from tkinter import \*  
from tkinter import messagebox, ttk  
  
  
class Gasan:  
 def \_\_init\_\_(self, username, password, email, confirm\_password):  
 self.username = username  
 self.password = password  
 self.email = email  
 self.confirm\_password = confirm\_password  
  
 def \_\_str\_\_(self):  
 return f"Username: {self.username}, Email: {self.email}"  
  
  
class AutoGasStationFrame(Frame):  
 def \_\_init\_\_(self, master):  
 super().\_\_init\_\_(master)  
 self.master = master  
 self.master.title("Auto Gas Station")  
 self.master.geometry('800x600')  
 self.master.resizable(False, False)  
 self.colors = ["black", "red", "blue", "green", "purple", "orange"]  
 self.fonts = ["Arial", "Helvetica", "Times New Roman", "Courier", "Verdana"]  
  
 self.users = {  
 "admin12": "admin123",  
 "sardar": "password123",  
 }  
  
 self.wallet\_balance = 1000  
 self.products = {  
 "Diesel": 5.0,  
 "92": 6.0,  
 "Premium": 7.0,  
 "Super": 8.0,  
 "Chips": 2.5,  
 "Soda": 1.5,  
 "Chocolate": 3.0,  
 "Gum": 0.5  
 }  
 self.snacks = {  
 "Chips": 2.5,  
 "Soda": 1.5,  
 "Chocolate": 3.0,  
 "Gum": 0.5  
 }  
  
 self.current\_user = None  
 self.current\_window = None  
 self.cart\_items = []  
  
 self.welcome\_label = Label(self, text="Welcome to AutoGasStation", font=("Arial", 24, "bold"), bg="lightgray")  
 self.welcome\_label.pack(pady=20)  
  
 self.wallet\_label = Label(self, text=f"Wallet: ${self.wallet\_balance:.2f}", font=("Arial", 12, "bold"),  
 bg="lightgray")  
 self.wallet\_label.pack()  
  
 self.create\_menu()  
 self.login\_window()  
  
 self.set\_light\_theme()  
  
 def update\_wallet\_label(self):  
 if self.wallet\_label.winfo\_exists(): # Check if the label still exists  
 self.wallet\_label.config(text=f"Wallet: ${self.wallet\_balance:.2f}")  
  
 def create\_menu(self):  
 menu = Menu(self.master)  
 self.master.config(menu=menu)  
  
 file\_menu = Menu(menu, tearoff=0)  
 menu.add\_cascade(label="Registration", menu=file\_menu)  
 file\_menu.add\_command(label="Login", command=self.login\_window)  
 file\_menu.add\_command(label="Register", command=self.register\_window)  
 file\_menu.add\_command(label="Exit", command=self.master.quit)  
  
 self.view\_menu = Menu(menu, tearoff=0)  
 menu.add\_cascade(label="View", menu=self.view\_menu)  
 self.view\_menu.add\_command(label="Show Products", command=self.show\_products, state=DISABLED)  
 self.view\_menu.add\_command(label="View Cart", command=self.view\_cart, state=DISABLED)  
 self.view\_menu.add\_command(label="Profile", command=self.show\_profile\_window, state=DISABLED)  
 self.view\_menu.add\_command(label="Settings", command=self.settings\_window, state=DISABLED)  
 self.view\_menu.add\_command(label="Buy", command=self.buy\_product, state=DISABLED)  
 self.view\_menu.add\_command(label="My Wallet", command=self.show\_wallet, state=DISABLED)  
 self.view\_menu.add\_command(label="Edit Profile", command=self.edit\_profile\_window,  
 state=DISABLED)   
  
 def set\_light\_theme(self):  
 self.master.config(bg="white")  
 self.config(bg="white")  
  
 for widget in self.winfo\_children():  
 widget.config(bg="white", fg="black", font=("Arial", 12))  
  
 def set\_dark\_theme(self):  
 self.master.config(bg="black")  
 self.config(bg="black")  
  
 for widget in self.winfo\_children():  
 widget.config(bg="black", fg="white", font=("Arial", 12))  
  
 def show\_products(self):  
 self.clear\_frame()  
 products\_label = Label(self, text="Available Products and Prices:")  
 products\_label.pack()  
  
