# Secure User Profile Management System (SUPMS) With Tkinter GUI Version Code

import tkinter as tk  
from tkinter import ttk, messagebox  
import re  
import mysql.connector  
  
# -------------------- DATABASE CONNECTION --------------------  
conn\_obj = mysql.connector.connect(  
 host="localhost",  
 user="root",  
 password="67808033",  
 database="Python\_Project"  
)  
cur\_obj = conn\_obj.cursor()  
  
# -------------------- BACKEND FUNCTIONS --------------------  
def strong\_password(password):  
 return (  
 len(password) >= 6 and  
 re.search(r'[A-Z]', password) and  
 re.search(r'[a-z]', password) and  
 re.search(r'[0-9]', password) and  
 re.search(r'[\W\_]', password)  
 )  
  
def strong\_userid(user\_id):  
 return (  
 len(user\_id) >= 3 and  
 re.search(r'[A-Z]', user\_id) and  
 re.search(r'[0-9]', user\_id)  
 )  
  
def data\_entry\_sql(full\_name, address, phone\_number, user\_id, password):  
 sql = "INSERT INTO cust\_details (full\_name,address,phone\_number,user\_id,password) VALUES (%s, %s, %s, %s, %s)"  
 try:  
 cur\_obj.execute(sql, (full\_name, address, phone\_number, user\_id, password))  
 conn\_obj.commit()  
 return True  
 except mysql.connector.Error as e:  
 conn\_obj.rollback()  
 messagebox.showerror("Database Error", str(e))  
 return False  
  
def data\_retrieve(user\_id):  
 query = "SELECT \* FROM cust\_details WHERE user\_id=%s"  
 try:  
 cur\_obj.execute(query, (user\_id,))  
 return cur\_obj.fetchone()  
 except mysql.connector.Error as e:  
 conn\_obj.rollback()  
 messagebox.showerror("Database Error", str(e))  
 return None  
  
def data\_retrieve\_by\_phone\_and\_id(phone\_number, cust\_id):  
 query = "SELECT \* FROM cust\_details WHERE phone\_number=%s AND cust\_id=%s"  
 try:  
 cur\_obj.execute(query, (phone\_number, cust\_id))  
 return cur\_obj.fetchone()  
 except mysql.connector.Error as e:  
 conn\_obj.rollback()  
 messagebox.showerror("Database Error", str(e))  
 return None  
  
def update\_user\_details(cust\_id, name, address, phone, password):  
 qry = "UPDATE cust\_details SET full\_name=%s, address=%s, phone\_number=%s, password=%s WHERE cust\_id=%s"  
 try:  
 cur\_obj.execute(qry, (name, address, phone, password, cust\_id))  
 conn\_obj.commit()  
 return True  
 except mysql.connector.Error as e:  
 conn\_obj.rollback()  
 messagebox.showerror("Database Error", str(e))  
 return False  
  
def delete\_user(user\_id):  
 try:  
 cur\_obj.execute("DELETE FROM cust\_details WHERE user\_id=%s", (user\_id,))  
 conn\_obj.commit()  
 return True  
 except mysql.connector.Error as e:  
 conn\_obj.rollback()  
 messagebox.showerror("Database Error", str(e))  
 return False  
  
# -------------------- FRONTEND CLASSES --------------------  
class App(tk.Tk):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.title("Customer Management System")  
 self.geometry("650x450")  
 self.configure(bg="#f0f4f7")  
 self.style = ttk.Style(self)  
 self.style.theme\_use("clam")  
  
 self.grid\_rowconfigure(0, weight=1)  
 self.grid\_columnconfigure(0, weight=1)  
  
 self.frames = {}  
 for F in (MainMenu, LoginPage, RegisterPage, ProfileViewPage, UpdateProfilePage, DeleteAccountPage):  
 frame = F(self)  
 self.frames[F] = frame  
 frame.grid(row=0, column=0, sticky="nsew")  
  
 self.show\_frame(MainMenu)  
  
 def show\_frame(self, cont):  
 frame = self.frames[cont]  
 frame.tkraise()  
  
class MainMenu(ttk.Frame):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent, padding=20)  
 ttk.Label(self, text="Customer Management System", font=("Arial", 20, "bold")).pack(pady=20)  
  
 ttk.Button(self, text="Login", command=lambda: parent.show\_frame(LoginPage)).pack(pady=5)  
 ttk.Button(self, text="Register", command=lambda: parent.show\_frame(RegisterPage)).pack(pady=5)  
 ttk.Button(self, text="Profile View", command=lambda: parent.show\_frame(ProfileViewPage)).pack(pady=5)  
 ttk.Button(self, text="Update Profile", command=lambda: parent.show\_frame(UpdateProfilePage)).pack(pady=5)  
 ttk.Button(self, text="Delete Account", command=lambda: parent.show\_frame(DeleteAccountPage)).pack(pady=5)  
 ttk.Button(self, text="Exit", command=parent.destroy).pack(pady=10)  
  
class LoginPage(ttk.Frame):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent, padding=20)  
 ttk.Label(self, text="Login", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=2, pady=10)  
  
 ttk.Label(self, text="User ID:").grid(row=1, column=0, sticky="e", pady=5)  
 self.user\_id\_entry = ttk.Entry(self)  
 self.user\_id\_entry.grid(row=1, column=1, pady=5)  
  
 ttk.Label(self, text="Password:").grid(row=2, column=0, sticky="e", pady=5)  
 self.password\_entry = ttk.Entry(self, show="\*")  
 self.password\_entry.grid(row=2, column=1, pady=5)  
  
 ttk.Button(self, text="Login", command=self.login\_user).grid(row=3, column=0, columnspan=2, pady=10)  
 ttk.Button(self, text="Back", command=lambda: parent.show\_frame(MainMenu)).grid(row=4, column=0, columnspan=2)  
  
 def login\_user(self):  
 user\_id = self.user\_id\_entry.get()  
 password = self.password\_entry.get()  
 user\_details = data\_retrieve(user\_id)  
 if user\_details and user\_details[-1] == password:  
 messagebox.showinfo("Success", "Login Successful!")  
 else:  
 messagebox.showerror("Error", "Invalid credentials!")  
  
class RegisterPage(ttk.Frame):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent, padding=20)  
 ttk.Label(self, text="Register", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=2, pady=10)  
  
 labels = ["Full Name", "Address", "Phone Number", "User ID", "Password"]  
 self.entries = {}  
 for idx, label in enumerate(labels, start=1):  
 ttk.Label(self, text=label + ":").grid(row=idx, column=0, sticky="e", pady=5)  
 entry = ttk.Entry(self, show="\*" if label == "Password" else "")  
 entry.grid(row=idx, column=1, pady=5)  
 self.entries[label] = entry  
  
 ttk.Button(self, text="Register", command=self.register\_user).grid(row=len(labels)+1, column=0, columnspan=2, pady=10)  
 ttk.Button(self, text="Back", command=lambda: parent.show\_frame(MainMenu)).grid(row=len(labels)+2, column=0, columnspan=2)  
  
 def register\_user(self):  
 full\_name = self.entries["Full Name"].get().upper()  
 address = self.entries["Address"].get()  
 phone\_number = self.entries["Phone Number"].get()  
 user\_id = self.entries["User ID"].get()  
 password = self.entries["Password"].get()  
  
 if not (phone\_number.isdigit() and len(phone\_number) == 10):  
 messagebox.showerror("Error", "Invalid phone number!")  
 return  
 if not strong\_userid(user\_id):  
 messagebox.showerror("Error", "User ID must include uppercase and digit!")  
 return  
 if not strong\_password(password):  
 messagebox.showerror("Error", "Password must include uppercase, lowercase, digit, and special character!")  
 return  
  
 if data\_entry\_sql(full\_name, address, phone\_number, user\_id, password):  
 messagebox.showinfo("Success", "Registration successful!")  
 self.master.show\_frame(LoginPage)  
  
class ProfileViewPage(ttk.Frame):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent, padding=20)  
 ttk.Label(self, text="Profile View", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=2, pady=10)  
  
 ttk.Label(self, text="Phone Number:").grid(row=1, column=0, sticky="e", pady=5)  
 self.phone\_entry = ttk.Entry(self)  
 self.phone\_entry.grid(row=1, column=1, pady=5)  
  
 ttk.Label(self, text="Customer ID:").grid(row=2, column=0, sticky="e", pady=5)  
 self.cust\_id\_entry = ttk.Entry(self)  
 self.cust\_id\_entry.grid(row=2, column=1, pady=5)  
  
 ttk.Button(self, text="View Profile", command=self.view\_profile).grid(row=3, column=0, columnspan=2, pady=10)  
 ttk.Button(self, text="Back", command=lambda: parent.show\_frame(MainMenu)).grid(row=4, column=0, columnspan=2)  
  
 def view\_profile(self):  
 phone = self.phone\_entry.get()  
 cust\_id = self.cust\_id\_entry.get()  
 if not (phone.isdigit() and len(phone) == 10):  
 messagebox.showerror("Error", "Invalid phone number!")  
 return  
 if not cust\_id.isdigit():  
 messagebox.showerror("Error", "Customer ID must be digits!")  
 return  
 user = data\_retrieve\_by\_phone\_and\_id(phone, cust\_id)  
 if user:  
 messagebox.showinfo("Profile", f"Name: {user[1]}\nAddress: {user[2]}\nPhone: {user[3]}\nUser ID: {user[4]}")  
 else:  
 messagebox.showerror("Error", "No data found!")  
  
class UpdateProfilePage(ttk.Frame):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent, padding=20)  
 ttk.Label(self, text="Update Profile", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=2, pady=10)  
  
 self.user\_id\_entry = ttk.Entry(self)  
 self.pass\_entry = ttk.Entry(self, show="\*")  
 ttk.Label(self, text="User ID:").grid(row=1, column=0, sticky="e", pady=5)  
 self.user\_id\_entry.grid(row=1, column=1, pady=5)  
 ttk.Label(self, text="Password:").grid(row=2, column=0, sticky="e", pady=5)  
 self.pass\_entry.grid(row=2, column=1, pady=5)  
  
 self.name\_entry = ttk.Entry(self)  
 self.addr\_entry = ttk.Entry(self)  
 self.phone\_entry = ttk.Entry(self)  
 self.newpass\_entry = ttk.Entry(self, show="\*")  
  
 labels = ["New Full Name", "New Address", "New Phone", "New Password"]  
 entries = [self.name\_entry, self.addr\_entry, self.phone\_entry, self.newpass\_entry]  
 for idx, (lbl, ent) in enumerate(zip(labels, entries), start=3):  
 ttk.Label(self, text=lbl + ":").grid(row=idx, column=0, sticky="e", pady=5)  
 ent.grid(row=idx, column=1, pady=5)  
  
 ttk.Button(self, text="Update", command=self.update\_profile).grid(row=7, column=0, columnspan=2, pady=10)  
 ttk.Button(self, text="Back", command=lambda: parent.show\_frame(MainMenu)).grid(row=8, column=0, columnspan=2)  
  
 def update\_profile(self):  
 user\_id = self.user\_id\_entry.get()  
 password = self.pass\_entry.get()  
 user\_details = data\_retrieve(user\_id)  
 if not user\_details or user\_details[-1] != password:  
 messagebox.showerror("Error", "Invalid login!")  
 return  
 cust\_id = user\_details[0]  
 new\_name = self.name\_entry.get().upper() or user\_details[1]  
 new\_addr = self.addr\_entry.get() or user\_details[2]  
 new\_phone = self.phone\_entry.get() or user\_details[3]  
 if new\_phone and (not new\_phone.isdigit() or len(new\_phone) != 10):  
 messagebox.showerror("Error", "Invalid phone number!")  
 return  
 new\_pass = self.newpass\_entry.get() or user\_details[-1]  
 if new\_pass != user\_details[-1] and not strong\_password(new\_pass):  
 messagebox.showerror("Error", "Weak password!")  
 return  
 if update\_user\_details(cust\_id, new\_name, new\_addr, new\_phone, new\_pass):  
 messagebox.showinfo("Success", "Profile updated!")  
  
class DeleteAccountPage(ttk.Frame):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent, padding=20)  
 ttk.Label(self, text="Delete Account", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=2, pady=10)  
  
 ttk.Label(self, text="User ID:").grid(row=1, column=0, sticky="e", pady=5)  
 self.user\_id\_entry = ttk.Entry(self)  
 self.user\_id\_entry.grid(row=1, column=1, pady=5)  
  
 ttk.Label(self, text="Password:").grid(row=2, column=0, sticky="e", pady=5)  
 self.password\_entry = ttk.Entry(self, show="\*")  
 self.password\_entry.grid(row=2, column=1, pady=5)  
  
 ttk.Button(self, text="Delete Account", command=self.delete\_account).grid(row=3, column=0, columnspan=2, pady=10)  
 ttk.Button(self, text="Back", command=lambda: parent.show\_frame(MainMenu)).grid(row=4, column=0, columnspan=2)  
  
 def delete\_account(self):  
 user\_id = self.user\_id\_entry.get()  
 password = self.password\_entry.get()  
 user\_details = data\_retrieve(user\_id)  
 if not user\_details or user\_details[-1] != password:  
 messagebox.showerror("Error", "Invalid login!")  
 return  
 if messagebox.askyesno("Confirm", "Are you sure you want to delete your account?"):  
 if delete\_user(user\_id):  
 messagebox.showinfo("Deleted", "Account deleted successfully!")  
 self.master.show\_frame(MainMenu)  
  
# -------------------- RUN APP --------------------  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = App()  
 app.mainloop()  
 conn\_obj.close()