LAMPIRAN SOURCE CODE

Source Code ANPR :

```
# Function Form From Camera
@login_required(login_url="/admin/login")
def upload_file(request):
    current_datetime = datetime.now()
    start_t = current_datetime.replace(hour=0, minute=0, second=0)
    end_t = current_datetime.replace(hour=23, minute=59,
second=59)
    #Method yang diperbolehkan hanya POST
    if request.method == "POST":
        #Get image data
        if(request.POST['type_form'] == 'cam'):
            image_data = request.POST.get('image', None)
            format, imgstr = image_data.split(';base64,')
            ext = format.split('/')[-1]
            fname = current_datetime.strftime("%Y-%m-%d-%H%M%S")
            image_data = ContentFile(base64.b64decode(imgstr),
name=f'{fname}.{ext}')
        elif(request.POST['type_form'] == 'upload'):
            file_content = request.FILES['image'].read()
            fname = request.FILES['image'].name
            image_data = ContentFile(file_content, name=fname)

        gate = Gate.objects.get(id=request.POST['gate_id'])
        if image_data and gate:
            vehicle = Vehicle.objects.create(file_name=image_data,
gate=gate)

            #Menyiapkan image untuk prediksi CNN
            img = image.load_img('media/'+str(vehicle.file_name),
target_size=(128, 128, 3))
            x = image.img_to_array(img)
            x = x/127.5-1
            x = np.expand_dims(x, axis=0)
            images = np.vstack([x])

            #Prediksi CNN
            prediction_array = model.predict(images)
            class_names = ['mobil', 'motor']
            label = class_names[np.argmax(prediction_array)]
            accuracy = 100 * np.max(prediction_array)

            #Jika akurasi CNN lebih dari 75 maka kendaraan boleh
masuk
            if accuracy >= 75:
                #Penerapan OCR untuk mendapatkan karakter plat
nomor
                ocr_reader = easyocr.Reader(['en'])
                ocr_result =
ocr_reader.readtext('media/'+str(vehicle.file_name))
                ocr_plat_number = ''
                for re in (ocr_result):
                    ocr_plat_number += str(re[1])+' '

                vehicle.accuracy = accuracy
                vehicle.label = label
```

```
                vehicle.desc = request.POST['type']
                vehicle.plat_number = ocr_plat_number

                list_parkiran = ''

                #Cek kendaraan apakah cocok dengan tipe parkiran
                if vehicle.label == gate.parkiran.type:
                    success = True
                    #Cek apakah kendaraan sudah ada di parkiran
                    check_vehicle_in_parkiran =
Vehicle.objects.filter(gate=gate, plat_number=ocr_plat_number,
desc="masuk")
                    check_vehicle_ex_parkiran =
Vehicle.objects.filter(gate=gate, plat_number=ocr_plat_number,
desc="keluar")

                    if request.POST['type'] == 'masuk':
                        if check_vehicle_in_parkiran and not
check_vehicle_ex_parkiran:
                            vehicle.delete()
                            msg = 'Kendaraan sudah ada di
parkiran'

                            success = False
                        else:
                            msg = 'Kendaraan dapat masuk ke
parkiran'
                            vehicle.save()
                            gate.capacity += 1
                    elif request.POST['type'] == 'keluar':
                        if check_vehicle_in_parkiran:
                            msg = 'Kendaraan dapat keluar dari
parkiran'

                            vehicle.save()
                            gate.capacity -= 1
                        else:
                            vehicle.delete()
                            msg = 'Kendaraan tidak ada di
parkiran'
                            success = False
                    gate.save()
                else:
                    success = False
                    vehicle.delete()
                    list_parkiran =
Gate.objects.filter(parkiran__type=vehicle.label,
start_time__range=[start_t, end_t])

                    msg = 'Kendaraan tidak cocok dengan tipe
parkiran'
            else:
                success = False
                msg = 'Sistem memprediksi kendaraan bertipe '+
label + ' dengan akurasi ' + str(accuracy)[:5] +', akurasi
tersebut dibawah ketentuan, silahkan upload ulang foto plat nomor
