MINI PROJECT REPORT

on

'International Cuisine Ordering System' using Python

Submitted by

A. AVINASH SASTRY (RA2311004010007) V. VENKATA HAVISH (RA2311004010041)

Semester – II

Academic Year: 2023-24 Even

Under the guidance of

Dr. B. Priyalakshmi Assistant Professor, Department of ECE

In partial fulfilment for the Course

of

21CSS101J - PROGRAMMING FOR PROBLEM SOLVING



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

College of Engineering and Technology, SRM Institute of Science and Technology

SRM Nagar, Kattankulathur – 603203, Kancheepuram District, Tamil Nadu.

April 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this activity report for the course 21CSS101J -PROGRAMMING FOR PROBLEM SOLVING is

the bonafide work of A. Avinash Sastry (RA2311004010007) who carried out the work under my supervision.

SIGNATURE

Dr. B. Priyalakshmi Assistant Professor Department of ECE SRMIST Kattankulathur

SIGNATURE

Dr. Shanthi Prince Head of The Department Department of ECE SRMIST Kattankulathur

TABLE OF CONTENTS

S.NO.	CONTENT	PAGE NO.
1	ABSTRACT	4
2	OBJECTIVE	4
3	INTRODUCTION	5
4	SYSTEM DESIGN AND SOURCE CODE	5
5	RESULTS (SCREENSHOTS)	11
6	REFERENCES	14

Abstract:

"International Cuisine Ordering System" is a user-friendly graphical application designed to streamline the process of ordering food from various international cuisines. With an intuitive interface, users can easily navigate through menus featuring a diverse range of dishes, including American, Italian, Indian, French, and Chinese specialties. Upon selecting their desired items and quantities, the system calculates the total order amount, incorporating tax where applicable. The application ensures a seamless ordering experience, enhancing customer satisfaction and efficiency in restaurant operations.

Objective:

The objective of the "International Cuisine Ordering System" is to provide a convenient and efficient platform for customers to browse, select, and order food items from diverse international cuisines. By offering a user-friendly interface and seamless navigation, the system aims to enhance the dining experience for customers while optimizing order management for restaurant staff. Key objectives include facilitating easy menu exploration, accurate order placement, and transparent calculation of total costs, thereby promoting customer satisfaction and operational efficiency within the restaurant environment.

Introduction:

In today's interconnected world, the culinary landscape has expanded to encompass a rich tapestry of flavors and traditions from around the globe. As diners increasingly seek diverse and authentic dining experiences, restaurants are faced with the challenge of catering to a wide range of culinary preferences. To meet this demand and streamline the ordering process, we introduce the "International Cuisine Ordering System." This innovative application offers a curated selection of dishes from prominent international cuisines, including American, Italian, Indian, French, and Chinese fare, all within a single platform. By providing customers with an intuitive interface to explore menus, select their favorite dishes, and place orders seamlessly, our system aims to elevate the dining experience while optimizing operational efficiency for restaurant staff. Join us as we embark on a culinary journey that celebrates diversity, convenience, and culinary excellence.

System Design:

1. Menu Definition:

- Defines menus for various cuisines, each containing a list of items with their respective prices in INR.

2. Tkinter GUI:

- Creates a graphical user interface using Tkinter library.
- Displays buttons for each cuisine to allow users to select a menu.
- Displays menu items with prices when a cuisine button is clicked.
- Provides entry fields for users to input the quantity of each item they want to order.

3. Order Calculation:

- Calculates the total order amount based on the quantities entered by the user and the prices of selected items.

- Applies a fixed tax rate (7.5%) to calculate the tax amount.

4. Checkout Functionality:

- Provides a "Proceed to Checkout" button to calculate the total amount and display it

in a message box.

- Clears the screen after checkout to return to the main menu.

5. Basic Error Handling:

- Displays error messages in case of invalid input (e.g., negative quantities) when calculating the total amount.

