REPORT FOR PIZZA ORDER

As a project work for Course

PYTHON PROGRAMMING (INT 213)

Name : SACHIN

Registration Number : 12018125

Name : TUSHAR SOLANKI

Registration Number : 12017456

Program : B.Tech. (CSE)

Semester : Third

School : School of Computer Science and Engineering

Name of the University : Lovely Professional University

Date of submission : 20th NOVEMBER 2021

AKNOWLEDGEMENT

I Would like to express my special thanks of gratitude to my teacher Mrs. Ankita Wadhawan who give me the golden opportunity to do this wonderful project on the topic Pizza Order. It helped me increase my knowledge and skills.
I am really thankful to them.

TABLE OF CONTENTS

ABSTARCT: -

The "Pizza Ordering System" has been developed to override the problems prevailing in the participating manual system. This software is supported to eliminate and in some cases reduce the hardships faced by the existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data. No formal knowledge is needed for the user to use this system.

The purpose of the project is to build an application program to reduce the manual work.

INTRODUCTION: -

2.1 CONTEXT: - This project has been done as part of my course for the CSE at Lovely Professional University. Supervised by Ankita Wadhawan, I have one month to fulfill the requirements in order to succeed the module.

2.2 INTRODUCTION: - A pizzeria specialized in custom made pizzas is currently taking orders by phone. The current system where the customer calls the pizzeria takes time of employees to answer the phone and is more work consuming than necessary. They want to allow customers to customize and order their pizzas online. The pizzeria also aims to increase the sales, due to the easy to use order online. The system will give the employees more time to work rather then to accept orders by phone, also the potential increase in customers are enough reason for the pizzeria to accept the change.

2.3 GOAL: - Our goal is to deliver a database with a user interface where customers can select various ingredients for their own pizza and place their order.. The focus is to create an easy to use, which will allow a first time customer to complete their order with ease.

TEAM MEMBERS: -

SACHIN: -

CONTRIBUTION: -

1. Coding
2. Report

TUSHAR SOLANKI: -

CONTRIBUTION: -

1. Coding
2. Report

LIBRARIES: -

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Some Tkinter widgets are:

1	Button The Button widget is used to display buttons in your application.
2	Checkbutton The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
3	Entry The Entry widget is used to display a single-line text field for accepting values from a user.
4	Label The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
5	Listbox The Listbox widget is used to provide a list of options to a user.
6	Menubutton The Menubutton widget is used to display menus in your application.
7	Menu The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

8	Message The Message widget is used to display multiline text fields for accepting values from a user.
9	Radiobutton The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
10	Text The Text widget is used to display text in multiple lines.

