DAFTAR PUSTAKA

[3] Kalansuriya, T. R., & Dharmaratne, A. T. (2013). Facial Image Classification Based on Age and Gender, 44–50.

[4] Wulansari, D., Djamal, E. C., & Ilyas, R. (2017). Identifikasi Gender Berdasarkan Citra Wajah Menggunakan Deteksi Tepi dan Backpropagation, 10–14.

[5] Kalansuriya, T. R., & Dharmaratne, A. T. (2015). Neural Network based Age and Gender Classification for Facial Images, (April). https://doi.org/10.4038/icter.v7i2.7154

[6] Shinichi Terada. (2018). Firms strop goods as facial hair finds favor, 1–4.

[7] Sachdeva, S. (2018). HIRSUTISM : EVALUATION AND TREATMENT, 55(1), 3–7. https://doi.org/10.4103/0019-5154.60342

[8] Szeliski, R. 2010. Computer Vision: Algorithms and Applications. Springer

[9] Zhao, W. (2006). Face processing: advanced modeling and methods. Journal of Electronic ….

[10] Putra, Darma. 2010. Pengolahan Citra Digital. Yogyakarta: Andi.

[11] Mukhopadhyay, Jayanta. 2011. Image and Video Processing in the Compressed Domain. Chapman and Hall/CRC.

[12] Shulur dan Amalga, 2015. Perancangan Aplikasi Deteksi Wajah Menggunakan Algoritma Viola-Jones, Bandung: Universitas Pasundan.

[13] Li, Stan Z & Jain, Anil K. 2011. Handbook of Face Recognition. Springer

[14] Lopez, L.S. 2010. Local Binary Patterns applied to Face Detection and Recognition. Universitat Politecnica De Catalunya

[15] Prasetyo, E. 2014. Data Mining Mengubah Data Menjadi Informasi Menggunakan Matlab. Penerbit Andi. Yogyakarta.

[16] Prado, Kelvin Salton do. 2017. Face Recognition: Understanding LBPH Algorithm. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. diakses pada 11 Juli 2021

[17] python.org

[18] opencv.org

LAMPIRAN

- **Source code program**

a) Source code fungsi deteksi wajah untuk data latih

```
from threading import Thread
import cv2
import config
from ml.face_function import face_rotate
from ml.train_dataset import generate_face_xml

total_sample = config.TOTAL_SAMPLE


class GenerateDataset(Thread):
    def __init__(self, user_id, video_comp):
        super().__init__()

        self.sample_number = 0
        self.user_id = user_id
        self.video_comp = video_comp

    def run(self):
        detector_path = 'haarcascade_frontalface_default.xml'
        detector = cv2.CascadeClassifier(detector_path)
        sf = 1.1
        mn = 8

        while True:
            _, image = self.video_comp.vid.read()
            frame_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(frame_gray, scaleFactor=sf, minNeighbors=mn,
                                              minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)
```

```
            for (x, y, w, h) in faces:
                eye_gray = frame_gray[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
                img = (face_rotate(eye_gray))
                if img is not None:
                    frame = img
                    cv2.imwrite(
                        "training_data{0}/user.{1}.{2}.jpg".format(total_sample, str(self.user_id),
                                                                   str(self.sample_number)),
                        frame)
                    print("Ekstrak frame dengan wajah : {0} dari {1}".format(str(self.sample_number), total_sample))
                    self.sample_number = self.sample_number + 1
            if self.sample_number > total_sample:
                break
        print('--------------------')
        print('Ekstraksi selesai')
        generate_face_xml()
```

b) Source code fungsi rotasi wajah

```python
import cv2
import math


def face_rotate(image):
    haar = 'haarcascade_frontalface_default.xml'
    haar_eye = 'haarcascade_eye.xml'
    face_detector = cv2.CascadeClassifier(haar)
    eye_detector = cv2.CascadeClassifier(haar_eye)

    Theta = 0
    rows, cols = image.shape
    eyes = eye_detector.detectMultiScale(image)
    for (sx, sy, sw, sh) in eyes:
        # input berupa dua mata
        if eyes.shape[0] == 2:
            # menentukan mata kiri dan kanan
            if eyes[1][0] > eyes[0][0]:
                # beda tinggi di antara mata
                DY = ((eyes[1][1] + eyes[1][3] / 2) - (eyes[0][1] + eyes[0][3] / 2))
                # beda lebar antara mata
                DX = ((eyes[1][0] + eyes[1][2] / 2) - eyes[0][0] + (eyes[0][2] / 2))
            else:
                # beda tinggi antara mata
                DY = (-(eyes[1][1] + eyes[1][3] / 2) + (eyes[0][1] + eyes[0][3] / 2))
                # beda wajah antara mata
                DX = (-(eyes[1][0] + eyes[1][2] / 2) + eyes[0][0] + (eyes[0][2] / 2))
```

```python

            # memastikan rotasi jika diperlukan
            if (DX != 0.0) and (DY != 0.0):
                # perhitungan sudut euclidian
                Theta = math.degrees(math.atan(round(float(DY) / float(DX), 2)))
                print("Theta  ", str(Theta))

            M = cv2.getRotationMatrix2D((cols / 2, rows / 2), Theta, 1)  # menentukan matrix rotasi
            image = cv2.warpAffine(image, M, (cols, rows))
            # cv2.imshow('ROTATED', Image)                                        # rotasi wajah

            Face2 = face_detector.detectMultiScale(image, 1.1, 8)  # deteksi wajah
            for (FaceX, FaceY, FaceWidth, FaceHeight) in Face2:
                CroppedFace = image[FaceY + int(FaceHeight / 4.7): FaceY + int(FaceHeight - int(FaceHeight / 6.5)),
                              FaceX + int(FaceWidth / 4.7):FaceX + int(FaceWidth - int(FaceWidth / 4.7))]
                return CroppedFace
```

c) Source code fungsi latih model rekognisi wajah

```python
import os
import cv2
import numpy as np
from PIL import Image
import config


def get_image_label(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faceSamples = []
    faceIDs = []
    for imagePath in imagePaths:
        PILImg = Image.open(imagePath).convert('L')
        PILImg = PILImg.resize((54, 59))
        imgNum = np.array(PILImg, 'uint8')
        faceID = int(os.path.split(imagePath)[-1].split('.')[1])
        faceSamples.append(imgNum)
        faceIDs.append(faceID)

    return np.array(faceIDs), faceSamples



def generate_face_xml():
    training_path = "training_data{0}".format(config.TOTAL_SAMPLE)
    face_Recognizer = cv2.face.LBPHFaceRecognizer_create(1, 6, 8, 8)
    face_IDs, FaceSamples = get_image_label(training_path)

    print('TRAINING......')
    face_Recognizer.train(FaceSamples, face_IDs)
    face_Recognizer.save("training{0}.xml".format(config.TOTAL_SAMPLE))
    print('XML FILE SAVED....')

```

d) Source code GUI

```python
from enum import Enum
from datetime import datetime
import math

from statistics import mode
from tkinter import *
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
from reportlab.lib import colors
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
import csv
import cv2
import time

from db.crud import crud_user, crud_absensi
from db.models.absensi import AbsensiType
from db.session import get_db
from ml.generate_dataset import GenerateDataset
import config

total_sample = config.TOTAL_SAMPLE
haar = 'haarcascade_frontalface_default.xml'
haar_eye = 'haarcascade_eye.xml'
face_detector = cv2.CascadeClassifier(haar)
eye_detector = cv2.CascadeClassifier(haar_eye)
font = cv2.FONT_HERSHEY_SIMPLEX
```

```python
def get_list_absensi_user(user_id):
    with get_db() as db:
        return crud_absensi.get_list_absensi_by_user_id(db, user_id)


def get_list_absensi_date(date):
    with get_db() as db:
        return crud_absensi.get_list_absensi_by_date(db, date)


def get_user_absensi_today(user_id):
    with get_db() as db:
        return crud_absensi.get_absensi_today_by_user_id(db, user_id)


def add_absensi(absensi_type, user_id, curent_datetime):
    absensi_type = AbsensiType(absensi_type)
    with get_db() as db:
        dtstring_date = curent_datetime.strftime("%Y-%m-%d")
        dtstring_hour = curent_datetime.strftime("%H:%M:%S")
        crud_absensi.create_absensi(db, user_id=user_id, type=absensi_type, date=dtstring_date, time=dtstring_hour)


def get_all_users():
    with get_db() as db:
        return crud_user.get_list_users(db)
```

```python
def create_user(name):
    with get_db() as db:
        return crud_user.create_user(db, name=name)


def get_option_user_dicts():
    return {user.name: user.id for user in get_all_users()}


def get_list_user_names():
    return ["", *[user.name for user in get_all_users()]]


class ExportType(str, Enum):
    USER = "Nama Pegawai"
    DATE = "Tanggal"


```

```python
class App(Tk):
    def __init__(self, video_source):
        super().__init__()

        self.bind('<Escape>', lambda e: self.quit())
        self.title("Absensi Pegawai")
        self.resizable(0, 0)

        self.hadir_analisis = []
        self.hadir_insert = []
        self.hadir_name = "Tidak Dikenali"
        self.count = 0

        self.face_recognizer = cv2.face.LBPHFaceRecognizer_create(1, 6, 8, 8)
        self.face_recognizer.read('training{0}.xml'.format(total_sample))

        self.selected_user = StringVar()
        self.selected_user.set("-- Pilih --")

        self.list_user_names = get_list_user_names()
        self.option_users_dict = get_option_user_dicts()

        # open video source (by default this will try to open the computer webcam)
        self.video_comp = VideoCaptureComp(video_source)

        # Create a canvas that can fit the above video source size
        self.canvas = Canvas(self, width=self.video_comp.width, height=self.video_comp.height)
        self.canvas.grid(column=0, row=0, rowspan=15)

```

```python
        # SWITCH
        self.face_recognition_switch = BooleanVar(value=True)
        self.switch_frame = ttk.Frame(self, padding=(12, 3, 12, 3))
        self.switch_frame.grid(column=1, row=0, sticky=W)
        self.on_button = Radiobutton(self.switch_frame,
                                     text="ON",
                                     variable=self.face_recognition_switch,
                                     indicatoron=False,
                                     value=True,
                                     width=8,
                                     command=self.on_switch_change)
        self.off_button = Radiobutton(self.switch_frame,
                                      text="OFF",
                                      variable=self.face_recognition_switch,
                                      indicatoron=False,
                                      value=False,
                                      width=8,
                                      command=self.on_switch_change)
        self.on_button.grid(column=0, row=0)
        self.off_button.grid(column=1, row=0)

```

```
        # ABSENSI
        self.absensi_type = StringVar()
        self.absensi_type.set(AbsensiType.CHECK_IN.value)
        self.frame_absensi_type = ttk.Frame(self,
                                            borderwidth=2,
                                            relief="groove",
                                            width=200,
                                            height=50,
                                            padding=(12, 3, 12, 3))
        self.frame_absensi_type.configure(height=self.frame_absensi_type["height"],
                                          width=self.frame_absensi_type["width"])
        self.frame_absensi_type.grid_propagate(0)
        self.frame_absensi_type.grid(column=1, row=1, sticky=W)
        self.label_type_absensi = Label(self.frame_absensi_type,
                                        text="Jenis Absensi:",
                                        anchor='w')
        self.rb_datang = ttk.Radiobutton(self.frame_absensi_type,
                                         text=AbsensiType.CHECK_IN.value,
                                         variable=self.absensi_type,
                                         value=AbsensiType.CHECK_IN.value,
                                         command=self.on_absensi_type_change)
```

```
        self.rb_pulang = ttk.Radiobutton(self.frame_absensi_type,
                                         text=AbsensiType.CHECK_OUT.value,
                                         variable=self.absensi_type,
                                         value=AbsensiType.CHECK_OUT.value,
                                         command=self.on_absensi_type_change)
        self.label_type_absensi.grid(column=1, row=0, sticky=W)
        self.rb_datang.grid(column=1, row=1, sticky=W)
        self.rb_pulang.grid(column=2, row=1, sticky=W)
        self.frame_absensi_type.columnconfigure(1, weight=1)
        self.frame_absensi_type.columnconfigure(2, weight=1)

```

```
        # ADD NEW USER
        self.user_name = StringVar()
        self.frame_new_user = ttk.Frame(self,
                                        borderwidth=2,
                                        relief="groove",
                                        width=200,
                                        height=100,
                                        padding=(12, 3, 12, 3))
        self.frame_new_user.configure(height=self.frame_new_user["height"], width=self.frame_new_user["width"])
        self.frame_new_user.grid_propagate(0)
        self.frame_new_user.grid(column=1, row=2, sticky=W)
        self.label_add_data = ttk.Label(self.frame_new_user, text="Tambah Data")
        self.label_name = ttk.Label(self.frame_new_user, text="Nama")
        self.input_name = ttk.Entry(self.frame_new_user, textvariable=self.user_name)
        self.button_add_data = Button(self.frame_new_user,
                                      text='Tambah',
                                      command=(lambda value=self.input_name: self.add_new_user(user_name=value.get())))
        self.label_add_data.grid(column=1, row=0, sticky=W)
        self.label_name.grid(column=1, row=1, sticky=W)
        self.input_name.grid(column=1, row=2)
        self.button_add_data.grid(column=1, row=3, pady=5, sticky=W)

```

```
        # EXPORT
        self.export_type = StringVar()
        self.selected_date = StringVar()
        self.export_type.set(ExportType.USER.value)
        self.frame_export = ttk.Frame(self,
                                      borderwidth=2,
                                      relief="groove",
                                      width=200,
                                      height=200,
                                      padding=(12, 3, 12, 3))
        self.frame_export.configure(height=self.frame_export["height"], width=self.frame_export["width"])
        self.frame_export.grid_propagate(0)
        self.frame_export.grid(column=1, row=3, sticky=NW)
        self.label_export_data = ttk.Label(self.frame_export, text="Export Data")
        self.label_type_export = Label(self.frame_export,
                                       text="Berdasarkan:",
                                       anchor='w')
        self.rb_user = ttk.Radiobutton(self.frame_export,
                                       text=ExportType.USER.value,
                                       variable=self.export_type,
                                       value=ExportType.USER.value,
                                       command=self.on_export_type_change)
```

```
        self.rb_date = ttk.Radiobutton(self.frame_export,
                                       text=ExportType.DATE.value,
                                       variable=self.export_type,
                                       value=ExportType.DATE.value,
                                       command=self.on_export_type_change)
        self.label_select_user = Label(self.frame_export, text="Pilih user:")
        self.option_users = OptionMenu(self.frame_export, self.selected_user, *self.option_users_dict.keys())
        self.label_date = Label(self.frame_export, text="Tanggal (YYYY-MM-DD):")
        self.input_date = ttk.Entry(self.frame_export, textvariable=self.selected_date)
        self.button_export = Button(self.frame_export, text='Export',
                                    command=(lambda value=self.export_type: self.export_absensi(value.get())))
        self.label_export_data.grid(column=1, row=0, sticky=W)
        self.label_type_export.grid(column=1, row=1, sticky=W)
        self.rb_user.grid(column=1, row=2, sticky=W)
        self.rb_date.grid(column=1, row=3, sticky=W)
        self.button_export.grid(column=1, row=6, pady=5, sticky=W)
```

```
        # INFO MESSAGE
        self.info_message = StringVar()
        self.label_info = Label(self, textvariable=self.info_message)
        self.label_info.grid(column=1, row=4, sticky=W)

        self.on_absensi_type_change()
        self.on_export_type_change()

        # After it is called once, the update method will be automatically called every delay milliseconds
        self.delay = 50
        self.update_frame()

    def refresh_face_recognizer(self):
        self.face_recognizer = cv2.face.LBPHFaceRecognizer_create(1, 6, 8, 8)
        self.face_recognizer.read('training{0}.xml'.format(total_sample))

    def update_info_message(self, message):
        self.info_message.set(message)

    def on_absensi_type_change(self):
        selected_absensi_type = self.absensi_type.get()
        message = "Absensi {0} Aktif".format(selected_absensi_type)
        self.update_info_message(message)
        print(message)

    def on_change_selected_user(self, selected_user):
        self.selected_user.set(selected_user)
```

```
244     def on_change_selected_user(self, selected_user):
245         self.selected_user.set(selected_user)
246 
247     def on_export_type_change(self):
248         selected_export_type = self.export_type.get()
249         if selected_export_type == ExportType.USER.value:
250             self.label_date.grid_remove()
251             self.input_date.grid_remove()
252             self.label_select_user.grid(column=1, row=4, sticky=W)
253             self.option_users.grid(column=1, row=5, sticky=W)
254         else:
255             self.label_select_user.grid_remove()
256             self.option_users.grid_remove()
257             self.label_date.grid(column=1, row=4, sticky=W)
258             self.input_date.grid(column=1, row=5, sticky=W)
259 
260     def on_switch_change(self):
261         face_recognition_active = self.face_recognition_switch.get()
262         if face_recognition_active:
263             message = "Absensi Diaktifkan"
264             self.refresh_face_recognizer()
265         else:
266             message = "Absensi Dinonaktifkan"
267         print(message)
268         self.update_info_message(message)
269 
```

e) Source code rekognisi wajah

```
269 
270     def update_frame(self):
271         # Get a frame from the video source
272         ret, frame = self.video_comp.get_frame()
273 
274         face_recognition_active = self.face_recognition_switch.get()
275         if face_recognition_active:
276             frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
277             faces = face_detector.detectMultiScale(frame_gray, scaleFactor=1.1, minNeighbors=8, minSize=(30, 30))
278             if len(faces) == 1:
279                 self.count = 0
280                 time_start = time.perf_counter()
281                 for (x, y, w, h) in faces:
282                     face = frame[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
283                     eye_gray = frame_gray[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
284                     eye_image = eye_gray
285 
286                     # Bounding box wajah
287                     cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 1)
288 
289                     rows, cols = eye_image.shape
290                     eyes = eye_detector.detectMultiScale(eye_image, 1.3, 5)
291                     for (sx, sy, sw, sh) in eyes:
292                         # Bounding box mata
293                         cv2.rectangle(face, (sx, sy), (sx + sw, sy + sh), (0, 255, 0), 1)
294                         # input berupa dua mata
295                         if eyes.shape[0] == 2:
296                             # menentukan mata kiri dan kanan
297                             if eyes[1][0] > eyes[0][0]:
```

```
297                 if eyes[1][0] > eyes[0][0]:
298                     # beda tinggi di antara mata
299                     DY = ((eyes[1][1] + eyes[1][3] / 2) - (eyes[0][1] + eyes[0][3] / 2))
300                     # beda lebar antara mata
301                     DX = ((eyes[1][0] + eyes[1][2] / 2) - eyes[0][0] + (eyes[0][2] / 2))
302                 else:
303                     # beda tinggi antara mata
304                     DY = (-(eyes[1][1] + eyes[1][3] / 2) + (eyes[0][1] + eyes[0][3] / 2))
305                     # beda wajah antara mata
306                     DX = (-(eyes[1][0] + eyes[1][2] / 2) + eyes[0][0] + (eyes[0][2] / 2))
307
308                 # memastikan rotasi jika diperlukan
309                 if (DX != 0.0) and (DY != 0.0):
310                     # perhitungan sudut euclidian
311                     Theta = math.degrees(math.atan(round(float(DY) / float(DX), 2)))
312                     print("Theta  ", str(Theta))
313
314                     M = cv2.getRotationMatrix2D((cols / 2, rows / 2), Theta, 1)  # menentukan matrix rotasi
315                     eye_image = cv2.warpAffine(eye_image, M, (cols, rows))
```

```
317                     Face2 = face_detector.detectMultiScale(eye_image, 1.1, 8)  # deteksi wajah untuk isolasi wajah
318                     for (FaceX, FaceY, FaceWidth, FaceHeight) in Face2:
319                         CroppedFace = eye_image[FaceY + int(FaceHeight / 4.7): FaceY + int(
320                             FaceHeight - int(FaceHeight / 6.5)),
321                                       FaceX + int(FaceWidth / 4.7):FaceX + int(
322                                           FaceWidth - int(FaceWidth / 4.7))]
323                         CroppedFace = cv2.resize(CroppedFace, (54, 59))
324                         # Bounding box wajah
325                         # cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 1)
326
327                         id, distance = self.face_recognizer.predict(CroppedFace)
328                         if distance >= 70:
329
330                             NameID = 0
331                             distanceTxt = "dist : {0}".format(round(distance))
332                             print('distance :' + str(round(distance)))
333                             print('id :' + str(NameID))
334                             self.hadir_analisis.append(NameID)
335
336                         else:
337                             NameID = id
338                             distanceTxt = "dist : {0}".format(round(distance))
339                             print('distance :' + str(round(distance)))
340                             print('id :' + str(NameID))
341
```

```
                    if len(self.hadir_analisis) < 15:
                        self.hadir_analisis.append(NameID)
                        print('analisis data :' + str(self.hadir_analisis))
                    else:
                        detected_user_id = mode(self.hadir_analisis)
                        total_detected_user_id = self.hadir_analisis.count(detected_user_id)
                        total_detected_unknown = self.hadir_analisis.count(0)
                        is_accepted = total_detected_user_id + total_detected_unknown >= 10
                        if is_accepted:
                            print('terbanyak muncul :' + str(detected_user_id))

                            current_datetime = datetime.now()
                            dtstring_date = current_datetime.strftime("%d/%m/%Y")
                            dtstring_time = current_datetime.strftime("%H:%M:%S")
                            dtstring_hour_minute = current_datetime.strftime("%H:%M")

                            if detected_user_id == 0 or detected_user_id >= len(self.list_user_names):
                                self.hadir_name = 'Tidak dikenali'
                            else:
                                self.hadir_name = self.list_user_names[detected_user_id]
                                absensi_type = AbsensiType(self.absensi_type.get())
                                user_absensi = get_user_absensi_today(detected_user_id)
                                if (user_absensi is None and absensi_type == AbsensiType.CHECK_IN) \
                                        or (user_absensi and absensi_type == AbsensiType.CHECK_OUT):
                                    add_absensi(self.absensi_type.get(), detected_user_id,
                                                current_datetime)
                                    time_elapsed = (time.perf_counter() - time_start)
                                    print('waktu proses :' + str(time_elapsed))
                                    message = "{0} {1} pada pukul {2}".format(self.hadir_name,
                                                                              self.absensi_type.get().lower(),
                                                                              dtstring_hour_minute)
                                    self.update_info_message(message)

                                    self.hadir_insert.append(self.hadir_name)
                                    self.hadir_insert.append(dtstring_time)
                                    self.hadir_insert.append(dtstring_date)
                                    print(self.hadir_insert)

                                    with open('daftarhadirdatang.csv', 'a+', newline='') as csv_file:
                                        writer = csv.writer(csv_file, delimiter=',')
                                        writer.writerow(self.hadir_insert)

                                    '''oldcsv = pd.read_csv('daftarhadirdatang.csv')
                                    newcsv = oldcsv.drop_duplicates(subset = 'name')'''

                                    print(str(self.hadir_name) + ' melakukan absen pada :' + dtstring_time)
                                self.hadir_analisis.clear()
                                self.hadir_insert.clear()

                    cv2.putText(frame, str(self.hadir_name), (x + 5, y - 5), font, 1, (255, 255, 255), 2)
                    cv2.putText(frame, str(distanceTxt), (x + 5, y + h - 5), font, 1, (0, 255, 0), 1)
```

```
            if len(faces) == (0):
                self.count = self.count + 1
                if self.count>10:
                    self.count=0
                    self.hadir_name='tidak dikenali'
                    self.hadir_insert.clear()
                    self.hadir_analisis.clear()

        if ret:
            self.photo = ImageTk.PhotoImage(image=Image.fromarray(frame))
            self.canvas.create_image(0, 0, image=self.photo, anchor=NW)

        self.after(self.delay, self.update_frame)
```

f) Lanjutan source code GUI

```
407     def update_option_users(self):
408         self.list_user_names = get_list_user_names()
409         self.option_users_dict = get_option_user_dicts()
410         self.selected_user.set("-- Pilih --")
411         menu = self.option_users['menu']
412         menu.delete(0, 'end')
413         for user_name in self.option_users_dict:
414             menu.add_command(label=user_name, command=lambda value=user_name: self.on_change_selected_user(value))
415 
416     def add_new_user(self, user_name):
417         new_user = create_user(user_name)
418         if new_user:
419             is_agree_make_data = messagebox.askyesno(
420                 title="Buat dataset wajah",
421                 message="Sistem akan menonaktifkan absensi untuk membuat dataset wajah {0}. "
422                         "Pastikan wajah {0} tampak di kamera. Lanjutkan?".format(new_user.name))
423             if is_agree_make_data:
424                 if self.face_recognition_switch:
425                     self.face_recognition_switch.set(0)
426                     self.generate_dataset(user_id=new_user.id)
427 
428     def generate_dataset(self, user_id):
429         self.update_info_message("Membuat Dataset Wajah...")
430         generate_dataset_thread = GenerateDataset(user_id, self.video_comp)
431         generate_dataset_thread.daemon = True
432         generate_dataset_thread.start()
433         self.monitor_generate_dataset_thread(generate_dataset_thread)
```

```
435     def monitor_generate_dataset_thread(self, thread):
436         if thread.is_alive():
437             if thread.sample_number < total_sample:
438                 self.after(50, lambda: self.update_info_message(
439                     "Ekstrak frame : {0} dari {1}".format(thread.sample_number, total_sample)))
440             # check the thread every 50ms
441             self.after(50, lambda: self.monitor_generate_dataset_thread(thread))
442         else:
443             self.update_option_users()
444             self.after(100, lambda: messagebox.showinfo(
445                 title="Sukses",
446                 message="Dataset wajah berhasil dibuat. Silakan tekan tombol ON untuk mengaktifkan absensi."))
447 
448     def export_absensi(self, export_type):
449         if export_type == ExportType.USER.value:
450             selected_user = self.option_users_dict.get(self.selected_user.get())
451             if selected_user:
452                 list_data = get_list_absensi_user(selected_user)
453                 print(export_type, self.selected_user.get(), "Total: {0}".format(len(list_data)))
454                 self.generate_pdf(export_type, list_data, self.selected_user.get())
455         else:
456             selected_date = self.selected_date.get()
457             if selected_date:
458                 list_data = get_list_absensi_date(selected_date)
459                 print(export_type, self.selected_date.get(), "Total: {0}".format(len(list_data)))
460                 self.generate_pdf(export_type, list_data, selected_date)
461 
```

```
    def generate_pdf(self, export_type, list_data, keyword):
        file_name = "Export_{0}_{1}_{2}.pdf".format(export_type, keyword, datetime.now().strftime("%Y%m%d_%H%M%S"))
        doc = SimpleDocTemplate(file_name, pagesize=A4)
        elements = []
        content = []
        headers = ["No.", "Tanggal", "Nama", "Jam Datang", "Jam Pulang", "Status"]
        content.append(headers)
        for index, data in enumerate(list_data):
            number = index + 1
            date = data.date
            name = data.user.name
            check_in_time = data.check_in_time or ""
            check_out_time = data.check_out_time or ""
            status = data.status.value
            row = [number, date, name, check_in_time, check_out_time, status]
            content.append(row)

        t = Table(content)
        t.setStyle(TableStyle([
            ('INNERGRID', (0, 0), (-1, -1), 0.25, colors.black),
            ('BOX', (0, 0), (-1, -1), 0.25, colors.black),
        ]))
        elements.append(t)
        doc.build(elements)
        message = "Data berhasil di-export dengan nama file {0}".format(file_name)
        self.update_info_message("Export Finish")
        print(message)
        messagebox.showinfo(title="Sukses", message=message)


```

```
class VideoCaptureComp:
    def __init__(self, video_source):
        # Open the video source
        if type(video_source) is int:
            self.vid = cv2.VideoCapture(video_source, cv2.CAP_DSHOW)
        else:
            self.vid = cv2.VideoCapture(video_source)

        if not self.vid.isOpened():
            raise ValueError("Unable to open video source", video_source)

        # Get video source width and height
        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)

    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            if ret:
                # Return a boolean success flag and the current frame converted to BGR
                return ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            else:
                return ret, None
        else:
            return False, None

```

```
    # Release the video source when the object is destroyed
    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()
            cv2.destroyAllWindows()

```

g) Source code model user

```python
from sqlalchemy import Column, String, Integer
from db.base_class import Base


class User(Base):
    id = Column(Integer, autoincrement=True, primary_key=True, index=True)
    name = Column(String(100), nullable=False)

    def __str__(self):
        return str(self.__dict__)

```

h) Source code model presensi

```python
from enum import Enum

from sqlalchemy import Column, Integer, ForeignKey, Enum as SQLEnum, Time, Date
from sqlalchemy.orm import relationship

from db.base_class import Base


class AbsensiType(str, Enum):
    CHECK_IN = "Datang"
    CHECK_OUT = "Pulang"


class AbsensiStatus(str, Enum):
    ABSENT = "Tanpa Keterangan"
    PRESENT = "Hadir"
    HOME = "Pulang"


```

```python
class Absensi(Base):
    id = Column(Integer, autoincrement=True, primary_key=True, index=True)

    user_id = Column(Integer, ForeignKey("user.id"))
    user = relationship("User", backref="list_absensi", lazy='subquery')

    type = Column(SQLEnum(AbsensiType), default=AbsensiType.CHECK_IN)
    date = Column(Date)
    check_in_time = Column(Time)
    check_out_time = Column(Time)
    status = Column(SQLEnum(AbsensiStatus), default=AbsensiStatus.PRESENT)

    def __str__(self):
        return str(self.__dict__)

```

i) Source code CRUD user

```python
from sqlalchemy.orm import Session

from db.models import User


def get_list_users(db: Session):
    return db.query(User).all()


def get_user(db: Session, user_id: int):
    return db.query(User).filter(User.id == user_id).first()


def create_user(db, name):
    new_user = User(name=name)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)
    return new_user

```

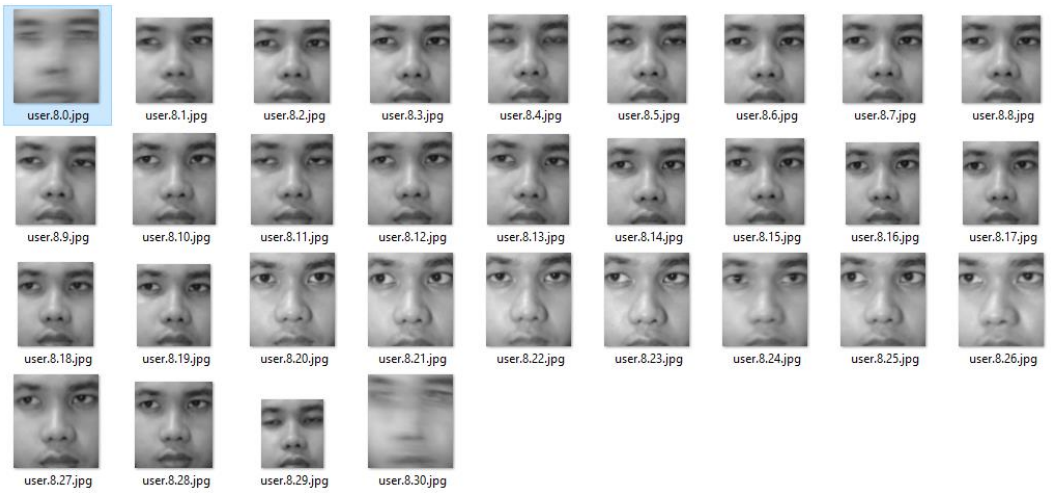
j) Source code CRUD presensi

```python
from datetime import datetime

from sqlalchemy.orm import Session

from db.models import Absensi
from db.models.absensi import AbsensiType, AbsensiStatus


def get_list_absensi(db: Session):
    return db.query(Absensi).all()


def get_list_absensi_by_user_id(db: Session, user_id: int):
    return db.query(Absensi).filter(Absensi.user_id == user_id).all()


def get_list_absensi_by_date(db: Session, date: str):
    return db.query(Absensi).filter(Absensi.date == date).all()


def get_absensi(db: Session, absensi_id: int):
    return db.query(Absensi).filter(Absensi.id == absensi_id).first()


def get_absensi_today_by_user_id(db: Session, user_id: int):
    current_date = datetime.now().strftime("%Y-%m-%d")
    return db.query(Absensi).filter(Absensi.user_id == user_id, Absensi.date == current_date).first()
```

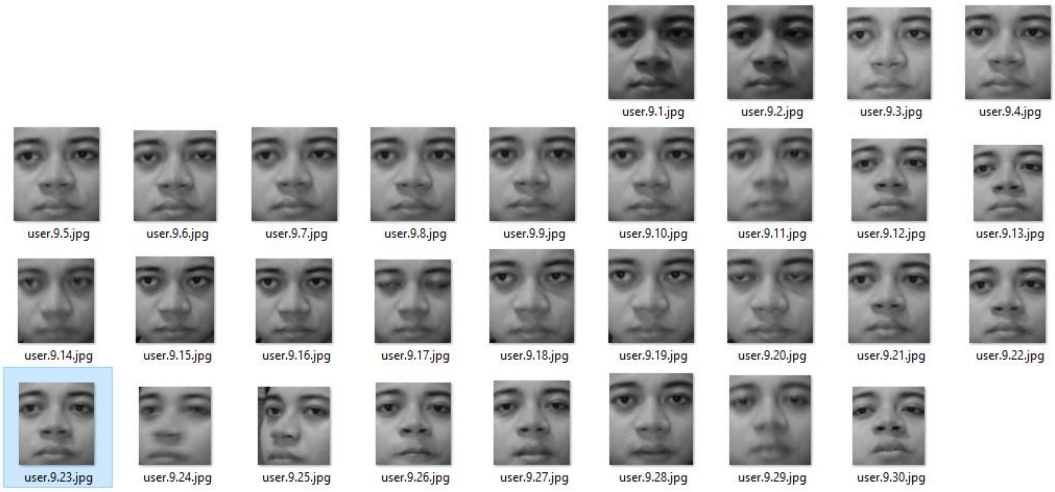
```python

def get_absensi_today_by_type_and_user_id(db: Session, type: AbsensiType, user_id: int):
    current_date = datetime.now().strftime("%Y-%m-%d")
    return db.query(Absensi).filter(Absensi.type == type, Absensi.user_id == user_id,
                                    Absensi.date == current_date).first()


def create_absensi(db: Session, user_id: int, type: AbsensiType, date: str, time: str):
    if type == AbsensiType.CHECK_IN:
        new_absensi = Absensi(user_id=user_id, type=type, date=date, check_in_time=time)
        db.add(new_absensi)
        db.commit()
        db.refresh(new_absensi)
        return new_absensi
    else:
        absensi = get_absensi_today_by_user_id(db, user_id)
        if absensi:
            db.query(Absensi).filter(Absensi.id == absensi.id).update({
                Absensi.type: type,
                Absensi.check_out_time: time,
                Absensi.status: AbsensiStatus.HOME
            })
            db.commit()
        return absensi

```

- **Data training**

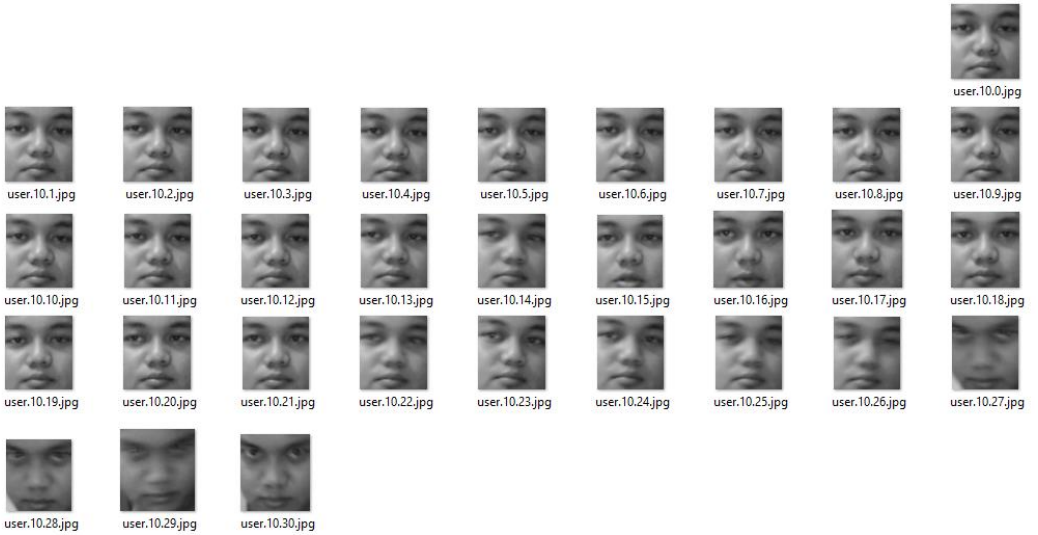
a) Data 0



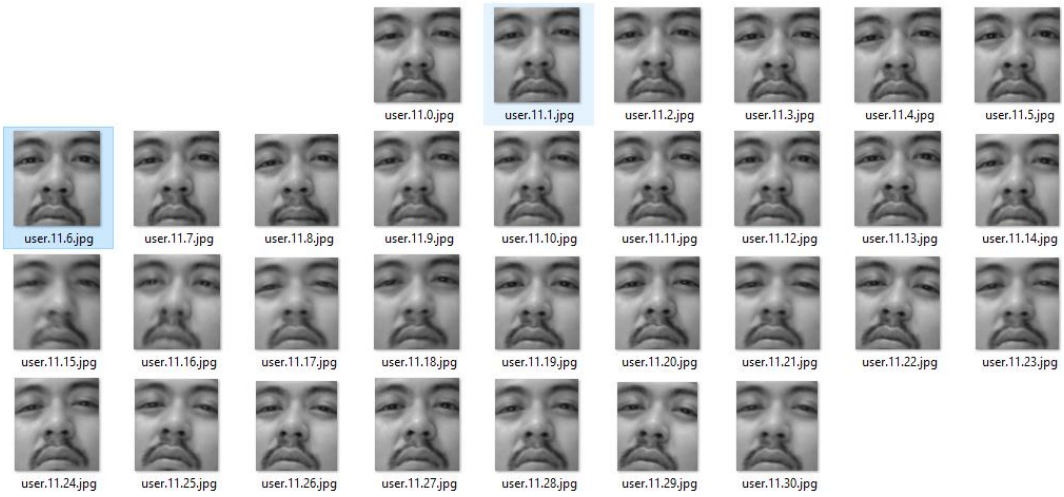
b) Data 1



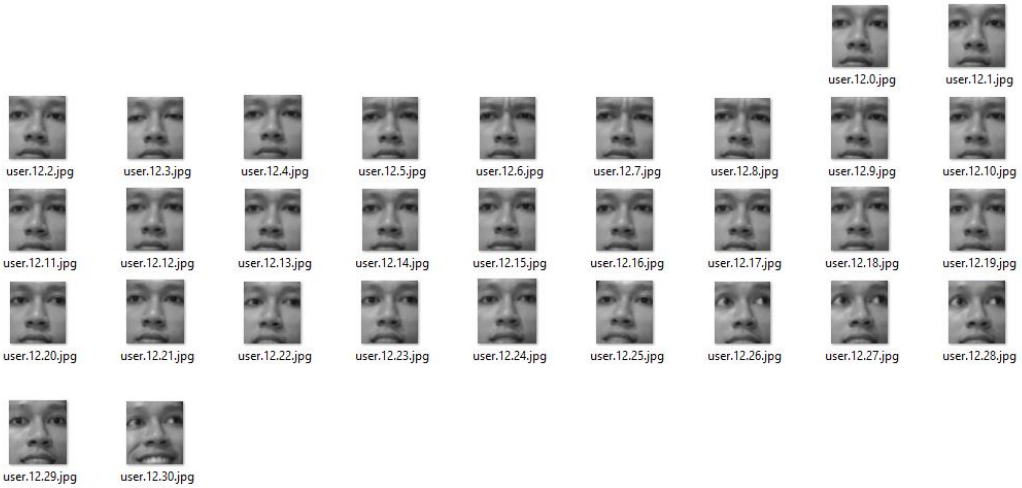
c) Data 2



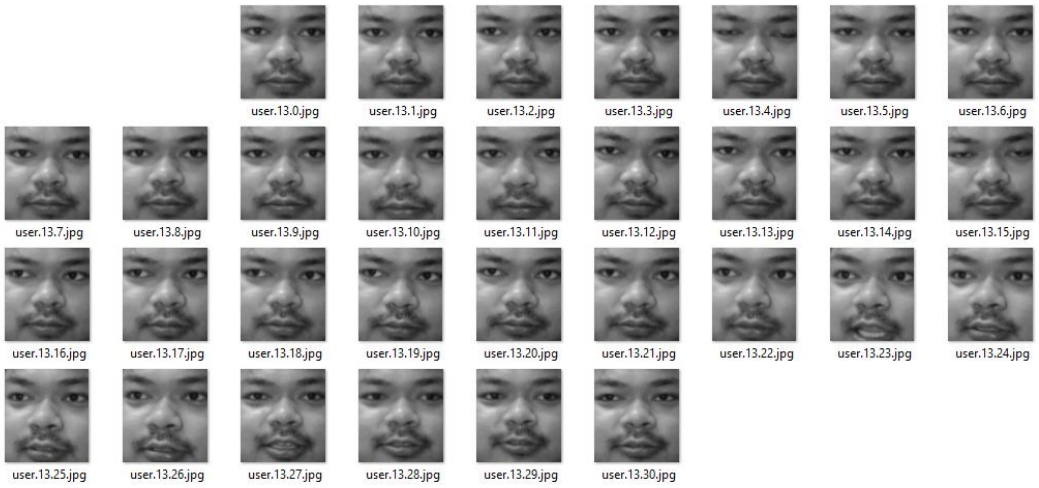
d) Data 3



e) Data 4



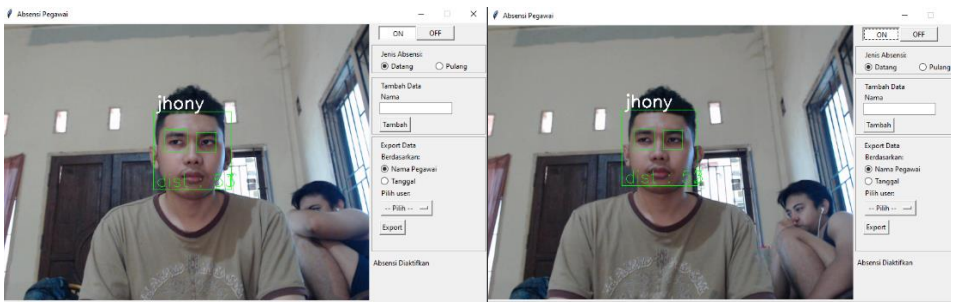
f) Data 5

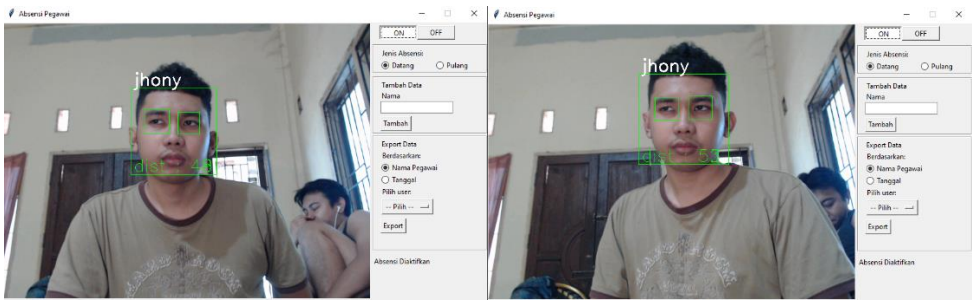


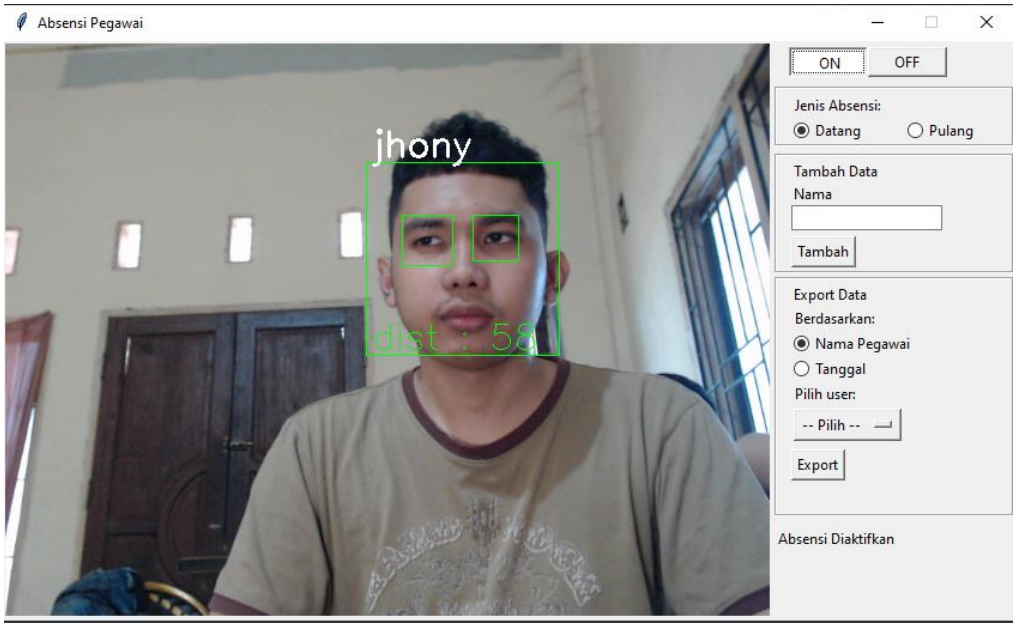
- **Data uji**

  a) **Data yang dilatih**

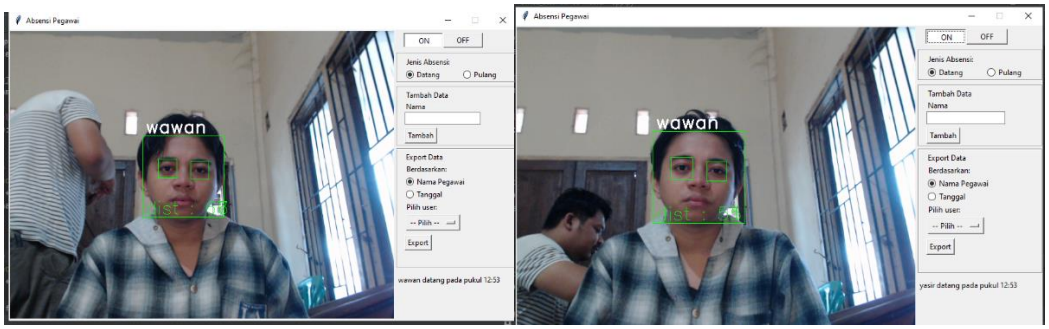
     1. **Data 0**

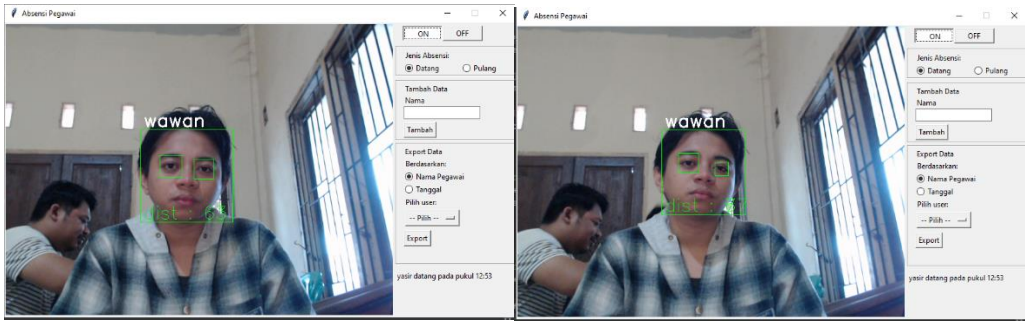


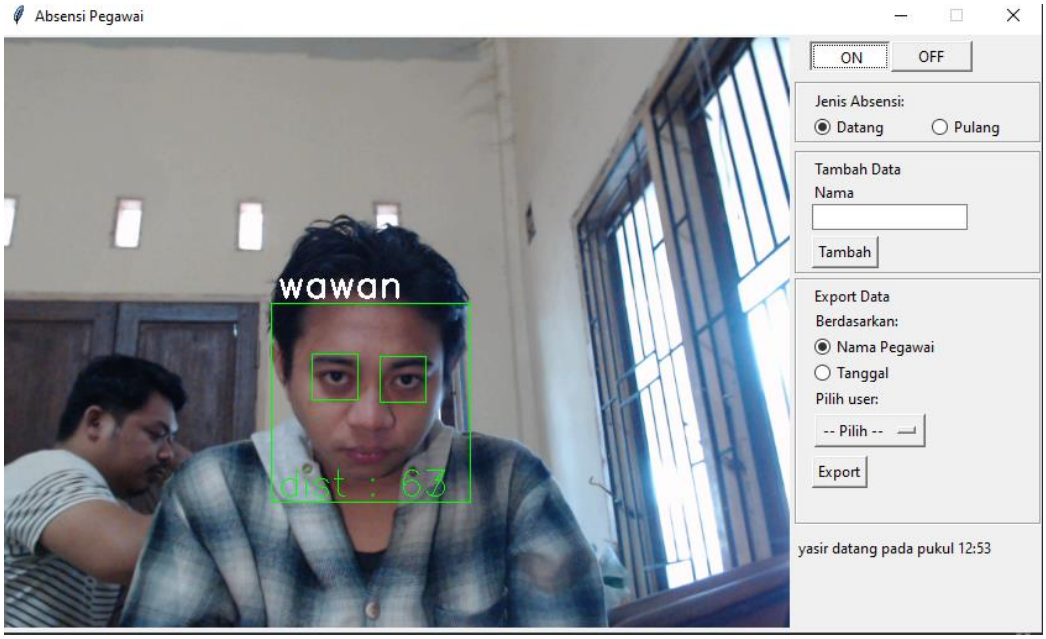




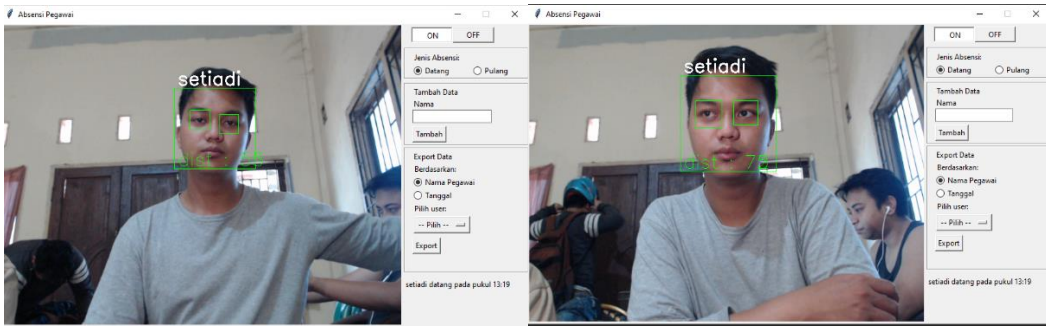
2. Data 1

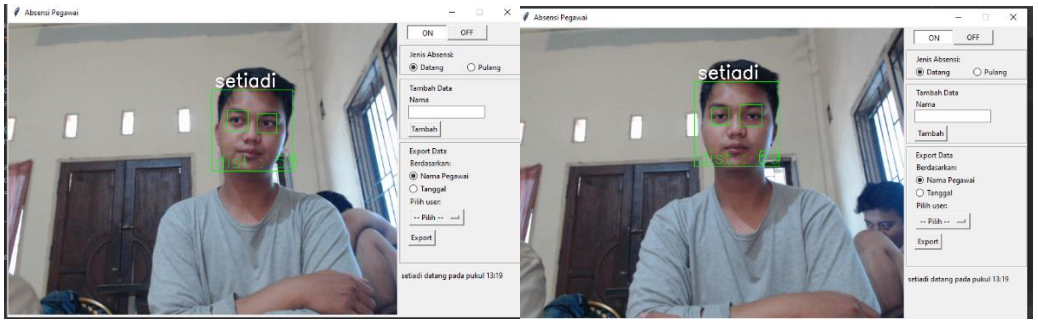

Absensi Pegawai
wawan
dist : 63
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
wawan datang pada pukul 12:53
yasir datang pada pukul 12:53

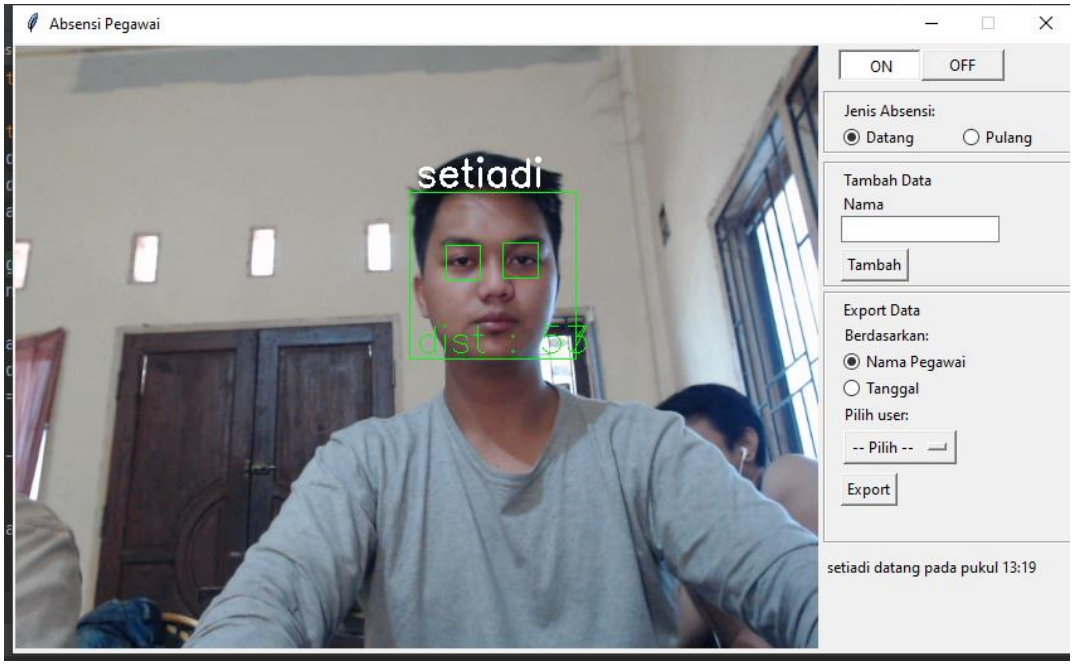

Absensi Pegawai
wawan
dist : 63
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
yasir datang pada pukul 12:53


Absensi Pegawai
wawan
dist : 63
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
yasir datang pada pukul 12:53

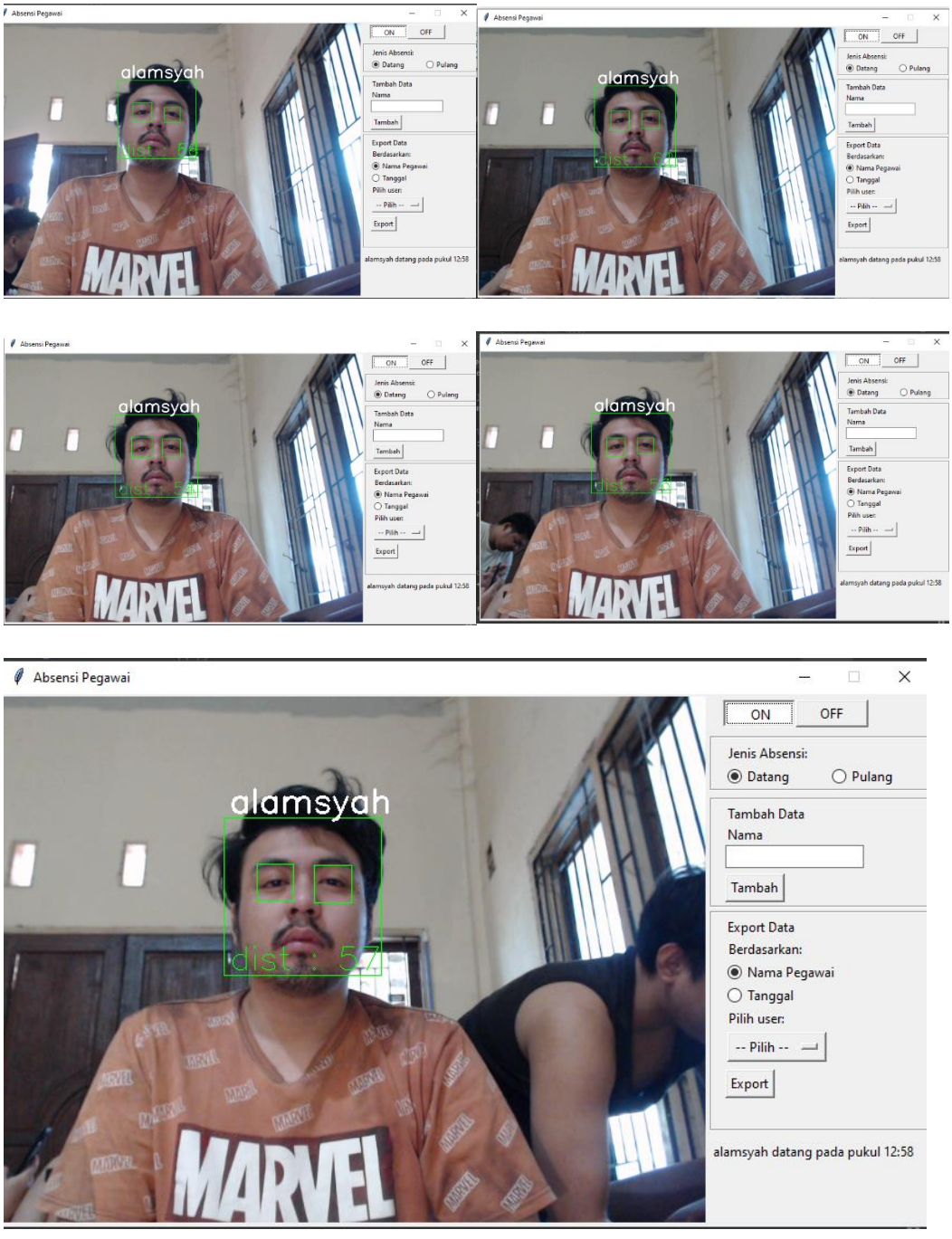
3. Data 2


Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
setiadi
setiadi datang pada pukul 13:19

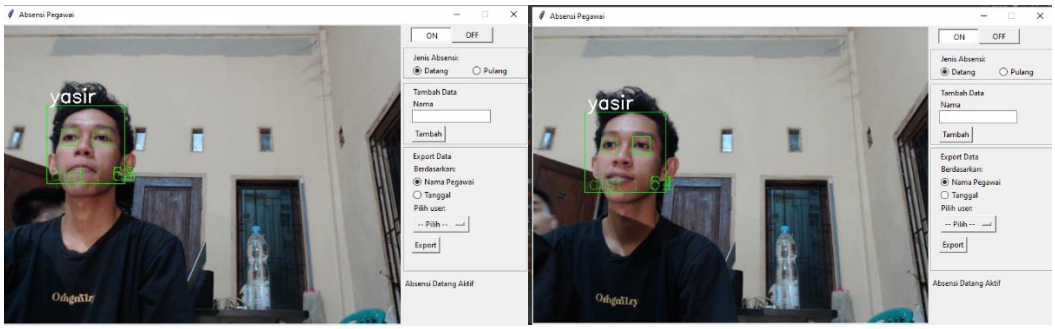

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
setiadi
setiadi datang pada pukul 13:19

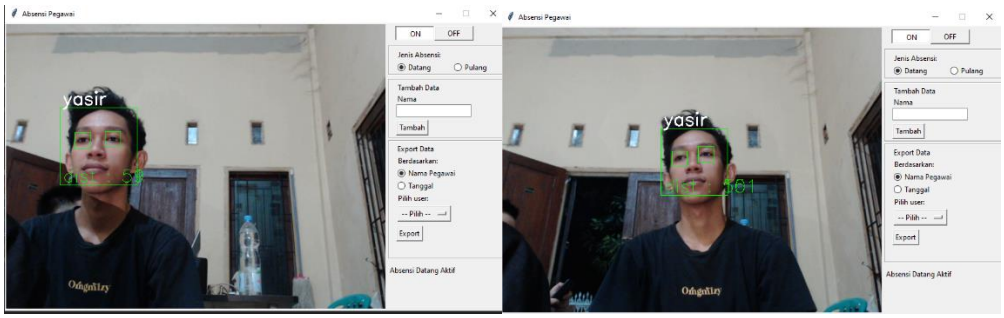

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
setiadi
dist : 67
setiadi datang pada pukul 13:19

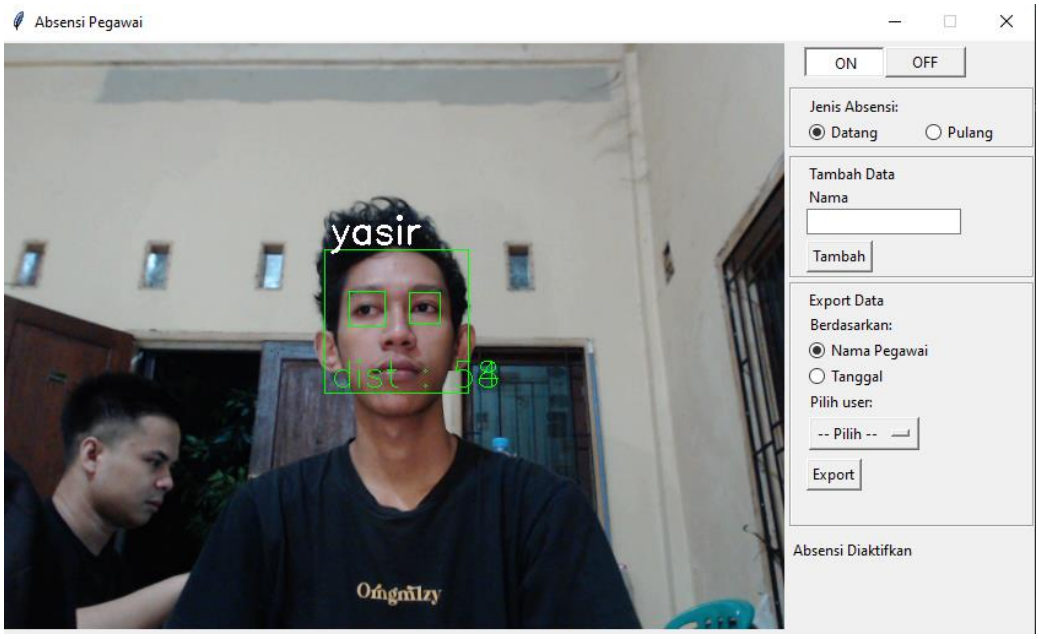
4. Data 3


Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
alamsyah datang pada pukul 12:58
alamsyah
dist : 57
MARVEL

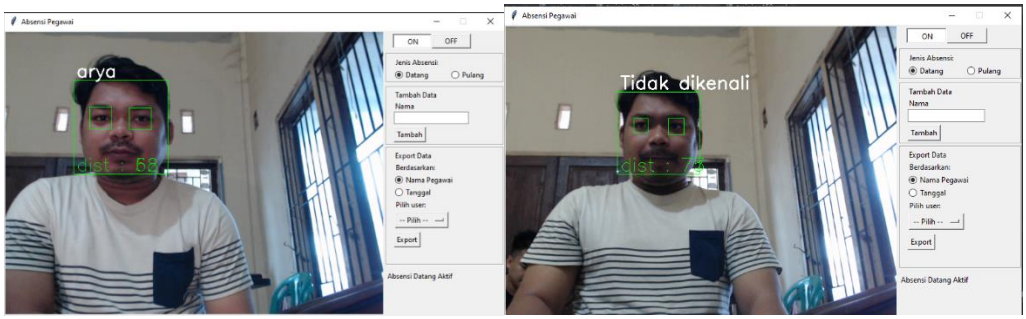
5. Data 4

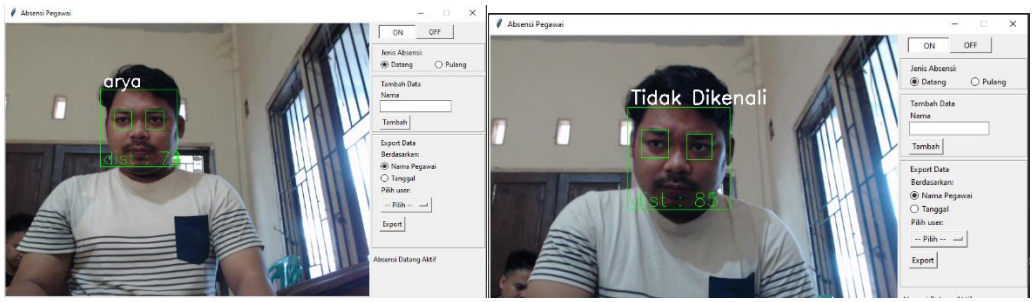

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
yasir

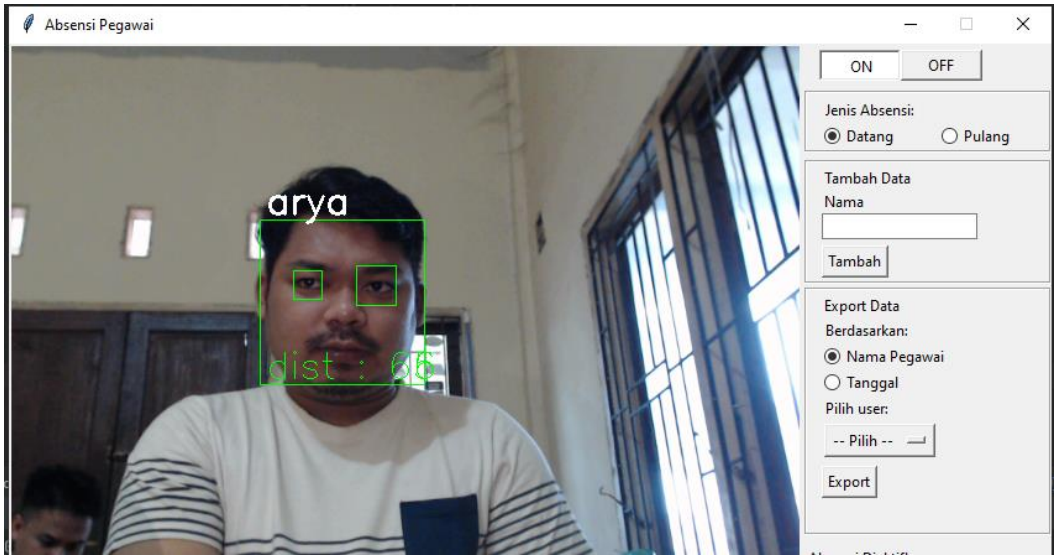

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
yasir


Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan
yasir
dist : 58

6. Data 5

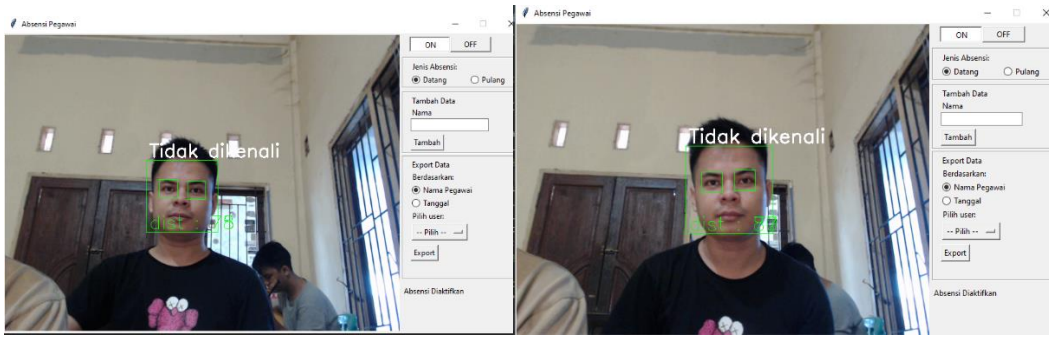

Absensi Pegawai
arya
dist : 68
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
Absensi Pegawai
Tidak dikenali
dist : 74
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif

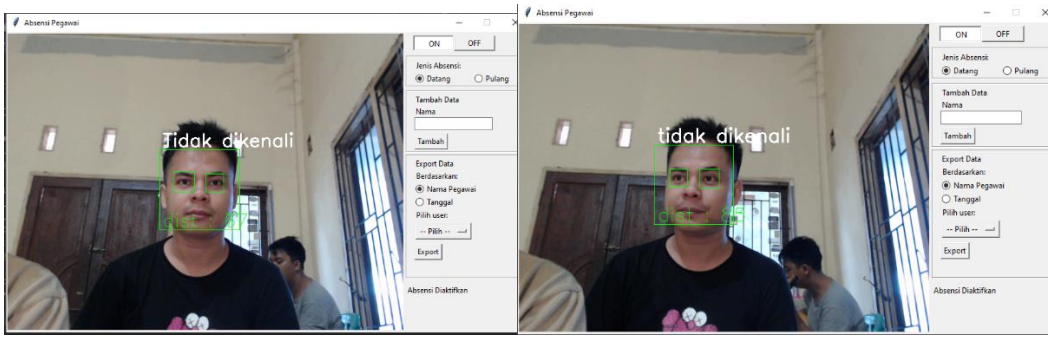

Absensi Pegawai
arya
dist : 74
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
Absensi Pegawai
Tidak Dikenali
dist : 85
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export

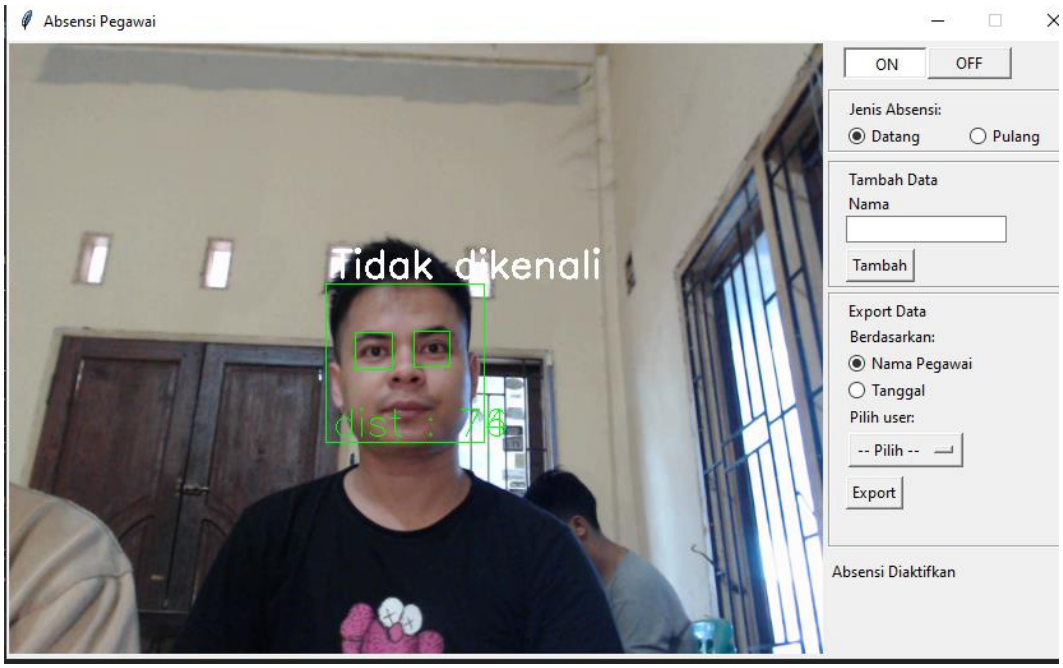

Absensi Pegawai
arya
dist : 66
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export