 for product, price in self.products.items():  
 product\_info\_label = Label(self, text=f"{product}: ${price:.2f}")  
 product\_info\_label.pack()  
  
 snacks\_label = Label(self, text="Available Snacks and Prices:")  
 snacks\_label.pack()  
  
 for snack, price in self.snacks.items():  
 snack\_info\_label = Label(self, text=f"{snack}: ${price:.2f}")  
 snack\_info\_label.pack()  
  
 def login\_window(self):  
 self.clear\_frame()  
  
 self.username\_label = Label(self, text="Username:")  
 self.username\_label.pack()  
  
 self.username\_entry = Entry(self)  
 self.username\_entry.pack()  
  
 self.password\_label = Label(self, text="Password:")  
 self.password\_label.pack()  
  
 self.password\_entry = Entry(self, show="\*")  
 self.password\_entry.pack()  
  
 self.login\_button = Button(self, text="Login", command=self.login)  
 self.login\_button.pack()  
  
 self.show\_password\_button = Button(self, text="Show Password", command=self.show\_password)  
 self.show\_password\_button.pack()  
  
 self.current\_window = self.login\_window  
  
 def register\_window(self):  
 self.clear\_frame()  
  
 self.username\_label = Label(self, text="Username:")  
 self.username\_label.pack()  
  
 self.username\_entry = Entry(self)  
 self.username\_entry.pack()  
  
 self.email\_label = Label(self, text="Email:")  
 self.email\_label.pack()  
  
 self.email\_entry = Entry(self)  
 self.email\_entry.pack()  
  
 self.password\_label = Label(self, text="Password:")  
 self.password\_label.pack()  
  
 self.password\_entry = Entry(self, show="\*")  
 self.password\_entry.pack()  
  
 self.confirm\_password\_label = Label(self, text="Confirm Password:")  
 self.confirm\_password\_label.pack()  
  
 self.confirm\_password\_entry = Entry(self, show="\*")  
 self.confirm\_password\_entry.pack()  
  
 self.show\_password\_button = Button(self, text="Show Password", command=self.show\_password\_register)  
 self.show\_password\_button.pack()  
  
 self.register\_button = Button(self, text="Register", command=self.perform\_register\_user)  
 self.register\_button.pack()  
  
 self.current\_window = self.register\_window  
  
 def show\_password\_register(self):  
 current\_show\_state = self.password\_entry.cget("show")  
 if current\_show\_state == "":  
 self.password\_entry.configure(show="\*")  
 self.confirm\_password\_entry.configure(show="\*")  
 else:  
 self.password\_entry.configure(show="")  
 self.confirm\_password\_entry.configure(show="")  
  
 def admin\_window(self):  
 self.clear\_frame()  
  
 self.background\_label = Label(self, bg="lightyellow")  
 self.background\_label.place(relwidth=1, relheight=1)  
  
 self.add\_user\_button = Button(self, text="Add User", command=self.add\_user)  
 self.add\_user\_button.pack()  
  
 self.delete\_user\_button = Button(self, text="Delete User", command=self.delete\_user)  
 self.delete\_user\_button.pack()  
  
 self.view\_users\_button = Button(self, text="View Users", command=self.view\_users\_window)  
 self.view\_users\_button.pack()  
  
 self.view\_menu.entryconfig("Edit Profile", state=NORMAL)  
  
 self.add\_product\_button = Button(self, text="Add Product", command=self.add\_product)  
 self.add\_product\_button.pack()  
  
 self.modify\_price\_button = Button(self, text="Modify Price", command=self.modify\_price\_window)  
 self.modify\_price\_button.pack()  
  
 self.delete\_product\_button = Button(self, text="Delete Product", command=self.delete\_product\_window)  
 self.delete\_product\_button.pack()  
  
 self.view\_menu.entryconfig("Buy", state=NORMAL)  
  
 self.current\_window = self.admin\_window  
  
 def view\_users\_window(self):  
 users\_window = Toplevel(self)  
 users\_window.title("View Users")  
  
 users\_label = Label(users\_window, text="List of Users:")  
 users\_label.pack()  
  
 users\_listbox = Listbox(users\_window, width=40, height=10)  
 users\_listbox.pack()  
  
 for username in self.users.keys():  
 users\_listbox.insert(END, username)  
  
