LEMBAR SOURCE CODE

***SOURCE CODE FACE RECOGNITION* :**

```
import cv2
from flask import Flask, render_template, Response, request,
redirect, session, jsonify
import mysql.connector
from datetime import datetime
import time
from PIL import Image
import numpy as np
import os

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

# Koneksi database
conn = mysql.connector.connect(host='localhost',
                               database='db_farhan',
                               user='root',
                               password='')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainer/trainer.yml")
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)

last_save_time = 0  # Variabel untuk menyimpan waktu terakhir gambar
disimpan

def save_data_to_mysql(nama, tanggal, gambar):
    try:
        if conn.is_connected():
            cursor = conn.cursor()
            sql = "INSERT INTO face_data (nama, tanggal, gambar)
VALUES (%s, %s, %s)"
            val = (nama, tanggal, gambar)
            cursor.execute(sql, val)
            conn.commit()
            cursor.close()
    except Exception as e:
        print("Error while connecting to MySQL", e)

font = cv2.FONT_HERSHEY_SIMPLEX
```

```
names = ["None", "Farhan", "Fachri", "Mrs. Timur", "Mr. Yahya",
"CS"]

cam = cv2.VideoCapture(0)
cam.set(3, 640)  # set lebar video
cam.set(4, 480)  # set tinggi video
minW = 0.1 * cam.get(3)
minH = 0.1 * cam.get(4)

detected_user = None

def gen_frames():
    global detected_user
    while True:
        ret, frame = cam.read()
        if not ret:
            break
        else:
            frame = cv2.flip(frame, 1)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = faceCascade.detectMultiScale(
                gray,
                scaleFactor=1.2,
                minNeighbors=5,
                minSize=(int(minW), int(minH)),
            )
            for x, y, w, h in faces:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0,
255, 0), 2)
                id, confidence = recognizer.predict(gray[y : y + h,
x : x + w])
                if confidence < 100:
                    detected_user = names[id]
                    confidence = "  {0}%".format(round(100 -
confidence))
                    cv2.putText(frame, str(detected_user), (x + 5, y
- 5), font, 1, (255, 255, 255), 2)
                else:
                    detected_user = "unknown"
                    confidence = "  {0}%".format(round(100 -
confidence))
                    cv2.putText(frame, str(detected_user), (x + 5, y
- 5), font, 1, (255, 255, 255), 2)
                cv2.putText(frame, str(confidence), (x + 5, y + h -
5), font, 1, (255, 255, 0), 1)

            ret, buffer = cv2.imencode('.jpg', frame)
```

```
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-
replace; boundary=frame')

@app.route('/detected_user')
def detected_user_route():
    global detected_user
    return jsonify({"user": detected_user})

@app.route('/')
def index():
    if 'username' in session:
        return redirect('/dashboard')
    else:
        return render_template('index.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username=%s AND
password=%s", (username, password))
        user = cursor.fetchone()

        if user:
            session['username'] = username
            return redirect('/dashboard')
        else:
            error = "Invalid username or password"
            return render_template('login.html', error=error)
    else:
        return render_template('login.html')

@app.route('/dashboard')
def dashboard():
    if 'username' in session:
        return render_template('dashboard.html')
    else:
```

```
        return redirect('/login')

@app.route("/add_dataset")
def add_dataset():
    return render_template("add_dataset.html")

@app.route("/start_capture", methods=["POST"])
def start_capture():
    face_id = request.form.get("face_id")
    if face_id is not None:
        # Inisialisasi kamera
        cam = cv2.VideoCapture(0)
        cam.set(3, 640)  # set video width
        cam.set(4, 480)  # set video height
        # Load pre-trained classifier untuk deteksi wajah
        face_detector =
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
        print("\n [INFO] Memulai pengambilan wajah. Lihat kamera dan
tunggu ...")
        count = 0
        while True:
            ret, img = cam.read()
            img = cv2.flip(img, 1)  # Flip horizontally
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_detector.detectMultiScale(gray, 1.3, 5)
            for x, y, w, h in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0,
0), 2)
                count += 1
                # Simpan gambar yang diambil ke dalam folder
datasets
                cv2.imwrite(
                    f"dataset/User.{str(face_id)}.{str(count)}.jpg",
                    gray[y : y + h, x : x + w],
                )
                cv2.imshow("image", img)
            k = cv2.waitKey(100) & 0xFF  # Tekan 'ESC' untuk keluar
dari video
            if k == 27 or count >= 30:
                break
        print("\n [INFO] Pengambilan sampel wajah selesai.")
        cam.release()
        cv2.destroyAllWindows()
        return "Dataset gambar berhasil disimpan untuk Face ID: " +
face_id
    else:
        return "Error: No face ID provided in the form data."
```

```
@app.route("/train")
def train():
    train_model()
    return render_template("train.html")
def train_model():
    # Path untuk dataset wajah
    path = "dataset"
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector =
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    # Fungsi untuk mendapatkan gambar dan label data
    def getImagesAndLabels(path):
        imagePaths = [os.path.join(path, f) for f in
os.listdir(path)]
        faceSamples = []
        ids = []

        for imagePath in imagePaths:
            PIL_img = Image.open(imagePath).convert("L")  # Konversi
ke skala abu-abu
            img_numpy = np.array(PIL_img, "uint8")

            id = int(os.path.split(imagePath)[-1].split(".")[1])
            faces = detector.detectMultiScale(img_numpy)

            for x, y, w, h in faces:
                faceSamples.append(img_numpy[y : y + h, x : x + w])
                ids.append(id)

        return faceSamples, ids
        print("\n [INFO] Melatih wajah. Ini akan memakan waktu
beberapa detik. Tunggu ...")
    faces, ids = getImagesAndLabels(path)
    recognizer.train(faces, np.array(ids))

    # Simpan model ke dalam file trainer/trainer.yml
    recognizer.write("trainer/trainer.yml")
    print(
        "\n [INFO] {0} wajah telah dilatih. Program
selesai".format(len(np.unique(ids)))
    )

@app.route('/riwayat')
def riwayat():
    if 'username' in session:
        cursor = conn.cursor(dictionary=True)
```

```
        cursor.execute("SELECT * FROM face_data ORDER BY tanggal
DESC LIMIT 10")
        riwayat = cursor.fetchall()
        return render_template('history.html', riwayat=riwayat)
    else:
        return redirect('/login')

@app.route('/logout')
def logout():
    session.pop('username', None)
    return redirect('/login')

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

Lampiran 3 *SOURCE CODE*