

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI

GÖRÜNTÜ İŞLEME İLE NESNE TANIMA YAPAN
AİRBNB UYGULAMASI

B191210055 – Gizem Yiğit
B191210067 – Zelal İnanc
B191210010 – Oğuzhan Türkoğlu

Bölüm : **BİLGİSAYAR MÜHENDİSLİĞİ**
Danışman : **Öğr. Gör. Dr. Can Yüzkollar**

2022-2023 Güz Dönemi

ÖNSÖZ

Görüntü işleme, İngilizce adıyla Image Processing dijital ortamlar üzerinden bazı bilgisayar algoritmaları ve görsel teknikler kullanılarak kaydedilmiş görüntüyü amaca uygun hale getirme yöntemidir. Her görüntü kullanılabilirlik açısından elverişli olmayabilir. Görüntü işleme teknolojisi ile artık her görüntü amaca uygun kullanım için özelleştirilebilir. Kaydedilen görüntü bilgisayar algoritmaları tarafından bir fonksiyona dönüştürülür. Görüntü iki boyut olarak algılanır; her renk, her nokta fonksiyonun parçaları olarak algılanır ve algoritmalar bu fonksiyon üzerinde işlem yapabilir.

İÇİNDEKİLER

ÖNSÖZ.....	iii
İÇİNDEKİLER.....	iv
SİMGELER VE KISALTMALAR LİSTESİ.....	vi
ŞEKİLLER LİSTESİ.....	vii
ÖZET.....	viii

BÖLÜM 1.

GÖRÜNTÜ İŞLEME UYGULAMASI

- 1.1. Görüntü İşleme
- 1.2. Görüntü İşleme Basamakları

BÖLÜM 2.

API UYGULAMASI

- 2.1. Neden Flask Kullandık
- 2.2. Api Dosyasına Eklenen Dataset Dosyaları
- 2.3. Yolo_Detection_Images.Py Scripti Üzerinde Yapılan Değişiklikler
- 2.4. App.Py Dosyasında Yapılan Değişiklikler
- 2.5. Index.Js Dosyasında Yapılan Değişiklikler

BÖLÜM 3.

ASP.NET CORE WEB UYGULAMASI

- 3.1. .Net Core Nedir?
- 3.2. Projede Katmanlı Mimarı Kullanılması
- 3.3. Veri Tabanı Uygulaması
- 3.4. Admin İşlemlerinin Yapılması
- 3.5. İndex Sayfası
- 3.6. Ev Detay Sayfası

BÖLÜM 4.

SONUÇLAR VE ÖNERİLER

SİMGELER VE KISALTMALAR LİSTESİ

ASP	: Active Server Pages
C#	: C Sharp Language
.NET	: Dot Net Framework
HTML	: Hyper-Text Mark-up Language
CSS	: Cascading Style Sheets
MSSQL	: Microsoft Structured Query Language
PYTHON	: Python Language
YOLO	: You Only Look Once
API	: Application Programming Interface

ŞEKİLLER LİSTESİ

Şekil 1.1.	Darknet klonlama kodu
Şekil 1.2.	Darknet Opencv kodu
Şekil 1.3.	Show import kodu
Şekil 1.4.	Upload fonksiyonu kodu
Şekil 1.5.	Download fonksiyonu kodu
Şekil 1.6.	Drive fotoğraf yükleme kodu
Şekil 1.7.	Train.py dosyası
Şekil 1.8.	Test.py dosyası
Şekil 1.9.	Model eğitimi
Şekil 1.10.	İmshow komutu
Şekil 1.11.	Test için fotoğraf ekleme
Şekil 1.12.	Etiketlenmiş fotoğraf
Şekil 1.13.	Etiketlenmiş fotoğraf
Şekil 1.14.	Avarage Loss Grafiği
Şekil 1.15.	Tanımlanmış Nesnelerin Tablosu
Şekil 1.16.	Oran Tablosu
Şekil 2.1.	Yolo_detection_images.py dosyası
Şekil 2.2.	Get_text fonksiyonu
Şekil 2.3.	RunModel fonksiyonu
Şekil 2.4.	Def home metodu
Şekil 2.5.	Mask_image metodu
Şekil 2.6.	İndex.js dosyası
Şekil 2.7.	Tanımlanmış nesne fotoğrafı
Şekil 3.1.	Katmanlı mimarı yapısı
Şekil 3.2.	Context sınıfı
Şekil 3.3.	Veri tabanı diyagramı

- Şekil 3.4. IGenereicDal sınıfı
- Şekil 3.5. Kayıtlı evler sayfası
- Şekil 3.6. Kategori ekleme sayfası
- Şekil 3.7. Yeni ev kaydı sayfası
- Şekil 3.8. Tanınmış nesnelerin fotoğrafı
- Şekil 3.9. Anasayfa
- Şekil 3.10. Ev detay sayfası

ÖZET

Anahtar Kelimeler: Görüntü işleme, Nesne Tespiti, Web Uygulama

Bu proje bir Airbnb sitesi uygulamasıdır. Projede görüntü işleme kullanılarak, eve ait yüklenen oda fotoğrafları üzerinden o odada bulunan belirli eşyaların tespit edilmesi ve evin özellikleri olarak işaretlenmesi amaçlanmıştır. Kullanıcı siteye yüklenen ev ilanlarını arayabilir, evin özelliklerini ve fotoğraflarını görüntüleyebilir, dilerse rezervasyon işlemini gerçekleştirebilir.

BÖLÜM 1. GİRİŞ

Görüntü işleme bir görüntünün içerisindeki önemli bilgilerin okunması, bu resimlerden anlam çıkartılması ve işlenmesi için kullanılan bir yöntemdir. Görüntüdeki bir nesne hakkında insan görmesi gibi benzer şekilde nitel bilgilerin edinilmesi ve kullanılması görüntü işlemenin temel amaçlarındandır. Kullanılacak olan görüntülerde bulunan nesnelerin tespiti, tanımlanması gibi ihtiyaçları karşılayacak çok fazla yöntem geliştirilmiştir.

Nesne Tespiti gibi uygulamalar son yıllarda oldukça popüler derin öğrenme alanlarından biridir.

1.1. Görüntü İşleme

Görüntü işleme teknikleri mevcut resimlerin analizi ile çeşitli bilgiler edinmeye yarayan sistemlerdir.

