Restaurant Food Ordering System

Date: 27th November 2020

SAP ID: Neel Karia (60001190033), Neel Nilesh Panchal (6001190034), Parth Dhund (6001190037)

Batch: B3 Elex

Introduction:

This project aims to implement an order management system for restaurants. This system was developed using Python and its libraries, such as Pandas (for handling data) and Tkinter (for the system's GUI).

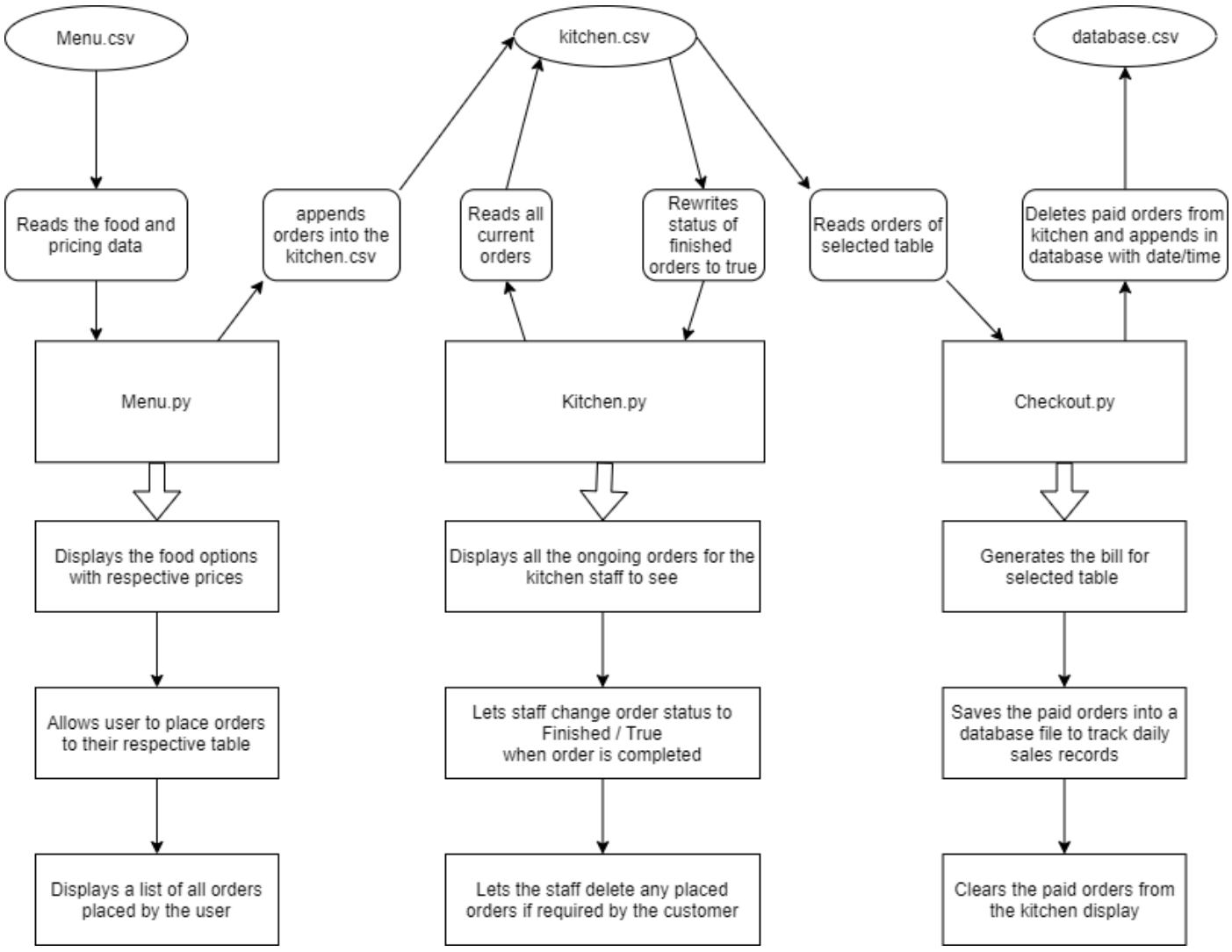
To keep the project as close to an actual life situation as possible, the system has been divided into three main files with their GUIs: one for the customer end, one for the kitchen staff, and one for the cashier.

The customers place their order through the Menu GUI, which displays the Dishes and Price on the left screen and current orders on the right for the selected table. Once the order has been placed, it is written in the Kitchen.csv file.

The Kitchen GUI, which is supposed to be controlled by the kitchen staff, displays all current orders with their status and table numbers. If an order is completed, they can turn its status to proper/completed. Customers who wish to cancel a particular order can do so using the same window. The canceled orders are removed from Kitchen.csv, and the orders with the status ‘True’ are processed for bill generation.

The cashier runs the Checkout GUI, which generates and displays the bill for the order of the selected table (Provided all orders are completed). After bill generation, the order from the kitchen is cleared.

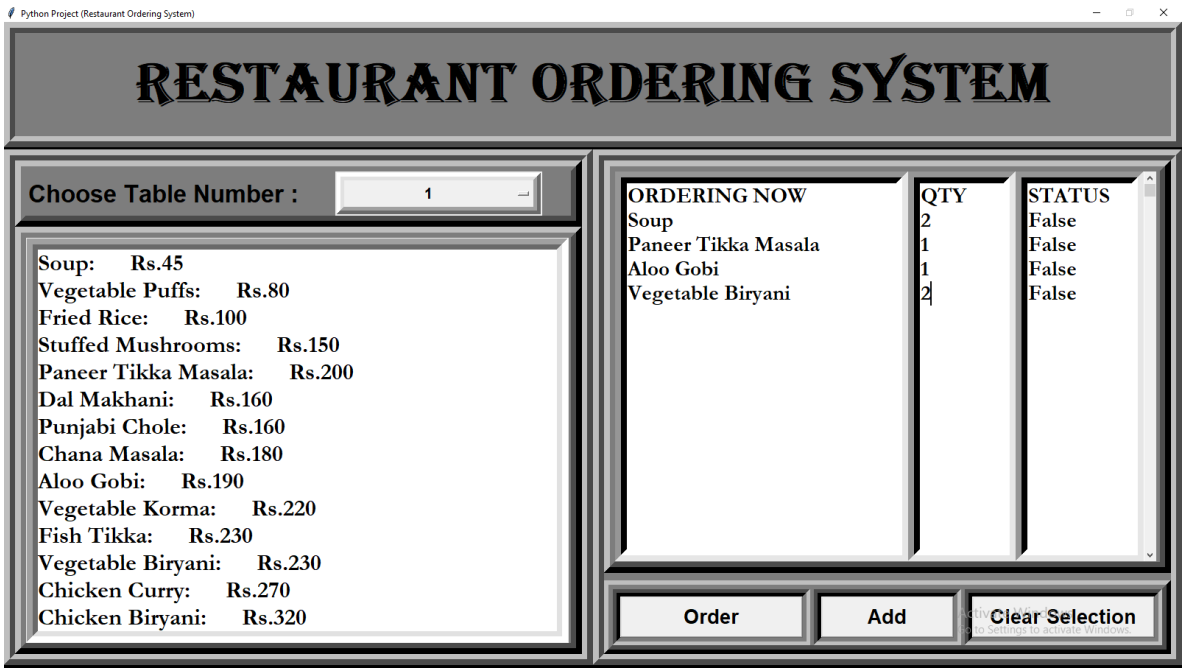
Flow chart:


Menu.csv
kitchen.csv
database.csv
Reads the food and pricing data
appends orders into the kitchen.csv
Reads all current orders
Rewrites status of finished orders to true
Reads orders of selected table
Deletes paid orders from kitchen and appends in database with date/time
Menu.py
Kitchen.py
Checkout.py
Displays the food options with respective prices
Displays all the ongoing orders for the kitchen staff to see
Generates the bill for selected table
Allows user to place orders to their respective table
Lets staff change order status to Finished / True when order is completed
Saves the paid orders into a database file to track daily sales records
Displays a list of all orders placed by the user
Lets the staff delete any placed orders if required by the customer
Clears the paid orders from the kitchen display

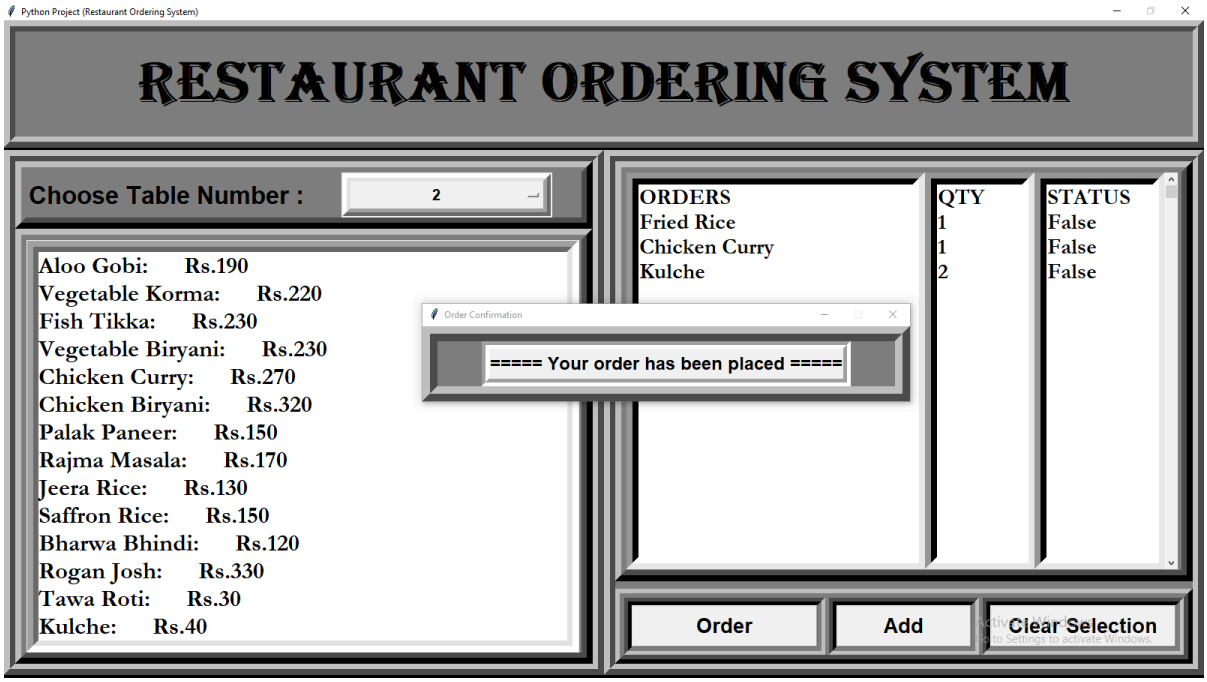
Output screenshots:

