CAFÉ MANAGEMENT SYSTEM

(USING PYTHON AND MySQL)

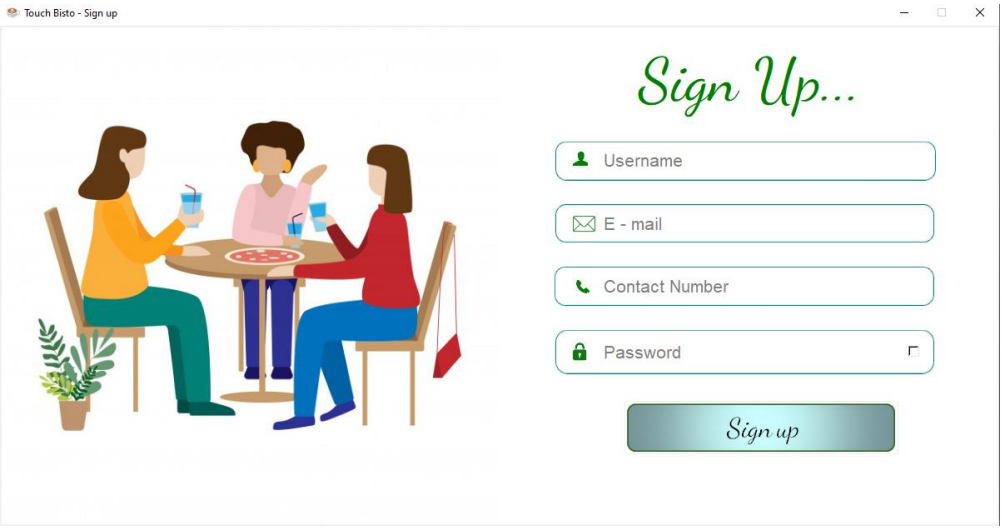


TABLE OF CONTENTS

SYNOPSIS

The **'Cafe Management System'** is an application that has been designed for the benefit of local Cafes for better administration of the facilities availed by customers and workers.

The system helps the administration to keep track of sales, employee details and bills. It has an exclusive feature for registering into the employees into the system with ease. The employee also can view his details and sales of the day.

The project was done using Python 3.8(64-bit) as front-end. Python was executed in PyCharm (2020.2.3) which is an environment that is extensively used for Python. This platform also helps the user to learn and execute different code at the same time.

MySQL 8.0 has been used as the back-end software where all data has been stored in a database, in the form of relations. The module mysql.connector allows user to connect Python and MySQL and execute MySQL queries directly from Python.

INTRODUCTION TO THE PROJECT

The **'Cafe Management system'** will allow efficient administration of Cafes and coffee shops. It will help the management of the shop to keep a record of multiple parameters involved in smooth functioning of a cafe. It also allows easy registration and display of the sales of beverages and snakes. It also helps in maintaining details of employees.

The features of the system can be accessed when logged in. In case the user is a first timer the user can register himself/herself by entering the required details in the signup window which can be accessed from the login page as well as home page.

The home page acts as a gateway to the

- Menu page
- Employee details page
- Billing page

❖ **Menu :**

The Menu page displays the menu of the cafe providing the users with the information of delicacies offered by the café.

❖ **Employee details:**

The employee details page manages the details of employees working in the café. The user can add employee, delete employee, view employees and search for employee details.

❖ **Billing:**

The billing page is used to print the customer's bill and save a copy for the user's use. It is used to file reports on the sales of the day and progress of employees sales and marketing skills.

SYSTEM CONFIGURATION

SOFTWARE USED:

- Python 3.8(64-bit)-Front end
- PyCharm 2020.2.3
- Operating system – Microsoft® Windows 10 Pro (64-bit operatingsystem, x64-based processor)
- MySQL 8.0 – Back end

HARDWARE USED:

- 8 GB RAM (2 GB+ recommended)
- Hard disk capacity: 1 TB
- Basic CPU – Intel® Core™ i7-8565U CPU @1.80 GHz
- Output device – 1024x768 Monitor (resolution)
- Input device – Mouse, Keyboard

ABOUT PYTHON

Python was developed by Guido Van Rossum in 1991. Python is based on or influenced by two programming languages :

- ABC language
- Modula-3

It is a high-level language that was created for general purpose of programming.

Advantages:

- **Cross-Platform Language:** Python runs equally well on a variety of platforms like Windows, Linux/UNIX, Macintosh etc.
- **Interpreted language:** Makes it easy-to-debug language as it executes the code line by line, making it suitable for beginners also.
- **Expressive language:** Being HLL(High Level Language) it is capable to expressing the code simpler than others.
- **Free and Open Source:** Python language is freely available i.e. without any cost. Its source code is also available online. It can be extended, modified and improved according to the will of the user.

Disadvantages:-

- Python is a slow language since it is interpreted. It has lesser libraries when compared to C, C++ etc.

- This application is designed using Python 3.9(64-bit) with the environment called PyCharm 2020.1

SOURCE CODE

```
"..............................................................................................................................
.....................................................................................................................'
#Importing required modules
import os
from pprint import pprint
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import *
import tkinter.scrolledtext as tkscrolled
from tkinter import messagebox
from tkinter import ttk
from tkinter import filedialog
import mysql.connector as m
'..............................................................................................................................
.....................................................................................................................'
#Connecting my sql to python
pipe=m.connect(host='localhost',user='root',password='12345',database='touch_bistro',auth_plugin='mys
ql_native_password')
wire=pipe.cursor()
'..............................................................................................................................
.....................................................................................................................'
#Creating tk window and hiding it
root=tk.Tk()
root.withdraw()
'..............................................................................................................................
.....................................................................................................................'
#Creating login window
login=None
def Login():

    #Checking if other windows are open or not
    if (signup is not None) and signup.winfo_exists():
        signup.withdraw()
    if (home is not None) and home.winfo_exists():
        home.withdraw()
    if (menu is not None) and menu.winfo_exists():
        menu.withdraw()
    if (billing is not None) and billing.winfo_exists():
        billing.withdraw()
    if (employee_details is not None) and employee_details.winfo_exists():
        employee_details.withdraw()

    #Size,icon and title of login page
    global login
    login=Toplevel()
    window_width = 1200
    window_height = 600
```

```
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    position_top = int(screen_height/2 -window_height/2)
    position_right = int(screen_width / 2 - window_width/2)
    login.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')
    login.title('Touch Bistro - Login')
    login.iconbitmap("latte.ico")
    login.resizable(False,False)

    #Creating the canvas
    C1 = Canvas(login,width=1200, height=600,borderwidth=0,bd=0 ,bg='white')
    C1.pack(expand=YES, fill=BOTH)

    #Inserting the cafe pic
    Cafe1= PhotoImage(file="k1.png")
    C1.create_image(0,0,anchor=NW,image=Cafe1)

    #Inserting the username and password image
    usernamepassword_img=PhotoImage(file="l2.png")
    C1.create_image(750,0,anchor=NW,image=usernamepassword_img)

    #SQL link for login page
    def try_login():
        chk_cmmd=wire.execute('Select USERNAME,PASSWORD from USER where
USERNAME="{}" and PASSWORD
="{}"'.format(username_entrybox.get().upper(),password_entrybox.get().upper()))
        fetch_cmmd=wire.fetchall()
        rows=0
        for i in fetch_cmmd:
            rows+=1
        if rows>0:
            messagebox.showinfo('Touch  Bistro','You have successfully logged in !!',icon='info')
            Home()
        else:
            messagebox.showinfo('Touch  Bistro','The required details do not match. Please try again or sign
up for a new id.',icon='info')

    #Inserting login image
    loginbtn_img=PhotoImage(file="l6.png")
    loginbtn= Button(login,image=loginbtn_img,command=try_login,bd=0)
    C1.create_window(850,450,anchor=NW,window=loginbtn)

    #Inserting the heading
    loginName=C1.create_text(940,60,text='Welcome.....', font=('Dancing Script',60),fill='#610B21')

    # call function when we click on username entry box
    def click(*args):
        username_entrybox.config(fg='brown')
        username_entrybox.delete(0, 'end')

    # call function when we leave username entry box
```

```
    def leave(*args):
        if username_entrybox.get()==="":
            username_entrybox.config(fg='grey')
            username_entrybox.delete(0, 'end')
            username_entrybox.insert(0, 'Username ')
            login.focus()

    # Add username Entry Box
    username_entrybox = Entry(login, width=15,bg='white',bd=0,font=('PTSans-Regular',16),fg='grey')
    username_entrybox.insert(0, 'Username')
    C1.create_window(950,250,window=username_entrybox)

    # Use bind method for username entry
    username_entrybox.bind("<Button-1>", click)
    username_entrybox.bind("<Leave>", leave)

    # call function when we click on password entry box
    def Click(*args):
            password_entrybox.config(fg='#8A4B08')
            password_entrybox.delete(0, 'end')

    # call function when we leave password entry box
    def Leave(*args):
        if password_entrybox.get()==="":
            password_entrybox.config(fg='grey')
            password_entrybox.delete(0, 'end')
            password_entrybox.insert(0, 'Password')
            login.focus()

    # Add password Entry Box
    password_entrybox = Entry(login, width=15,bg='white',bd=0,font=('PTSans-Regular',16),fg='grey')
    password_entrybox.insert(0,'Password')
    C1.create_window(950,340,window=password_entrybox)

    # Use bind method for password entry
    password_entrybox.bind("<Button-1>", Click)
    password_entrybox.bind("<Leave>", Leave)

    #Creating checkbox for show/hiding password
    def Stars():
        if (var.get()==1):
            password_entrybox.config(show='*')
        elif (var.get()==0):
            password_entrybox.config(show="")
    var=IntVar()
    CheckBox = Checkbutton(login, text="Hide
password",command=Stars,bg='white',offvalue=0,onvalue=1,variable=var)
    C1.create_window(920,390,window=CheckBox)

    #Creating button for sign in
    C1.create_text(1000,571,text="Don't have an account yet?")
```

```
    signup_btn=Button(login,text='Sign up here',bd=0,bg='white',fg='blue',command=Signup)
    C1.create_window(1110,572,window=signup_btn)


    #Looping the login window
    login.mainloop()
'.....................................................................................................................................
.....................................................................................................................'
#Creating Signup page
signup=None
def Signup():

    #Checking if other windows are open or not
    if (login is not None) and login.winfo_exists():
        login.withdraw()
    if (home is not None) and home.winfo_exists():
        home.withdraw()
    if (menu is not None) and menu.winfo_exists():
        menu.withdraw()
    if (billing is not None) and billing.winfo_exists():
        billing.withdraw()
    if (employee_details is not None) and employee_details.winfo_exists():
         employee_details.withdraw()

    #Size,icon and title of  signup page
    global signup
    signup=Toplevel()
    window_width = 1200
    window_height = 600
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    position_top = int(screen_height/2 -window_height/2)
    position_right = int(screen_width / 2 - window_width/2)
    signup.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')
    signup.title('Touch Bisto - Sign up')
    signup.iconbitmap("latte.ico")
    signup.resizable(False,False)



    #Connecting signup details to sql
    def try_signup():
        if len(password_entrybox.get())>=8:
            if password_entrybox.get().isalnum()==True:
                    wire.execute('INSERT INTO user
values("{}","{}","{}","{}")'.format(username_entrybox.get(),email_entrybox.get(),contactnum_entrybo
x.get(),password_entrybox.get()))
                    messagebox.showinfo('Success',"You've successfully registered.")
                    signup.withdraw()
                    Home()
            else:
```

```
            messagebox.showinfo('Touch Bistro','Please enter another password.')
    else:
        messagebox.showinfo('Touch Bistro','Please enter another password.')

#Creating the canvas for signup page
C2 = Canvas(signup,width=1200 ,height=600,borderwidth=0,bg='white')
C2.pack(expand=YES, fill=BOTH)

coffee_img=PhotoImage(file="k13.png")
C2.create_image(0,0,anchor=NW,image=coffee_img)
kk=PhotoImage(file="jhbnm1.png")
C2.create_image(600,0,anchor=NW,image=kk)

#Heading for the sign up page
signupName=C2.create_text(900,60,text='Sign Up...', font=('Dancing Script',55),fill='green')

#Creating entry boxes


# call function when we click on username entry box
def CLick(*args):
    username_entrybox.config(fg='green')
    username_entrybox.delete(0, 'end')

# call function when we leave username entry box
def LEave(*args):
    if username_entrybox.get()==="":
        username_entrybox.config(fg='grey')
        username_entrybox.delete(0, 'end')
        username_entrybox.insert(0, 'Username ')
        signup.focus()

# Add username Entry Box
username_entrybox = Entry(signup, width=25,bg='white',bd=0,font=('PTSans-Regular',16),fg='grey')
username_entrybox.insert(0, 'Username')
C2.create_window(875,162,window=username_entrybox)

# Use bind method for username entry
username_entrybox.bind("<Button-1>", CLick)
username_entrybox.bind("<Leave>", LEave)

# call function when we click on email entry box
def CLIck(*args):
        email_entrybox.config(fg='green')
        email_entrybox.delete(0, 'end')

# call function when we leave email entry box
def LEAve(*args):
    if email_entrybox.get()==="":
        email_entrybox.config(fg='grey')
        email_entrybox.delete(0, 'end')
```

```
        email_entrybox.insert(0, 'E - Mail')
        signup.focus()

    # Add email Entry Box
    email_entrybox = Entry(signup, width=25,bg='white',bd=0,font=('PTSans-Regular',16),fg='grey')
    email_entrybox.insert(0,'E - mail')
    C2.create_window(875,238,window=email_entrybox)

    # Use bind method for email entry
    email_entrybox.bind("<Button-1>", CLIck)
    email_entrybox.bind("<Leave>", LEAve)

    # call function when we click on contact number entry box
    def CLICk(*args):
        contactnum_entrybox.config(fg='green')
        contactnum_entrybox.delete(0, 'end')

    # call function when we leave contact number entry box
    def LEAVe(*args):
      if contactnum_entrybox.get()==="":
        contactnum_entrybox.config(fg='grey')
        contactnum_entrybox.delete(0, 'end')
        contactnum_entrybox.insert(0, 'Contact Number')
        signup.focus()

    # Add contact number Entry Box
    contactnum_entrybox = Entry(signup, width=25,bg='white',bd=0,font=('PTSans-
Regular',16),fg='grey')
    contactnum_entrybox.insert(0,'Contact Number')
    C2.create_window(875,313,window=contactnum_entrybox)

    # Use bind method for contact number entry
    contactnum_entrybox.bind("<Button-1>", CLICk)
    contactnum_entrybox.bind("<Leave>", LEAVe)

    # call function when we click on password entry box
    def CLICK(*args):
        password_entrybox.config(fg='green')
        password_entrybox.delete(0, 'end')

    # call function when we leave password entry box
    def LEAVE(*args):
      if password_entrybox.get()==="":
        password_entrybox.config(fg='grey')
        password_entrybox.delete(0, 'end')
        password_entrybox.insert(0, 'Password')
        signup.focus()

    # Add password Entry Box
    password_entrybox = Entry(signup, width=25,bg='white',bd=0,font=('PTSans-Regular',16),fg='grey')
    password_entrybox.insert(0,'Password')
```

```
    C2.create_window(875,392,window=password_entrybox)

    # Use bind method for password entry
    password_entrybox.bind("<Button-1>", CLICK)
    password_entrybox.bind("<Leave>", LEAVE)

    #Creating checkbox for show/hiding password
    def STars():
        if (vAr.get()==1):
            password_entrybox.config(show='*')
        elif (vAr.get()==0):
            password_entrybox.config(show="")
    vAr=IntVar()
    CHeckBox = Checkbutton(signup,
text="",command=STars,bg='white',offvalue=0,onvalue=1,variable=vAr)
    C2.create_window(1100,390,window=CHeckBox)
    #Creating the signup button
    signup_img=PhotoImage (file='signin_button.png')
    signup_btn=Button(signup,image=signup_img,bd=0,command=try_signup)
    C2.create_window(750,450,anchor=NW,window=signup_btn)
    #Looping the signup window
    signup.mainloop()
'.....
.....'
#Creating home page
home=None
def Home():
    #Checking if other windows are open or not
    if (login is not None) and login.winfo_exists():
        login.withdraw()
    if (signup is not None) and signup.winfo_exists():
        signup.withdraw()
    if (menu is not None) and menu.winfo_exists():
        menu.withdraw()
    if (billing is not None) and billing.winfo_exists():
        billing.withdraw()
    if (employee_details is not None) and employee_details.winfo_exists():
        employee_details.withdraw()

    #Size,icon and title of home page
    global home
    home=Toplevel()
    W, H= home.winfo_screenwidth(), home.winfo_screenheight()
    home.geometry("%dx%d+0+0" % (W, H))
    home.title('Touch Bistro- Home')
    home.iconbitmap("latte.ico")
    home.resizable(True,True)

    #Creating scrollbar for canvas
    vertScrollbar = Scrollbar(home, orient='vertical')
    vertScrollbar.pack(side='right', fill='y')
```

```
    #Creating canvas for home page
    C3 = Canvas(home,width=W ,height=H,borderwidth=0,scrollregion=(0, 0, 1360, 1125),
yscrollcommand=vertScrollbar.set,bg='white')
    vertScrollbar.config(command=C3.yview)
    C3.pack(expand=YES, fill=BOTH)
    #Adding background
    home_bg=PhotoImage(file='MY Post.png')
    C3.create_image(0,0,anchor=NW,image=home_bg)
    #Adding buttons for navigation
    menu_img=PhotoImage(file='menu_btn1.png')
    menu_btn=Button(home,image=menu_img,bd=0,bg='#4E4B48',command=Menu)
    C3.create_window(422,310,anchor=NW,window=menu_btn)
    employee_details_img=PhotoImage(file='employee_details_img.png')

employee_details_btn=Button(home,image=employee_details_img,bd=0,bg='#4E4B48',command=Empl
oyee_details)
    C3.create_window(422,420,anchor=NW,window=employee_details_btn)
    #Heading
    C3.create_text(700,790,text='About Us',font=('Dancing Script',60),fill='#610B21')
    #Inserting image
    cafeimg=PhotoImage(file='lk.png')
    C3.create_image(0,870,anchor=NW,image=cafeimg)
    #Inserting the text
    abtus_text=Text(home,font=("PT Sans-
Regular",13),height=12,width=85,fg="#986216",bd=0,bg="white",wrap=WORD)
    C3.create_window(900,1000,window=abtus_text)
    abtus_text.insert(tk.END,'Touch Bistro started out as a small upscale breakfast and lunch spot. Our
dream was to bring together people with great food and gourmet coffees. A place where groups could
come and meet, business meetings could be held and friends could get together and linger over laughter
and long conversations. A place that felt as comfortable as home, with exceptional food. We started out
in a small cafe coffee house style restaurant and quickly outgrew our last building. It was perfect. A
large elegant dining area with a small cafe style room in the back. A kitchen where we could be creative
and expand our menu, and that we did.')
    abtus_text.tag_configure("center", justify='left')
    abtus_text.tag_add("left", 1.0, "end")
    #Looping the home page
    home.mainloop()
'.....
.....'
'.....
.....'
    #Creating values
snacks_values=['Cheese burger','Beef Burger','Creamy Mushroom','Taco']
sides_values=['French Fries','Mozarella Stick','Garlic Potato','Hummus']
beverages_values=['Ristretto','Expresso','Caffe Macchiato','Cappucino','Black Coffee','Cold
Coffee','South Indian Filter Coffee','Iced Tea']
milkshake_values=['Chocolate Milkshake','Vannila Milkshake','Strawberry Milkshake,','Caramel
Milkshake']
code_values={'A1':'Cheese burger','A2':'Beef Burger','A3':'Creamy Mushroom','A4':'Taco',
             'B1':'French Fries','B2':'Mozarella Stick','B3':'Garlic Potato','B4':'Hummus',
```

```
            'C1':'Ristretto','C2':'Expresso','C3':'Caffe Macchiato','C4':'Cappucino','C5':'Black 
Coffee','C6':'Cold Coffee','C7':'South Indian Filter Coffee','C8':'Iced Tea',
            'D1':'Chocolate Milkshake','D2':'Vannila Milkshake','D3':'Strawberry 
Milkshake','D4':'Caramel Milkshake'}
cost_values={'Cheese burger':'120','Beef Burger':'135','Creamy Mushroom':'130','Taco':'125',
            'French Fries':'120','Mozarella Stick':'100','Garlic Potato':'140','Hummus':'110',
            'Ristretto':'145','Expresso':'140','Caffe Macchiato':'110','Cappucino':'150','Black 
Coffee':'100','Cold Coffee':'150','South Indian Filter Coffee':'120','Iced Tea':'130',
            'Chocolate Milkshake':'110','Vannila Milkshake':'110','Strawberry Milkshake':'110','Caramel 
Milkshake':'110'}
'..................................................................................................................................................
...............................................................................................................................................'
#Creating the menu screen
menu=None
def Menu():

   #Checking if other windows are open or not
   if (login is not None) and login.winfo_exists():
      login.withdraw()
   if (signup is not None) and signup.winfo_exists():
      signup.withdraw()
   if (home is not None) and home.winfo_exists():
      home.withdraw()
   if (billing is not None) and billing.winfo_exists():
      billing.withdraw()
   if (employee_details is not None) and employee_details.winfo_exists():
      employee_details.withdraw()

   #Size,icon and title of home page
   global menu
   menu=Toplevel()
   W, H= menu.winfo_screenwidth(), menu.winfo_screenheight()
   window_width = 1200
   window_height = 675
   screen_width = root.winfo_screenwidth()
   screen_height = root.winfo_screenheight()
   position_top = int(screen_height/2 -window_height/2)
   position_right = int(screen_width / 2 - window_width/2)
   menu.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')
   menu.title('Touch Bistro- Menu')
   menu.iconbitmap("latte.ico")
   menu.resizable(False,False)

   #Creating canvas for employee details page
   C5=Canvas(menu,width=1200,height=675,borderwidth=0,bg='white')
   C5.pack(expand=YES, fill=BOTH)

   #Adding menu image
   menu_img=PhotoImage(file='menu.png')
   C5.create_image(0,0,anchor=NW,image=menu_img)
```

```
    #Looping the menu page
    menu.mainloop()
'.....................................................................................................................................
.....................................................................................................................................'
#Creating the billing screen
billing=None
def Billing():

    #Checking if other windows are open or not
    if (login is not None) and login.winfo_exists():
        login.withdraw()
    if (signup is not None) and signup.winfo_exists():
        signup.withdraw()
    if (home is not None) and home.winfo_exists():
        home.withdraw()
    if (employee_details is not None) and employee_details.winfo_exists():
        employee_details.withdraw()

    #Variables
    menu_category = ["Beverages","Snacks","Sides","Milkshakes"]
    menu_category_dict = {"Beverages":"1 Beverages.txt","Snacks":"2 Snacks.txt","Sides":"3
Sides.txt","Milkshakes":"4 Milkshakes.txt"}

    order_dict = {}
    for i in menu_category:
        order_dict[i] = {}

    os.chdir(os.path.dirname(os.path.abspath(__file__)))

#Backend Functions
    def load_menu():
        menuCategory.set("")
        menu_table.delete(*menu_table.get_children())
        menu_file_list = os.listdir("Menu")
        for file in menu_file_list:
            f = open("Menu\\" + file , "r")
            category=""
            while True:
                line = f.readline()
                if(line=""):
                    menu_table.insert(",END,values=["","",""])
                    break
                elif (line=="\n"):
                    continue
                elif(line[0]=='#'):
                    category = line[1:-1]
                    name = "\t\t"+line[:-1]
                    price = ""
                elif(line[0]=='*'):
                    name = line[:-1]
                    price = ""
```

```
            else:
                name = line[:line.rfind(" ")]
                price = line[line.rfind(" ")+1:-3]

            menu_table.insert(",END,values=[name,price,category])
    #menu_table.insert(",END,values=["Masala Dosa","50"])
def load_order():
    order_table.delete(*order_table.get_children())
    for category in order_dict.keys():
        if order_dict[category]:
            for lis in order_dict[category].values():
                order_table.insert(",END,values=lis)
            update_total_price()

def add_button_operation():
    name = itemName.get()
    rate = itemRate.get()
    category = itemCategory.get()
    quantity = itemQuantity.get()

if name in order_dict[category].keys():
    tmsg.showinfo("Error", "Item already exist in your order")
    return
if not quantity.isdigit():
    tmsg.showinfo("Error", "Please Enter Valid Quantity")
    return
lis = [name,rate,quantity, str(int(rate)*int(quantity)),category]
order_dict[category][name] = lis
load_order()

def load_item_from_menu(event):
    cursor_row = menu_table.focus()
    contents = menu_table.item(cursor_row)
    row = contents["values"]

    itemName.set(row[0])
    itemRate.set(row[1])
    itemCategory.set(row[2])
    itemQuantity.set("1")

def load_item_from_order(event):
    cursor_row = order_table.focus()
    contents = order_table.item(cursor_row)
    row = contents["values"]

    itemName.set(row[0])
    itemRate.set(row[1])
    itemQuantity.set(row[2])
    itemCategory.set(row[4])

def show_button_operation():
```

```
    category = menuCategory.get()
    if category not in menu_category:
        tmsg.showinfo("Error", "Please select valid Choice")
    else:
        menu_table.delete(*menu_table.get_children())
        f = open("Menu"// + menu_category_dict[category] , "r")
        while True:
            line = f.readline()
            if(line=""):
                break
            if (line[0]=='#' or line=="\n"):
                continue
            if(line[0]=='*'):
                name = "\t"+line[:-1]
                menu_table.insert(",END,values=[name,"","""])
            else:
                name = line[:line.rfind(" ")]
                price = line[line.rfind(" ")+1:-3]
                menu_table.insert(",END,values=[name,price,category])

def clear_button_operation():
    itemName.set("")
    itemRate.set("")
    itemQuantity.set("")
    itemCategory.set("")

def cancel_button_operation():
    names = []
    for i in menu_category:
        names.extend(list(order_dict[i].keys()))

    if len(names)==0:
        tmsg.showinfo("Error", "Your order list is Empty")
        return
    ans = tmsg.askquestion("Cancel Order", "Are You Sure to Cancel Order?")
    if ans=="no":
        return
    order_table.delete(*order_table.get_children())
    for i in menu_category:
        order_dict[i] = {}
    clear_button_operation()
    update_total_price()

def update_button_operation():
    name = itemName.get()
    rate = itemRate.get()
    category = itemCategory.get()
    quantity = itemQuantity.get()

    if category==="":
        return
```

```
        if name not in order_dict[category].keys():
            tmsg.showinfo("Error", "Item is not in your order list")
            return
        if order_dict[category][name][2]==quantity:
            tmsg.showinfo("Error", "No changes in Quantity")
            return
        order_dict[category][name][2] = quantity
        order_dict[category][name][3] = str(int(rate)*int(quantity))
        load_order()

def remove_button_operation():
    name = itemName.get()
    category = itemCategory.get()

    if category=="":
        return
    if name not in order_dict[category].keys():
        tmsg.showinfo("Error", "Item is not in your order list")
        return
    del order_dict[category][name]
    load_order()

def update_total_price():
    price = 0
    for i in menu_category:
        for j in order_dict[i].keys():
            price += int(order_dict[i][j][3])
    if price == 0:
        totalPrice.set("")
    else:
        totalPrice.set("Rs. "+str(price)+"  /-")

def bill_button_operation():
    customer_name = customerName.get()
    customer_contact = customerContact.get()
    names = []
    for i in menu_category:
        names.extend(list(order_dict[i].keys()))
    if len(names)==0:
        tmsg.showinfo("Error", "Your order list is Empty")
        return
    if customer_name=="" or customer_contact==="":
        tmsg.showinfo("Error", "Customer Details Required")
        return
    if not customerContact.get().isdigit():
        tmsg.showinfo("Error", "Invalid Customer Contact")
        return
    ans = tmsg.askquestion("Generate Bill", "Are You Sure to Generate Bill?")
    ans = "yes"
    if ans=="yes":
        bill = Toplevel()
```

```
    bill.title("Bill")
    bill.geometry("670x500+300+100")
    bill.wm_iconbitmap("Coffee.ico")
    bill_text_area = Text(bill, font=("times new roman", 12))
    st = "\t\t\t\tTouch Bistro\n\t\t\tVadavalli,Coimbatore-641041\n"
    st += "\t\t\tGST.NO:- 27AHXPP3379HIZH\n"
    st += "-"*61 + "BILL" + "-"*61 + "\nDate:- "

    #Date and time
    t = time.localtime(time.time())
    week_day_dict =
{0:"Monday",1:"Tuesday",2:"Wednesday",3:"Thursday",4:"Friday",5:"Saturday",6:"Sunday"}
    st += f"{t.tm_mday} / {t.tm_mon} / {t.tm_year} ({week_day_dict[t.tm_wday]})"
    st += " "*10 + f"\t\t\t\t\t\tTime:- {t.tm_hour} : {t.tm_min} : {t.tm_sec}"

    #Customer Name & Contact
    st += f"\nCustomer Name:- {customer_name}\nCustomer Contact:- {customer_contact}\n"
    st += "-"*130 + "\n" + " "*4 + "DESCRIPTION\t\t\t\t\tRATE\tQUANTITY\t\tAMOUNT\n"
    st += "-"*130 + "\n"

    #List of Items
    for i in menu_category:
        for j in order_dict[i].keys():
            lis = order_dict[i][j]
            name = lis[0]
            rate = lis[1]
            quantity = lis[2]
            price = lis[3]
            st += name + "\t\t\t\t\t" + rate + "\t      " + quantity + "\t\t  " + price + "\n\n"
    st += "-"*130

    #Total Price
    st += f"\n\t\t\tTotal price : {totalPrice.get()}\n"
    st += "-"*130

    #display bill in new window
    bill_text_area.insert(1.0, st)

    #write into file
    folder = f"{t.tm_mday},{t.tm_mon},{t.tm_year}"
    if not os.path.exists(f"Bill Records\\{folder}"):
        os.makedirs(f"Bill Records\\{folder}")
    file = open(f"Bill Records\\{folder}\\{customer_name+customer_contact}.txt", "w")
    file.write(st)
    file.close()

    #Clear operaitons
    order_table.delete(*order_table.get_children())
    for i in menu_category:
        order_dict[i] = {}
    clear_button_operation()
```

```
        update_total_price()
        customerName.set("")
        customerContact.set("")

        bill_text_area.pack(expand=True, fill=BOTH)
        bill.focus_set()
        bill.protocol("WM_DELETE_WINDOW", close_window)

    def close_window():
        tmsg.showinfo("Thanks", "Thanks for using our service")
        billing.destroy()

    #Frontend functions
    billing = Topview()
    w, h = billing.winfo_screenwidth(), billing.winfo_screenheight()
    billing.geometry("%dx%d+0+0" % (w, h))
    billing.title("Touch Bistro")
    billing.wm_iconbitmap("Burger.ico")
    billing.config(bg="#704F32")
    billing.wm_maxsize()

    #Title
    style_button = ttk.Style()
    style_button.configure("TButton",font = ("times new roman",10,"bold"),
   background="#704F32")
    #735C36

    #Customer
    customer_frame = LabelFrame(billing,text="Customer Details",font=("times new roman", 15,
"bold"),bd=8, bg="#704F32", relief=FLAT,fg='white')
    customer_frame.pack(side=TOP, fill="x")

    customer_name_label = Label(customer_frame, text="Name", font=("times new roman", 15,
"bold"),bg = "#704F32", fg="white")
    customer_name_label.grid(row = 0, column = 0)

    customerName = StringVar()
    customerName.set("")
    customer_name_entry = Entry(customer_frame,width=20,font=("times new roman",
15),bd=5,textvariable=customerName,relief=FLAT)
    customer_name_entry.grid(row = 0, column=1,padx=50)

    customer_contact_label = Label(customer_frame, text="Contact", font=("times new roman", 15,
"bold"),bg = "#704F32", fg="white")
    customer_contact_label.grid(row = 0, column = 2)

    customerContact = StringVar()
    customerContact.set("")
    customer_contact_entry = Entry(customer_frame,width=20,font=("times new roman
",15),bd=5,textvariable=customerContact,relief=FLAT)
    customer_contact_entry.grid(row = 0, column=3,padx=50)
```

```
  #Menu
   menu_frame = Frame(billing,bd=8, bg="#704F32", relief=FLAT)
   menu_frame.place(x=0,y=125,height=585,width=680)

   menu_label = Label(menu_frame, text="Menu", font=("times new roman", 20, "bold"),bg =
"#704F32", fg="white", pady=0)
   menu_label.pack(side=TOP,fill="x")

   menu_category_frame = Frame(menu_frame,bg="#704F32",pady=10)
   menu_category_frame.pack(fill="x")

   combo_lable = Label(menu_category_frame,text="Select Type", font=("times new roman", 12,
"bold"),bg = "#704F32", fg="white")
   combo_lable.grid(row=0,column=0,padx=10)

   menuCategory = StringVar()
   combo_menu =
ttk.Combobox(menu_category_frame,values=menu_category,textvariable=menuCategory)
   combo_menu.grid(row=0,column=1,padx=30)

   show_button = ttk.Button(menu_category_frame,
text="Show",width=10,command=show_button_operation)
   show_button.grid(row=0,column=2,padx=60)

   show_all_button = ttk.Button(menu_category_frame, text="Show
All",width=10,command=load_menu)
   show_all_button.grid(row=0,column=3)

   # Menu Table
   menu_table_frame = Frame(menu_frame)
   menu_table_frame.pack(fill=BOTH,expand=1)

   scrollbar_menu_x = Scrollbar(menu_table_frame,orient=HORIZONTAL)
   scrollbar_menu_y = Scrollbar(menu_table_frame,orient=VERTICAL)

   style = ttk.Style()
   style.configure("Treeview.Heading",font=("times new roman",13, "bold"))
   style.configure("Treeview",font=("times new roman",12),rowheight=25)

   menu_table = ttk.Treeview(menu_table_frame,style = "Treeview",
         columns =("name","price","category"),xscrollcommand=scrollbar_menu_x.set,
         yscrollcommand=scrollbar_menu_y.set)

   menu_table.heading("name",text="Name")
   menu_table.heading("price",text="Price")
   menu_table["displaycolumns"]=("name", "price")
   menu_table["show"] = "headings"
   menu_table.column("price",width=50,anchor='center')

   scrollbar_menu_x.pack(side=BOTTOM,fill=X)
```

```
    scrollbar_menu_y.pack(side=RIGHT,fill=Y)

    scrollbar_menu_x.configure(command=menu_table.xview)
    scrollbar_menu_y.configure(command=menu_table.yview)

    menu_table.pack(fill=BOTH,expand=1)

    load_menu()
    menu_table.bind("<ButtonRelease-1>",load_item_from_menu)

    #Item Frame
    item_frame = Frame(billing,bd=8, bg="#704F32", relief=FLAT)


    item_frame.place(x=680,y=125,height=230,width=680)

    item_title_label = Label(item_frame, text="Item", font=("times new roman", 20, "bold"),bg =
"#704F32", fg="white")
    item_title_label.pack(side=TOP,fill="x")

    item_frame2 = Frame(item_frame, bg="#704F32")
    item_frame2.pack(fill=X)

    item_name_label = Label(item_frame2, text="Name", font=("times new roman", 12, "bold"),bg =
"#704F32", fg="white")
    item_name_label.grid(row=0,column=0)

    itemCategory = StringVar()
    itemCategory.set("")

    itemName = StringVar()
    itemName.set("")
    item_name = Entry(item_frame2, font=("times new roman
",12),textvariable=itemName,state=DISABLED, width=25)
    item_name.grid(row=0,column=1,padx=10)

    item_rate_label = Label(item_frame2, text="Rate", font=("times new roman", 12, "bold"),bg =
"#704F32", fg="white")
    item_rate_label.grid(row=0,column=2,padx=40)

    itemRate = StringVar()
    itemRate.set("")
    item_rate = Entry(item_frame2, font=("times new roman
",12),textvariable=itemRate,state=DISABLED, width=10)
    item_rate.grid(row=0,column=3,padx=10)

    item_quantity_label = Label(item_frame2, text="Quantity", font=("times new roman", 12, "bold"),bg
= "#704F32", fg="white")
    item_quantity_label.grid(row=1,column=0,padx=30,pady=15)

    itemQuantity = StringVar()
```

```
    itemQuantity.set("")
    item_quantity = Entry(item_frame2, font=("times new roman",12),textvariable=itemQuantity,
width=10)
    item_quantity.grid(row=1,column=1)

    item_frame3 = Frame(item_frame, bg="#704F32")
    item_frame3.pack(fill=X)

    add_button = ttk.Button(item_frame3, text="Add Item",command=add_button_operation)
    add_button.grid(row=0,column=0,padx=40,pady=30)

    remove_button = ttk.Button(item_frame3, text="Remove Item",command=remove_button_operation)
    remove_button.grid(row=0,column=1,padx=40,pady=30)

    update_button = ttk.Button(item_frame3, text="Update
Quantity",command=update_button_operation)
    update_button.grid(row=0,column=2,padx=40,pady=30)

    clear_button = ttk.Button(item_frame3, text="Clear",width=8,command=clear_button_operation)
    clear_button.grid(row=0,column=3,padx=40,pady=30)

    #Order Frame
    order_frame = Frame(billing,bd=8, bg="#704F32", relief=FLAT)
    order_frame.place(x=680,y=335,height=370,width=680)

    order_title_label = Label(order_frame, text="Your Order", font=("times new roman", 20, "bold"),bg =
"#704F32", fg="white")
    order_title_label.pack(side=TOP,fill="x")

    # Order Table
    order_table_frame = Frame(order_frame)
    order_table_frame.place(x=0,y=40,height=260,width=680)

    scrollbar_order_x = Scrollbar(order_table_frame,orient=HORIZONTAL)
    scrollbar_order_y = Scrollbar(order_table_frame,orient=VERTICAL)

    order_table = ttk.Treeview(order_table_frame,
        columns =("name","rate","quantity","price","category"),xscrollcommand=scrollbar_order_x.set,
        yscrollcommand=scrollbar_order_y.set)

    order_table.heading("name",text="Name")
    order_table.heading("rate",text="Rate")
    order_table.heading("quantity",text="Quantity")
    order_table.heading("price",text="Price")
    order_table["displaycolumns"]=("name", "rate","quantity","price")
    order_table["show"] = "headings"
    order_table.column("rate",width=100,anchor='center', stretch=NO)
    order_table.column("quantity",width=100,anchor='center', stretch=NO)
    order_table.column("price",width=100,anchor='center', stretch=NO)

    order_table.bind("<ButtonRelease-1>",load_item_from_order)
```

```
    scrollbar_order_x.pack(side=BOTTOM,fill=X)
    scrollbar_order_y.pack(side=RIGHT,fill=Y)

    scrollbar_order_x.configure(command=order_table.xview)
    scrollbar_order_y.configure(command=order_table.yview)

    order_table.pack(fill=BOTH,expand=1)

    total_price_label = Label(order_frame, text="Total Price",font=("times new roman", 12, "bold"),bg =
"#704F32", fg="white")
    total_price_label.pack(side=LEFT,anchor=SW,padx=20,pady=10)

    totalPrice = StringVar()
    totalPrice.set("")
    total_price_entry = Entry(order_frame, font=("times new roman",
12),textvariable=totalPrice,state=DISABLED, width=10)
    total_price_entry.pack(side=LEFT,anchor=SW,padx=0,pady=10)

    bill_button = ttk.Button(order_frame, text="Generate
Bill",width=12,command=bill_button_operation)
    bill_button.pack(side=LEFT,anchor=SW,padx=80,pady=10)

    cancel_button = ttk.Button(order_frame, text="Cancel Order",command=cancel_button_operation)
    cancel_button.pack(side=LEFT,anchor=SW,padx=20,pady=10)

    billing.mainloop()
'.....
.....'
#Creating the employee details screen
employee_details=None
def Employee_details():

    #Checking if other windows are open or not
    if (login is not None) and login.winfo_exists():
        login.withdraw()
    if (signup is not None) and signup.winfo_exists():
        signup.withdraw()
    if (home is not None) and home.winfo_exists():
        home.withdraw()

    #Size,icon and title of employee details page
    global employee_details
    employee_details=Toplevel()

    employee_details.title("Employee Management")
    W, H= employee_details.winfo_screenwidth(), employee_details.winfo_screenheight()
    employee_details.geometry("%dx%d+0+0" % (W, H))
    employee_details.config(bg="#2c3e50")
```

```
    name = StringVar()
    designation = StringVar()
    dob = StringVar()
    gender = StringVar()
    email = StringVar()
    contact = StringVar()

    entries_frame = Frame(employee_details, bg="#535c68")
    entries_frame.pack(side=TOP, fill=X)
    title = Label(entries_frame, text="Employee Management", font=("Times New Roman", 20, "bold"),
bg="#535c68", fg="white")
    title.grid(row=0, columnspan=2, padx=10, pady=20, sticky="w")

    lblName = Label(entries_frame, text="Name", font=("Times New Roman", 16), bg="#535c68",
fg="white")
    lblName.grid(row=1, column=0, padx=10, pady=10, sticky="w")
    txtName = Entry(entries_frame, textvariable=name, font=("Times New Roman", 16), width=30)
    txtName.grid(row=1, column=1, padx=10, pady=10, sticky="w")

    lbldesignation = Label(entries_frame, text="designation", font=("Times New Roman", 16),
bg="#535c68", fg="white")
    lbldesignation.grid(row=1, column=2, padx=10, pady=10, sticky="w")
    txtdesignation = Entry(entries_frame, textvariable=designation, font=("Times New Roman", 16),
width=30)
    txtdesignation.grid(row=1, column=3, padx=10, pady=10, sticky="w")

    lbldob = Label(entries_frame, text="D.O.B", font=("Times New Roman", 16), bg="#535c68",
fg="white")
    lbldob.grid(row=2, column=0, padx=10, pady=10, sticky="w")
    txtDob= Entry(entries_frame, textvariable=dob, font=("Times New Roman", 16), width=30)
    txtDob.grid(row=2, column=1, padx=10, pady=10, sticky="w")

    lblEmail = Label(entries_frame, text="Email", font=("Times New Roman", 16), bg="#535c68",
fg="white")
    lblEmail.grid(row=2, column=2, padx=10, pady=10, sticky="w")
    txtEmail = Entry(entries_frame, textvariable=email, font=("Times New Roman", 16), width=30)
    txtEmail.grid(row=2, column=3, padx=10, pady=10, sticky="w")

    lblGender = Label(entries_frame, text="Gender", font=("Times New Roman", 16), bg="#535c68",
fg="white")
    lblGender.grid(row=3, column=0, padx=10, pady=10, sticky="w")
    comboGender = ttk.Combobox(entries_frame, font=("Times New Roman", 16), width=28,
textvariable=gender, state="readonly")
    comboGender['values'] = ("Male", "F","Non Binary")
    comboGender.grid(row=3, column=1, padx=10, sticky="w")

    lblContact = Label(entries_frame, text="Contact No", font=("Times New Roman", 16),
bg="#535c68", fg="white")
    lblContact.grid(row=3, column=2, padx=10, pady=10, sticky="w")
    txtContact = Entry(entries_frame, textvariable=contact, font=("Times New Roman", 16), width=30)
    txtContact.grid(row=3, column=3, padx=10, sticky="w")
```

```
    def getData(event):
        selected_row = table.focus()
        data = table.item(selected_row)
        global row
        row = data["values"]
        print(row)
        name.set(row[0])
        designation.set(row[1])
        dob.set(row[2])
        email.set(row[4])
        gender.set(row[3])
        contact.set(row[5])

    def displayAll():
        wire.execute('select * from employee')
        result=wire.fetchall()
        for i in result:
            table.insert("", END,values=(i))

    def add_employee():
        if txtName.get() == "" or txtdesignation.get() == "" or txtDob.get() == "" or txtEmail.get() == "" or
comboGender.get() == "" or txtContact.get() == "" :
            messagebox.showerror("Erorr in Input", "Please Fill All the Details")
            return
        for i in table.get_children():
            table.delete(i)
        wire.execute('insert into employee values
("{}","{}","{}","{}","{}",{})'.format(txtName.get(),txtdesignation.get(), txtDob.get()
,comboGender.get(), txtEmail.get() , txtContact.get()))
        pipe.commit()
        messagebox.showinfo("Success", "Record Inserted")
        clearAll()
        displayAll()

    def update_employee():
        if txtName.get() == "" or txtdesignation.get() == "" or txtDob.get() == "" or txtEmail.get() == "" or
comboGender.get() == "" or txtContact.get() == "" :
            messagebox.showerror("Erorr in Input", "Please Fill All the Details")
            return
        for i in table.get_children():
            table.delete(i)
        wire.execute('update employee set
name="{}",designation="{}",DOB="{}",gender="{}",email="{}",contactnum={} where
name="{}"'.format(txtName.get(),txtdesignation.get(), txtDob.get() ,comboGender.get(), txtEmail.get() ,
txtContact.get(),txtName.get()))
        messagebox.showinfo("Success", "Record Updated")
        clearAll()
        displayAll()

    def delete_employee():
```

```
        for i in table.get_children():
            table.delete(i)
        wire.execute('delete from employee where name="{}"'.format(txtName.get()))
        messagebox.showinfo("Success", "Record Deleted")

        clearAll()
        displayAll()

    def clearAll():
        name.set("")
        designation.set("")
        dob.set("")
        gender.set("")
        email.set("")
        contact.set("")

    btn_frame = Frame(entries_frame, bg="#535c68")
    btn_frame.grid(row=6, column=0, columnspan=4, padx=10, pady=10, sticky="w")
    btnAdd = Button(btn_frame, command=add_employee, text="Add Details", width=15, font=("Times
New Roman", 16, "bold"), fg="white",
                bg="#16a085", bd=0).grid(row=0, column=0)
    btnEdit = Button(btn_frame, command=update_employee, text="Update Details", width=15,
font=("Times New Roman", 16, "bold"),
                fg="white", bg="#2980b9",
                bd=0).grid(row=0, column=1, padx=10)
    btnDelete = Button(btn_frame, command=delete_employee, text="Delete Details", width=15,
font=("Times New Roman", 16, "bold"),
                fg="white", bg="#c0392b",
                bd=0).grid(row=0, column=2, padx=10)
    btnClear = Button(btn_frame, command=clearAll, text="Clear Details", width=15, font=("Times New
Roman", 16, "bold"), fg="white",
                bg="#f39c12",
                bd=0).grid(row=0, column=3, padx=10)

    # Table Frame
    tree_frame = Frame(employee_details, bg="#ecf0f1")
    tree_frame.place(x=0, y=290, width=1350, height=520)
    style = ttk.Style()
    style.configure("mystyle.Treeview", font=('Times New Roman', 18), rowheight=50)

    # Modify the font of the body
    style.configure("mystyle.Treeview.Heading", font=('Times New Roman', 12))

    # Modify the font of the headings
    table = ttk.Treeview(tree_frame, columns=(1, 2, 3, 4, 5, 6), style="mystyle.Treeview")
    table.heading("1", text="Name",anchor="w")
    table.column("1", width=10)
    table.heading("2", text="Designation",anchor="w")
    table.column("2", width=10)
    table.heading("3", text="D.O.B",anchor="w")
    table.column("3", width=10)
```

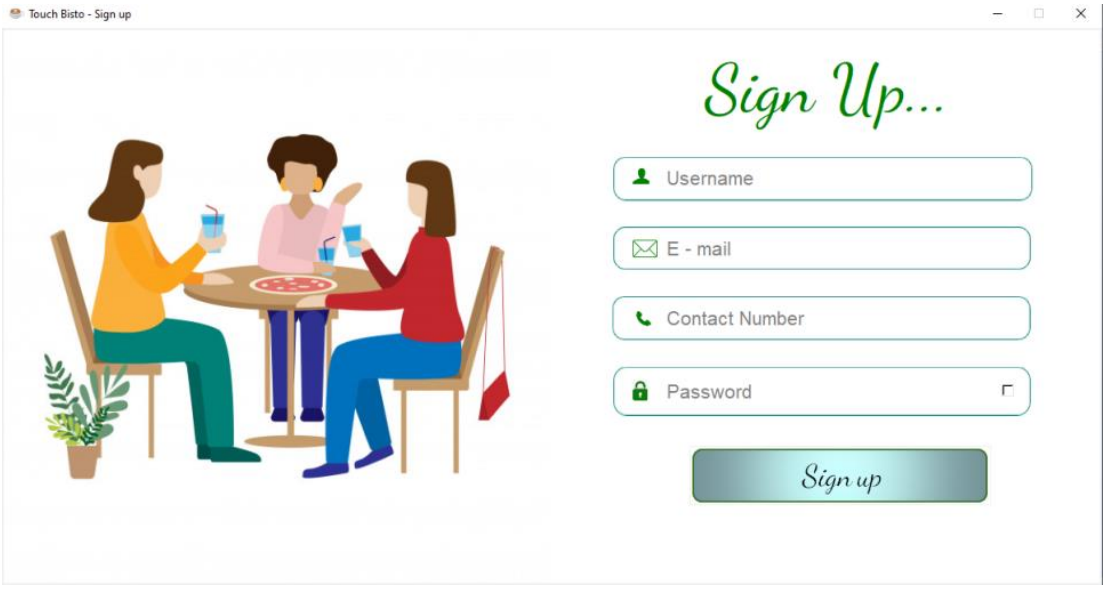
```
    table.heading("4", text="Gender",anchor="w")
    table.column("4", width=1)
    table.heading("6", text="Contact Number",anchor="w")
    table.column("6", width=1)
    table.heading("5", text="Email",anchor="w")
    table.column("5", width=1)
    table['show'] = 'headings'
    table.bind("<ButtonRelease-1>", getData)
    table.pack(fill=X)

    displayAll()

    #Looping the employee details window
    employee_details.mainloop()
'.....
.....'
#Displaying the login window
#Login()
#Signup()
#Home()
#Menu()
Billing()
#Employee_details()
'.....
.....'
#Looping the entire program
root.mainloop()
'.....
.....'
#Closing the connection
pipe.close()
'.....
.....
```

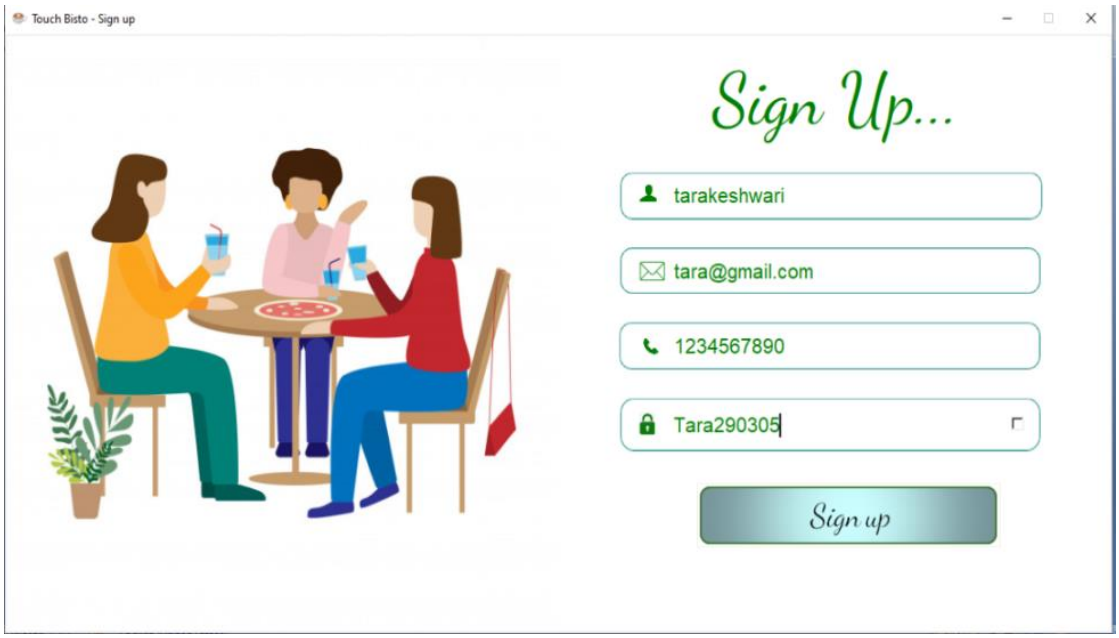
OUTPUT

SIGN UP WINDOW:

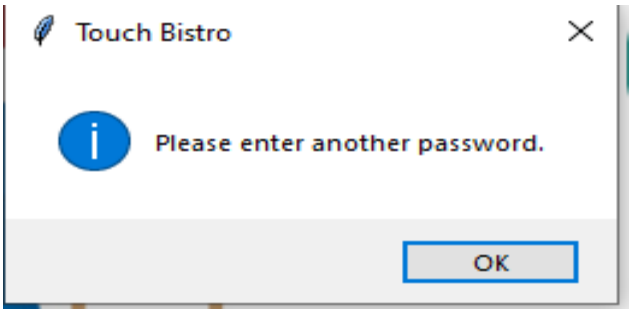


In this window the user will provide their credentials to register themselves.

SIGNUP WINDOW: ENTERING CREDENTIALS

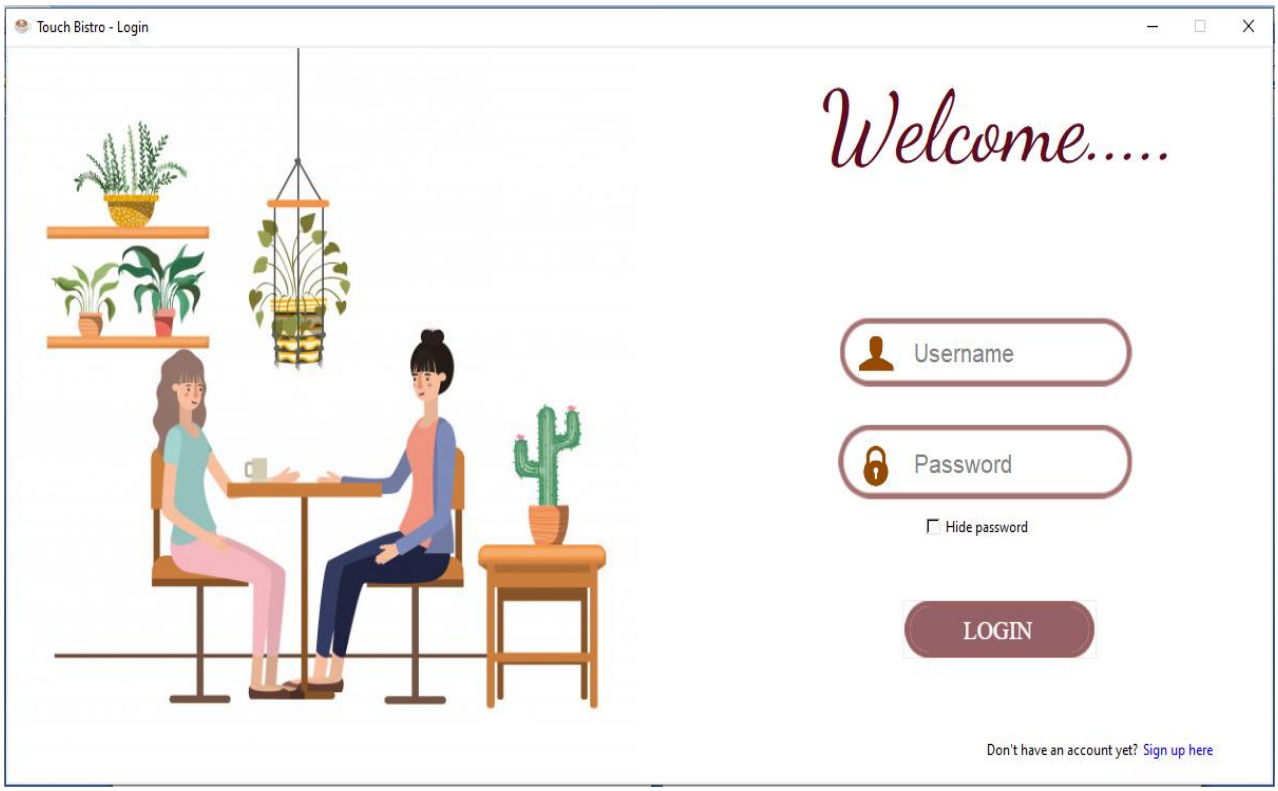


MESSAGE BOX: WRONG PASSWORD ENTRY



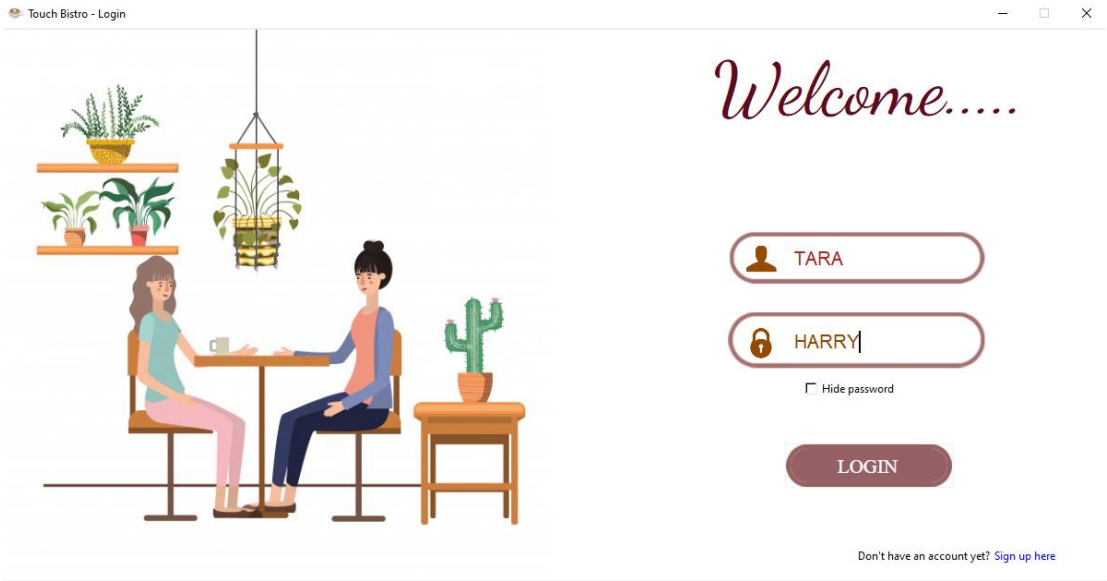
If the password is alphanumeric and contains at least 8 characters then, upon clicking the signup button the user is directed towards the login window, else the above window appears.

LOGIN WINDOW:

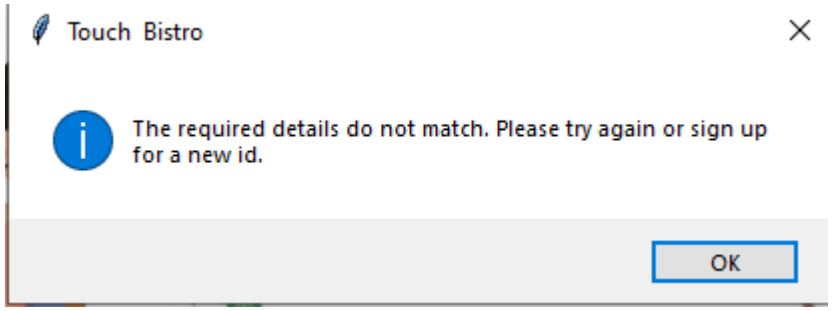


In this window, the user needs to enter the login details like username and password if they are a registered user. Otherwise they need to click on the ‘sign up here’ button to register.

LOGIN WINDOW: ENTERING CREDENTIALS



MESSAGE BOX: WRONG CREDENTIALS

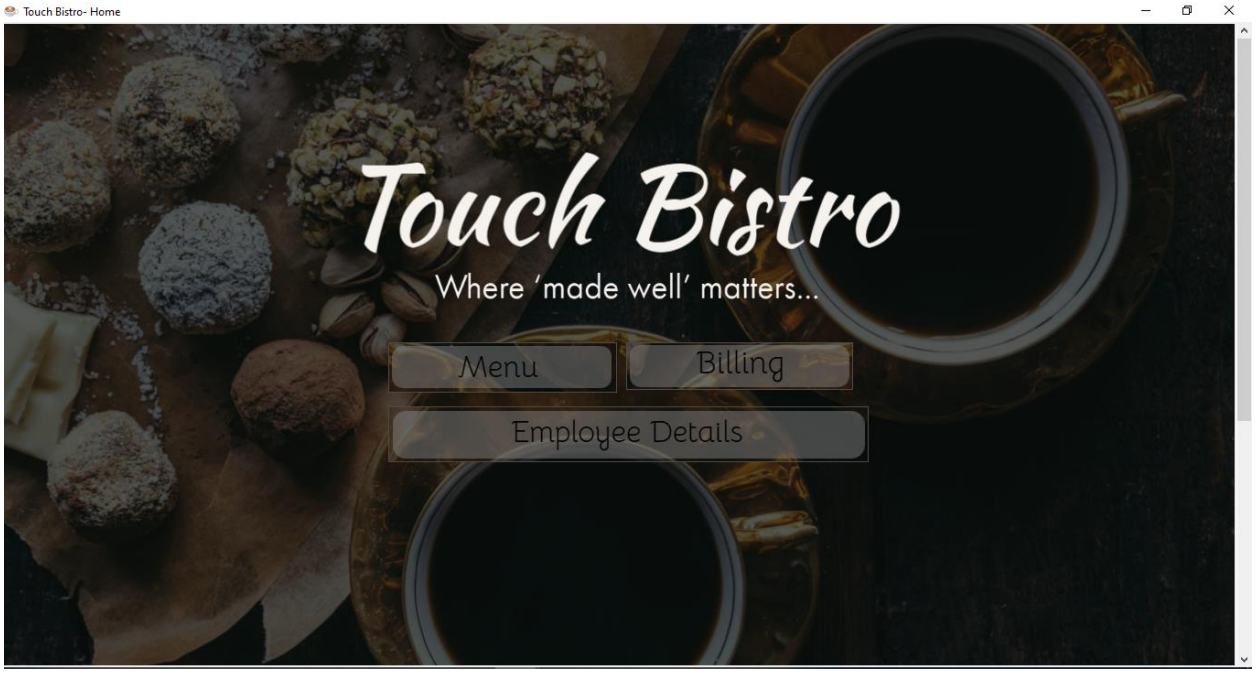


The above message box will appear if the user enters mismatched data or has not registered for an id.

LOGIN WINDOW: CORRCT CREDENTIALS

If the credentials are correct the user is directed to the home window.

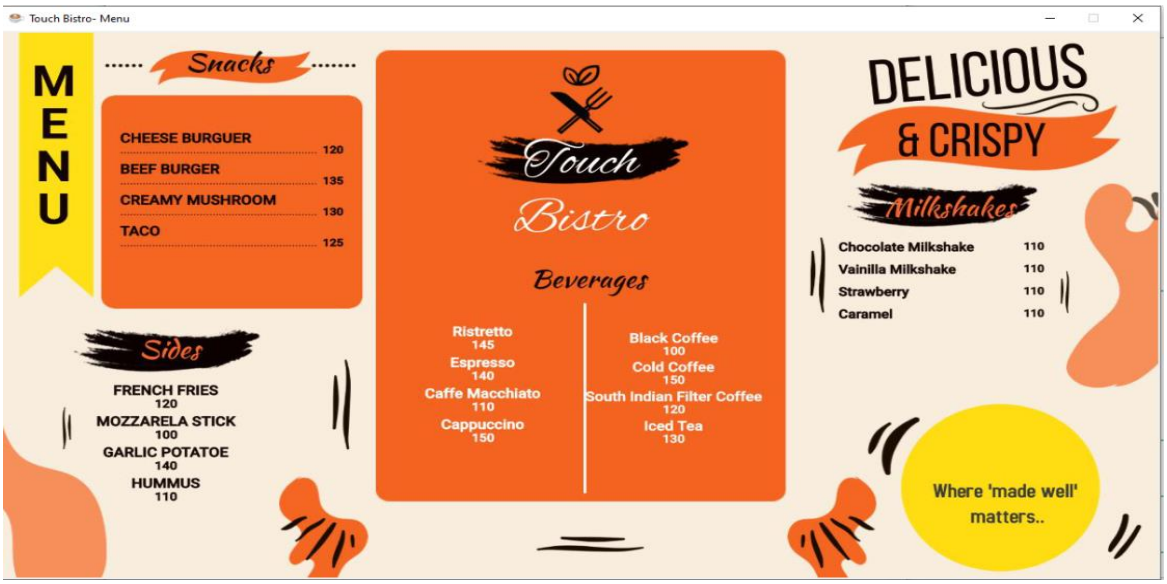
HOME WINDOW:



The menu button redirects the user to the menu page, the billing button to the billing page and employee details button to the employee details page. The about us info can be seen by user by scrolling down the page.

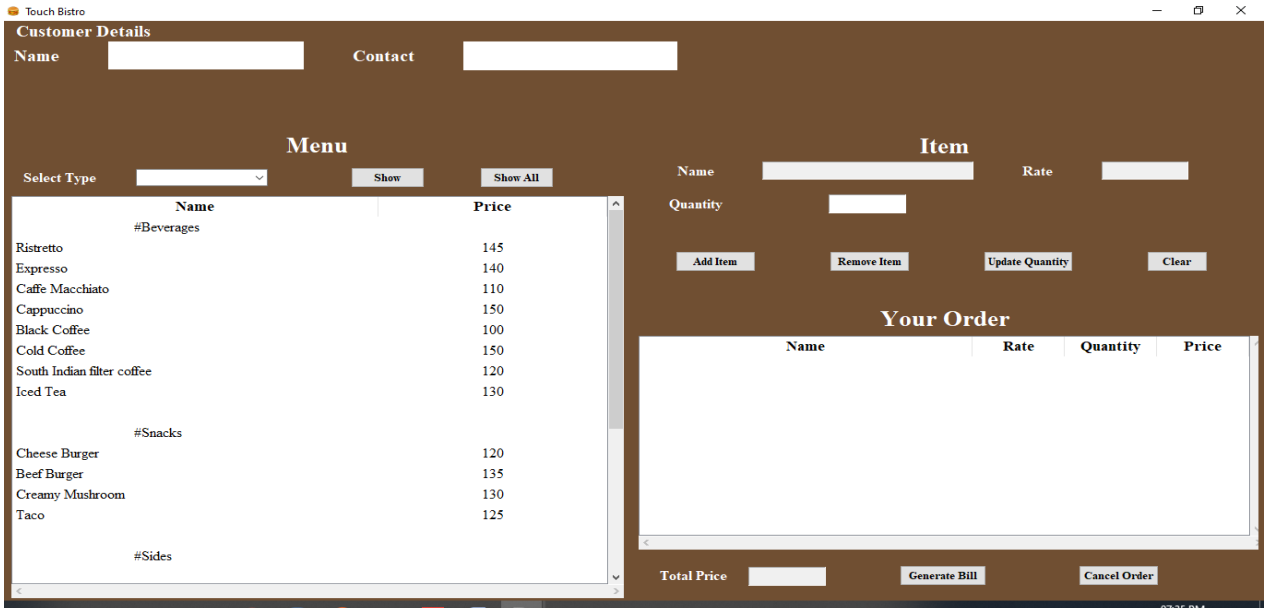
MENU WINDOW:

This window is displayed after clicking the menu button in home page.

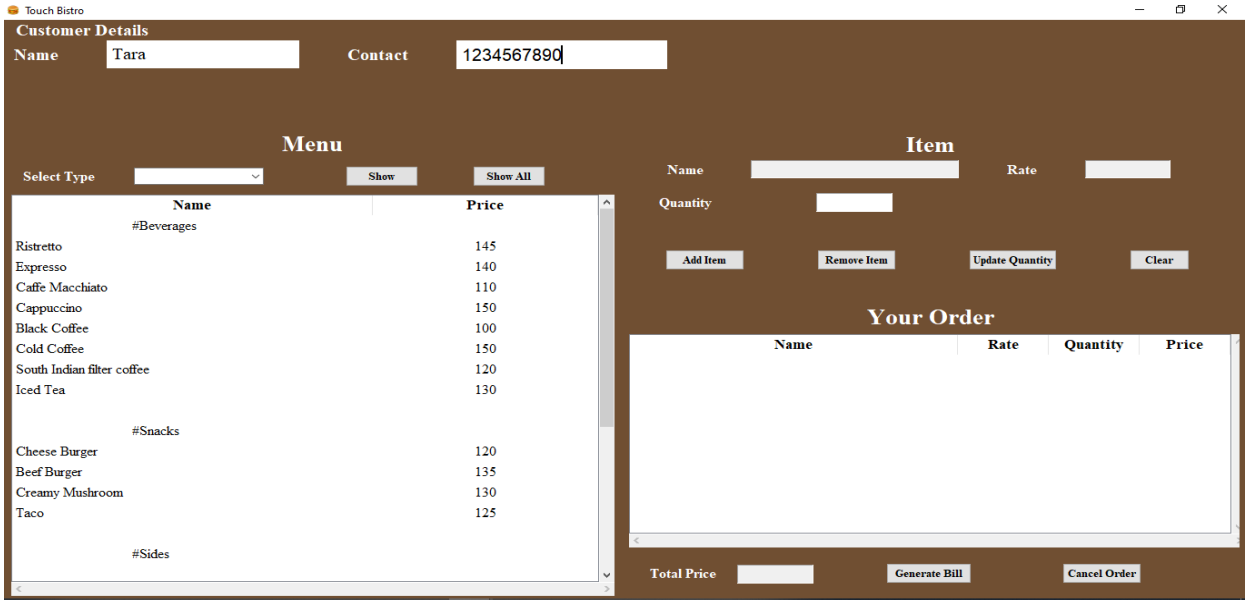


BILLING PAGE:

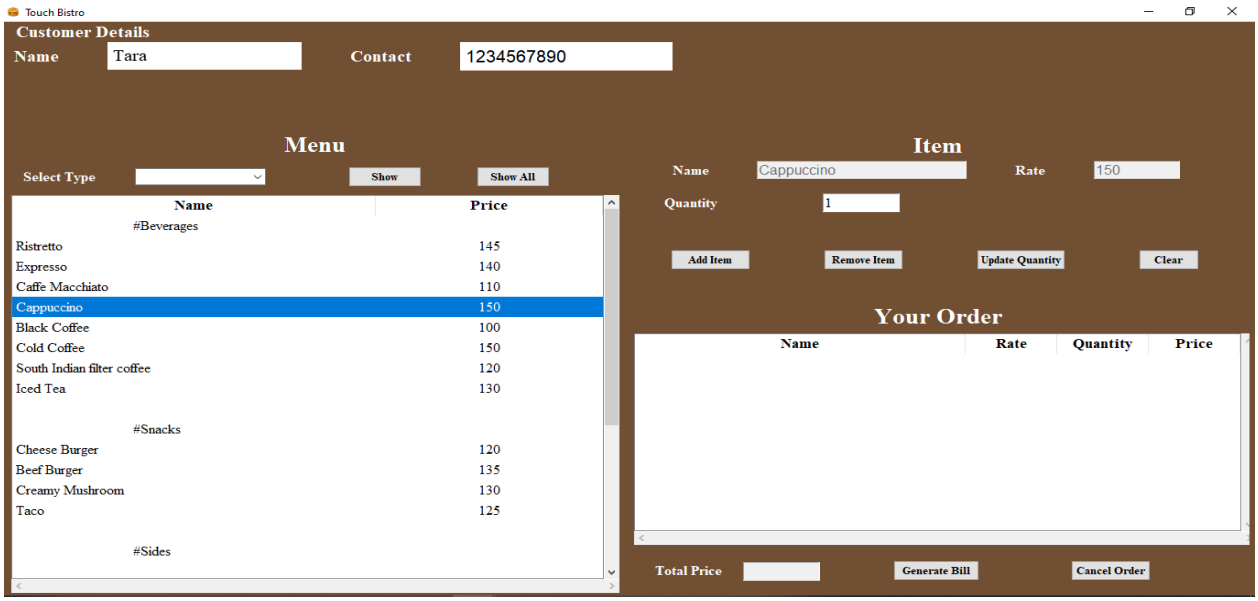
This page is displayed after clicking on these billing button in home page.



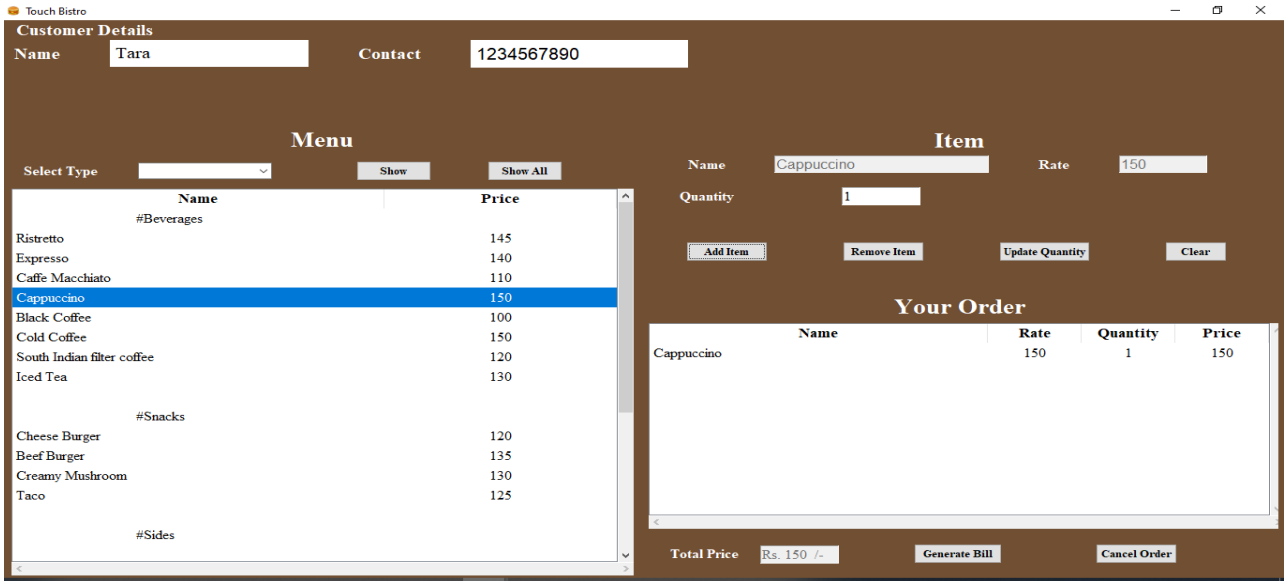
The user is required to enter the customer's name and contact number.



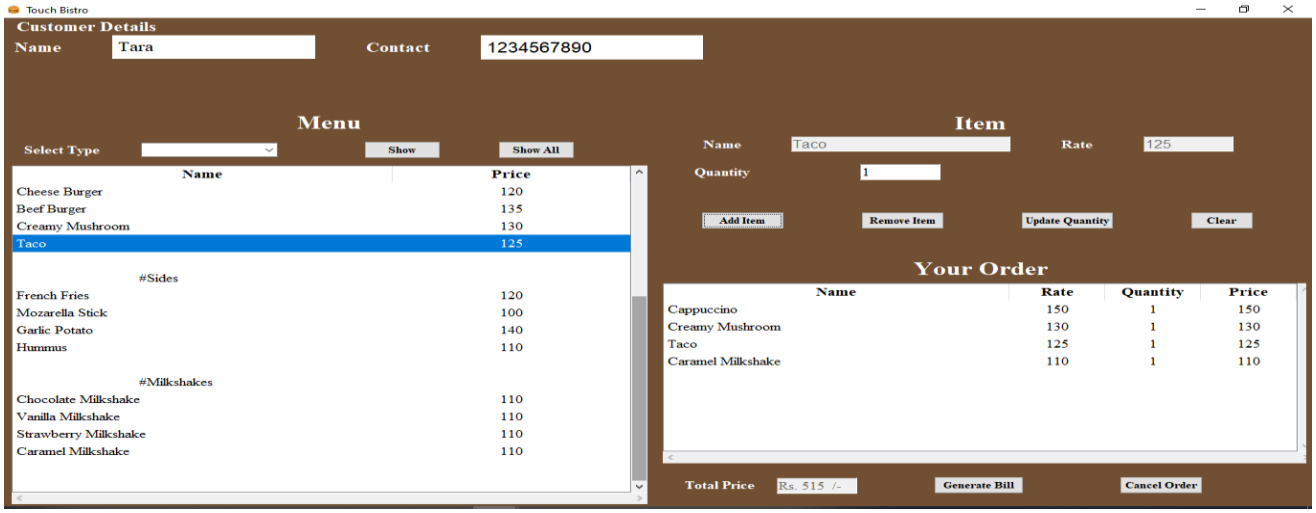
After entering the customer details, the user needs to select the item purchased from the menu.



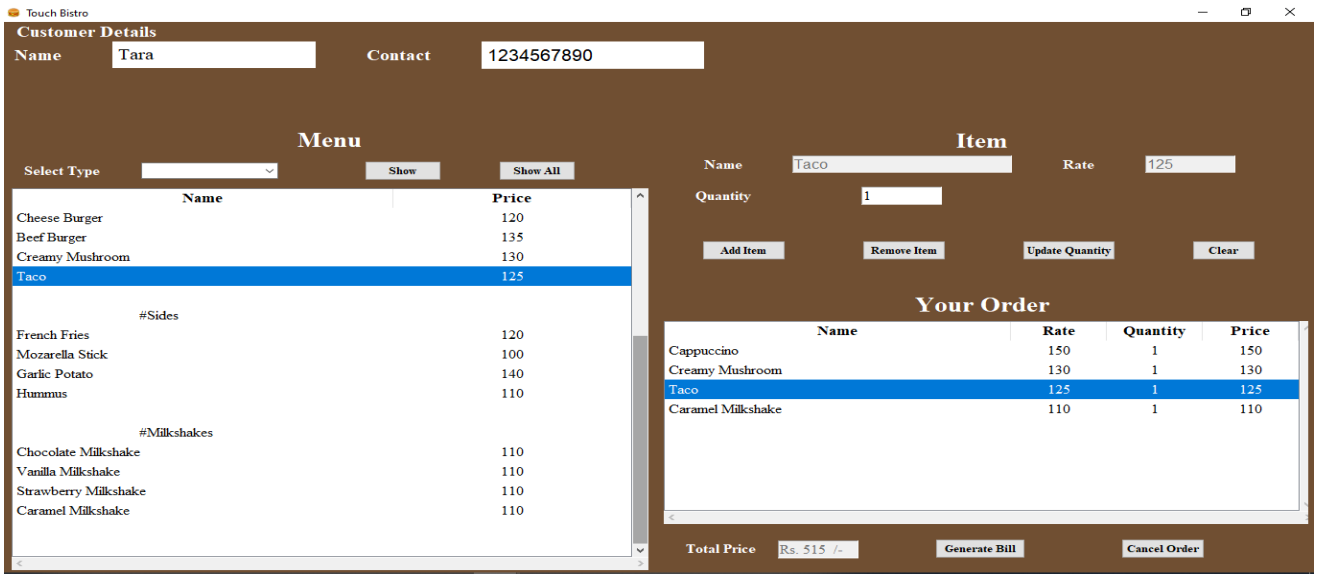
After selecting the purchased item the user needs to click on the add item button. Upon doing so, the window will appear like this.



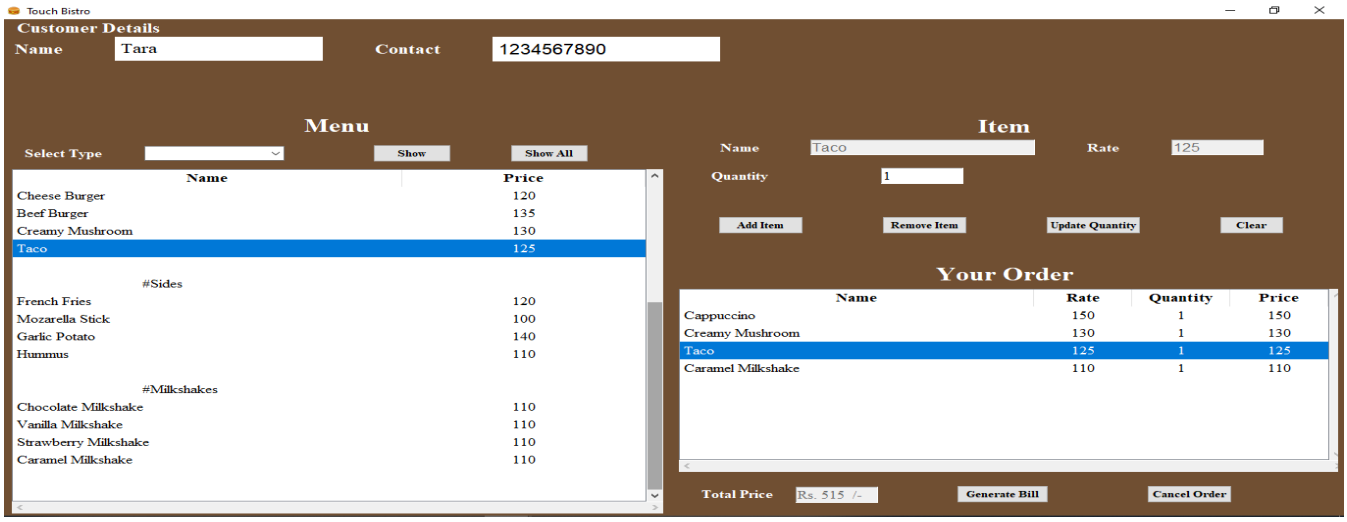
The user can add the other items if they wish so in a similar manner.



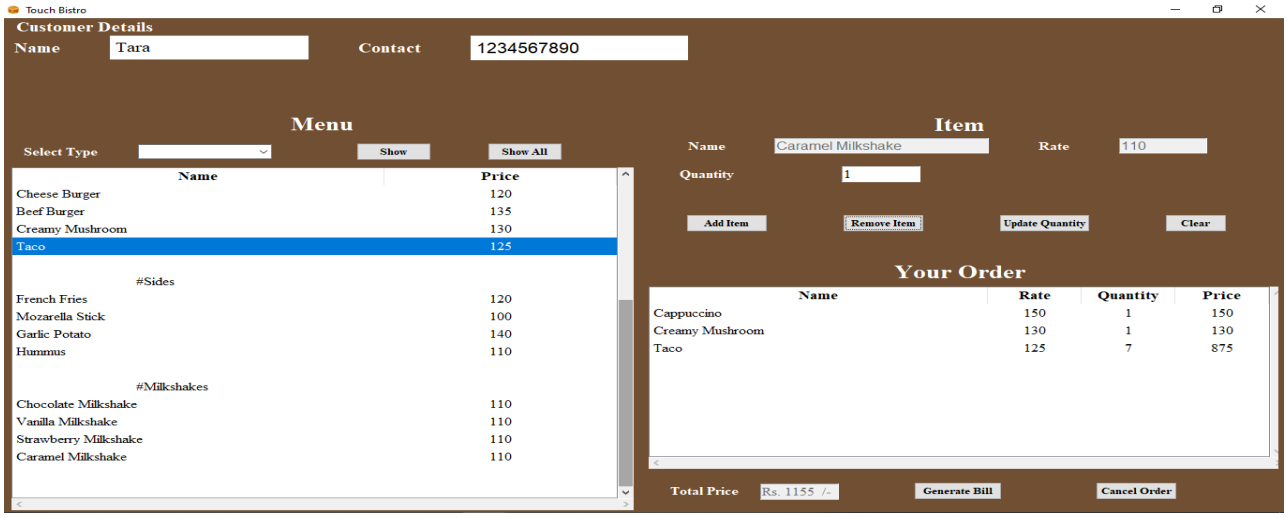
To update quatity of a selectd item,the user has to select item in the your order panel and enter the required quantity in the appropriate field and click on update quality button.



. After updating the window looks like this.

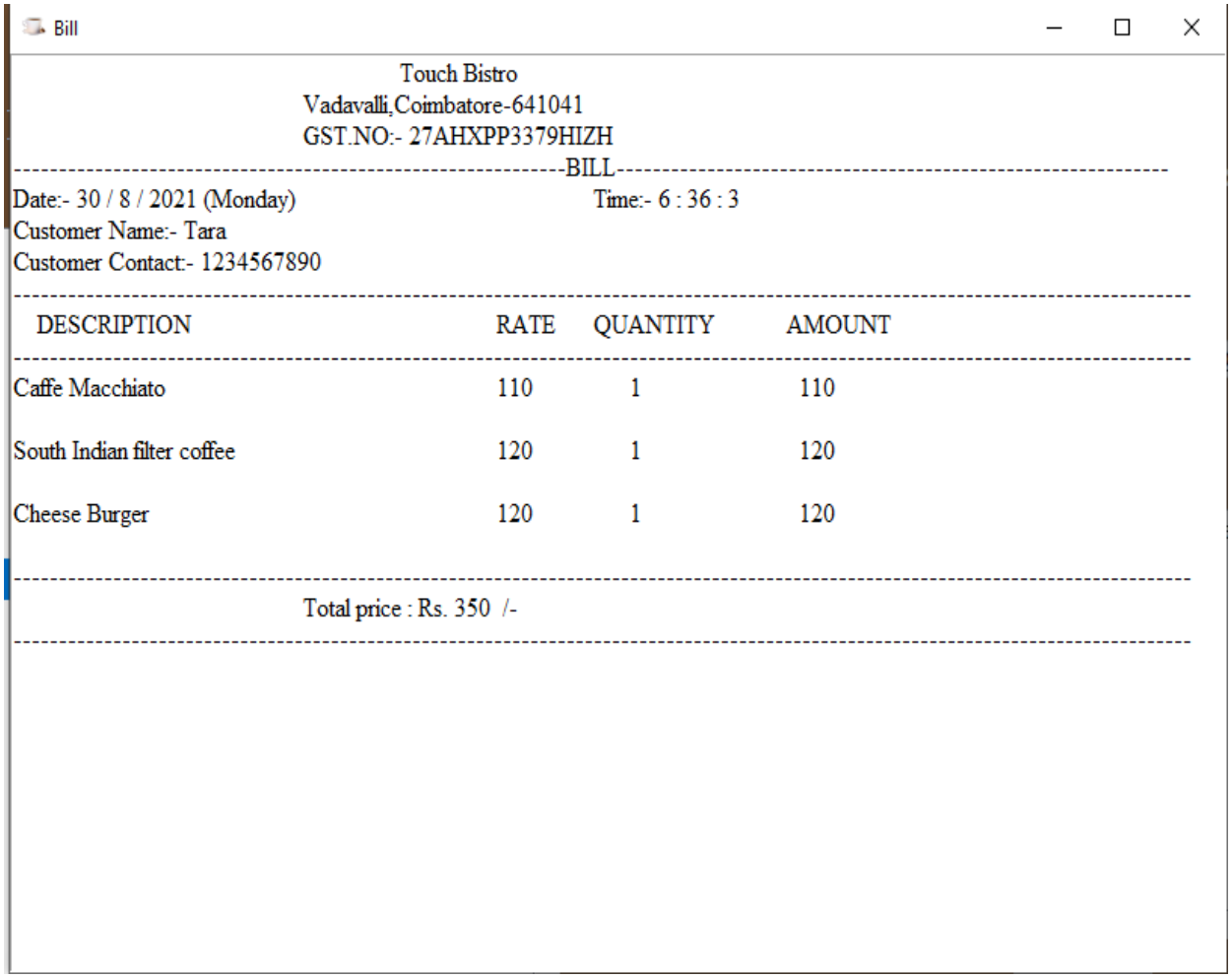


To remove a required item the user needs to select the item and click on remove item button.



To generate bill,, the user needs to click on generate bill button and confirm it in the following window by clicking on yes button.

Upon doing so, a window similar to the following image will appear showing the bill of the appropriate purchases made.



Bill

Touch Bistro
Vadavalli,Coimbatore-641041
GST.NO:- 27AHXPP3379HIZH

----------BILL----------

Date:- 30 / 8 / 2021 (Monday) Time:- 6 : 36 : 3
Customer Name:- Tara
Customer Contact:- 1234567890

DESCRIPTION	RATE	QUANTITY	AMOUNT
Caffe Macchiato	110	1	110
South Indian filter coffee	120	1	120
Cheese Burger	120	1	120

Total price : Rs. 350 /-

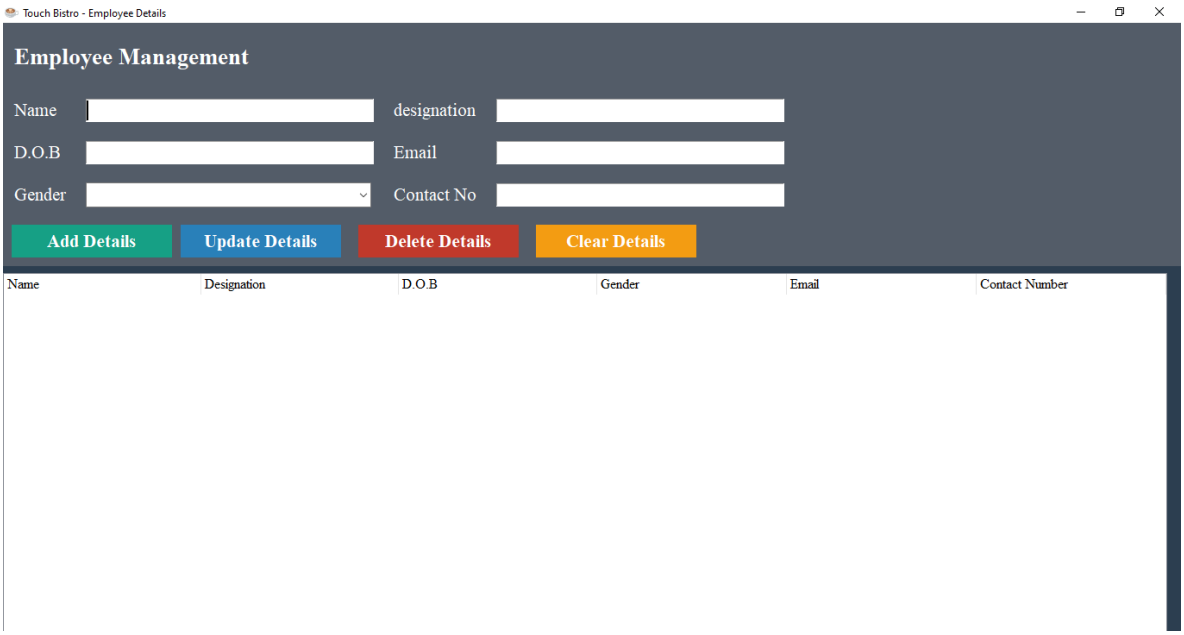
A copy of the bill will be stored in a txt file named with the customer details in a folder called bill records which the user can access in future

To cancel an order, the user needs to click on cancel order button and confirm it by clicking on the yes button in the following window.

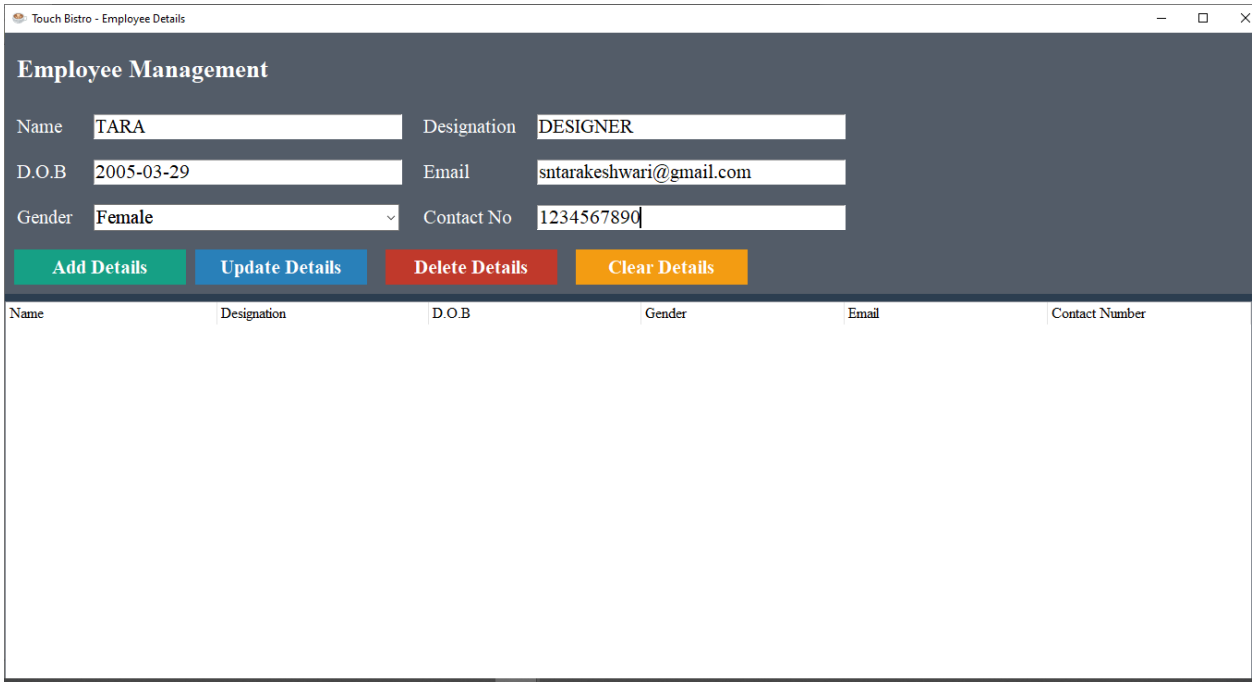
Upon doing so the billing page will remove all the details entered by the user and remove the details from the bill records folder too.

EMPLOYEE DETAILS WINDOW:

This window is displayed after clicking the employee details button in the home page.

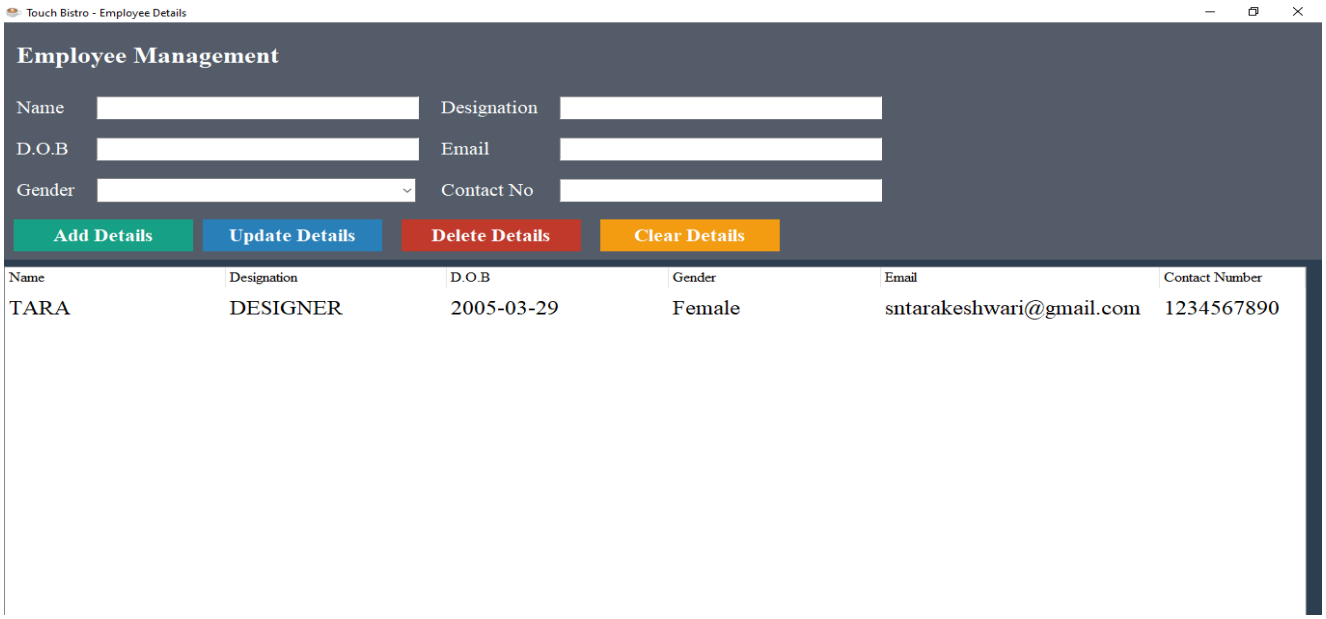


EMPLOYEE DETAILS WINDOW: AFTER ADDING EMPLOYEE



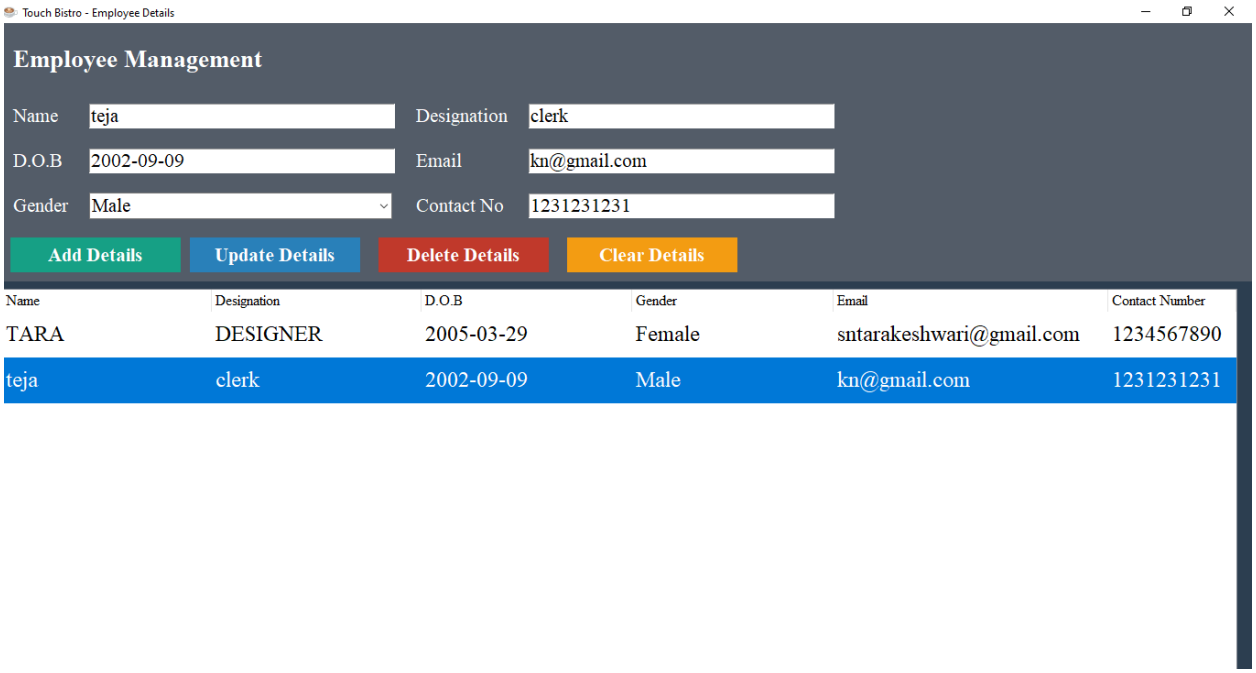
The user needs to enter the details of the employee in the required fields.

EMPLOYEE DETAILS WINDOW: AFTER ENTERING AN EMPLOYEE'S DETAILS

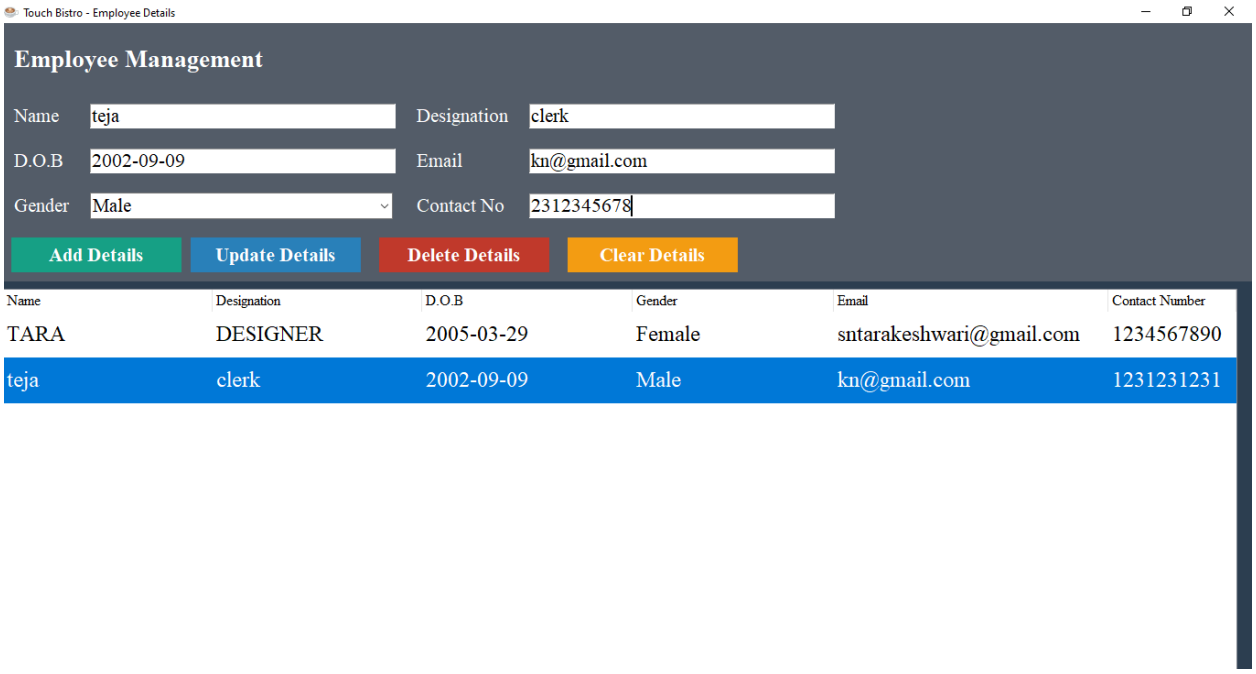


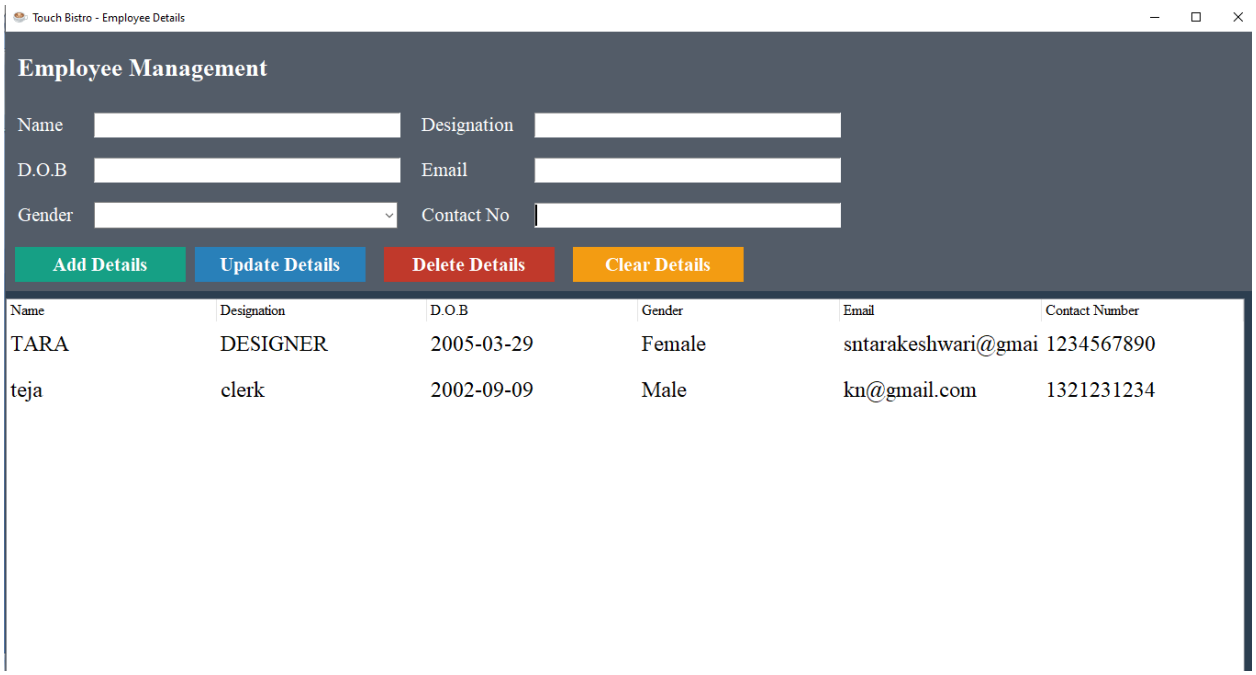
EMPLOYEE DETAILS WINDOW: UPDATING AN EMPLOYEE'S DETAILS

To update employee details, the user has to select the employee details the user wishes to update.

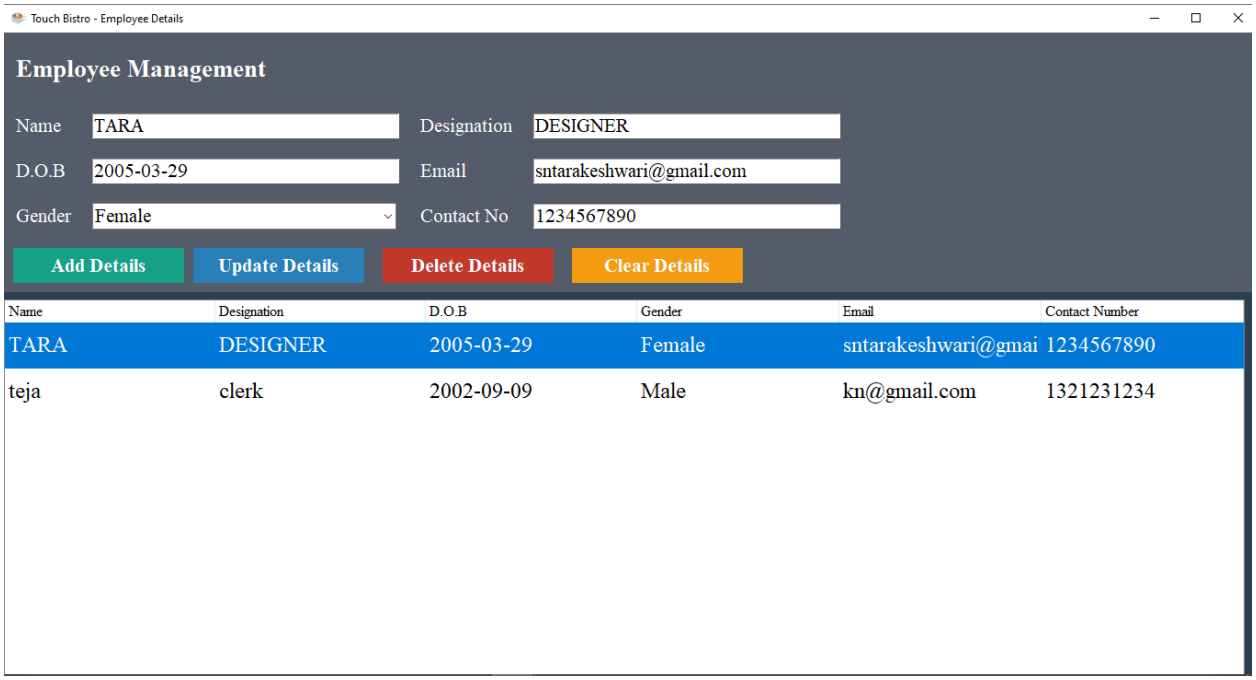


After selecting the user needs to change the details of whatever they wish to change.

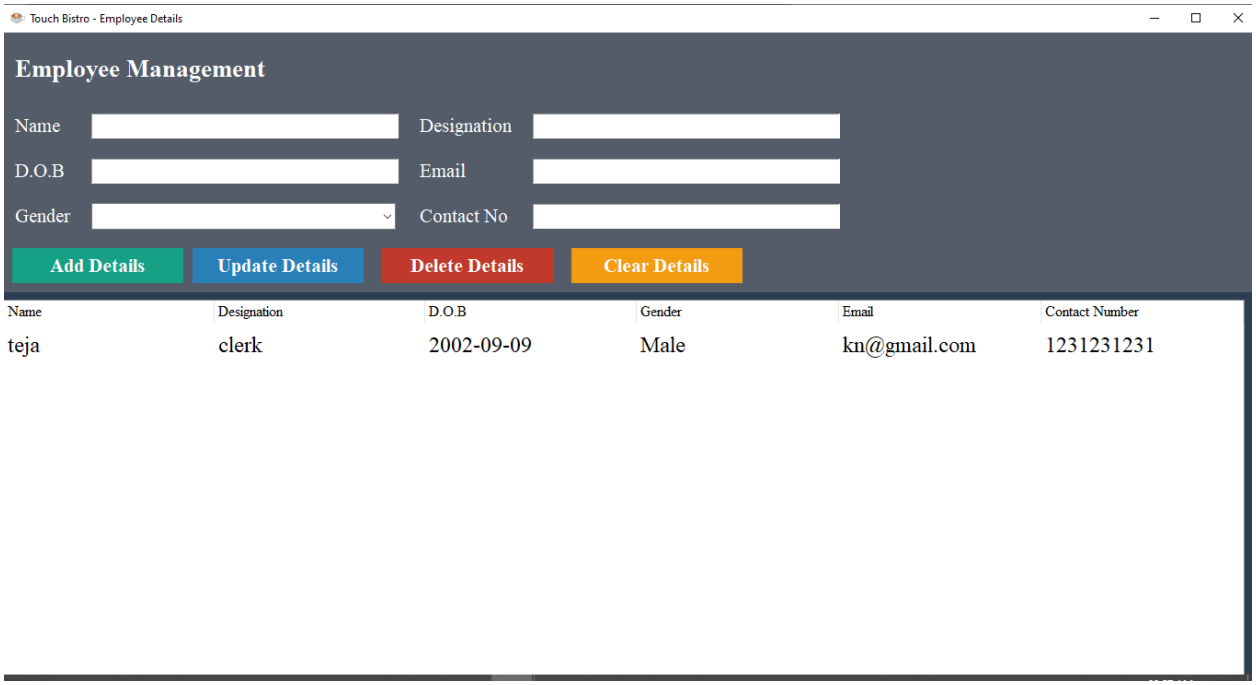




To delete a employee, the user first needs t select the recordd to be deleted.



The updated window will be as follows.


Touch Bistro - Employee Details
Employee Management
Name
Designation
D.O.B
Email
Gender
Contact No
Add Details
Update Details
Delete Details
Clear Details
Name
Designation
D.O.B
Gender
Email
Contact Number
teja
clerk
2002-09-09
Male
kn@gmail.com
1231231231

FUTURE ENHANCEMENTS

This application can be adopted for any further development by adding dynamic windows instead of static windows. This will make the administration more efficient and processes can be carried out at a faster pace. It is very flexible, and changes can be made without facing much difficulties. Further extension can be made easily.

The application can also be developed in such a way that it may be used in various platforms. Since Python 3.8 is very flexible and code can be compiled separately, we can incorporate any modular programs in this application. Thus, even after the development of application the user can make changes easily.

Report generated can also be changed according to the user's changing needs. This application can be easily modified and proper documentation about this application is done for further reference, modification and for future development

BIBLIOGRAPHY

- https://www.tutorialspoint.com/
- https://www.stackoverflow.com
- https://www.geeksforgeeks.org/
- https://www.educba.com/