Görüntü işleme sistemlerinin yoğunlaştığı nokta ise milyonlarca görsel kimliği arasındaki benzerlikleri veya farklılıkları bularak yeni bilgiler elde etmektir. Güçlü kişisel bilgisayarların, büyük boyutlu bellek cihazlarının ve grafik yazılımlarının kolay erişilebilirliği görüntü işleme sistemlerinin her geçen gün daha popüler olmasını sağlamaktadır.

1.2. Görüntü İşleme Basamakları

1.2.1. Labellmg İle Veri Etiketleri ve Dataset Oluşturma

Kendi veri setimizi oluşturmak için Labellmg adlı arayüzünü kullandık.

Bu program, görüntülerin üzerindeki istediğimiz bir veya birden çok alanın istediğimiz şekilde etiketlenebilmesini sağlamakta. Bu etiketleme sonucunda bize her görüntüye ait bir “xml” dosyası veriyor. Bu dosya sayesinde görüntülerimizi sınıflandırabiliyoruz.

Böylelikle elde ettiğimiz dosyalar dataset eğitimi için hazır hale geliyor.

1.2.2. Dataset Eğitilmesi

1.2.2.1 Yolov3 Kullanımı

“You Only Look Once” olarak da bilinen Yolo en basit tanımıyla bir Nesne Tanıma modelidir. Kendi Datasetimizin eğitimi için de bu modeli kullandık.

Yolo’da nesne tanıma işlemi aynı anda gerçekleşen 2 farklı aşamadan oluşuyor.

Bunlardan birincisi Object Detection yani nesnenin konumunu bulmaktır. İkinci ise classification yani konumunu bulduğumuz nesnenin hangi sınıfa ait olduğunu bulmaktır.

1.2.2.2 Darknet Kullanımı

Modelimizi eğitirken bazı kütüphanelere bazı özel dosyalara ve Python scriptlerine ihtiyaç duyduğumuzdan ötürü, Darknet reposunu Colab uygulamasına klonladık.

(1.1)

```
!git clone https://github.com/AlexeyAB/darknet
```

1.2.2.3 Colab Kullanımı

Colab, yüksek hızlı ekran kartı olan Nvidia Tesla K80’ i ücretsiz olarak kullanımımıza sunuyor.

Darknet reposuna ait Opencv kütüphanesini kullandık.

(1.2)

```
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
```

Darknet kütüphanelerinin Google Colab dosyaları içerisinde kurulmasıyla ilk adımı tamamlıyoruz.

1.2.3. Yardımcı Fonksiyonların Kullanılması

Show fonksiyonu ile fotoğrafların tanımlanmış bir şekilde görüntülenmesini sağlıyoruz.

Matplotlib, figürlerin görselleştirilmesini sağlayan 2 Boyutlu bir çizim kütüphanesidir.

Fotoğraf üzerinde etiketlenecek kareyi gösterebilmek için cv2 kütüphanesini fonksiyonumuzun içine import ediyoruz.

(1.3)

```
def imshow(path):
    import cv2
    import matplotlib.pyplot as plt
    %matplotlib inline
```

Fotoğrafları projeye yükleyebilmek için Upload() ve fotoğrafların yeniden istendiği durumda, fotoğrafları indirmek için Download() fonksiyonlarını kullandık.

(1.4)

```
def upload():
    from google.colab import files
    uploaded = files.upload()
    for name, data in uploaded.items():
        with open(name, 'wb') as f:
            f.write(data)
        print('saved file', name)
```

(1.5)

```
def download(path):
    from google.colab import files
    files.download(path)
```

Daha önceden etiketlenmiş olan fotoğraflarımızı colabde dağıtırmak üzere drive'a drive.mount komutu ile Drive'a fotoğrafları yükledik.

(1.6)

```
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')
```

Ardından datasetimizin olduğu Drive dosyasını Darknet dosya dizinine ekledik.

1.2.4. Eğitim İçin Gerekli Dosyaların Hazırlanması

1.2.4.1. Config Dosyasının Düzenlenmesi

Darknet dizinine ait yolov3_custom.cfg dosyasını kendi eğiteceğimiz sınıflara göre düzenlemesini yaptık.

Dosyada bulunan batch ve subdivision değerlerini 64 ve 16 olarak güncelledik. Max_batches değerini 2000*eğitilen sınıf sayısı ile çarparak güncelledik. Steps değerine (%80 Max_batches, %90 Max_batches) olacak şekilde güncelledik. [yolo] başlığı altındaki sınıf değerlerini eğitilen sınıf sayısı ile değiştirdik. Filters değişkenlerini de (eğitim yapılacak sınıf sayısı + 5)* 3 değerine eşitledik.

1.2.4.2. Obj.names ve Obj.data Dosyalarının Düzenlenmesi

Yolov3 isimli klasörünün içine obj.names isimli bir dosya oluşturduk ve eğitim yapacağımız sınıf isimlerini yazdık.

Aynı klasörün içine obj.data isimli bir dosya oluşturduk ve eğitim yapacağımız nesne sayısını, eğitim yaparken kullanılacak train.txt, text.txt ve obj.names isimli dosyaların adreslerini ve eğitim sonucu bulduğumuz ağırlıkları kaydedeceğimiz dizini yazdık.

1.2.4.3. Train ve Test Dosyaları

Daha önceden oluşturduğumuz drive klasörüne generate_train.py isimli Python scriptini yazdık.

Burada yaptığımız işlem obj dosyasını data dosyasıyla birleştirip, img uzantılı dosyaları oluşturduğumuz yeni yol ile birleştirerek image_files arrayine ekledik.

Sonrasında train.txt dosyasını açıp bu array içindeki değerleri txt dosyasına yazdık ve dosyayı kapattık.

(1.7)

```
import os
image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
os.chdir("..")
with open("train.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

Sonrasında generate_test.py scriptini oluşturduk ve aynı işlemleri yaparak array içindeki değerleri test.txt dosyasına yazdırdık.

Oluşturduğumuz dosyaları Colab dosyasında çalıştırıyoruz.

(1.8)

```
import os

image_files = []
os.chdir(os.path.join("data", "test"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/test/" + filename)
os.chdir("..")
with open("test.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

1.2.5. Dataset Eğitiminin Başlaması

Eğitimimizin süresi veri setinizdeki fotoğraf sayısı, fotoğrafların kalitesi, eğitim yaptığınız nesne sayısı gibi faktörlere göre değişebilir. Modelimizin doğruluğu için loss değeri azalmayı durdurana kadar çalıştırıp veri setimize göre mümkün olan en doğru modeli eğittik.