- **Menu.py**

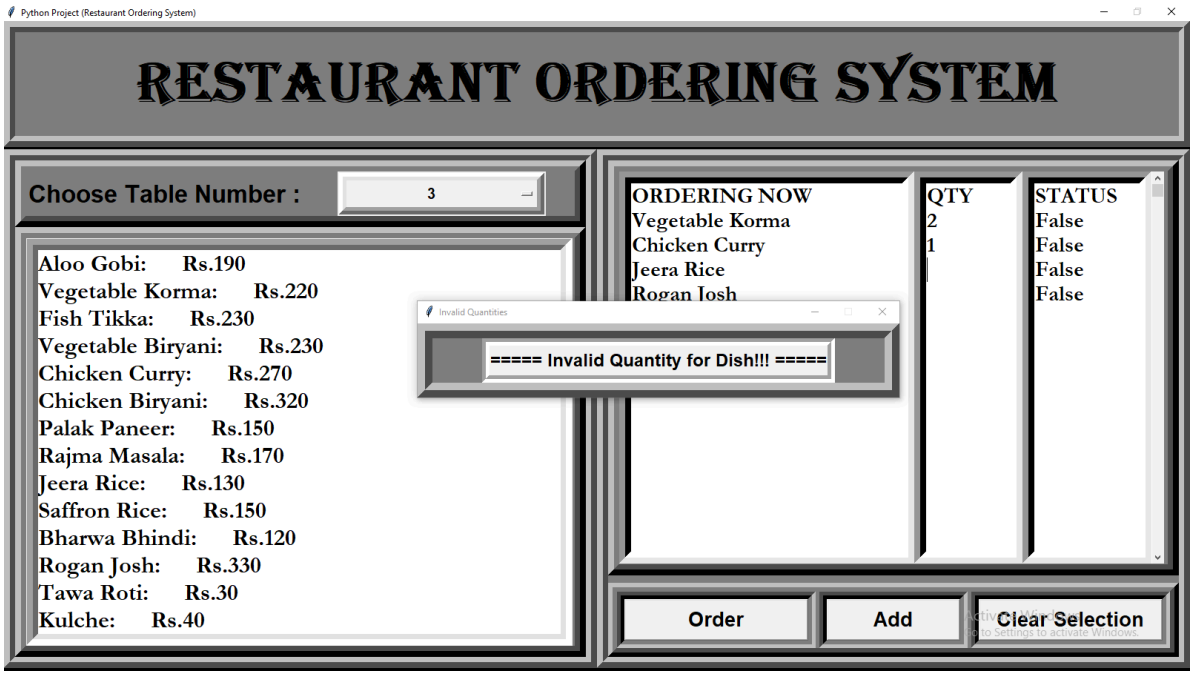
1. Initial menu window that the customer will use to place orders



2. Pop-up for finalization of orders

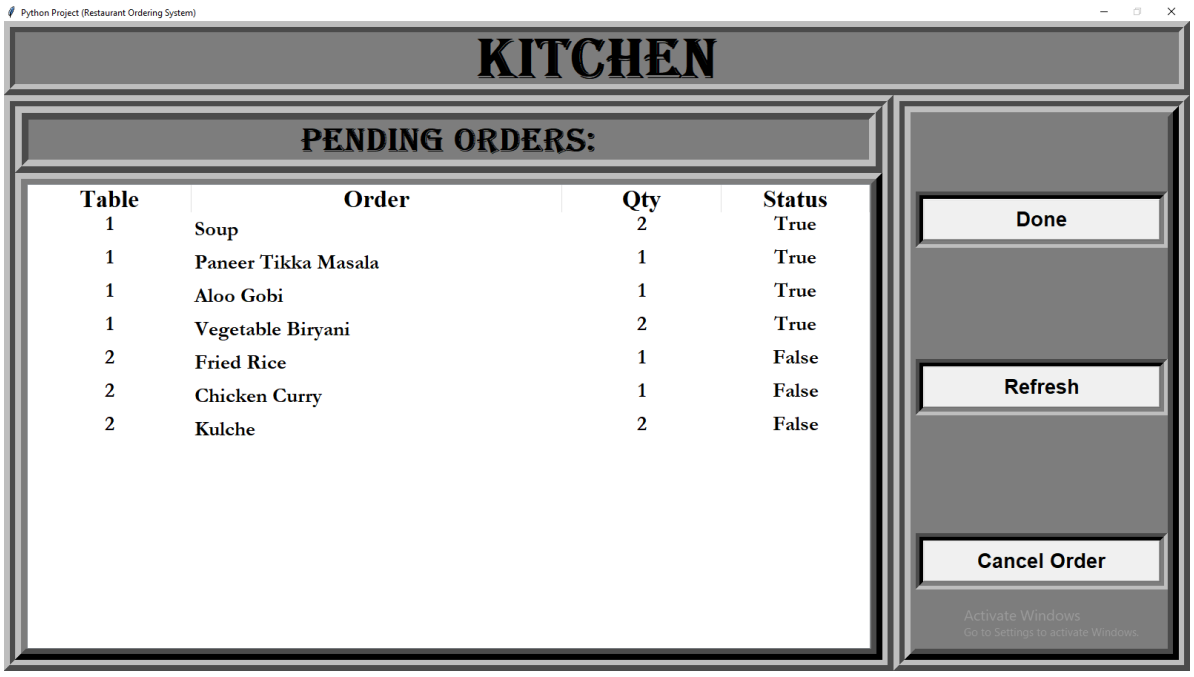


3. Alert for inappropriate input of quantities



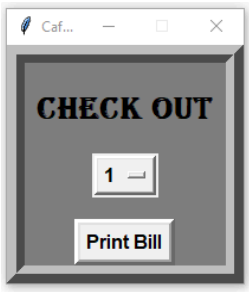
- **Kitchen.py**

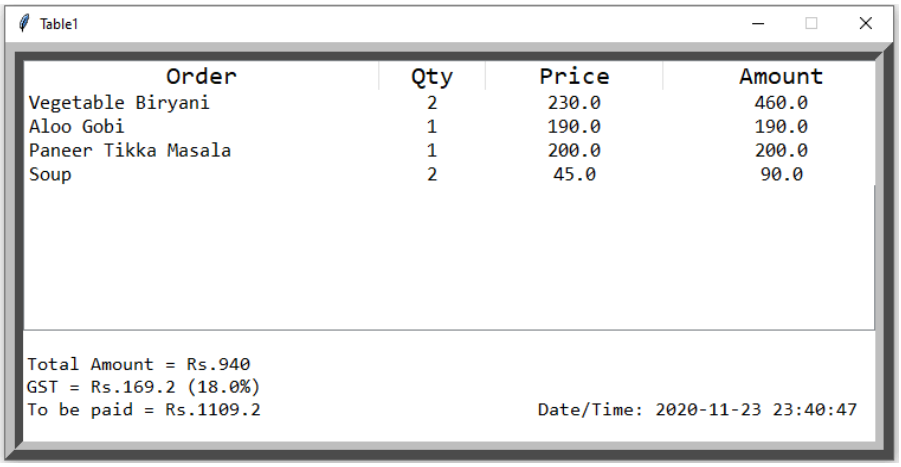
1. Kitchen window displaying the current orders



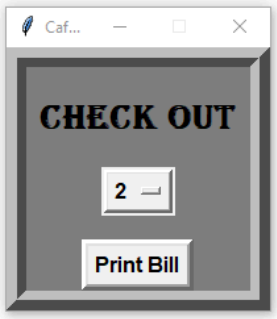
- **Checkout.py**

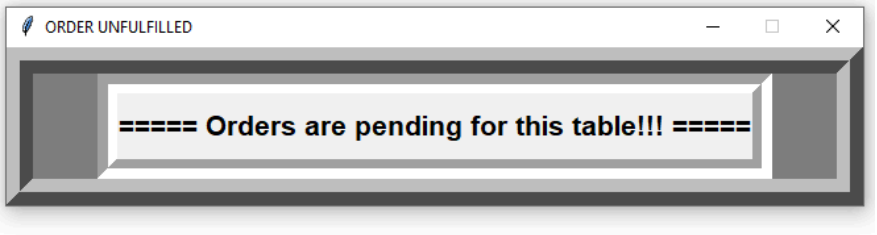
1. Generated bill for orders of “table 1”



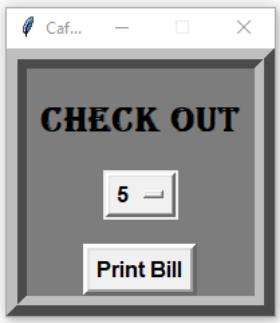


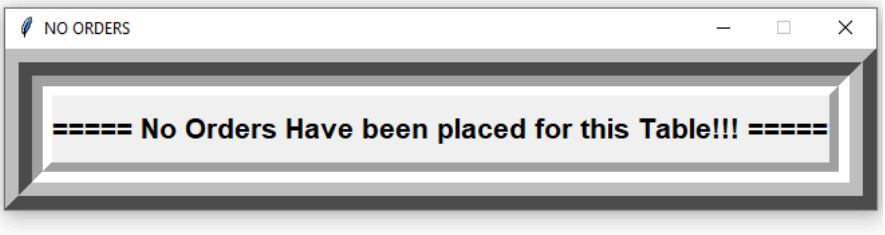
2. Orders pending for “table 2,” as seen in the kitchen window





3. Alert for No orders being placed for “table 5”





Codes:

- **Menu.py**

```
#__________function to stop individual scrolling of textboxes
def scrollwheel(event):
    return 'break'


#____________________function for scrollbar
def onscroll(axis, *args):
    global Tqty,Tstat,Torder
    Tqty.yview(*args)
    Torder.yview(*args)
    Tstat.yview(*args)


#____________________ Function for Add Button to add dishes
def add_item():
    x = []
    y = []
    b = True
    sel_order1 = [str(box1.get(idx)) for idx in box1.curselection()]
    #Separating SELECTED Order and Price from Listbox into 2 lists
    for j in sel_order1:
        l = j.split(sep=":      Rs.")
        for s in l:
            if b == True:
                x.append(s)
                b = False
            else:
                y.append(s)
                b = True
```

```
    global to_kitchen

    to_kitchen = pd.DataFrame()


to_kitchen["order"],to_kitchen["price"],to_kitchen["qty"],to_kitchen["status"],
to_kitchen["table"] = x,y,1,"False",int(table.get())

    to_kitchen = to_kitchen[["table","order","qty","price","status"]]



    #Changing state of textbox for editing

    Torder.configure(state='normal')

    Tstat.configure(state='normal')

    Tqty.configure(state='normal')

    #Clearing Textboxes of previous values

    Torder.delete('1.0',END)

    Tqty.delete('1.0',END)

    Tstat.delete('1.0',END)

    # _____________Inserting added items to textboxes

    Torder.insert('end',"ORDERING NOW" +"\n")

    Tqty.insert('end',"QTY" + "\n")

    Tstat.insert('end',"STATUS" + "\n")

    for i in range(len(to_kitchen["order"])):

        Torder.insert('end',to_kitchen["order"].tolist()[i] +"\n")

        Tqty.insert('end',str(to_kitchen["qty"].tolist()[i]) + "\n")

        Tstat.insert('end', str(to_kitchen["status"].tolist()[i]) + "\n")

    #Changing state of textbox to stop editing once orders are entered

    Torder.configure(state='disabled')

    Tstat.configure(state='disabled')

    global b_add

    b_add = True


#_________________Clearing selections
```

```
def clear_sel():

    box1.selection_clear(0,END)

    change_table()


#_____________Pop-up for order confirmation

def confirm(*args):

    confirm_frame = Toplevel(relief='ridge', bd=20, bg='grey')

    confirm_frame.title("Order Confirmation")

    confirm_frame.geometry("650x100+550+400")

    confirm_frame.resizable(0, 0)

    confirm_label = Label(confirm_frame, text="===== Your order has been placed
=====", font=('Arial', 18, 'bold'), bd=10, relief='groove',pady=20)

    confirm_label.pack()


#_________________________Pop-up for no orders

def no_order(*args):

    no_order_frame = Toplevel(relief='ridge', bd=20, bg='grey')

    no_order_frame.title("Order Confirmation")

    no_order_frame.geometry("650x100+550+400")

    no_order_frame.resizable(0, 0)

    no_order_label = Label(no_order_frame, text="===== No new orders added!!!
=====", font=('Arial', 18, 'bold'), bd=10, relief='groove',pady=20)

    no_order_label.pack()


#_________________________

def order_button(*args):

    if b_add == False:

        no_order()

        return

    qty = Tqty.get("1.0","end-1c").split("\n")
```

```
    while ("" in qty):

        qty.remove("")

    if len(qty[1:]) != len(to_kitchen["qty"].tolist()):

        confirm_frame = Toplevel(relief='ridge', bd=20, bg='grey')

        confirm_frame.title("Invalid Quantities")

        confirm_frame.geometry("650x100+550+400")

        confirm_frame.resizable(0, 0)

        confirm_label = Label(confirm_frame, text="===== Invalid Quantity for
Dish!!! =====", font=('Arial', 18, 'bold'),bd=10, relief='groove', pady=20)

        confirm_label.pack()

        return

    to_kitchen["qty"] = qty[1:]

    for i in range(len(to_kitchen["qty"].tolist())):

            if int(to_kitchen["qty"][i]) < 1:

                to_kitchen.drop(i, inplace = True)

    to_kitchen.to_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv",mode='a',header=False,index=False)# FILE LOC HERE

    change_table()

    confirm()



def change_table(*args):

    #Changing state of textbox for editing

    Torder.configure(state='normal')

    Tstat.configure(state='normal')

    Tqty.configure(state='normal')

    df_orders=pd.read_csv("D:\Local Disk\CLG\Python\Mini project\Kitchen.csv")
#________FILE LOC HERE

    #Dropping orders of tables other than the selected table

    for n in range(len(df_orders["order"])):

        if int(df_orders["table"][n]) != int(table.get()):

            df_orders.drop(n, inplace = True)
```

```
    df_orders.drop(["table"], axis = 1, inplace = True)
    #Clearing Textboxes of previous values
    Torder.delete('1.0',END)
    Tqty.delete('1.0',END)
    Tstat.delete('1.0',END)
    #Inserting current table's orders to textboxes
    Torder.insert('end',"ORDERS" +"\n")
    Tqty.insert('end',"QTY" + "\n")
    Tstat.insert('end',"STATUS" + "\n")
    for i in range(len(df_orders["order"])):
        Torder.insert('end',df_orders["order"].tolist()[i] +"\n")
        Tqty.insert('end',str(df_orders["qty"].tolist()[i]) + "\n")
        Tstat.insert('end', str(df_orders["status"].tolist()[i]) + "\n")
    # Changing state of textbox to stop editing once orders are entered
    Torder.configure(state='disabled')
    Tstat.configure(state='disabled')
    Tqty.configure(state='disabled')
    global b_add
    b_add = False


from tkinter import *
import pandas as pd
import tkinter.ttk as ttk


tab_n = [1, 2, 3, 4, 5, 6] #__________LIST OF TABLE NUMBERS
root = Tk()
root.title("Python Project (Restaurant Ordering System)")   #____Header Title
w, h = root.winfo_screenwidth(), root.winfo_screenheight()  #Width and height
according to users PC
t = h - 200 #THIS WAS -150
```

```
m2 = w/2


root.geometry("%dx%d+0+0" % (w, h))     #Defining Window Geometry

root.state('zoomed')

root.configure(bg='black')

root.resizable(0, 0)                    #Disabling Resizing of Window


#__________Frame and Label for Title

title = Frame(root, width=w, bd=15, height=170, relief='ridge', bg='grey')

title.pack(side=TOP)

title_label = Label(title, font=('Algerian', 60, 'bold'), bg='grey',
text="RESTAURANT ORDERING SYSTEM", justify=CENTER)

title_label.pack(pady=20)

title.pack_propagate(0)


#____________________Left Partition

left = Frame(root, width=w/2, height=t, bd=15, relief='ridge', bg='grey')

left.pack(side=LEFT)

left.pack_propagate(0)


#____________________Right Partition

right = Frame(root, width=w/2, height=t, bd=15, relief='ridge', bg='grey')

right.pack(side=RIGHT)

right.pack_propagate(0)


#_____________Frame for table selection button

tab_button_frame = Frame(left,width=w/2,height=90, bd=15, relief='raised',
bg='grey')

tab_button_frame.pack(side=TOP)

tab_button_frame.pack_propagate(0)
```

```
#____________Frame for menu

menu = Frame(left, width=w/2, height=t-90,bd=15,relief='raised',bg='grey')

menu.pack(side=BOTTOM)

menu.grid_propagate(0)

menu.pack_propagate(0)


#_____________Frame for order summary

summary = Frame(right, width=w/2, height=t-140,bd=15,relief='raised',bg='grey')

summary.pack(side=TOP)

summary.pack_propagate(0)

summary.grid_propagate(0)


#_____________Frame for buttons

bottom_frame = Frame(right,width=w/2,height=100, bd=12, relief='raised',
bg='grey')

bottom_frame.pack(side=BOTTOM)

bottom_frame.pack_propagate(0)


#_________________Frame and button for ordering items

order_button_frame =
Frame(bottom_frame,width=w/4,height=90,bd=10,relief='sunken',bg='grey')

order_button_frame.pack(side=LEFT)

order =
Button(order_button_frame,text="Order",command=order_button,width=14,height=1,f
ont=('Arial',20,'bold'))

order.pack()


#_________________Frame and button for clearing selection

clear_button_frame =
Frame(bottom_frame,width=w/4,height=1,bd=10,relief='sunken',bg='grey')

clear_button_frame.pack(side=RIGHT)
```

```
clear = Button(clear_button_frame,text="Clear
Selection",font=('Arial',20,'bold'),width=14,height=1,command=clear_sel)

clear.pack()



#___________________Frame and button for adding items

add_button_frame =
Frame(bottom_frame,width=w/4,height=1,bd=10,relief='sunken',bg='grey')

add_button_frame.pack(side=BOTTOM)

add =
Button(add_button_frame,text="Add",font=('Arial',20,'bold'),width=10,height=1,c
ommand=add_item)

add.pack()



table = StringVar(root)     #Defining table for optionmenu

table.set('Table Number')  # set the default option



#_____________________OptionMenu for table numbers

table_menu = OptionMenu(tab_button_frame, table, *tab_n)

table_menu.configure(width=20,height=3,font=('Arial',15,'bold'),relief='raised'
,bd=10)

Label(tab_button_frame, text="Choose Table Number : ",font=('Arial',25,'bold'),
bg='grey').pack(side=LEFT)

table_menu.pack()

table.trace('w',change_table)



#_____________________________Listbox for Menu

box1 =
Listbox(menu,bd=15,height=14,width=47,font=('Garamond',24,'bold'),selectmode="m
ultiple")

box1.pack(side=LEFT)

box1.pack_propagate(0)



#_________________Common Scrollbar for textboxes
```

```
yscrollbar = Scrollbar(summary, orient='vertical',command=lambda *args:
onscroll('y-axis', *args))

yscrollbar.grid(row=0, column=4,sticky=N+S+W)



#_______Textbox for orders

Torder = Text(summary, bg='white', bd=16, height=int(t/45),
width=int(m2/30),font=('Garamond',22,'bold'))

Torder.grid(row=0, column=1)

Torder.bind('<MouseWheel>', scrollwheel)



#_______Textbox for quantity

Tqty = Text(summary, bg='white', bd=16, height=int(t/45),
width=int(m2/90),font=('Garamond',22,'bold'))

Tqty.grid(row=0,column=2)

Tqty.bind('<MouseWheel>', scrollwheel)



#_______Textbox for status

Tstat = Text(summary, bg='white', bd=16, height=int(t/45),
width=int(m2/75),font=('Garamond',22,'bold'))

Tstat.grid(row=0,column=3)

Tstat.bind('<MouseWheel>', scrollwheel)    #Binding scrollwheel function to
mouse-scroll event



#_________reading menu dataframe

df_dish = pd.read_csv("D:\Local Disk\CLG\Python\Mini
project\Menu.csv")#_____________________FILE LOCATION HERE



#_____________________Inserting Order and Price from menu into listbox

for i in range(len(df_dish["Dish"])):

    box1_el = str(df_dish["Dish"][i])+":      Rs."+ str(df_dish["Price"][i])

    box1.insert('end',box1_el)

root.mainloop()             #_____________Keeps the root window in loop
```

