2024

برنامج التحكم في الحضور والغياب مع ضبط الوقت

YOUNESS ZAINI

STE /R18M387 OZ

14/02/2024

```
import cv2
import tkinter as tk
from tkinter import ttk, messagebox
import qrcode
from PIL import Image, ImageTk
import datetime
import sqlite3
from openpyxl import Workbook
from fpdf import FPDF

class AttendanceControlApp:
    def __init__(self, root):
        self.root = root
        self.root.title(" YOUNESS ZAINI POINTEGE ATTANDANCE ")

        # Variables
        self.qr_code_data = tk.StringVar()
        self.check_in_time = None
        self.check_out_time = None
        self.total_time_var = tk.StringVar(value="Durée totale : non disponible")
        self.database_connection = sqlite3.connect("présence.db")
        self.create_table_if_not_exists()

        # GUI components
        self.label_qr_code = ttk.Label(root, text="QR Code:")
        self.entry_qr_code = ttk.Entry(root, textvariable=self.qr_code_data, width=30)
        self.btn_generate_qr = ttk.Button(root, text="Générer un code QR",
command=self.generate_qr_code)
        self.label_qr_image = ttk.Label(root)
        self.btn_start_scanning = ttk.Button(root, text="Démarrer la numérisation",
command=self.start_scanning)
        self.btn_change_camera = ttk.Button(root, text="Changer de caméra",
command=self.change_camera)
        self.btn_check_in_out = ttk.Button(root, text="Arrivée / Départ",
command=self.check_in_out)
        self.label_status = ttk.Label(root, text="Statut : Non enregistré/Non extrait")
        self.btn_generate_pdf = ttk.Button(root, text="Générer un rapport PDF",
command=self.generate_pdf_report)
        self.btn_generate_xlsx = ttk.Button(root, text="Générer un rapport XLSX",
command=self.generate_xlsx_report)
        self.label_total_time = ttk.Label(root, textvariable=self.total_time_var)
        self.title_label = ttk.Label(root, text="Système de contrôle de présence",
font=("Helvetica", 20, "bold"))

        # Treeview with modern styling
        self.treeview_style = ttk.Style()
        self.treeview_style.configure("Treeview", font=('Helvetica', 10))

        self.treeview = ttk.Treeview(root, columns=("ID", "QR Data", "Check-In Time",
"Check-Out Time"), show="headings", style="Treeview")
        self.treeview.heading("#1", text="ID")
        self.treeview.heading("#2", text="Données QR")
        self.treeview.heading("#3", text="Heure d'arrivée")
        self.treeview.heading("#4", text="Heure de départ")
        self.treeview.column("#1", width=50, anchor="center")
        self.treeview.column("#2", width=150, anchor="center")
        self.treeview.column("#3", width=150, anchor="center")
        self.treeview.column("#4", width=150, anchor="center")
        self.treeview.bind("<ButtonRelease-1>", self.on_treeview_click)
        self.refresh_treeview()

        # Layout
        self.title_label.grid(row=0, column=0, columnspan=5, pady=10)
        self.label_qr_code.grid(row=1, column=0, padx=10, pady=10)
        self.entry_qr_code.grid(row=1, column=1, padx=10, pady=10)
        self.btn_generate_qr.grid(row=1, column=2, padx=10, pady=10)
        self.label_qr_image.grid(row=2, column=0, columnspan=4, padx=10, pady=10)
        self.btn_start_scanning.grid(row=3, column=1, pady=10)
        self.btn_change_camera.grid(row=3, column=2, pady=10)
        self.btn_check_in_out.grid(row=4, column=1, pady=10)
        self.label_status.grid(row=5, column=0, columnspan=4, pady=5)
        self.label_total_time.grid(row=6, column=0, columnspan=4, pady=5)
        self.btn_generate_pdf.grid(row=7, column=0, pady=10)
        self.btn_generate_xlsx.grid(row=7, column=3, pady=10)
        self.treeview.grid(row=8, column=0, columnspan=5, pady=10)

        # Initialize camera source
        self.camera_source = 0  # Default camera source
        self.cap = None

    def create_table_if_not_exists(self):
        cursor = self.database_connection.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS attendance (
                            id INTEGER PRIMARY KEY AUTOINCREMENT,
                            qr_data TEXT,
                            check_in_time TEXT,
                            check_out_time TEXT)''')
        self.database_connection.commit()

    def generate_qr_code(self):
        data = self.qr_code_data.get()
        if data:
            qr = qrcode.QRCode(
                version=1,
                error_correction=qrcode.constants.ERROR_CORRECT_L,
                box_size=10,
                border=4,
            )
            qr.add_data(data)
            qr.make(fit=True)

            qr_code_image = qr.make_image(fill_color="black", back_color="white")
            qr_code_image.save("généré_qr.png")

            img = Image.open("généré_qr.png")
            img = img.resize((200, 200), Image.ANTIALIAS)
            img = ImageTk.PhotoImage(img)

            self.label_qr_image.config(image=img)
            self.label_qr_image.image = img
        else:
            messagebox.showinfo("Erreur", "Veuillez saisir les données du code QR.")

    def start_scanning(self):
        if self.cap is not None and not self.cap.isOpened():
            self.cap.release()

        self.cap = cv2.VideoCapture(self.camera_source)

        while True:
            ret, frame = self.cap.read()

            if not ret:
                messagebox.showinfo("Erreur", "Échec de la capture vidéo.")
                break

            detector = cv2.QRCodeDetector()
            data, vertices, qr_code = detector.detectAndDecode(frame)

            if data:
                messagebox.showinfo("Code QR scanné", f"Données: {data}")
                self.qr_code_data.set(data)
                self.check_in_out()
                break
            cv2.imshow("Scanner de codes QR", frame)

            if cv2.waitKey(1) & 0xFF == 27:  # Press 'Esc' to exit
                break

        if self.cap is not None:
            self.cap.release()

        cv2.destroyAllWindows()
        self.refresh_treeview()

    def change_camera(self):
        if self.cap is not None:
            self.cap.release()

        self.camera_source = (self.camera_source + 1) % 2  # Change camera source
        messagebox.showinfo("Caméra changée", f"Passé à l'appareil photo
{self.camera_source}")

    def check_in_out(self):
        data = self.qr_code_data.get()
        if data:
            cursor = self.database_connection.cursor()
            cursor.execute("SELECT * FROM attendance WHERE qr_data = ? ORDER BY id DESC
LIMIT 1", (data,))
            result = cursor.fetchone()

            if result is None or result[3] is not None:
                self.check_in()
            else:
                self.check_out()

    def check_in(self):
        data = self.qr_code_data.get()
        self.check_in_time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        cursor = self.database_connection.cursor()
        cursor.execute("INSERT INTO attendance (qr_data, check_in_time) VALUES (?, ?)",
(data, self.check_in_time))
        self.database_connection.commit()

        self.label_status.config(text=f"Statut : enregistré à {self.check_in_time}")
        self.calculate_total_time()
        self.refresh_treeview()

    def check_out(self):
        data = self.qr_code_data.get()
        self.check_out_time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        cursor = self.database_connection.cursor()
        cursor.execute("UPDATE attendance SET check_out_time = ? WHERE qr_data = ? AND
check_out_time IS NULL", (self.check_out_time, data))
        self.database_connection.commit()

        self.label_status.config(text=f"Statut : Récupéré à {self.check_out_time}")
        self.calculate_total_time()
        self.refresh_treeview()

    def calculate_total_time(self):
        if self.check_in_time and self.check_out_time:
            check_in_datetime = datetime.datetime.strptime(self.check_in_time, '%Y-%m-
%d %H:%M:%S')
            check_out_datetime = datetime.datetime.strptime(self.check_out_time, '%Y-
%m-%d %H:%M:%S')
            total_time = check_out_datetime - check_in_datetime
            self.total_time_var.set(f"Temps total: {str(total_time)}")
    def generate_pdf_report(self):
        filename = f"rapport_de_assiduité_{datetime.datetime.now().strftime('%Y-%m-
%d___%H-%M-%S')}.pdf"
        pdf = FPDF()
        pdf.add_page()
        pdf.set_font("Arial", size=12)

        cursor = self.database_connection.cursor()
        cursor.execute("SELECT * FROM attendance ORDER BY id")
        rows = cursor.fetchall()

        pdf.cell(200, 10, txt="Rapport de présence", ln=True, align='C')

        for row in rows:
            pdf.cell(50, 10, txt=str(row[0]), border=1)
            pdf.cell(50, 10, txt=row[1], border=1)
            pdf.cell(50, 10, txt=row[2], border=1)
            pdf.cell(50, 10, txt=row[3], border=1)
            pdf.ln()

        pdf.output('youness zaini.pdf')
        messagebox.showinfo("Rapport PDF généré", f"Rapport PDF enregistré sous
{filename}")

    def generate_xlsx_report(self):
        filename = f"rapport_de_assiduité_{datetime.datetime.now().strftime('%Y-%m-
%d___%H-%M-%S')}.xlsx"
        workbook = Workbook()
        sheet = workbook.active
        sheet.title = "Données de fréquentation"

        header = ["ID", "QR Data", "Check-In Time", "Check-Out Time"]
        sheet.append(header)

        cursor = self.database_connection.cursor()
        cursor.execute("SELECT * FROM attendance ORDER BY id")
        rows = cursor.fetchall()

        for row in rows:
            sheet.append(row)

        workbook.save('youness zaini.xlsx')
        messagebox.showinfo("Rapport XLSX généré", f"Rapport XLSX enregistré sous
{filename}")

    def refresh_treeview(self):
        cursor = self.database_connection.cursor()
        cursor.execute("SELECT * FROM attendance ORDER BY id")
        rows = cursor.fetchall()

        # Clear existing data in Treeview
        for item in self.treeview.get_children():
            self.treeview.delete(item)

        for row in rows:
            self.treeview.insert("", "end", values=row)

    def on_treeview_click(self, event):
        selected_item = self.treeview.selection()[0]
        qr_data = self.treeview.item(selected_item, "values")[1]

        confirmation = messagebox.askyesno("Supprimer l'enregistrement", f"Do you want
to delete the record with QR Data: {qr_data}?")
        if confirmation:
            self.delete_record()
    def delete_record(self):
        selected_item = self.treeview.selection()[0]
        record_id = self.treeview.item(selected_item, "values")[0]

        cursor = self.database_connection.cursor()
        cursor.execute("DELETE FROM attendance WHERE id = ?", (record_id,))
        self.database_connection.commit()

        messagebox.showinfo("Enregistrement supprimé", f"Enregistrer avec pièce
d'identité {record_id} supprimé.")
        self.refresh_treeview()

if __name__ == "__main__":
    root = tk.Tk()
    app = AttendanceControlApp(root)
    root.mainloop()
```