 def user\_window(self):  
 self.clear\_frame()  
  
 self.background\_label = Label(self, bg="lightgreen")  
 self.background\_label.place(relwidth=1, relheight=1)  
  
 self.view\_menu.entryconfig("Buy", state=NORMAL)  
  
 self.show\_products\_button = Button(self, text="Show Products", command=self.show\_products)  
 self.show\_products\_button.pack()  
  
 self.cart\_button = Button(self, text="View Cart", command=self.view\_cart)  
 self.cart\_button.pack()  
  
 self.profile\_button = Button(self, text="Profile", command=self.show\_profile\_window)  
 self.profile\_button.pack()  
  
 self.view\_menu.entryconfig("Edit Profile", state=NORMAL)  
  
 self.current\_window = self.user\_window  
  
 def settings\_window(self):  
 self.clear\_frame()  
  
 self.font\_var = StringVar(value="Arial")  
 self.text\_color\_var = StringVar(value="black")  
 self.theme\_var = StringVar(value="light")  
  
 self.font\_label = Label(self, text="Font:")  
 self.font\_label.pack()  
  
 self.font\_entry = Entry(self, textvariable=self.font\_var)  
 self.font\_entry.pack()  
  
 self.text\_color\_label = Label(self, text="Text Color:")  
 self.text\_color\_label.pack()  
  
 self.text\_color\_entry = Entry(self, textvariable=self.text\_color\_var)  
 self.text\_color\_entry.pack()  
  
 self.theme\_label = Label(self, text="Theme:")  
 self.theme\_label.pack()  
  
 self.theme\_light\_button = Radiobutton(self, text="Light", variable=self.theme\_var, value="light")  
 self.theme\_light\_button.pack()  
  
 self.theme\_dark\_button = Radiobutton(self, text="Dark", variable=self.theme\_var, value="dark")  
 self.theme\_dark\_button.pack()  
  
 self.save\_settings\_button = Button(self, text="Save Settings", command=self.save\_settings)  
 self.save\_settings\_button.pack()  
  
 self.current\_window = self.settings\_window  
  
 def clear\_frame(self):  
 for widget in self.winfo\_children():  
 widget.destroy()  
  
 def login(self):  
 username = self.username\_entry.get()  
 password = self.password\_entry.get()  
  
 if self.authenticate\_user(username, password):  
 self.current\_user = username  
 if self.is\_admin(username):  
 self.admin\_window()  
 else:  
 self.user\_window()  
  
 self.view\_menu.entryconfig("Show Products", state=NORMAL)  
 self.view\_menu.entryconfig("View Cart", state=NORMAL)  
 self.view\_menu.entryconfig("Profile", state=NORMAL)  
 self.view\_menu.entryconfig("Settings", state=NORMAL)  
 self.view\_menu.entryconfig("Buy", state=NORMAL)  
 self.view\_menu.entryconfig("My Wallet", state=NORMAL)  
 else:  
 messagebox.showerror("Login Error", "Invalid username or password")  
  
 def authenticate\_user(self, username, password):  
 return username in self.users and self.users[username] == password  
  
 def is\_admin(self, username):  
 return username == "admin12" or username == "admin123"  
  
 def perform\_register\_user(self):  
 username = self.username\_entry.get()  
 email = self.email\_entry.get()  
 password = self.password\_entry.get()  
 confirm\_password = self.confirm\_password\_entry.get()  
  
 new\_user = Gasan(username, password, email, confirm\_password)  
  
 if username == "" or email == "" or password == "" or confirm\_password == "":  
 messagebox.showerror("Registration Error", "All fields are required")  
 return  
  
 if password != confirm\_password:  
 messagebox.showerror("Registration Error", "Passwords do not match")  
 return  
  
 if "@" not in email or "." not in email:  
 messagebox.showerror("Registration Error", "Please enter a valid email address")  
 return  
  
 if username in self.users:  
 messagebox.showerror("Registration Error", "Username Already exists")  
 return  
  
 self.users[username] = password  
  
 messagebox.showinfo("Registration Successful", "User registered successfully")  
 self.login\_window()  
  
 def show\_password(self):  
 current\_show\_state = self.password\_entry.cget("show")  
 if current\_show\_state == "":  
 self.password\_entry.configure(show="\*")  
 else:  
 self.password\_entry.configure(show="")  
  
