CERTIFICATE

This is to certify that the project entitled "**ADMIRABLE DINNING WITH CONTACT-FREE ORDERING SYSTEM**" being submitted by **K. BHAVYA SWARUPA REDDY (20H51A6615) , M. UDAY (20H51A04N7), V. THARUNI (20H51A6224), V.APOORVA (20H51A6623).**

ABSTRACT

This project is titled as “ADMIRABLE DINNING WITH CONTACT-FREE ORDERING”. In view of the COVID-19situation where human contact is rendered to be as contagious. The restaurant owners are worried sick that they would face a huge loss. Hence the main idea of the ADMIRABLE DINNING WITH CONTACT-FREE ORDERING”. is to reduce the involvement of human touch as much as possible in the process of ordering food in a restaurant. Using the device, you can order the food from the menu The order of food given by user is sent directly into the kitchen where the chefs work their magic. The total bill amount is also visible to the customer on ADMIRABLE DINNING WITH CONTACT-FREE ORDERING”. so the customer can pay the bill directly.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

TABLE OF CONTENTS

I. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled as “ADMIRABLE DINNING WITH CONTACT-FREE ORDERING”. This software provides facility to order food in a restaurant with minimal human contact with the food or with the customer. This project uses TKINTER for providing GUI interface. In that tool we have access to various python packages that help in sending order to the kitchen and sending bill copies to both manager of the restaurant and the customer.

1.2 PROJECT PURPOSE

In view of the COVID-19 situation where human contact is rendered to be as contagious. The restaurant owners are worried sick that they would face a huge loss. Hence the main idea of the ADMIRABLE DINNING WITH CONTACT-FREE ORDERING. is to reduce the involvement of human touch as much as possible in the process of ordering food in arestaurant.

1.3 PROJECT FEATURES

The main feature of this project is that it provides a COVID safe environment for the customers who come to a restaurant by providing the food with minimal human interaction as possible. This stimulates a positive feeling in the customers minds and may let them forget about the COVID situation for a while. Another feature that can be added to a restaurant along with this project is a conveyor belt for sending the food from kitchen to customer table.

II. SYSTEM ANALYSIS

2. SYSTEMANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. TheSystem is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, "what must be done to solve the problem?" The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

A detailed study of the process must be made by various techniques like calculator frame, file dialog, message box etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal isreviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

2.2 EXISTING SYSTEM

The existing system refers to the normal working of a restaurant where there are different groups of workers doing their specified job for smooth running of the restaurant and one of the key roles in a restaurant is that of the waiters, who wait at the table, take order from customer, deliver the order to chef and bring the food to the table.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- More Human interaction

- Time consuming.
- Some staff may lack in skills to remember all the items in menu along with their price and quantity and confirm the bill payment

To avoid all these limitations and make the working more accurately the system needs to be implemented efficiently.

2.3 PROPOSED SYSTEM

The aim of proposed system is to develop a system of improved facilities. Theproposed system can overcome all the limitations of the existing system. The system provides Less human interaction and reduce the time taken to deliver the order to the kitchen and the food to the table. The existing system has several disadvantages and many more difficulties towork well. The proposed system tries to eliminate or reduce these difficulties up to some extent. The proposed system helps the owner of restaurant to work user friendly and he can easily do his jobs without time lagging.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features

- Ensure minimal human interaction.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor : Intel Dual Core@ CPU 2.90GHz.
- Hard disk : 1GB and above.
- RAM : 2GB and above.
- Monitor : 5 inches or above.

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows XP and above
- Languages : Python 3.0.7 and above
- Backend : MySQL 8.0.27 and above
- IDE : TKINTER 3.0.7 and above

III. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCITECTURE

This project architecture shows the procedure followed for contactless ordering using python, starting from customer login to final bill generation.

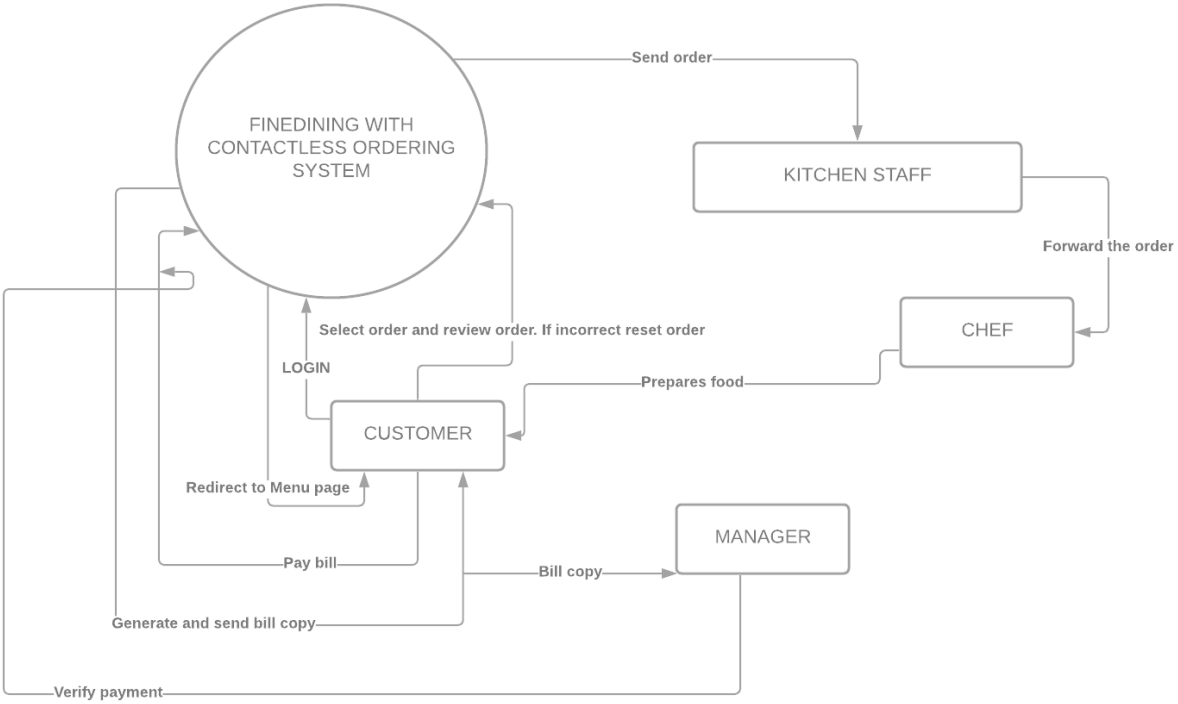


Figure 3.1: Project Architecture of ADMIRABLE DINNING WITH CONTACT-FREE ORDERING.

3.2 DESCRIPTION

First the customer will login in to the system using an OTP sent to the customers registered mobile number. Then the customer is redirected to menu frame where customer can select order and send the order to kitchen staff who will deliver the order to chef. The chef will prepare the food. Then the food is delivered in two ways to the customer's table. One is self-service and the other is a conveyor belt which carries the food to the table. A bill is generated in the system and appropriate discount is applied if the customer is a repeating customer. The customer pays the bill via a UPI mode which can be verified the manager. The Bill copies are also forwarded top both the customers mobile number as well as the manager

3.3 USE CASE DIAGRAM

In the use case diagram, we have basically three actors who are the customer, the kitchen staff and the manager. The user has the rights to login, access to menu andto receive bill copy in SMS. Whereas the manager has to monitor the inventory, access to record of bills and payments and the kitchen staff is responsible to monitor and deliver the orders to chef and also monitor and manage the inventory and food availability.

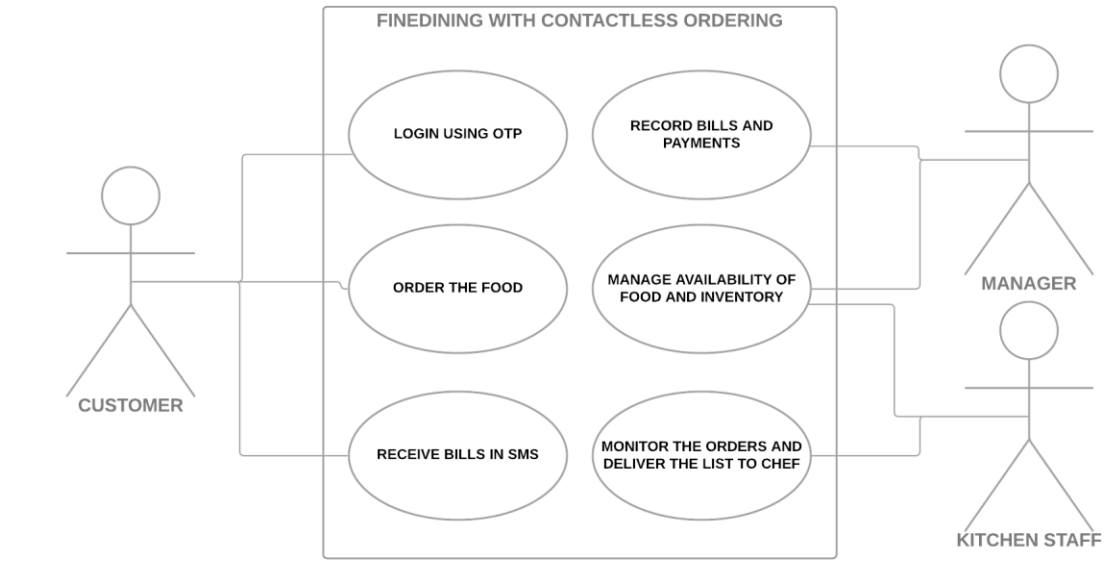


Figure 3.2: Use Case Diagram for ADMIRABLE DINNING WITH CONTACT-FREE ORDERING.

3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

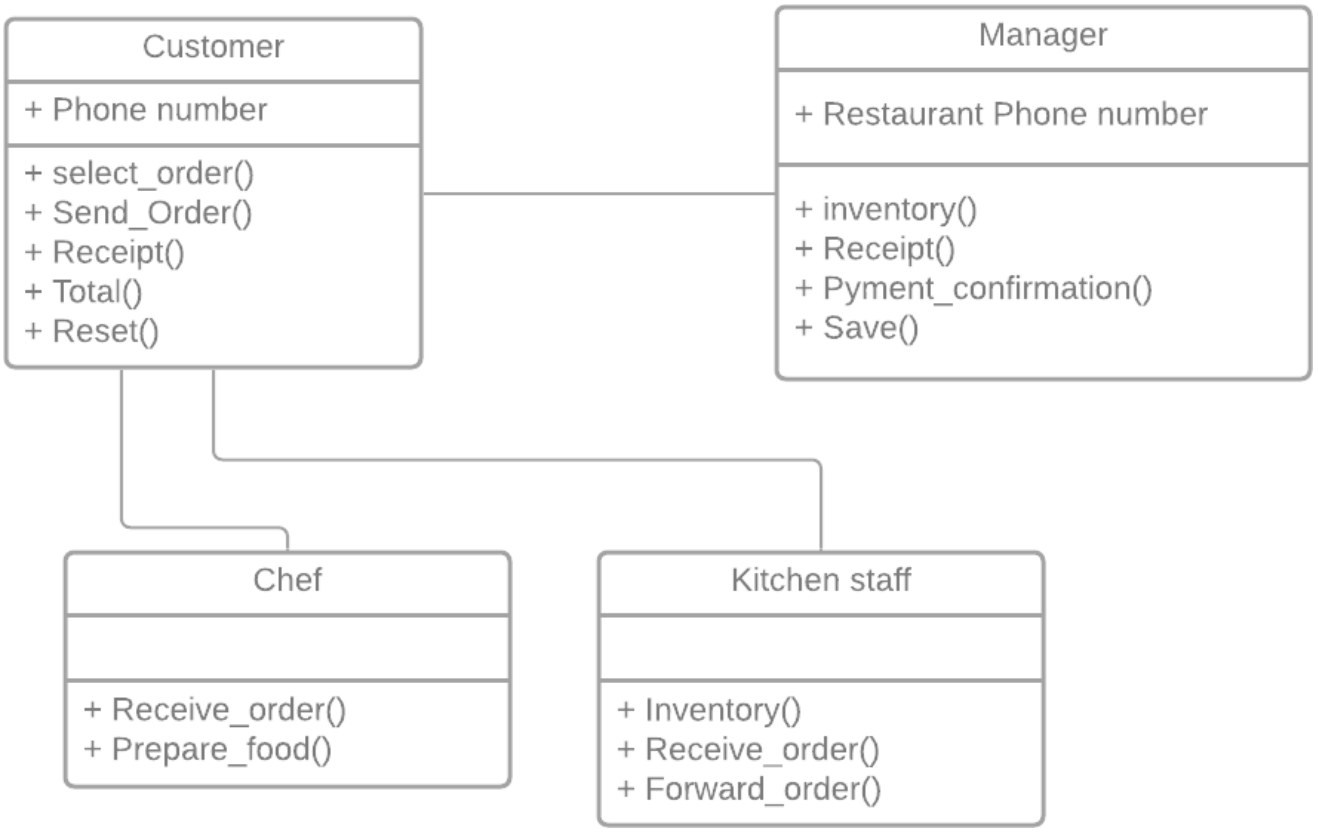


Figure 3.3: Class Diagram for ADMIRABLE. DINNING WITH CONTACT-FREE ORDERING.

3.5 SEQUENCE DIAGRAM

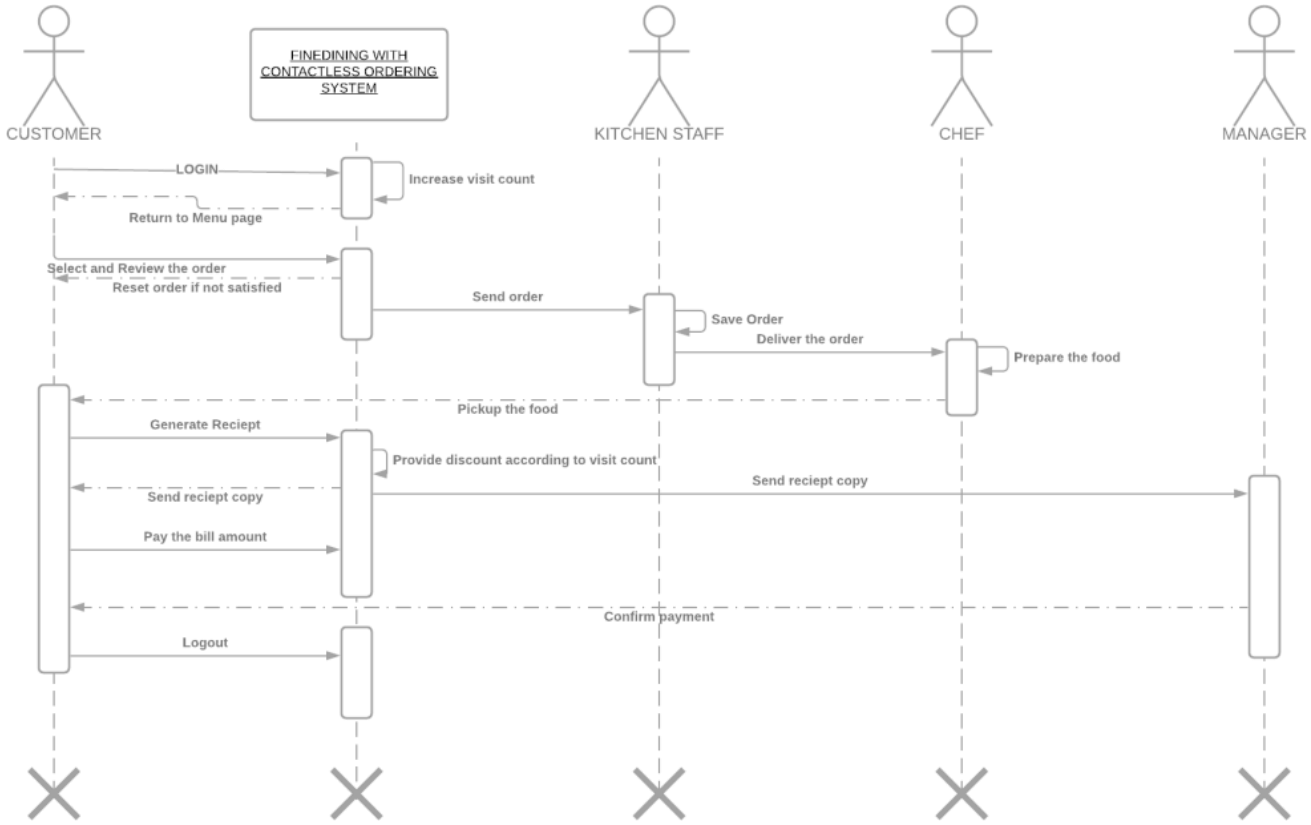


Figure 3.4: Sequence Diagram for ADMIRABLE DINNING WITH CONTACT-FREE ORDERING.

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

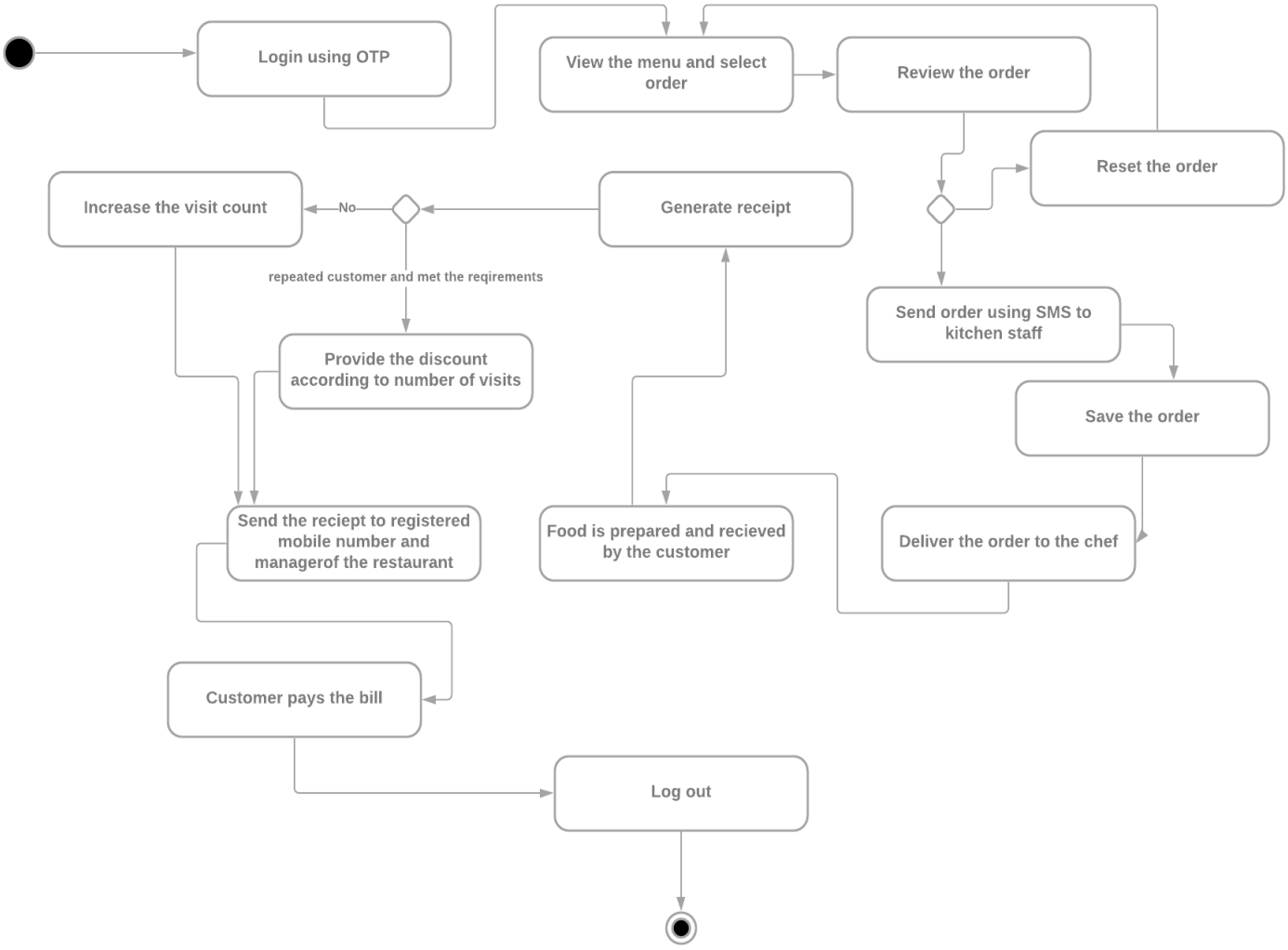


Figure 3.5: Activity Diagram for ADMIRABLE DINNING WITH CONTACT-FREE ORDERING.

IV. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

Registration page:

```
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import pymysql

def login_window():
    root.destroy()

def clear():
    entryemail.delete(0, END)
    entrycontact.delete(0, END)
    entrypassword.delete(0, END)
    entryconfirmpassword.delete(0, END)
    entryfirstname.delete(0, END)
    entrylastname.delete(0, END)
    entryanswer.delete(0, END)
    comboquestion.current(0)
    check.set(0)

def register():
    if entryfirstname.get() == '' or entrylastname.get() == '' or entryemail.get() == '' or
entrycontact.get() == '' or \
            entrypassword.get() == '' or entryconfirmpassword.get() == '' or comboquestion.get() ==
'Select' or entryanswer.get() == '':
        showerror('Error', "All Fields Are Required", parent=root)


    elif entrypassword.get() != entryconfirmpassword.get():
        showerror('Error', "Password Mismatch", parent=root)


    elif check.get() == 0:
        showerror('Error', "Please Agree To Our Terms & Conditions", parent=root)


    else:
        try:
            con = pymysql.connect(host='localhost', user='root', password='1305', database='register')
            cur = con.cursor()
            cur.execute('select * from student where email=%s', entryemail.get())
            row = cur.fetchone()
            if row != None:
```

```
            showerror('Error', "User Already Exists", parent=root)
        else:


            cur.execute(
                'insert into student (f_name,l_name,email,contact,question,answer,password)
values(%s,%s,%s,%s,%s,%s,%s)',
                (entryfirstname.get(), entrylastname.get(), entryemail.get(), entrycontact.get(),
                 comboquestion.get(),
                 entryanswer.get(), entrypassword.get()))
            con.commit()
            con.close()
            showinfo('Success', "Registration Successful", parent=root)
            clear()
            root.destroy()
            import login

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)

root = Tk()
root.geometry('1350x710+0+10')
root.title('Registration Form')


bg = PhotoImage(file='bg.png')
bgLabel = Label(root, image=bg)
bgLabel.place(x=0, y=0)


registerFrame = Frame(root, bg='white', width=650, height=650)
registerFrame.place(x=630, y=30)


titleLabel = Label(registerFrame, text='Registration Form', font=('arial', 22, 'bold '), bg='white',
                   fg='deep pink', )
titleLabel.place(x=200, y=5)


firstnameLabel = Label(registerFrame, text='First Name', font=('times new roman', 18, 'bold'),
bg='white',
                       fg='gray20', )
firstnameLabel.place(x=20, y=80)
entryfirstname = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entryfirstname.place(x=20, y=115, width=250)


lastnameLabel = Label(registerFrame, text='Last Name', font=('times new roman', 18, 'bold'),
bg='white',
                      fg='gray20', )
```

```
lastnameLabel.place(x=370, y=80)
entrylastname = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entrylastname.place(x=370, y=115, width=250)


contactLabel = Label(registerFrame, text='Contact Number', font=('times new roman', 18, 'bold'),
bg='white',
                     fg='gray20', )
contactLabel.place(x=20, y=200)
entrycontact = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entrycontact.place(x=20, y=235, width=250)


emailLabel = Label(registerFrame, text='Email', font=('times new roman', 18, 'bold'), bg='white',
fg='gray20', )
emailLabel.place(x=370, y=200)
entryemail = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entryemail.place(x=370, y=235, width=250)


questionLabel = Label(registerFrame, text='Security Question', font=('times new roman', 18,
'bold'), bg='white',
                      fg='gray20', )
questionLabel.place(x=20, y=320)
comboquestion = ttk.Combobox(registerFrame, font=('times new roman', 16), state='readonly',
justify=CENTER)
comboquestion['values'] = ('Select', 'Your First Pet Name?', 'Your Birth Place Name?', 'Your Best
Friend Name?',
                           'Your Favourite Teacher?', 'Your Favourite Hobby?')
comboquestion.place(x=20, y=355, width=250)
comboquestion.current(0)
answerLabel = Label(registerFrame, text='Answer', font=('times new roman', 18, 'bold'),
bg='white',
                    fg='gray20', )
answerLabel.place(x=370, y=320)
entryanswer = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entryanswer.place(x=370, y=355, width=250)


passwordLabel = Label(registerFrame, text='Password', font=('times new roman', 18, 'bold'),
bg='white',
                      fg='gray20', )
passwordLabel.place(x=20, y=440)
entrypassword = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entrypassword.place(x=20, y=475, width=250)


confirmpasswordLabel = Label(registerFrame, text='Confirm Password', font=('times new roman',
18, 'bold'),
                             bg='white',
```

```
                        fg='gray20', )
confirmpasswordLabel.place(x=370, y=440)
entryconfirmpassword = Entry(registerFrame, font=('times new roman', 18), bg='lightgray')
entryconfirmpassword.place(x=370, y=475, width=250)


check = IntVar()
checkButton = Checkbutton(registerFrame, text='I Agree All The Terms & Conditions',
variable=check, onvalue=1,
                      offvalue=0, font=('times new roman', 14, 'bold'), bg='white')
checkButton.place(x=20, y=530)


button = PhotoImage(file='button.png')
registerbutton = Button(registerFrame, image=button, bd=0, cursor='hand2', bg='white',
activebackground='white'
                    , activeforeground='white', command=register)
registerbutton.place(x=250, y=580)


loginimage = PhotoImage(file='login.png')
loginbutton1 = Button(root, image=loginimage, bd=0, cursor='hand2', bg='gold',
activebackground='gold',
                    activeforeground='gold')
loginbutton1.place(x=240, y=330)


root.mainloop()
```

LOGIN PAGE AND RESET PASSWORD PAGE:

```
from tkinter import *
from tkinter import messagebox
import pymysql
from tkinter import ttk
###################finctions


def reset_password():
    if mailentry.get()=='':
        messagebox.showerror('Error','please enter the email adress to reset your password')
    else:
        con=pymysql.connect(host='localhost',user='root',password='1305',database='register')
        cur=con.cursor()
        cur.execute('select * from student where email=%s',mailentry.get())
        row=cur.fetchone()
        if row==None:
            messagebox.showerror('Error','Please enter the valid email address')
        else:
            con.close()
            def change_password():


                if securityquesCombo.get()=='select' or answerEntry.get()==''or newPassEntry.get()=='':
                    messagebox.showerror('Error','All fields are reequired')
                else:
                    con=pymysql.connect(host='localhost',user='root',password='1305',database='register')
                    cur=con.cursor()
                    cur.execute('select * from student where email=%s and question=%s and
answer=%s',(mailentry.get(),securityquesCombo.get(),answerEntry.get()))
                    row=cur.fetchone()
                    if row==None:
                        messagebox.showerror('Error','security question or answer is incorrect',parent=root2)
                    else:
                        cur.execute('update student set password=%s where email=%s',(newPassEntry.get(),mailentry.get()))


                        con.commit()
                        con.close()
                        messagebox.showinfo('success','Passwrod is reset, please login with new password',parent=root2)
                        securityquesCombo.current(0)
                        answerEntry.delete(0,END)
                        newPassEntry.delete(0,END)
                        root2.destroy()


            root2=Toplevel()
            root2.title('Forgot Password')
            root2.geometry('470x560+400+60')
            root2.config(bg='white')
            root2.focus_force()
            root2.grab_set()
            forgetpassLabel=Label(root2,text='Forgot Password',font=('times new roman',22,'bold'),bg='white',fg='green')
            forgetpassLabel.place(x=128,y=10)
            securityqueslabel=Label(root2,text='Security Question',font=('times new roman ',19,'bold'),bg='white')
            securityqueslabel.place(x=68,y=220)


            securityquesCombo=ttk.Combobox(root2,font=('times new roman ',19),state='readonly',width=22)
            securityquesCombo['values']=('select','your first pet name?','your birth place name?','Your Best Friend Name?','Your
favourite Teacher?','your Favorite Hobby?')
```

```
    securityquesCombo.place(x=60,y=260)
    securityquesCombo.current(0)


    answerLabel=Label(root2,text='Answer',font=('times new roman',19,'bold'),bg='white')
    answerLabel.place(x=60,y=310)
    answerEntry=Entry(root2,font=('times new roman',19),bg='white',width=18)
    answerEntry.place(x=60,y=350)


    newPassLabel = Label(root2, text='NEW PASSWORD', font=('times new roman', 19, 'bold'), bg='white')
    newPassLabel.place(x=60, y=400)
    newPassEntry = Entry(root2, font=('times new roman', 19), bg='white', width=18)
    newPassEntry.place(x=60, y=440)


    changepassButton=Button(root2,text='change
password',font=('arial',17,'bold'),bg='green',fg='white',cursor='hand2',activebackground='green',activeforeground='white',co
mmand=change_password)
    changepassButton.place(x=130,y=500)









    root2.mainloop()
```

MENU PAGE, CALCULATOR, SEND, SAVE, TOTAL, RECIEPT, RESET:

```
from tkinter import *


from tkinter import filedialog,messagebox
import random
import time
import requests

#functions


def reset():
    textReciept.delete(1.0,END)
    e_Butter_Chicken.set('0')
    e_Chicken_Kolapuri.set('0')
    e_Telangana_Chicken.set('0')
    e_Mutton_Kheema.set('0')
    e_Mutton_Curry.set('0')
    e_Fish_Pulusu.set(0)
    e_Veg_Kofta.set('0')
    e_Butter_Panner.set('0')
    e_Mushroom.set('0')
    e_Chicken_Dum_Biryani.set('0')
    e_Chicken_Frypiece_Biryani.set('0')
    e_Chicken_Biryani_spl.set('0')
    e_Mutton_Nallighosh_Biryani.set('0')
    e_Mutton_Juicy_Biryani.set("0")
    e_Fish_Biryani.set('0')
    e_Prawn_Biryani.set('0')
    e_Veg_Pulao.set('0')
    e_Panner_Biryani.set('0')
    e_Chicken_Mangolia.set('0')
    e_Chicken_Manchuria.set('0')
    e_Chicken_Drumsticks.set('0')
    e_Chicken_kabab.set('0')
    e_Egg_Manchuria.set('0')
    e_Veg_Manchuria.set('0')
    e_Fried_Mushroom.set('0')
    e_BabyCorn.set('0')
    e_Panner_Tikka.set('0')
    textButter_Chicken.config(state=DISABLED)
    textChicken_Kolapuri.config(state=DISABLED)
    textTelangana_Chicken.config(state=DISABLED)
    textMutton_Kheema.config(state=DISABLED)
    textMutton_Curry.config(state=DISABLED)
    textFish_Pulusu.config(state=DISABLED)
    textVeg_Kofta.config(state=DISABLED)
    textButter_Panner.config(state=DISABLED)
    textMushroom.config(state=DISABLED)
    textChicken_Dum_Biryani.config(state=DISABLED)
    textChicken_Frypiece_Biryani.config(state=DISABLED)
    textChicken_Biryani_Spl.config(state=DISABLED)
    textMutton_Nallighosh_Biryani.config(state=DISABLED)
    textMutton_Juicy_Biryani.config(state=DISABLED)
    textFish_Biryani.config(state=DISABLED)
    textPrawn_Biryani.config(state=DISABLED)
    textVeg_Pulao.config(state=DISABLED)
    textPanner_Biryani.config(state=DISABLED)
```

```
    textChicken_Mangolia.config(state=DISABLED)
    textChicken_Manchuria.config(state=DISABLED)
    textChicken_Drumsticks.config(state=DISABLED)
    textChicken_kabab.config(state=DISABLED)
    textEgg_Manchuria.config(state=DISABLED)
    textVeg_Manchuria.config(state=DISABLED)
    textFried_Mushroom.config(state=DISABLED)
    textBabyCorn.config(state=DISABLED)
    textPanner_Tikka.config(state=DISABLED)


    var1.set(0)
    var2.set(0)
    var3.set(0)
    var4.set(0)
    var5.set(0)
    var6.set(0)
    var7.set(0)
    var8.set(0)
    var9.set(0)
    var10.set(0)
    var11.set(0)
    var12.set(0)
    var13.set(0)
    var14.set(0)
    var15.set(0)
    var16.set(0)
    var17.set(0)
    var18.set(0)
    var19.set(0)
    var20.set(0)
    var21.set(0)
    var22.set(0)
    var23.set(0)
    var24.set(0)
    var25.set(0)
    var26.set(0)
    var27.set(0)


    costofMaincoursevar.set('')
    costofBiryanivar.set('')
    costofStartersvar.set('')
    costofSubtotalvar.set('')
    costofServicetaxvar.set('')
    costofTotalcostvar.set('')




def send():
    def send_msg():
        message=textarea.get(1.0,END)
        number=numberfield.get()
        auth='YhKOpRoCA1MEdIcexnQUJb0GNT7XrByH9jagDvwV45ikL3zl8Wl5FOApSawH9kxcojL84KPzifJum0qB'
        url='https://www.fast2sms.com/dev/bulk'
```

```
        params={
            'authorization':auth,
            'message':message,
            'numbers':number,
            'sender-id':'CHKSMS',
            "route":'p',
            'language':'english'


        }
        response=requests.get(url,params=params)
        dic=response.json()
        result=dic.get('return')
        if result==True:
            messagebox.showinfo('send successfully','message sent successfully')


        else:
            messagebox.showerror('error','something went wrong')


    root2=Toplevel()
    root2.title("Send Bill")
    root2.geometry('485x485+50+50')




    numberLabel=Label(root2,text='Mobile Number',font=("arial",18,'bold'))
    numberLabel.pack()
    numberfield=Entry(root2,font=('helvetica',22,'bold'),bd=3,width=24)
    numberfield.pack()
    billLabel=Label(root2,text="Bill Details",font=("arial",18,'bold underline'))
    billLabel.pack()




    textarea=Text(root2,font=("arial",12,'bold'),width=42,height=14)
    textarea.pack()
    textarea.insert(END,'Receipt Ref:\t\t'+billnumber+'\t\t'+date+'\n\n')
    if costofMaincoursevar.get()!=0:
        textarea.insert(END, f'cost of Main course\t\t\t{priceofMaincourse}\n')
    if costofBiryanivar.get()!=0:
        textarea.insert(END, f'cost of Biryani\t\t\t{priceofBiryani}\n')
    if costofStartersvar.get()!=0:
        textarea.insert(END, f'cost of Starters\t\t\t{priceofStarters}\n')
    textarea.insert(END, f'Sub Total\t\t\t{subtotalofItems}\n')
    textarea.insert(END, f'Service Tax\t\t\t{50}\n')
    textarea.insert(END, f'Total Cost\t\t\t{subtotalofItems+50}\n')
    sendButton = Button(root2, text='SEND', font=("arial", 19, 'bold'), command=send_msg)
    sendButton.pack()


    root2.mainloop()


def save():
    url=filedialog.asksaveasfile(mode="w", defaultextension='.txt')
```

```
    bill_data=textReciept.get(1.0,END)
    url.write(bill_data)
    url.close()




    messagebox.showinfo('Information','Your Bill is Save')


def receipt():
    global billnumber, date
    textReciept.delete(1.0,END)
    x=random.randint(0,1000)
    billnumber="BILL"+str(x)
    date=time.strftime("%d/%m/%Y")
    textReciept.insert(END,"Receipt ref:\t\t"+billnumber+"\t\t"+date+"\n')
    textReciept.insert(END,'**********************\n')
    textReciept.insert(END,'Items:\t\tCost of Items(Rs)\n')
    textReciept.insert(END, '**********************\n')
    if e_Butter_Chicken.get()!='0':
        textReciept.insert(END,f'Butter Chicken\t\t\t{int(e_Butter_Chicken.get())*349}\n\n')
    if e_Chicken_Kolapuri.get()!='0':
        textReciept.insert(END,f'Chicken Kolapuri\t\t\t{int(e_Chicken_Kolapuri.get())*369}\n\n')
    if e_Telangana_Chicken.get()!='0':
        textReciept.insert(END,f'Telangana Chicken Chicken\t\t\t{int(e_Telangana_Chicken.get())*369}\n\n')
    if e_Mutton_Kheema.get()!='0':
        textReciept.insert(END,f'Mutton Kheema\t\t\t{int(e_Mutton_Kheema.get())*449}\n\n')
    if e_Mutton_Curry.get()!='0':
        textReciept.insert(END,f'Mutton curry \t\t\t{int(e_Mutton_Curry.get())*424}\n\n')
    if e_Fish_Pulusu.get()!='0':
        textReciept.insert(END,f'Fish Pulusu\t\t\t{int(e_Fish_Pulusu.get())*489}\n\n')
    if e_Veg_Kofta.get()!='0':
        textReciept.insert(END,f'Veg Kofta\t\t\t{int(e_Veg_Kofta.get())*299}\n\n')
    if e_Butter_Panner.get()!='0':
        textReciept.insert(END,f'Butter Panner\t\t\t{int(e_Butter_Panner.get())*199}\n\n')
    if e_Mushroom.get()!='0':
        textReciept.insert(END,f'Mushroom\t\t\t{int(e_Mushroom.get())*169}\n\n')
    if e_Chicken_Dum_Biryani.get()!='0':
        textReciept.insert(END,f'Chicken Dum Biryani\t\t\t{int(e_Chicken_Dum_Biryani.get())*380}\n\n')
    if e_Chicken_Frypiece_Biryani.get()!='0':
        textReciept.insert(END,f'Chicken Frypiece Biryani\t\t\t{int(e_Chicken_Frypiece_Biryani.get())*399}\n\n')
    if e_Chicken_Biryani_spl.get()!='0':
        textReciept.insert(END,f'Chicken Biryani Spl\t\t\t{int(e_Chicken_Biryani_spl.get())*399}\n\n')
    if e_Mutton_Nallighosh_Biryani.get()!='0':
        textReciept.insert(END,f'Mutton Nallighosh Biryani\t\t\t{int(e_Mutton_Nallighosh_Biryani.get())*449}\n\n')
    if e_Mutton_Juicy_Biryani.get()!='0':
        textReciept.insert(END,f'Mutton Juicy Biryani\t\t\t{int(e_Mutton_Juicy_Biryani.get())*469}\n\n')
    if e_Fish_Biryani.get()!='0':
        textReciept.insert(END,f'Fish Biryani\t\t\t{int(e_Fish_Biryani.get())*510}\n\n')
    if e_Prawn_Biryani.get()!='0':
        textReciept.insert(END,f'Prawn Biryani\t\t\t{int(e_Prawn_Biryani.get())*470}\n\n')
    if e_Veg_Pulao.get()!='0':
        textReciept.insert(END,f'Veg Pulao\t\t\t{int(e_Veg_Pulao.get())*220}\n\n')
    if e_Panner_Biryani.get()!='0':
        textReciept.insert(END,f'Panner Biryani\t\t\t{int(e_Panner_Biryani.get())*250}\n\n')
    if e_Chicken_Mangolia.get()!='0':
```

```
        textReciept.insert(END,f'Chicken Mangolia\t\t\t{int(e_Chicken_Mangolia.get())*210}\n\n')
    if e_Chicken_Manchuria.get()!='0':
        textReciept.insert(END,f'Chicken Manchuria\t\t\t{int(e_Chicken_Manchuria.get())*189}\n\n')
    if e_Chicken_Drumsticks.get()!='0':
        textReciept.insert(END,f'Chicken Drumsticks\t\t\t{int(e_Chicken_Drumsticks.get())*210}\n\n')
    if e_Chicken_kabab.get()!='0':
        textReciept.insert(END,f'Chicken Kabab\t\t\t{int(e_Chicken_kabab.get())*240}\n\n')
    if e_Egg_Manchuria.get()!='0':
        textReciept.insert(END,f'Egg Manchuria\t\t\t{int(e_Egg_Manchuria.get())*199}\n\n')
    if e_Veg_Manchuria.get()!='0':
        textReciept.insert(END,f'Veg Manchuria\t\t\t{int(e_Veg_Manchuria.get())*170}\n\n')
    if e_Fried_Mushroom.get()!='0':
        textReciept.insert(END,f'Fried Mushroom\t\t\t{int(e_Fried_Mushroom.get())*170}\n\n')
    if e_BabyCorn.get()!='0':
        textReciept.insert(END,f'Baby Corn\t\t\t{int(e_BabyCorn.get())*190}\n\n')
    if e_Panner_Tikka.get()!='0':
        textReciept.insert(END,f'Panner Tikka\t\t\t{int(e_Panner_Tikka.get())*199}\n\n')
    textReciept.insert(END, '**********************\n')
    if costofMaincoursevar.get()!='0 Rs':
        textReciept.insert(END, f'cost of Main course\t\t\t{priceofMaincourse}\n\n')
    if costofBiryanivar.get()!='0 Rs':
        textReciept.insert(END, f'cost of Biryani\t\t\t{priceofBiryani}\n\n')
    if costofStartersvar.get()!='0 Rs':
        textReciept.insert(END, f'cost of Starters\t\t\t{priceofStarters}\n\n')
    textReciept.insert(END, f'Sub Total\t\t\t{subtotalofItems}\n\n')
    textReciept.insert(END, f'Service Tax\t\t\t{50}\n\n')
    textReciept.insert(END, f'Total Cost\t\t\t{subtotalofItems+50}\n\n')
    textReciept.insert(END, '**********************\n')


def totalcost():
    global priceofMaincourse , priceofBiryani , priceofStarters , subtotalofItems


    item1=int(e_Butter_Chicken.get())
    item2=int(e_Chicken_Kolapuri.get())
    item3=int(e_Telangana_Chicken.get())
    item4=int(e_Mutton_Kheema.get())
    item5=int(e_Mutton_Curry.get())
    item6=int(e_Fish_Pulusu.get())
    item7=int(e_Veg_Kofta.get())
    item8=int(e_Butter_Panner.get())
    item9=int(e_Mushroom.get())


    item10=int(e_Chicken_Dum_Biryani.get())
    item11=int(e_Chicken_Frypiece_Biryani.get())
    item12=int(e_Chicken_Biryani_spl.get())
    item13=int(e_Mutton_Nallighosh_Biryani.get())
    item14=int(e_Mutton_Juicy_Biryani.get())
    item15=int(e_Fish_Biryani.get())
    item16=int(e_Prawn_Biryani.get())
    item17=int(e_Veg_Pulao.get())
    item18=int(e_Panner_Biryani.get())


    item19=int(e_Chicken_Mangolia.get())
    item20=int(e_Chicken_Manchuria.get())
    item21=int(e_Chicken_Drumsticks.get())
    item22=int(e_Chicken_kabab.get())
```

```
    item23=int(e_Egg_Manchuria.get())
    item24=int(e_Veg_Manchuria.get())
    item25=int(e_Fried_Mushroom.get())
    item26=int(e_BabyCorn.get())
    item27=int(e_Panner_Tikka.get())



priceofMaincourse=(item1*349)+(item2*369)+(item3*369)+(item4*449)+(item5*424)+(item6*489)+(item7*299)+(item8*
199)+(item9*169)

priceofBiryani=(item10*350)+(item11*399)+(item12*399)+(item13*449)+(item14*469)+(item15*510)+(item16*470)+(ite
m17*220)+(item18*250)

priceofStarters=(item19*210)+(item20*189)+(item21*210)+(item22*240)+(item23*199)+(item24*170)+(item25*170)+(ite
m26*190)+(item27*199)
    costofMaincoursevar.set(str(priceofMaincourse)+'Rs')
    costofBiryanivar.set(str(priceofBiryani)+"Rs")
    costofStartersvar.set(str(priceofStarters)+"Rs")


    subtotalofItems=priceofStarters+priceofMaincourse+priceofBiryani
    costofSubtotalvar.set(str(subtotalofItems)+ "Rs")


    costofServicetaxvar.set('50 Rs')


    tottalcost=subtotalofItems+50
    costofTotalcostvar.set(str(tottalcost)+"Rs")




def Butter_Chicken():
    if var1.get()==1:
        textButter_Chicken.config(state=NORMAL)
        textButter_Chicken.delete(0,END)
        textButter_Chicken.focus()
    else:
        textButter_Chicken.config(state=DISABLED)
        e_Butter_Chicken.set("0")


def Chicken_Kolapuri():
    if var2.get()==1:
        textChicken_Kolapuri.config(state=NORMAL)
        textChicken_Kolapuri.delete(0,END)
        textChicken_Kolapuri.focus()
    else:
        textChicken_Kolapuri.config(state=DISABLED)
        e_Chicken_Kolapuri.set("0")
```

```
def Telangana_Chicken():
    if var3.get()==1:
        textTelangana_Chicken.config(state=NORMAL)
        textTelangana_Chicken.delete(0,END)
        textTelangana_Chicken.focus()
    else:
        textTelangana_Chicken.config(state=DISABLED)
        e_Telangana_Chicken.set("0")


def Mutton_Kheema():
    if var4.get()==1:
        textMutton_Kheema.config(state=NORMAL)
        textMutton_Kheema.delete(0,END)
        textMutton_Kheema.focus()
    else:
        textMutton_Kheema.config(state=DISABLED)
        e_Mutton_Kheema.set("0")


def Mutton_Curry():
    if var5.get()==1:
        textMutton_Curry.config(state=NORMAL)
        textMutton_Curry.delete(0,END)
        textMutton_Curry.focus()
    else:
        textMutton_Curry.config(state=DISABLED)
        e_Mutton_Curry.set("0")


def Fish_Pulusu():
    if var6.get()==1:
        textFish_Pulusu.config(state=NORMAL)
        textFish_Pulusu.delete(0,END)
        textFish_Pulusu.focus()
    else:
        textFish_Pulusu.config(state=DISABLED)
        e_Fish_Pulusu.set("0")


def  Veg_Kofta():
    if var7.get()==1:
        textVeg_Kofta.config(state=NORMAL)
        textVeg_Kofta.delete(0,END)
        textVeg_Kofta.focus()
    else:
        textVeg_Kofta.config(state=DISABLED)
        e_Veg_Kofta.set("0")


def Butter_Panner():
    if var8.get()==1:
        textButter_Panner.config(state=NORMAL)
        textButter_Panner.delete(0,END)
        textButter_Panner.focus()
    else:
        textButter_Panner.config(state=DISABLED)
        e_Butter_Panner.set("0")
def  Mushroom():
    if var9.get()==1:
```

```
        textMushroom.config(state=NORMAL)
        textMushroom.delete(0,END)
        textMushroom.focus()
    else:
        textMushroom.config(state=DISABLED)
        e_Mushroom.set("0")


def Chicken_Dum_Biryani():
    if var10.get()==1:
        textChicken_Dum_Biryani.config(state=NORMAL)
        textChicken_Dum_Biryani.delete(0,END)
        textChicken_Dum_Biryani.focus()
    else:
        textChicken_Dum_Biryani.config(state=DISABLED)
        e_Chicken_Dum_Biryani.set("0")


def Chicken_Frypiece_Biryani():
    if var11.get()==1:
        textChicken_Frypiece_Biryani.config(state=NORMAL)
        textChicken_Frypiece_Biryani.delete(0,END)
        textChicken_Frypiece_Biryani.focus()
    else:
        textChicken_Frypiece_Biryani.config(state=DISABLED)
        e_Chicken_Frypiece_Biryani.set("0")


def Chicken_Biryani_Spl():
    if var12.get()==1:
        textChicken_Biryani_Spl.config(state=NORMAL)
        textChicken_Biryani_Spl.delete(0,END)
        textChicken_Biryani_Spl.focus()
    else:
        textChicken_Biryani_Spl.config(state=DISABLED)
        e_Chicken_Biryani_spl.set("0")




def Mutton_Nallighosh_Biryani():
    if var13.get()==1:
        textMutton_Nallighosh_Biryani.config(state=NORMAL)
        textMutton_Nallighosh_Biryani.delete(0,END)
        textMutton_Nallighosh_Biryani.focus()
    else:
        textMutton_Nallighosh_Biryani.config(state=DISABLED)
        e_Mutton_Nallighosh_Biryani.set("0")


def Mutton_Juicy_Biryani():
    if var14.get()==1:
        textMutton_Juicy_Biryani.config(state=NORMAL)
        textMutton_Juicy_Biryani.delete(0,END)
        textMutton_Juicy_Biryani.focus()
    else:
        textMutton_Juicy_Biryani.config(state=DISABLED)
        e_Mutton_Juicy_Biryani.set("0")
```

```
def Fish_Biryani():
    if var15.get()==1:
        textFish_Biryani.config(state=NORMAL)
        textFish_Biryani.delete(0,END)
        textFish_Biryani.focus()
    else:
        textFish_Biryani.config(state=DISABLED)
        e_Fish_Biryani.set("0")


def Prawn_Biryani():
    if var16.get()==1:
        textPrawn_Biryani.config(state=NORMAL)
        textPrawn_Biryani.delete(0,END)
        textPrawn_Biryani.focus()
    else:
        textPrawn_Biryani.config(state=DISABLED)
        e_Prawn_Biryani.set("0")


def Veg_Pulao():
    if var17.get()==1:
        textVeg_Pulao.config(state=NORMAL)
        textVeg_Pulao.delete(0,END)
        textVeg_Pulao.focus()
    else:
        textVeg_Pulao.config(state=DISABLED)
        e_Veg_Pulao.set("0")


def Panner_Biryani():
    if var18.get()==1:
        textPanner_Biryani.config(state=NORMAL)
        textPanner_Biryani.delete(0,END)
        textPanner_Biryani.focus()
    else:
        textPanner_Biryani.config(state=DISABLED)
        e_Prawn_Biryani.set("0")




def Chicken_Mangolia():
    if var19.get()==1:
        textChicken_Mangolia.config(state=NORMAL)
        textChicken_Mangolia.delete(0,END)
        textChicken_Mangolia.focus()
    else:
        textChicken_Mangolia.config(state=DISABLED)
        e_Chicken_Mangolia.set("0")


def Chicken_Manchuria():
    if var20.get()==1:
        textChicken_Manchuria.config(state=NORMAL)
        textChicken_Manchuria.delete(0,END)
        textChicken_Manchuria.focus()
    else:
        textChicken_Manchuria.config(state=DISABLED)
```

```
        e_Chicken_Manchuria.set("0")



def Chicken_Drumsticks():
    if var21.get()==1:
        textChicken_Drumsticks.config(state=NORMAL)
        textChicken_Drumsticks.delete(0,END)
        textChicken_Drumsticks.focus()
    else:
        textChicken_Drumsticks.config(state=DISABLED)
        e_Chicken_Drumsticks.set("0")


def Chicken_Kabab():
    if var22.get()==1:
        textChicken_kabab.config(state=NORMAL)
        textChicken_kabab.delete(0,END)
        textChicken_kabab.focus()
    else:
        textChicken_kabab.config(state=DISABLED)
        e_Chicken_kabab.set("0")


def Egg_Manchuria():
    if var23.get()==1:
        textEgg_Manchuria.config(state=NORMAL)
        textEgg_Manchuria.delete(0,END)
        textEgg_Manchuria.focus()
    else:
        textEgg_Manchuria.config(state=DISABLED)
        e_Egg_Manchuria.set("0")


def Veg_Manchuria():
    if var24.get()==1:
        textVeg_Manchuria.config(state=NORMAL)
        textVeg_Manchuria.delete(0,END)
        textVeg_Manchuria.focus()
    else:
        textVeg_Manchuria.config(state=DISABLED)
        e_Veg_Manchuria.set("0")


def Fried_Mushroom():
    if var25.get()==1:
        textFried_Mushroom.config(state=NORMAL)
        textFried_Mushroom.delete(0,END)
        textFried_Mushroom.focus()
    else:
        textFried_Mushroom.config(state=DISABLED)
        e_Fried_Mushroom.set("0")


def Baby_Corn():
    if var26.get()==1:
        textBabyCorn.config(state=NORMAL)
        textBabyCorn.delete(0,END)
        textBabyCorn.focus()
```

```
    else:
        textBabyCorn.config(state=DISABLED)
        e_BabyCorn.set("0")


def Panner_Tikka():
    if var27.get()==1:
        textPanner_Tikka.config(state=NORMAL)
        textPanner_Tikka.delete(0,END)
        textPanner_Tikka.focus()
    else:
        textPanner_Tikka.config(state=DISABLED)
        e_Panner_Tikka.set("0")



root=Tk()
root.geometry("1350x750+0+0")
root.title("Fine Dining With Contactless Ordering System")
root.config(bg="light yellow")
topFrame=Frame(root,bd=10,relief=RIDGE,bg="light yellow")
topFrame.pack(side=TOP)


labelTitle=Label(topFrame,text="MENU",font=("times new roman",30,"bold"),fg="dark green",bd=9,bg="light
green",width=51)
labelTitle.grid(row=0,column=0)




#frames
menuFrame=Frame(root,bd=10,relief=RIDGE)
menuFrame.pack(side=LEFT)


costFrame=Frame(menuFrame,bd=4,relief=RIDGE)
costFrame.pack(side=BOTTOM)


maincourseFrame=LabelFrame(menuFrame,text="Main Course",font=("arial",19,"bold"),bd=10,relief=RIDGE)
maincourseFrame.pack(side=LEFT)
biryanisFrame=LabelFrame(menuFrame,text="Biryani",font=("arial",19,"bold"),bd=10,relief=RIDGE)
biryanisFrame.pack(side=LEFT)
startersFrame=LabelFrame(menuFrame,text="Starters",font=("arial",19,"bold"),bd=10,relief=RIDGE)
startersFrame.pack(side=LEFT)


rightFrame=Frame(root,bd=15,relief=RIDGE)
rightFrame.pack(side=RIGHT)
calculatorFrame=Frame(rightFrame,bd=1,relief=RIDGE)
calculatorFrame.pack()


recieptFrame=Frame(rightFrame,bd=4,relief=RIDGE)
recieptFrame.pack()
buttonFrame=Frame(rightFrame,bd=2,relief=RIDGE)
buttonFrame.pack()
```

```
#variables
var1=IntVar()
var2=IntVar()
var3=IntVar()
var4=IntVar()
var5=IntVar()
var6=IntVar()
var7=IntVar()
var8=IntVar()
var9=IntVar()
var10=IntVar()
var11=IntVar()
var12=IntVar()
var13=IntVar()
var14=IntVar()
var15=IntVar()
var16=IntVar()
var17=IntVar()
var18=IntVar()
var19=IntVar()
var20=IntVar()
var21=IntVar()
var22=IntVar()
var23=IntVar()
var24=IntVar()
var25=IntVar()
var26=IntVar()
var27=IntVar()


e_Butter_Chicken=StringVar()
e_Chicken_Kolapuri=StringVar()
e_Telangana_Chicken=StringVar()
e_Mutton_Kheema=StringVar()
e_Mutton_Curry=StringVar()
e_Fish_Pulusu=StringVar()
e_Veg_Kofta=StringVar()
e_Butter_Panner=StringVar()
e_Mushroom=StringVar()


e_Chicken_Dum_Biryani=StringVar()
e_Chicken_Frypiece_Biryani=StringVar()
e_Chicken_Biryani_spl=StringVar()
e_Mutton_Nallighosh_Biryani=StringVar()
e_Mutton_Juicy_Biryani=StringVar()
e_Fish_Biryani=StringVar()
e_Prawn_Biryani=StringVar()
e_Veg_Pulao=StringVar()
e_Panner_Biryani=StringVar()



e_Chicken_Mangolia=StringVar()
e_Chicken_Manchuria=StringVar()
e_Chicken_Drumsticks=StringVar()
e_Chicken_kabab=StringVar()
e_Egg_Manchuria=StringVar()
e_Veg_Manchuria=StringVar()
e_Fried_Mushroom=StringVar()
```

```
e_BabyCorn=StringVar()
e_Panner_Tikka=StringVar()


costofMaincoursevar=StringVar()
costofBiryanivar=StringVar()
costofStartersvar=StringVar()
costofSubtotalvar=StringVar()
costofServicetaxvar=StringVar()
costofTotalcostvar=StringVar()

e_Butter_Chicken.set("0")
e_Chicken_Kolapuri.set("0")
e_Telangana_Chicken.set("0")
e_Mutton_Kheema.set('0')
e_Mutton_Curry.set("0")
e_Fish_Pulusu.set("0")
e_Veg_Kofta.set("0")
e_Butter_Panner.set("0")
e_Mushroom.set("0")


e_Chicken_Dum_Biryani.set("0")
e_Chicken_Frypiece_Biryani.set("0")
e_Chicken_Biryani_spl.set("0")
e_Mutton_Nallighosh_Biryani.set("0")
e_Mutton_Juicy_Biryani.set("0")
e_Fish_Biryani.set("0")
e_Prawn_Biryani.set("0")
e_Veg_Pulao.set("0")
e_Panner_Biryani.set("0")




e_Chicken_Mangolia.set("0")
e_Chicken_Manchuria.set("0")
e_Chicken_Drumsticks.set("0")
e_Chicken_kabab.set("0")
e_Egg_Manchuria.set("0")
e_Veg_Manchuria.set("0")
e_Fried_Mushroom.set("0")
e_BabyCorn.set("0")
e_Panner_Tikka.set("0")




##maincourse
Butter_Chicken =Checkbutton(maincourseFrame,text="Butter Chicken-
349",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var1,command=Butter_Chicken)
Butter_Chicken.grid(row=0,column=0,sticky=W)
Chicken_Kolapuri =Checkbutton(maincourseFrame,text="Chicken kolapuri-
369",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var2,command=Chicken_Kolapuri)
Chicken_Kolapuri.grid(row=1,column=0,sticky=W)
Telangana_Chicken =Checkbutton(maincourseFrame,text="Telangana Chicken-
369",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var3,command=Telangana_Chicken)
Telangana_Chicken.grid(row=2,column=0,sticky=W)
Mutton_Kheema=Checkbutton(maincourseFrame,text="Mutton Kheema-
449",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var4,command=Mutton_Kheema)
```

```
Mutton_Kheema.grid(row=3,column=0,sticky=W)
Mutton_Curry=Checkbutton(maincourseFrame,text="Mutton Curry-
424",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var5,command=Mutton_Curry)
Mutton_Curry.grid(row=4,column=0,sticky=W)
Fish_Pulusu=Checkbutton(maincourseFrame,text="Fish Pulusu-
489",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var6,command=Fish_Pulusu)
Fish_Pulusu.grid(row=5,column=0,sticky=W)
Veg_Kofta=Checkbutton(maincourseFrame,text="Veg Kofta-
299",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var7,command=Veg_Kofta)
Veg_Kofta.grid(row=6,column=0,sticky=W)
Butter_Panner=Checkbutton(maincourseFrame,text="Butter Panner-
199",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var8,command=Butter_Panner)
Butter_Panner.grid(row=7,column=0,sticky=W)
Mushroom=Checkbutton(maincourseFrame,text="Mushroom-
169",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var9,command=Mushroom)
Mushroom.grid(row=8,column=0,sticky=W)


#entry fields for maincourse
textButter_Chicken=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Butte
r_Chicken)
textButter_Chicken.grid(row=0,column=1)
textChicken_Kolapuri=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Ch
icken_Kolapuri)
textChicken_Kolapuri.grid(row=1,column=1)
textTelangana_Chicken=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_T
elangana_Chicken)
textTelangana_Chicken.grid(row=2,column=1)
textMutton_Kheema=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Mut
ton_Kheema)
textMutton_Kheema.grid(row=3,column=1)
textMutton_Curry=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Mutto
n_Curry)
textMutton_Curry.grid(row=4,column=1)
textFish_Pulusu=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Fish_Pul
usu)
textFish_Pulusu.grid(row=5,column=1)
textVeg_Kofta=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Veg_Koft
a)
textVeg_Kofta.grid(row=6,column=1)
textButter_Panner=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Butter
_Panner)
textButter_Panner.grid(row=7,column=1)
textMushroom=Entry(maincourseFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Mushroo
m)
textMushroom.grid(row=8,column=1)


#Biryaniframe
Chicken_Dum_Biryani=Checkbutton(biryanisFrame,text="Chicken Dum Biryani-
350",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var10,command=Chicken_Dum_Biryani)
Chicken_Dum_Biryani.grid(row=0,column=0,sticky=W)
Chicken_Frypiece_Biryani=Checkbutton(biryanisFrame,text="Chicken Frypiece Biryani-
399",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var11,command=Chicken_Frypiece_Biryani)
Chicken_Frypiece_Biryani.grid(row=1,column=0,sticky=W)
Chicken_Biryani_Spl=Checkbutton(biryanisFrame,text="Chicken Biryani Spl-
399",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var12,command=Chicken_Biryani_Spl)
Chicken_Biryani_Spl.grid(row=2,column=0,sticky=W)
Mutton_Nallighosh_Biryani=Checkbutton(biryanisFrame,text="Mutton Nallighosh Biryani-
449",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var13,command=Mutton_Nallighosh_Biryani)
```

```
Mutton_Nallighosh_Biryani.grid(row=3,column=0,sticky=W)
Mutton_Juicy_Biryani=Checkbutton(biryanisFrame,text="Mutton Juicy Biryani-
469",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var14,command=Mutton_Juicy_Biryani)
Mutton_Juicy_Biryani.grid(row=4,column=0,sticky=W)
Fish_Biryani=Checkbutton(biryanisFrame,text="Fish Biryani-
510",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var15,command=Fish_Biryani)
Fish_Biryani.grid(row=5,column=0,sticky=W)
Prawn_Biryani=Checkbutton(biryanisFrame,text="Prawn Biryani-
470",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var16,command=Prawn_Biryani)
Prawn_Biryani.grid(row=6,column=0,sticky=W)
Veg_Pulao=Checkbutton(biryanisFrame,text="Veg Pulao-
220",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var17,command=Veg_Pulao)
Veg_Pulao.grid(row=7,column=0,sticky=W)
Panner_Biryani=Checkbutton(biryanisFrame,text="Panner Biryani-
250",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var18,command=Panner_Biryani)
Panner_Biryani.grid(row=8,column=0,sticky=W)


#entry fields for biryani
textChicken_Dum_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_C
hicken_Dum_Biryani)
textChicken_Dum_Biryani.grid(row=0,column=1)
textChicken_Frypiece_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=
e_Chicken_Frypiece_Biryani)
textChicken_Frypiece_Biryani.grid(row=1,column=1)
textChicken_Biryani_Spl=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Ch
icken_Biryani_spl)
textChicken_Biryani_Spl.grid(row=2,column=1)
textMutton_Nallighosh_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable
=e_Mutton_Nallighosh_Biryani)
textMutton_Nallighosh_Biryani.grid(row=3,column=1)
textMutton_Juicy_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_M
utton_Juicy_Biryani)
textMutton_Juicy_Biryani.grid(row=4,column=1)
textFish_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Fish_Birya
ni)
textFish_Biryani.grid(row=5,column=1)
textPrawn_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Prawn_Bi
ryani)
textPrawn_Biryani.grid(row=6,column=1)
textVeg_Pulao=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Veg_Pulao)
textVeg_Pulao.grid(row=7,column=1)
textPanner_Biryani=Entry(biryanisFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Panner_
Biryani)
textPanner_Biryani.grid(row=8,column=1)



#Startersframe
Chicken_Mangolia=Checkbutton(startersFrame,text="Chicken Mangolia-
210",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var19,command=Chicken_Mangolia)
Chicken_Mangolia.grid(row=0,column=0,sticky=W)
Chicken_Manchuria=Checkbutton(startersFrame,text="Chicken Manchuria-
189",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var20,command=Chicken_Manchuria)
Chicken_Manchuria.grid(row=1,column=0,sticky=W)
Chicken_Drumsticks=Checkbutton(startersFrame,text="Chicken Drumsticks-
210",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var21,command=Chicken_Drumsticks)
```

```
Chicken_Drumsticks.grid(row=2,column=0,sticky=W)
Chicken_Kabab=Checkbutton(startersFrame,text="Chicken Kabab-
240",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var22,command=Chicken_Kabab)
Chicken_Kabab.grid(row=3,column=0,sticky=W)
Egg_Manchuria=Checkbutton(startersFrame,text="Egg Manchuria-
199",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var23,command=Egg_Manchuria)
Egg_Manchuria.grid(row=4,column=0,sticky=W)
Veg_Manchuria=Checkbutton(startersFrame,text="Veg Manchuria-
170",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var24,command=Veg_Manchuria)
Veg_Manchuria.grid(row=5,column=0,sticky=W)
Fried_Mushroom=Checkbutton(startersFrame,text="Fried Mushroom-
170",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var25,command=Fried_Mushroom)
Fried_Mushroom.grid(row=6,column=0,sticky=W)
Baby_Corn=Checkbutton(startersFrame,text="Baby Corn-
190",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var26,command=Baby_Corn)
Baby_Corn.grid(row=7,column=0,sticky=W)
Panner_Tikka=Checkbutton(startersFrame,text="Panner Tikka-
199",font=("arial",8,"bold"),onvalue=1,offvalue=0,variable=var27,command=Panner_Tikka)
Panner_Tikka.grid(row=8,column=0,sticky=W)
```

#entry fields for starters

```
textChicken_Mangolia=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Chick
en_Mangolia)
textChicken_Mangolia.grid(row=0,column=1)
textChicken_Manchuria=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Chic
ken_Manchuria)
textChicken_Manchuria.grid(row=1,column=1)
textChicken_Drumsticks=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Chic
ken_Drumsticks)
textChicken_Drumsticks.grid(row=2,column=1)
textChicken_kabab=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Chicken_
kabab)
textChicken_kabab.grid(row=3,column=1)
textEgg_Manchuria=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Egg_Man
churia)
textEgg_Manchuria.grid(row=4,column=1)
textVeg_Manchuria=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Veg_Ma
nchuria)
textVeg_Manchuria.grid(row=5,column=1)
textFried_Mushroom=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Fried_
Mushroom)
textFried_Mushroom.grid(row=6,column=1)
textBabyCorn=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_BabyCorn)
textBabyCorn.grid(row=7,column=1)
textPanner_Tikka=Entry(startersFrame,font=("arial",12,"bold"),bd=7,width=6,state=DISABLED,textvariable=e_Panner_Tik
ka)
textPanner_Tikka.grid(row=8,column=1)
```

#costlabels and entry fields

```
labelcostofMaincourse=Label(costFrame,text="Cost of Maincourse",font=("arial",16,"bold"),bg="light yellow")
```

```
labelcostofMaincourse.grid(row=0,column=0)


textCostofMaincourse=Entry(costFrame,font=("arial",16,"bold"),bd=6,width=14,state="readonly",textvariable=costofMainc
oursevar)
textCostofMaincourse.grid(row=0,column=1,padx=41)


labelcostofBiryani=Label(costFrame,text="Cost of Biryani",font=("arial",16,"bold"),bg="light yellow")
labelcostofBiryani.grid(row=1,column=0)


textCostofBiryani=Entry(costFrame,font=("arial",16,"bold"),bd=6,width=14,state="readonly",textvariable=costofBiryanivar)
textCostofBiryani.grid(row=1,column=1,padx=41)


labelcostofStarters=Label(costFrame,text="Cost of Starters",font=("arial",16,"bold"),bg="light yellow")
labelcostofStarters.grid(row=2,column=0)


textCostofStarters=Entry(costFrame,font=("arial",16,"bold"),bd=6,width=14,state="readonly",textvariable=costofStartersvar
)
textCostofStarters.grid(row=2,column=1,padx=41)




labelSubtotal=Label(costFrame,text="Sub total ",font=("arial",16,"bold"),bg="light yellow")
labelSubtotal.grid(row=0,column=2)


textCostofSubtotal=Entry(costFrame,font=("arial",16,"bold"),bd=6,width=14,state="readonly",textvariable=costofSubtotalva
r)
textCostofSubtotal.grid(row=0,column=3,padx=41)


labelServicetax=Label(costFrame,text="Service Tax",font=("arial",16,"bold"),bg="light yellow")
labelServicetax.grid(row=1,column=2)


textCostofServicetax=Entry(costFrame,font=("arial",16,"bold"),bd=6,width=14,state="readonly",textvariable=costofServicet
axvar)
textCostofServicetax.grid(row=1,column=3,padx=41)


labelTotalcost=Label(costFrame,text="Total Cost",font=("arial",16,"bold"),bg="light yellow")
labelTotalcost.grid(row=2,column=2)


textTotalcost=Entry(costFrame,font=("arial",16,"bold"),bd=6,width=14,state="readonly",textvariable=costofTotalcostvar)
textTotalcost.grid(row=2,column=3,padx=41)




#buttons


buttonTotal=Button(buttonFrame,text="Total",font=("arial",14,"bold"),command=totalcost)
buttonTotal.grid(row=0,column=0)
```

```
buttonReceipt=Button(buttonFrame,text="Receipt",font=("arial",14,"bold"),command=receipt)
buttonReceipt.grid(row=0,column=1)
buttonSave=Button(buttonFrame,text="Save",font=("arial",14,"bold"),command=save)
buttonSave.grid(row=0,column=2)
buttonSend=Button(buttonFrame,text="Send",font=("arial",14,"bold"),command=send)
buttonSend.grid(row=0,column=3)
buttonReset=Button(buttonFrame,text="Reset",font=("arial",14,"bold"),command=reset)
buttonReset.grid(row=0,column=4)


#textarea for receipt


textReciept=Text(recieptFrame,font=("arial",12,"bold"),width=42,height=14)
textReciept.grid(row=0,column=0)




#calculator
operator="
def buttonClick(numbers):
    global operator
    operator=operator+numbers
    calculatorField.delete(0,END)
    calculatorField.insert(END,operator)


def  clear():
    global operator
    operator="
    calculatorField.delete(0,END)


def answer():
    global operator
    result=str(eval(operator))
    calculatorField.delete(0,END)
    calculatorField.insert(0,result)
    operator="

calculatorField=Entry(calculatorFrame,font=("arial",16,"bold"),width=32)
calculatorField.grid(row=0,column=0,columnspan=4)


button7=Button(calculatorFrame,text="7",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("7")
)
button7.grid(row=1,column=0)
button8=Button(calculatorFrame,text="8",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("8")
)
button8.grid(row=1,column=1)
button9=Button(calculatorFrame,text="9",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("9")
)
button9.grid(row=1,column=2)
buttonPlus=Button(calculatorFrame,text="+",font=("arial",16,"bold"),width=4,bg="yellow",command=lambda:buttonClick(
"+"))
buttonPlus.grid(row=1,column=3)
button4=Button(calculatorFrame,text="4",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("4")
)
button4.grid(row=2,column=0)
```

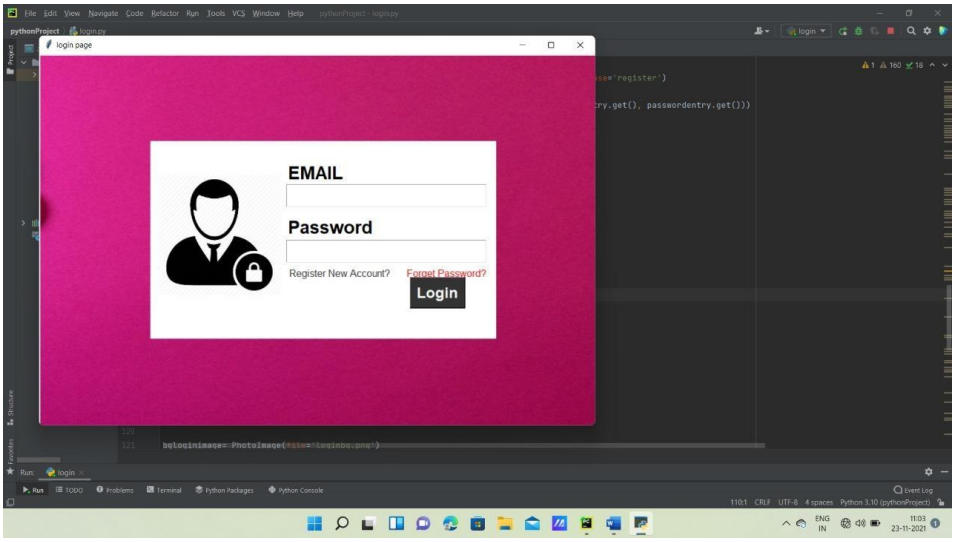
```
button5=Button(calculatorFrame,text="5",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("5")
)
button5.grid(row=2,column=1)
button6=Button(calculatorFrame,text="6",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("6")
)
button6.grid(row=2,column=2)
buttonMinus=Button(calculatorFrame,text="-
",font=("arial",16,"bold"),width=4,bg="yellow",command=lambda:buttonClick("-"))
buttonMinus.grid(row=2,column=3)
button1=Button(calculatorFrame,text="1",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("1")
)
button1.grid(row=3,column=0)
button2=Button(calculatorFrame,text="2",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("2")
)
button2.grid(row=3,column=1)
button3=Button(calculatorFrame,text="3",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("3")
)
button3.grid(row=3,column=2)
buttonMult=Button(calculatorFrame,text="",font=("arial",16,"bold"),width=4,bg="yellow",command=lambda:buttonClick("
"))
buttonMult.grid(row=3,column=3)
buttonAns=Button(calculatorFrame,text="Ans",font=("arial",16,"bold"),width=7,bg="yellow",command=answer)
buttonAns.grid(row=4,column=0)
buttonClear=Button(calculatorFrame,text="Clear",font=("arial",16,"bold"),width=7,bg="yellow",command=clear)
buttonClear.grid(row=4,column=1)
button0=Button(calculatorFrame,text="0",font=("arial",16,"bold"),width=7,bg="yellow",command=lambda:buttonClick("0")
)
button0.grid(row=4,column=2)
buttonDiv=Button(calculatorFrame,text="/",font=("arial",16,"bold"),width=4,bg="yellow",command=lambda:buttonClick("/
"))
buttonDiv.grid(row=4,column=3)

root.mainloop()
```

V. SCREENSHOTS

5. Screenshots

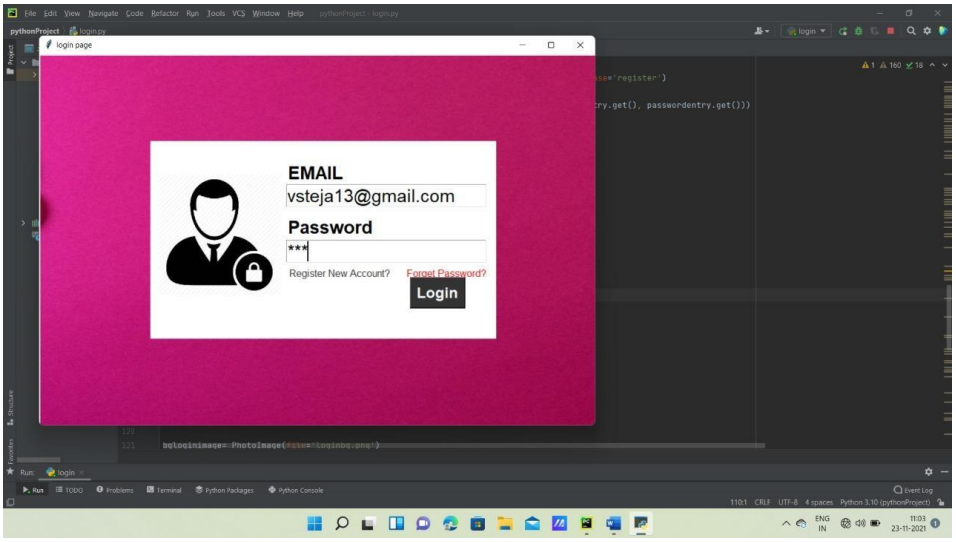
5.1 Login Page

The login page allows a user to log into their account so that the visit count will increase and appropriate discount will be applied.

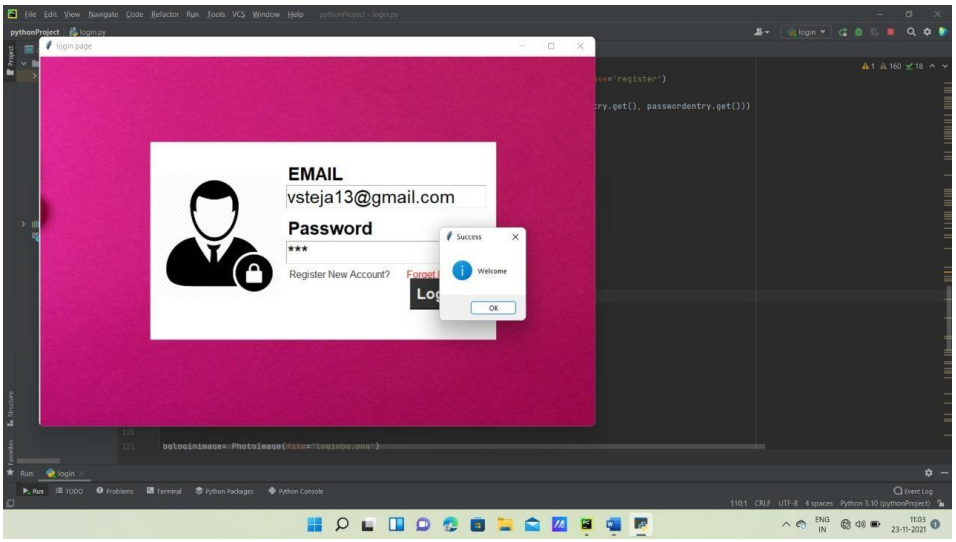


Screenshot 5.1.1: Login page without data filled.

- Here the customer needs to fill the credentials of his/ her previous login.



Screenshot 5.1.2: Login page with data filled.

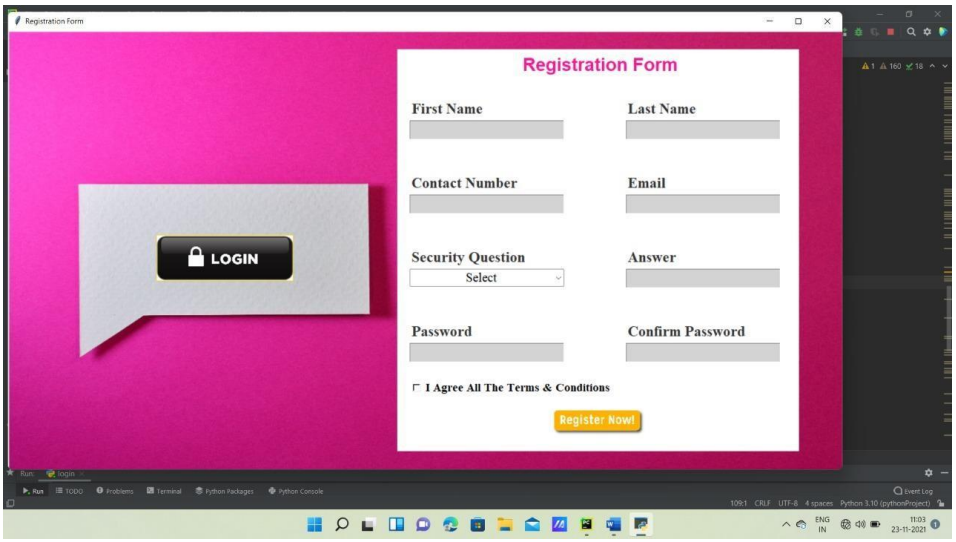


Screenshot 5.1.3: Login Success message.

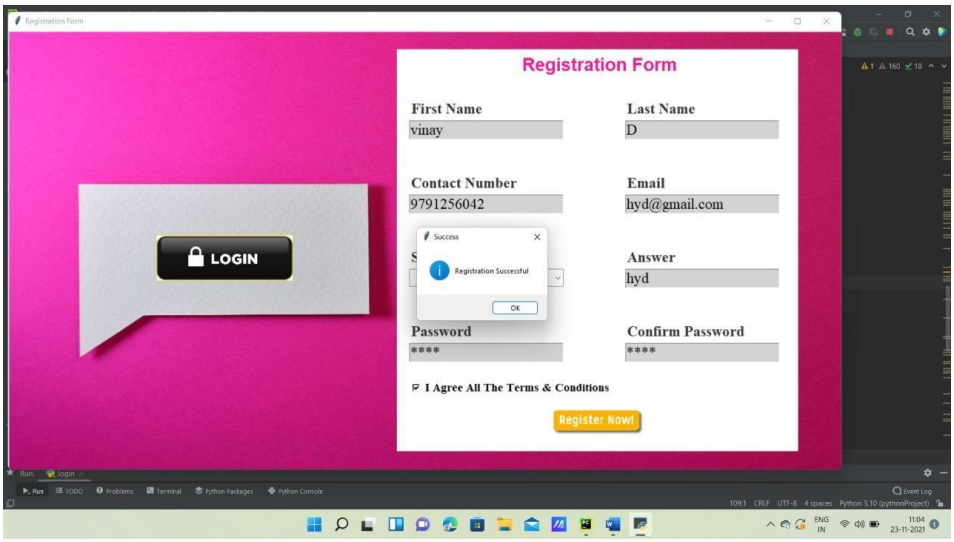
- When user enters the credentials and gets a welcome message, they are redirected to Menu page

5.2 REGISTER PAGE:

Registration page allows a new user to create an account and the visit count will be initialized to 1.



Screenshot 5.2.1: Registration form without data filled.

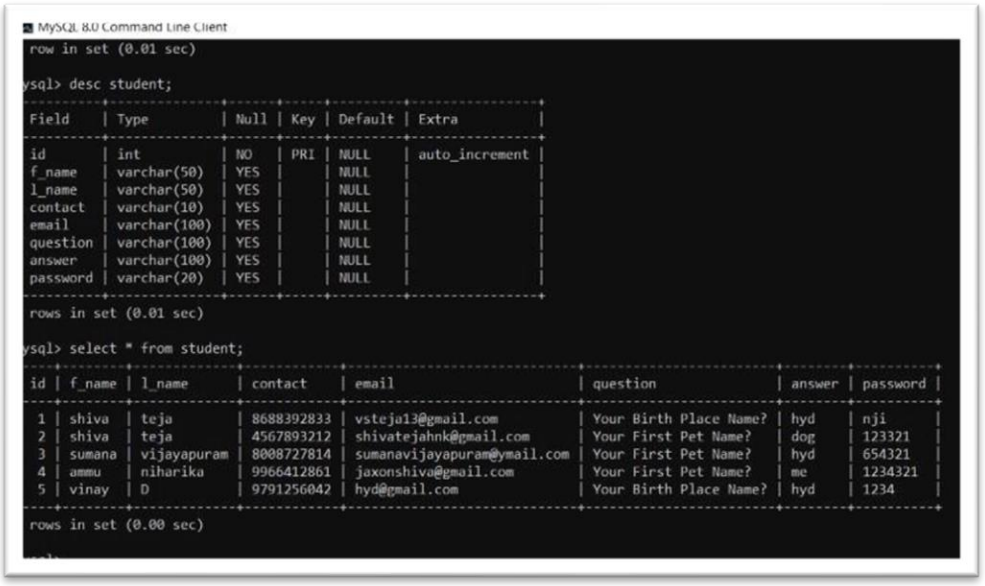


Screenshot 5.2.2: Registration success message.

- When a new user comes to the restaurant they will register and these credentials can be used to login for their next visits

5.3 DATABASE TABLES:

In this screenshot we can see the tables where customer details are stored from login and registration forms.



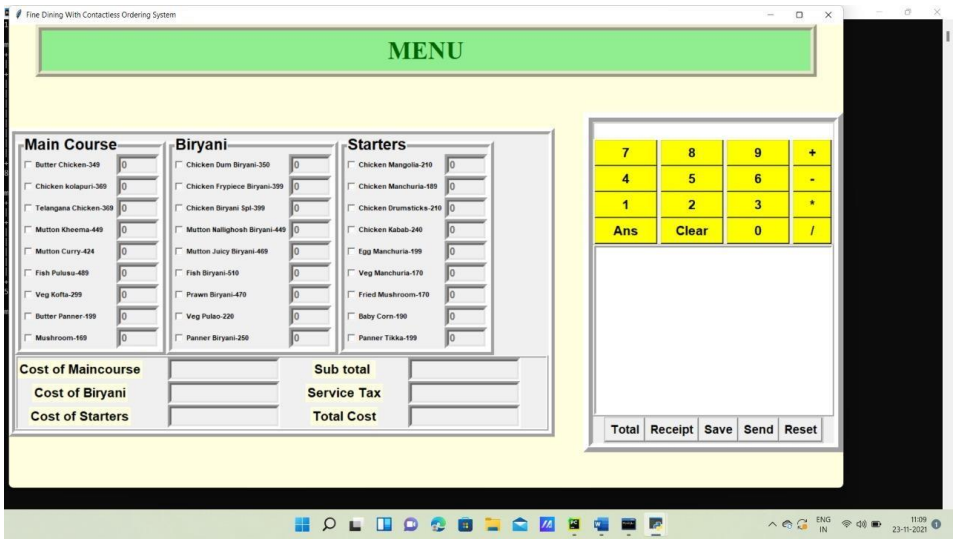
```
MySQL 8.0 Command Line Client
row in set (0.01 sec)

ysql> desc student;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| f_name   | varchar(50)  | YES  |     | NULL    |                |
| l_name   | varchar(50)  | YES  |     | NULL    |                |
| contact  | varchar(10)  | YES  |     | NULL    |                |
| email    | varchar(100) | YES  |     | NULL    |                |
| question | varchar(100) | YES  |     | NULL    |                |
| answer   | varchar(100) | YES  |     | NULL    |                |
| password | varchar(20)  | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
rows in set (0.01 sec)

ysql> select * from student;
+----+--------+-----------+------------+-------------------------------+-----------------------+--------+----------+
| id | f_name | l_name    | contact    | email                         | question              | answer | password |
+----+--------+-----------+------------+-------------------------------+-----------------------+--------+----------+
|  1 | shiva  | teja      | 8688392833 | vsteja13@gmail.com            | Your Birth Place Name? | hyd    | nji      |
|  2 | shiva  | teja      | 4567893212 | shivatejahnk@gmail.com        | Your First Pet Name?   | dog    | 123321   |
|  3 | sumana | vijayapuram | 8008727814 | sumanavijayapuram@ymail.com | Your First Pet Name?   | hyd    | 654321   |
|  4 | ammu   | niharika  | 9966412861 | jaxonshiva@gmail.com          | Your First Pet Name?   | me     | 1234321  |
|  5 | vinay  | D         | 9791256042 | hyd@gmail.com                 | Your Birth Place Name? | hyd    | 1234     |
+----+--------+-----------+------------+-------------------------------+-----------------------+--------+----------+
rows in set (0.00 sec)
```

Screenshot 5.3: Tables in database backend.

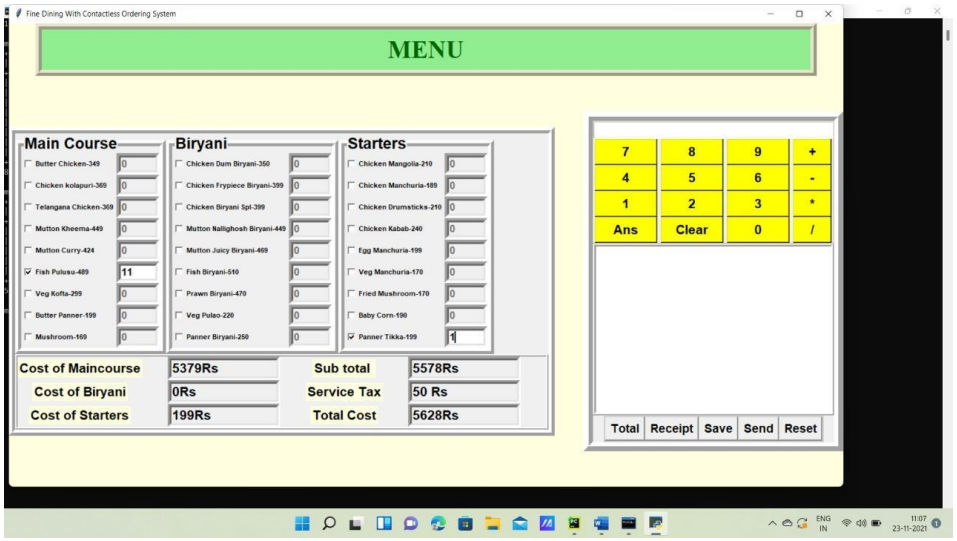
- The data entered during registration time are stored in the above database

5.4 Menu page:

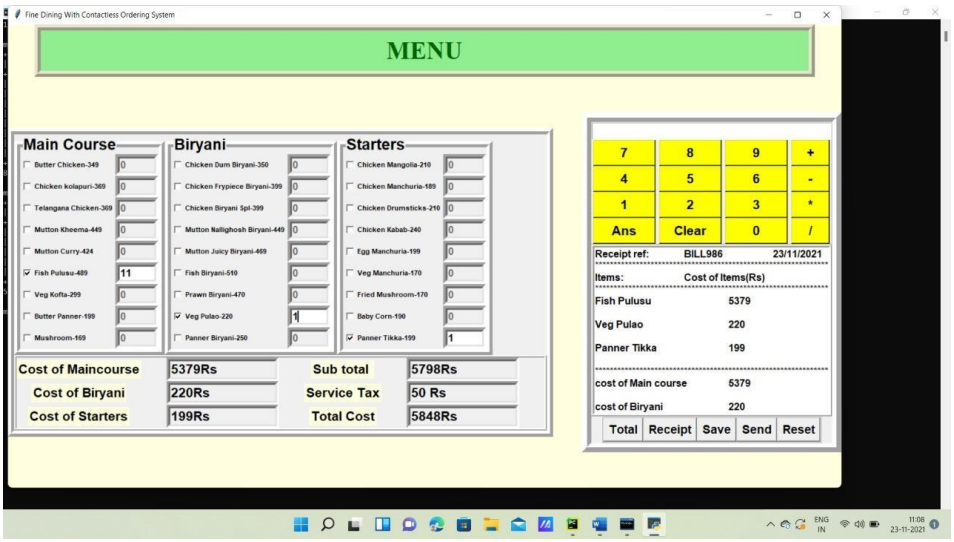
The menu page allows the customer to order the food, find the total, generate a receipt and send a copy of the receipt to their mobile number , save the bill copy and reset the order.



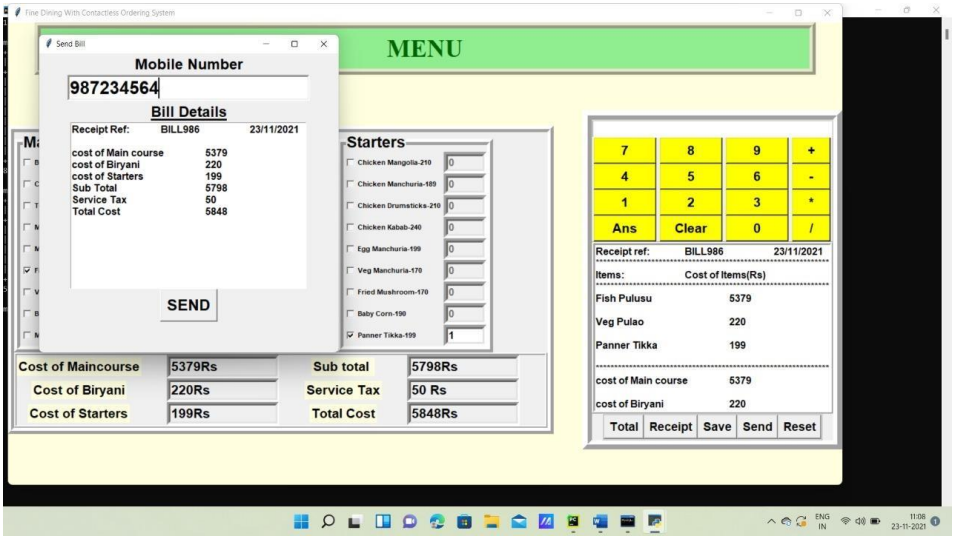
Screenshot 5.4.1: Menu page redirected from login page.



Screenshot 5..4.2: Menu with order selected and total amount calculated.



Screenshot 5.4.3: Menu with order selected, total amount calculated and receipt generated.



Screenshot 5.4.4: Receipt generated and sent to the specified phone number and kitchen.

- Using the send button, the user can send a copy of the bill to desired phone number via SMS

VI. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TEST CASES

6.3.1 CUSTOMER LOGIN

Test case ID	Test case name	Purpose	Test Case	Output
1	Customer 1st login	Use it for visit count Initial-1	The customer enters the mobile number and the OTP received to that mobile number	Login successful
2	Customer n^{th} login	Use it for visit count Count-n	The customer enters the mobile number and the OTP received to that mobile number	Login successful

6.3.2 SELECTING ORDER

Test case ID	Test case name	Purpose	Input	Output
1	Selection 1	To check if the system performs its task	An order is selected	Order is sent.
2	Selection 2	To check if the system performs its task	An order is selected with quantity zero	Order is sent excluding the items with quantity as zero.
3	Selection 3	To check if the system performs its task	An order is selected without quantity for all items	Send button is inactive.

VII. CONCLUSION

7. CONCLUSION & FUTURESCOPE

7.1 PROJECT CONCLUSION

The project titled as “ADMIRABLE DINNING WITH CONTACT-FREE ORDERING” is a console-based application. This software is beneficial to the restaurant owners in terms of business being carried on and also the customers in terms of being affected with COVID-19as there will be minimal human contact. This software is developed with scalability in mind. The software is developed with package approach. All packages in the system have been tested with valid data and invalid data and everything work successfully. Thus the system has fulfilled all the objectives identified and is able to replace the existing system.

The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. The project gives good idea on developing a full-fledged application satisfying the user requirements.

The system is very flexible and versatile. Validation checks induced have greatly reduced errors. Provisions have been made to upgrade the software. The application has been tested with live data and has provided a successful result. Hence the software has proved to work efficiently.

7.2 FUTURE SCOPE

In future we can use other techniques to forward the order directly intro kitchen using cloud network without the help of kitchen staff. The software can be developed further into an application which includes lot of modules because the proposed system is developed on the view of future.