

LAMPIRAN



Lampiran 1 *Code app.py*

```

from flask import Flask, render_template, request, redirect, url_for, session,
Response, flash, send_file
from pipeline_core import FaceRecognitionPipeline
import cv2

app = Flask(__name__)

system = FaceRecognitionPipeline()

# --- ROUTES AUTHENTICATION ---
@app.route('/')
def index():
    """Route untuk halaman utama Kiosk Mahasiswa (Standby Mode)."""
    return render_template('index_kiosk.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    """Route untuk memproses autentikasi Dosen/Admin."""
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        try:
            user = system.login_admin(username, password)
            if user:
                session['admin'] = True
                session['nama'] = user['nama_lengkap']
                return redirect(url_for('admin_dashboard'))
            else:

```

```

        return render_template('login.html', error="Username atau Password
Salah!")

    except Exception as e:
        return render_template('login.html', error=f"Terjadi Kesalahan Server:
{str(e)}")

    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))

@app.route('/register', methods=['GET', 'POST'])
def register_admin():
    """Route untuk menangani pendaftaran akun Dosen/Admin baru."""
    if request.method == 'POST':
        nama = request.form['nama']
        username = request.form['username']
        password = request.form['password']

        try:
            result = system.register_dosen(nama, username, password)
            if result is True:
                return redirect(url_for('login'))
            else:
                return render_template('register_admin.html', error=result)
        except Exception as e:
            return render_template('register_admin.html', error=f"Terjadi Kesalahan
Server: {str(e)}")

```

```

return render_template('register_admin.html')

# --- ROUTES ADMIN ---
@app.route('/admin_dashboard')
def admin_dashboard():
    """Route untuk menampilkan Panel Kontrol Admin (Hanya bisa diakses jika
    sudah login)."""
    if not session.get('admin'): return redirect(url_for('login'))

    try:
        # Ambil data statistik dari MySQL
        attendance_list = system.get_todays_attendance()
        total_reg = system.get_total_users()
        users_list = system.get_all_users()

        from datetime import date
        str_date = date.today().strftime("%d %B %Y")

        return render_template('admin_dashboard.html',
                               nama=session['nama'],
                               data=attendance_list,
                               totalreg=total_reg,
                               tanggal=str_date,
                               users_data=users_list,
                               threshold=system.DISTANCE_THRESHOLD)
    except Exception as e:
        # Kembalikan error dengan format html sederhana jika dashboard gagal
        dimuat
        return f"<h3>Error memuat dashboard: {str(e)}</h3>", 500

@app.route('/api/train_model', methods=['POST'])

```

```

def api_train_model():
    if not session.get('admin'): return redirect(url_for('login'))
    try:
        system.train_model()
        flash("Model AI berhasil dilatih ulang! Wajah baru kini dapat dikenali.",
"success")
    except Exception as e:
        flash(f"Gagal melatih model: {str(e)}", "danger")
    return redirect(url_for('admin_dashboard'))

@app.route('/api/update_threshold', methods=['POST'])
def api_update_threshold():
    if not session.get('admin'): return redirect(url_for('login'))
    new_val = request.form.get('threshold', 4500)
    if system.set_threshold(new_val):
        flash(f"Threshold berhasil disimpan ke angka {new_val}!", "success")
    else:
        flash("Gagal menyimpan threshold.", "danger")
    return redirect(url_for('admin_dashboard'))

@app.route('/pca_plot.png')
def pca_plot():
    import os
    file_path = os.path.join(app.root_path, 'scatter_pca_skripsi.png')
    if os.path.exists(file_path):
        return send_file(file_path, mimetype='image/png')
    return "Not Found", 404

@app.route('/add_user', methods=['GET', 'POST'])
def add_user():

```

```
"""Route untuk mendaftarkan Mahasiswa baru dan memulai deteksi wajah
(Smart Capture)."""
```

```
if not session.get('admin'): return redirect(url_for('login'))
```

```
if request.method == 'POST':
```

```
    nama = request.form['nama']
```

```
    nim = request.form['nim'] # Digunakan sebagai username unik
```

```
    try:
```

```
        new_id = system.register_mahasiswa(nama, nim)
```

```
        if new_id:
```

```
            # Panggil Smart Capture
```

```
            system.capture_smart_ux(new_id, nama)
```

```
            return redirect(url_for('admin_dashboard'))
```

```
        else:
```

```
            # Modifikasi untuk menampilkan Pesan Error jika NIM sudah ada
            (Constraint UNIQUE Database)
```

```
            return "<h3>Error: NIM sudah terdaftar!</h3><br><a
href='/add_user'>Kembali</a>", 400
```

```
    except Exception as e:
```

```
        return f"<h3>Terjadi Kesalahan Pendaftaran: {str(e)}</h3>", 500
```

```
    return render_template('add_user.html')
```

```
@app.route('/delete/<int:user_id>')
```

```
def delete_user(user_id):
```

```
    """Route untuk menghapus mahasiswa beserta data wajah dan riwayat
    presensinya."""
```

```
    if not session.get('admin'): return redirect(url_for('login'))
```

```
    system.delete_user(user_id)
```

```
    return redirect(url_for('admin_dashboard'))
```

```

@app.route('/update_face/<int:user_id>')
def update_face(user_id):
    """Route untuk memperbarui data wajah mahasiswa yang sudah ada tanpa
    menghapus akun."""
    if not session.get('admin'): return redirect(url_for('login'))

    try:
        success = system.retrain_user_face(user_id)
        if success:
            return redirect(url_for('admin_dashboard'))
        else:
            return "<h3>User tidak ditemukan!</h3>", 404
    except Exception as e:
        return f"<h3>Terjadi Kesalahan Update Wajah: {str(e)}</h3>", 500

@app.route('/export_attendance')
def export_attendance():
    """Route untuk mengunduh riwayat presensi hari ini ke format Excel."""
    if not session.get('admin'): return redirect(url_for('login'))

    try:
        excel_file = system.get_excel_file()
        if excel_file:
            from flask import send_file
            return send_file(excel_file, download_name='laporan_kehadiran.xlsx',
as_attachment=True)
        return "Gagal membuat laporan", 500
    except Exception as e:
        return f"<h3>Terjadi Kesalahan Server: {str(e)}</h3>", 500

