DAFTAR PUSTAKA

Ade Mubarok, Ivan Sofyan, Ali Akbar Rismayadi, & Ina Najiyah. (2018). Sistem Keamanan Rumah Menggunakan RFID, Sensor PIR dan Modul GSM Berbasis Mikrokontroller. *Jurnal Informatika*, *5*(1), 137–144.

Agus Purnama. (2022, October 2). *Limit Switch Dan Saklar Push ON*. ELEKTRONIKA DASAR. https://elektronika-dasar.web.id/limit-switch-dan-saklar-push-on/

Alam, R. G., Rozali, T., & Edo, W. P. (2015). Implementasi Algoritma Eigenface Untuk Face Recognition Pada Objeck Foto Id Card. *Telematik*, *7*(2), 1648–1656.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. *International Conference on Engineering and Technology (ICET)*, 1–6.

Amazon Web Services, I. (2022). *Apa itu API RESTful?* https://aws.amazon.com/id/what-is/restful-api/

Andri Nugraha Ramdhon, & Fadly Febriya. (2021). Penerapan Face Recognition Pada Sistem Presensi. *Journal of Applied Computer Science and Technology*, *2*(1), 12–17. https://doi.org/10.52158/jacost.v2i1.121

Apsari, R. J., & Prapanca, A. (2018). Monitoring Keamanan Rumah Dengan Menggunakan Mikrokontroler Melalui Web. *Jurnal Manajemen Informatika*, *8*(1), 87–95. www.electrodragon.com/product/nodemcu-

B. P. Statistik. (2021). *Statistik Kriminal 2021*.

Budihartato, W. (2018). *Elektronika Digital dan Sistem Embedded*. Andi Offset.

Dhanny, S., Andikha, D. P., Kezia, S., & Jenisa, F. (2021). Implementasi Convolutional Neural Network untuk Facial Recognition. *Media Informatika*, *20*(2), 66–79.

Eki Riyadani, M. (n.d.). *Sistem Keamanan Untuk Otorisasi Pada Smart Home Menggunakan Pengenalan Wajah Dengan Library OpenCV*.

Fauzan, A., Novamizanti, L., & Fuadah, Y. N. (2018). PERANCANGAN SISTEM DETEKSI WAJAH UNTUK PRESENSI KEHADIRAN MENGGUNAKAN METODE LBPH (Local Binary Pattern Histogram) BERBASIS ANDROID. *EProceedings of Engineering*, *5*(3), 5403–5412.

Hanafi, A., Sukarsa, I. M., & Agung Cahyawan Wiranatha, A. A. K. (2017). Pertukaran Data Antar *Database* Dengan Menggunakan Teknologi API. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, *8*(1), 22–30. https://doi.org/10.24843/lkjiti.2017.v08.i01.p03

Jurjawi, I. (2020). *Implementasi Pengenalan Wajah Secara Real Time untuk Sistem Absensi Menggunakan Metode Pembelajaran Deep Learning dengan Pustaka Open CV (Computer Vision)*.

Mundial, I. Q., Ul Hassan, M. S., Tiwana, M. I., Qureshi, W. S., & Alanazi, E. (2020). Towards facial recognition problem in COVID-19 pandemic. *2020 4th International Conference on Electrical, Telecommunication and Computer Engineering, ELTICOM 2020 - Proceedings*, 210–214. https://doi.org/10.1109/ELTICOM50775.2020.9230504

Pricop, E. (2019). *Biometrics the secret to securing industrial control systems*. https://ics.kasper-

Rama Mitra, A. (2021). Perancangan Aplikasi Sistem Pengenalan Wajah Dengan Metode Convolutional Neural Network (CNN) Untuk Pencatatan Kehadiran Karyawan. *Jurnal Instrumentasi Dan Teknologi Informatika (JITI)*, *3*(1), 1–11.

Satriadi, A., Wahyudi, W., & Christyono, Y. (2019). Perancangan home automation berbasis NodeMCU. *Transient: Jurnal Ilmiah Teknik Elektro*, *8*(1), 64–71. https://ejournal3.undip.ac.id/index.php/transient

Singgalen, R. (2017). Sistem Pengenalan Wajah sebagai Akses Loker Penyimpanan Barang. *Telekontran: Jurnal Ilmiah Telekomunikasi, Kendali Dan Elektronika Terapan*, *5*(2), 149–158.

