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

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: ₹{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()
```

```
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.

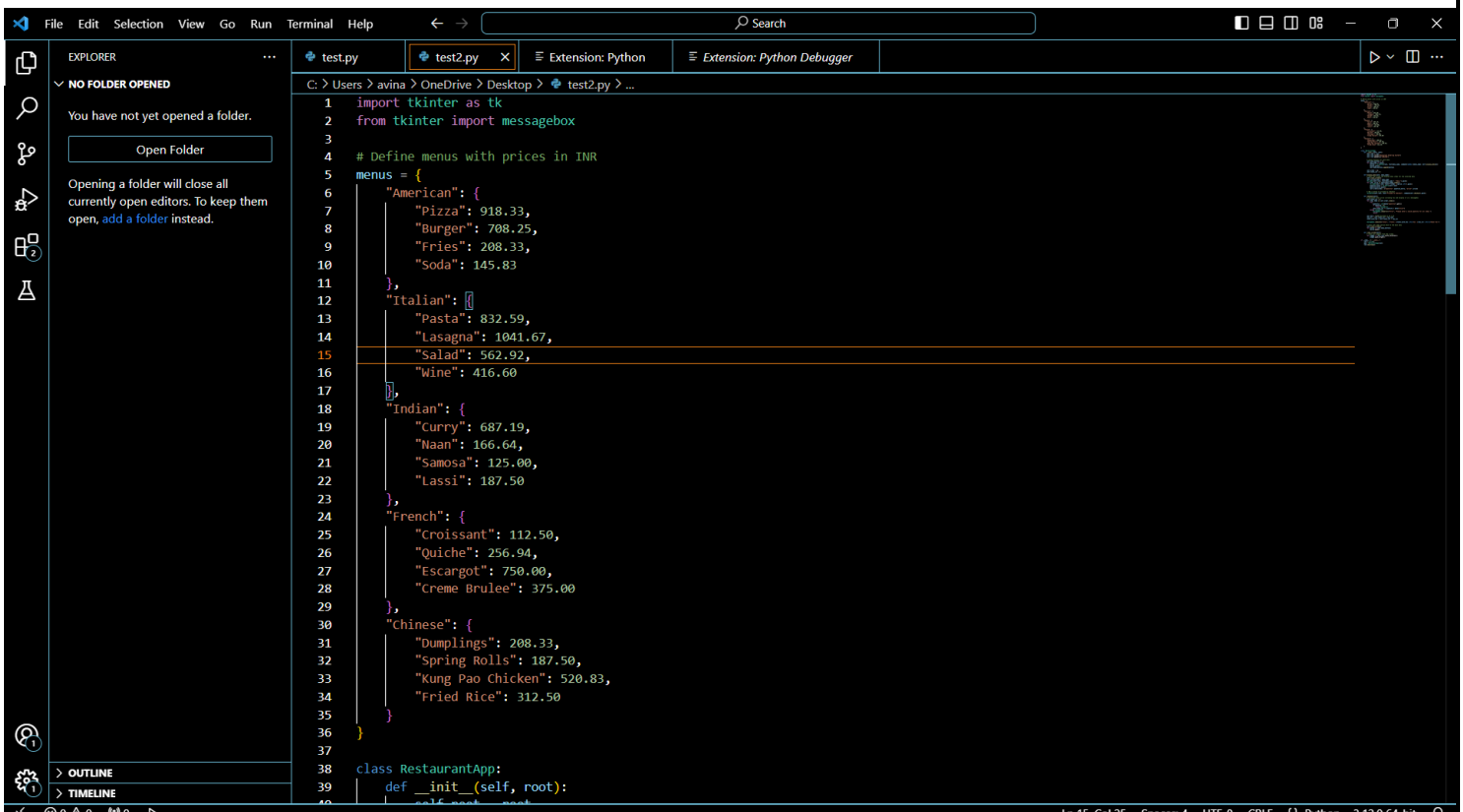
clear_screen Function:

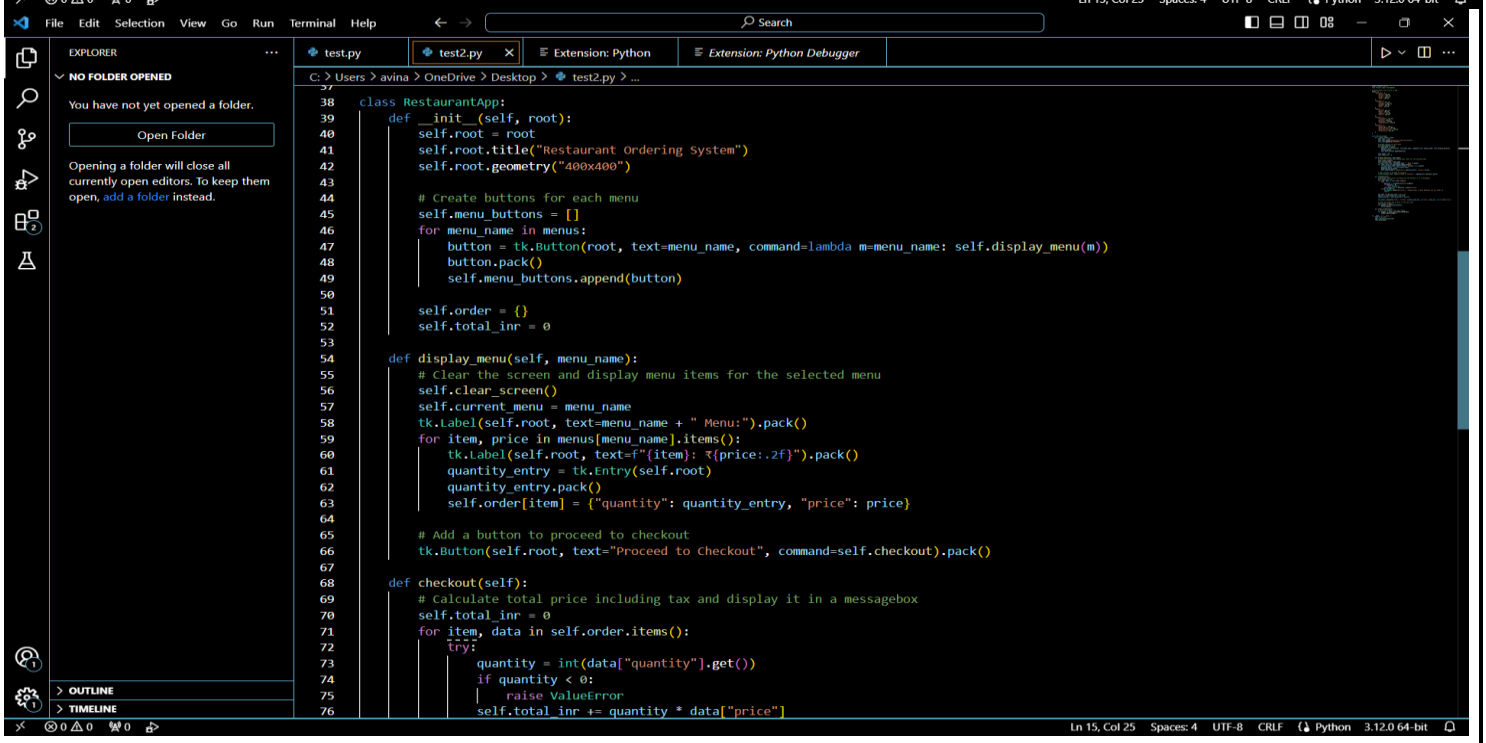
This Function removes all widgets from the root window. It is used to clear the screen before displaying a new menu or going back to the main menu.

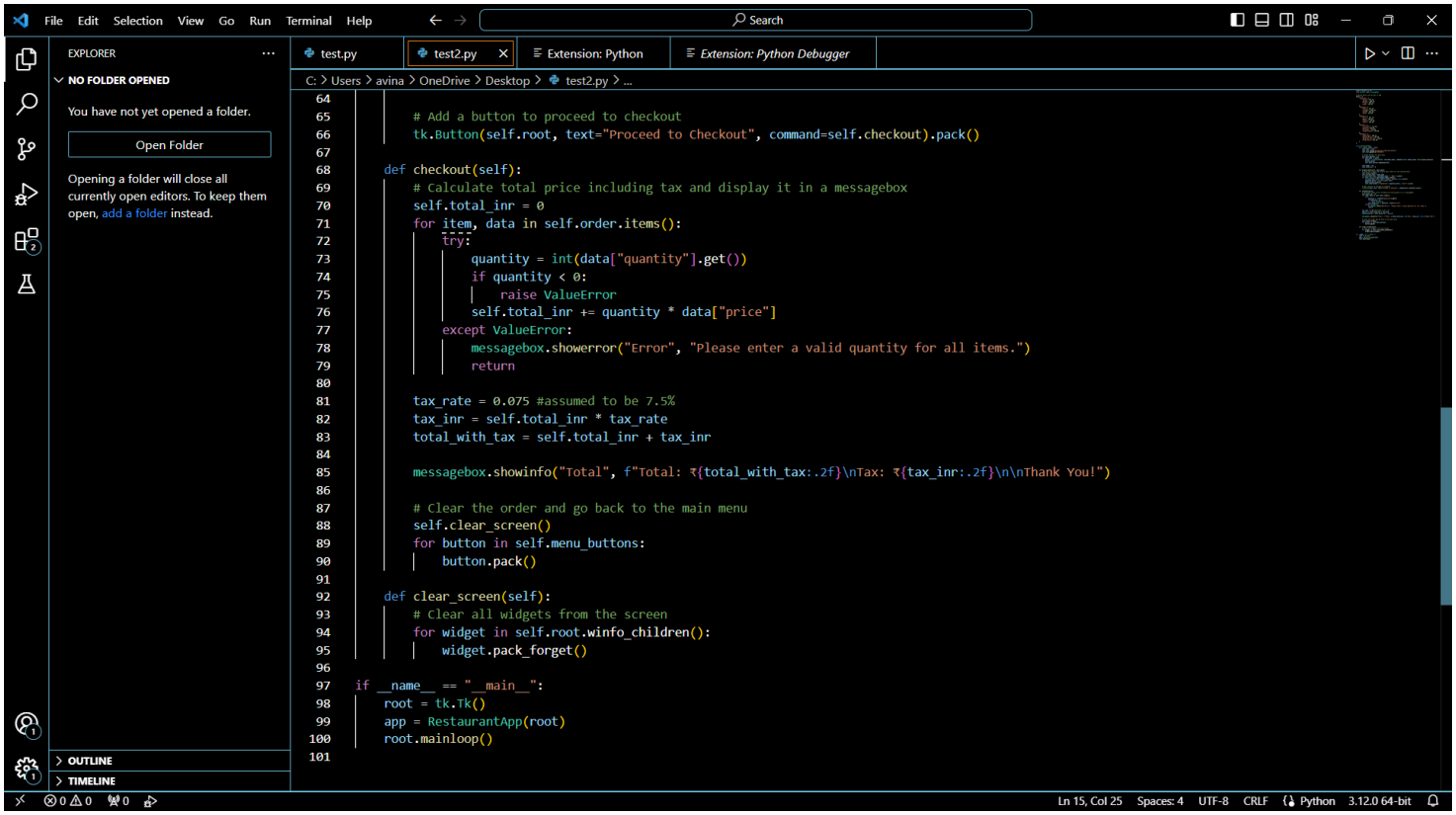
Main Execution:

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):

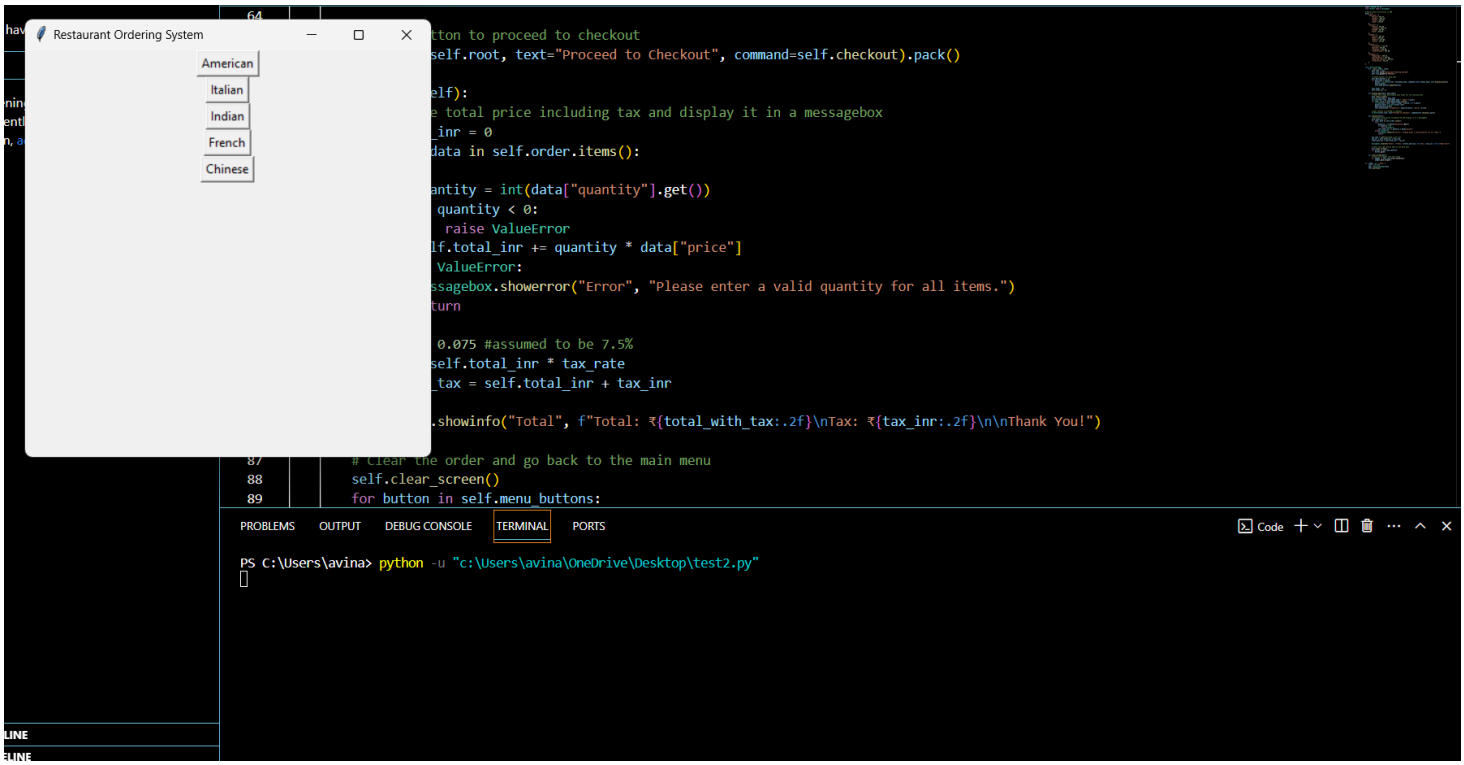


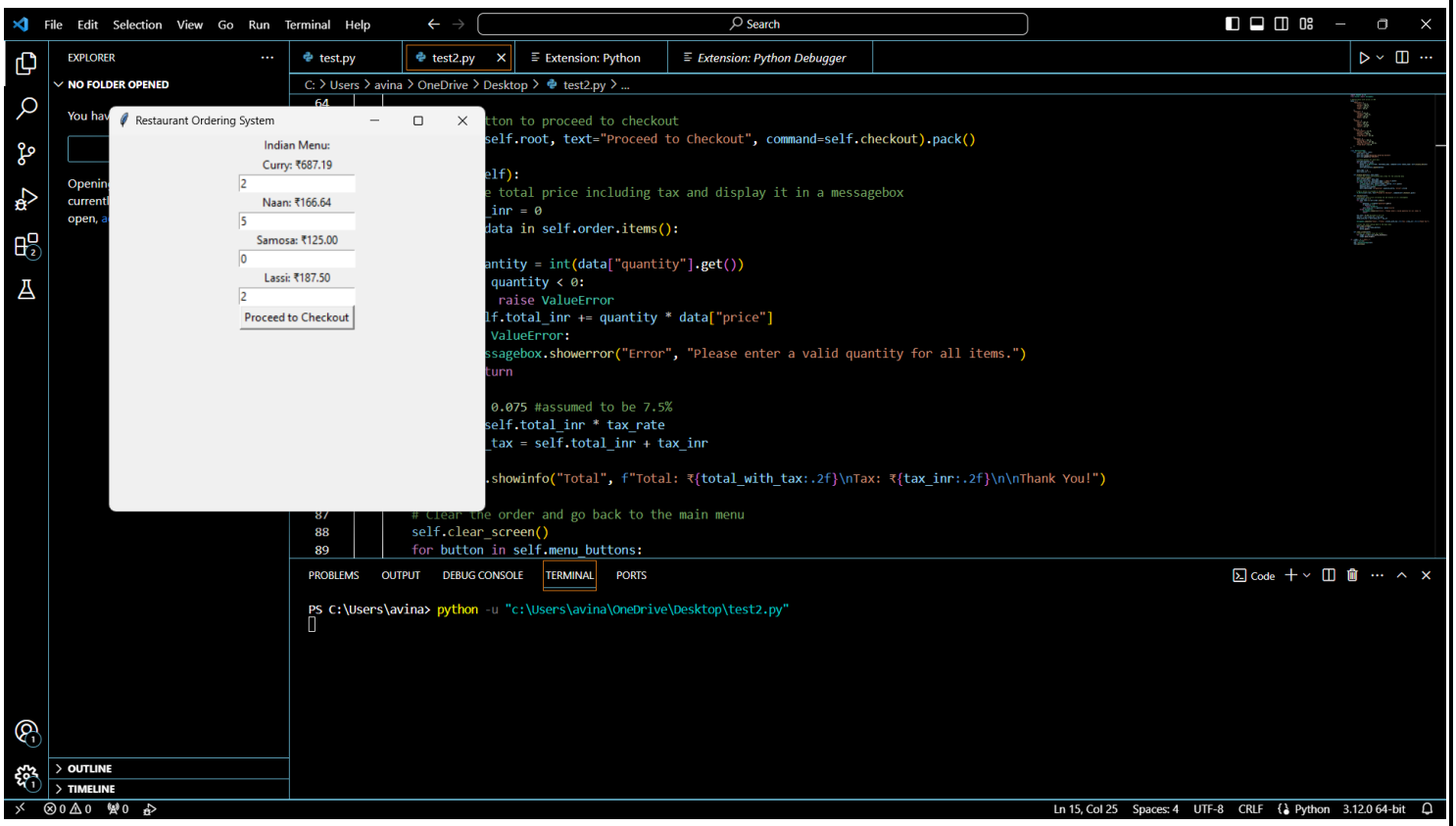


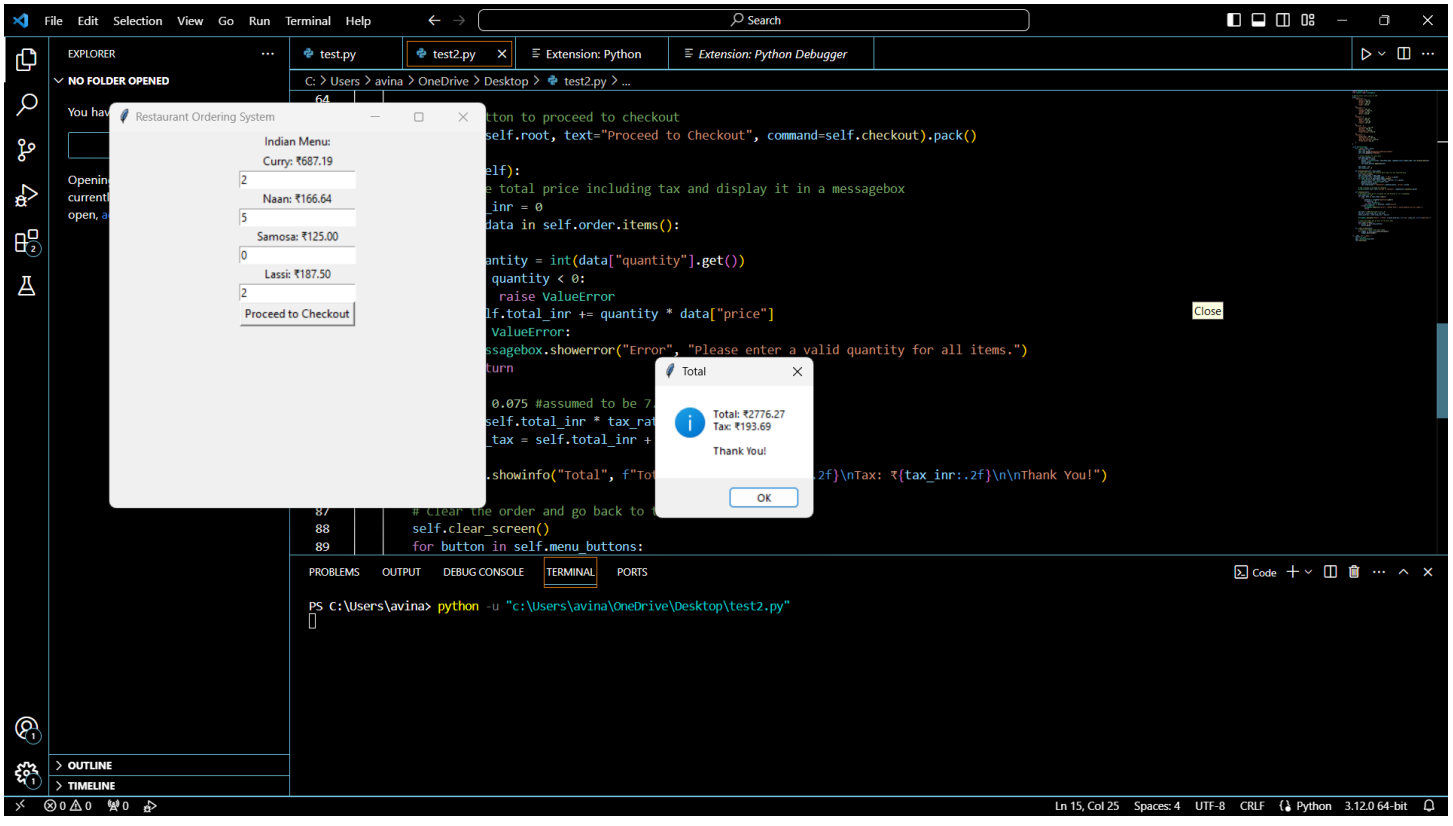


```
        # Add a button to proceed to checkout
        tk.Button(self.root, text="Proceed to Checkout", command=self.checkout).pack()

    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: ₹{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()

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

Execution of code:




Restaurant Ordering System
Indian Menu:
Curry: ₹687.19
2
Naan: ₹166.64
5
Samosa: ₹125.00
0
Lassi: ₹187.50
2
Proceed to Checkout
PS C:\Users\avina> python -u "c:\Users\avina\OneDrive\Desktop\test2.py"


Restaurant Ordering System
Indian Menu:
Curry: ₹687.19
2
Naan: ₹166.64
5
Samosa: ₹125.00
0
Lassi: ₹187.50
2
Proceed to Checkout
Total
Total: ₹2776.27
Tax: ₹193.69
Thank You!
OK
PS C:\Users\avina> python -u "c:\Users\avina\OneDrive\Desktop\test2.py"

References:

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