Source Code:

import tkinter as tk from tkinter import messagebox

```
# Define menus with prices in INR
menus = {
  "American": {
     "Pizza": 918.33,
     "Burger": 708.25,
     "Fries": 208.33.
     "Soda": 145.83
  },
  "Italian": {
     "Pasta": 832.59,
     "Lasagna": 1041.67,
     "Salad": 562.92,
     "Wine": 416.60
  },
  "Indian": {
     "Curry": 687.19,
     "Naan": 166.64,
     "Samosa": 125.00,
     "Lassi": 187.50
  },
  "French": {
     "Croissant": 112.50,
     "Quiche": 256.94,
```

```
"Escargot": 750.00,
     "Creme Brulee": 375.00
  },
  "Chinese": {
     "Dumplings": 208.33,
     "Spring Rolls": 187.50,
     "Kung Pao Chicken": 520.83,
     "Fried Rice": 312.50
  }
}
class RestaurantApp:
  def init_(self, root):
     self.root = root
    self.root.title("Restaurant Ordering System")
     self.root.geometry("400x400")
    # Create buttons for each menu
     self.menu buttons = []
    for menu name in menus:
       button = tk.Button(root, text=menu_name, command=lambda m=menu_name:
self.display_menu(m))
       button.pack()
       self.menu_buttons.append(button)
    self.order = { }
    self.total inr = 0
  def display_menu(self, menu_name):
    # Clear the screen and display menu items for the selected menu
    self.clear screen()
    self.current_menu = menu_name
    tk.Label(self.root, text=menu name + " Menu:").pack()
    for item, price in menus[menu_name].items():
       tk.Label(self.root, text=f"{item}: ₹{price:.2f}").pack()
       quantity_entry = tk.Entry(self.root)
```

```
quantity_entry.pack()
       self.order[item] = {"quantity": quantity_entry, "price": price}
     # Add a button to proceed to checkout
     tk.Button(self.root, text="Proceed to Checkout", command=self.checkout).pack()
  def checkout(self):
     # Calculate total price including tax and display it in a messagebox
     self.total inr = 0
     for item, data in self.order.items():
       try:
          quantity = int(data["quantity"].get())
          if quantity < 0:
            raise ValueError
          self.total_inr += quantity * data["price"]
       except ValueError:
          messagebox.showerror("Error", "Please enter a valid quantity for all
items.")
          return
     tax rate = 0.075 #assumed to be 7.5%
     tax inr = self.total inr * tax rate
     total_with_tax = self.total_inr + tax_inr
     messagebox.showinfo("Total", f"Total: \mathbb{Z} total with tax:.2f}\nTax:
₹{tax inr:.2f}\n\nThank You!")
     # Clear the order and go back to the main menu
     self.clear screen()
     for button in self.menu buttons:
       button.pack()
  def clear screen(self):
     # Clear all widgets from the screen
     for widget in self.root.winfo_children():
       widget.pack forget()
```

```
8
```

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

How it all works:

Importing Necessary Libraries:

We import the tkinter library to create the GUI application. We import messagebox from tkinter to display message boxes for showing the total amount and any errors.

Define Menus:

We define different menus with their respective items and prices in Indian Rupees (INR). Each menu is represented as a dictionary where the keys are the menu items and the values are their prices.

RestaurantApp Class:

This class represents the main application.

In the constructor __init__(), we initialize the Tkinter root window, set its title and size, and create an empty list menu_buttons to store menu buttons.

We create buttons for each menu defined in the menus dictionary. Each button is associated with a command to display the menu for the corresponding cuisine when clicked.

We initialize an empty dictionary order to store the user's order, and total_inr to keep track of the total amount of the order.

display_menu Method:

This method is called when a menu button is clicked. It clears the screen to remove any existing widgets. It sets self.current_menu to the selected menu name.

It displays the selected menu's items along with entry fields for the user to enter the quantity of each item they want to order.

It creates a button labeled "Proceed to Checkout" which, when clicked, calls the checkout function.

checkout Function:

This method calculates the total order amount including tax and displays it in a messagebox.

It iterates over the items in the order dictionary, retrieves the quantity entered by the user for each item, and calculates the total amount.

It calculates the tax (assumed to be 7.5% of the total order amount).

It displays the total amount along with the tax in a messagebox using messagebox.showinfo.

It clears the screen to go back to the main menu after the order is completed.

clear_screen Function:

This Function removes all widgets from the root window. It is used to clear the screen before displaying a new menu or going back to the main menu.

Main Execution:

We create a Tkinter root window.

We instantiate the RestaurantApp class with the root window.

We start the Tkinter event loop using root.mainloop(), which keeps the application running until the user closes the window

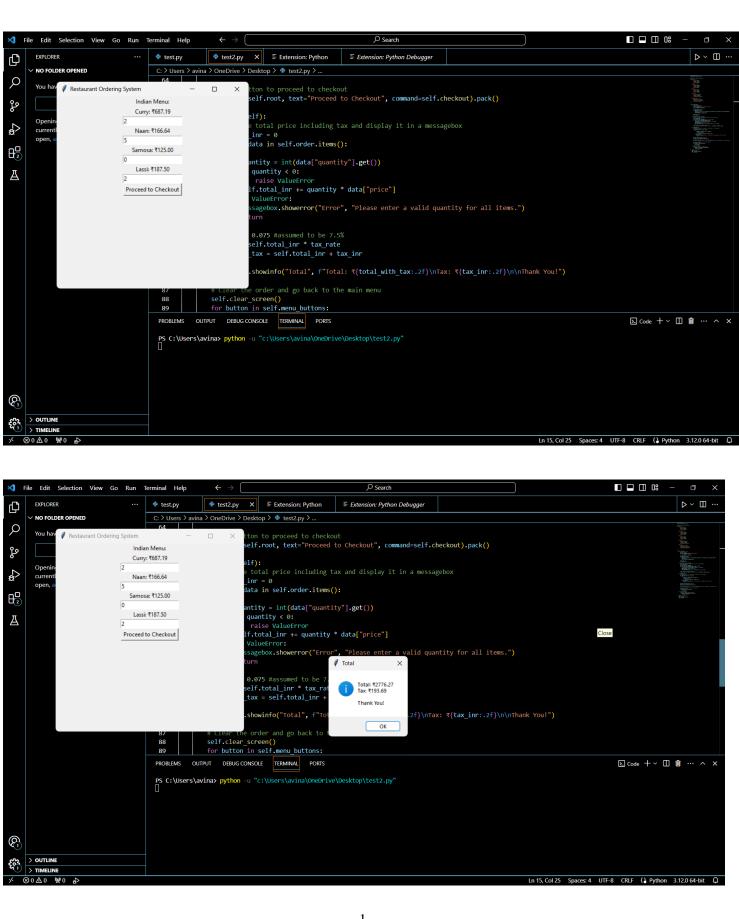
Results (Screenshots):

ile Edit Selection View Go Run	Terminal Help	$\leftrightarrow \rightarrow$					□□□0: -	- 0 :
EXPLORER	🕏 test.py	🔹 test2.py 🗙						⊳ × ⊡ ·
✓ NO FOLDER OPENED		vina > OneDrive > Deskto						
You have not yet opened a folder.	1 impor	rt tkinter as tk						1000 1000
	2 from 3	tkinter import mes	sagebox					1000 1000 1000 1000
Open Folder		fine menus with pr	ices in INR					Maria maria
Opening a folder will close all	5 menus 6 '	s = { "American": {						
currently open editors. To keep them open, add a folder instead.	7 8	"Pizza": 918.3						ECOENC.
	9	"Burger": 708.2 "Fries": 208.3						Carlo cara
	10 11	"Soda": 145.83						
	12	}, "Italian": {{						
	13 14	"Pasta": 832.59 "Lasagna": 1042						
	15	"Salad": 562.92						
	16 17	"Wine": 416.60						
	18	"Indian": {						
	19 20	"Curry": 687.19 "Naan": 166.64						
	21	"Samosa": 125.0	90,					
	22	"Lassi": 187.50 },						
	24	"French": {	10.50					
	25 26	"Croissant": 1 "Quiche": 256.9						
	27 28	"Escargot": 750 "Creme Brulee"	.00,					
	29]	},	575.00					
	30 ' 31	"Chinese": { "Dumplings": 20						
	32	"Spring Rolls"						
	33 34	"Kung Pao Chick "Fried Rice":						
	35	}	12.50					
	55							
	36 }							
> OUTLINE	36 } 37 38 class	s RestaurantApp:						
> OUTLINE > TIMELINE	36 } 37 38 class	s RestaurantApp: definit(self,	root):					
>TIMELINE ③o∆o %ao 4a>	36 } 37 38 class 39 c	definit(self,	root):	Ø Search		Ln 15, Col 25 Spaces: 4 UTF-8		
> TIMELINE	36 } 37 38 class 39 c	definit(self,	root): **	,∕⊃ Search ≣ Extension: Python Debugger			CRLF () Python	
≻TIMELINE ⊗o∆o %a⁰o g> ile Edit Selection View Go Run े	36 } 37 38 class 39 c Terminal Help	definit(self,	Extension: Python)			a
>TIMELINE 90 & 0 \$¥0 \$> ile Edit Selection View Go Run EXPLORER ···	36 } 37 38 class 39 c Terminal Help	definit(self, salf seat reactions ← →	Extension: Python p > ♦ test2.py >)			a
> TIMELINE 0 0 ∆ 0 92 0 6> le Edit Selection View Go Run EXPLORER	36 } 37 38 38 class 39 c rerminal Help ● test.py c: > Users > avi 38 class 39 d 39 d	definit(self,	E Extension: Python p > ● test2.py > root): t	Extension: Python Debugger)			đ
> TIMELINE) 0 △ 0 99 0 6> le Edit Selection View Go Run EXPLORER > NO FOLGER OPENED You have not yet opened a folder.	36 } 37 38 class 39 c Terminal Help • test.py C: > Users > avi 38 class 39 d	definit(self,	<pre>Extension: Python p > ● test2.py > root): t 'Restaurant Orderin</pre>	Extension: Python Debugger				đ
> TIMELINE D	36) 37 38 class 39 c Terminal Help	definit(self,	<pre>F Extension: Python p > ● test2.py > root): t ("Restaurant Orderin try("480x408")</pre>	Extension: Python Debugger				đ
> TIMELINE 0 ▲ 0 ₩ 0 ☆> e Edit Selection View Go Run EXPLORER ···· > NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all	36) 37 38 class 39 c Terminal Help	definit(self,	<pre>></pre>	Extension: Python Debugger				a
→ TIMELINE 0 ▲ 0 ♀ 0 ↔ e Edit Selection View Go Run EXPLORER > NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 class 39 c terminal Help	definit(self, 	<pre>Extension: Python p > ● test2.py > root): t ("Restaurant Orderin try("400×400") s = for each menu ns = [] n menus: Button(root, text=m</pre>	Extension: Python Debugger				a
→ TIMELINE 0 ▲ 0 ♀ 0 ↔ e Edit Selection View Go Run EXPLORER > NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 class 39 c Terminal Help ♥ test.py C:>Users > avi 38 class 39 d 40 d 41 4 42 43 44 45 46 d	definit(self,	<pre>Extension: Python p > ● test2.py > root): t ("Restaurant Orderin try("400×400") s = for each menu ns = [] n menus: Button(root, text=m</pre>	Extension: Python Debugger g System") enu_name, command=lambda m=n				a
→ TIMELINE 0 ▲ 0 ♀ 0 ↔ e Edit Selection View Go Run EXPLORER > NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 class 39 c Terminal Help	definit(self,	<pre>> Extension: Python p > ● test2.py > root): t t ("Restaurant Orderin try("400×400") s for each menu ns = □ n menus: .Button(root, text=m () uttons.append(button</pre>	Extension: Python Debugger g System") enu_name, command=lambda m=n				a
→ TIMELINE 0 ▲ 0 ♀ 0 ↔ e Edit Selection View Go Run EXPLORER > NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 class 39 c Terminal Help	definit(self,	<pre>Extension: Python p > ● test2.py > root): t t ("Restaurant Orderin try("400x400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button</pre>	Extension: Python Debugger g System") enu_name, command=lambda m=n				a
> TIMELINE 0 ▲ 0 ♀ 0 ↔ 2 Edit Selection View Go Run EXPLORER ✓ NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will Cose all currently open editors. To keep them	36 } 37 38 class 39 c terminal Help	<pre>definit(self, ifiself, _</pre>	<pre>Extension: Python p > ● test2.py > root): t t ("Restaurant Orderin try("400×400") s = [] n menus: Button(root, text=m () uttons.append(button = 0 lf, menu_name):</pre>	<pre> Extension: Python Debugger g System") enu_name, command=lambda m=m) </pre>	enu_name: self.display_menu			٥
> TIMELINE 0 ▲ 0 ♀ 0 ↔ 2 Edit Selection View Go Run EXPLORER ✓ NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will Cose all currently open editors. To keep them	36 } 37 38 class 39 c Terminal Help ● testpy C:>Users > avi 38 class 38 class 38 class 40 d 41 4 41 42 43 44 45 56 50 50 51 52 53	<pre>definit(self, ifiself, _</pre>	<pre>> Extension: Python p > ● test2.py > root): t ("Restaurant Orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 1f, menu_name): een and display menu</pre>	Extension: Python Debugger g System") enu_name, command=lambda m=n	enu_name: self.display_menu			٥
> TIMELINE 0 ▲ 0 ♀ 0 ↔ e Edit Selection View Go Run DXPLORER > No FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36) 37 38 class 39 c 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	<pre>definit(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self,init) self.root.title self.root.geome # Create button self.root.geome # Create button self.order = () _self.total_inr def display_menu(se # Clear_the scr self.clear_scr</pre>	<pre>\$ Extension: Python p > • test2.py > root): t t ("Restaurant orderin try("400x400") s = 0 in menus: .Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() u = monu_name</pre>	<pre>Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu</pre>	enu_name: self.display_menu			٥
> TIMELINE 0 ▲ 0 ♀ 0 ↔ e Edit Selection View Go Run DXPLORER > No FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 class 39 c terminal Help	<pre>definit(self, ifiself,</pre>	<pre>Extension: Python p > • test2.py > root): t ("Restaurant Orderin try("400×400") s for each menu ns = [] n menus: Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = menu_name oot, text=menu_name in menus[menu_name]</pre>	<pre>Extension: Python Debugger g System") nenu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items():</pre>	enu_name: self.display_menu			٥
> TIMELINE 0 ▲ 0 ♥ 0 ↔ e Edit Selection View Go Run EXPLORER > No FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 class 39 c 10 terminal Help	<pre>definit(self, if end if end if end init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init) self.menu_button_pack initinit initinit initinit self.corder = () self.total_init display_menu(se # Clear the scr self.current_me tk.tabel(self, for item, price initend initinit)</pre>	<pre>Extension: Python p > ● test2.py > root): t t ("Restaurant Orderin try("400×400") s for each menu ns = [] n menus: Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = menu_name jf.root, text=f'(ite try = tk.tnry(self.</pre>	<pre>Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items(): m); <(price:.2f)").pack()</pre>	enu_name: self.display_menu			a
> TIMELINE D	36) 37 38 class 39 c 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	<pre>definit(self, init(self, init(self, init_) > OneDrive > Deskto b RestaurantApp: lefinit(self, self.root = roo self.root.title self.root.title self.root.title self.root.title button.pack self.menu_butto for menu_name i button.pack self.order = () self.total_inr def display_menu(se # Clear the scr self.clear_scr self.cle</pre>	<pre>\$ Extension: Python p > ● test2.py > root): t t ("Restaurant orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = moru_name oot, text=menu_name lf.root, text=f"(ite try = tk.Entry(celf. try.pack()</pre>	<pre>Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items(): m): <{price:.2f}").pack() root)</pre>	wenu_name: self.display_menu			đ
> TIMELINE D	36 } 37 38 class 39 c terminal Help test,py c:> Users > avi 38 class 39 d 40 41 41 42 43 40 44 44 44 45 50 51 52 53 54 d 55 56 56 57 58 58 60 61 62 63 64	<pre>definit(self, init(self, init(self, init_) oneDrive > Deskto s RestaurantApp: lefinit(self, self.root.title self.root.title self.root.title self.root.geome # Create button for menu_name i button.pack self.order = () self.otdal_inr def display_menu(se # Clear the scr self.clear_scre self.cl</pre>	<pre>Extension: Python p > • test2.py > root): t ("Restaurant Orderin try("400x400") s for each menu ns = [] n menus: Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = menu_name oot, text=menu_name [lf.root, text=fr(ite try = tk.Entry(self. try.pack() item] = {"quantity":</pre>	<pre>Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items(): m): <(price:.2f)").pack() root) quantity_entry, "price": pr</pre>	wenu_name: self.display_menu			đ
> TIMELINE D	36 } 37 38 38 class 39 c test.py C:> Users > avi 38 class 39 d 40 d 41 d 42 d 43 d 44 d 45 d 60 50 51 52 53 53 54 d 55 55 56 57 58 59 60 61 62 63 64 65 66 66	<pre>definit(self,</pre>	<pre>\$ Extension: Python p > ● test2.py > root): t ("Restaurant Orderin try("400x400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 1f, menu_name): een and display menu en() nu = menu_name oot, text=menu_name in menus[menu_name] if.root, text=r"(ite try = tk.Entry(self. try.pack() item] = "quantity": to proceed to checko</pre>	<pre>Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items(): m): <(price:.2f)").pack() root) quantity_entry, "price": pr</pre>	enu_name: self.display_menu ice}			đ
> TIMELINE 0 0 ∆ 0 1⁄2 0 5> le Edit Selection View Go Run EXPLORER	36 } 37 38 class 39 C 38 class 40 d 41 42 46 50 51 52 53 54 55 56 57 58 <td><pre>definit(self,</pre></td> <td><pre>Extension: Python p > ● test2.py > root): t t ("Restaurant orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = menu_name oot, text=menu_name lf.root, text=f"{ite try = tk.Entry(self. try.pack() item] = {"quantity": to proceed to checko root, text="Proceed</pre></td> <td><pre>Extension: Python Debugger g System") enu_name, command-lambda m=m) items for the selected menu + " Menu:").pack() .items(): m): = ({price:.2f}").pack() root) quantity_entry, "price": pr ut</pre></td> <td>enu_name: self.display_menu ice}</td> <td></td> <td></td> <td>đ</td>	<pre>definit(self,</pre>	<pre>Extension: Python p > ● test2.py > root): t t ("Restaurant orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = menu_name oot, text=menu_name lf.root, text=f"{ite try = tk.Entry(self. try.pack() item] = {"quantity": to proceed to checko root, text="Proceed</pre>	<pre>Extension: Python Debugger g System") enu_name, command-lambda m=m) items for the selected menu + " Menu:").pack() .items(): m): = ({price:.2f}").pack() root) quantity_entry, "price": pr ut</pre>	enu_name: self.display_menu ice}			đ
> TIMELINE 0 0 ∆ 0 12 0 5 Ie Edit Selection View Go Run EXPLORER	36) 37 38 class 39 class 39 class 39 d test,py C:>Users>avi 38 class 39 d 40 41 42 43 44 44 45 46 46 47 48 49 50 51 52 53 4 d 55 55 56 56 57 55 58 60 60 61 62 63 64 66 67 66 d 69 d	<pre>definit(self, init(self, init(self, init) @ test2.py X ima > OneDrive > Desktor s RestaurantApp: definit(self, self.root = roo self.root.title self.root.title self.root.title self.root.title button.pack self.menu_button self.order = () self.total_inr def display_menu(se # Clear the scr self.clar_scre self.order[# Add a button tk.Button(self. def checkout(self): # calculate tot</pre>	<pre>\$ Extension: Python p > • test2.py > root): t t ("Restaurant orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 1f, menu_name): een and display menu en() nu = menu_name oot, text=menu_name [1f.root, text=f"(ite try = tk.Entry(self. try.pack() item] = {"quantity": to proceed to checko root, text="Proceed al price including t</pre>	<pre>Extension: Python Debugger g System") enu_name, command-lambda m=m) items for the selected menu + " Menu:").pack() .items(): m): = ({price:.2f}").pack() root) quantity_entry, "price": pr ut</pre>	<pre>kenu_name: self.display_menu ice}</pre>			đ
> TIMELINE 0 0 ∆ 0 10 0 5 ile Edit Selection View Go Run EXPLORER	36 } 37 38 38 class 39 d 40 test.py C:>Users>avi 38 class 39 d 40 d 41 d 42 d 43 d 44 d 57 58 50 51 52 53 53 d 55 55 56 57 58 60 61 62 63 64 64 65 66 66 67 68 67 70 71 71	<pre>definit(self, init) init(self, init) self.total_init self.total_init self.current me tk.label(self, init) init_y.en self.order = init_y.en self.orderinit_y.en self.or</pre>	<pre>\$ Extension: Python p > • test2.py > root): t t ("Restaurant orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 1f, menu_name): een and display menu en() nu = menu_name oot, text=menu_name [1f.root, text=f"(ite try = tk.Entry(self. try.pack() item] = {"quantity": to proceed to checko root, text="Proceed al price including t</pre>	<pre>Extension: Python Debugger g System") eenu_name, command-lambda m=m) items for the selected menu + " Menu:").pack() .items(): my: = ({price:.2f}").pack() root) quantity_entry, "price": pr ut to Checkout", command=self.c ax and display it in a messa</pre>	<pre>kenu_name: self.display_menu ice}</pre>			a
> TIMELINE 3 0 ▲ 0 ♥ 0 ♪ ile Edit Selection View Go Run EXPLORER ···· > No FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 38 class 39 C 20 Users > avi 38 class 39 d 40 d 42 d 43 d 44 d 45 d6 67 53 54 d 55 56 57 58 59 60 61 62 63 64 66 67 67 68	<pre>definit(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init(self, init) self.root.grows initinitiniti self.total_initi initi isplay_menutsi isplay_menutsi itcurrenti ititcurrenti itcurrenti ititcurrenti itcurrenti itcurrentititititititit</pre>	<pre>\$ Extension: Python p > ● test2.py > root): t ("Restaurant Orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 lf, menu_name): een and display menu en() nu = menu[name) out, text=menu_name in menus[menu_name] if.root, text=menu_name if.root, text=f"(ite try = tk.Entry(self. try.pack() item] = "quantity": to proceed to checko root, text="Proceed al price including t = 0</pre>	<pre>E Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items(): my: "(price:.2f}").pack() root) quantity_entry, "price": pr ut to Checkout", command=self.c ax and display it in a messa);</pre>	<pre>kenu_name: self.display_menu ice}</pre>			đ
> TIMELINE 0 0 ∆ 0 10 0 5 ile Edit Selection View Go Run EXPLORER > No FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them	36 } 37 38 38 class 39 C 38 class 39 d 40 d 41 42 43 44 45 46 47 8 50 51 52 55 58 96 61 62 62 63 64 65 66 67 68 d 67 68 67 70 71 72	<pre>definit(self, init(self, init(self, init(self, init) OneDrive > Deskto init(self, isit(self, self.root = roo self.root.geome # Create button self.nemu_button for menu_button = tk button = tk button = tk button = tk button = tk button = tk self.total_inr eself.clear_scre scl_clear_scre scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ scl_clear_scre_ s</pre>	<pre>Extension: Python p > ● test2.py > root): t ("Restaurant orderin try("400×400") s for each menu ns = [] n menus: .Button(root, text=m () uttons.append(button = 0 1f, menu_name): een and display menu en() nu = moru_name oot, text=menu_name [i.f.root, text=f"(itee try = tk.Entry(self. try.pack() item] = {"quantity": to proceed to checko root, text="Proceed al price including t = 0 in self.order.items()</pre>	<pre>E Extension: Python Debugger g System") enu_name, command=lambda m=m) items for the selected menu + " Menu:").pack() .items(): my: "(price:.2f}").pack() root) quantity_entry, "price": pr ut to Checkout", command=self.c ax and display it in a messa);</pre>	<pre>kenu_name: self.display_menu ice}</pre>			a

×1 -	ile Edit Selection View Go Run	erminal Help ←	\rightarrow			
¢	EXPLORER ····	🔹 test.py	t2.py × ≣ Extension: Python			▷ ~ □ …
	✓ NO FOLDER OPENED	C: > Users > avina > OneDr	rive > Desktop > 🌵 test2.py >			
Q	You have not yet opened a folder.	64 65 # Add	a button to proceed to check			
r So	Open Folder	66 tk.But 67 68 def checko		<pre>to Checkout", command=self.checkout).</pre>	pack()	
å	Opening a folder will close all currently open editors. To keep them open, add a folder instead.	69 # Calc 70 self.t	culate total price including t total_inr = 0	ax and display it in a messagebox		
₽₽			<pre>tem, data in self.order.items(</pre>			r gaginariani " Taging parasi " € dagina
Д		74 75 76 77 80 81 tax_ri 82 tax_ini 83 total_ 85 messag 86 # Clei 88 self.cl 89 for bu 90 bu 91	<pre>if quantity < 0:</pre>	* data["price"] "", "Please enter a valid quantity for % eax_inr al: T{total_with_tax:.2f}\nTax: T{tax_		
€;} €;}	> OUTLINE > TIMELINE	93 # clea 94 for wi 95 wi 96 97 ifname == 98 root = tk.	.Tk() taurantApp(root)			
	≥0∆0 %20 ±>				Ln 15, Col 25 Spaces: 4 L	JTF-8 CRLF {} Python 3.12.0 64-bit

Execution of code:

hav	Restaurant Ordering System	_		tton to proceed to checkout		
	American			<pre>self.root, text="Proceed to Checkout", command=self.checkout).pack()</pre>		1000- 1000-
	Italian			elf):		
nin entl n, a	Indian			e total price including tax and display it in a messagebox		Contraction of the second seco
enti n. a	French			_inr = 0		- The second
				data in self.order.items():		national and a second s
	Chinese			antity = int(data["quantity"].get())		
				quantity < 0:		
				raise ValueError		
				lf.total_inr += quantity * data["price"]		
				ValueError:		
				<pre>ssagebox.showerror("Error", "Please enter a valid quantity for all items.")</pre>		
				turn		
				0.075 #assumed to be 7.5%		
				self.total_inr * tax_rate		
				_tax = self.total_inr + tax_inr		
				.showinfo("Total", f"Total: ₹{total_with_tax:.2f}\nTax: ₹{tax_inr:.2f}\n\nThank You!")		
	8/		# Clear the	e order and go back to the main menu		
	88		self.clear			
	89		for button	in self.menu_buttons:		
	PROBL	LEMS OUTP	UT DEBUG CO	ONSOLE TERMINAL PORTS	∑ Code + × □	🛍 … ^ ×
	PS C	:\Users\avi	na> python	-u "c:\Users\avina\OneDrive\Desktop\test2.py"		
LINE						
IUN						



References:

- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://www.udemy.com/course/complete-python-bootcamp/
- <u>https://www.freecodecamp.org/news/learning-python-from-zero-to-hero-120ea540b567/</u>