SCREENSHOTS: -

CODE: -

```
from tkinter import *
from random import *
from datetime import datetime
import tkinter.messagebox as tkMessageBox
import sqlite3

conn=sqlite3.connect('Pizza_delivery.db')
c = conn.cursor()

'''The following four comments needs to be executed before starting the first time implementation of the project
   because they are needed to create the tables once'''
c.execute("CREATE TABLE IF NOT EXISTS Pizza(Order_no int,Name text,Address text,type int,mobile int,email text,time text)")
c.execute("CREATE TABLE IF NOT EXISTS Canceled_Pizza(Order_no int,Name text,Address text,type int,mobile int,email text,time text)")
c.execute("CREATE TABLE IF NOT EXISTS Served_Pizza(Order_no int,Name text,Address text,type int,mobile int,email text,time text)")
c.execute("CREATE TABLE IF NOT EXISTS Pending_Pizza(Order_no int,Name text,Address text,type int,mobile int,email text,time text)")

'''These following four comments can be executed according to the need if in case we need to drop the existing tables'''

# c.execute("Drop Table Pizza")
# c.execute("Drop Table Canceled_Pizza")
# c.execute("Drop table Served_Pizza")
# c.execute("Drop table Pending_Pizza")


global x, orders, getname, getaddress, order_number, m, canceled, cancel_times, var
x = int((random() * 10000))
orders = []
times = []
getname = []
getaddress = []
canceled = []
cancel_times = []


def Order():
    def ordernow():
        global getname, getaddress, m
        try:
            getname.append(entername.get())
            getaddress.append(enteraddress.get())
            gettype = int(m)
            getmob = str(entermob.get())
            getmail = entermail.get()
            if (getname == "" or getaddress == "" or getmob == "" or getmail == ""):
                tkMessageBox.showinfo("Empty Field", "Any Field can't be EMPTY")
                order.destroy()
            else:
                global times, x, orders
                now = (datetime.now())
                d = str(
                    str(now.hour) + ":" + str(now.minute) + "::" + str(now.second) + " on " + str(now.day) + "/" + str(
                        now.month) + "/" + str(now.year))
                times.append(d)
                x += 1
                string = "Your order id is: " + str(x)
                tkMessageBox.showinfo("Order Placed", string)
                orders.append(x)
                i = (len(getname) - 1)
                make_list = [x, getname[i], getaddress[i], gettype, getmob, getmail, d]
                c.execute("INSERT INTO Pizza VALUES(?,?,?,?,?,?,?)", make_list)
                conn.commit()
```

```
            conn.commit()
            order.destroy()
    except NameError or ValueError or UnboundLocalError:
        tkMessageBox.showerror("Error", "Please enter valid ceredentials")
        order.destroy()

order = Tk()
global var
var = IntVar()

def sel():
    global m, var
    m = int(var.get())
    return

order_w = Label(order, text="Order Pizza", bg="black", fg="white")
order_w.grid(row=0, column=2)

label_a = Label(order, text="  ")
label_a.grid(row=1)

name = Label(order, text="Name:  ")
name.grid(row=2, column=1)

entername = Entry(order, bd=3)
entername.grid(row=2, column=2)

address = Label(order, text="Address:   ")
address.grid(row=3, column=1)

enteraddress = Entry(order, bd=3)
enteraddress.grid(row=3, column=2)

ptype = Label(order, text="Pizza Type:       ")
ptype.grid(row=4, column=1)

R1 = Radiobutton(order, text="Small (95/-)", value=1, variable=var, command=sel)
R1.grid(row=4, column=2, sticky=W)
R2 = Radiobutton(order, text="Medium (195/-)", value=2, variable=var, command=sel)
R2.grid(row=4, column=3, sticky=W)
R3 = Radiobutton(order, text="Big (295/-)", value=3, variable=var, command=sel)
R3.grid(row=4, column=4, sticky=W)

mobile = Label(order, text="Mobile no:      ")
mobile.grid(row=5, column=1)

entermob = Entry(order, bd=3)
entermob.grid(row=5, column=2)

email = Label(order, text="Email id:      ")
email.grid(row=6, column=1)

entermail = Entry(order, bd=3)
entermail.grid(row=6, column=2)

order_now = Button(order, text="Order Now", bg="yellow", fg="blue", command=ordernow)
order_now.grid(row=7, column=2)

order.mainloop()
```

```
def Cancel():
    def cancel_now():
        if (enter_order.get() == "" or enter_name.get() == ""):
            tkMessageBox.showinfo("Empty Field", "Any Field can't be EMPTY")
            cancel.destroy()
        else:
            y = [int(enter_order.get())]
            conn = sqlite3.connect('Pizza_delivery.db')
            c = conn.cursor()
            flag = 0
            for row in c.execute('Select Order_no from Pizza'):
                if str(row) == str(tuple(y)):
                    flag = 1
                    break
            y = int(str(y[0]))
            l = []
            l.append(y)

            if (flag == 1):
                c.execute('Insert into Canceled_Pizza Select * from Pizza where Order_no=?', l)
                c.execute('Delete from Pizza where Order_no=?', l)
                conn.commit()
                tkMessageBox.showinfo("Order Cancelled Succesfully", "Your Order has been cancelled")
                global orders, canceled, cancel_times
                canceled.append(y)
                now = datetime.now()
                d = str(
                    str(now.hour) + ":" + str(now.minute) + "::" + str(now.second) + " on " + str(now.day) + "/" + str(
                        now.month) + "/" + str(now.year))
                cancel_times.append(d)
                if (len(orders) > 0):
                    i = -1
                    while i < len(orders):
                        i += 1
                        if y == orders[i]:
                            break

                    del orders[i]
            else:
                tkMessageBox.showinfo("Invalid Order id", "Order ID you entered is not correct")
                cancel.destroy()
                exit

    cancel = Tk()

    label_a = Label(cancel, text="  ")
    label_a.grid(row=1)

    c = Label(cancel, text="Cancel Order", bg="black", fg="white")
    c.grid(row=2, column=2)

    label_a = Label(cancel, text="  ")
    label_a.grid(row=3)

    name = Label(cancel, text="Name:       ")
    name.grid(row=4, column=1)

    enter_name = Entry(cancel, bd=3)
    enter_name.grid(row=4, column=2)

    Order_id = Label(cancel, text="Order ID:     ")
    Order_id.grid(row=5, column=1)

    enter_order = Entry(cancel, bd=3)
    enter_order.grid(row=5, column=2)

    label_a = Label(cancel, text="  ")
    label_a.grid(row=6)

    Cancel_now = Button(cancel, text="Cancel Now", bg="yellow", fg="Blue", command=cancel_now)
    Cancel_now.grid(row=7, column=2)

    label_a = Label(cancel, text="      ")
    label_a.grid(column=3)

    cancel.mainloop()


def Track():
    def track_now():
        y = [int(enter_order.get())]
        conn = sqlite3.connect('Pizza_delivery.db')
        c = conn.cursor()
        flag = 0
        for row in c.execute('Select Order_no from Pizza'):
            if str(row) == str(tuple(y)):
                flag = 1
                break
        y = int(str(y[0]))
        l = []
        l.append(y)
        if (flag == 1):
            for row in c.execute('Select time from Pizza where Order_no=?', l):
                t = str(row)
            if t[4] != ":":
                thour = str(t[3]) + str(t[4])
                if t[7] != ":":
                    tminute = str(t[6]) + str(t[7])
                else:
                    tminute = str(t[6])
            else:
                thour = str(t[3])
                if t[6] != ":":
                    tminute = str(t[5]) + str(t[6])
                else:
                    tminute = str(t[5])
            tdate = str(t[-13]) + str(t[-12])

            now = datetime.now()
            ttime = now.replace(day=int(tdate), hour=int(thour), minute=int(tminute), second=0, microsecond=0)
            if (now.day > ttime.day):
                tkMessageBox.showinfo("Delivered", "Your Order is already Delivered")
                track.destroy()
            else:
                if ttime.day == now.day and ttime.hour == now.hour and now.minute <= ttime.minute + 20:
                    tkMessageBox.showinfo("Track Order",
                                          "Your order is Preparing\nIt will be delivered within 20 minutes of order")
                    track.destroy()
                else:
                    tkMessageBox.showinfo("Track Order", "Your order is Ready\nIt will be delivered soon")
```

```
                tkMessageBox.showinfo("Track Order", "Your order is Ready\nIt will be delivered soon")
                track.destroy()
        else:
            tkMessageBox.showinfo("Track Order", "Please enter a valid Order ID")

    track = Tk()

    label_a = Label(track, text="  ")
    label_a.grid(row=1)

    c = Label(track, text="Track Order", bg="black", fg="white")
    c.grid(row=2, column=2)

    label_a = Label(track, text="  ")
    label_a.grid(row=3)

    order = Label(track, text="Order ID:       ")
    order.grid(row=4, column=1)

    enter_order = Entry(track, bd=3)
    enter_order.grid(row=4, column=2)

    label_a = Label(track, text="  ")
    label_a.grid(row=5)

    Track_now = Button(track, text="Track Now", bg="yellow", fg="Blue", command=track_now)
    Track_now.grid(row=6, column=2)

    label_a = Label(track, text="      ")
    label_a.grid(column=3)

    label_a = Label(track, text="      ")
    label_a.grid(column=4)

    label_a = Label(track, text="      ")
    label_a.grid(column=5)

    track.mainloop()


def NewPizzaOrder():
    global orders, times
    new_orders = Tk()
    t = Text(new_orders, height=15, width=100)
    t.pack()
    s = "\n                    RECENT ORDERS\n\nORDER NO    NAME     ADDRESS      TIME \n"
    t.insert(END, s)
    i = 0
    for i in range(len(orders)):
        string = str(orders[i]) + "     " + str(getname[i]) + "    " + str(getaddress[i]) + "    " + str(times[i]) + "\n"
        t.insert(END, string)

    t2 = Text(new_orders, height=18, width=100)
    t2.pack()
    t2.insert(END, "                         ALL ORDERS\n")
    t2.insert(END, "\nID,  NAME,  ADDRESS, TYPE, MOBILE,  EMAIL,    TIME/DATE\n")

    for row in c.execute('select * from Pizza'):
        t2.insert(END, row)
        t2.insert(END, "\n")
    conn.commit()
    conn.commit()
    new_orders.mainloop()


def CanceledOrder():
    global canceled, cancel_times
    canceled_orders = Tk()

    t = Text(canceled_orders, height=15, width=100)
    t.pack()
    s = "\n              RECENTLY CANCELED ORDERS\n\nORDER NO                      TIME\n"
    t.insert(END, s)
    i = 0
    while i < (len(canceled)):
        string = str(canceled[i]) + "              " + str(cancel_times[i]) + "\n"
        t.insert(END, string)
        i += 1
    t2 = Text(canceled_orders, height=18, width=100)
    t2.pack()
    s = "\n          ALL CANCELED ORDERS\n\n"
    t2.insert(END, s)
    t2.insert(END, "\nID,  NAME,  ADDRESS, TYPE, MOBILE,  EMAIL,    TIME/DATE\n")

    for row in c.execute('select * from Canceled_Pizza'):
        t2.insert(END, row)
        t2.insert(END, "\n")
    conn.commit()
    canceled_orders.mainloop()


def ServedOrder():
    served = Tk()
    ser = []
    t = Text(served, height=15, width=100)
    t.pack()
    s = "\n        SERVED ORDERS  \n"
    t.insert(END, s)
    t.insert(END, "\nID,  NAME,  ADDRESS, TYPE, MOBILE,  EMAIL,    TIME/DATE\n")
    for row in c.execute('Select * from Pizza'):
        ser.append(str(row))

    i = 0
    while i < len(ser):
        row = ser[i]
        ts = str(row)
        if (ts[-12] != " "):
            tt = ts[-12] + ts[-11]
        else:
            tt = ts[-11]
        o = str(ts[1:5])
        od = []
        od.append(o)
        if (ts[-12] != " "):
            if (ts[-18] != ":"):
                if (ts[-22] != ":"):
                    tm = ts[-22] + ts[-21]
                else:
                    tm = ts[-21]
            else:
                if (ts[-21] != ":"):
                    tm = ts[-21] + ts[-20]
```

```
                tm = ts[-21] + ts[-20]
            else:
                tm = ts[-20]
        else:
            if (ts[-17] != ":"):
                if (ts[-21] != ":"):
                    tm = ts[-21] + ts[-20]
                else:
                    tm = ts[-20]
            else:
                if (ts[-20] != ":"):
                    tm = ts[-20] + ts[-19]
                else:
                    tm = ts[-19]
        now = datetime.now()
        ttime = now.replace(day=int(tt), hour=0, minute=int(tm), second=0, microsecond=0)
        if ((now.day != ttime.day) or (now.day == ttime.day and now.minute > ttime.minute + 20)):
            c.execute("Insert into Served_Pizza select * from Pizza where Order_no=?", od)

        del od
        conn.commit()
        i += 1

    for row in c.execute('Select * from Served_Pizza'):
        t.insert(END, row)
        t.insert(END, "\n")
    c.execute('Delete from Served_Pizza where 1=1')
    conn.commit()
    served.mainloop()


def PendingOrder():
    pending = Tk()
    pen = []
    t = Text(pending, height=15, width=100)
    t.pack()
    s = "\n       PENDING ORDERS  \n"
    t.insert(END, s)
    t.insert(END, "\nID,  NAME,  ADDRESS, TYPE, MOBILE,  EMAIL,    TIME/DATE\n")
    for row in c.execute('Select * from Pizza'):
        pen.append(row)

    i = 0
    while i < len(pen):
        row = pen[i]
        ts = str(row)
        if (ts[-12] != " "):
            tt = ts[-12] + ts[-11]
        else:
            tt = ts[-11]
        o = str(ts[1:5])
        od = []
        od.append(o)
        if (ts[-12] != " "):
            if (ts[-18] != ":"):
                if (ts[-22] != ":"):
                    tm = ts[-22] + ts[-21]
                else:
                    tm = ts[-21]
            else:
                if (ts[-21] != ":"):
                    tm = ts[-22] + ts[-21]
                else:
                    tm = ts[-21]
            else:
                if (ts[-21] != ":"):
                    tm = ts[-21] + ts[-20]
                else:
                    tm = ts[-20]
        else:
            if (ts[-17] != ":"):
                if (ts[-21] != ":"):
                    tm = ts[-21] + ts[-20]
                else:
                    tm = ts[-20]
            else:
                if (ts[-20] != ":"):
                    tm = ts[-20] + ts[-19]
                else:
                    tm = ts[-19]
        now = datetime.now()
        ttime = now.replace(day=int(tt), hour=0, minute=int(tm), second=0, microsecond=0)
        if ttime.day == now.day and now.minute <= ttime.minute + 20:
            c.execute("Insert into Pending_Pizza select * from Pizza where Order_no=?", od)
        conn.commit()
        i += 1

    for row in c.execute('Select * from Pending_Pizza'):
        t.insert(END, row)
        t.insert(END, "\n")

    c.execute("Delete from Pending_Pizza where 1=1")
    conn.commit()
    pending.mainloop()


def Customer():
    cust_window = Tk()

    label_a = Label(cust_window, text="  ")
    label_a.grid(row=0)

    label_a = Label(cust_window, text="  ")
    label_a.grid(column=0)

    label_0 = Label(cust_window, text="Customer", fg="white", bg="black")
    label_0.grid(row=1, column=3)

    label_a = Label(cust_window, text="  ")
    label_a.grid(row=2)

    button_1 = Button(cust_window, text="Order Pizza", bg="light blue", relief="raised", height="5", width="10",
                      command=Order)
    button_1.grid(row=3, column=1)

    label_a = Label(cust_window, text="  ")
    label_a.grid(column=2)

    button_2 = Button(cust_window, text="Cancel Order", bg="light blue", relief="raised", height="5", width="10",
                      command=Cancel)
    button_2.grid(row=3, column=3)
```

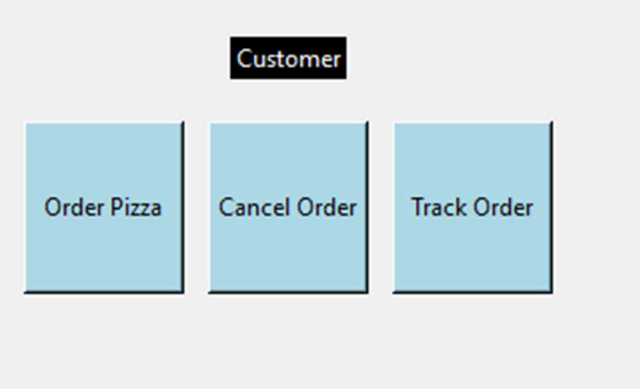
```
    button_2.grid(row=3, column=3)

    label_a = Label(cust_window, text="  ")
    label_a.grid(column=4)

    button_3 = Button(cust_window, text="Track Order", bg="light blue", relief="raised", height="5", width="10",
                      command=Track)
    button_3.grid(row=3, column=5)

    label_a = Label(cust_window, text="  ")
    label_a.grid(column=6)

    cust_window.mainloop()


def Vendor():
    ven_window = Tk()

    label_a = Label(ven_window, text=" ")
    label_a.grid(row=0)

    label_a = Label(ven_window, text=" ")
    label_a.grid(column=0)

    label_1 = Label(ven_window, text="Vendor", fg="white", bg="black")
    label_1.grid(row=1, column=2)

    label_a = Label(ven_window, text=" ")
    label_a.grid(row=2)

    button_4 = Button(ven_window, text="New Pizza Order", bg="light blue", relief="raised", height="6", width="20",
                      command=NewPizzaOrder)
    button_4.grid(row=3, column=1)

    button_5 = Button(ven_window, text="Canceled Order", bg="light blue", relief="raised", height="6", width="20",
                      command=CanceledOrder)
    button_5.grid(row=3, column=3)

    label_a = Label(ven_window, text=" ")
    label_a.grid(row=4)

    button_6 = Button(ven_window, text="Served Order", bg="light blue", relief="raised", height="6", width="20",
                      command=ServedOrder)
    button_6.grid(row=5, column=1)

    button_7 = Button(ven_window, text="Pending Order", bg="light blue", relief="raised", height="6", width="20",
                      command=PendingOrder)
    button_7.grid(row=5, column=3)

    label_a = Label(ven_window, text="  ")
    label_a.grid(column=4)

    ven_window.mainloop()


window = Tk()

label_a = Label(window, text="    ")
label_a.grid(column=0)

button_m1 = Button(window, text=" Customer ", bg="light green", height="4", width="15", relief="raised",
```

```
def Vendor():
    ven_window = Tk()

    label_a = Label(ven_window, text=" ")
    label_a.grid(row=0)

    label_a = Label(ven_window, text=" ")
    label_a.grid(column=0)

    label_1 = Label(ven_window, text="Vendor", fg="white", bg="black")
    label_1.grid(row=1, column=2)

    label_a = Label(ven_window, text=" ")
    label_a.grid(row=2)

    button_4 = Button(ven_window, text="New Pizza Order", bg="light blue", relief="raised", height="6", width="20",
                      command=NewPizzaOrder)
    button_4.grid(row=3, column=1)

    button_5 = Button(ven_window, text="Canceled Order", bg="light blue", relief="raised", height="6", width="20",
                      command=CanceledOrder)
    button_5.grid(row=3, column=3)

    label_a = Label(ven_window, text=" ")
    label_a.grid(row=4)

    button_6 = Button(ven_window, text="Served Order", bg="light blue", relief="raised", height="6", width="20",
                      command=ServedOrder)
    button_6.grid(row=5, column=1)

    button_7 = Button(ven_window, text="Pending Order", bg="light blue", relief="raised", height="6", width="20",
                      command=PendingOrder)
    button_7.grid(row=5, column=3)

    label_a = Label(ven_window, text="  ")
    label_a.grid(column=4)

    ven_window.mainloop()


window = Tk()

label_a = Label(window, text="    ")
label_a.grid(column=0)

button_m1 = Button(window, text=" Customer ", bg="light green", height="4", width="15", relief="raised",
                   command=Customer)
button_m1.grid(row=1, column=1)

label_a = Label(window, text="  ")
label_a.grid(column=2)

button_m2 = Button(window, text=" Vendor ", bg="light green", height="4", width="15", relief="raised", command=Vendor)
button_m2.grid(row=1, column=3)

label_a = Label(window, text="   ")
label_a.grid(column=4)

window.mainloop()
conn.close()
```

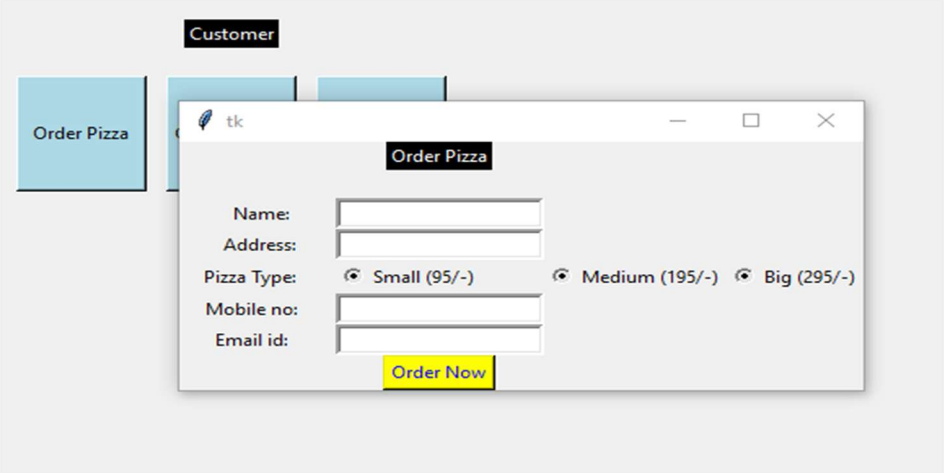
RESULT: -

HOME PAGE: -

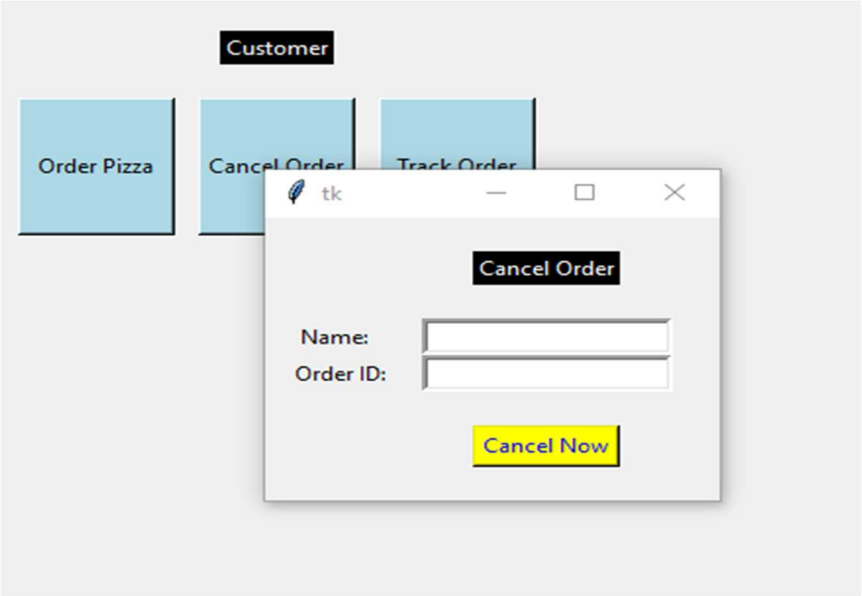
FOR CUSTOMER: -



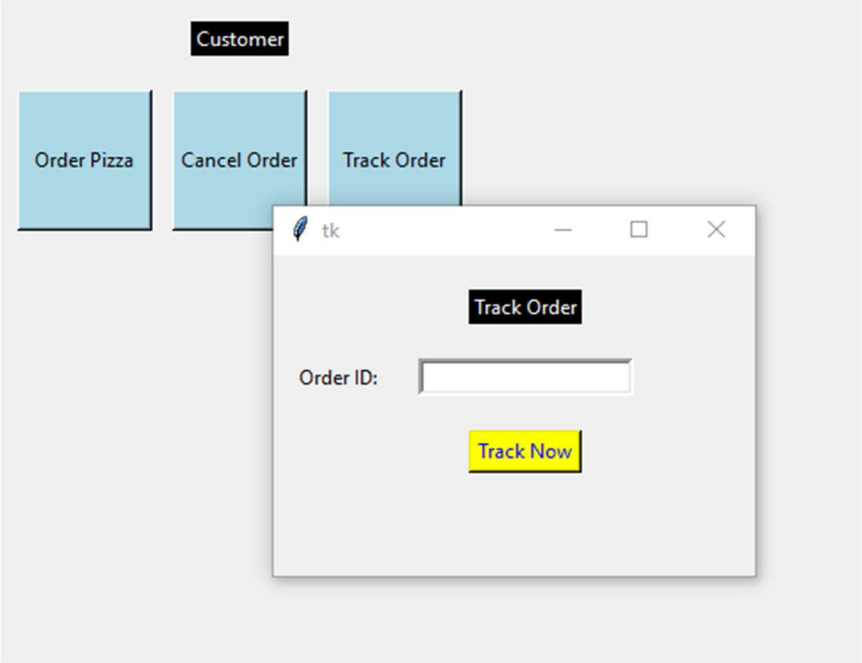
ORDER PIZZA: -



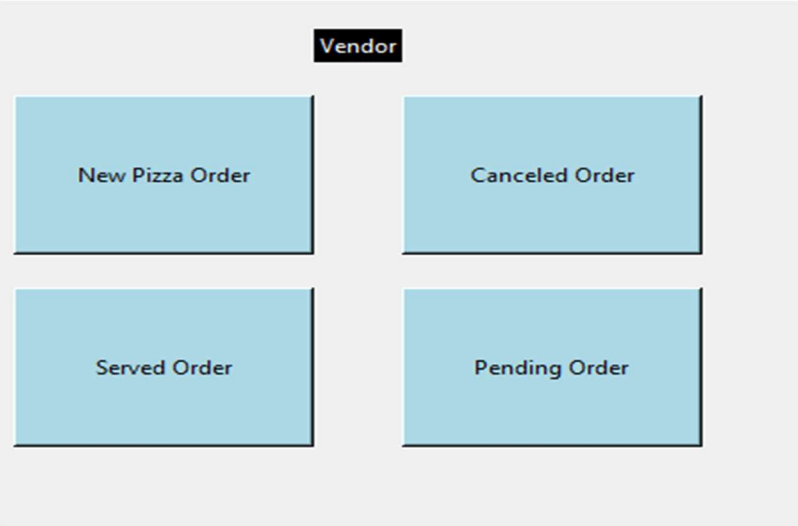
CANCEL ORDER: -



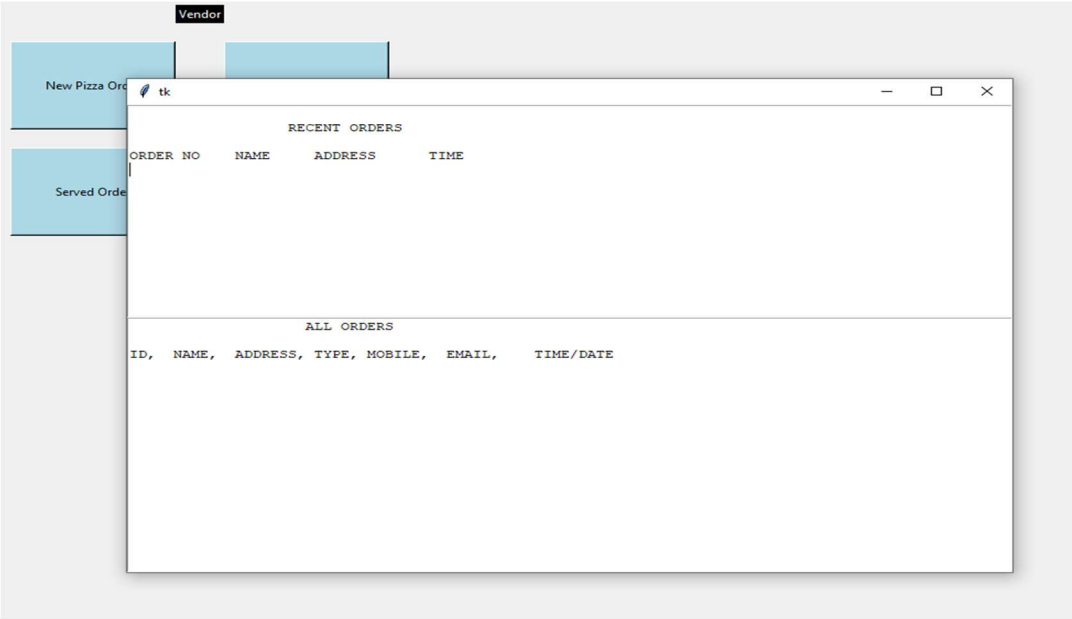
TRACK ORDER: -



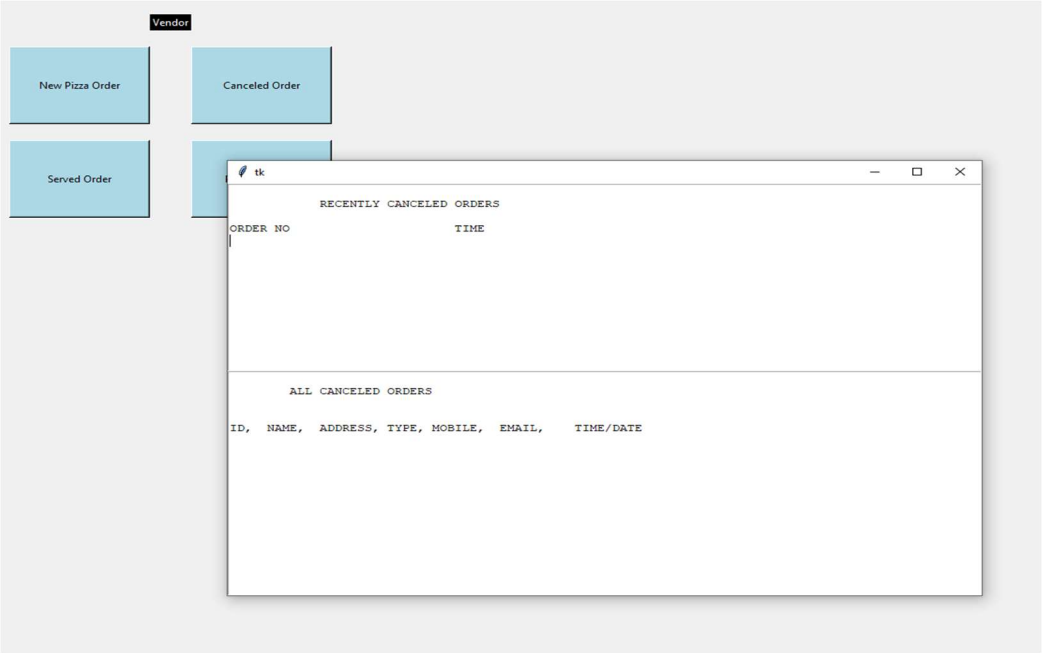
FOR VENDOR: -



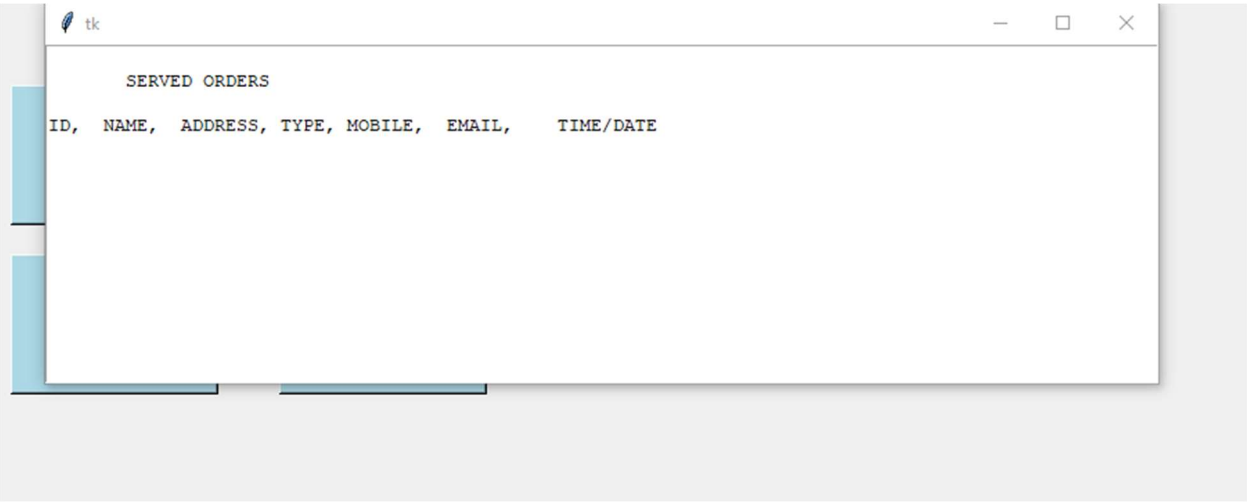
NEW PIZZA ORDER: -



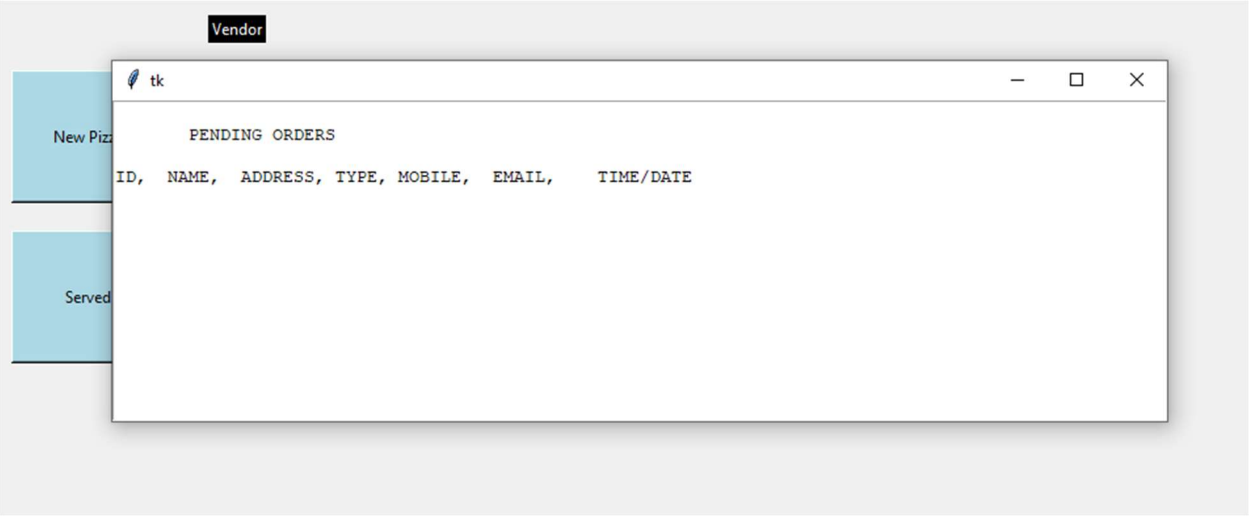
CANCEL ORDER: -



SERVED ORDER: -



PENDING ORDERS: -



CONCLUSION: -

It is our team's hope that this document will be of huge help with understanding of our little project as we have used a different approach which has proved beneficial for us. This project will help to reduce the manual work.

REFERENCES: -

www.geeksforgeeks.com
www.w3schools.com
www.tutorialspoint.com
www.youtube.com