customer.py:

import tkinter as tk  
from tkinter import ttk, messagebox  
import pymysql  
from pymysql import OperationalError  
import datetime  
class Database:  
 def \_\_init\_\_(self):  
 self.config = {  
 "host": "localhost",  
 "user": "root",  
 "password": "zcy123456",   
 "database": "restaurant\_db",  
 "cursorclass": pymysql.cursors.DictCursor  
 }  
 self.connect()  
   
 def connect(self):  
 try:  
 self.conn = pymysql.connect(\*\*self.config)  
 except OperationalError as e:  
 messagebox.showerror("数据库连接失败", f"请检查配置: {str(e)}")  
 raise  
  
 def query(self, sql, params=None):  
 with self.conn.cursor() as cursor:  
 cursor.execute(sql, params or ())  
 return cursor.fetchall()  
  
 def execute(self, sql, params=None):  
 with self.conn.cursor() as cursor:  
 cursor.execute(sql, params or ())  
 self.conn.commit()  
  
class CustomerApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("膳食阁-食客点餐")  
 self.root.geometry("1000x600")  
 self.db = Database()  
 self.current\_table\_id = 1  
 self.selected\_items = []  
 self.cart\_buttons = {}  
 self.current\_order\_id = None  
  
 self.create\_widgets()  
 self.load\_menu\_items()  
  
 def create\_widgets(self):  
 top\_frame = ttk.Frame(self.root)  
 top\_frame.pack(fill=tk.X, padx=10, pady=10)  
  
 ttk.Label(top\_frame, text="选择桌号:").pack(side=tk.LEFT, padx=5)  
  
 self.table\_var = tk.StringVar(value="1")  
 table\_combo = ttk.Combobox(top\_frame, textvariable=self.table\_var, values=[str(i) for i in range(1, 21)],  
 width=5)  
 table\_combo.pack(side=tk.LEFT, padx=5)  
 table\_combo.bind("<<ComboboxSelected>>", self.update\_table\_id)  
  
 self.order\_info\_lbl = ttk.Label(top\_frame, text="订单: 未生成")  
 self.order\_info\_lbl.pack(side=tk.RIGHT, padx=20)  
  
 middle\_frame = ttk.Frame(self.root)  
 middle\_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=5)  
  
 ttk.Label(middle\_frame, text="菜单", font=("微软雅黑", 12, "bold")).pack(pady=5)  
 self.menu\_frame = ttk.Frame(middle\_frame)  
 self.menu\_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=5)  
  
 ttk.Label(middle\_frame, text="购物车", font=("微软雅黑", 12, "bold")).pack(pady=5)  
 self.cart\_frame = ttk.Frame(middle\_frame)  
 self.cart\_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=5)  
  
 bottom\_frame = ttk.Frame(self.root)  
 bottom\_frame.pack(fill=tk.X, padx=10, pady=10)  
  
 self.submit\_btn = ttk.Button(bottom\_frame, text="提交订单", command=self.submit\_order, state=tk.DISABLED)  
 self.submit\_btn.pack(side=tk.RIGHT, padx=10)  
  
 def update\_table\_id(self, event):  
 try:  
 self.current\_table\_id = int(self.table\_var.get())  
 except ValueError:  
 self.current\_table\_id = 1  
 self.table\_var.set("1")  
  
 def load\_menu\_items(self):  
 for widget in self.menu\_frame.winfo\_children():  
 widget.destroy()  
  
 columns = ("id", "name", "price", "profit", "desc", "add")  
 self.menu\_tree = ttk.Treeview(self.menu\_frame, columns=columns, show="headings")  
  
 self.menu\_tree.heading("id", text="ID")  
 self.menu\_tree.heading("name", text="名称")  
 self.menu\_tree.heading("price", text="价格")  
 self.menu\_tree.heading("profit", text="利润")  
 self.menu\_tree.heading("desc", text="描述")  
 self.menu\_tree.heading("add", text="操作")  
  
 self.menu\_tree.column("id", width=50, anchor=tk.CENTER)  
 self.menu\_tree.column("name", width=120, anchor=tk.CENTER)  
 self.menu\_tree.column("price", width=80, anchor=tk.CENTER)  
 self.menu\_tree.column("profit", width=80, anchor=tk.CENTER)  
 self.menu\_tree.column("desc", width=200, anchor=tk.W)  
 self.menu\_tree.column("add", width=80, anchor=tk.CENTER)  
  
 self.menu\_tree.pack(fill=tk.BOTH, expand=True)  
 self.menu\_tree.bind("<ButtonRelease-1>", self.on\_menu\_click)  
  
 menu\_data = self.db.query("SELECT \* FROM menu")  
 for item in menu\_data:  
 self.menu\_tree.insert("", tk.END, values=(  
 item["dish\_id"],  
 item["dish\_name"],  
 f"{item['price']}元",  
 f"{item['profit\_per\_dish']}元",  
 item["description"],  
 "添加"  
 ))  
  
 def on\_menu\_click(self, event):  
 region = self.menu\_tree.identify\_region(event.x, event.y)  
 if region != "cell":  
 return  
  
 item = self.menu\_tree.selection()[0]  
 column = int(self.menu\_tree.identify\_column(event.x).replace('#', ''))  
  
 if column == 6:   
 dish\_id = self.menu\_tree.item(item, "values")[0]  
 self.add\_to\_cart(dish\_id)  
  
 def add\_to\_cart(self, dish\_id):  
 menu\_item = next((item for item in self.db.query("SELECT \* FROM menu WHERE dish\_id = %s", (dish\_id,))), None)  
 if not menu\_item:  
 return  
  
 existing\_item = next((item for item in self.selected\_items if item["dish\_id"] == dish\_id), None)  
  
 if existing\_item:  
 existing\_item["quantity"] += 1  
 else:  
 self.selected\_items.append({  
 "dish\_id": dish\_id,  
 "dish\_name": menu\_item["dish\_name"],  
 "price": menu\_item["price"],  
 "quantity": 1  
 })  
  
 self.update\_cart()  
 self.submit\_btn.config(state=tk.NORMAL if self.selected\_items else tk.DISABLED)  
  
 def update\_cart(self):  
 for widget in self.cart\_frame.winfo\_children():  
 widget.destroy()  
  
 if not self.selected\_items:  
 ttk.Label(self.cart\_frame, text="购物车为空").pack(pady=20)  
 return  
  
 cart\_columns = ("name", "price", "quantity", "subtotal", "action")  
 cart\_tree = ttk.Treeview(self.cart\_frame, columns=cart\_columns, show="headings")  
  
 cart\_tree.heading("name", text="名称")  
 cart\_tree.heading("price", text="单价")  
 cart\_tree.heading("quantity", text="数量")  
 cart\_tree.heading("subtotal", text="小计")  
 cart\_tree.heading("action", text="操作")  
  
 cart\_tree.column("name", width=120, anchor=tk.CENTER)  
 cart\_tree.column("price", width=80, anchor=tk.CENTER)  
 cart\_tree.column("quantity", width=80, anchor=tk.CENTER)  
 cart\_tree.column("subtotal", width=80, anchor=tk.CENTER)  
 cart\_tree.column("action", width=120, anchor=tk.CENTER)  
  
 cart\_tree.pack(fill=tk.BOTH, expand=True)  
  
 for item in self.selected\_items:  
 subtotal = item["price"] \* item["quantity"]  
 cart\_tree.insert("", tk.END, values=(  
 item["dish\_name"],  
 f"{item['price']}元",  
 item["quantity"],  
 f"{subtotal}元",  
 f"修改"  
 ))  
  
 total = sum(item["price"] \* item["quantity"] for item in self.selected\_items)  
 ttk.Label(self.cart\_frame, text=f"总价: {total}元", font=("微软雅黑", 12, "bold")).pack(pady=10)  
  
 def decrease\_quantity(self, dish\_id):  
 item = next((item for item in self.selected\_items if item["dish\_id"] == dish\_id), None)  
 if item:  
 if item["quantity"] > 1:  
 item["quantity"] -= 1  
 else:  
 self.selected\_items.remove(item)  
 self.update\_cart()  
 self.submit\_btn.config(state=tk.NORMAL if self.selected\_items else tk.DISABLED)  
  
 def increase\_quantity(self, dish\_id):  
 item = next((item for item in self.selected\_items if item["dish\_id"] == dish\_id), None)  
 if item:  
 item["quantity"] += 1  
 self.update\_cart()  
  
 def generate\_order\_id(self):  
 try:  
 today = datetime.date.today()  
 date\_str = today.strftime("%Y%m%d")  
  
 sql = """  
 SELECT MAX(SUBSTRING(order\_id, 9, 4)) AS max\_seq   
 FROM orders   
 WHERE create\_date = %s  
 """  
 result = self.db.query(sql, (today,))  
 max\_seq = result[0]["max\_seq"] if result and result[0]["max\_seq"] else "0000"  
  
 new\_seq = int(max\_seq) + 1  
  
 seq\_str = f"{new\_seq:04d}"  
  
 return f"{date\_str}{seq\_str}"  
  
 except Exception as e:  
 messagebox.showerror("错误", f"生成订单号失败: {str(e)}")  
 return None  
  
 def submit\_order(self):  
 if not self.selected\_items:  
 messagebox.showinfo("提示", "购物车为空")  
 return  
  
 if not self.current\_table\_id or self.current\_table\_id < 1 or self.current\_table\_id > 20:  
 messagebox.showerror("错误", "请选择有效桌号（1-20）")  
 return  
  
 try:  
 order\_id = self.generate\_order\_id()  
 if not order\_id:  
 return  
  
 self.db.execute("START TRANSACTION")  
  
 for item in self.selected\_items:  
 subtotal = item["price"] \* item["quantity"]  
 self.db.execute(  
 """INSERT INTO orders (order\_id, table\_id, dish\_id, quantity, subtotal, order\_time, create\_date)  
 VALUES (%s, %s, %s, %s, %s, NOW(), CURDATE())""",  
 (order\_id, self.current\_table\_id, item["dish\_id"], item["quantity"], subtotal)  
 )  
  
 self.db.execute("COMMIT")  
  
 self.current\_order\_id = order\_id  
 self.order\_info\_lbl.config(text=f"订单: {order\_id}")  
  
 total\_amount = sum(item["price"] \* item["quantity"] for item in self.selected\_items)  
 messagebox.showinfo(  
 "订单提交成功",  
 f"桌号 {self.current\_table\_id}\n"  
 f"订单号: {order\_id}\n"  
 f"共 {len(self.selected\_items)} 种菜品\n"  
 f"总价: {total\_amount:.2f} 元\n\n"  
 "订单已提交至后厨！"  
 )  
  
 self.selected\_items = []  
 self.update\_cart()  
 self.submit\_btn.config(state=tk.DISABLED)  
  
 except Exception as e:  
 self.db.execute("ROLLBACK")  
 messagebox.showerror("错误", f"订单提交失败: {str(e)}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = CustomerApp(root)  
 root.mainloop()