b) Data yang tidak dilatih

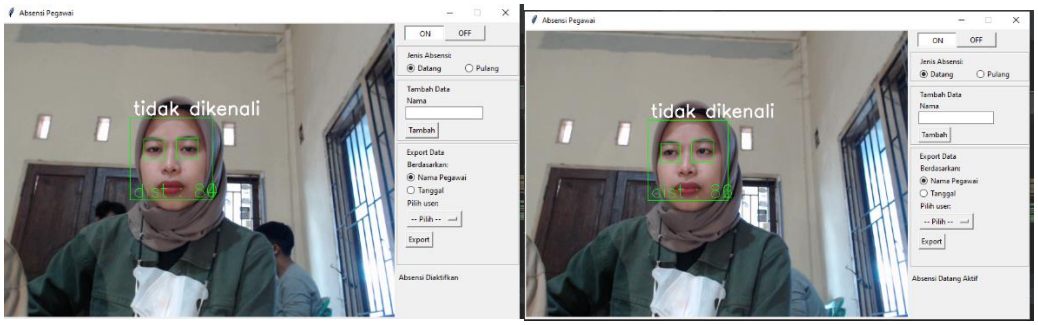
1. Data 0

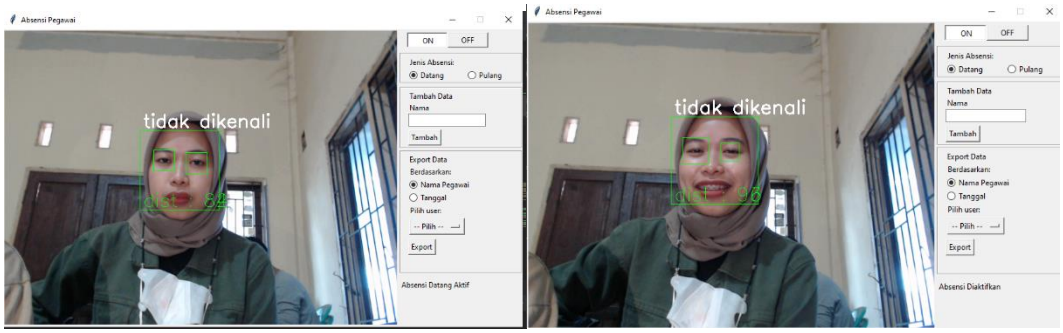


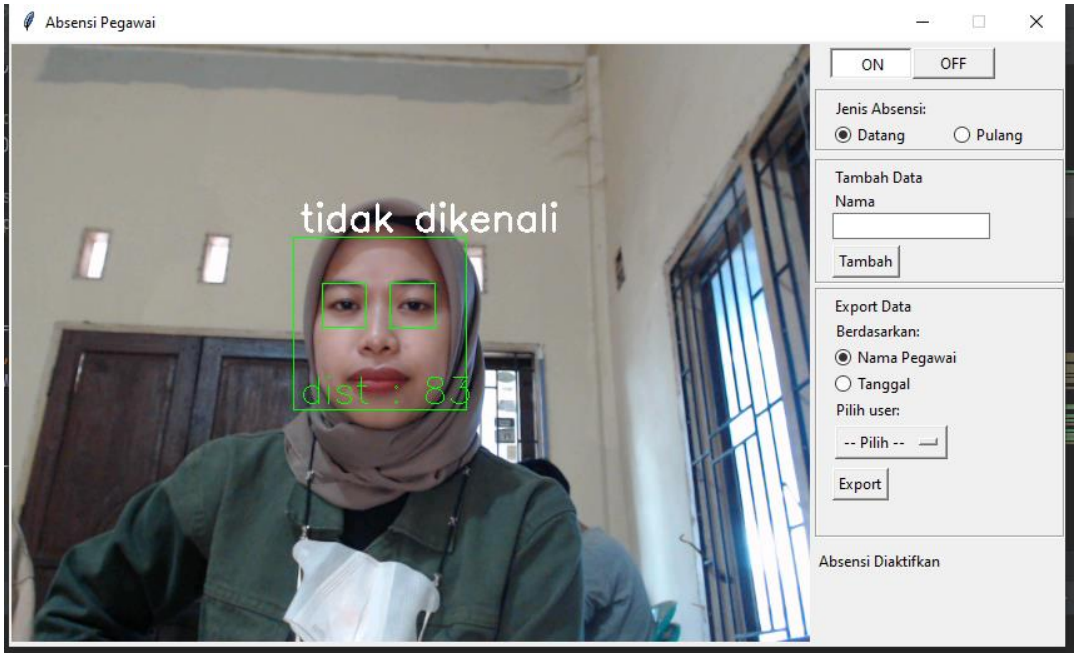




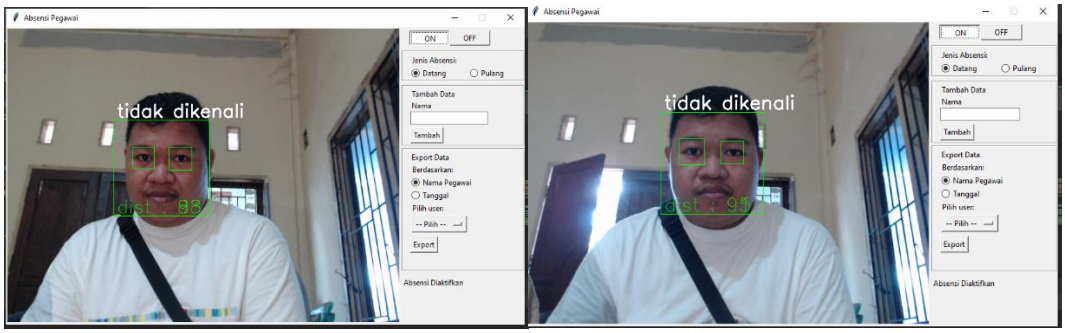
2. Data 1

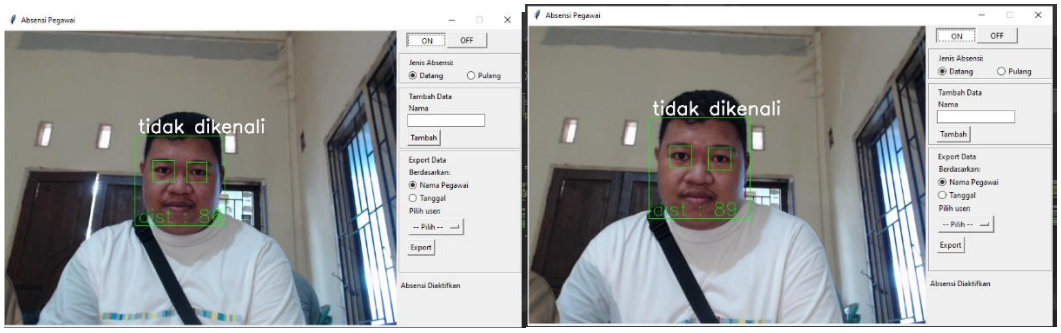

Absensi Pegawai
tidak dikenali
dist : 84
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan
Absensi Datang Aktif

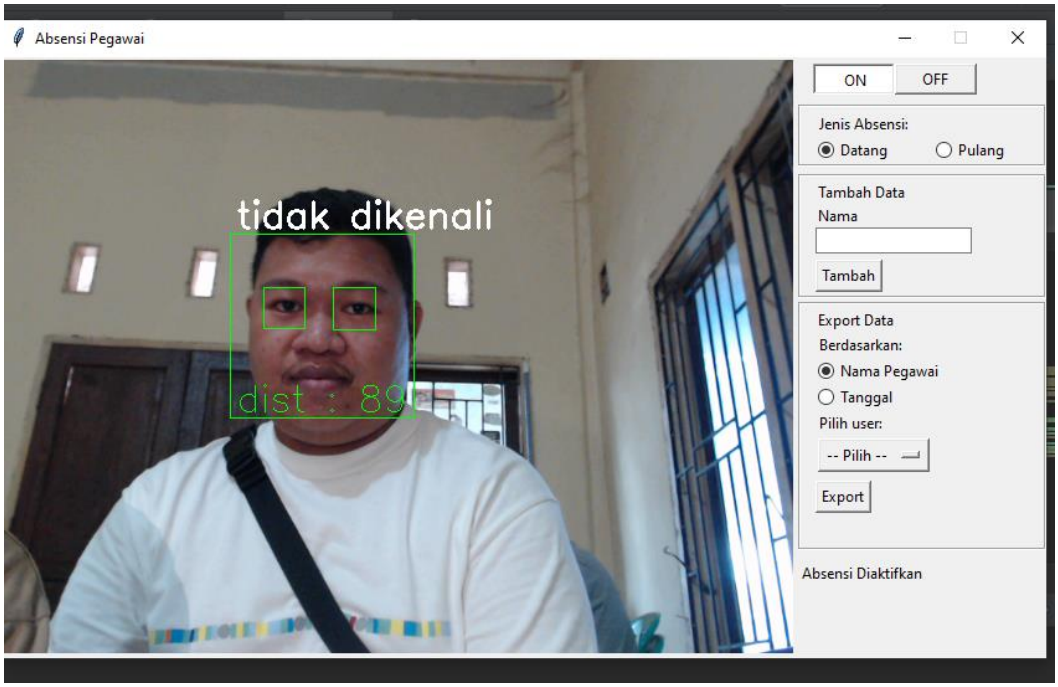

Absensi Pegawai
tidak dikenali
dist : 84
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
Absensi Diaktifkan


Absensi Pegawai
tidak dikenali
dist : 83
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan

3. Data 2


Absensi Pegawai
tidak dikenali
dist : 95
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan


Absensi Pegawai
tidak dikenali
dist : 89
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan


Absensi Pegawai
tidak dikenali
dist : 89
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan