Sri Adichunchanagiri Shikshana Trust ®

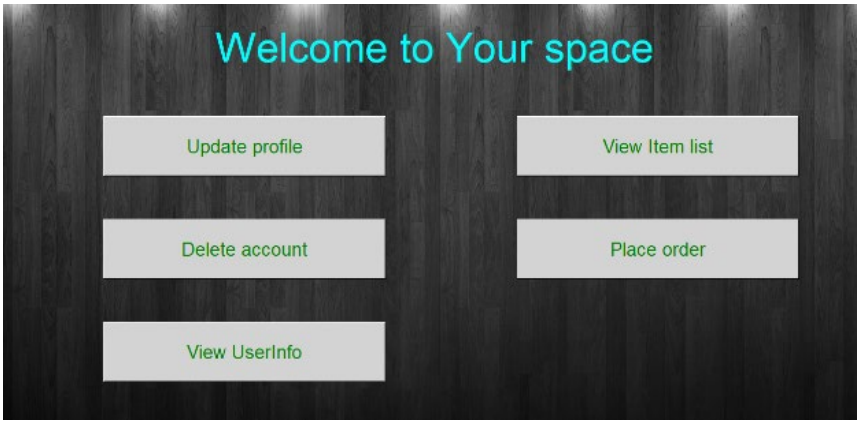
BGS NATIONAL PUBLIC SCHOOL

(Affiliated to Central Board of Secondary Education, New Delhi)
Hulimavu, Bannerghatta Road, Bengaluru - 560 076

Computer Science Investigatory Project

Year: 2021 –2022

Topic: Inventory Management system



NAME:- **SUJAY J**

Class :- **12A**

ROLL NO:- **37**

Inventory Management system

CERTIFICATE

This is to certify that **Sujay Jayakumar**of class XII of BGS National Public School has successfully completed the Investigatory Project in Computer Science for **ALL INDIA SENIOR SECONDARY CERTIFICATE EXAMINATION (AISSCE)** prescribed by CBSE in the year 2021-2022.

Date:

Principal Sign

External Examiner Internal Examiner

ACKNOWLEDGEMENTS

I would like to express my gratitude to our Respected **Academic Director Dr Sri. S A Nair**, **Principal Ms. Sreekala G Kumar**, **and Vice Principal Ms Savitha Suverna** for being a constant pillar of support.

I would then like to thank our **Computer Science teacher,Ms. Babitha E Z** for helping me with this project and guiding us throughout.

I would also like to thank God Almighty and our parents for always being by our side and our fellow mates who were always ready to help.

TABLE OF CONTENTS

INTRODUCTION

This project mainly focuses on the development of an application program on an ecommerce software where customer can shop and order, receive a bill and we can manage customer database and the shop items.

It is a convenient software where the database manages all the customer details in a very organized way and has easy access to the data and can be modified and updated easily.

The gui interface for the billing system provides an interactive way to buy items from the shop and receive a self-generated bill. The user can update, delete and view their account details and order products from the inventory directly through their account portals.

The admin has access to all the customer details as well as the products list in the inventory and can change the same guided by the easy-to-use application program.

Project Selection

I adopted this idea to provide an interactive interface between users to place orders and make bills from our shop. The application program provides a quality experience to the user using data files. Concepts of Python and MySQL were used to develop the application.

The admin creates a database consisting of 1 table: customer. Users can place order of various items using a simple and efficient interface. All this has been achieved through the efficient extraction from and injection into the database and csv file keeping track of the inventory of the products available.

WORKING ENVIRONMENT

OPTIMUM REQUIREMENTS

- **Operating System** – Windows 7,8,10,11
- **Processor** – Must be clocked over 1.5 GHz
- **Graphics Driver** – Intel Integrated Graphics
- **RAM** – 2 GB or more.
- **Hard Disk** – 128 GB
- **Python interpreter** – Python IDLE 3.6
- **MySQL, MS Excel**

LIBRARIES & MODULES

Libraries	Purpose
tkinter	To give a GUI interface
csv	To access csv files
Pillow	To provide background information and formatting
My SQL connector	To provide a database for easier access of information
MessageBox	To display error and information messages
Tempfile	This module creates temporary files and directories

Data Dictionary

USER DEFINED FUNCTIONS

Functions	PURPOSE
MAIN PROJECT CODE	
def main():	imports main window file
def f():	imports gui_register window
def admin():	imports admin_window
def login():	login screen
def login_verify():	verify login credentials
def login_sucess():	login success screen
def password_not_recognised():	password not recognised
def user_not_found():	user not found
def delete_username_used()	delete screen
def delete_login_success():	success screen
def delete_login():	delete success screen
def delete_register():	delete register screen
def delete_password_not_recognised()	deletes delete_password_not_recognised screen
def delete_user_not_found_screen():	deletes delete_user_not_found_screen

def delete_main():	delete main account screen
def main_account_screen():	creates the main screen layout
GUI REGISTER WINDOW	
def submit():	submits the user credential values into database
MAIN WINDOW	
delete()	delete confirmation box
delete1()	executes account delete command
view()	displays customer account details
delete2()	destroys view window
update()	update customer window
submit()	saves the updated changes
delete4()	destroys the update window
login_success_screen()	update success info box
viewlist()	displays item list
place_order()	calls in the billing file
main_account_screen()	main window defining
BILLING APP	
product()	creates list of all products
change()	checks quantity and updates the inventory by subtracting bought item quantity
total()	prints the total calculated cost and items

receipt()	prints receipt of the order
print1()	creates bill in text file
exit()	quit the billing application
ADMIN WINDOW	
update()	accepts customer id to open account update window
update1()	open account update window
submit()	updates the changes
delete4()	close update screen
login_success_screen()	update success info box
delete1()	deletes the account id entered
delete2()	close the delete window
delete()	open window to delete the account
users_info()	accepts customer id to be viewed
view()	displays account details
delete2()	destroys the view screen
item()	accepts changes to be made to the inventory
item2()	registers the changes in the inventory csv file
main_account_screen()	design for the admin window

SOURCE CODE

1st screen:

```
from tkinter import *
import tkinter as tk
from PIL import Image,ImageTk
import mysql.connector
my=mysql.connector.connect(host='localhost',user='root',database='project',password='')
cursor=my.cursor()
if my.is_connected():
   print('Database connected')

def main():
   import main_window

def f():
   import gui_RegisterWindow

def admin():
   import admin_window



def login():
   global canvas3
   global login_screen
   global username_verify
   global password_verify
   global cid_verify
   username_verify = StringVar()
   password_verify = StringVar()
   cid_verify = StringVar()


   global username_login_entry
   global password_login_entry
   global cid_login_entry
   login_screen = Toplevel(main_screen)
   login_screen.title("Login")
   login_screen.geometry("330x260")


   # Read the Image
   image = Image.open("loginpic.jpg")
```

```
    # Reszie the image using resize() method
    resize_image = image.resize((330, 260))
    img = ImageTk.PhotoImage(resize_image)

    # Create Canvas
    canvas3= tk.Canvas(login_screen, width = 400,height = 400)

    canvas3.pack(fill = "both", expand = True)

    # Display image
    canvas3.create_image( 0, 0, image = img, anchor = "nw")

    # Add Text
    canvas3.create_text(170,25, text = "Please enter details below to log into ur
acc",font=("david",12),fill='white')
    canvas3.create_text(120,58, text="Username * ",font=("david", 12),fill='white')
    canvas3.create_text(120,112, text="Password * ",font=("david", 12),fill='white')
    canvas3.create_text(120,170, text="Customer ID * ",font=("david", 12),fill='white')
    #entry box
    username_login_entry = tk.Entry(login_screen,textvariable=username_verify)
    canvas3.create_window(140,80, window=username_login_entry)
    password_login_entry= tk.Entry(login_screen,textvariable=password_verify, show='*')
    canvas3.create_window(140,130, window=password_login_entry)
    cid_login_entry= tk.Entry(login_screen,textvariable=cid_verify)
    canvas3.create_window(140,190, window=cid_login_entry)
    #add buttons
    button1=Button(login_screen,text="Login", height="1", width="10",command =
login_verify,font=("david", 10))
    # Display Buttons
    button1_canvas = canvas3.create_window(96, 210,anchor = "nw",window = button1)
    login_screen.mainloop()


def login_verify():
    global username1
    username1 = username_verify.get()
    password1 = password_verify.get()
    cid1 = cid_verify.get()
    sql='''SELECT * FROM customer;'''
    cursor.execute(sql)
    my.close()
    data=cursor.fetchall()
    #print(data)
    for i in data:
```

```
        #print(i)
        #print(i[0])
        if i[0]==cid1:
            if password1==i[2]:
                login_sucess()
                current_user=open('current_user.txt','w')
                current_user.write(cid1)
                current_user.close()

                break
            else:
                password_not_recognised()
                current_user=open('current_user.txt','w')
                current_user.close()
                break
        else:
            continue
    else:
        user_not_found()

# Designing popup for login success

def login_sucess():
    global login_success_screen
    login_success_screen = Toplevel(login_screen)
    login_success_screen.title("Success")
    login_success_screen.geometry("150x100")
    Label(login_success_screen, text="Login Success").pack()
    Button(login_success_screen, text="OK", command=main).pack()

# Designing popup for login invalid password

def password_not_recognised():
    global password_not_recog_screen
    password_not_recog_screen = Toplevel(login_screen)
    password_not_recog_screen.title("Error")
    password_not_recog_screen.geometry("150x100")
    Label(password_not_recog_screen, text="Invalid Password",fg="red").pack()
    Button(password_not_recog_screen, text="Try Again",
command=delete_password_not_recognised).pack()

# Designing popup for user not found

def user_not_found():
    global user_not_found_screen
```

```
    user_not_found_screen = Toplevel(login_screen)
    user_not_found_screen.title("Error")
    user_not_found_screen.geometry("150x100")
    Label(user_not_found_screen, text="User Not Found",fg="red").pack()
    Button(user_not_found_screen, text="Try Again", command=delete_user_not_found_screen).pack()
    Label(user_not_found_screen, text="Register yourself now!").pack()

# Deleting popups
def delete_username_used():
    username_used_screen.destroy()

def delete_login_success():
    login_success_screen.destroy()

def delete_login():
    login_screen.destroy()

def delete_register():
    register_screen.destroy()

def delete_password_not_recognised():
    password_not_recog_screen.destroy()

def delete_user_not_found_screen():
    user_not_found_screen.destroy()

def delete_main():
    main_account_screen_screen.destroy()

def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("2000x1500")
    # Read the Image
    image = Image.open("download.jpg")

    # Reszie the image using resize() method
    resize_image = image.resize((1590, 840))
    img=ImageTk.PhotoImage(resize_image)

    # Create Canvas
    canvas1 = Canvas(main_screen, width = 400,height = 400, relief=SUNKEN, bd=15)

    canvas1.pack(fill = "both", expand = True)
```

```
    # Display image
    canvas1.create_image( 0, 0, image=img, anchor = "nw")

    # Add Text
    canvas1.create_text(700,110, text = "Welcome to Shopezee",font=("david", 45, 'bold'),fill='purple')
    canvas1.create_text(650,380, text = "New here? Register now!",font=("david", 20))

    #add buttons
    button1=Button(main_screen,text="Login", height="2", width="20",command = login,font=("david",
25,
 'bold'),fg="lawn green",bg='blue',bd=5)
    button2=Button(main_screen,text="Register", height="2", width="20",command=f,font=("david", 25
, 'bold'),fg="lawn green",bg='blue',bd=5)
    button3=Button(main_screen,text="ADMIN",                                                    height="1",
width="10",command=admin,font=("david",
15),fg="white",bg='dark blue',relief=RIDGE,bd=2)
    # Display Buttons
    button1_canvas = canvas1.create_window( 450, 200, anchor = "nw",window = button1)
    button2_canvas = canvas1.create_window( 450,420,anchor = "nw",window = button2)
    button3_canvas = canvas1.create_window( 1200, 630,anchor = "nw",window = button3)
    main_screen.mainloop()
main_account_screen()
'''#==============================================================================
def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("2000x1500")
    # Read the Image
    image = Image.open("download.png")

    # Reszie the image using resize() method
    resize_image = image.resize((1585, 830))
    img=ImageTk.PhotoImage(resize_image)

    # Create Canvas
    canvas1 = Canvas(main_screen, width = 400,height = 400)

    canvas1.pack(fill = "both", expand = True)

    # Display image
    canvas1.create_image( 0, 0, image=img, anchor = "nw")

    # Add Text
    canvas1.create_text(740,30, text = "Welcome to Your space",font=("david", 35),fill='cyan')
    #canvas1.create_text(750,365, text = "New here? Register now",font=("david", 20))
```

```
    #add buttons
    button1=Button(main_screen,text="Update profile", height="1", width="15",command = "",font=("david",
20),fg="green",bg='light grey')
    button2=Button(main_screen,text="Delete account", height="1", width="15",command="",font=("david",
 20),fg="green",bg='light grey')
    button3=Button(main_screen,text="View UserInfo", height="1", width="15",command = "",font=("david",
20),fg="green",bg='light grey')
    button4=Button(main_screen,text="View Item list", height="1", width="15",command = "",font=("david"
, 20),fg="green",bg='light grey')
    button5=Button(main_screen,text="Place order", height="1", width="15",command = "",font=("david",
 20),fg="green",bg='light grey')

    # Display Buttons
    button1_canvas = canvas1.create_window( 300, 100, anchor = "nw",window = button1)
    button2_canvas = canvas1.create_window( 300, 200,anchor = "nw",window = button2)