Suhesti, T. (2014). *Bahasa Pemrograman Python*. DOCPLAYER. https://docplayer.info/49238665-Bahasa-pemrograman-python.html

LAMPIRAN

LAMPIRAN 1 Isi app.py

```
import os
from flask import Flask, render_template, Response, request, flash, session, url_for, redirect
from flask_socketio import SocketIO
from werkzeug.security import check_password_hash, generate_password_hash
from flask_mysqldb import MySQL
from core_service.facerecognition import Recognizer
import requests



app = Flask(__name__)
#koneksi
app.secret_key = 'bebasapasaja'
app.config['MYSQL_HOST'] ='localhost'
app.config['MYSQL_USER'] ='root'
app.config['MYSQL_PASSWORD'] =''
app.config['MYSQL_DB'] ='login_table'
mysql = MySQL(app)


socketio = SocketIO(app)


PATH = '\\'.join(os.path.abspath(__file__).split('\\')[0:-1])
recognizer = Recognizer(
    socketio=socketio,
    facerecognition_model = os.path.join(PATH, "core_Service\\bin\\frozen_graph_4.pb"),
    labels_filename=os.path.join(PATH, "core_Service\\labels.csv"),
    facedetection_model=os.path.join(PATH,
"core_Service\\bin\\haarcascade_frontalface_default.xml"),
    camera_src=0
```

```
)

@app.route("/")
def index():
    if 'loggedin' in session:
        camera = request.args.get("camera")
        data = requests.get('http://api.teknikaja.my.id/get-data').json()
        statUser = data[0]["state"]

        if camera is not None and camera == 'off':
            recognizer.close()
            flash("Camera turn off!", "info")
        elif camera is not None and camera == 'on':
            recognizer.open()
            flash("Camera turn on!", "success")
        print("camera status", recognizer.status())

        return render_template("index.html", status=statUser, is_camera = recognizer.status())

    flash('Harap Login dulu','danger')
    return redirect(url_for('login'))

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

```
        #cek username atau email
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM tb_users WHERE username=%s OR email=%s',(username, email))
        akun = cursor.fetchone()
        if akun is None:
            cursor.execute('INSERT INTO tb_users VALUES (NULL, %s, %s, %s)', (username, email, generate_password_hash(password)))
            mysql.connection.commit()
            flash('Registrasi Berhasil','success')
            return redirect(url_for('login'))
        else :
            flash('Username atau email sudah ada','danger')
    return render_template('registrasi.html')


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

        #cek data username
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM tb_users WHERE email=%s',(email, ))
        akun = cursor.fetchone()
        if akun is None:
            flash('Login Gagal, Cek Email Anda','danger')
        elif not check_password_hash(akun[3], password):
            flash('Login gagal, Cek Password Anda', 'danger')
        else:
```

```
            session['loggedin'] = True
            session['username'] = akun[1]
            return redirect(url_for('index'))
    return render_template('login.html')

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

@app.route("/history")
def history():
    username = "Muslimin"
    data = requests.get('http://api.teknikaja.my.id/get-history').json()
    return render_template("history.html", data=data, user=username)

@app.route("/tes")
def tes():
     username = "Muslimin"
     data = requests.get('http://api.teknikaja.my.id/get-history').json()
     return render_template("tes.html", data=data, user=username)

@app.route('/video_feed')
def video_feed():
    return Response(recognizer.gen_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
if __name__ == '__main__':
    socketio.run(app, debug=True)
```

LAMPIRAN 2 Isi facerecognition.py