dengan resolusi dan posisi plat nomor lebih baik'
                messages.success(request, msg)
```

```
                vehicle.delete()

                url =
'/app/'+str(gate.parkiran.id)+'/'+request.POST['type']+'/gate'
                return HttpResponseRedirect(url)

            return render(request, "app/result.html", {
                'gate': gate,
                'type': request.POST['type'],
                'vehicle': vehicle,
                'label': label,
                'success': success,
                'msg': msg,
                'list_parkiran': list_parkiran
            })
        else:
            return HttpResponse("Format file tidak didukung,
silahkan uplaod gambar dengan ekstensi jpg, png, atau jpeg")
    else:
        return HttpResponse("Method Tidak Diizinkan")

@login_required(login_url="/admin/login")
def dashboard(request):
    now = datetime.now()
    search_date = request.GET.get('date_input')
    if search_date:
        convert_searh_date = datetime.strptime(search_date, '%Y-
%m-%d')
        current_datetime =
datetime.combine(convert_searh_date.date(), now.time())
    else:
        current_datetime = now
    start_t = current_datetime.replace(hour=0, minute=0, second=0)
    end_t = current_datetime.replace(hour=23, minute=59,
second=59)

    #Data Motor
    #Mendapatkan gate motor yang buka di hari ini
    gates_motor_id =
list(Gate.objects.filter(start_time__range=[start_t, end_t],
parkiran__type="motor").values_list('id', flat=True))

    #Mendapatkan nama parkiran yang akan digunakan sbg label
grafik
    gates_motor_for_label_parkiran =
list(Gate.objects.filter(start_time__range=[start_t, end_t],
parkiran__type="motor").values_list('parkiran_id', flat=True))
    parkiran_motor_names =
list(Parkiran.objects.filter(id__in=gates_motor_for_label_parkiran
).values_list('name', flat=True))

    #Mendapatkan jumlah kendaraan yang masuk sesuai gate yang buka
    datas_motors_masuk =
Vehicle.objects.filter(gate_id__in=gates_motor_id, label="motor",
desc="masuk").values('gate_id').annotate(count=Count('id'))
    datas_motors_masuk_array = [entry['count'] for entry in
datas_motors_masuk]

    #Mendapatkan jumlah kendaraan yang keluar sesuai gate yang
```

```
buka
    datas_motors_keluar =
Vehicle.objects.filter(gate_id__in=gates_motor_id, label="motor",
desc="keluar").values('gate_id').annotate(count=Count('id'))
    datas_motors_keluar_array = [entry['count'] for entry in
datas_motors_keluar]
    #Data Motor

    #Data Mobil
    #Mendapatkan gate motor yang buka di hari ini
    gates_mobil_id =
list(Gate.objects.filter(start_time__range=[start_t, end_t],
parkiran__type="mobil").values_list('id', flat=True))

    #Mendapatkan nama parkiran yang akan digunakan sbg label
grafik
    gates_mobil =
list(Gate.objects.filter(start_time__range=[start_t, end_t],
parkiran__type="mobil").values_list('parkiran_id', flat=True))
    parkiran_mobil_names =
list(Parkiran.objects.filter(id__in=gates_mobil).values_list('name
', flat=True))

    #Mendapatkan jumlah kendaraan yang masuk sesuai gate yang buka
    datas_mobil_masuk =
Vehicle.objects.filter(gate_id__in=gates_mobil_id, label="mobil",
desc="masuk").values('gate_id').annotate(count=Count('id'))
    datas_mobil_masuk_array = [entry['count'] for entry in
datas_mobil_masuk]

    datas_mobil_keluar =
Vehicle.objects.filter(gate_id__in=gates_mobil_id, label="mobil",
desc="keluar").values('gate_id').annotate(count=Count('id'))
    datas_mobil_keluar_array = [entry['count'] for entry in
datas_mobil_keluar]
    #Data Mobil

    return render(request, "app/dashboard.html" ,{
        'current_datetime': current_datetime,
        'labels_motor': json.dumps(parkiran_motor_names),
        'labels_mobil': json.dumps(parkiran_mobil_names),
        'datas_motor_masuk': json.dumps(datas_motors_masuk_array),
        'datas_motor_keluar':
json.dumps(datas_motors_keluar_array),
        'datas_mobil_masuk': json.dumps(datas_mobil_masuk_array),
        'datas_mobil_keluar':
json.dumps(datas_mobil_keluar_array),
    })
```

Source Code CNN :

```
#Import library yang dibutuhkan

#CNN
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense,
Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization,
Flatten,Input
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
from keras.callbacks import ModelCheckpoint
```

Lampiran 3 Source Code CNN dan ANPR