(1.9)

```
!./darknet detector train data/obj.data cfg/yolov3-obj.cfg yolov3.conv.137 -dont_show -map
```

Modelimizi eğittikten sonra eğitim sırasında loss değerimizin nasıl değiştiğini test edebilmek için show metodu ile grafik oluşturduk.

(1.10)

```
imshow('chart.png')
```

1.2.6. Modelin Test Edilmesi

Eğitim yaparken batch ve subdivison değerlerini sınıf sayılarımıza göre düzenlerken amacımız modelimizin deep learning katmanlarına aynı anda birçok fotoğraf gönderiyor olmamızdı.

Modeli test etmek için göndereceğimiz fotoğraf bir tane olduğu için batch ve subdivision değerlerini 1'e eşitledik.

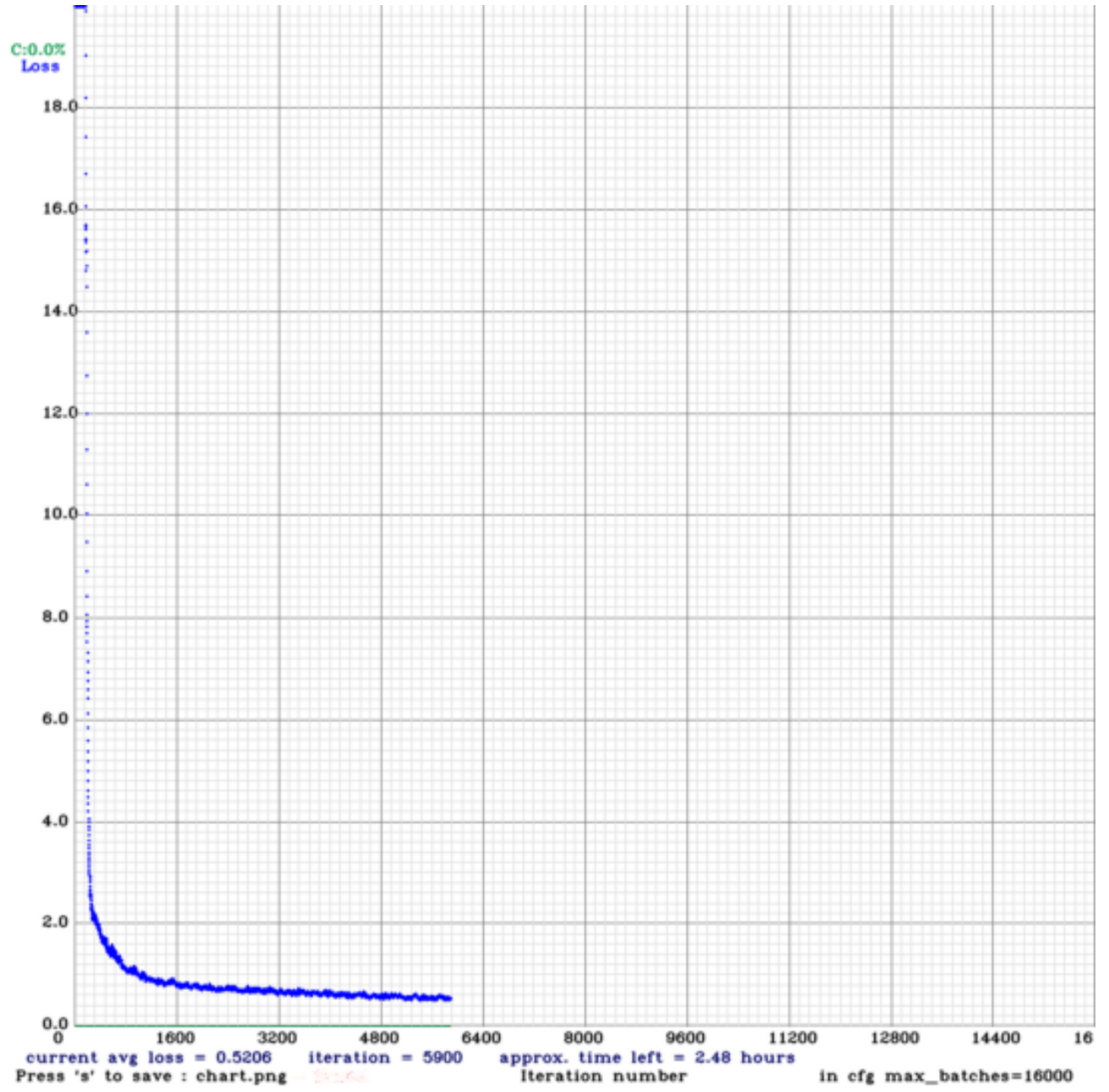
Test için fotoğraf ekledik ve nesne tanıma yapıp yapmadığını işlem sonunda kontrol ettik.

(1.11)

```
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov4-obj.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov4-obj.cfg
%cd ..
!./darknet detector test data/obj.data cfg/yolov3-obj.cfg /mydrive/yolov3/backup/yolov3-obj_last.weights /mydrive/images/mutfak.jpg -thresh 0.3
imshow('predictions.jpg')
```

1.2.7. Etiketlenmiş Görseller ve Grafikler


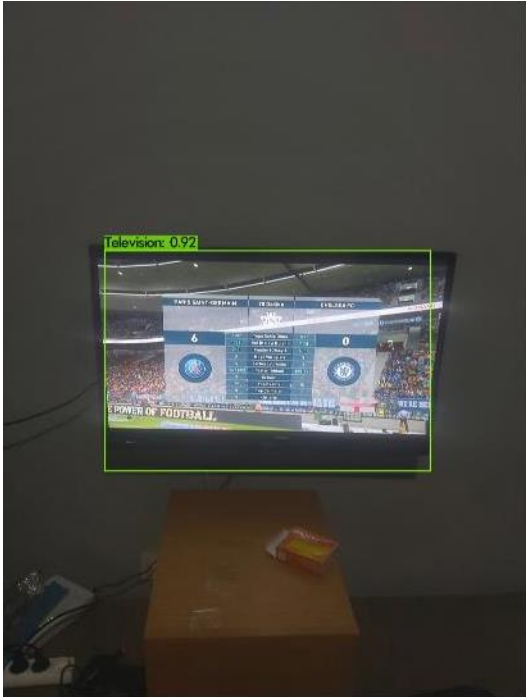
(1.14)




Tesla P100 GPU ve 52GB Ram kullanılarak yapılan eğitim sonucu average loss değerini 0.5206 'a kadar düştüğü yukarıdaki grafikten gözlemleniyor.