```
import os
import cv2
from flask import redirect
import pandas as pd
import numpy as np
import datetime
import requests
import time, asyncio,threading
from threading import Timer
class Recognizer():
    def __init__(self,
                 socketio,
                 facerecognition_model = "frozen_graph_4.pb",
                 labels_filename="labels.csv",
                 facedetection_model="haarcascade_frontalface_default.xml",
                 camera_src=0):

        self.socketio = socketio

        if os.path.isfile(labels_filename) == False:
            raise Exception("Can't find %s" % labels_filename)

        self.labels = pd.read_csv(labels_filename)['0'].values

        self.camera_src = camera_src
        self.camera = None

        self.face_cascade = cv2.CascadeClassifier(facedetection_model)

        self.net = cv2.dnn.readNet(facerecognition_model)
```

```
        self.net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
        self.net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
        self.layerOutput = self.net.getUnconnectedOutLayersNames()

        self.curr_frame = None

        #self.label_stat = {}
        self.label_count = {}
        self.label_time = {}
        for name in self.labels:
            #self.label_stat[name] = False
            self.label_count[name] = 0
            self.label_time[name] = datetime.datetime.now()



    def predict(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = self.face_cascade.detectMultiScale(gray, 1.1, 5)

        for (x, y, w, h) in faces:
            face_img = gray[y:y+h, x:x+w]
            face_img = cv2.resize(face_img, (50, 50))

            blob = cv2.dnn.blobFromImage(face_img, 1.0, (50, 50), (0, 0, 0), swapRB=True,
crop=False)
            self.net.setInput(blob)
            output = self.net.forward(self.layerOutput)

            idx = output[0].argmax(axis=1)[0]
            confidence = output[0].max(axis=1)[0]*100
```

```
if confidence > 70:
    if self.labels[idx]=='Muslimin':

        curr_label = self.labels[idx]
        label_text = "%s" % (curr_label)
        if self.label_count[curr_label] > 5:
            self.socketio.emit("prediction", {
                                'frame' :self.get_curr_frame(),
                                'label' : curr_label,
                                #'status' : not self.label_stat[curr_label],
                                'time' : self.get_str_datetime()
                                })
            self.socketio.sleep(0.1)
            #self.label_stat[curr_label] = not self.label_stat[curr_label]
            self.label_time[curr_label] = datetime.datetime.now()
            self.label_count[curr_label] = 0
            r = requests.put('http://api.teknikaja.my.id/1', json={'state': 'OPEN'})
            r.status_code

        else :
            if self.check_diff_time(curr_label):
                self.label_count[curr_label] += 1
        #r = requests.put('http://127.0.0.1:5005/1', json={'state': 'OPEN'})
        #r.status_code

    else:
        label_text = "UNKNOWN"
else :
    # if time.sleep(200):
    #     self.camera.release()
    #     self.socketio.emit("prediction", {
```

```
#                                 'frame' :self.get_curr_frame(),
#                                 #'status' : not self.label_stat[curr_label],
#                                 'time' : self.get_str_datetime()
#                                 })

label_text = "UNKNOWN"



def hello():
    print ("hello, world")
    try:
        self.camera.release()
        self.socketio.emit("prediction", {
                                    'frame' :self.get_curr_frame(),
                                     'label' : "close",
                                    'time' : self.get_str_datetime()
                                       })
    except:
         print('ss')

Timer(10, hello).start()


        frame = self.draw_ped(frame, label_text, x, y, x + w, y + h, color=(0,255,255), text_color=(50,50,50))
    return frame
```

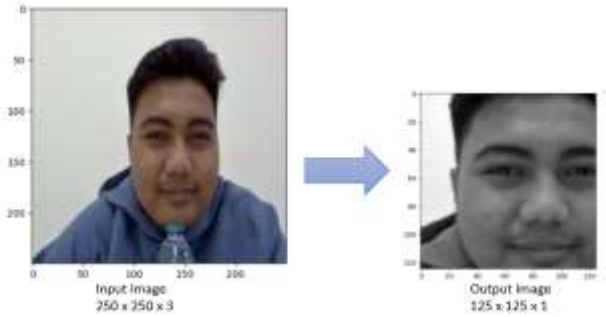
```
    def draw_ped(self, img, label, x0, y0, xt, yt, color=(255,127,0), text_color=(255,255,255)):
        (w, h), baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.8, 1)
        cv2.rectangle(img,
                      (x0, y0 + baseline),
                      (max(xt, x0 + w), yt),
                      color,
                      2)
        cv2.rectangle(img,
                      (x0, y0 - h),
                      (x0 + w, y0 + baseline),
                      color,
                      -1)
        cv2.putText(img,
                    label,
                    (x0, y0),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.8,
                    text_color,
                    1,
                    cv2.LINE_AA)
        return img


    def gen_frames(self):
        while True:
            if self.camera is None :
                self.open()
            success, frame = self.camera.read()
            if not success:
                break
            else:
```

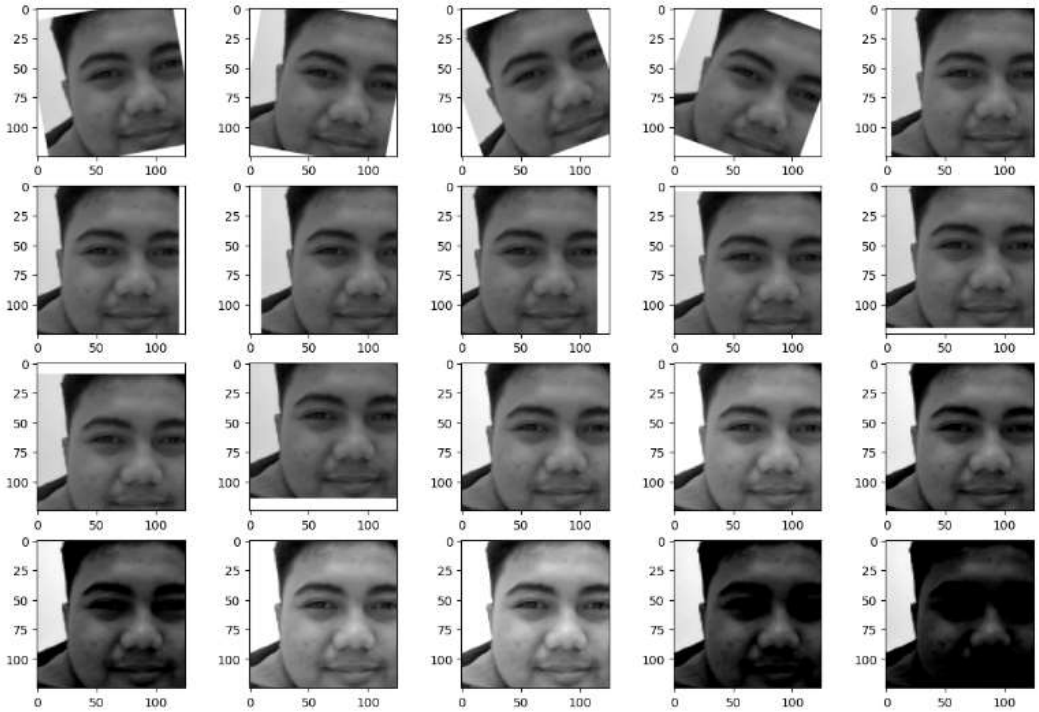
```
            try :
                self.curr_frame = frame.copy()
                frame = self.predict(frame)
            except Exception as e:
                print("[ERROR] ", e)
                self.camera.release()
                self.camera = None
                break
            ret, buffer = 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')

def close(self):
    if self.camera is not None :
        self.camera.release()
        self.camera = None

def open(self):
    self.camera = cv2.VideoCapture(self.camera_src)
    self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 480)
    self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 320)

def status(self):
    return self.camera is not None

def get_curr_frame(self):
    frame = cv2.resize(self.curr_frame, (0,0), fx=0.2, fy=0.2)
    ret, buffer = cv2.imencode('.png', frame)
    return buffer.tobytes()
```

```
def get_str_datetime(self):
    return datetime.datetime.now().strftime("%d/%m/%Y %H:%M:%S")

def check_diff_time(self, label):
    label_time = self.label_time[label]
    now = datetime.datetime.now()

    return now - label_time > datetime.timedelta(seconds=5)
```

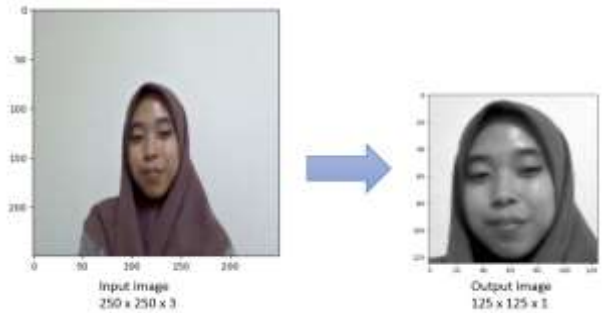
LAMPIRAN 3 Hasil preproses setiap kelas data

Amirul Muminin

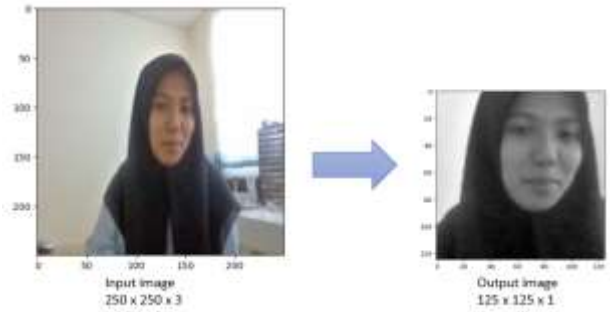




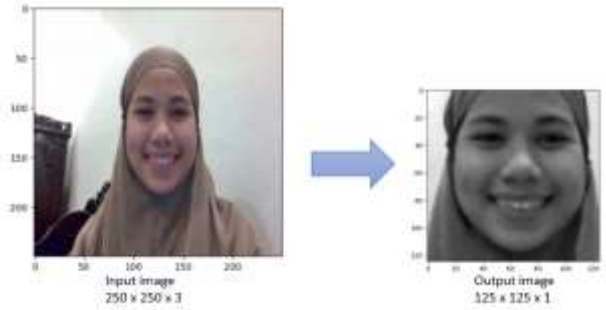
Fina

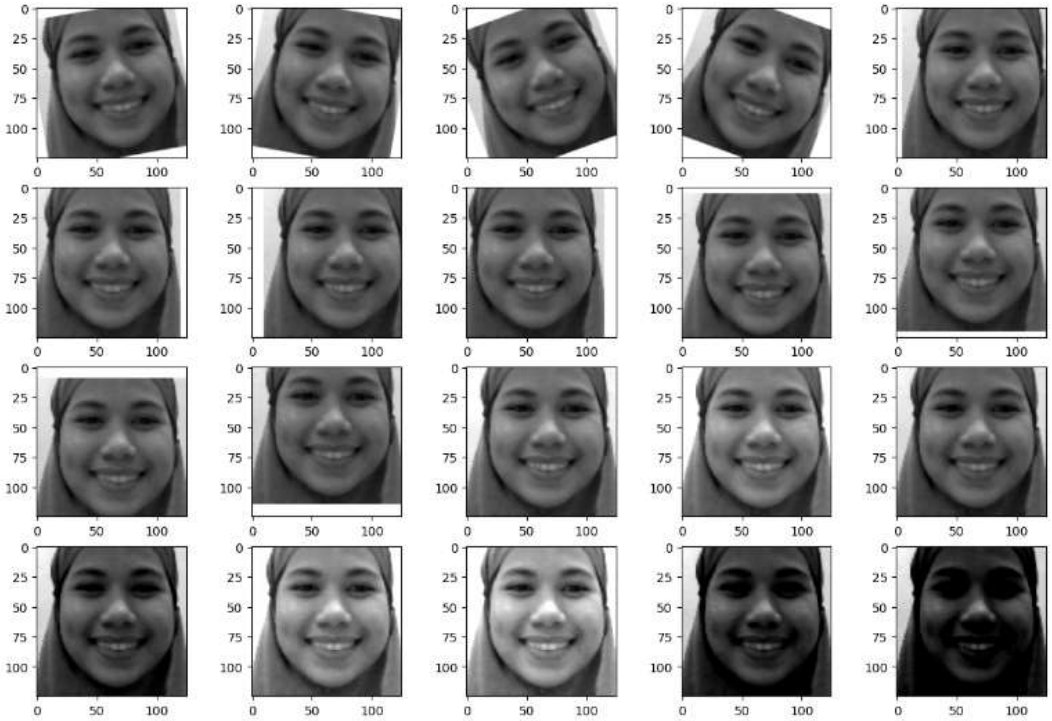


Isny

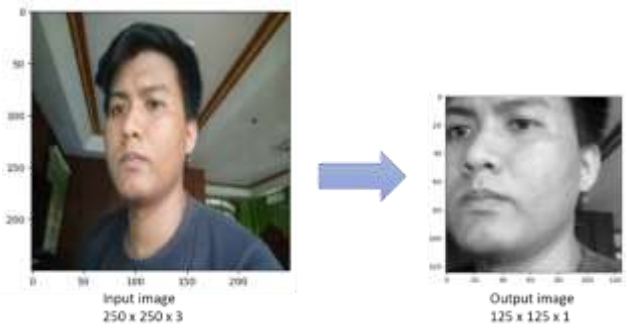


Kiky

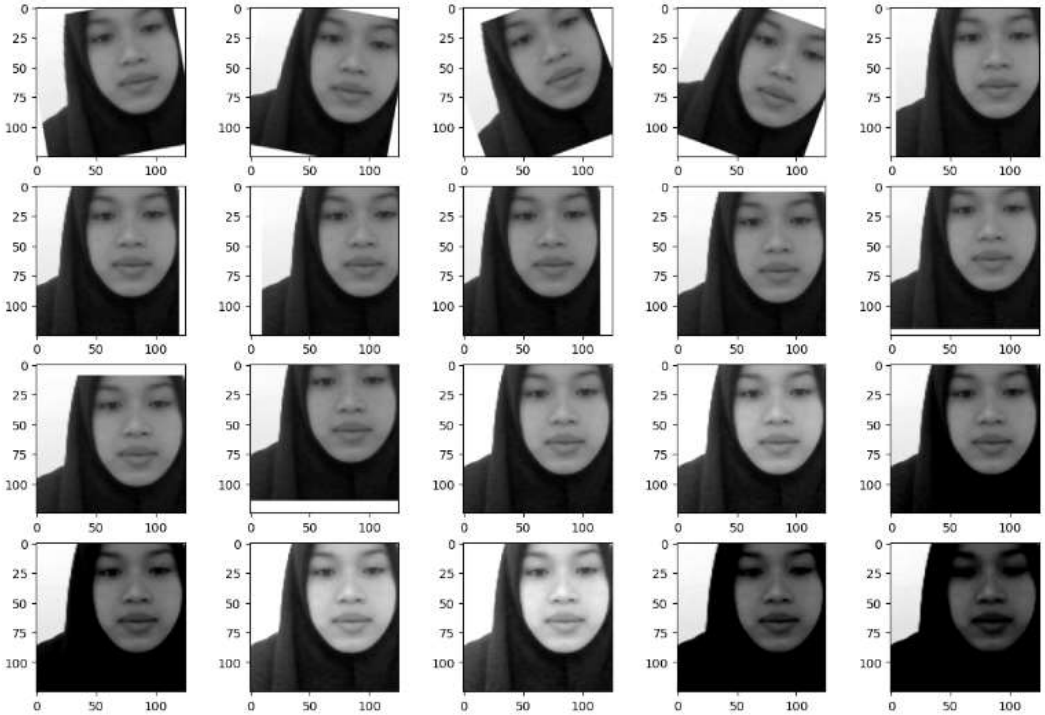




Muslimin

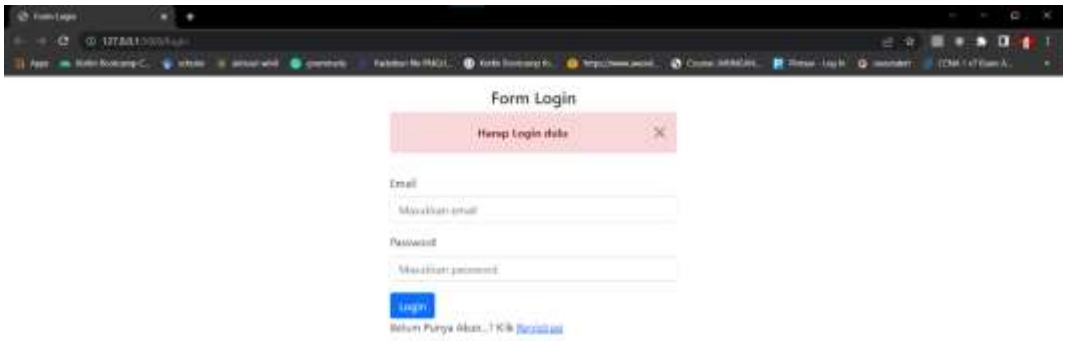


Risma

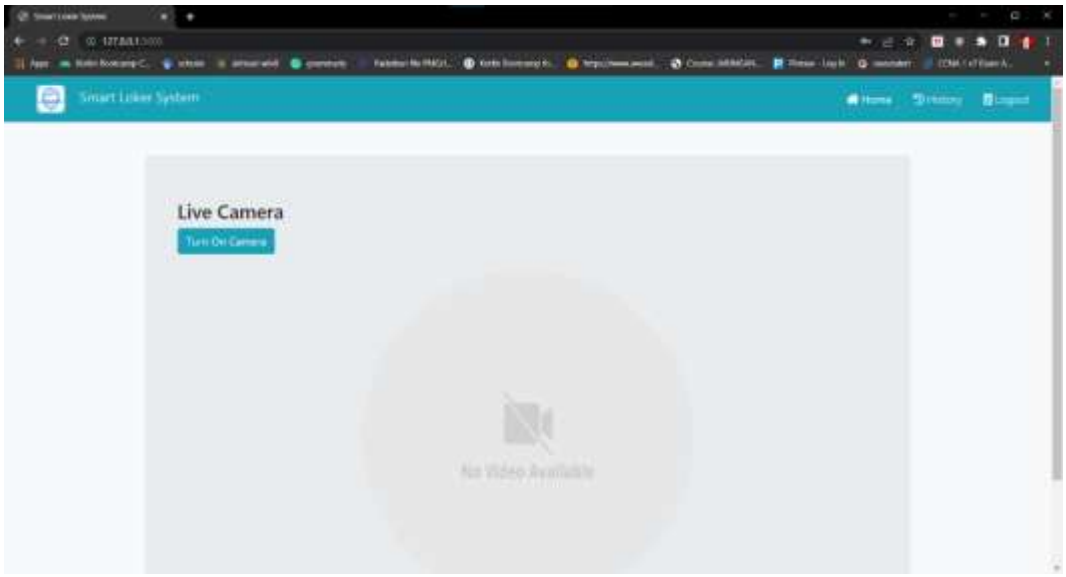


LAMPIRAN 4 Tampilan Pengujian Web

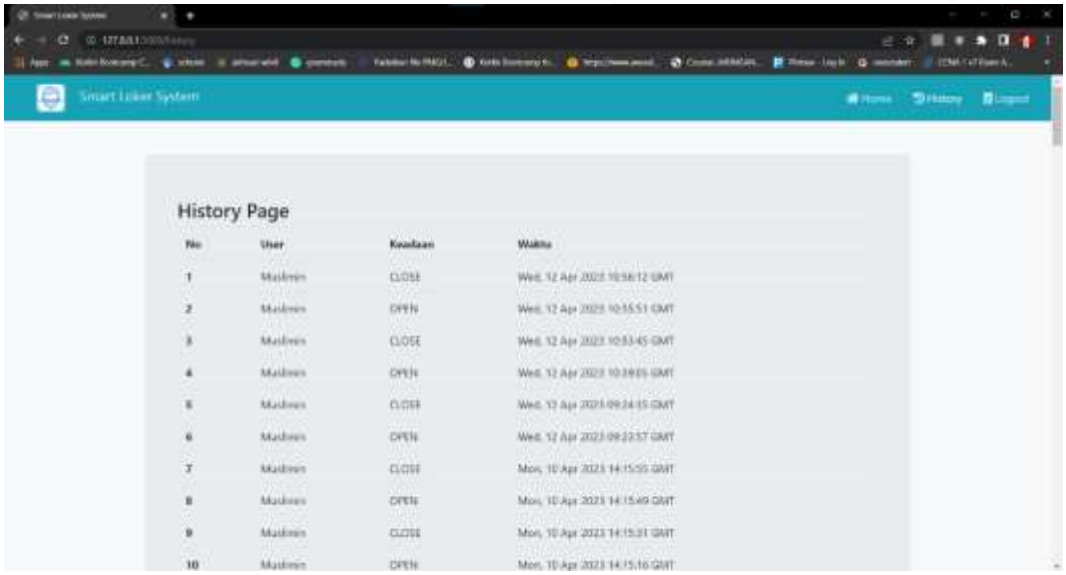
Halaman login



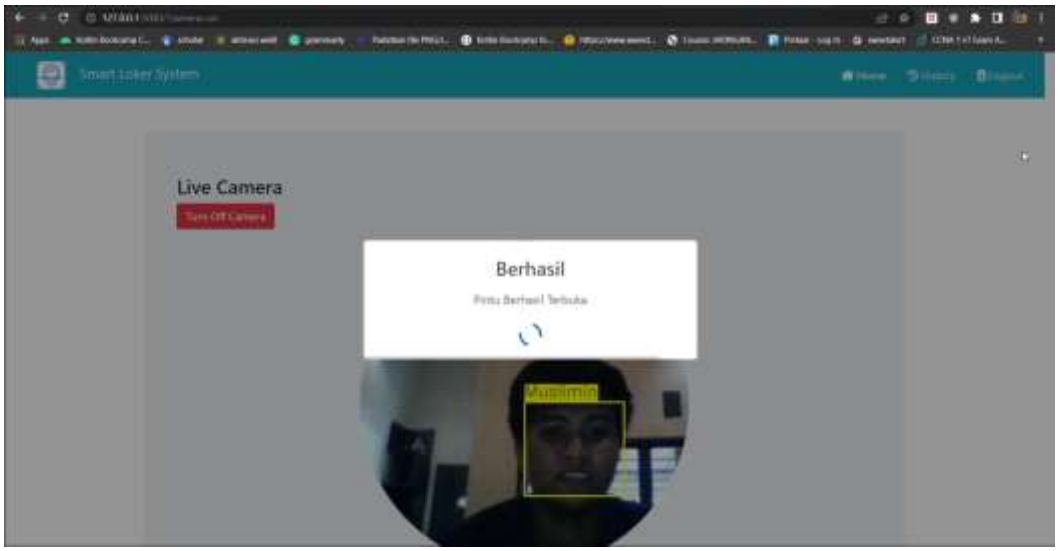
Halaman awal



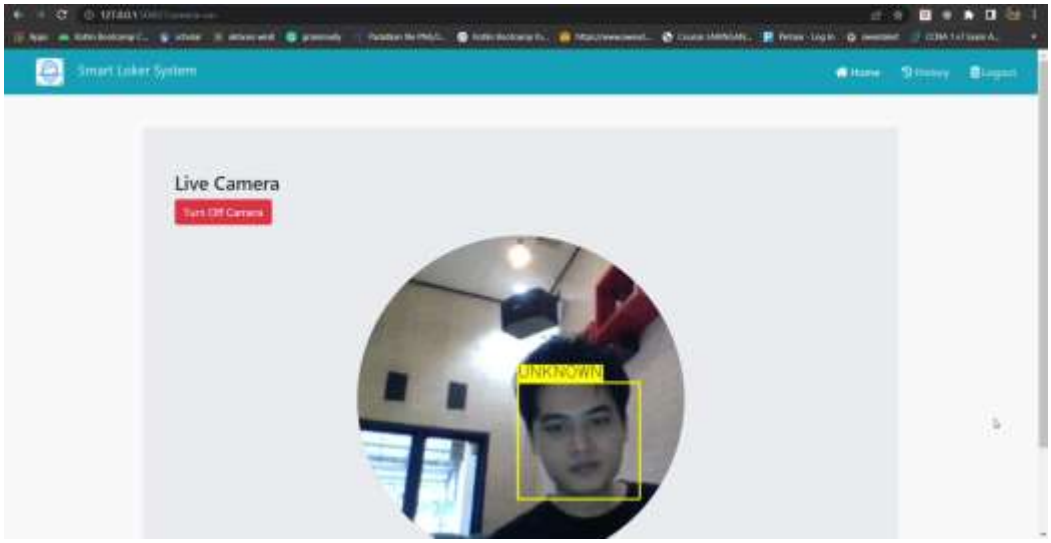
Halaman history



Kamera on dan wajah dikenali



Kamera on dan wajah tidak dikenali



Kamera on dan wajah tidak dikenali selama 10 detik