3. Source Code

```
import cv2
import requests
import base64
from roboflow import Roboflow

API_KEY = "5fbsx9JHZdIBoA9wAGBP"
MODEL_ENDPOINT = "people-object-detection-rxb7e"
VERSION = 3

rf = Roboflow(api_key=API_KEY)
project = rf.workspace().project(MODEL_ENDPOINT)
model = project.version(VERSION).model

def detect_objects(frame):
    try:
        _, buffer = cv2.imencode('.jpg', frame)
        img_str = base64.b64encode(buffer).decode('utf-8')
        response = requests.post(
            f"https://detect.roboflow.com/{MODEL_ENDPOINT}/{VERSION}",
            params={
                "api_key": API_KEY,
                "confidence": 50,
                "overlap": 60,
                "format": "json"
            },
            data=img_str,
            headers={"Content-Type": "application/x-www-form-urlencoded"}
        )
        predictions = response.json()

        # Pastikan semua prediksi memiliki bbox
        for pred in predictions['predictions']:
            if 'bbox' not in pred:
                pred['bbox'] = {
                    'x': pred['x'],
                    'y': pred['y'],
                    'width': pred['width'],
                    'height': pred['height']
                }

        return predictions
    except Exception as e:
        print(f"An error occurred: {e}")
        return None
```

```
import MySQLdb
import cv2
```

```
import pytz
import numpy as np
from flask import Flask, Response, current_app, render_template, redirect,
url_for, flash, request, jsonify
from flask_login import LoginManager, login_user, login_required, logout_user,
current_user, UserMixin
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime
from db import app
from models import create_admin, get_admin_by_username, create_report
from camera import VideoCamera
from api import detect_objects
from pathlib import Path
import base64
from io import BytesIO
from PIL import Image
import os
import time
import hashlib
import threading
import math
import logging
logging.basicConfig(level=logging.DEBUG)
from dateutil.parser import isoparse # Tambahkan import ini
from config import Config, get_db_connection

app = Flask(__name__)
app.secret_key = 'bd8c592b8fc3bd94861eda0932c8d7c2'

# Zona waktu Indonesia/Jakarta
jakarta = pytz.timezone('Asia/Jakarta')

# Membuat koneksi ke database
mysql = get_db_connection()
if mysql is None:
    raise Exception("Database connection failed")


login_manager = LoginManager(app)
login_manager.login_view = 'auth'

image_save_dir = os.path.join("static", "images")
os.makedirs(image_save_dir, exist_ok=True)

class Admin(UserMixin):
    def_init_(self, id, username, nama):
        self.id = id
        self.username = username
        self.nama = nama

    def is_active(self):
        return True
```

```
    def is_authenticated(self):
        return True

    def is_anonymous(self):
        return False

    def get_id(self):
        return str(self.id)

@login_manager.user_loader
def load_user(user_id):
    cursor = mysql.cursor()
    cursor.execute("SELECT * FROM admin WHERE id = %s", [user_id])
    admin = cursor.fetchone()
    cursor.close()
    if admin:
        return Admin(id=admin[0], username=admin[1], nama=admin[2])
    return None

def get_admin_by_username(username):
    cursor = mysql.cursor()
    cursor.execute("SELECT * FROM admin WHERE username = %s", [username])
    admin = cursor.fetchone()
    cursor.close()
    return admin

def get_week_number(date_str):
    # Convert the string to a datetime object
    date = datetime.strptime(date_str, '%Y-%m-%d').date()

    # Get the ISO calendar week number
    year, week_num, weekday = date.isocalendar()

    return week_num


@app.route('/auth', methods=['GET', 'POST'])
def auth():
    action = request.args.get('action')
    if action == 'register' and request.method == 'POST':
        username = request.form.get('username')
        nama = request.form.get('nama')
        password = request.form.get('password')

        try:
            cursor = mysql.cursor()
            cursor.execute("SELECT * FROM admin WHERE username = %s",
[username])
            existing_admin = cursor.fetchone()

            if existing_admin:
```

```
                flash('Username sudah tersedia. Tolong gunakan username lain.',
'danger')
                return render_template('auth.html', action='register')
            create_admin(username, nama, password)
            cursor.close()

            flash('Pendaftaran berhasil! Kamu bisa masuk sekarang.', 'success')