    button3_canvas = canvas1.create_window( 300, 300, anchor = "nw",window = button3)
    button4_canvas = canvas1.create_window( 300, 400, anchor = "nw",window = button4)
    button5_canvas = canvas1.create_window( 300, 450, anchor = "nw",window = button4)
    main_screen.mainloop()
main_account_screen()'''
2nd screen:
from tkinter import *
import tkinter as tk
from PIL import Image,ImageTk
import csv
import mysql.connector
my=mysql.connector.connect(host='localhost',user='root',database='project',password='')
cursor=my.cursor()
if my.is_connected():
    print('Database connected')

def update():
    global root2
    global box1
    global customer
    root2=Tk()
    root2.title('update')
    root2.geometry('400x400')
    box1=Entry(root2, width=30)
    box1.grid(row=0,column=1,padx=20)
```

```
    customer=Label(root2, text='CID to be updated:')
    customer.grid(row=0,column=0)
    '''cursor.execute("select * from customer where customer_id='"+ box1.get() +"'")

    while True:
        data=cursor.fetchone()
        if data==None:
            break
        else:
            print(data)
            r=list(data)
    print(r)'''

    update_btn=Button(root2,text='update record from database',command=update1)
    update_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)
def update1():
    global root3
    global r
    root3=Tk()
    root3.title('Review and Update profile')
    root3.geometry('400x400')
    cursor.execute("select * from customer where customer_id='"+ box1.get() +"'")
    while True:
        data=cursor.fetchone()
        if data==None:
            break
        else:
            print(data)
            r=list(data)
    print(r)
    #creating text boxes

    global box_customer_id
    global box_f_name
    global box_l_name
    global box_date_of_birth
    global box_contact_number
    global box_address
    global box_city
    global box_state
    global box_zipcode
    global box_gender
    global box_username
    global box_password
```

```
box_customer_id=Entry(root3, width=30)
box_customer_id.grid(row=0,column=1,padx=20)
box_customer_id.insert('end', r[0])


box_f_name=Entry(root3, width=30)
box_f_name.grid(row=1,column=1)
box_f_name.insert('end', r[3])

box_l_name=Entry(root3, width=30)
box_l_name.grid(row=2,column=1)
box_l_name.insert('end', r[4])

box_date_of_birth=Entry(root3, width=30)
box_date_of_birth.grid(row=3,column=1)
box_date_of_birth.insert('end', r[5])

box_contact_number=Entry(root3, width=30)
box_contact_number.grid(row=4,column=1)
box_contact_number.insert('end', r[6])

box_address=Entry(root3, width=30)
box_address.grid(row=5,column=1)
box_address.insert('end', r[7])

box_city=Entry(root3, width=30)
box_city.grid(row=6,column=1)
box_city.insert('end', r[8])

box_state=Entry(root3, width=30)
box_state.grid(row=7,column=1)
box_state.insert('end', r[9])

box_zipcode=Entry(root3, width=30)
box_zipcode.grid(row=8,column=1)
box_zipcode.insert('end', r[10])

box_gender=Entry(root3,width=30)
box_gender.grid(row=9,column=1)
box_gender.insert('end', r[11])


box_username=Entry(root3,width=30)
```

```
box_username.grid(row=11,column=1)
box_username.insert('end', r[1])

box_password=Entry(root3,width=30)
box_password.grid(row=12,column=1)
box_password.insert('end', r[2])

#creating text box labels
customer_id_label=Label(root3, text='CID')
customer_id_label.grid(row=0,column=0)

f_name_label=Label(root3, text='First name')
f_name_label.grid(row=1,column=0)

l_name_label=Label(root3, text='last name')
l_name_label.grid(row=2,column=0)

date_of_birth_label=Label(root3, text='date of birth')
date_of_birth_label.grid(row=3,column=0)

contact_number_label=Label(root3, text='contact number')
contact_number_label.grid(row=4,column=0)

address_label=Label(root3, text='address')
address_label.grid(row=5,column=0)

city_label=Label(root3, text='city')
city_label.grid(row=6,column=0)

state_label=Label(root3, text='state')
state_label.grid(row=7,column=0)

zipcode_label=Label(root3, text='zipcode')
zipcode_label.grid(row=8,column=0)

gender_label=Label(root3, text='Gender')
gender_label.grid(row=9,column=0)


username_label=Label(root3, text='Username')
username_label.grid(row=11,column=0)

password_label=Label(root3, text='Password')
password_label.grid(row=12,column=0)
```

```
    submit_btn=Button(root3,text='Update record in database',command=submit)
    submit_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)

def submit():
    cursor.execute("delete from customer where customer_id='"+ box1.get() +"'")
    my.commit()
    customer_id=box_customer_id.get()
    f_name=box_f_name.get()
    l_name=box_l_name.get()
    date_of_birth=box_date_of_birth.get()
    contact_number=box_contact_number.get()
    address=box_address.get()
    city=box_city.get()
    state=box_state.get()
    zipcode=box_zipcode.get()
    gender=box_gender.get()
    username=box_username.get()
    password=box_password.get()

    #insert into table
    if(customer_id=='' or f_name=='' or username=='' or password=''):
        MessageBox.showinfo('Insert Status','All Fields are required')
    else:
        cursor.execute('''insert into custome
r
values("{}","{}","{}","{}","{}","{}","{}","{}","{}","{}","{}","{}");'''.format(customer_id,username,password,f
_name,l_na
me,date_of_birth,contact_number,address,city,state,zipcode,gender))
        my.commit()
    #clear text boxes already
    box_customer_id.delete(0,END)
    box_f_name.delete(0,END)
    box_l_name.delete(0,END)
    box_date_of_birth.delete(0,END)
    box_contact_number.delete(0,END)
    box_address.delete(0,END)
    box_city.delete(0,END)
    box_state.delete(0,END)
    box_zipcode.delete(0,END)
    box_gender.delete(0,END)
    box_username.delete(0,END)
    box_password.delete(0,END)
    login_success_screen()
```

```
    '''except:
        global error_screen
        error_screen = Toplevel(main_screen)
        error_screen.title("Update Unsucessful")
        error_screen.geometry("150x100")
        Label(error_screen, text="Update failed, Try again").pack()
        Button(error_screen, text="OK", command=delete3).pack()

def delete3():
    error_screen.destroy()'''
def delete4():
    login_success_screen.destroy()
    root3.destroy()
    root2.destroy()

def login_success_screen():
    global login_success_screen
    login_success_screen = Toplevel(main_screen)
    login_success_screen.title("Update status")
    login_success_screen.geometry("250x100")
    Label(login_success_screen, text="Update Success").pack()
    Button(login_success_screen, text="OK", command=delete4).pack()

def delete1():
    global root
    r=[]
    lang=box_customer_id.get()
    sql='''select customer_id from customer;'''
    cursor.execute(sql)
    while True:
        data=cursor.fetchone()
        if data==None:
            break
        else:
            print(data[0])
            r.append(data[0])
    if lang in r:
        cursor.execute("delete from customer where customer_id='"+ lang +"'")
        my.commit()
        customer_id_label=Label(root1, text='CID deleted sucessfully')
        customer_id_label.grid(row=5,column=0)
        box_customer_id.delete(0,END)
    else:
        root=Tk()
        root.title('delete')
```

```
        root.geometry('200x200')
        customer_id_label=Label(root, text='CID not found')
        customer_id_label.grid(row=0,column=0)

        delete_btn=Button(root,text='Try again',command=delete2)
        delete_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)
        box_customer_id.delete(0,END)

def delete2():
    root.destroy()
    box_customer_id.delete(0,END)



    '''cursor.execute("delete from customer where customer_id='"+ box_customer_id.get() +"'")
        my.commit()
        root=Tk()
        root.title('Delete sucess')
        root.geometry('200x200')
        customer_id_label=Label(root, text='CID deleted sucessfully')
        customer_id_label.grid(row=0,column=0)

        delete_btn=Button(root,text='OK',command=delete1)
        delete_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)
    except:
        root=Tk()
        root.title('delete')
        root.geometry('200x200')
        customer_id_label=Label(root, text='CID not found')
        customer_id_label.grid(row=0,column=0)

        delete_btn=Button(root,text='Try again',command=delete1)
        delete_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)'''

def delete():
    global root1
    global box_customer_id
    root1=Tk()
    root1.title('delete')
    root1.geometry('400x400')
    box_customer_id=Entry(root1, width=30)
    box_customer_id.grid(row=0,column=1,padx=20)

    customer_id_label=Label(root1, text='CID to be deleted:')
    customer_id_label.grid(row=0,column=0)
```

```
    delete_btn=Button(root1,text='delete record from database',command=delete1)
    delete_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)

    my=mysql.connector.connect(host='localhost',user='root',database='project',password='')
    cursor=my.cursor()
    if my.is_connected():
        print('Database connected')
    cursor.execute("delete from customer where customer_id='"+ box_customer_id.get() +"'")
    my.commit()
    my.close()

    '''if(e_customer_id.get()==''):
        MessageBox.showinfo('Delete Status','CID is compulsory for deleting record')
    else:
        my=mysql.connector.connect(host='localhost',user='root',database='project',password='')
        cursor=my.cursor()
        if my.is_connected():
            print('Database connected')
        cursor.execute("delete from customer where customer_id='"+ e_customer_id.get() +"'")
        cursor.execute('commit');

        e_customer_id.delete(0,'end')
        e_f_name.delete(0,'end')
        e_username.delete(0,'end')
        MessageBox.showinfo('Delete Status','Deleted succesfully')'''

def users_info():
    global root4
    global box2
    global customer1

    root4=Tk()
    root4.title('Display userinfo')
    root4.geometry('400x200')

    box2=Entry(root4, width=30)
    box2.grid(row=0,column=1,padx=20)
    customer1=Label(root4, text='CID to be displayed:')
    customer1.grid(row=0,column=0)

    update_btn=Button(root4,text='Enter',command=view)
    update_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)
```

```
def view():
    global root5
    root5=Tk()
    root5.title('View customer Info')
    root5.geometry('400x400')

    cursor.execute("select * from customer where customer_id='"+ box2.get() +"'")
    while True:
        data=cursor.fetchone()
        if data==None:
            break
        else:
            print(data)
            r=list(data)



    customer_id1_label=Label(root5, text=r[0])
    customer_id1_label.grid(row=0,column=1)

    f_name1_label=Label(root5, text=r[3])
    f_name1_label.grid(row=1,column=1)

    l_name1_label=Label(root5, text=r[4])
    l_name1_label.grid(row=2,column=1)

    date_of_birth1_label=Label(root5, text=r[5])
    date_of_birth1_label.grid(row=3,column=1)

    contact_number1_label=Label(root5, text=r[6])
    contact_number1_label.grid(row=4,column=1)

    address1_label=Label(root5, text=r[7])
    address1_label.grid(row=5,column=1)

    city1_label=Label(root5, text=r[8])
    city1_label.grid(row=6,column=1)

    state1_label=Label(root5, text=r[9])
    state1_label.grid(row=7,column=1)

    zipcode1_label=Label(root5, text=r[10])
    zipcode1_label.grid(row=8,column=1)
```

```
gender1_label=Label(root5, text=r[11])
gender1_label.grid(row=9,column=1)


username1_label=Label(root5, text=r[1])
username1_label.grid(row=11,column=1)

password1_label=Label(root5, text=r[2])
password1_label.grid(row=12,column=1)

customer_id_label=Label(root5, text='CID')
customer_id_label.grid(row=0,column=0)

f_name_label=Label(root5, text='First name')
f_name_label.grid(row=1,column=0)

l_name_label=Label(root5, text='last name')
l_name_label.grid(row=2,column=0)

date_of_birth_label=Label(root5, text='date of birth')
date_of_birth_label.grid(row=3,column=0)

contact_number_label=Label(root5, text='contact number')
contact_number_label.grid(row=4,column=0)

address_label=Label(root5, text='address')
address_label.grid(row=5,column=0)

city_label=Label(root5, text='city')
city_label.grid(row=6,column=0)

state_label=Label(root5, text='state')
state_label.grid(row=7,column=0)

zipcode_label=Label(root5, text='zipcode')
zipcode_label.grid(row=8,column=0)

gender_label=Label(root5, text='Gender')
gender_label.grid(row=9,column=0)


username_label=Label(root5, text='Username')
username_label.grid(row=11,column=0)

password_label=Label(root5, text='Password')
```

```
    password_label.grid(row=12,column=0)

    close_btn=Button(root5,text='Close',command=delete2)
    close_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)

def delete2():
    root5.destroy()
    root4.destroy()

def item():
    global root6
    root6=Tk()
    root6.title('View customer Info')
    root6.geometry('400x400')

    global box_item
    global box_price
    global box_quantity

    box_item=Entry(root6,width=30)
    box_item.grid(row=0,column=1)

    item_label=Label(root6, text='Enter the item name')
    item_label.grid(row=0,column=0)

    box_price=Entry(root6,width=30)
    box_price.grid(row=1,column=1)

    box_quantity=Entry(root6,width=30)
    box_quantity.grid(row=2,column=1)

    price_label=Label(root6, text='Enter the new price')
    price_label.grid(row=1,column=0)

    quantity_label=Label(root6, text='Enter updated quantity')
    quantity_label.grid(row=2,column=0)

    close_btn=Button(root6,text='Update',command=item2)
    close_btn.grid(row=20,column=0,columnspan=2,pady=10,ipadx=100)

def item2():
    f=open('products.csv','r+',newline='')
    csv_r=csv.reader(f)
    csv_w=csv.writer(f)
    p=box_item.get()
```

```
    L=[]
    found=0
    for rec in csv_r:
        if rec[0]==p:
            f.seek(pos)
            rec[1]=box_price.get()
            rec[2]=box_quantity.get()
            csv_w.writerow(rec)
            found=1
        L.append(rec)
    if found==1:
        f.seek(0)
        csv_w.writerows(L)
        MessageBox.showinfo('Update Status','Updated succesfully')

    else:
        print('Record not found')
        MessageBox.showerror('Update Status','Update failed')
    f.close()

# Designing Main(first) window

def main_account_screen():
    global main_screen
    main_screen = tk.Toplevel()
    main_screen.geometry("2000x1500")
    # Read the Image
    image = Image.open("download.png")

    # Reszie the image using resize() method
    resize_image = image.resize((1585, 830))
    img=ImageTk.PhotoImage(resize_image)

    # Create Canvas
    canvas1 = Canvas(main_screen, width = 400,height = 400)

    canvas1.pack(fill = "both", expand = True)

    # Display image
    canvas1.create_image( 0, 0, image=img, anchor = "nw")

    # Add Text
    canvas1.create_text(740,30, text = "Welcome to Your space",font=("david", 35),fill='cyan')
    #canvas1.create_text(750,365, text = "New here? Register now",font=("david", 20))
```

```
    #add button
    button1=Button(main_screen,text="Update profiles", height="2", width="25",command = update,font=("david",
 20),fg="green",bg='light grey')
    button2=Button(main_screen,text="Delete account", height="2", width="25",command=delete,font=("david"
, 20),fg="green",bg='light grey')
    button3=Button(main_screen,text="View User\'s Info", height="2", width="25",command=
 users_info,font=("david", 20),fg="green",bg='light grey')
    button4=Button(main_screen,text="Add Items", height="2", width="25",command = item,font=("david"
, 20),fg="green",bg='light grey')
    #button5=Button(main_screen,text="Place order", height="1", width="15",command=
 place_order,font=("david", 20),fg="green",bg='light grey')

    # Display Buttons
    button1_canvas = canvas1.create_window( 300, 100, anchor = "nw",window = button1)
    button2_canvas = canvas1.create_window( 300, 300,anchor = "nw",window = button2)
    button3_canvas = canvas1.create_window( 300, 500, anchor = "nw",window = button3)
    button4_canvas = canvas1.create_window( 900, 200, anchor = "nw",window = button4)
    #button5_canvas = canvas1.create_window( 300, 500, anchor = "nw",window = button5)
    main_screen.mainloop()
main_account_screen()
my.close()
```

3rd screen:

```
from tkinter import *
from tkinter import messagebox
import tkinter as tk
import tempfile
import os
import csv

def product():
    global l
    global cost1
    l=[]
    cost1=[]
    f=open('products.csv','r',newline='')
    r=csv.reader(f)
    for i in r:
        l.append(i)
        cost1.append(i[1])
    print(l)
    print('cost1=',cost1)
product()
```

```
root=tk.Toplevel()
root.title('Billing Manangement System')
root.geometry('1280x720')
bg_color='#2D9290'


#=====================variables===================
Bread=IntVar()
Wine=IntVar()
Rice=IntVar()
Gal=IntVar()
Total=IntVar()

cb=StringVar()
cw=StringVar()
cr=StringVar()
cg=StringVar()
total_cost=StringVar()
# ===========Function===============
def change():
    f=open('products.csv','r+',newline='')
    csv_r=csv.reader(f)
    csv_w=csv.writer(f)
    l=[]
    b1=[b,w,r]
    i=0
    global fraud
    fraud=0
    for rec in csv_r:
        if b1[i]>int(rec[2]):
            messagebox.showerror('Error','Sufficient stock unavailable, \nPlease visit the item list ')
            i=i+1
            fraud=1
            root.destroy()
            import bil
            break
        else:
            #for rec in csv_r:
            rec[2]=int(rec[2])-int(b1[i])
            l.append(rec)
            i+=1
    if fraud==0:
        f.seek(0)
        csv_w.writerows(l)
        f.close()
```

```
    else:
        pass


q1=[]

def total():
    global b
    global w
    global r

    for i in q:
        s=int(i.get())
        q1.append(s)

    if q1==[] or len(q1)==2 or len(q1)==1:
        messagebox.showerror('Error','Please select number of quantity')
    else:
        b=q1[0]
        w=q1[1]
        r=q1[2]

        change()

        if fraud==0:
            t=int(b)*int(cost1[0])+int(w)*int(cost1[1])+int(r)*int(cost1[2])
            print(t)
            Total.set(b + w + r)
            total_cost.set('₹ ' + str(round(t, 2)))

            cb.set('₹ '+str(round(int(int(b)*int(cost1[0])),2)))
            cw.set('₹ '+str(round(int(int(w)*int(cost1[1])),2)))
            cr.set('₹ '+str(round(int(int(r)*int(cost1[2])),2)))



            messagebox.showinfo('Success','Your order has been confirmed')

        else:
            textarea.delete(1.0,END)
            Bread.set(0)
            Wine.set(0)
            Rice.set(0)
            Total.set(0)
```

```
        cb.set('')
        cw.set('')
        cr.set('')
        total_cost.set('')

def receipt():
    textarea.delete(1.0,END)
    textarea.insert(END,' Items\tNumber of Items\t  Cost of Items\n')
    textarea.insert(END,f'\nPhone\t\t{b}\t  {cb.get()}')
    textarea.insert(END,f'\n\nlaptop\t\t{w}\t  {cw.get()}')
    textarea.insert(END,f'\n\nHDD\t\t{r}\t  {cr.get()}')
    textarea.insert(END, f"\n\n================================")
    textarea.insert(END,f'\nTotal Price\t\t{Total.get()}\t{total_cost.get()}')
    textarea.insert(END, f"\n================================")


def print1():
    with open('Bill.txt', "w", encoding="utf-8") as f:
        q=textarea.get('1.0','end-1c')
        print(q)
        f.write(str(q))


    messagebox.showinfo('Success','Your receipt is saved')
    #filename=tempfile.mktemp('.txt')
    #open(filename,'w').write(q)
    #os.startfile(filename,'Print')


'''def reset():
    textarea.delete(1.0,END)
    for i in range (len(l)):
        textvar[i].set(0)

    Total.set(0)

    cb.set('')
    cw.set('')
    cr.set('')
    cg.set('')
    total_cost.set('')'''

def exit():
    if messagebox.askyesno('Exit','Do you really want to exit'):
        root.destroy()
```

```
title=Label(root,pady=5,text="Billing Manangement System",bd=12,bg=bg_color,fg='white',font=('times new
 roman', 35 ,'bold'),relief=GROOVE,justify=CENTER)
title.pack(fill=X)

#===============Product Details=================
F1 = LabelFrame(root, text='Product Details', font=('times new romon', 18, 'bold'),
fg='gold',bg=bg_color,bd=15,relief=RIDGE)
F1.place(x=5, y=90,width=800,height=500)

#=====================Heading=========================
itm=Label(F1, text='Items', font=('Helvetic',25, 'bold','underline'), fg='black',bg=bg_color)
itm.grid(row=0,column=0,padx=20,pady=15)

n=Label(F1, text='Number of Items', font=('Helvetic',25, 'bold','underline'), fg='black',bg=bg_color)
n.grid(row=0,column=1,padx=30,pady=15)

cost=Label(F1, text='Cost of Items', font=('Helvetic',25, 'bold','underline'), fg='black',bg=bg_color)
cost.grid(row=0,column=2,padx=30,pady=15)

#===============Product============
p=1
q=[]
textvar=[cb,cw,cr]
jj=0
for j in range (len(l)):

   bread=Label(F1, text=l[j][0], font=('times new rommon',20, 'bold'), fg='lawngreen',bg=bg_color)
   bread.grid(row=p,column=0,padx=20,pady=15)

   b_txt=Entry(F1,font='arial 15 bold',relief=SUNKEN,bd=7,textvariable='',justify=CENTER)
   b_txt.grid(row=p,column=1,padx=20,pady=15)
   q.append(b_txt)
   cb_txt=Entry(F1,font='arial 15 bold',relief=SUNKEN,bd=7,textvariable=textvar[jj],justify=CENTER)
   cb_txt.grid(row=p,column=2,padx=20,pady=15)
   p+=1
   jj=jj+1

t=Label(F1, text='Total', font=('times new rommon',20, 'bold'), fg='lawngreen',bg=bg_color)
t.grid(row=5,column=0,padx=20,pady=15)
t_txt=Entry(F1,font='arial 15 bold',relief=SUNKEN,bd=7,textvariable=Total,justify=CENTER)
t_txt.grid(row=5,column=1,padx=20,pady=15)
totalcost_txt=Entry(F1,font='arial 15 bold',relief=SUNKEN,bd=7,textvariable=total_cost,justify=CENTER)
```

```
totalcost_txt.grid(row=5,column=2,padx=20,pady=15)

#=====================Bill areea====================
F2=Frame(root,relief=GROOVE,bd=10)
F2.place(x=820,y=90,width=430,height=500)
bill_title=Label(F2,text='Receipt',font='arial 15 bold',bd=7,relief=GROOVE).pack(fill=X)
scrol_y=Scrollbar(F2,orient=VERTICAL)
scrol_y.pack(side=RIGHT,fill=Y)
textarea=Text(F2,font='arial 15',yscrollcommand=scrol_y.set)
textarea.pack(fill=BOTH)
scrol_y.config(command=textarea.yview)



#=====================Buttons=======================
F3 =Frame(root,bg=bg_color,bd=15,relief=RIDGE)
F3.place(x=5, y=590,width=1270,height=120)

btn1 = Button(F3, text='Total', font='arial 25 bold', padx=5, pady=5,
bg='yellow',fg='red',width=10,command=total)
btn1.grid(row=0,column=0,padx=20,pady=10)

btn2 = Button(F3, text='Receipt', font='arial 25 bold', padx=5, pady=5,
bg='yellow',fg='red',width=10,command=receipt)
btn2.grid(row=0,column=1,padx=10,pady=10)

btn3 = Button(F3, text='Save recipt', font='arial 25 bold', padx=5, pady=5,
 bg='yellow',fg='red',width=10,command=print1)
btn3.grid(row=0,column=2,padx=10,pady=10)

'''btn4 = Button(F3, text='Reset', font='arial 25 bold', padx=5, pady=5,
bg='yellow',fg='red',width=10,command=reset)
btn4.grid(row=0,column=3,padx=10,pady=10)'''

btn5 = Button(F3, text='Exit', font='arial 25 bold', padx=5, pady=5,
bg='yellow',fg='red',width=10,command=exit)
btn5.grid(row=0,column=4,padx=10,pady=10)
root.mainloop()
4th screen
import mysql.connector

mycon=mysql.connector.connect(host='localhost',
                    user='root',
                    database='project',
                    password='')
```

```
if mycon.is_connected():
   print('Database connected')




cursor=mycon.cursor()


#creating customer table
sql='''CREATE TABLE customer
      (customer_id char(4) PRIMARY KEY,
       username varchar(50),
       password varchar(30),
       first_name varchar(100),
       last_name varCHAR(100),
       date_of_birth DATE,
       contact_number varchar(10),
       address varCHAR(200),
       city varCHAR(100),
       state varCHAR(100),
       zipcode INT(6),
       gender varchar(6));'''
cursor.execute(sql)
mycon.commit()
mycon.close()
```

5th screen

```
import csv
products=open('products.csv','w',newline='')
l=[]
l2=['laptop',900,28]
l1=['Phone',200,25]
l3=['HDD',200,24]
l.append(l1)
l.append(l2)
l.append(l3)
w=csv.writer(products)
w.writerows(l)
products.close()
```

6th screen:

```
import mysql.connector
from tkinter import *
import tkinter.messagebox as MessageBox
```

```
import tkinter as tk
mycon=mysql.connector.connect(host='localhost',
                        user='root',
                        database='project',
                        password='')
if mycon.is_connected():
   print('Database connected')
cursor=mycon.cursor()

root=Tk()
root.title('Register')
root.geometry('500x400')

#cursor=mycon.cursor()


#creating customer table
#sql='''CREATE TABLE customer
#      (customer_id char(4) PRIMARY KEY,
#       username varchar(50),
#       password varchar(30),
#       first_name varchar(100),
#       last_name varCHAR(100),
#       date_of_birth DATE,
#       contact_number varchar(10),
#       address varCHAR(200),
#       city varCHAR(100),
#       state varCHAR(100),
#       zipcode INT(6),
#       gender varchar(6));'''
#cursor.execute(sql)'''
#mycon.commit()
#mycon.close()




#creating text boxes
box_customer_id=Entry(root, width=30)
box_customer_id.grid(row=0,column=1,padx=20)

box_f_name=Entry(root, width=30)
box_f_name.grid(row=1,column=1)
```

```
box_l_name=Entry(root, width=30)
box_l_name.grid(row=2,column=1)

box_date_of_birth=Entry(root, width=30)
box_date_of_birth.grid(row=3,column=1)

box_contact_number=Entry(root, width=30)
box_contact_number.grid(row=4,column=1)

box_address=Entry(root, width=30)
box_address.grid(row=5,column=1)

box_city=Entry(root, width=30)
box_city.grid(row=6,column=1)

box_state=Entry(root, width=30)
box_state.grid(row=7,column=1)

box_zipcode=Entry(root, width=30)
box_zipcode.grid(row=8,column=1)

box_gender=Entry(root,width=30)
box_gender.grid(row=9,column=1)


box_username=Entry(root,width=30)
box_username.grid(row=11,column=1)

box_password=Entry(root,width=30)
box_password.grid(row=12,column=1)

#creating text box labels
customer_id_label=Label(root, text='CID (contact dealer for ur unique code)')
customer_id_label.grid(row=0,column=0)

f_name_label=Label(root, text='First name')
f_name_label.grid(row=1,column=0)

l_name_label=Label(root, text='last name')
l_name_label.grid(row=2,column=0)

date_of_birth_label=Label(root, text='date of birth(yyyy-mm-dd)')
date_of_birth_label.grid(row=3,column=0)

contact_number_label=Label(root, text='contact number (10 digit integer)')
```

```
contact_number_label.grid(row=4,column=0)

address_label=Label(root, text='address')
address_label.grid(row=5,column=0)

city_label=Label(root, text='city')
city_label.grid(row=6,column=0)

state_label=Label(root, text='state')
state_label.grid(row=7,column=0)

zipcode_label=Label(root, text='zipcode (6 digit integer)')
zipcode_label.grid(row=8,column=0)

gender_label=Label(root, text='Gender(M/F)')
gender_label.grid(row=9,column=0)


username_label=Label(root, text='Username')
username_label.grid(row=11,column=0)

password_label=Label(root, text='Password')
password_label.grid(row=12,column=0)


#creating submit funtion
def submit():
    customer_id=box_customer_id.get()
    f_name=box_f_name.get()
    l_name=box_f_name.get()
    date_of_birth=box_date_of_birth.get()
    contact_number=box_contact_number.get()
    address=box_address.get()
    city=box_city.get()
    state=box_state.get()
    zipcode=box_zipcode.get()
    gender=box_gender.get()
    username=box_username.get()
    password=box_password.get()
```

Log of project

LOG-1:15/6/2021

- Ideation
- Discussion of topic - Inventory

LOG-2:27/6/2021

- Searched for various modules available to store and retrieve customer data

LOG-3:10/7/2021

- Learning to work with Tkinter

LOG-4:23/7/2021

- Created SQL table Customer

LOG-5:24/7/2021

- Worked on sign up and login options using Python-SQL Connectivity

LOG-6:26/7/2021

- Worked on user options: Creating a User Defined account commands

LOG-7:30/7/2021

- Worked on admin options: Updating user profile details in SQL table Customer

LOG-8:3/8/2021

- Worked on sign up and login options using Python-SQL Connectivity
- Worked on user options: update,delete,place order button added in user home screen

LOG-9:4/8/2021

- Worked on user options: viewing product list

LOG-10:5/8/2021

- Worked on admin options: Updating user details to SQL table Customer
- Worked on admin options: Deleting user details from SQL table Customer
- Worked on admin options: Updating product details in CSV table Products

LOG-11:6/8/2021

- Worked on user options: Playing Songs from a defined Playlist

LOG-12:7/8/2021

- Worked on admin options: Integrating admin options(Add, Update and Delete)

LOG-13:14/8/2021

- Worked on user options: Integrating user options

LOG-14:1/9/2021

- Worked on user options: Search for a Song

LOG-15:8/9/2021

- Worked on user options: Delete a User Defined Playlist from SQL table Playlist

LOG-16:12/9/2021

- Discussion to include lyrics feature using text files

LOG-17:6/10/2021

- Creating Text files

LOG-17:10/10/2021

- Linking text files

LOG-17:14/10/2021

- Linking CSV files

LOG-17:17/10/2021

- Creating Database

LOG-17:21/10/2021

- Organizing files

LOG-17:2/11/2021

- Addition of background

LOG-17:6/11/2021

- Worked on user options: Introduction of lyric feature in song screen

LOG-18:9/11/2021

- Worked on integrating all the program files together

LOG-19:12/11/2021

- Worked on various bug fixes and improvements

LOG-20:15/11/2021

- Improvements in the design of GUI interface and layout
- Worked with color combinations and backgrounds