```

```

@app.route('/export_users')
def export_users():
    if not session.get('admin'): return redirect(url_for('login'))

    conn = system._get_db()
    if conn:
        import pandas as pd
        import io
        from flask import send_file
        try:
            df = pd.read_sql("SELECT id, username as NIM, nama_lengkap as
Nama FROM users WHERE role='mahasiswa'", conn)
            output = io.BytesIO()
            with pd.ExcelWriter(output, engine='openpyxl') as writer:
                df.to_excel(writer, index=False)
            output.seek(0)
            return send_file(output, download_name='daftar_mahasiswa.xlsx',
as_attachment=True)
        except Exception as e:
            print(f"Error exporting users: {e}")
            return "Gagal membuat laporan mahasiswa", 500
        finally:
            conn.close()
    return "Koneksi database gagal", 500

# --- ROUTES VIDEO STREAMING ---
def gen_frames():
    cap = cv2.VideoCapture(0)
    while True:
        success, frame = cap.read()
        if not success: break

```

```
frame = system.process_frame(frame)

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')

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

Lampiran 2 *Code pipeline_core.py*

```

import cv2
import os
import shutil
import threading
import pytsx3
import numpy as np
import joblib
import time
import mysql.connector
import io
import pandas as pd
from datetime import datetime, date
import re
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
from werkzeug.security import generate_password_hash, check_password_hash
from visualisasi_knn import generate_pca_plot

class FaceRecognitionPipeline:
    def __init__(self):
        # --- KONFIGURASI DINAMIS ---
        self.FACE_SIZE = (100, 100)

        # Coba muat threshold dari config.json, jika gagal gunakan default 4500
        self.config_path = 'config.json'
        self.DISTANCE_THRESHOLD = 4500
        try:
            import json
            if os.path.exists(self.config_path):
                with open(self.config_path, 'r') as f:

```

```

        data = json.load(f)
        if 'DISTANCE_THRESHOLD' in data:
            self.DISTANCE_THRESHOLD =
int(data['DISTANCE_THRESHOLD'])
    except Exception as e:
        print(f"Gagal memuat config.json: {e}")

    self.N_IMGS = 100
    self.process_every_n_frames = 3 # Optimasi: Proses 1 dari tiap 3 frame
    self.frame_count = 0
    self.last_detected_faces = [] # Cache untuk wajah yang terdeteksi

    # Konfigurasi Database
    self.db_config = {
        'host': 'localhost',
        'user': 'root',
        'password': "",
        'database': 'skripsi_presensi'
    }

    self._init_folders()
    self._init_db()

    # Load AI
    self.face_detector =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    self.ey_detector = cv2.CascadeClassifier('haarcascade_eye.xml') #
Tambahan detektor mata
    self.model = None
    self.pca = None
    self.load_model()

```

```

self.user_cache = self.get_user_map()
self.attendance_cache = set()

def set_threshold(self, new_val):
    try:
        self.DISTANCE_THRESHOLD = int(new_val)
        import json
        with open(self.config_path, 'w') as f:
            json.dump({'DISTANCE_THRESHOLD':
self.DISTANCE_THRESHOLD}, f)
        print(f"Threshold berhasil diubah ke
{self.DISTANCE_THRESHOLD}")
        return True
    except Exception as e:
        print(f"Error set threshold: {e}")
        return False

# --- DATABASE MANAGER ---
def _get_db(self):
    try: return mysql.connector.connect(**self.db_config)
    except: return None

def _init_db(self):
    conn = self._get_db()
    if not conn: return
    c = conn.cursor()
    c.execute("CREATE TABLE IF NOT EXISTS users (
                id INT AUTO_INCREMENT PRIMARY KEY, username
VARCHAR(50) UNIQUE,

```

```

        password VARCHAR(255), nama_lengkap VARCHAR(100), role
        VARCHAR(20))")

        c.execute("CREATE TABLE IF NOT EXISTS absensi (
            id INT AUTO_INCREMENT PRIMARY KEY, user_id INT,
            nama_lengkap VARCHAR(100),
            tanggal DATE, waktu TIME, status VARCHAR(20),
            FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
            CASCADE)")

        c.execute("SELECT * FROM users WHERE username='admin'")
        if not c.fetchone():
            pass_hash = generate_password_hash('admin123')
            c.execute("INSERT INTO users (username, password, nama_lengkap,
            role) VALUES (%s, %s, %s, %s)",
                ('admin', pass_hash, 'Dosen Administrator', 'admin'))
            conn.commit()
            conn.close()

        def get_user_map(self):
            conn = self._get_db()
            if not conn: return {}
            c = conn.cursor(dictionary=True)
            c.execute("SELECT id, nama_lengkap FROM users")
            res = {row['id']: row['nama_lengkap'] for row in c.fetchall()}
            conn.close()
            return res

        # --- AUTHENTICATION ---
        def login_admin(self, username, password):
            conn = self._get_db()
            if not conn: raise Exception("Koneksi Database Gagal. Pastikan
            MySQL/XAMPP menyala.")

```

```

try:
    c = conn.cursor(dictionary=True)
    c.execute("SELECT * FROM users WHERE username=%s AND
role='admin'", (username,))
    user = c.fetchone()
    if user and check_password_hash(user['password'], password): return
user
    return None
finally:
    conn.close()

def register_dosen(self, nama, username, password):
    # Validasi Password Ketat
    if len(password) < 8:
        return "Password minimal 8 karakter!"
    if not re.search(r"[A-Z]", password):
        return "Password harus mengandung minimal 1 huruf kapital!"
    if not re.search(r"[a-z]", password):
        return "Password harus mengandung minimal 1 huruf kecil!"
    if not re.search(r"[0-9]", password):
        return "Password harus mengandung minimal 1 angka!"
    if not re.search(r"[!@#$%^&*(),.?\":{}|<>]", password):
        return "Password harus mengandung minimal 1 simbol khusus
(!@#$...)!"

    conn = self._get_db()
    if not conn: return "Koneksi Database Gagal. Pastikan MySQL/XAMPP
menyala."
    try:
        c = conn.cursor()

```

```

        c.execute("SELECT * FROM users WHERE username=%s",
(username,))

        if c.fetchone(): return "Username sudah digunakan!"

        pass_hash = generate_password_hash(password)

        c.execute("INSERT INTO users (username, password, nama_lengkap,
role) VALUES (%s, %s, %s, %s)",

                (username, pass_hash, nama, 'admin'))

        conn.commit()
        return True
    except Exception as e:
        return f"Error: {str(e)}"
    finally:
        conn.close()

def register_mahasiswa(self, nama, nim):
    conn = self._get_db()
    if not conn: return None
    try:
        c = conn.cursor()

        c.execute("INSERT INTO users (username, password, nama_lengkap,
role) VALUES (%s, %s, %s, %s)",

                (nim, 'mahasiswa123', nama, 'mahasiswa'))

        conn.commit()
        new_id = c.lastrowid
        self.user_cache[new_id] = nama
        return new_id
    except:
        return None
    finally:
        conn.close()

```

```

def delete_user(self, user_id):
    conn = self._get_db()
    if not conn: return False
    c = conn.cursor()
    nama = self.user_cache.get(int(user_id))
    c.execute("DELETE FROM users WHERE id=%s", (user_id,))
    conn.commit(); conn.close()
    if int(user_id) in self.user_cache: del self.user_cache[int(user_id)]
    if nama:
        folder = f'static/faces/{nama.replace(" ", "_")}_{user_id}'
        if os.path.exists(folder): shutil.rmtree(folder)
    self.train_model()
    return True

def retrain_user_face(self, user_id):
    """Memicu ulang pengambilan data wajah tanpa menghapus akun user"""
    nama = self.user_cache.get(int(user_id))
    if not nama: return False

    folder = f'static/faces/{nama.replace(" ", "_")}_{user_id}'
    # Hapus dataset foto wajah lama jika ada
    if os.path.exists(folder):
        shutil.rmtree(folder)

    # Panggil jendela capture UI pintar
    self.capture_smart_ux(user_id, nama)
    return True

# --- AI LOGIC (ADAPTASI DARI APP_1 / PIPELINE_CORE_1) ---
def _init_folders(self):
    if not os.path.exists('static/faces'): os.makedirs('static/faces')

```

```

if not os.path.exists('static/unknown_faces'):
os.makedirs('static/unknown_faces') # Folder Log Orang Asing

def load_model(self):
    if os.path.exists('static/face_recognition_model.pkl'):
        self.model = joblib.load('static/face_recognition_model.pkl')
    if os.path.exists('static/face_pca_model.pkl'):
        self.pca = joblib.load('static/face_pca_model.pkl')

def speak(self, text):
    def _run():
        try:
            from gtts import gTTS
            import pygame
            import io

            # Buat audio di memori agar cepat (tidak perlu simpan file fisik)
            tts = gTTS(text=text, lang='id')
            fp = io.BytesIO()
            tts.write_to_fp(fp)
            fp.seek(0)

            # Mainkan audio menggunakan pygame
            pygame.mixer.init()
            pygame.mixer.music.load(fp)
            pygame.mixer.music.play()
            while pygame.mixer.music.get_busy():
                pygame.time.Clock().tick(10)
        except Exception as e:
            print(f"[TTS Error] Gagal memutar suara: {e}")
    threading.Thread(target=_run, daemon=True).start()

```

```

# === [UPDATE UTAMA] LOGIKA PENGAMBILAN GAMBAR CITRA
ASLI -> PROSES ===

def capture_smart_ux(self, user_id, nama):
    folder = f'static/faces/{nama.replace(" ", "_")}_{user_id}'
    if not os.path.exists(folder): os.makedirs(folder)

    cap = cv2.VideoCapture(0)
    i = 0
    last_time = time.time()

    # Setup Video Backup
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    video_path = os.path.join(folder, f'video_backup_{nama}.avi')
    out = cv2.VideoWriter(video_path, cv2.VideoWriter_fourcc(*'MJPG'),
    20.0, (width, height))

    sample_bgr = None # Untuk laporan visualisasi

    while True:
        ret, frame = cap.read()
        if not ret: break
        out.write(frame) # Rekam Video

    # Instruksi Smart UX
    if i < 20: msg, col = "LIHAT LURUS", (0,255,0)
    elif i < 40: msg, col = "TENGOK KIRI", (255,255,0)
    elif i < 60: msg, col = "TENGOK KANAN", (0,255,255)
    elif i < 80: msg, col = "DONGAK / TUNDUK", (255,0,255)
    else: msg, col = "SENYUM", (255,255,255)

```

```

# Deteksi Wajah dengan Skala Frame (Optimasi 1)
# Mengecilkan frame untuk deteksi mempercepat proses CPU secara signifikan
small_frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
gray = cv2.cvtColor(small_frame, cv2.COLOR_BGR2GRAY)
faces = self.face_detector.detectMultiScale(gray, scaleFactor=1.2,
minNeighbors=5, minSize=(40, 40))

# Tampilan UI
cv2.rectangle(frame, (0,0), (640, 80), (0,0,0), -1)
cv2.putText(frame, f'Progress: {i}/{self.N_IMGS}', (20,35),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,255), 2)
cv2.putText(frame, msg, (20, 70), cv2.FONT_HERSHEY_SIMPLEX,
0.8, col, 2)

for (x,y,w,h) in faces:
    # Mengembalikan box ke skala asli
    x, y, w, h = x*2, y*2, w*2, h*2

# OPTIMASI 3: Crop Lebih ketat (Tight Bounding Box)
# Kurangi 10% dari tiap sisi agar background dan telinga/rambut
minim masuk
pad_w, pad_h = int(w*0.1), int(h*0.1)
tx, ty, tw, th = x+pad_w, y+pad_h, w-(2*pad_w), h-(2*pad_h)

# OPTIMASI 2: Validasi Ganda Mata
# Cari mata HANYA di setengah bagian atas wajah
# Ambil dari grayscale full resolution untuk deteksi mata yang akurat
gray_full = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
roi_gray_mata = gray_full[ty:ty+int(th/2), tx:tx+tw]

```

```

mata = self.eye_detector.detectMultiScale(roi_gray_mata,
scaleFactor=1.1, minNeighbors=3, minSize=(15, 15))

mata_terdeteksi = len(mata) >= 1 # Minimal ada 1 mata yang terlihat

if time.time() - last_time > 0.25 and mata_terdeteksi: # Delay capture
dan Validasi

# 1. AMBIL CITRA ASLI Wajah (RGB) dengan ukuran ketat
face_raw = frame[ty:ty+th, tx:tx+tw]

# 2. SIMPAN CITRA ASLI
cv2.imwrite(f"{folder}/{i}.jpg", face_raw)

# Simpan sampel untuk laporan (foto ke-50)
if i == 50: sample_bgr = face_raw

i += 1
last_time = time.time()
cv2.rectangle(frame, (tx,ty), (tx+tw, ty+th), (0,255,0), 3)
else:
# Warna kotak merah/biru jika gagal deteksi mata atau masih delay
warn_col = (255,0,0) if not mata_terdeteksi else (0,165,255)
cv2.rectangle(frame, (tx,ty), (tx+tw, ty+th), warn_col, 2)

cv2.imshow("Registrasi Wajah - Server", frame)
if i >= self.N_IMGS or cv2.waitKey(1) == 27: break

cap.release()
out.release()
cv2.destroyAllWindows()

```

```

# Buat Gambar Laporan Visualisasi
if sample_bgr is not None:
    self._save_report_images(sample_bgr, folder)

self.train_model()

def _save_report_images(self, img_bgr, folder_path):
    """Membuat grafik perbandingan visualisasi preprocessing"""
    try:
        import matplotlib.pyplot as plt
        img_bgr = cv2.resize(img_bgr, (200, 200))

        # Simpan file fisik tahapan
        cv2.imwrite(os.path.join(folder_path, "1_original.jpg"), img_bgr)
        gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
        cv2.imwrite(os.path.join(folder_path, "2_grayscale.jpg"), gray)
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
        eq = clahe.apply(gray)
        cv2.imwrite(os.path.join(folder_path, "3_equalized.jpg"), eq)

        # Buat Plot
        img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
        plt.figure(figsize=(10, 4))

        plt.subplot(1, 3, 1); plt.imshow(img_rgb); plt.title("1. Asli (RGB)");
plt.axis('off')

        plt.subplot(1, 3, 2); plt.imshow(gray, cmap='gray'); plt.title("2.
Grayscale"); plt.axis('off')

        plt.subplot(1, 3, 3); plt.imshow(eq, cmap='gray'); plt.title("3. Histogram
Eq"); plt.axis('off')

```

```

plt.tight_layout()
plt.savefig(os.path.join(folder_path, "hasil_visualisasi.png"))
plt.close()
except: pass

# == [UPDATE UTAMA] FILTER TRAINING ==
def train_model(self):
    print("[INFO] Training dimulai...")
    faces, labels = [], []

    # Perbarui cache setiap kali training berjalan agar data identitas pengguna
    baru selalu termuat
    self.user_cache = self.get_user_map()

    if not os.path.exists('static/faces'): return

    for user_folder in os.listdir('static/faces'):
        path = os.path.join('static/faces', user_folder)
        try: u_id = int(user_folder.split('_')[-1])
        except: continue

    for f in os.listdir(path):
        # Hanya memproses file gambar wajah (.jpg) dan mengabaikan file
        laporan/visualisasi
        skip_files = ['1_original.jpg', '2_grayscale.jpg', '3_equalized.jpg']
        if f.endswith('.jpg') and f not in skip_files:
            # Baca sebagai RGB Asli terlebih dahulu
            img = cv2.imread(os.path.join(path, f))
            if img is not None:
                # Preprocessing dilakukan SAAT TRAINING, agar dataset asli
                tetap terjaga

```

```

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
resized_face = cv2.resize(gray, self.FACE_SIZE)

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
eq_face = clahe.apply(resized_face)

# 1. Simpan gambar asli
faces.append(eq_face.ravel())
labels.append(u_id)

# 2. Data Augmentation (Horizontal Flip)
flipped = cv2.flip(eq_face, 1)
faces.append(flipped.ravel())
labels.append(u_id)

if faces:
    # 1. Terapkan PCA (Reduksi 10.000 dimensi -> ~400 komponen unik)
    # n_components minimal = jumlah wajah, atau maksimal 400 agar tidak
    membebani memori
    n_comp = min(len(faces), 400)
    pca = PCA(n_components=n_comp, whiten=True)
    faces_pca = pca.fit_transform(np.array(faces))

    # 2. Latih KNN dengan data hasil PCA (Jarak menggunakan Cosine)
    knn = KNeighborsClassifier(n_neighbors=5, metric='cosine')
    knn.fit(faces_pca, labels)

    # 3. Simpan kedua Model (PCA & KNN)
    joblib.dump(pca, 'static/face_pca_model.pkl')
    joblib.dump(knn, 'static/face_recognition_model.pkl')

```

```

self.pca = pca
self.model = knn
print(f"[INFO] Training Selesai. Dimensi direduksi dari 10000 ->
{n_comp} dengan PCA.")

# 4. Refresh Model AI di Memory Utama agar siap dipakai proses
Recognition
self.load_model()

# 5. [BARU] Otomatis membuat ulang grafik PCA Scatter Plot untuk
Web
try:
    print("[INFO] Memperbarui Grafik PCA Scatter Plot...")
    generate_pca_plot('static/faces', self.user_cache)
    print("[INFO] Grafik berhasil diperbarui.")
except Exception as e:
    print(f"[ERROR] Gagal memperbarui grafik PCA: {e}")

# --- RECOGNITION (SAMA SEPERTI SEBELUMNYA) ---
def process_frame(self, frame):
    self.frame_count += 1

    # Ekstrak gray sekali di awal
    gray_full = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Optimasi Frame Skipping: Hanya deteksi wajah tiap n frame
    if self.frame_count % self.process_every_n_frames == 0:
        # Pengecilan frame untuk mempercepat Haar Cascade
        small_frame = cv2.resize(gray_full, (0, 0), fx=0.5, fy=0.5)
        # Karena skala 0.5, minSize juga harus setengahnya

```

```

faces = self.face_detector.detectMultiScale(small_frame,
scaleFactor=1.2, minNeighbors=5, minSize=(40, 40))

# Kembalikan koordinat wajah ke skala asli
self.last_detected_faces = [(x*2, y*2, w*2, h*2) for (x,y,w,h) in faces]

for (x,y,w,h) in self.last_detected_faces:
    name_display = "Unknown"
    col = (0,0,255)

# OPTIMASI 3: Crop Lebih ketat (Tight Bounding Box) saat
pengenalan
    pad_w, pad_h = int(w*0.1), int(h*0.1)
    tx, ty, tw, th = x+pad_w, y+pad_h, w-(2*pad_w), h-(2*pad_h)

    if self.model and self.pca:
        # 3. Potong area wajah SAJA secara ketat, barulah di-equalize agar
seragam dengan model
        # Menggunakan gray_full agar resolusinya tidak pecah
        face_crop = gray_full[ty:ty+th, tx:tx+tw]

        # Validasi array kosong sebelum resize untuk mencegah error
cv2.error
        if face_crop.size == 0: continue

        face_resized = cv2.resize(face_crop, self.FACE_SIZE)
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
        face_eq = clahe.apply(face_resized)

        # 4. Prediksi dengan PCA -> KNN
        vec_raw = face_eq.ravel().reshape(1,-1)

```

```

vec_pca = self.pca.transform(vec_raw) # Terapkan filter dimensi

dist, _ = self.model.kneighbors(vec_pca, n_neighbors=1)

# Jarak Threshold untuk Cosine (0.0=Sangat mirip, 1.0=Berbeda
jauh)

# Mengubah input Web (misal: 4500) menjadi skala Cosine (0.45)
threshold_cosine = self.DISTANCE_THRESHOLD / 10000.0

# Hitung Persentase Kecocokan (Cosine Similarity)
# Cosine distance: 0 = 100% mirip, 1 = 0% mirip
dist_val = dist[0][0]
similarity_pct = max(0.0, (1.0 - dist_val) * 100.0)

if dist_val < threshold_cosine:
    pred_id = self.model.predict(vec_pca)[0] # Fixed: vec to vec_pca
    name = self.user_cache.get(pred_id, "Unknown")
    if name != "Unknown":
        name_display = f"{name} ({similarity_pct:.1f}%)"
        col = (0,255,0)
        self._log_attendance(pred_id, name)
    else:
        name_display = f"Unknown ({similarity_pct:.1f}%)"
else:
    name_display = f"Unknown ({similarity_pct:.1f}%)"
# Logika Pencatatan Akses Keamanan (Orang Asing)
# Jika wajah tidak cocok, beri label Unknown
# Kita membuat mekanisme delay sederhana berdasarkan waktu
sistem