 def add\_user(self):  
 add\_user\_window = Toplevel(self)  
 add\_user\_window.title("Add User")  
  
 username\_label = Label(add\_user\_window, text="Username:")  
 username\_label.pack()  
  
 username\_entry = Entry(add\_user\_window)  
 username\_entry.pack()  
  
 password\_label = Label(add\_user\_window, text="Password:")  
 password\_label.pack()  
  
 password\_entry = Entry(add\_user\_window, show="\*")  
 password\_entry.pack()  
  
 add\_button = Button(add\_user\_window, text="Add User",  
 command=lambda: self.perform\_add\_user(username\_entry.get(), password\_entry.get()))  
 add\_button.pack()  
  
 def perform\_add\_user(self, username, password):  
 if username == "" or password == "":  
 messagebox.showerror("Add User Error", "Username and password are required")  
 return  
  
 if username in self.users:  
 messagebox.showerror("Add User Error", "Username already exists")  
 return  
  
 self.users[username] = password  
  
 messagebox.showinfo("Add User Successful", "User added successfully")  
  
 def delete\_user(self):  
 delete\_user\_window = Toplevel(self)  
 delete\_user\_window.title("Delete User")  
  
 username\_label = Label(delete\_user\_window, text="Username:")  
 username\_label.pack()  
  
 username\_entry = Entry(delete\_user\_window)  
 username\_entry.pack()  
  
 delete\_button = Button(delete\_user\_window, text="Delete User",  
 command=lambda: self.perform\_delete\_user(username\_entry.get()))  
 delete\_button.pack()  
  
 def perform\_delete\_user(self, username):  
 if username == "":  
 messagebox.showerror("Delete User Error", "Username is required")  
 return  
  
 if username not in self.users:  
 messagebox.showerror("Delete User Error", "Username does not exist")  
 return  
  
 if username == self.current\_user:  
 messagebox.showerror("Delete User Error", "You cannot delete your own account")  
 return  
  
 del self.users[username]  
  
 messagebox.showinfo("Delete User Successful", "User deleted successfully")  
  
 def edit\_profile\_window(self):  
 edit\_profile\_window = Toplevel(self)  
 edit\_profile\_window.title("Edit Profile")  
  
 current\_user\_label = Label(edit\_profile\_window, text=f"Current User: {self.current\_user}")  
 current\_user\_label.pack()  
  
 new\_password\_label = Label(edit\_profile\_window, text="New Password:")  
 new\_password\_label.pack()  
  
 new\_password\_entry = Entry(edit\_profile\_window, show="\*")  
 new\_password\_entry.pack()  
  
 confirm\_new\_password\_label = Label(edit\_profile\_window, text="Confirm New Password:")  
 confirm\_new\_password\_label.pack()  
  
 confirm\_new\_password\_entry = Entry(edit\_profile\_window, show="\*")  
 confirm\_new\_password\_entry.pack()  
  
 save\_button = Button(edit\_profile\_window, text="Save", command=lambda: self.save\_new\_password(  
 new\_password\_entry.get(), confirm\_new\_password\_entry.get(), edit\_profile\_window))  
 save\_button.pack()  
  
 def save\_new\_password(self, new\_password, confirm\_password, window):  
 if new\_password != confirm\_password:  
 messagebox.showerror("Password Error", "Passwords do not match.")  
 elif new\_password == "":  
 messagebox.showerror("Password Error", "New password cannot be empty.")  
 else:  
 self.users[self.current\_user] = new\_password  
 messagebox.showinfo("Password Change", "Password changed successfully.")  
 window.destroy()  
 def add\_product(self):  
 add\_product\_window = Toplevel(self)  
 add\_product\_window.title("Add Product")  
  
 product\_name\_label = Label(add\_product\_window, text="Product Name:")  
 product\_name\_label.pack()  
  
 product\_name\_entry = Entry(add\_product\_window)  
 product\_name\_entry.pack()  
  
 price\_label = Label(add\_product\_window, text="Price:")  
 price\_label.pack()  
  
 price\_entry = Entry(add\_product\_window)  
 price\_entry.pack()  
  
 add\_button = Button(add\_product\_window, text="Add Product",  
 command=lambda: self.perform\_add\_product(product\_name\_entry.get(), price\_entry.get()))  
 add\_button.pack()  
  
 def perform\_add\_product(self, product\_name, price):  
 if product\_name == "" or price == "":  
 messagebox.showerror("Add Product Error", "Product name and price are required")  
 return  
  
 try:  
 price = float(price)  
 except ValueError:  
 messagebox.showerror("Add Product Error", "Invalid price. Please enter a valid number.")  
 return  
  
 if product\_name in self.products:  
 messagebox.showerror("Add Product Error", "Product already exists")  
 return  
  
 self.products[product\_name] = price  
  
 messagebox.showinfo("Add Product Successful", "Product added successfully")  
  
 def modify\_price\_window(self):  
 modify\_price\_window = Toplevel(self)  
 modify\_price\_window.title("Modify Price")  
  
 product\_name\_label = Label(modify\_price\_window, text="Product Name:")  
 product\_name\_label.pack()  
  
 product\_name\_var = StringVar(value="")  
 product\_name\_entry = Entry(modify\_price\_window, textvariable=product\_name\_var)  
 product\_name\_entry.pack()  
  
 new\_price\_label = Label(modify\_price\_window, text="New Price:")  
 new\_price\_label.pack()  
  
 new\_price\_var = StringVar(value="")  
 new\_price\_entry = Entry(modify\_price\_window, textvariable=new\_price\_var)  
 new\_price\_entry.pack()  
  
 modify\_button = Button(modify\_price\_window, text="Modify Price",  
 command=lambda: self.perform\_modify\_price(product\_name\_var.get(), new\_price\_var.get()))  
 modify\_button.pack()  
  
 def perform\_modify\_price(self, product\_name, new\_price):  
 if product\_name == "" or new\_price == "":  
 messagebox.showerror("Modify Price Error", "Product name and new price are required")  
 return  
  
 try:  
 new\_price = float(new\_price)  
 except ValueError:  
 messagebox.showerror("Modify Price Error", "Invalid price. Please enter a valid number.")  
 return  
  
 if product\_name not in self.products:  
 messagebox.showerror("Modify Price Error", "Product does not exist")  
 return  
  
 self.products[product\_name] = new\_price  
  
 messagebox.showinfo("Modify Price Successful", "Product price modified successfully")  
  
 def delete\_product\_window(self):  
 delete\_product\_window = Toplevel(self)  
 delete\_product\_window.title("Delete Product")  
  
 product\_name\_label = Label(delete\_product\_window, text="Product Name:")  
 product\_name\_label.pack()  
  
 product\_name\_entry = Entry(delete\_product\_window)  
 product\_name\_entry.pack()  
  
 delete\_button = Button(delete\_product\_window, text="Delete Product",  
 command=lambda: self.perform\_delete\_product(product\_name\_entry.get()))  
 delete\_button.pack()  
  
 def perform\_delete\_product(self, product\_name):  
 if product\_name == "":  
 messagebox.showerror("Delete Product Error", "Product name is required")  
 return  
  
 if product\_name not in self.products:  
 messagebox.showerror("Delete Product Error", "Product does not exist")  
 return  
  
 del self.products[product\_name]  
  
 messagebox.showinfo("Delete Product Successful", "Product deleted successfully")  
  
 def buy\_product(self):  
 buy\_window = Toplevel(self)  
 buy\_window.title("Buy Product")  
  
 product\_label = Label(buy\_window, text="Select a product:")  
 product\_label.pack()  
  
 selected\_product = StringVar(value="")  
 product\_combobox = ttk.Combobox(buy\_window, textvariable=selected\_product)  
 product\_combobox['values'] = list(self.products.keys())  
 product\_combobox.pack()  
  
 quantity\_label = Label(buy\_window, text="Enter quantity:")  
 quantity\_label.pack()  
  
 quantity\_var = IntVar(value=1) # Set default value and data type to int  
 quantity\_spinbox = Spinbox(buy\_window, textvariable=quantity\_var, from\_=0, to=100, increment=1)  
 quantity\_spinbox.pack()  
  
 add\_to\_cart\_button = Button(buy\_window, text="Add to Cart",  
 command=lambda: self.add\_to\_cart(selected\_product.get(), quantity\_var.get()))  
 add\_to\_cart\_button.pack()  
  
 def add\_to\_cart(self, product\_name, quantity):  
 if product\_name == "" or quantity <= 0:  
 messagebox.showerror("Add to Cart Error", "Product name and valid quantity are required")  
 return  
  
 if product\_name not in self.products:  
 messagebox.showerror("Add to Cart Error", "Product does not exist")  
 return  
  
 self.cart\_items.append({"product": product\_name, "quantity": quantity})  
  
 messagebox.showinfo("Add to Cart Successful", "Product added to cart successfully")  
 return  
  
 def purchase\_items(self):  
 total\_amount = 0.0  
  
 for item in self.cart\_items:  
 product\_name = item["product"]  
 quantity = item["quantity"]  
 price = self.products[product\_name]  
 total\_amount += price \* quantity  
  
 if total\_amount > self.wallet\_balance:  
 messagebox.showerror("Insufficient Balance", "You don't have enough balance for this purchase.")  
 else:  
 self.wallet\_balance -= total\_amount  
 self.cart\_items = []  
 self.update\_wallet\_label()  
 self.view\_cart()  
 messagebox.showinfo("Purchase Complete", "You purchased items worth ${:.2f}. "  
 "Your new wallet balance is ${:.2f}.".format(total\_amount,  
 self.wallet\_balance))  
  
 def view\_cart(self):  
 cart\_window = Toplevel(self)  
 cart\_window.title("View Cart")  
 cart\_label = Label(cart\_window, text="Cart Items:")  
 cart\_label.pack()  
  
 if not self.cart\_items:  
 empty\_cart\_label = Label(cart\_window, text="Cart is empty.")  
 empty\_cart\_label.pack()  
  
 total\_amount = 0.0  
 for item in self.cart\_items:  
 product\_name = item["product"]  
 quantity = item["quantity"]  
 price = self.products[product\_name]  
 item\_total = price \* quantity  
 total\_amount += item\_total  
 item\_label = Label(cart\_window, text=f"{product\_name} x {quantity} - ${item\_total:.2f}")  
 item\_label.pack()  
 purchase\_button = Button(cart\_window, text="Purchase", command=self.purchase\_items)  
 purchase\_button.pack()  
  
 total\_label = Label(cart\_window, text=f"Total Amount: ${total\_amount:.2f}")  
 total\_label.pack()  
  
 def show\_profile\_window(self):  
 profile\_window = Toplevel(self)  
 profile\_window.title("Profile")  
  
 current\_user\_label = Label(profile\_window, text=f"Current User: {self.current\_user}")  
 current\_user\_label.pack()  
  
 email\_label = Label(profile\_window, text=f"Password: {self.users[self.current\_user]}")  
 email\_label.pack()  
  
 def show\_wallet(self):  
 wallet\_window = Toplevel(self)  
 wallet\_window.title("Wallet")  
  
 wallet\_balance\_label = Label(wallet\_window, text=f"Wallet Balance: ${self.wallet\_balance:.2f}")  
 wallet\_balance\_label.pack()  
  
 def save\_settings(self):  
 font = self.font\_var.get()  
 text\_color = self.text\_color\_var.get()  
 theme = self.theme\_var.get()  
  
 self.master.option\_add('\*Font', font)  
 self.master.option\_add('\*Foreground', text\_color)  
  
 if theme == "light":  
 self.set\_light\_theme()  
 elif theme == "dark":  
 self.set\_dark\_theme()  
  
 messagebox.showinfo("Settings Saved", "Settings have been saved successfully")  
  
 def logout(self):  
 self.current\_user = None  
 self.cart\_items = []  
  
 self.clear\_frame()  
 self.create\_menu()  
 self.login\_window()  
  
 self.view\_menu.entryconfig("Show Products", state=DISABLED)  
 self.view\_menu.entryconfig("View Cart", state=DISABLED)  
 self.view\_menu.entryconfig("Profile", state=DISABLED)  
 self.view\_menu.entryconfig("Settings", state=DISABLED)  
 self.view\_menu.entryconfig("Buy", state=DISABLED)  
 self.view\_menu.entryconfig("My Wallet", state=DISABLED)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = Tk()  
 auto\_gas\_station = AutoGasStationFrame(root)  
 auto\_gas\_station.pack(fill="both", expand=True)  
 root.mainloop()