(1.15)

 <p>Sofa bed: 0.34</p>	<p>Sofa Bed %34</p>
 <p>Bed: 0.92</p>	<p>Bed %92</p>
 <p>Television: 0.19</p> <p>Kette: 0.10</p> <p>Bed: 0.11</p> <p>Bed: 0.10</p>	<p>Television %19 Bed %11 Bed %10</p>

	<p>Oven %10</p> <p>Oven %21</p>
	<p>Television %92</p>

 <p>A photograph of a bedroom with a bed, a dresser, and a window. A green bounding box is drawn around the bed, with the text "Bed: 0.69" next to it.</p>		Bed %69
 <p>A photograph of a bedroom with a bed, a chair, and a window. A green bounding box is drawn around the bed, with the text "Bed: 0.19" next to it.</p>		Bed %19
 <p>A photograph of a kitchen with a washing machine, a sink, and a stove. A green bounding box is drawn around the washing machine, with the text "Washing machine: 0.18" next to it.</p>		Washing Machine %18

Eğitim sonrası yapay zekayı test aşamasında kullanılan bazı fotoğraflar ve test sonrası ulaşılan yüzdelik değerler yukarıdaki tabloda belirtilmiştir.

(1.16)

Eğitilen Nesne	Eğitim Fotoğrafları	Test Fotoğrafları
Sofa Bed	%55	%43
Television	%75	%65

Yapay zeka eğitimi bittikten sonra detaylı test işlemi için eğitimde kullanılan 100 fotoğrafı ve eğitim sonrası test işlemi için belirlenen 100 fotoğraf yapay zeka üzerinde denenmiştir. Yapay zekanın tanıma yüzdeleri yukarıdaki grafikte belirtilmiştir.

BÖLÜM 2. YOLOV3 VE FLASK KULLANARAK NESNE TANIMA APİSİ

Projemizde kullanılacak Api için bu kaynak dosyasından faydalandık. Projemizin gerektirdikleri doğrultusunda bu kaynak kodlar üzerinde eklemeler ve değişiklikler yaparak Apiyi projemize uygun hale getirdik.

2.1. Neden Flask Kullandık

Flask bir python frameworküdür. Flask hem çabuk öğrenilebilen hemde benchmarklarına bakıldığında performansı gayet yüksek bir frameworktür. Flask'in artıları esnek bir yapıda olması, öğrenilmesi ve uygulaması kolay olması, yönlendirme URL'leri kolay ve derlenmeye ihtiyaç duymaması, işlemlerin süresinin kısılması artı yanlarıdır.

2.2. Api Dosyasına Eklenen Dataset Dosyaları

Daha önce eğitim sırasında oluşturduğumuz Yolov3.cfg config dosyasını Api dizinine ekledik. Aynı şekilde kendi datasetimize ait sınıf isimlerini içeren coco.names isimli dosyayı Api dizinine ekledik. Son olarak datasetimizin eğitimi sonucunda oluşan yolov3.weight isimli ağırlıkları içeren dosyamızı da Api dizinine ekledik.

2.3. Yolo_decection_images.py Scripti Üzerinde Yapılan Değişiklikler

Bu script etiket dosyası, config dosyası ve ağırlık dosyasının yolunu kullanabilmemizi sağlayan fonksiyonlara sahiptir.

(2.1)

```
def get_labels(labels_path):
    #Eğittiğimiz modele ait sınıfları tutan coco.names dosyasının yüklenmesi
    lpath=os.path.sep.join([yolo_path, labels_path])
    LABELS = open(lpath).read().strip().split("\n")
    return LABELS

def get_colors(LABELS):
    #Her sınıfı temsil etmesi için farklı renklerin oluşturulması
    np.random.seed(42)
    COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),dtype="uint8")
    return COLORS

def get_weights(weights_path):
    # YOLO ağırlıklarının ve model konfigürasyon dosya yolunun verilmesi
    weightsPath = os.path.sep.join([yolo_path, weights_path])
    return weightsPath

def get_config(config_path):
    configPath = os.path.sep.join([yolo_path, config_path])
    return configPath

def load_model(configpath,weightspath):
    print("[INFO] YOLO diskten yükleniyor")
    net = cv2.dnn.readNetFromDarknet(configpath, weightspath)
    return net
```

Eğitilmiş sınıfların etiket isimlerini döndüren fonksiyonunu yazdık. Fotoğraf, etiket ve renk parametrelerini alan bu fonksiyon içerde yapılan işlemler sonucunda, tespit edilen nesnelerin etiket isimlerini, bir array içinde geri döndürür.

(2.2)

```
def get_text(image, net, LABELS, COLORS):
    (H, W) = image.shape[:2]

    ln = net.getLayerNames()
    ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]

    blob = ...
    net.setInput(blob)
    start = time.time()
    layerOutputs = net.forward(ln)
    end = time.time()

    print("[INFO] YOLO {:.6f} sn sürdü ".format(end - start))

    boxes = []
    confidences = []
    classIDs = []

    for output in layerOutputs:
        for detection in output:
            idxs = cv2.dnn.NMSBoxes(boxes, confidences, confthres,
                                   nmsthres)
            liste=[]

            if len(idxs) > 0:
                for i in idxs.flatten():
                    (x, y) = (boxes[i][0], boxes[i][1])
                    (w, h) = (boxes[i][2], boxes[i][3])

                    color = [int(c) for c in COLORS[classIDs[i]]]

                    text1 = "{}".format(LABELS[classIDs[i]])
                    List=[]

                    liste.append(text1)

    return liste
```

Tanıyacağı fotoğrafı parametre olarak alan bu fonksiyonumuz kullanılacak dosyaların yollarını alır, get_text fonksiyonunda bu yolları parametre değeri olarak gönderir ve sonucunda etiket isimlerinin olduğu array, txt isimli değere atanır ve geri döndürülür.