# untuk menghindari spamming simpan foto

```

```

        if not hasattr(self, 'unknown_timer'): self.unknown_timer =
time.time()

        if time.time() - self.unknown_timer > 3.0: # Jika tertahan di kamera
3 detik

            self._log_unknown_face(frame[ty:ty+th, tx:tx+tw])
            self.unknown_timer = time.time()

        cv2.rectangle(frame, (x,y), (x+w, y+h), col, 2)
        cv2.putText(frame, name_display, (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, col, 2)

    return frame

def _log_unknown_face(self, face_bgr):
    # Simpan jejak foto orang asing ke folder
    now_time = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f'static/unknown_faces/unknown_{now_time}.jpg'
    cv2.imwrite(filename, face_bgr)

    # Jika ada tabel database khusus log keamanan, query INSERT bisa
diletakkan disini.

    print(f"[KEAMANAN] Terdeteksi wajah tak dikenal! ({filename})")

def _log_attendance(self, user_id, nama):
    today = date.today().strftime("%Y-%m-%d")
    now_time_obj = datetime.now()
    now_time_str = now_time_obj.strftime("%H:%M:%S")
    now_hour = now_time_obj.hour

    # Aturan Batas Waktu Absen Skripsi
    # 00:00 - 08:00 = Hadir

```

```

# 08:00 - 09:00 = Terlambat
# > 09:00 = Ditolak (Sistem tutup)
if now_hour < 8:
    status_absen = 'Hadir'
elif 8 <= now_hour < 9:
    status_absen = 'Terlambat'
else:
    # Jam absen habis, jangan catat ke database, hanya ucapkan peringatan
    self.speak(f'Maaf {nama}, Jam Absen telah berakhir.')
    return

cache_key = f'{user_id}_{today}'
if cache_key in self.attendance_cache: return

conn = self._get_db()
if not conn: return
try:
    c = conn.cursor()
    c.execute("SELECT * FROM absensi WHERE user_id=%s AND
tanggal=%s", (user_id, today))
    if not c.fetchone():
        c.execute("INSERT INTO absensi (user_id, nama_lengkap, tanggal,
waktu, status) VALUES (%s, %s, %s, %s, %s)",
            (user_id, nama, today, now_time_str, status_absen))
    conn.commit()
    if status_absen == 'Hadir':
        self.speak(f'Selamat Datang {nama}')
    else:
        self.speak(f'{nama}, Anda Terlambat')
    self.attendance_cache.add(cache_key)
finally:

```

```

        conn.close()

    def get_todays_attendance(self):
        conn = self._get_db()
        if not conn: return []
        try:
            c = conn.cursor(dictionary=True)
            today = date.today().strftime("%Y-%m-%d")
            c.execute("SELECT * FROM absensi WHERE tanggal=%s ORDER
BY waktu DESC", (today,))
            res = c.fetchall()
            return res
        finally:
            conn.close()

    def get_total_users(self):
        conn = self._get_db()
        if not conn: return 0
        try:
            c = conn.cursor()
            c.execute("SELECT COUNT(*) FROM users WHERE
role='mahasiswa'")
            res = c.fetchone()[0]
            return res
        finally:
            conn.close()

    def get_all_users(self):
        conn = self._get_db()
        if not conn: return []
        try:

```

```

        c = conn.cursor(dictionary=True)
        c.execute("SELECT id, username, nama_lengkap FROM users WHERE
role='mahasiswa' ORDER BY nama_lengkap ASC")

        res = c.fetchall()

        return res
    finally:
        conn.close()

def get_attendance_history(self):
    conn = self._get_db()
    if not conn: return []
    try:
        c = conn.cursor(dictionary=True)
        c.execute("SELECT * FROM absensi ORDER BY id DESC")
        res = c.fetchall()
        return res
    finally:
        conn.close()

def get_excel_file(self):
    conn = self._get_db()
    if not conn: return None
    try:
        df = pd.read_sql("SELECT nama_lengkap, tanggal, waktu, status
FROM absensi ORDER BY tanggal DESC", conn)

        output = io.BytesIO()

        with pd.ExcelWriter(output, engine='openpyxl') as writer:
            df.to_excel(writer, index=False)

        output.seek(0)

        return output
    except Exception as e:

```

```
print(f'Excel generation error: {e}')
```

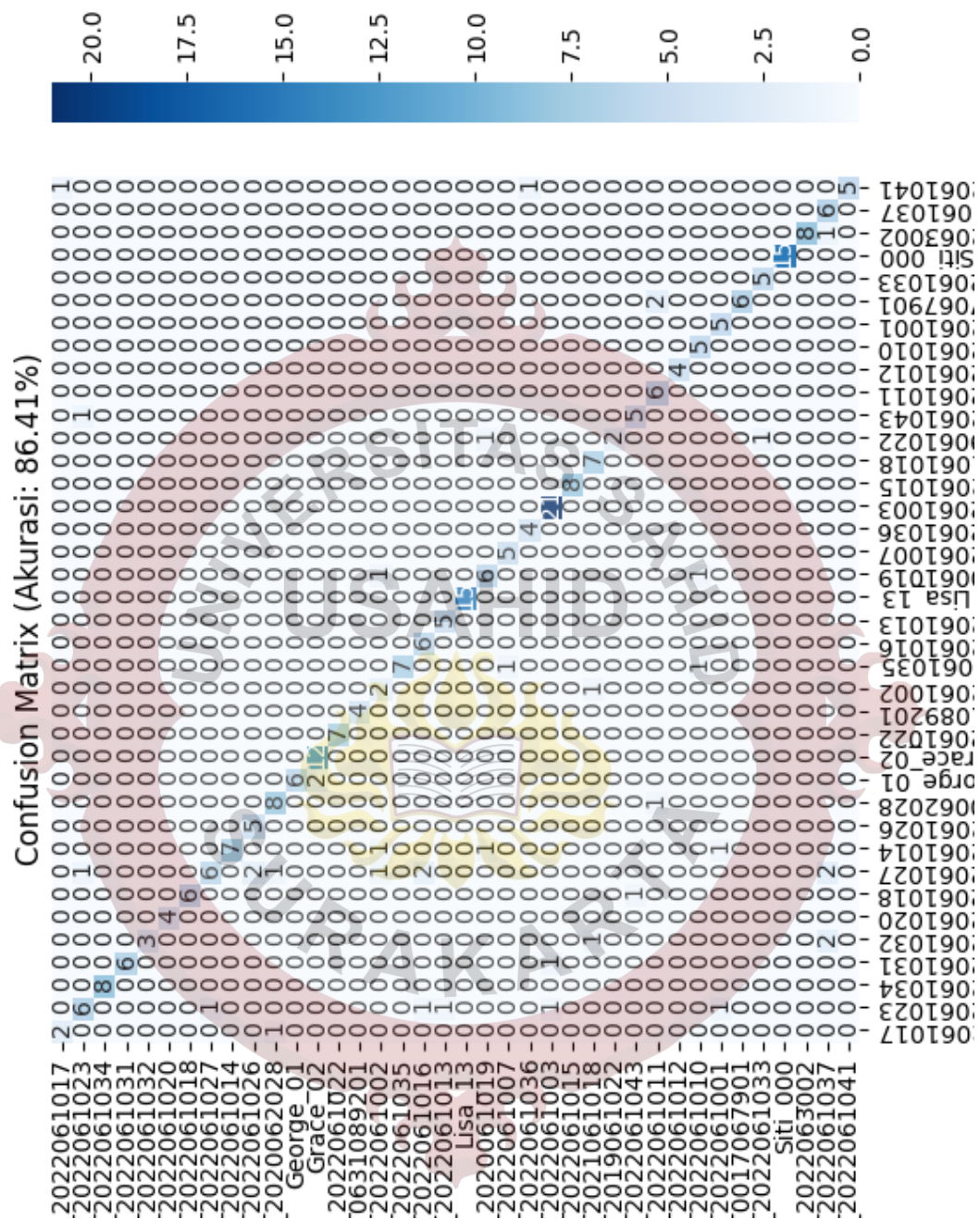
```
return None
```

```
finally:
```

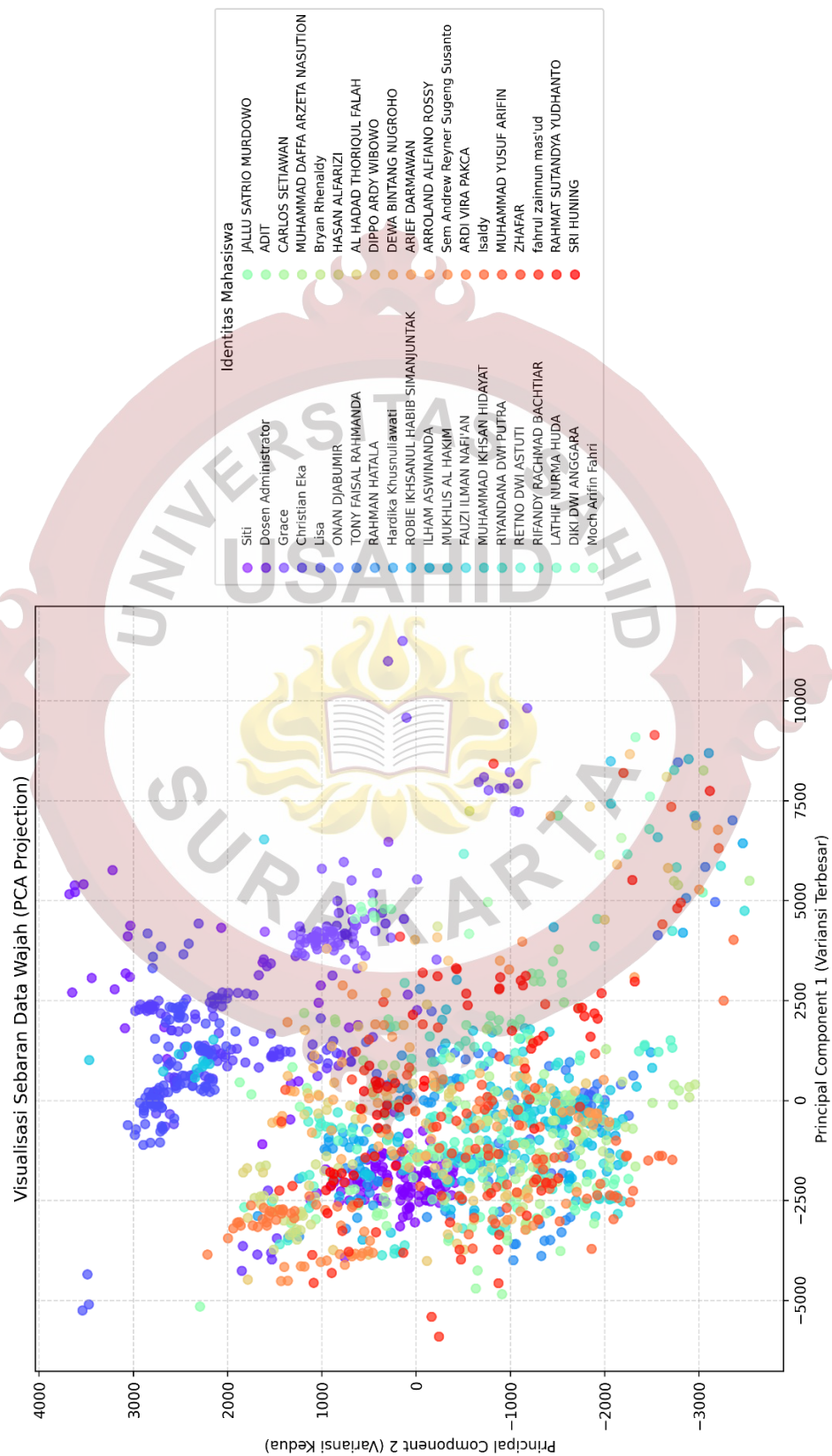
```
    conn.close()
```



Lampiran 3 Hasil Evaluasi Model



Lampiran 4 Hasil PCA Scatter



Lampiran 5 Contoh Dataset Wajah (Sample 30 Wajah per Orang)

Lampiran 6 Contoh Dataset Wajah (Sample 100 Wajah per Orang)



SURAT PENCATATAN CIPTAAN

Dalam rangka perlindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan : EC002026156427, 27 Agustus 2026

Pencipta

Nama : **Christian Eka Wibisono, Dr. Sri Huning Anwariningsih, S.T., M.Kom dkk**

Alamat : **Jl. Papandayan B-12 No. 4, RT.003/RW.008, Indramayu, Kab. Indramayu, Jawa Barat, 45212**

Kewarganegaraan : **Indonesia**

Pemegang Hak Cipta

Nama : **Universitas Sahid Surakarta**

Alamat : **Jl. Adi Sucipto No.154, Laweyan, Kota Surakarta, Jawa Tengah, 57144**

Kewarganegaraan : **Indonesia**

Jenis Ciptaan : **Program Komputer**

Judul Ciptaan : **FACE RECOGNITION PADA SISTEM PRESENSI BERBASIS WAJAH DENGAN METODE HAAR CASCADE**

Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia : **27 Agustus 2026, di Kota Surakarta**

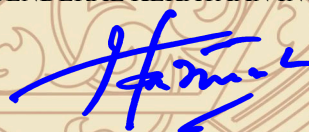
Jangka waktu perlindungan : **Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.**

Nomor Pencatatan : **001447492**

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.

Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

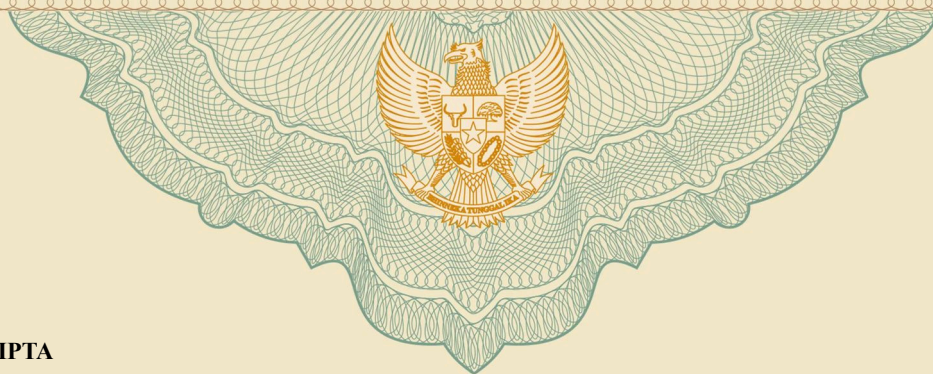
a.n. MENTERI HUKUM
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL



Dr. Hermansyah Siregar, S.H., M.H.
NIP. 197010161992031001



1. Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.
2. Surat Pencatatan ini telah disegel secara elektronik menggunakan segel elektronik yang diterbitkan oleh Balai Besar Sertifikasi Elektronik, Badan Siber dan Sandi Negara.
3. Surat Pencatatan ini dapat dibuktikan keasliannya dengan memindai kode QR pada dokumen ini dan informasi akan ditampilkan dalam browser.



LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Christian Eka Wibisono	JL. Papandayan B-12 No. 4, RT.003/RW.008 Indramayu, Kab. Indramayu
2	Dr. Sri Huning Anwariningsih, S.T., M.Kom	Gg. Sawo II/10, RT.001/RW.004, Laweyan, Kota Surakarta
3	Hardika Khusnuliawati, S.Kom., M.Kom	Cendono RT 01/ RW 07, Sugihan, Bendosari, Sukoharjo Bendosari, Kab. Sukoharjo



Disclaimer:

1. Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.
2. Surat Pencatatan ini telah disegel secara elektronik menggunakan segel elektronik yang diterbitkan oleh Balai Besar Sertifikasi Elektronik, Badan Siber dan Sandi Negara.
3. Surat Pencatatan ini dapat dibuktikan keasliannya dengan memindai kode QR pada dokumen ini dan informasi akan ditampilkan dalam browser.

