Lampiran

Main.py

```
import sys
from PyQt5.QtWidgets import QApplication
from ui.main_window import MainWindow

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

main_window.py

```
from PyQt5.QtWidgets import QWidget, QVBoxLayout, QPushButton, QLabel
from PyQt5.QtGui import QFont
from PyQt5.QtCore import Qt

from ui.register_siswa import RegisterSiswaWindow
from ui.register_guru import RegisterGuruWindow
from ui.jadwal_window import JadwalWindow
from ui.training_window import TrainingWindow
from ui.absensi_window import AbsensiWindow
from ui.lihat_absensi import LihatAbsensiWindow
from ui.rekap_window import RekapWindow

class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Sistem Absensi Wajah - Admin")
        self.setGeometry(100, 100, 400, 500)

        layout = QVBoxLayout()

        title = QLabel("Sistem Absensi Wajah\nBerbasis Realtime")
        title.setAlignment(Qt.AlignCenter)
        title.setFont(QFont("Arial", 16, QFont.Bold))
        layout.addWidget(title)

        buttons = [
            ("📸 Register Wajah Siswa", RegisterSiswaWindow),
            ("👨‍🏫 Register Wajah Guru", RegisterGuruWindow),
            ("🗓️ Jadwal Pelajaran", JadwalWindow),
            ("🧠 Training Wajah", TrainingWindow),
            ("👷 Absensi (Realtime)", AbsensiWindow),
            ("📋 Lihat Absensi", LihatAbsensiWindow),
            ("📊 Rekapitulasi Absensi", RekapWindow),
        ]

        for text, window_class in buttons:
            btn = QPushButton(text)
            btn.setMinimumHeight(40)
            btn.clicked.connect(lambda checked, cls=window_class: self.buka_window(cls))
            layout.addWidget(btn)

        self.setLayout(layout)

    def buka_window(self, window_class):
        self.window = window_class()
        self.window.show()
```

register_siswa.py

```
import os
import cv2
from PyQt5.QtWidgets import (
    QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout,
    QMessageBox, QComboBox
)
from PyQt5.QtCore import QTimer
from PyQt5.QtGui import QIntValidator
from logic.face_utils import detect_faces, crop_face
```

```
SAVE_DIR = "data/siswa"
TOTAL_IMAGES = 300
CAMERA_INDEX = 1  # Ubah ini kalau kamera eksternal tidak terbaca

class RegisterSiswaWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Register Wajah Siswa")
        self.image_count = 0
        self.capture = None

        # Input
        self.nis_input = QLineEdit()
        self.nis_input.setPlaceholderText("NIS")
        self.nis_input.setValidator(QIntValidator())  # Hanya angka

        self.nama_input = QLineEdit()
        self.nama_input.setPlaceholderText("Nama Siswa")

        self.kelas_input = QLineEdit()
        self.kelas_input.setPlaceholderText("Kelas")

        self.jk_input = QComboBox()
        self.jk_input.addItems(["L", "P"])  # L = Laki-laki, P = Perempuan

        # Tombol mulai
        self.start_button = QPushButton("Mulai Ambil Wajah")
        self.start_button.clicked.connect(self.start_capture)

        self.info_label = QLabel("Isi data siswa lalu klik mulai.")

        # Layout
        self.layout = QVBoxLayout()
        self.layout.addWidget(QLabel("NIS:"))
        self.layout.addWidget(self.nis_input)
        self.layout.addWidget(QLabel("Nama Siswa:"))
        self.layout.addWidget(self.nama_input)
        self.layout.addWidget(QLabel("Kelas:"))
        self.layout.addWidget(self.kelas_input)
        self.layout.addWidget(QLabel("Jenis Kelamin:"))
        self.layout.addWidget(self.jk_input)
        self.layout.addWidget(self.start_button)
        self.layout.addWidget(self.info_label)

        self.setLayout(self.layout)

        # Timer
        self.timer = QTimer()
        self.timer.timeout.connect(self.update_frame)

    def start_capture(self):
        nis = self.nis_input.text().strip()
        nama = self.nama_input.text().strip()
        kelas = self.kelas_input.text().strip()
        jk = self.jk_input.currentText()

        if not nis or not nama or not kelas or not jk:
            QMessageBox.warning(self, "Peringatan", "Semua kolom harus diisi!")
            return

        # Format folder: NIS_Nama Siswa_Kelas_JK
        self.folder_name = f"{nis}_{nama}_{kelas}_{jk}"
        self.save_path = os.path.join(SAVE_DIR, self.folder_name)
        os.makedirs(self.save_path, exist_ok=True)
```

```
                self.capture = cv2.VideoCapture(CAMERA_INDEX)
        self.image_count = 0
        self.timer.start(30)
         self.info_label.setText("Mengambil gambar... (Tekan 'q' untuk berhenti)")

    def update_frame(self):
                ret, frame = self.capture.read()
        if not ret:
             self.info_label.setText("Gagal mengakses kamera.")
            return

                    boxes = detect_faces(frame)
        for box in boxes:
            x1, y1, x2, y2 = box
            # Tambahkan bounding box
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

              # Tambahkan teks jumlah gambar
             cv2.putText(frame, f"Gambar: {self.image_count}/{TOTAL_IMAGES}", (x1, y1 - 10),
                         cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

               # Simpan gambar wajah
            if self.image_count < TOTAL_IMAGES:
                        face = crop_face(frame, box)
                    img_path = os.path.join(self.save_path, f"{self.image_count + 1}.jpg")
                cv2.imwrite(img_path, face)
                self.image_count += 1
            else:
                self.timer.stop()
                self.capture.release()
                self.info_label.setText("Pengambilan wajah selesai.")
                QMessageBox.information(self, "Selesai", f"{TOTAL_IMAGES} wajah telah disimpan.")
                cv2.destroyAllWindows()
                return

         # Tampilkan frame dan cek tombol q
        cv2.imshow("Pendaftaran Wajah Siswa", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            self.timer.stop()
             self.capture.release()
             self.info_label.setText("Pengambilan dihentikan oleh pengguna.")
             QMessageBox.information(self, "Dihentikan", f"{self.image_count} gambar telah disimpan.")
             cv2.destroyAllWindows()

         def closeEvent(self, event):
        self.timer.stop()
        if self.capture:
             self.capture.release()
        cv2.destroyAllWindows()
```

register_guru.py

```
import os
import cv2
from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QMessageBox
from PyQt5.QtCore import QTimer
from logic.face_utils import detect_faces, crop_face, draw_bounding_box

SAVE_DIR = "data/guru"
TOTAL_IMAGES = 300
CAMERA_INDEX = 1  # Ubah jika kamera eksternal tidak terbaca
```

```
class RegisterGuruWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Register Wajah Guru")
        self.image_count = 0
        self.capture = None

        # Input
        self.nama_input = QLineEdit()
        self.nama_input.setPlaceholderText("Nama Guru")
        self.mapel_input = QLineEdit()
        self.mapel_input.setPlaceholderText("Mata Pelajaran")

        self.start_button = QPushButton("Mulai Ambil Wajah")
        self.start_button.clicked.connect(self.start_capture)

        self.info_label = QLabel("Isi data guru lalu klik mulai.")
        self.layout = QVBoxLayout()
        self.layout.addWidget(self.nama_input)
        self.layout.addWidget(self.mapel_input)
        self.layout.addWidget(self.start_button)
        self.layout.addWidget(self.info_label)

        self.setLayout(self.layout)

        # Timer
        self.timer = QTimer()
        self.timer.timeout.connect(self.update_frame)

    def start_capture(self):
        nama = self.nama_input.text().strip()
        mapel = self.mapel_input.text().strip()
        if not nama or not mapel:
            QMessageBox.warning(self, "Peringatan", "Semua kolom harus diisi!")
            return

        # Format: Nama Guru + "_" + Mata Pelajaran
        self.folder_name = f"{nama}_{mapel}"
        self.save_path = os.path.join(SAVE_DIR, self.folder_name)
        os.makedirs(self.save_path, exist_ok=True)

        self.capture = cv2.VideoCapture(CAMERA_INDEX)
        self.image_count = 0
        self.timer.start(30)
        self.info_label.setText("Mengambil gambar... Tekan 'q' untuk berhenti.")

    def update_frame(self):
        ret, frame = self.capture.read()
        if not ret:
            self.info_label.setText("Gagal mengakses kamera.")
            return

        boxes = detect_faces(frame)
        for box in boxes:
            draw_bounding_box(frame, box)
            x, y, w, h = box
            # Tambahkan teks jumlah gambar di atas bounding box
            cv2.putText(frame, f"{self.image_count}/{TOTAL_IMAGES}", (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

            if self.image_count < TOTAL_IMAGES:
                face = crop_face(frame, box)
```

```
                    img_path =
os.path.join(self.save_path,
f"{self.image_count + 1}.jpg")
                cv2.imwrite(im
g_path, face)
               self.image_count
+= 1
           else:
                self.timer.sto
p()
                self.capture.r
elease()
                self.info_labe
l.setText("Pengambilan    wajah
selesai.")
                QMessageBox.in
formation(self,      "Selesai",
f"{TOTAL_IMAGES}  wajah  telah
disimpan.")
                cv2.destroyAll
Windows()
               return

        # Tampilkan frame
        cv2.imshow("Pendaftaran
Wajah Guru", frame)

         # Deteksi tombol 'q'
untuk    menghentikan    secara
manual
       if cv2.waitKey(1) & 0xFF
== ord('q'):
            self.timer.stop()
             self.capture.relea
se()
             self.info_label.se
tText(f"Pengambilan
dihentikan.  {self.image_count}
gambar disimpan.")
             QMessageBox.inform
ation(self,       "Dihentikan",
f"Pengambilan
dihentikan.\n{self.image_count
} gambar telah disimpan.")
             cv2.destroyAllWind
ows()

        def    closeEvent(self,
event):
        self.timer.stop()
        if self.capture:
             self.capture.relea
se()
        cv2.destroyAllWindows()
```

jadwal_window.py

```
import os
import pandas as pd
from   PyQt5.QtWidgets   import
QWidget,   QLabel,   QLineEdit,
QPushButton,       QVBoxLayout,
QComboBox,         QMessageBox,
QTimeEdit
from PyQt5.QtCore import QTime

JADWAL_PATH                    =
"data/jadwal.xlsx"

class JadwalWindow(QWidget):
    def __init__(self):
        super().__init__()
         self.setWindowTitle("I
nput Jadwal Mata Pelajaran")
          self.setGeometry(200,
200, 400, 300)

        # Input
            self.guru_input  =
QLineEdit()
         self.guru_input.setPla
ceholderText("Nama Guru")

           self.mapel_input  =
QLineEdit()
         self.mapel_input.setPl
aceholderText("Mata
Pelajaran")

            self.hari_combo  =
QComboBox()
         self.hari_combo.addIte
ms(["Senin", "Selasa", "Rabu",
"Kamis", "Jumat", "Sabtu"])

             self.jam_mulai  =
QTimeEdit()
         self.jam_mulai.setDisp
layFormat("HH:mm")
         self.jam_mulai.setTime
(QTime.currentTime())

           self.jam_selesai  =
QTimeEdit()
         self.jam_selesai.setDi
splayFormat("HH:mm")
         self.jam_selesai.setTi
me(QTime.currentTime().addSecs
(3600))

          self.submit_button  =
QPushButton("Simpan Jadwal")
```

```
        self.submit_button.cli
cked.connect(self.simpan_jadwa
l)

        # Layout
        layout = QVBoxLayout()
        layout.addWidget(QLabe
l("Nama Guru:"))
        layout.addWidget(self.
guru_input)
        layout.addWidget(QLabe
l("Mata Pelajaran:"))
        layout.addWidget(self.
mapel_input)
        layout.addWidget(QLabe
l("Hari:"))
        layout.addWidget(self.
hari_combo)
        layout.addWidget(QLabe
l("Jam Mulai:"))
        layout.addWidget(self.
jam_mulai)
        layout.addWidget(QLabe
l("Jam Selesai:"))
        layout.addWidget(self.
jam_selesai)
        layout.addWidget(self.
submit_button)

        self.setLayout(layout)

    def simpan_jadwal(self):
                    guru    =
self.guru_input.text().strip()
                   mapel    =
self.mapel_input.text().strip(
)
                    hari    =
self.hari_combo.currentText()
                jam_mulai   =
self.jam_mulai.time().toString
("HH:mm")
              jam_selesai   =
self.jam_selesai.time().toStri
ng("HH:mm")

          if not guru or not
mapel:
            QMessageBox.warnin
g(self, "Peringatan", "Semua
kolom harus diisi!")
            return

        data = {
           "Nama Guru": [guru],
            "Mata Pelajaran":
[mapel],
           "Hari": [hari],
                "Jam Mulai":
[jam_mulai],
              "Jam Selesai":
[jam_selesai],
        }

                    df_new    =
pd.DataFrame(data)

                           if
os.path.exists(JADWAL_PATH):
                df_existing  =
pd.read_excel(JADWAL_PATH)
                df_combined  =
pd.concat([df_existing,
df_new], ignore_index=True)
        else:
           df_combined = df_new

        df_combined.to_excel(J
ADWAL_PATH, index=False)
        QMessageBox.informatio
n(self, "Berhasil", "Jadwal
berhasil disimpan.")
        self.close()

training_window.py
from PyQt5.QtWidgets import
QWidget, QPushButton, QLabel,
QVBoxLayout,      QMessageBox,
QFileDialog
from
matplotlib.backends.backend_qt
5agg import FigureCanvasQTAgg
as FigureCanvas
from matplotlib.figure import
Figure
from      logic.training_utils
import train_embeddings

class TrainingWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("T
raining Wajah")
          self.setGeometry(200,
200, 600, 500)

            self.info_label  =
QLabel("Pilih metode training
di bawah ini:")
```

```
        self.train_all_button = QPushButton("Training Semua Wajah")
        self.train_all_button.clicked.connect(self.training_semua)

        self.train_one_button = QPushButton("Training 1 Folder Wajah")
        self.train_one_button.clicked.connect(self.training_satu_folder)

        # Canvas untuk plotting matplotlib
        self.canvas = FigureCanvas(Figure(figsize=(5, 3)))
        self.ax = self.canvas.figure.subplots()

        layout = QVBoxLayout()
        layout.addWidget(self.info_label)
        layout.addWidget(self.train_all_button)
        layout.addWidget(self.train_one_button)
        layout.addWidget(self.canvas)

        self.setLayout(layout)

    def training_semua(self):
        print("=== Mulai Training Semua Wajah ===")
        self.info_label.setText("Proses training semua wajah sedang berjalan...")

        jumlah_siswa, jumlah_guru, history = train_embeddings()

        print(f"Training selesai: {jumlah_siswa} wajah siswa, {jumlah_guru} wajah guru")

        QMessageBox.information(
            self,
            "Training Selesai",
            f"Training semua selesai!\n\nSiswa: {jumlah_siswa} wajah\nGuru: {jumlah_guru} wajah"
        )

        if history:
            print("Menampilkan grafik akurasi dan loss...")
            self.plot_history(history)
        else:
            print("Tidak ada riwayat training (history tidak tersedia)")

        self.info_label.setText("Pilih metode training di bawah ini:")

    def training_satu_folder(self):
        folder_path = QFileDialog.getExistingDirectory(self, "Pilih Folder Wajah")
        if folder_path:
            print(f"=== Mulai Training Folder: {folder_path} ===")
            self.info_label.setText(f"Proses training folder: {folder_path}")

            jumlah, history = train_embeddings(folder_path=folder_path)

            print(f"Training folder selesai: {jumlah} wajah")

            QMessageBox.information(
                self,
                "Training Selesai",
                f"Training folder selesai!\n\nJumlah wajah: {jumlah}"
            )

            if history:
                print("Menampilkan grafik akurasi dan loss...")
                self.plot_history(history)
            else:
```

```
                print("Tidak ada
riwayat training (history tidak
tersedia)")

            self.info_label.se
tText("Pilih metode training di
bawah ini:")

       def  plot_history(self,
history):
        self.ax.clear()

            if  'accuracy'  in
history.history:
            self.ax.plot(histo
ry.history['accuracy'],
label='Train Accuracy')
          if 'val_accuracy' in
history.history:
            self.ax.plot(histo
ry.history['val_accuracy'],
label='Val Accuracy')
              if   'loss'   in
history.history:
            self.ax.plot(histo
ry.history['loss'],
label='Train Loss')
            if  'val_loss'  in
history.history:
            self.ax.plot(histo
ry.history['val_loss'],
label='Val Loss')

         self.ax.set_title("Tra
ining Accuracy & Loss")
         self.ax.set_xlabel("Ep
och")
         self.ax.set_ylabel("Va
lue")
        self.ax.legend()
        self.canvas.draw()
```

absensi_window.py

```
import os
import cv2
import pandas as pd
from datetime import datetime
from   PyQt5.QtWidgets   import
QWidget, QLabel, QVBoxLayout
from PyQt5.QtCore import QTimer
from  logic.face_utils  import
detect_faces,       crop_face,
recognize_face
from    logic.schedule_checker
import jadwal_aktif

# Konfigurasi global
CAMERA_INDEX = 1  # Ubah ke 0
jika pakai kamera internal, 1/2
kalau eksternal
ABSEN_PATH = "data/absensi"  #
Folder penyimpanan absensi

class AbsensiWindow(QWidget):
    """
     Jendela Absensi Otomatis
menggunakan Face Recognition.
    Deteksi wajah siswa/guru ->
catat absensi ke file Excel +
simpan foto.
    """

    def __init__(self):
        super().__init__()
         self.setWindowTitle("A
bsensi Otomatis")
          self.setGeometry(200,
200, 400, 200)

        # Label status
       self.label = QLabel("🔍
Mendeteksi wajah...")
        layout = QVBoxLayout()
         layout.addWidget(self.
label)
        self.setLayout(layout)

        # Inisialisasi kamera
              self.capture   =
cv2.VideoCapture(CAMERA_INDEX)
                    if    not
self.capture.isOpened():
            self.label.setText
("❌    Kamera    tidak    dapat
dibuka.")
            return

          # Timer untuk update
frame kamera
        self.timer = QTimer()
         self.timer.timeout.con
nect(self.update_frame)
        self.timer.start(30)

         # Set untuk menyimpan
wajah yang sudah tercatat
        self.tercatat = set()

    def update_frame(self):
          """Loop utama untuk
update  frame  dari  kamera  +
absensi otomatis."""
```

```
                ret, frame = self.capture.read()
            if not ret:
                self.label.setText("❌ Kamera tidak terdeteksi.")
                return

            # Ambil jadwal aktif
            jadwal = jadwal_aktif()
            if not jadwal:
                self.label.setText("⏰ Tidak ada jadwal aktif saat ini.")
                cv2.imshow("Absensi", frame)
                return

                    nama_guru = jadwal.get("Nama Guru", "Tidak Diketahui")
           mapel = jadwal.get("Mata Pelajaran", "Tidak Diketahui")
                    jam_mulai = jadwal.get("Jam Mulai", datetime.now().strftime("%H:%M:%S"))  # fallback jika tidak ada

              # Deteksi wajah pada frame
                        boxes = detect_faces(frame)
            if not boxes:
                self.label.setText("🔍 Tidak ada wajah terdeteksi.")
            else:
                for box in boxes:
                            face = crop_face(frame, box)
                    recognized = False

                            # Coba identifikasi wajah
                    for group in ["siswa", "guru"]:
                        nama, score = recognize_face(face, group=group)
                        if nama:
                            recognized = True
                            key = (nama, mapel)
                                # Tentukan warna bounding box
                            color = (0, 255, 0) if key in self.tercatat else (255, 255, 0)
                            text = f"{nama} ({score:.2f})"

                            # Gambar bounding box & teks
                            cv2.putText(frame, text, (box[0], box[1] - 10),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)
                            cv2.rectangle(frame, (box[0], box[1]), (box[2], box[3]), color, 2)

                            # Catat absensi jika belum
                            if key not in self.tercatat:
                                self.catat_absensi(nama, group, mapel, nama_guru, jam_mulai, frame)
                                self.tercatat.add(key)
                            break

                    # Jika wajah tidak dikenali
                    if not recognized:
                        cv2.putText(frame, "Unknown", (box[0], box[1] - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
                        cv2.rectangle(frame, (box[0], box[1]), (box[2], box[3]), (0, 0, 255), 2)

            # Tampilkan frame ke layar
            cv2.imshow("Absensi", frame)

        # Tekan 'q' untuk keluar
```

```
        if cv2.waitKey(1) & 0xFF
== ord('q'):
            print("[INFO] Tombol
'q'       ditekan.       Menutup
absensi...", flush=True)
            self.tutup_absensi()

        def  catat_absensi(self,
nama,   role,   mapel,   guru,
jam_pelajaran, frame):
        """
        Simpan data absensi ke
file Excel + foto.
        Format nama: NIS_NAMA
KELAS_JENIS KELAMIN
        """
        try:
            nis, nama_siswa,
kelas, jk = nama.split("_", 3)
        except ValueError:
            print(f"[WARNING]
Format   nama   tidak   sesuai:
{nama}")
            return

        tanggal    =
datetime.now().strftime("%Y-
%m-%d")
        jam_file    =
datetime.now().strftime("%H-
%M-%S")  # untuk nama file foto
        filename    =
os.path.join(ABSEN_PATH,
f"{tanggal}.xlsx")
        os.makedirs(ABSEN_PATH,
exist_ok=True)

        # Simpan foto wajah saat
absensi
        foto_filename   =
f"{nama}_{jam_file}.jpg"
        foto_path    =
os.path.join(ABSEN_PATH,
foto_filename)
        try:
            cv2.imwrite(foto_p
ath, frame)
        except Exception as e:
            print(f"[ERROR]
Gagal  menyimpan  foto  absensi:
{e}")

        # Data absensi dengan
kolom Foto Absensi
        new_data = {
            "NIS": nis,
            "Nama": nama_siswa,
            "JK": jk,
            "Kelas": kelas,
            "Nama Guru": guru,
            "Mata Pelajaran":
mapel,
            "Jam Pelajaran":
jam_pelajaran,  # diambil dari
jadwal
            "Tanggal": tanggal,
            "Status": "Hadir",
            "Foto Absensi":
foto_filename   # hanya nama
file, bisa path kalau mau
        }

        # Tambahkan ke file
Excel
        try:
            if
os.path.exists(filename):
                df  =
pd.read_excel(filename)
                df  =
pd.concat([df,
pd.DataFrame([new_data])],
ignore_index=True)
            else:
                df  =
pd.DataFrame([new_data])

            df.to_excel(filena
me, index=False)
        except Exception as e:
            print(f"[ERROR]
Gagal  menyimpan  absensi  ke
Excel: {e}")

        # Update status
        self.label.setText(f"✅
{nama_siswa} ({role}) berhasil
absen.")
        print(f"[ABSEN]
{nama_siswa}  ({role})  untuk
{mapel}     jam     pelajaran
{jam_pelajaran}", flush=True)

    def tutup_absensi(self):
        """Menutup  kamera,
timer, dan jendela absensi."""
        if  hasattr(self,
"timer"):
            self.timer.stop()
        if  hasattr(self,
"capture")                and
self.capture.isOpened():
```

```
            self.capture.relea
se()
        cv2.destroyAllWindows()
        self.close()
         print("[INFO] Jendela
absensi ditutup.", flush=True)

        def   closeEvent(self,
event):
         """Event saat jendela
ditutup manual."""
        self.tutup_absensi()


lihat_absensi.py
import os
import pandas as pd
from PyQt5.QtWidgets import (
        QWidget,   QVBoxLayout,
QTableWidget,
QTableWidgetItem,
      QPushButton,  QDateEdit,
QMessageBox,       QHBoxLayout,
QLabel
)
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import QDate,
Qt
from datetime import datetime

class
LihatAbsensiWindow(QWidget):
    def __init__(self):
        super().__init__()
         self.setWindowTitle("L
ihat & Hapus Absensi")
          self.setGeometry(200,
200, 1000, 500)

        layout = QVBoxLayout()

        # Pilih tanggal
             self.date_edit  =
QDateEdit()
         self.date_edit.setCale
ndarPopup(True)
         self.date_edit.setDate
(QDate.currentDate())
         layout.addWidget(self.
date_edit)

        # Tombol lihat
              self.load_btn  =
QPushButton("Lihat Absensi")
         self.load_btn.clicked.
connect(self.load_data)
         layout.addWidget(self.
load_btn)

        # Tabel absensi
                self.table   =
QTableWidget()
         layout.addWidget(self.
table)

        # Tombol hapus baris
               btn_layout   =
QHBoxLayout()
             self.delete_btn  =
QPushButton("Hapus        Baris
Dipilih")
         self.delete_btn.clicke
d.connect(self.hapus_baris)
         btn_layout.addWidget(s
elf.delete_btn)
         layout.addLayout(btn_l
ayout)

        self.setLayout(layout)

    def load_data(self):
                   tanggal    =
self.date_edit.date().toString
("yyyy-MM-dd")
                 file_path    =
f"data/absensi/{tanggal}.xlsx"

                     if    not
os.path.exists(file_path):
             QMessageBox.warnin
g(self, "Data Tidak Ditemukan",
f"Tidak    ada    data    untuk
{tanggal}")
            return

                    self.df    =
pd.read_excel(file_path)
         self.table.setRowCount
(len(self.df))
         self.table.setColumnCo
unt(len(self.df.columns))
         self.table.setHorizont
alHeaderLabels(self.df.columns
)

                  for    i    in
range(len(self.df)):
                    for   j   in
range(len(self.df.columns)):
                 column_name =
self.df.columns[j]
```

```
                if column_name
== "Foto":
                    foto_path =
str(self.df.iat[i, j])
                                if
os.path.exists(foto_path):
                            pixmap
=
QPixmap(foto_path).scaled(100,
100, Qt.KeepAspectRatio)
                        label =
QLabel()
                            label.
setPixmap(pixmap)
                            label.
setAlignment(Qt.AlignCenter)
                            self.t
able.setCellWidget(i, j, label)
                    else:
                            self.t
able.setItem(i,             j,
QTableWidgetItem("Foto    tidak
ditemukan"))
                else:
                    self.table
.setItem(i,                 j,
QTableWidgetItem(str(self.df.i
at[i, j])))

        self.table.resizeColum
nsToContents()
        self.table.resizeRowsT
oContents()

    def hapus_baris(self):
                    selected    =
self.table.currentRow()
        if selected == -1:
            QMessageBox.inform
ation(self,    "Info",    "Pilih
baris yang ingin dihapus")
            return

                    confirm    =
QMessageBox.question(self,
"Konfirmasi",    "Yakin    ingin
menghapus baris ini?")
                if   confirm   !=
QMessageBox.Yes:
            return

        self.df.drop(index=sel
ected, inplace=True)
                    tanggal    =
self.date_edit.date().toString
("yyyy-MM-dd")
                    file_path    =
f"data/absensi/{tanggal}.xlsx"
        self.df.reset_index(dr
op=True, inplace=True)
        self.df.to_excel(file_
path, index=False)

        self.load_data()

rekap_window.py
import os
import pandas as pd
from PyQt5.QtWidgets import (
       QWidget,   QVBoxLayout,
QLabel,          QTableWidget,
QTableWidgetItem,
     QPushButton,  QFileDialog,
QMessageBox,        QComboBox,
QApplication
)
from PyQt5.QtCore import Qt
from    reportlab.lib.pagesizes
import A4
from    reportlab.lib    import
colors
from reportlab.platypus import
SimpleDocTemplate,      Table,
TableStyle, Paragraph, Spacer
from      reportlab.lib.styles
import getSampleStyleSheet
import sys

class RekapWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("R
ekapitulasi Absensi Siswa")
         self.setGeometry(200,
200, 1000, 600)

        layout = QVBoxLayout()

        # Header Sekolah
       header = QLabel("<h2>SMK
MUHAMMADIYAH  PAREPARE<br>REKAP
ABSENSI SISWA</h2>")
        header.setAlignment(Qt
.AlignCenter)
        layout.addWidget(heade
r)

        # Input ComboBox Guru,
Mapel, Tahun Ajaran, Semester
            self.combo_guru  =
QComboBox()
```

```
            self.combo_mapel = QComboBox()
        self.combo_tahun_ajaran = QComboBox()
         self.combo_semester = QComboBox()

          # Tambahkan pilihan tahun ajaran
        self.combo_tahun_ajaran.addItems(["2023/2024", "2024/2025", "2025/2026"])
          # Tambahkan pilihan semester
        self.combo_semester.addItems(["Ganjil", "Genap"])

        layout.addWidget(QLabel("Nama Guru:"))
        layout.addWidget(self.combo_guru)
        layout.addWidget(QLabel("Mata Pelajaran:"))
        layout.addWidget(self.combo_mapel)
        layout.addWidget(QLabel("Tahun Ajaran:"))
        layout.addWidget(self.combo_tahun_ajaran)
        layout.addWidget(QLabel("Semester:"))
        layout.addWidget(self.combo_semester)

        # Muat jadwal Excel
        self.load_jadwal()

        # Tombol Load Rekap
             self.load_btn = QPushButton("Muat Rekap Absensi")
        self.load_btn.clicked.connect(self.muat_rekap)
        layout.addWidget(self.load_btn)

        # Tabel Absensi
                self.table = QTableWidget()
        layout.addWidget(self.table)

        # Tombol Ekspor Excel
            self.export_btn = QPushButton("Ekspor ke Excel")
        self.export_btn.clicked.connect(self.ekspor_data)
        layout.addWidget(self.export_btn)

        # Tombol Ekspor PDF
         self.export_pdf_btn = QPushButton("Ekspor ke PDF")
        self.export_pdf_btn.clicked.connect(self.ekspor_pdf)
        layout.addWidget(self.export_pdf_btn)

        # Footer (Paraf Guru)
       footer = QLabel("\nPARAF GURU")
        footer.setStyleSheet("font-weight: bold;")
        footer.setAlignment(Qt.AlignCenter)
        layout.addWidget(footer)

        self.setLayout(layout)

    def load_jadwal(self):
         """Load daftar guru & mapel dari file jadwal Excel"""
             jadwal_file = "data/jadwal.xlsx"
                    if not os.path.exists(jadwal_file):
            QMessageBox.warning(self, "Jadwal Tidak Ditemukan", f"File {jadwal_file} tidak ada.")
            return

        try:
                 df_jadwal = pd.read_excel(jadwal_file)
              required_cols = {"Nama Guru", "Mata Pelajaran"}
                     if not required_cols.issubset(df_jadwal.columns):
                QMessageBox.critical(self, "Error", f"File jadwal harus punya kolom: {', '.join(required_cols)}")
                return

            self.combo_guru.addItems(sorted(df_jadwal["Nama Guru"].dropna().unique().tolist()))
```

```
            self.combo_mapel.addItems(sorted(df_jadwal["Mata Pelajaran"].dropna().unique().tolist()))

        except Exception as e:
            QMessageBox.critical(self, "Error", f"Gagal membaca jadwal: {e}")

    def muat_rekap(self):
        folder = "data/absensi"
        all_data = []

            for filename in os.listdir(folder):
                if filename.endswith(".xlsx"):
                    filepath = os.path.join(folder, filename)
                    try:
                        df = pd.read_excel(filepath)
                        all_data.append(df)
                    except Exception as e:
                        print(f"Gagal membaca {filename}: {e}")
                        continue

        if not all_data:
            QMessageBox.warning(self, "Kosong", "Tidak ada data absensi ditemukan.")
            return

                    df_all = pd.concat(all_data, ignore_index=True)

       required_cols = {'Nama', 'NIS', 'JK', 'Kelas', 'Mata Pelajaran', 'Jam Pelajaran', 'Tanggal', 'Status', 'Nama Guru'}
                    if not required_cols.issubset(df_all.columns):
            QMessageBox.critical(self, "Error", f"File absensi harus memiliki kolom: {', '.join(required_cols)}")
            return

         # Filter sesuai guru, mapel, tahun ajaran, dan semester
                    guru = self.combo_guru.currentText()
                    mapel = self.combo_mapel.currentText()
                tahun_ajaran = self.combo_tahun_ajaran.currentText()
                    semester = self.combo_semester.currentText()

                df_filtered = df_all[(df_all['Nama Guru'] == guru) & (df_all['Mata Pelajaran'] == mapel)]

        if df_filtered.empty:
            QMessageBox.information(self, "Info", "Tidak ada data absensi untuk guru/mapel yang dipilih.")
            self.table.clear()
            return

         # Filter bulan sesuai semester
        df_filtered['Bulan'] = pd.to_datetime(df_filtered['Tanggal']).dt.month
       if semester == "Ganjil":
                df_filtered = df_filtered[df_filtered['Bulan'].between(7, 12)]
        else:  # Genap
                df_filtered = df_filtered[df_filtered['Bulan'].between(1, 6)]

        if df_filtered.empty:
            QMessageBox.information(self, "Info", "Tidak ada data absensi di semester yang dipilih.")
            self.table.clear()
            return

        # Hitung jumlah hadir
        hadir_count = (
            df_filtered[df_filtered['Status'] == "Hadir"]
              .groupby(['NIS', 'Nama', 'JK', 'Kelas']).size()
```

```
            .reset_index(name=
"Jumlah Kehadiran")
        )

        jam = df_filtered['Jam
Pelajaran'].iloc[0]   if   not
df_filtered.empty else ""

                  df_rekap    =
hadir_count.copy()
            df_rekap.insert(0,
"NO", range(1, len(df_rekap) +
1))
            df_rekap.insert(4,
"Mata Pelajaran", mapel)
        df_rekap.insert(5, "Jam
Pelajaran", jam)
            df_rekap.insert(6,
"Nama Guru", guru)
            df_rekap.insert(7,
"Tahun Ajaran", tahun_ajaran)
            df_rekap.insert(8,
"Semester", semester)

       self.df_rekap = df_rekap
                self.df_all   =
df_filtered  # simpan untuk PDF

         # Tampilkan di tabel
PyQt
        self.table.setRowCount
(len(df_rekap))
        self.table.setColumnCo
unt(len(df_rekap.columns))
        self.table.setHorizont
alHeaderLabels(df_rekap.column
s)

                  for    i    in
range(len(df_rekap)):
                    for   j   in
range(len(df_rekap.columns)):
                self.table.set
Item(i,                      j,
QTableWidgetItem(str(df_rekap.
iat[i, j])))

        self.table.resizeColum
nsToContents()

    def ekspor_data(self):
          if  not  hasattr(self,
'df_rekap'):
            QMessageBox.inform
ation(self, "Info", "Data belum
dimuat.")
            return

                  path,    _    =
QFileDialog.getSaveFileName(se
lf,       "Simpan       File",
"rekap_absensi.xlsx",    "Excel
Files (*.xlsx)")
        if path:
            self.df_rekap.to_e
xcel(path, index=False)
            QMessageBox.inform
ation(self,  "Sukses",  f"Data
berhasil diekspor ke:\n{path}")

    def ekspor_pdf(self):
          if  not  hasattr(self,
'df_rekap'):
            QMessageBox.inform
ation(self, "Info", "Data belum
dimuat.")
            return

                  path,    _    =
QFileDialog.getSaveFileName(se
lf,        "Simpan        PDF",
"rekap_absensi.pdf", "PDF Files
(*.pdf)")
        if not path:
            return

                      doc     =
SimpleDocTemplate(path,
pagesize=A4)
                   styles    =
getSampleStyleSheet()
        elements = []

        # Header
         elements.append(Paragr
aph("<b>ABSENSI    SISWA    SMK
MUHAMMADIYAH     PAREPARE</b>",
styles['Title']))
         elements.append(Spacer
(1, 12))

        # Info tambahan
         elements.append(Paragr
aph(f"Tahun      Ajaran      :
{self.combo_tahun_ajaran.curre
ntText()}", styles['Normal']))
         elements.append(Paragr
aph(f"Semester                :
{self.combo_semester.currentTe
xt()}", styles['Normal']))
         elements.append(Paragr
aph(f"Nama        Guru        :
```

```
{self.combo_guru.currentText()
}", styles['Normal']))
        elements.append(Paragr
aph(f"Mata      Pelajaran      :
{self.combo_mapel.currentText(
)}", styles['Normal']))
        elements.append(Paragr
aph(f"Jam      Pelajaran      :
{self.df_rekap['Jam
Pelajaran'].iloc[0]}",
styles['Normal']))
        elements.append(Paragr
aph(f"Kelas                  :
{self.df_rekap['Kelas'].iloc[0
]}", styles['Normal']))
        elements.append(Spacer
(1, 12))

         # Ambil semua tanggal
absensi unik
               tanggal_list   =
sorted(self.df_all['Tanggal'].
unique().tolist())

        # Buat header tabel
         table_header = ["NO",
"NIS", "NAMA SISWA", "JK"] +
tanggal_list     +     ["Jumlah
Kehadiran"]
                table_data   =
[table_header]

        # Isi data siswa
              for   i,   row   in
self.df_rekap.iterrows():
           nis, nama, jk, kelas
=   row["NIS"],   row["Nama"],
row["JK"], row["Kelas"]
            hadir_per_tanggal =
[]
                   for  tgl  in
tanggal_list:
                      subset =
self.df_all[(self.df_all['Tang
gal']      ==      tgl)      &
(self.df_all['NIS'] == nis)]
                       if  not
subset.empty                and
(subset['Status']           ==
"Hadir").any():
                    hadir_per_
tanggal.append("✓")
                else:
                    hadir_per_
tanggal.append("✗")
          jumlah = row["Jumlah
Kehadiran"]
            table_data.append(
[i+1,   nis,   nama,   jk]   +
hadir_per_tanggal + [jumlah])

        # Tambahkan baris Paraf
Guru
          paraf_row = ["", "",
"PARAF GURU", ""]  # kolom awal
(NO, NIS, Nama, JK)
        paraf_tanggal = []
       for tgl in tanggal_list:
                     subset   =
self.df_all[(self.df_all['Tang
gal']      ==      tgl)      &
(self.df_all['Nama  Guru']  ==
self.combo_guru.currentText())
]
           if not subset.empty:
                paraf_tanggal.
append("✓")
            else:
                paraf_tanggal.
append("")
                paraf_row   +=
paraf_tanggal + [""]  # kolom
terakhir   jumlah   kehadiran
kosong
         table_data.append(para
f_row)

        # Buat tabel PDF
                    table    =
Table(table_data,
repeatRows=1)
         table.setStyle(TableSt
yle([
                ('BACKGROUND',
(0,0),                  (-1,0),
colors.lightgrey),
             ('GRID', (0,0), (-
1,-1), 1, colors.black),
            ('ALIGN', (0,0), (-
1,-1), 'CENTER'),
              ('VALIGN', (0,0),
(-1,-1), 'MIDDLE'),
        ]))

        elements.append(table)
        doc.build(elements)

         QMessageBox.informatio
n(self,     "Sukses",     f"PDF
berhasil disimpan di:\n{path}")
```

```
if __name__ == "__main__":
                    app        =
QApplication(sys.argv)
    window = RekapWindow()
    window.show()
    sys.exit(app.exec_())

face_utils.py
import os
import cv2
import numpy as np
import pickle
from deepface import DeepFace
from ultralytics import YOLO
from datetime import datetime

# Load YOLO model untuk deteksi
wajah
model   =   YOLO("model/yolov8n-
face.pt")

# Parameter threshold
DETECTION_CONFIDENCE = 0.5
FACE_MATCH_THRESHOLD  =  0.6   #
Semakin kecil semakin ketat

# Path ke data embeddings
EMBEDDINGS_PATH                =
"data/embeddings/embeddings.pk
l"

# --- Fungsi deteksi wajah dari
frame ---
def detect_faces(frame):
                results        =
model.predict(source=frame,
conf=DETECTION_CONFIDENCE,
verbose=False)
    boxes = []
    for r in results:
                for    box    in
r.boxes.xyxy.cpu().numpy():
             x1, y1, x2, y2 =
map(int, box)
             boxes.append((x1,
y1, x2, y2))
    return boxes

# ---  Fungsi  ambil  embedding
wajah (FaceNet) ---
def get_embedding(face_img):
    try:
                 embedding    =
DeepFace.represent(face_img,
model_name='Facenet',
enforce_detection=False)[0]["e
mbedding"]
                          return
np.array(embedding)
    except:
        return None

# --- Simpan embeddings wajah
hasil training ke file ---
def
save_embeddings(data_dict):
    with open(EMBEDDINGS_PATH,
'wb') as f:
        pickle.dump(data_dict,
f)

#  ---  Load  embeddings  yang
sudah ditraining ---
def load_embeddings():
                  if        not
os.path.exists(EMBEDDINGS_PATH
):
        return {}
    with open(EMBEDDINGS_PATH,
'rb') as f:
        return pickle.load(f)

# --- Hitung cosine similarity
---
def cosine_similarity(a, b):
    a = np.array(a)
    b = np.array(b)
      return  np.dot(a,  b)  /
(np.linalg.norm(a)            *
np.linalg.norm(b))

# --- Fungsi untuk mencocokkan
wajah (return nama jika cocok)
---
def   recognize_face(face_img,
group='siswa'):
              embedding      =
get_embedding(face_img)
    if embedding is None:
        return None, 0.0

           all_embeddings     =
load_embeddings()
             candidates       =
all_embeddings.get(group, {})

    best_match = None
    best_score = 0

     for  name,  ref_embed  in
candidates.items():
```

```
                similarity = cosine_similarity(embedding, ref_embed)
            if similarity > best_score:
                best_score = similarity
            best_match = name

        if best_score > FACE_MATCH_THRESHOLD:
            return best_match, best_score
    else:
        return None, best_score

# --- Potong wajah dari frame ---
def crop_face(frame, box):
    x1, y1, x2, y2 = box
    return frame[y1:y2, x1:x2]

# --- Tambahkan bounding box ke frame ---
def draw_bounding_box(frame, box, label=""):
    x1, y1, x2, y2 = box
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
    if label:
        cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
```

schedule_checker.py

```
import pandas as pd
from datetime import datetime

def jadwal_aktif():
    try:
        df = pd.read_excel("data/jadwal.xlsx")
    except:
        return None

    now = datetime.now()
    hari_ini = now.strftime("%A")
    hari_indo = {
        "Monday": "Senin", "Tuesday": "Selasa", "Wednesday": "Rabu",
        "Thursday": "Kamis", "Friday": "Jumat", "Saturday": "Sabtu"
    }
    hari = hari_indo.get(hari_ini, "")

    jam_sekarang = now.strftime("%H:%M")

    for _, row in df.iterrows():
        if row["Hari"] == hari and row["Jam Mulai"] <= jam_sekarang <= row["Jam Selesai"]:
            return row.to_dict()

    return None
```

training_utils.py

```
import os
import cv2
import numpy as np
from logic.face_utils import get_embedding, save_embeddings

# Path folder wajah
SISWA_DIR = "data/siswa"
GURU_DIR = "data/guru"

# Ambil folder dalam path
def get_folders(path):
    return [name for name in os.listdir(path) if os.path.isdir(os.path.join(path, name))]

# Ambil semua wajah dari folder seseorang lalu hitung rata-rata embedding
def process_folder(folder_path):
    embeddings = []
    for img_name in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img_name)
        img = cv2.imread(img_path)
        if img is None:
            continue
        emb = get_embedding(img)
        if emb is not None:
            embeddings.append(emb)
```

```
    if embeddings:
                        return
np.mean(embeddings, axis=0)
    else:
        return None

# Fungsi training seluruh
folder siswa dan guru
def
train_embeddings(folder_path=N
one):
    data = {"siswa": {}, "guru":
{}}
            dummy_history     =
generate_dummy_history()

    if folder_path:  # Training
satu folder
                    emb     =
process_folder(folder_path)
        if emb is not None:
                     nama   =
os.path.basename(folder_path)
               if SISWA_DIR in
folder_path:
                 data["siswa"][
nama] = emb
             elif GURU_DIR in
folder_path:
                 data["guru"][n
ama] = emb
            save_embeddings(da
ta)
                    return  1,
dummy_history
        else:
                    return  0,
dummy_history

     # Training semua folder
siswa
          for    folder    in
get_folders(SISWA_DIR):
                        emb     =
process_folder(os.path.join(SI
SWA_DIR, folder))
        if emb is not None:
            data["siswa"][fold
er] = emb

    # Training semua folder guru
        for    folder    in
get_folders(GURU_DIR):
                    emb     =
process_folder(os.path.join(GU
RU_DIR, folder))
        if emb is not None:
           data["guru"][folder]
= emb

    save_embeddings(data)
            jumlah_siswa     =
len(data["siswa"])
             jumlah_guru     =
len(data["guru"])
        return   jumlah_siswa,
jumlah_guru, dummy_history

# Buat dummy history agar bisa
dipakai untuk plotting
class DummyHistory:
    def __init__(self):
        self.history = {
             "accuracy": [0.7,
0.8, 0.85, 0.87, 0.9],
               "val_accuracy":
[0.65, 0.75, 0.82, 0.84, 0.88],
            "loss": [0.8, 0.6,
0.4, 0.3, 0.2],
             "val_loss": [0.85,
0.65, 0.5, 0.35, 0.25]
        }

def generate_dummy_history():
    return DummyHistory()
```