(2.3)

```
def runModelText(image):
    labelsPath="./coco.names"
    cfgpath="cfg/yolov3.cfg"
    wpath="yolov3.weights"
    Lables=get_labels(labelsPath)
    CFG=get_config(cfgpath)
    Weights=get_weights(wpath)
    nets=load_model(CFG,Weights)
    Colors=get_colors(Lables)
    txt=get_text(image,nets,Lables,Colors)

    return txt
```

2.4. App.py Dosyasında Yapılan Değişiklikler

Bu kısımda redirect komutu ile app.py dosyasını çalıştırdığımız zaman anasayfa olarak yolunu verdiğimiz index sayfasına yönlendirme yapmaktayız.

(2.4)

```
@app.route('/')
def home():
    return redirect("https://localhost:44370/Home/Index")
```

İmage isimli dosya okunur ve string olarak dönüştürülür. cv2.imdecode() bir bellek önbelleğinden görüntü verilerini okumak ve bunu görüntü formatına dönüştürmek için kullanılır. Bu dönüştürdüğümüz değeri, nesne tanımayı gerçekleştiren fonksiyonumuza parametre olarak verdik ve karşılığında, nesneye ait sınıf ismini “txt” değişkeninde tuttuk.

(2.5)

```
@app.route('/detectObject' , methods=['POST'])
def mask_image():

    file = request.files['image'].read()
    npimg = np.fromstring(file, np.uint8)
    img = cv2.imdecode(npimg, cv2.IMREAD_COLOR)

    txt=runModelText(img)

    return (txt)
```

2.5. Index.js Dosyasında Yapılan Değişiklikler

Html sayfamızda fotoğrafları yüklemek ve apiye göndermek için yer alan “send” butonuna tıklandığında öncelikle fotoğrafın bir input içerisine alınması sağlanır. Bu input konumunda fotoğrafın olup olmamasına göre bir if sorgusu yazılır ve işlemler gerçekleştirilir.

FormData sınıfından yeni bir nesne oluşturulur. FormData tıpkı adından da anlaşılacağı gibi, form verilerini tutmak için tasarlandığından, bir HTML formuna

karşılık gelen nesne oluşturmak için JavaScript ile kullanılır. Oluşturduğumuz bu yeni FormData nesnesine image dosyalarını ekledik.

Sayfa içinde yapılan değişiklikleri Ajax kullanarak gerçekleştirdik. Apinin çalıştığı locali url olarak yazdık. Fonksiyonun başarılı olması durumunda, tanınması yapılmış nesnelerin sınıflarını Html sayfamızda yer alan checkbox formuna gönderdik.

(2.6)

```

window.onload = () => {
  $('#sendbutton').click(() => {

    imagebox = $('#imagebox')
    input = $('#imageinput')[0]
    if(input.files && input.files[0])
    {
      let formData = new FormData();
      formData.append('image', input.files[0]);

      $.ajax({
        url: "http://localhost:5000/detectObject",

        type: "POST",
        data: formData,
        cache: false,
        processData: false,
        contentType: false,

        error: function(data){
          console.log("upload error" , data);
          console.log(data.getAllResponseHeaders());
        },
        success: function(data){
          $.each(data, function (i, val) {

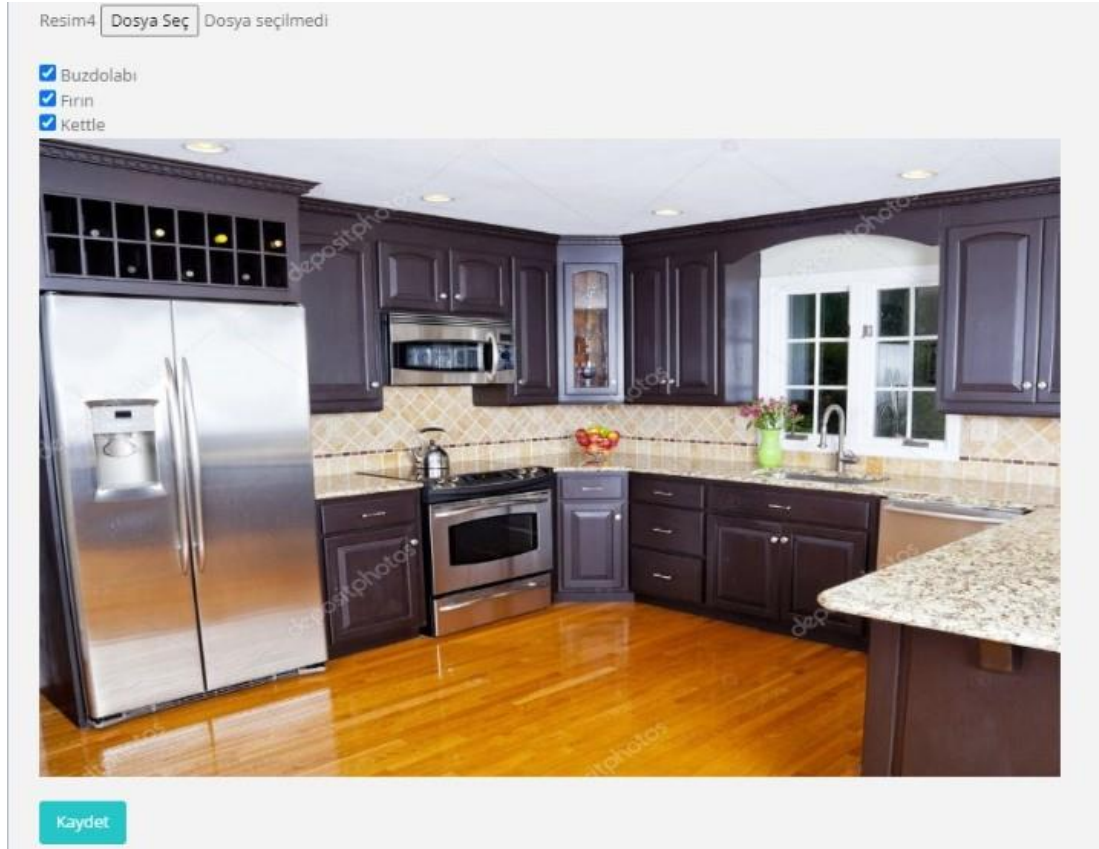
            $("#divID input[value='" + val + "']").prop('checked', true);

          });
        }
      });
    }
  });
};

```

Sonuç olarak yüklenen fotoğraflar üzerinden nesne tanıma yapıldı ve tanınmış nesnelerin sınıfları checkbox formunda işaretlenmiş oldu.

(2.7)



BÖLÜM 3. ASP.NET CORE İLE AIRBNB SİTESİ

3.1. .Net Core Nedir?

Microsoft tarafından açık kaynak kodlu olarak geliştirilen, cross platform olarak çalışabilen geliştirme platformudur.

3.1.1. .Net Core Avantajları

Asp.Net Core kullanmamızın en önemli nedenleri yüksek performansa sahip olması, birleşik MVC ve Web API Frameworkleri, çoklu ortamlar ve geliştirici modu, dependency injection özelliklerine sahip olmasıdır.

3.2. Projede Katmanlı Mimari Kullanılması

Günümüzde artan gereksinimlerden dolayı servis odaklı gibi mimarilerin gelişmesi ile bu üç katmanlı mimari çok katmanlı mimari haline gelmiştir.

Katmanlı mimaride temel olarak 4 katman vardır;

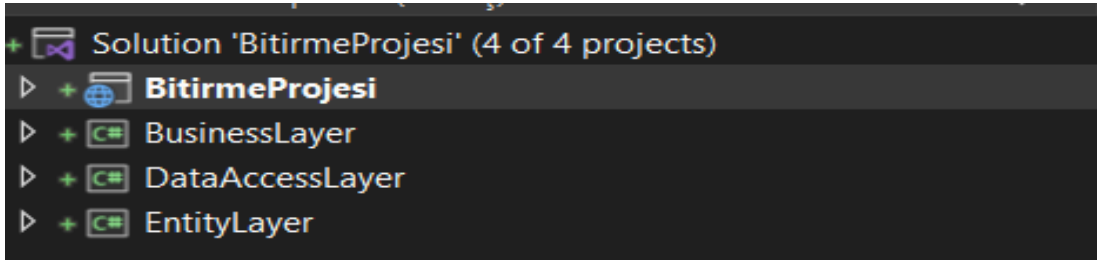
1. Sunum Katmanı (Presentation Layer)
2. İş Katmanı (Business Layer)
3. Veri Katmanı (Data Access Layer)
4. Varlık Katmanı(Entity Layer)

Veri Katmanı'nda sadece veriye erişim işlemleri gerçekleştirilir. Yani veri tabanı bağlantıları bu katmanda gerçekleştirilir. Veri tabanı bağlantısının yanı sıra ekleme, silme, güncelleme ve veri tabanından veri çekme gibi işlemler bu katmanda yapılmaktadır.

İş Katmanı'nda ise sadece iş kodları yazılır yani iş kuralları burada yazılır. Ancak Business katmanı da Data Access Katmanından yararlanır. Data Access'te çektiğimiz verileri Business katmanda işleriz.

Sunum Katmanı ise arayüz katmanı uygulamaların arayüz katmanıdır. Yani kullanıcıyla etkileşime geçilen işlemler bu katmanda gerçekleştirilir.

(3.1)



3.3. Veri Tabanı Uygulamaları

Projemizde Code First yaklaşımını kullandık. Öncelikle entityleri ve propertylerini kodlayıp, sonrasında veri tabanı migration işlemleri ile bağlantı kurduk.

(3.2)

```

using EntityLayer.Concrete;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

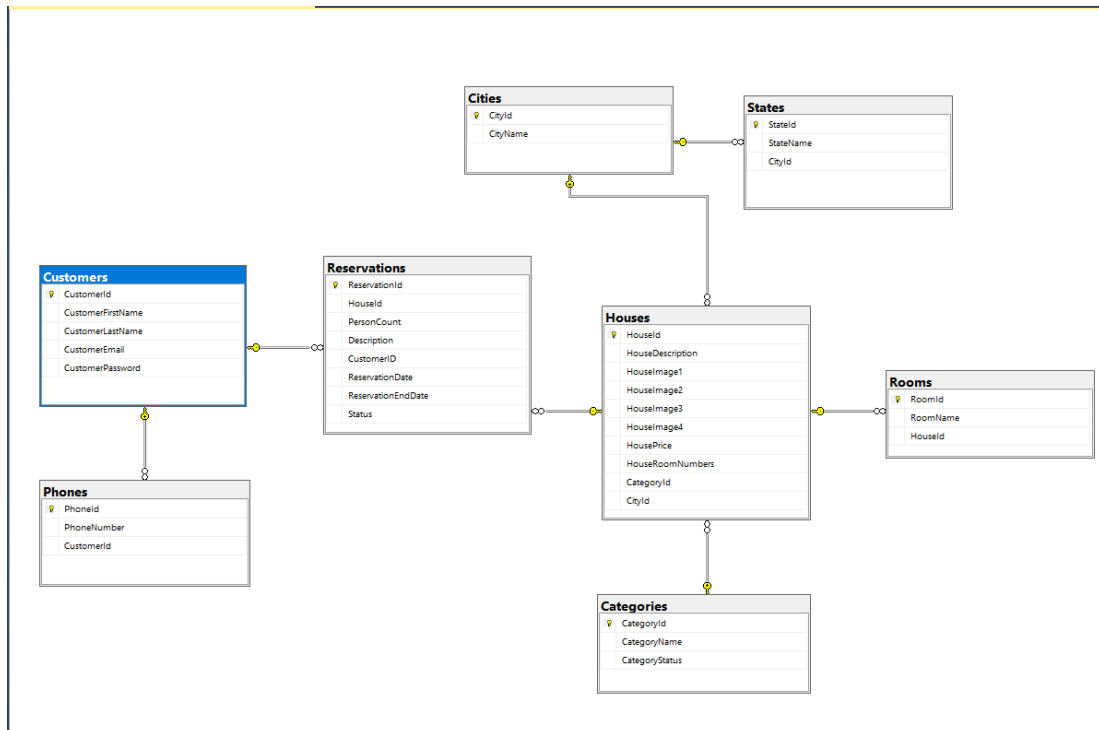
namespace DataAccessLayer.Concrete
{
    public class Context : DbContext
    {
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("server=(localdb)\\Local;database=BitirmeProjesiDB;integrated security=true;");
        }

        public DbSet<House> Houses { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Room> Rooms { get; set; }
        public DbSet<City> Cities { get; set; }
        public DbSet<State> States { get; set; }
        public DbSet<Customer> Customers { get; set; }
        public DbSet<Phone> Phones { get; set; }
        public DbSet<Reservation> Reservations { get; set; }
    }
}

```

Migration işlemi gerçekleştirildikten sonra veri tabanındaki diyagramımız şekilde yer almaktadır.

(3.3)



CRUD işlemleri gerçekleştirilirken, her sınıfta benzer fonksiyonlar kullanıldığından dolayı, bir Generic yapı oluşturduk. Sınıfların bu Generic yapıdan kalıtım alarak CRUD işlemlerini gerçekleştirebilmesini sağladık.

Aynı şekilde Business katmanı için de bir Generic Service oluşturduk ve CRUD işlemlerini gerçekleştirdik.

(3.4)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;

namespace DataAccessLayer.Abstract
{
    public interface IGenericDal<T>
    {
        List<T> GetList();
        List<T> List(Expression<Func<T, bool>> filter);

        T Get(Expression<Func<T, bool>> filter);

        void Insert(T t);
        void Delete(T t);
        void Update(T t);
    }
}
```

3.4. Admin İşlemlerinin Yapılması

3.4.1. Areas Kullanımı

Area yapısı ASP.Net Core projelerinde standart dizin mantığı dışında, daha esnek bir yapı oluşturmamızı sağlar.

Biz de projemizde Admin panelini Areas içinde yönettik. Admin kullanıcısı siteye yeni ev ve kategori girişi yapabilir, evin özelliklerini görüntü işleme kullanarak kolayca siteye ekleyebilir, rezervasyon kayıtlarına ulaşabilir.

(3.5)

Kategoriler						
Açıklama	Kategori	Ücret	Oda Sayısı	Resim	Sil	Güncelle
Ev	1	1500	4	39fa8880-aa41-4a9d-b8fe-0f5a5ca843f1.jpg	Pasif Yap	Güncelle
Ev2	1	1200	8	c881e9f2-bc24-4584-9fb5-cdb93ca39edf.jpg	Pasif Yap	Güncelle
Ev3	1	1300	5	8a409544-5325-4a1e-bdca-688b2b64cc2.jpg	Pasif Yap	Güncelle
Ev4	1	1800	9	31169278-3635-455a-8056-c15b7c580fc5.jpg	Pasif Yap	Güncelle
ev	1	1578	8	8c3818fc-b917-4cab-9f68-f7934335379d.jpg	Pasif Yap	Güncelle

Yeni Ev Ekle

(3.6)

Kategoriler			
ID	Kategori Adı	Sil	Güncelle
1	müstakil	Pasif Yap	Güncelle
2	apartman	Pasif Yap	Güncelle

Yeni Kategori Ekle

(3.7)

Yeni Ev Kaydı

Acıklama

Ev

Kategori

müstakil

Şehir

Adana

Ücret

45

Oda Sayısı

1

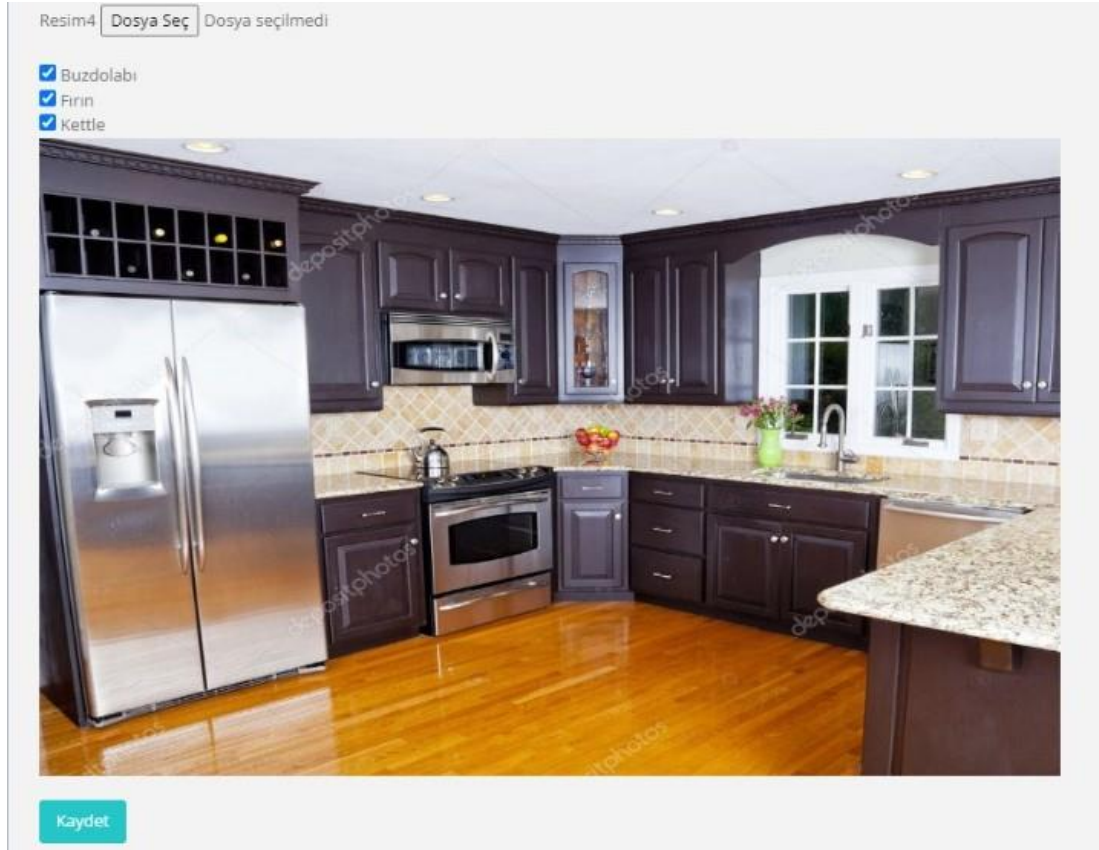
Dosya Seç Dosya seçilmedi

Dosya Seç Dosya seçilmedi

Resim3 Dosya Seç Dosya seçilmedi

Resim4 Dosya Seç Dosya seçilmedi

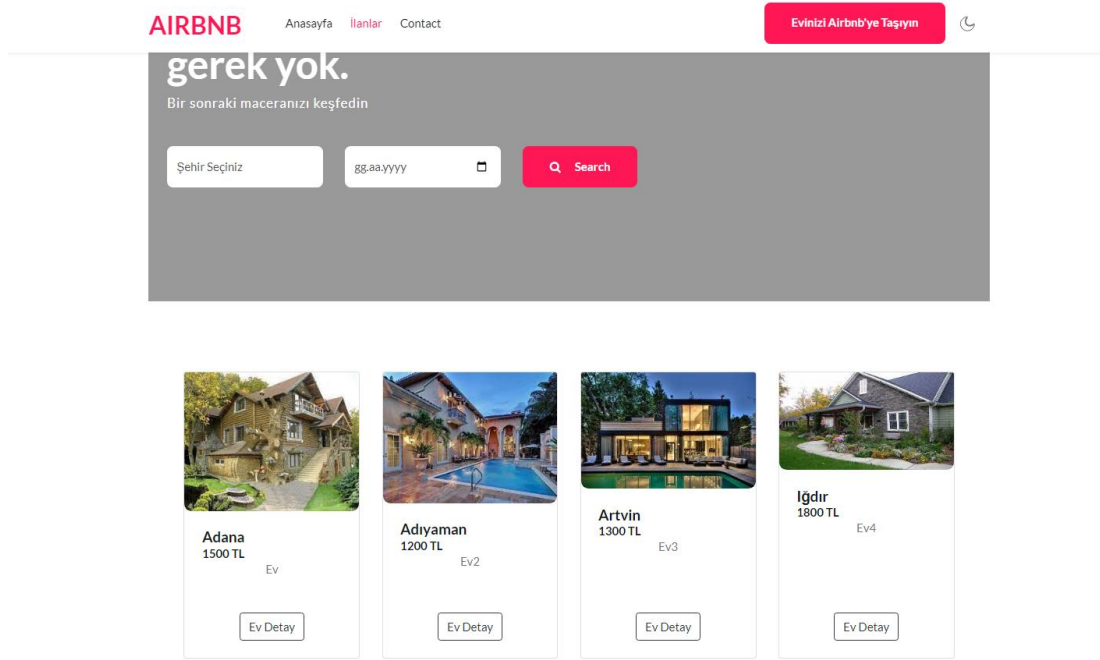
(3.8)



3.5. Index Sayfası

Her kullanıcının giriş yaptığında açılan sayfasıdır. Sayfa açıldığında kayıtlı bütün ev ilanları görüntülenir. SearcBar ile istenilen şehir ve tarihe göre filtreleme işlemi yapılır.

(3.9)

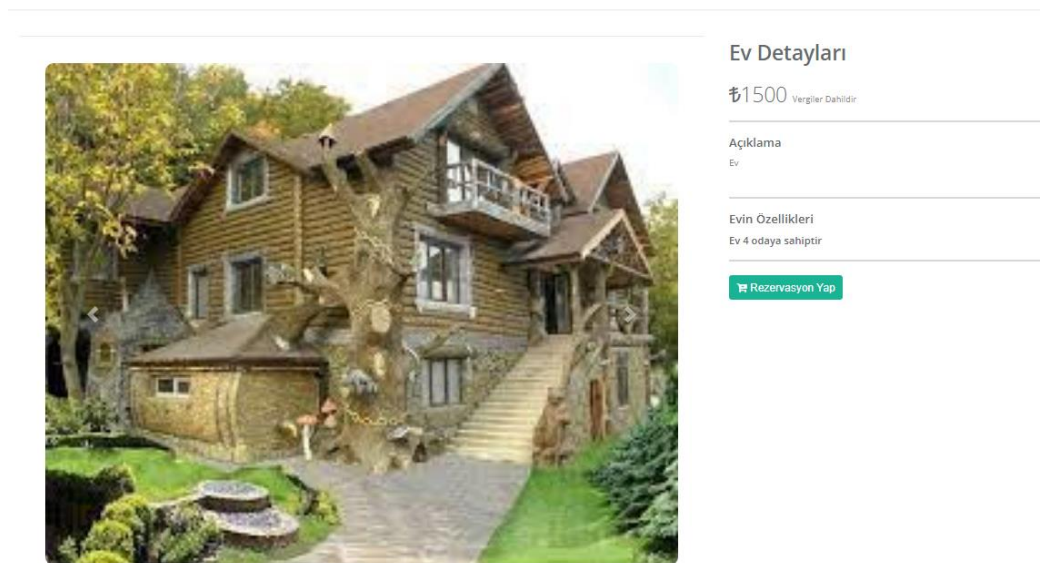


3.6. Ev Detay Sayfası

İlan üzerinde bulunan Ev Detay butonuna tıklandığında evin özellikleri ve fotoğrafları gelmektedir.

Kullanıcı dilerse Rezervasyon butonuna tıklayarak rezervasyon da oluşturabilir.

(3.10)



BÖLÜM 4. SONUÇLAR VE ÖNERİLER

Sonuç olarak projemizde, kullanıcılar Airbnb sayfamızı kullanarak ev ilanlarını görüntüleyebilecek, rezervasyon yapabilecek, siteye kaydolmalarıyla admin panelini kullanabilecek, siteye ev ilanı verebileceklerdir.

Ev ilanı vermek istediklerinde, evin fotoğraflarını Api aracılığı ile görüntü işleyebilecek ve eve ait nesnelerin tanınmasını kolaylaştırarak evin özelliklerini belirleyebileceklerdir.

KAYNAKLAR

- [1] Darknet Repository <https://github.com/pjreddie/darknet>
- [2] Yolov3 Colab Repository <https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial>
- [3] Object Detection Api <https://github.com/ViAsmit/YOLOv5-Flask>

ÖZGEÇMİŞ

Gizem Yiğit, Ankara’da 2000 yılında doğdu. Lise eğitimini Nuh Mehmet Küçükçalık Anadolu Lisesi’nde tamamladı. 2019 senesinde Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü’nde lisans eğitimine başladı.

Zelal İnanç, İstanbul’da 1997 yılında doğdu. Lise eğitimini İstanbul Büyükçekmece Atatürk Anadolu Lisesi’nde tamamladı. 2019 senesinde Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü’nde lisans eğitimine başladı.

Oğuzhan Türkoğlu, Adana’da 2000 yılında doğdu. Lise eğitimini Adana Anadolu Lisesi’nde tamamladı. 2019 senesinde Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü’nde lisans eğitimine başladı.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : GÖRÜNTÜ İŞLEME İLE NESNE TANIMA YAPAN AIRBNB UYGULAMASI

ÖĞRENCİLER : B191210055 Gizem Yiğit

B191210067 Zelal İnanc

B191210010 Oğuzhan Türkoğlu

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problem tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişkisi modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN: CAN YÜZKOLLAR

DANIŞMAN İMZASI: