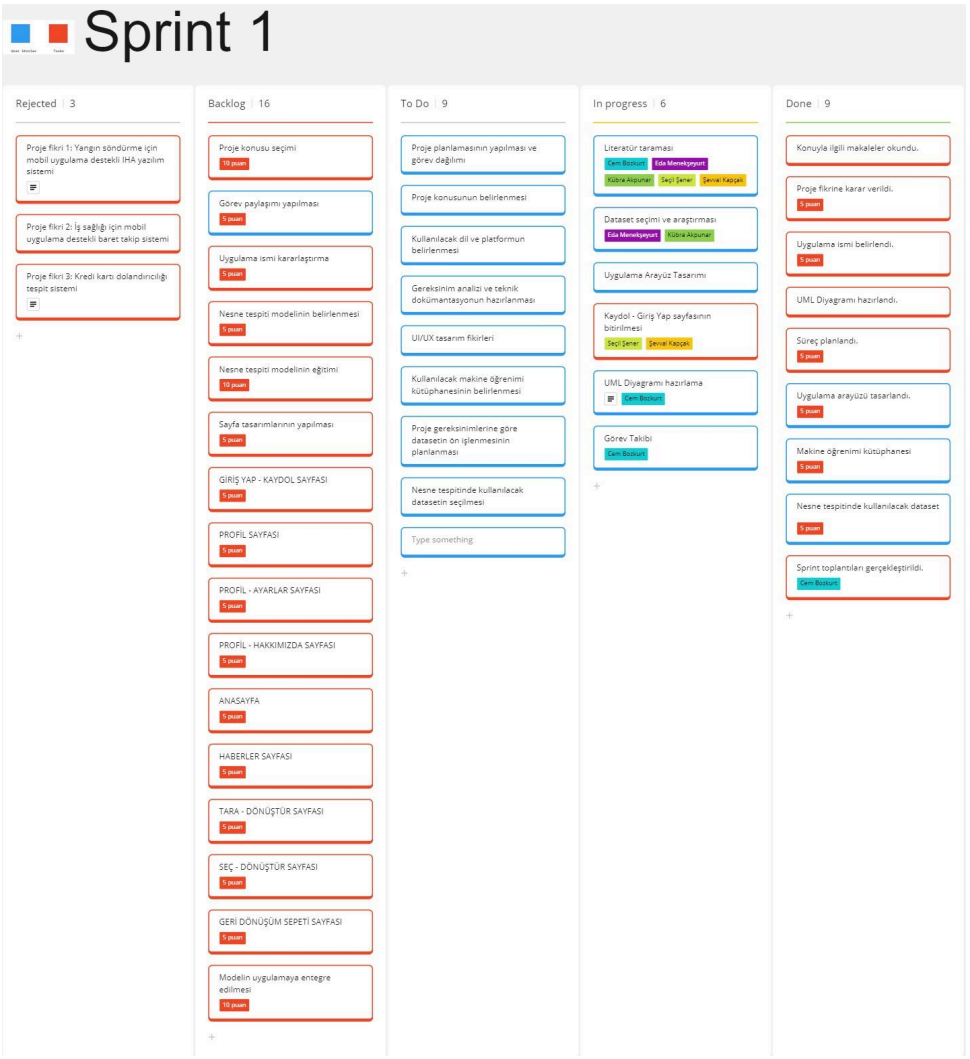
Sprint içinde tamamlanması gereken puan: 100 Puan

Puan tamamlama mantığı: 4 Sprint için toplamda 400 puanlık Backlog olması kararlaştırılmıştır. 1. Sprint için 100 Puan olarak hesaplanmıştır.

Backlog düzeni ve Story seçimleri: Backlog, planlanan ilerleme düzenine göre düzenlenmiştir. Story zorluklarına göre 5 ile 10 puan arasında puanlandırmalar yapılmıştır. Bu puanlamalar sonucunda sprint'lerin toplam puanı 400 olarak belirlenmiştir. Story'ler yapılacak işlere bölünmüştür. Miro Board'da gözüken mavi kartlar yapılacak işleri gösterirken, kırmızı kartlar story'leri temsil etmektedir.

Daily Scrum: Daily Scrum toplantıları, ihtiyaca uygun olarak hem Google Meetings hem de WhatsApp üzerinden gerçekleşmektedir.

Sprint Board Update: Sprint Board Ekran Görüntüsü



Sprint Review:

Sprint boyunca proje konusu kararlaştırılmış, uygulamanın temel hatları oluşturulmuştur. Toplantı sıklığını artırmaya karar verilerek görev dağılımları gerçekleştirilmiştir. Uygulamanın sayfalarının tasarımı yapılmış ve giriş yap - kaydol sayfası tamamlanmıştır. Developerlar tarafından Sprint puan hesapları yapılmıştır.

Sprint Review katılımcıları:

Eda Menekşeyurt, Kübra Akpunar, Seçil Şener, Cem Bozkurt, Şevval Kapçak.

Sprint Retrospective:

Toplantıların sıklıkla gerçekleştirilmesi kararlaştırılmıştır.

İkinci Sprint için görev listesi ve görev dağılımları erkenden belirlenmelidir.

SPRİNT 2

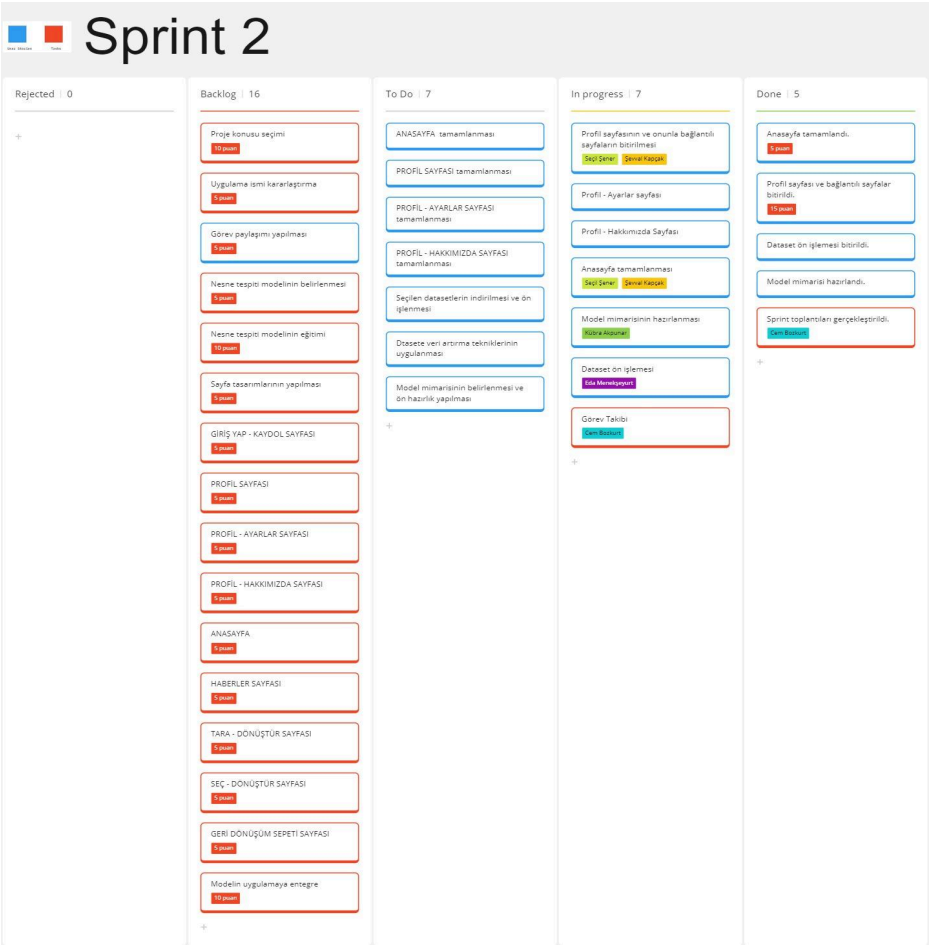
Sprint içinde tamamlanması gereken puan: 100 Puan

Puan tamamlama mantığı: 4 Sprint için toplamda 400 puanlık Backlog olması kararlaştırılmıştır. 2. Sprint için 100 Puan olarak hesaplanmıştır.

Backlog düzeni ve Story seçimleri: Backlog, planlanan ilerleme düzenine göre düzenlenmiştir. Story zorluklarına göre 5 ile 10 puan arasında puanlandırmalar yapılmıştır. Bu puanlamalar sonucunda sprint'lerin toplam puanı 400 olarak belirlenmiştir. Story'ler yapılacak işlere bölünmüştür. Miro Board'da gözüken mavi kartlar yapılacak işleri gösterirken, kırmızı kartlar story'leri temsil etmektedir.

Daily Scrum: Daily Scrum toplantıları, ihtiyaca uygun olarak hem Google Meetings hem de WhatsApp üzerinden gerçekleşmektedir.

Sprint Board Update: Sprint Board Ekran Görüntüsü



Sprint Review: Sprint boyunca uygulamanın profil sayfasına ve anasayfaya ağırlık verilmiştir. Görev dağılımı ağırlığı 3. Sprinte verilerek 4. Sprintin hata iyileştirmeleri ile tamamlanması kararlaştırılmıştır.

Sprint Review katılımcıları:

Eda Menekşeyurt, Kübra Akpunar, Seçil Şener, Cem Bozkurt, Şevval Kapçak.

Sprint Retrospective:

4. Sprint için görev listesi azaltılmalı ve bu görevler 3. Sprintte tamamlanmalıdır.

Son Sprint, uygulama denemeleri ve iyileştirmeleri gerçekleştirilmelidir.

SPRİNT 3:

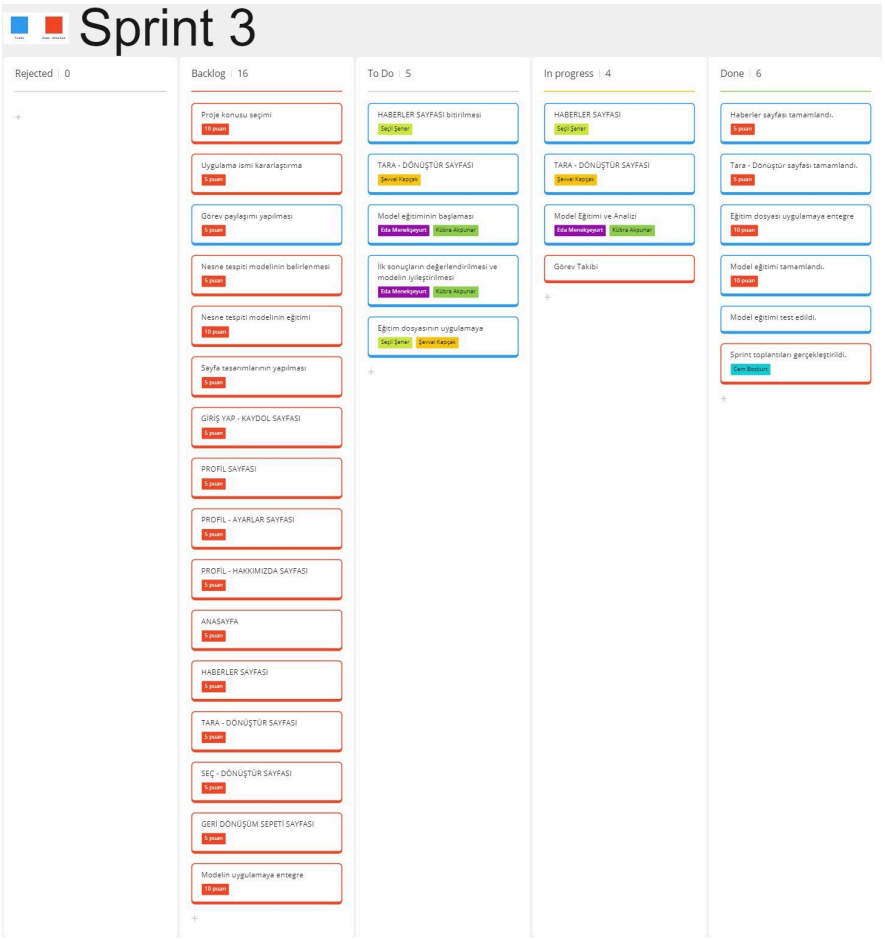
Sprint içinde tamamlanması gereken puan: 100 Puan

Puan tamamlama mantığı: 4 Sprint için toplamda 400 puanlık Backlog olması kararlaştırılmıştır. 3. Sprint için 100 Puan olarak hesaplanmıştır.

Backlog düzeni ve Story seçimleri: Backlog, planlanan ilerleme düzenine göre düzenlenmiştir. Story zorluklarına göre 5 ile 10 puan arasında puanlandırmalar yapılmıştır. Bu puanlamalar sonucunda sprint'lerin toplam puanı 400 olarak belirlenmiştir. Story'ler yapılacak işlere bölünmüştür. Miro Board'da gözüken mavi kartlar yapılacak işleri gösterirken, kırmızı kartlar story'leri temsil etmektedir.

Daily Scrum: Daily Scrum toplantıları, ihtiyaca uygun olarak hem Google Meetings hem de WhatsApp üzerinden gerçekleşmektedir.

Sprint Board Update: Sprint Board Ekran Görüntüsü



Sprint Review:

Sprint boyunca uygulamanın son sayfaları olan tara - dönüştür sayfası ve haberler sayfasının kodlamasına ardından da model eğitimine ağırlık verilmiştir. Görev dağılımı yapılmış ve uygulamanın yüksek yoğunluğu tamamlanmıştır.

Sprint Review katılımcıları:

Eda Menekşeyurt, Kübra Akpunar, Seçil Şener, Cem Bozkurt, Şevval Kapçak.

Sprint Retrospective:

Uygulamanın eksik kısımlarının kapsamın belirlenmesi gerekmektedir. Son Sprintte, uygulama denemeleri ve iyileştirmeleri gerçekleştirilmelidir.

SPRİNT 4:

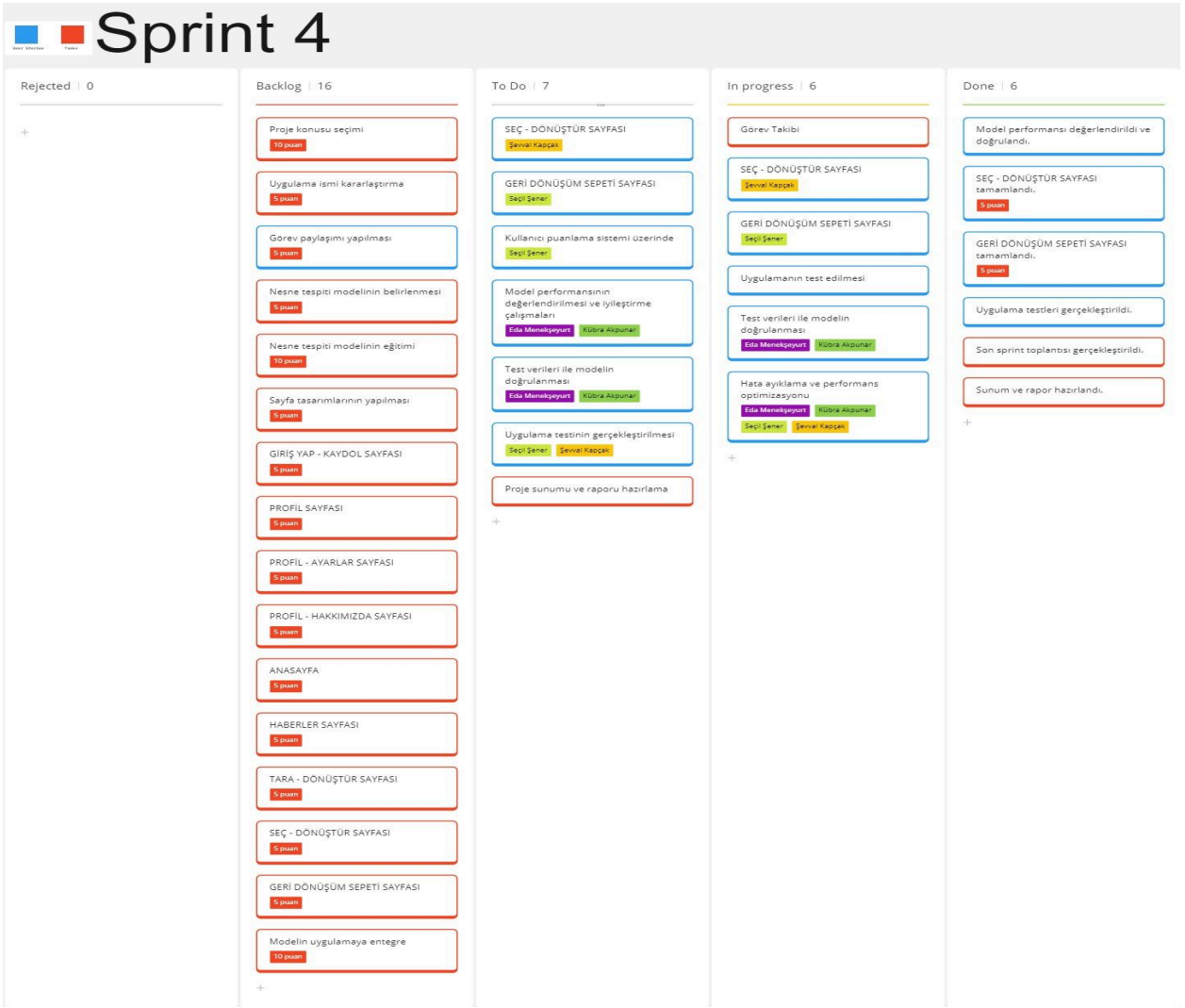
Sprint içinde tamamlanması gereken puan: 100 Puan

Puan tamamlama mantığı: 4 Sprint için toplamda 400 puanlık Backlog olması kararlaştırılmıştır. 4. Sprint için 100 Puan olarak hesaplanmıştır.

Backlog düzeni ve Story seçimleri: Backlog, planlanan ilerleme düzenine göre düzenlenmiştir. Story zorluklarına göre 5 ile 10 puan arasında puanlandırmalar yapılmıştır. Bu puanlamalar sonucunda sprint'lerin toplam puanı 400 olarak belirlenmiştir. Story'ler yapılacak işlere bölünmüştür. Miro Board'da gözüken mavi kartlar yapılacak işleri gösterirken, kırmızı kartlar story'leri temsil etmektedir.

Daily Scrum: Daily Scrum toplantıları, ihtiyaca uygun olarak hem Google Meetings hem de WhatsApp üzerinden gerçekleşmektedir.

Sprint Board Update: Sprint Board Ekran Görüntüsü



Sprint Retrospective: Grup içerisindeki kişiler görevlerini tamamlamıştır. Tahmin puanları dengeli bir şekilde belirlenmiştir. Backlog'daki işler tamamlanmıştır. Son sprintte çalışmalar hızlandırıldı ve iletişim sıklığı sağlanarak sorunlar konusunda hızlı çözümler ve aksiyonlar alınmaya odaklanıldı. Her aşama sonunda küçük toplantılarla süreç değerlendirmesi yapıldı.

Sonuç

Bu projede görüntü işleme ve nesne sınıflandırılması ile geri dönüşüm uygulaması yapılmıştır.Kullanıcıdan alınan fotoğraflar görüntü işleme teknikleriyle sınıflandırılmış ve kullanıcı dostu bir mobil uygulama ile kullanıcının çeşitli kategoriler için farklı puanlar kazanabileceği bir sistem inşaa edilmiştir. Bu ödül puanlama sistemi ile kullanıcılara çeşitli ödül fırsatları sunularak insanları geri dönüşüme teşvik edilmesi sağlanmıştır.Geri dönüşüm, yeni ürünlerin üretilmesi için gerekli olan doğal kaynakların kullanımını azaltır. Özellikle plastik, cam, kağıt ve metal gibi geri dönüştürülebilir malzemelerin geri dönüşümü, ormanların kesilmesini, madenlerin işlenmesini ve doğal yaşam alanlarının tahrip edilmesini azaltır. Ayrıca geri dönüşüm ile atık miktarı azaltılır bu sayede çevre kirliliği ve doğal kaynakların tükenmesi gibi sorunların önüne geçilir aynı zamanda enerji tasarrufu sağlanır. Toplumlar, atıklarını düzgün bir şekilde yönetmek ve çevreyi korumak için geri dönüşümün önemini giderek daha fazla anlamaya başlamıştır. Ülkemizde son yıllarda geri dönüşüm oranı artış gösterse de henüz istenilen seviyede değildir. Biz de bu farkındalığı toplumumuza katmak ve geri dönüşümü ödül sistemiyle motive edici hale dönüştürmek için bu projeyi geliştirdik. Projemizin geri dönüşüm konusunda farkındalık yaratmasını amaçladık.