            return redirect(url_for('auth', action='login'))

        except Exception as e:
            print(f"An error occurred: {e}")
            flash('Duh, terdapat kesalahan, nih. Coba lagi ya.', 'danger')
            return render_template('auth.html', action='register')

    if action == 'login' and request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        print(f"Username: {username}")
        print(f"Password: {password}")

        admin = get_admin_by_username(username)

        if admin:
            print(f"Admin Found: {admin[1]}")
            print(f"Stored Hashed Password: {admin[3]}")
            password_check = check_password_hash(admin[3], password)
            print(f"Password Check: {password_check}")
            if password_check:
                print("Password is correct")
                login_user(Admin(id=admin[0], username=admin[1],
nama=admin[2]))
                print("Redirecting to dashboard...")
                return redirect(url_for('dashboard'))
            else:
                print("Hm, sepertinya password kamu salah!")
        else:
            print("User tidak ditemukan, nih..")

        flash('Kredensial salah', 'danger')
        print("Kredensial salah")
        return render_template('auth.html', action='login')

    return render_template('auth.html', action=action)

@app.route('/')
def index():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))
    else:
        return redirect(url_for('auth', action='login'))
```

```
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('auth', action='login'))

@app.route('/dashboard')
@login_required
def dashboard():
    today = datetime.today().date()
    cursor = mysql.cursor()

    # Total deteksi hari ini
    cursor.execute("SELECT COUNT(*) FROM report WHERE tanggal = %s", [today])
    visits_today = cursor.fetchone()[0]

    # Total deteksi keseluruhan
    cursor.execute("SELECT COUNT(*) FROM report")
    total_deteksi = cursor.fetchone()[0]

    # Rata-rata pengunjung
    cursor.execute("SELECT AVG(total_deteksi) FROM report")
    rata_rata_pengunjung = cursor.fetchone()[0]

    # Data untuk grafik
    cursor.execute("SELECT tanggal, SUM(total_deteksi) FROM report GROUP BY
tanggal")
    report_data = cursor.fetchall()
    chart_labels = [str(row[0]) for row in report_data]
    chart_data = [row[1] for row in report_data]
    cursor.execute("SELECT tanggal, total_deteksi FROM report ORDER BY tanggal
ASC")
    report_data = cursor.fetchall()
    week_report = {}
    temp = 0
    tanggal = ""
    for row in report_data:
        if tanggal != str(row[0]):
            tanggal = str(row[0])
            temp = 0

        week_number= get_week_number(tanggal)
        key = "Minggu "+str(week_number)

        if key not in week_report:
            temp = row[1]
            week_report[key] = temp
        elif row[1] > temp:
            week_report[key] += row[1] - temp
            temp = row[1]
        elif row[1] == temp and temp == 1:
            week_report[key] += 1
```

```
        else:
            temp = row[1]
    week_report = dict(sorted(week_report.items()))
    week_report_labels = list(week_report.keys())
    week_report_data = list(week_report.values())
    # Semua laporan, urutkan berdasarkan tanggal dan waktu terbaru
    cursor.execute("SELECT tanggal, total_deteksi FROM report WHERE tanggal >=
CURRENT_DATE - INTERVAL 7 DAY ORDER BY waktu ASC")
    report_data = cursor.fetchall()
    lastWeekReport = {}
    temp = 0
    for row in report_data:
        key = str(row[0])
        if key not in lastWeekReport:
            temp = row[1]
            lastWeekReport[key] = temp
        elif row[1] > temp:
            lastWeekReport[key] += row[1] - temp
            temp = row[1]
        elif row[1] == temp and temp == 1:
            lastWeekReport[key] += 1
        else:
            temp = row[1]
    lastWeekReport = dict(sorted(lastWeekReport.items()))
    lastWeekReportLabel = list(lastWeekReport.keys())
    lastWeekReportData = list(lastWeekReport.values())
    # Semua laporan, urutkan berdasarkan tanggal dan waktu terbaru
    cursor.execute("SELECT * FROM report ORDER BY tanggal DESC, waktu DESC")
    reports = cursor.fetchall()
    cursor.execute("SELECT tanggal, COUNT(nomor_induk) FROM pengunjung_report
GROUP BY tanggal ORDER BY tanggal DESC")
    pengunjung_reports = cursor.fetchall()
    pengunjung_labels = [str(row[0]) for row in pengunjung_reports]
    pengunjung_data = [row[1] for row in pengunjung_reports]
    print(pengunjung_labels)

    # TEST CODING
    cursor.execute("SELECT tanggal, total_deteksi FROM report ORDER BY tanggal
ASC")
    report_data = cursor.fetchall()
    totalPengunjungReport  =  {}
    temp = 0
    for row in report_data:
        key = str(row[0])
        if key not in totalPengunjungReport:
            temp = row[1]
            totalPengunjungReport[key] = temp
        elif row[1] > temp:
            totalPengunjungReport[key] += row[1] - temp
            temp = row[1]
        elif row[1] == temp and temp == 1:
            totalPengunjungReport[key] += 1
```

```
        else:
            temp = row[1]
    totalPengunjungReport = dict(sorted(totalPengunjungReport.items(),
reverse=True))
    cursor.close()
    return render_template('dashboard.html',
                           visits_today=visits_today,
                           total_deteksi=total_deteksi,
                           rata_rata_pengunjung=rata_rata_pengunjung,
                           last_week_label = lastWeekReportLabel,
                           last_week_data = lastWeekReportData,
                           chart_labels=chart_labels,
                           chart_data=chart_data,
                           week_report_labels = week_report_labels,
                           week_report_data = week_report_data,
                           reports=reports,
                           pengunjung_reports = pengunjung_reports,
                           pengunjung_labels=pengunjung_labels,
                           pengunjung_data = pengunjung_data,
                           totalPengunjungReport = totalPengunjungReport)

@app.route('/delete_report/<int:report_id>', methods=['POST'])
def delete_report(report_id):
    try:
        cursor = mysql.cursor()
        cursor.execute("DELETE FROM report WHERE id = %s", (report_id,))
        mysql.commit()
        cursor.close()
        flash('Laporan berhasil dihapus', 'success')
    except Exception as e:
        flash('Terjadi kesalahan saat menghapus laporan', 'danger')
        print(f"Error: {e}")

    return redirect(url_for('dashboard'))

@app.route('/detect')
@login_required
def detect():
    return render_template('detect.html')

@app.route('/detected_people', methods=['POST'])
def detected_people():
    data = request.json
    image_data = data['image']
    total_deteksi = data['detectedCount']
    print("total deteksi", total_deteksi)
    # Remove the base64 header
    image_data = image_data.split(",")[1]

    # Decode the image data
    image_data = base64.b64decode(image_data)
```

```
    # Save the image
    now = datetime.now(jakarta)
    timestamp = now.strftime("%Y%m%d_%H%M%S")
    image_filename = f"{timestamp}.jpg"
    image_path = os.path.join(image_save_dir, image_filename)

     # Save the image
    with open(image_path, 'wb') as f:
        f.write(image_data)
    image_path = image_path.replace("\\", "/")
    image_path = image_path.replace("static/", "")
    hari = now.strftime("%A")
    tanggal = now.date()
    waktu = now.time()
    create_report(hari, tanggal, waktu, total_deteksi, image_path, mysql)

    return jsonify({"message": "success"})

@app.route('/pengunjung')
@login_required
def pengunjung():
    cursor = mysql.cursor()

    # Data  untuk  grafik
    cursor.execute("SELECT * FROM pengunjung")
    pengunjung = cursor.fetchall()
    return render_template('pengunjung.html',
                           pengunjung=pengunjung)

@app.route('/pengunjung', methods=["POST"])
@login_required
def pengunjungPost():
    data = request.form
    nama = data["nama"]
    nomor_induk = data["nomor-induk"]
    tanggal_lahir = data["tanggal-lahir"]
    # Data untuk grafik
    try:
        cursor = mysql.cursor()
        cursor.execute("INSERT INTO pengunjung (nama, nomor_induk,
tanggal_lahir) VALUES (%s, %s, %s)",
                       (nama, nomor_induk, tanggal_lahir))
        mysql.commit()
        cursor.close()
    except Exception as e:
        print(f"An error occurred: {e}")

    return redirect(url_for('pengunjung'))

@app.route('/qrscan', methods=["POST"])
@login_required
def qrscan():
```

```
    data = request.json
    nomorInduk = data["nomorInduk"]
    print(nomorInduk)
    now = datetime.now(jakarta)
    tanggal = now.date()
    waktu = now.time()
    try:
        cursor = mysql.cursor()
        # query =
        cursor.execute("SELECT nama FROM pengunjung WHERE nomor_induk =
'{0}'".format(nomorInduk))
        print("tes")
        pengunjung = cursor.fetchall()
        print(pengunjung)
        if(len(pengunjung) == 0):
            return jsonify({"code": 400,"msg": "QR Code tidak terdaftar"})
        nama = pengunjung[0][0]
        cursor.execute("INSERT INTO pengunjung_report (nomor_induk, tanggal,
waktu) VALUES (%s, %s, %s)",
                       (nomorInduk, tanggal, waktu))
        mysql.commit()
        cursor.close()
        return jsonify({"code": 200, "msg": "Selamat datang,
{0}".format(nama)})
    except Exception as e:
        print(e)
        return jsonify({"code": 500, "msg": (f"An error occurred: {e}")})

@app.route('/delete_pengunjung/<int:pengunjung_id>', methods=['POST'])
def delete_pengunjung(pengunjung_id):
    try:
        cursor = mysql.cursor()
        cursor.execute("DELETE FROM pengunjung WHERE id = %s",
(pengunjung_id,))
        mysql.commit()
        cursor.close()
        flash('Pengunjung berhasil dihapus', 'success')
    except Exception as e:
        flash('Terjadi kesalahan saat menghapus Pengunjung', 'danger')
        print(f"Error: {e}")

    return redirect(url_for('pengunjung'))

if__name__== '__main__':
    app.run(debug=True)
```

```
import cv2

class VideoCamera:
    def___init__(self):
        self.video = cv2.VideoCapture(0)  # Ganti 0 dengan URL RTSP kalo butuh
```

```
    def_del_(self):
        self.video.release()

    def get_frame(self):
        success, image = self.video.read()
        ret, jpeg = cv2.imencode('.jpg', image)
        return jpeg.tobytes(), image  # Mengembalikan frame untuk pemrosesan
lanjutan
```

```
import secrets
import MySQLdb

class Config:
    SECRET_KEY = secrets.token_hex(16)
    MYSQL_HOST = 'deteksipengunjung.mysql.pythonanywhere-services.com'
    MYSQL_USER = 'deteksipengunjun'
    MYSQL_PASSWORD = 'db_perpustakaan'
    MYSQL_DB = 'deteksipengunjun$default'

def get_db_connection():
    try:
        connection = MySQLdb.connect(
            host=Config.MYSQL_HOST,
            user=Config.MYSQL_USER,
            passwd=Config.MYSQL_PASSWORD,
            db=Config.MYSQL_DB
        )
        connection.ping(True)  # Reconnect if the connection is lost
        return connection
    except MySQLdb._exceptions.OperationalError as e:
        print(f"Error connecting to the database: {e}")
        return None
```

```
from flask import Flask
from flask_mysqldb import MySQL
import MySQLdb

app = Flask(_name_)
app.config.from_object('config.Config')

mysql = MySQL(app)
```

```
import MySQLdb
from werkzeug.security import generate_password_hash
from db import mysql

def create_admin(username, nama, password):
    cursor = mysql.connection.cursor()
    hashed_password = generate_password_hash(password, method='pbkdf2:sha256')
```

```
    print("Hashed Password:", hashed_password)
    cursor.execute("INSERT INTO admin (username, nama, password) VALUES (%s,
%s, %s)", (username, nama, hashed_password))
    mysql.connection.commit()
    cursor.close()

def get_admin_by_username(username):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM admin WHERE username = %s", [username])
    admin = cursor.fetchone()
    cursor.close()
    return admin

def create_report(hari, tanggal, waktu, total_deteksi, image_path, mysql):
    try:
        image_path = image_path.replace("\\", "/") # Ensure the path uses
forward slashes
        cursor = mysql.cursor()
        formatted_time = waktu.strftime("%H:%M:%S")
        query = "INSERT INTO report (hari, tanggal, waktu, total_deteksi,
image_path) VALUES (%s, %s, %s, %s, %s)"
        cursor.execute(query, (hari, tanggal, formatted_time, total_deteksi,
image_path))
        mysql.commit()
        cursor.close()
        print(f"Report created: {hari}, {tanggal}, {formatted_time},
{total_deteksi}, {image_path}")
    except Exception as e:
        print(f"Error creating report: {e}")
```

LEMBAR PERNYATAAN

Yang bertanda tangan dibawah ini:

Nama :Wilson Friendnadi

NIM 2018230215

Jurusan :Teknologi Informasi

Menyatakan bahwa Laporan Tugas Akhir ini saya susun sendiri berdasarkan hasil peninjauan, penelitian lapangan, wawancara serta memadukannya dengan buku- buku, literature atau bahan-bahan referensi lain yang terkait dan relevan di dalam penyelesaian Laporan Tugas Akhir ini.

Demikian pernyataan ini penulis buat dengan sesungguhnya.Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Jakarta, 20 Juni 2024

(Wilson Friendnadi)