import sqlite3

from tkinter import \*

from tkinter import messagebox, simpledialog, ttk

from datetime import datetime

import csv

from PIL import Image, ImageTk

# Banco de dados

banco = sqlite3.connect('registro\_presenca.db')

cursor = banco.cursor()

# Tabelas

cursor.execute('''

    CREATE TABLE IF NOT EXISTS pessoas (

        matricula TEXT PRIMARY KEY,

        nome TEXT,

        sessao TEXT

    )

''')

cursor.execute('''

    CREATE TABLE IF NOT EXISTS registros (

        id INTEGER PRIMARY KEY AUTOINCREMENT,

        nome TEXT,

        sessao TEXT,

        data TEXT,

        hora TEXT,

        status TEXT

    )

''')

banco.commit()

# Função de presença

def registrar\_presenca(*status*):

    matricula = entry\_matricula.get()

    cursor.execute("SELECT nome, sessao FROM pessoas WHERE matricula = ?", (matricula,))

    resultado = cursor.fetchone()

    if resultado:

        nome, sessao = resultado

        agora = datetime.now()

        data = agora.strftime('%d/%m/%Y')

        hora = agora.strftime('%H:%M:%S')

        msg = f"Nome: {nome}\nSessão: {sessao}\nData: {data}\nHora: {hora}\nStatus: {status}\n\nConfirmar registro?"

        resposta = messagebox.askyesno("Confirmação", msg)

        if resposta:

            cursor.execute("INSERT INTO registros (nome, sessao, data, hora, status) VALUES (?, ?, ?, ?, ?)", (nome, sessao, data, hora, status))

            banco.commit()

            messagebox.showinfo("Registrado", f"{status} registrado com sucesso!")

        entry\_matricula.delete(0, END)

    else:

        messagebox.showerror("Erro", "Matrícula não encontrada!")

# Mostrar status em tempo real

def mostrar\_status():

    janela\_status = Toplevel()

    janela\_status.title("Status de Presença")

    tree = ttk.Treeview(janela\_status, *columns*=("Nome", "Sessão", "Status", "Data", "Hora"), *show*="headings")

    tree.heading("Nome", *text*="Nome")

    tree.heading("Sessão", *text*="Sessão")

    tree.heading("Status", *text*="Status")

    tree.heading("Data", *text*="Data")

    tree.heading("Hora", *text*="Hora")

    tree.pack(*fill*=BOTH, *expand*=True)

    def atualizar():

        tree.delete(\*tree.get\_children())

        cursor.execute('''

            SELECT nome, sessao, status, data, hora FROM registros

            WHERE (nome, data, id) IN (

                SELECT nome, data, MAX(id) FROM registros GROUP BY nome, data

            )

        ''')

        for row in cursor.fetchall():

            cor = "green" if row[2] == "Entrada" else "red"

            tree.insert('', END, *values*=row, *tags*=(cor,))

        tree.tag\_configure("green", *background*="#d4ffd4")

        tree.tag\_configure("red", *background*="#ffd4d4")

    atualizar()

    Button(janela\_status, *text*="Atualizar", *command*=atualizar).pack(*pady*=5)

# Exportar CSV

def exportar\_csv():

    cursor.execute("SELECT nome, sessao, data, hora, status FROM registros")

    dados = cursor.fetchall()

    with open("registros\_exportados.csv", "w", *newline*='', *encoding*='utf-8') as f:

        escritor = csv.writer(f)

        escritor.writerow(["Nome", "Sessão", "Data", "Hora", "Status"])

        escritor.writerows(dados)

    messagebox.showinfo("Exportado", "Arquivo 'registros\_exportados.csv' criado com sucesso!")

# Cadastrar nova pessoa

def cadastrar\_pessoa():

    janela\_cadastro = Toplevel()

    janela\_cadastro.title("Cadastrar Pessoa")

    Label(janela\_cadastro, *text*="Matrícula:").pack()

    entry\_mat = Entry(janela\_cadastro)

    entry\_mat.pack()

    Label(janela\_cadastro, *text*="Nome:").pack()

    entry\_nome = Entry(janela\_cadastro)

    entry\_nome.pack()

    Label(janela\_cadastro, *text*="Sessão:").pack()

    entry\_sessao = Entry(janela\_cadastro)

    entry\_sessao.pack()

    def salvar():

        mat = entry\_mat.get()

        nome = entry\_nome.get()

        sessao = entry\_sessao.get()

        if mat and nome and sessao:

            try:

                cursor.execute("INSERT INTO pessoas (matricula, nome, sessao) VALUES (?, ?, ?)", (mat, nome, sessao))

                banco.commit()

                messagebox.showinfo("Sucesso", "Pessoa cadastrada com sucesso!")

                janela\_cadastro.destroy()

            except sqlite3.IntegrityError:

                messagebox.showerror("Erro", "Matrícula já cadastrada!")

        else:

            messagebox.showerror("Erro", "Preencha todos os campos!")

    Button(janela\_cadastro, *text*="Salvar", *command*=salvar).pack(*pady*=10)

# Editar/Deletar pessoas

def gerenciar\_pessoas():

    janela\_gerenciar = Toplevel()

    janela\_gerenciar.title("Gerenciar Pessoas")

    tree = ttk.Treeview(janela\_gerenciar, *columns*=("Matrícula", "Nome", "Sessão"), *show*="headings")

    tree.heading("Matrícula", *text*="Matrícula")

    tree.heading("Nome", *text*="Nome")

    tree.heading("Sessão", *text*="Sessão")

    tree.pack(*fill*=BOTH, *expand*=True)

    def carregar\_dados():

        tree.delete(\*tree.get\_children())

        cursor.execute("SELECT matricula, nome, sessao FROM pessoas")

        for row in cursor.fetchall():

            tree.insert('', END, *values*=row)

    def deletar():

        item = tree.selection()

        if item:

            matricula = tree.item(item[0])['values'][0]

            if messagebox.askyesno("Confirmação", f"Deseja deletar a pessoa com matrícula {matricula}?"):

                cursor.execute("DELETE FROM pessoas WHERE matricula = ?", (matricula,))

                banco.commit()

                carregar\_dados()

    def editar():

        item = tree.selection()

        if item:

            mat, nome\_antigo, sessao\_antiga = tree.item(item[0])['values']

            nova\_janela = Toplevel()

            nova\_janela.title("Editar Pessoa")

            Label(nova\_janela, *text*="Novo Nome:").pack()

            entry\_nome = Entry(nova\_janela)

            entry\_nome.insert(0, nome\_antigo)

            entry\_nome.pack()

            Label(nova\_janela, *text*="Nova Sessão:").pack()

            entry\_sessao = Entry(nova\_janela)

            entry\_sessao.insert(0, sessao\_antiga)

            entry\_sessao.pack()

            def salvar\_edicao():

                novo\_nome = entry\_nome.get()

                nova\_sessao = entry\_sessao.get()

                cursor.execute("UPDATE pessoas SET nome = ?, sessao = ? WHERE matricula = ?", (novo\_nome, nova\_sessao, mat))

                banco.commit()

                nova\_janela.destroy()

                carregar\_dados()

            Button(nova\_janela, *text*="Salvar", *command*=salvar\_edicao).pack(*pady*=5)

    carregar\_dados()

    Button(janela\_gerenciar, *text*="Editar", *command*=editar).pack(*side*=LEFT, *padx*=10, *pady*=10)

    Button(janela\_gerenciar, *text*="Deletar", *command*=deletar).pack(*side*=LEFT, *padx*=10, *pady*=10)

# Acesso admin com senha

def acesso\_admin():

    senha = simpledialog.askstring("Área do Administrador", "Digite a senha:", *show*='\*')

    if senha == "admin123":

        janela\_admin = Toplevel()

        janela\_admin.title("Área do Administrador")

        Button(janela\_admin, *text*="Cadastrar Nova Pessoa", *command*=cadastrar\_pessoa).pack(*pady*=10)

        Button(janela\_admin, *text*="Gerenciar Pessoas", *command*=gerenciar\_pessoas).pack(*pady*=5)

        Button(janela\_admin, *text*="Ver Status em Tempo Real", *command*=mostrar\_status).pack(*pady*=5)

        Button(janela\_admin, *text*="Exportar CSV", *command*=exportar\_csv).pack(*pady*=5)

    else:

        messagebox.showerror("Acesso negado", "Senha incorreta!")

# Janela principal de registro

def abrir\_registro():

    janela.destroy()  # Fecha tela inicial

    global entry\_matricula

    janela\_registro = Tk()

    janela\_registro.title("Registro de Presença")

    Label(janela\_registro, *text*="Digite sua matrícula:").pack(*pady*=10)

    entry\_matricula = Entry(janela\_registro)

    entry\_matricula.pack(*pady*=5)

    entry\_matricula.focus()

    Button(janela\_registro, *text*="Registrar Entrada", *command*=lambda: registrar\_presenca("Entrada")).pack(*pady*=5)

    Button(janela\_registro, *text*="Registrar Saída", *command*=lambda: registrar\_presenca("Saída")).pack(*pady*=5)

    Button(janela\_registro, *text*="Área do Administrador", *command*=acesso\_admin).pack(*pady*=5)

    janela\_registro.mainloop()

# Janela Inicial

janela = Tk()

janela.title("Bem-vindo ao Sistema de Presença")

try:

    imagem = Image.open("logo.png")

    imagem = imagem.resize((200, 200))

    img\_logo = ImageTk.PhotoImage(imagem)

    Label(janela, *image*=img\_logo).pack(*pady*=10)

except:

    Label(janela, *text*="(Logo não carregado)", *fg*="red").pack()

Label(janela, *text*="Bem-vindo!\n\nEste sistema permite registrar sua entrada e saída.\nDigite sua matrícula e confirme.", *font*=("Arial", 12)).pack(*pady*=10)

Button(janela, *text*="Entrar", *command*=abrir\_registro).pack(*pady*=10)

janela.mainloop()

O erro que você está recebendo (ModuleNotFoundError: No module named 'PIL') ocorre porque a biblioteca **Pillow** (que substitui a antiga **PIL**) não está instalada no seu ambiente Python.

Para resolver isso, basta instalar o Pillow. Execute o seguinte comando no terminal ou prompt de comando:

bash

CopiarEditar

pip install Pillow

Depois de instalar o Pillow, o código deverá funcionar corretamente, já que ele usa a classe Image e ImageTk da biblioteca Pillow para carregar e exibir imagens no seu aplicativo Tkinter.

Deixe-me saber se precisar de mais alguma coisa!