- **Kitchen.py**

```
#_____________________________________importing all the required libraries

from tkinter import *

from tkinter import font

import pandas as pd

import tkinter.ttk as ttk



#_____________________________________________________function for the Confirm
Button for finishing Order

def confirm(*args):



    df_dish = pd.read_csv("D:\Local Disk\CLG\Python\Mini project\Kitchen.csv")
#_____________________________________________File Loc Here

    x = kitchen.item(kitchen.selection())['values'][0]

    df_dish.loc[x,"status"] = True

    df_dish.to_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv",index=False)
#_____________________________________________File Loc Here



#_____________________________________Order Confirmation

    confirm_frame = Toplevel(relief='ridge', bd=20, bg='grey')

    confirm_frame.title("Billing Confirmation")

    confirm_frame.geometry("650x100+600+400")

    confirm_label = Label(confirm_frame, text="===== Order is ready to be
served =====", font=('Arial', 20, 'bold'), bd=15, relief='groove',pady=5)

    confirm_label.pack()
```

```
    refresh()



#_____________________________________________function for Refresh Button to
Display Orders with status update

def refresh():

    kitchen.delete(*kitchen.get_children())

    df_dish = pd.read_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv")#_____________________________________________________
_____________File Loc Here

    for i in range(len(df_dish["order"])):


kitchen.insert(parent='',index='end',values=(i,df_dish["table"][i],df_dish["ord
er"][i],df_dish["qty"][i],df_dish["status"][i]))     #inserting new values to
treeview



#________function to cancel an order

def cancel():



    df_dish = pd.read_csv("D:\Local Disk\CLG\Python\Mini project\Kitchen.csv")
#_____________________________________________File Loc Here

    x = kitchen.item(kitchen.selection())['values'][0]



    df_dish.drop(x, inplace = True)

    df_dish.to_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv",index=False)
#_____________________________________________File Loc Here



#___________Order cancellation

    cancel_win_frame = Toplevel(relief='ridge', bd=20, bg='grey')

    cancel_win_frame.title("Billing Confirmation")

    cancel_win_frame.geometry("650x100+600+400")

    cancel_win_label = Label(cancel_win_frame, text="===== Order is Cancelled
=====", font=('Arial', 20, 'bold'), bd=15, relief='groove',pady=5)

    cancel_win_label.pack()

    refresh()
```

```
#____________________Start of Mainloop GUI


tab_n = [1, 2, 3, 4, 5, 6]                    #LIST of Table Numbers

root = Tk()

root.title("Python Project (Restaurant Ordering System)")            #____Header
Title

w, h = root.winfo_screenwidth(), root.winfo_screenheight()

#Width and height according to users PC

t = h - 100




#_____________Main Window

root.geometry("%dx%d+0+0" % (w, h))

root.state('zoomed')

root.configure(bg='black')

root.resizable(0, 0)


#_____________Title of Main Window

title = Frame(root, width=w, bd=15, height=100, relief='ridge', bg='grey')

title.pack(side=TOP)

title.pack_propagate(0)

title_label = Label(title, font=('Algerian', 60 , 'bold'), bg='grey',
text="KITCHEN", justify=CENTER)

title_label.pack(padx=50)


#____________________Left Partition

left = Frame(root, width=3/4*w, height=t, bd=15, relief='ridge', bg='grey')

left.pack(side=LEFT)

left.pack_propagate(0)
```

```
#_____________________Right Partition

right = Frame(root, width=w/4, height=t, bd=15, relief='ridge', bg='grey')

right.pack(side=RIGHT)

right.pack_propagate(0)



#_____________________Frame and Label for pending orders

pending_frame = Frame(left,width=3/4*w,height=90, bd=18, relief='ridge',
bg='grey')

pending_frame.pack(side=TOP)

pending_frame.pack_propagate(0)

Label(pending_frame, text="Pending Orders:",font=('algerian',35,'bold'),
bg='grey',justify=CENTER).pack()



#_____________________ Frame for menu

menu = Frame(left, width=3/4*w, height=t-90,bd=15,relief='raised',bg='grey')

menu.pack(side=BOTTOM)

menu.grid_propagate(0)

menu.pack_propagate(0)



#_____________________Display window for Orders

style = ttk.Style()

style.configure("Treeview", highlightthickness=0, bd=10,
font=('Garamond',21,'bold'),rowheight=45)  # Modify the font of the body

style.configure("Treeview.Heading", font=('Times New Roman', 25,'bold'))

kitchen = ttk.Treeview(menu,
columns=("Index","Table","Order","Qty","Status"),height=29,selectmode='browse')

kitchen["displaycolumns"]=("Table","Order","Qty","Status")
#_____________________HIDES INDEX COLUMN

kitchen['show'] = 'headings'

kitchen.pack()

kitchen.pack_propagate(0)
```

```
kitchen.column("0",width=220,anchor='n')

kitchen.column("1",width=220,anchor='n')

kitchen.column("2",width=500,anchor='w')

kitchen.column("3",width=215,anchor='n')

kitchen.column("4",width=200,anchor='n')

kitchen.heading("1", text="Table")

kitchen.heading("2", text="Order")

kitchen.heading("3", text="Qty")

kitchen.heading("4", text="Status")



#____________________ Right Window GUI

right_frame = Frame(right,width=w/2,height=t, bd=15, relief='raised',
bg='grey')

right_frame.pack(side=RIGHT)

right_frame.pack_propagate(0)



#____________________Done Button GUI

done_button_frame =
Frame(right_frame,width=w/4,height=1,bd=10,relief='sunken',bg='grey')

done_button_frame.pack(side=TOP,pady=100)

done =
Button(done_button_frame,text="Done",width=20,height=1,font=('Arial',20,'bold')
, command= confirm)

done.pack()



#____________________Refresh Button GUI

refresh_button_frame =
Frame(right_frame,width=w/4,height=1,bd=10,relief='sunken',bg='grey')

refresh_button_frame.pack(side=TOP,pady=50)

refresh_button =
Button(refresh_button_frame,text="Refresh",width=20,height=1,font=('Arial',20,'
bold'),command= refresh)

refresh_button.pack()
```

```
#_____________________Cancel Button GUI

cancel_button_frame =
Frame(right_frame,width=w/4,height=20,bd=10,relief='sunken',bg='grey')

cancel_button_frame.pack(side=BOTTOM,pady = 80)

cancel = Button(cancel_button_frame,text="Cancel
Order",font=('Arial',20,'bold'),width=20,height=1, command= cancel)

cancel.pack()



#_____________________FILE Reading for Orders:

df_dish = pd.read_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv")#_____________________________File loc here

for i in range(len(df_dish["order"])):


kitchen.insert(parent='',index='end',values=(i,df_dish["table"][i],df_dish["ord
er"][i],df_dish["qty"][i],df_dish["status"][i]))

#inserting values to treeview



root.mainloop()  #_______Keeps the root window in loop
```

- **Checkout.py**

```
# import libraries

from tkinter import *

import pandas as pd

import tkinter.ttk as ttk

from datetime import datetime


# GST VARIABLE

GST = 0.18

# Function for when the button is pressed

def button_bill():

    # Reading CSV file

    df_bill = pd.read_csv("D:\Local Disk\CLG\Python\Mini project\Kitchen.csv")
#_____________________FILENAME(Bill)

    df_return = pd.read_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv")  # _____________________FILENAME(Return)

    df_bill["amount"] = df_bill["qty"] * df_bill["price"]



    for i in range(len(df_bill["amount"])):

        if int(df_bill["table"][i]) != int(table.get()):  # Dropping unwanted
table orders from bill

            df_bill.drop(i, inplace=True)



        elif df_bill["status"][i] == False:  # Checking if all orders are done
```

```
            #_____Pop-up for unfulfilled orders

            unfulfilled = Toplevel(relief='ridge', bd=20, bg='grey')

            unfulfilled.title("ORDER UNFULFILLED")

            unfulfilled.geometry("650x120+550+400")

            unfulfilled.resizable(0, 0)

            unfulfilled_label = Label(unfulfilled, text="===== Orders are
pending for this table!!! =====",

                                      font=('Arial', 16, 'bold'), bd=15,
relief='groove', pady=20)

            unfulfilled_label.pack()
            return



        elif int(df_bill["table"][i]) == int(table.get()):  # Dropping paid
orders from Kitchen.CSV

            df_return.drop(i, inplace=True)



    df_bill.drop(["table", "status"], axis=1, inplace=True)  # Dropping Table
number and Order status from final bill



    now = datetime.now()  # datetime object containing current date and time

    df_bill["time"] = now.replace(microsecond=0)  # Adding Timestamp to
database



    #Updating KITCHEN and DATABASE CSV FILES

    df_return.to_csv("D:\Local Disk\CLG\Python\Mini
project\Kitchen.csv",index=False)#_________________FILELOC

    df_bill.to_csv('D:\Local Disk\CLG\Python\Mini project\Database.csv',
mode='a', header=False,index=False)#_____________FILELOC

    if df_bill["amount"].sum() == 0 :  # Checking if table has any orders to
charge a bill

        #_____________________Pop-up for no orders

        no_order_frame = Toplevel(relief='ridge', bd=20, bg='grey')

        no_order_frame.title("NO ORDERS")

        no_order_frame.geometry("650x120+550+400")
```

```
        no_order_frame.resizable(0, 0)

        no_order_label = Label(no_order_frame, text="===== No Orders Have been
placed for this Table!!! =====",

                               font=('Arial', 16, 'bold'), bd=15,
relief='groove', pady=20)

        no_order_label.pack()

        return



    df_bill.drop(["time"], axis=1, inplace=True)  # Removing Timestamp from
dataframe



    #________Final Bill window information

    BillWindow = Toplevel(root)

    BillWindow.title("Table" + table.get())

    BillWindow.geometry("750x350")

    BillWindow.resizable(0, 0)



    #________  Frame for Treeview

    bill_frame = Frame(BillWindow, width=750, height=350, bd=15,
relief='ridge', bg='grey')

    bill_frame.pack()

    bill_frame.pack_propagate(0)



    #___________style configuration for treeview

    style = ttk.Style()

    style.configure("Treeview", font=('Consolas', 13))

    style.configure("Treeview.Heading", font=('Consolas', 16))



    #___________Constructing Treeview

    bill = ttk.Treeview(bill_frame, columns=("Order", "Qty", "Price",
"Amount"), height=10, selectmode='none')

    bill['show'] = 'headings'
```

```
    bill.pack()

    bill.pack_propagate(0)

    bill.column("0", width=300)

    bill.column("1", width=90, anchor='n')

    bill.column("2", width=150, anchor='n')

    bill.column("3", width=200, anchor='n')

    bill.heading("0", text="Order")

    bill.heading("1", text="Qty")

    bill.heading("2", text="Price")

    bill.heading("3", text="Amount")



    #_____________Printing Bill Amount

    Label(bill_frame, bg='white', anchor="w", font=('Consolas', 12),
justify=LEFT, width=300, height=5,

          text="Total Amount = Rs." + str(df_bill["amount"].sum()) +  # Total
and GST inclusion

                "\nGST = Rs." + str((df_bill["amount"].sum() * GST).round(2)) +
" (" + str(100 * GST) + "%)"

                 "\nTo be paid = Rs." + str((df_bill["amount"].sum() * (GST +
1)).round(2)) +

                "\t\t\t\tDate/Time: " +
str(now.replace(microsecond=0))).pack(side=BOTTOM)



    #_____________Inserting Order summary to bill treeview

    df_bill.index = range(0,len(df_bill["order"]))

    for j in range(len(df_bill["order"])):

        bill.insert(parent='', index=0, values=(df_bill["order"][j],
df_bill["qty"][j], float(df_bill["price"][j]), float(df_bill["amount"][j])))



# Main Window

root = Tk()

root.title("Cafe Kill Me")
```

```
root.resizable(0,0)

w, h = root.winfo_screenwidth(), root.winfo_screenheight()

root.geometry("200x200+400+300")

tab_no = [1, 2, 3, 4, 5, 6]  # __LIST OF TABLE NUMBERS



# Bill Window Frame

mainframe = Frame(root, width=200, bd=15, height=200, relief='ridge',
bg='grey')

mainframe.pack()

mainframe.pack_propagate(0)

mainframe_label = Label(mainframe, font=('Algerian', 20, 'bold'), bg='grey',
text="Check Out", justify=CENTER)

mainframe_label.pack(pady=20)



#Selected Table number call is "table.get()"_____________________________

table = StringVar(root)

#_________Option Menu to select Table number

table.set('Table Number')  # set the default option to 1

popupMenu = OptionMenu(mainframe, table, *tab_no)

popupMenu.configure(bd=3, font=('Arial', 12, 'bold'))

popupMenu.pack()



# CHECKOUT Button

button = Button(mainframe, text="Print Bill", command=button_bill,
font=('Arial', 12, 'bold'), bd=5)

button.pack(side=BOTTOM)

root.mainloop()     #_____________Keeps the root window in loop
```

CSV Files :

- **Menu.csv**
  https://drive.google.com/file/d/1qEomG9-NoOOK38o87__FtyNKuP7pBUSD/view?usp=sharing
- **Kitchen.csv**
  https://drive.google.com/file/d/1zxF-M7W5hztk_nxdTu9lIhoI1ZdBrU9R/view?usp=sharing
- **Database.csv**
  https://drive.google.com/file/d/1rIs9xQt4wW1EGIugNGByL849uGYDizsv/view?usp=sharing

Conclusion:

The project was successfully executed with all three GUI windows functioning as desired. Various exceptions and contingencies have also been placed so the program will only misbehave if appropriate inputs are provided.

This project can be further expanded by hosting the three CSV files on a server and accessing them through the Python codes. The system can then mirror a real-life situation in a restaurant with customers ordering on the menu GUIs right from their tables, staff fulfilling the orders from a display in the kitchen, and cashiers automatically being able to generate the final bill during checkout.

References:

Pandas Tutorial - GeeksforGeeks

How to Import an Excel File into Python using Pandas - Data to Fish

pandas.DataFrame.to_csv — pandas 1.1.4 documentation (pydata.org)

Python Tkinter - Menu button Widget - GeeksforGeeks

Open a new Window with a button in Python-Tkinter - GeeksforGeeks

Creating a multiple Selection using Tkinter - GeeksforGeeks

Python - GUI Programming (Tkinter) - Tutorialspoint

Python - Tkinter PanedWindow - Tutorialspoint

GUI Programming with Python: Text Widget (python-course. eu)

button - Adding items to Listbox in Python Tkinter - Stack Overflow

Tkinter TreeView Widget - AskPython

tkinter - Treeview: Basic example | tkinter Tutorial (riptutorial.com)

SettingwithCopyWarning: How to Fix This Warning in Pandas – Dataquest