LOG-17:20/11/2021

- Font color and checking for errors

LOG-17:28/11/2021

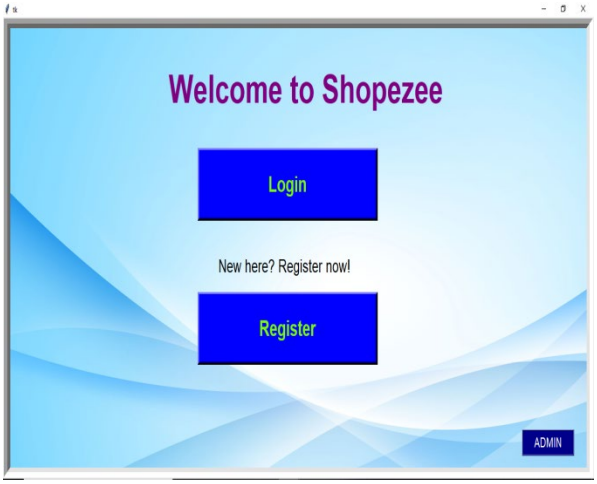
- Fixing Bugs

LOG-21:15/12/2021

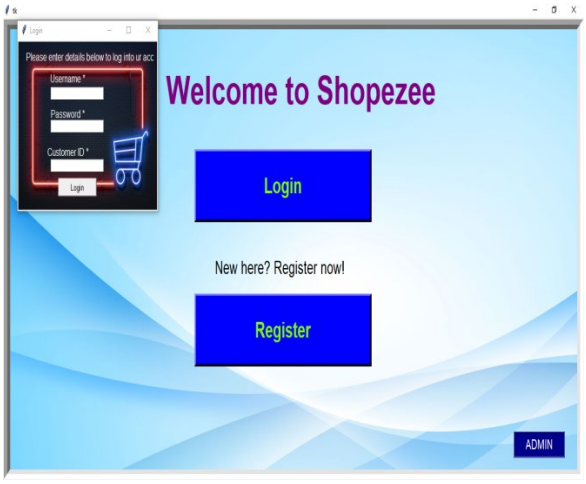
- Completion of Project and Submission

SAMPLE OUTPUT

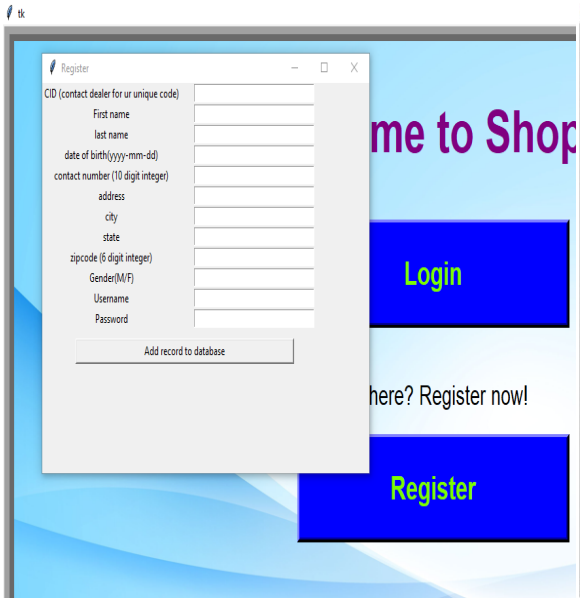
WELCOME SCREEN



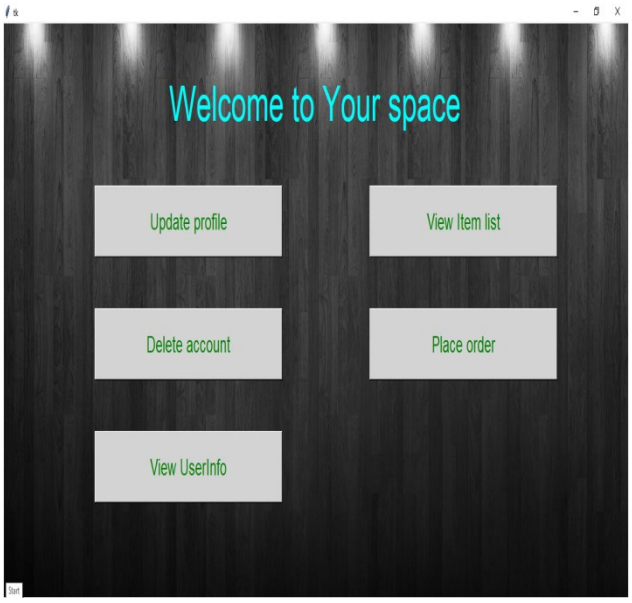
LOGIN OPTION



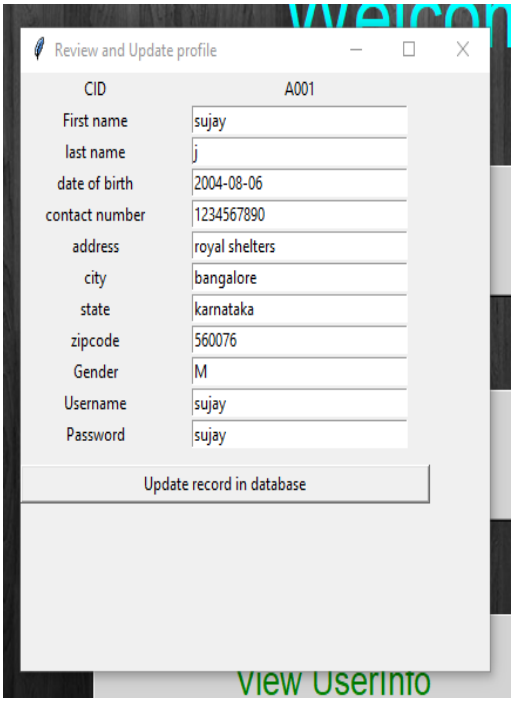
REGISTER WINDOW



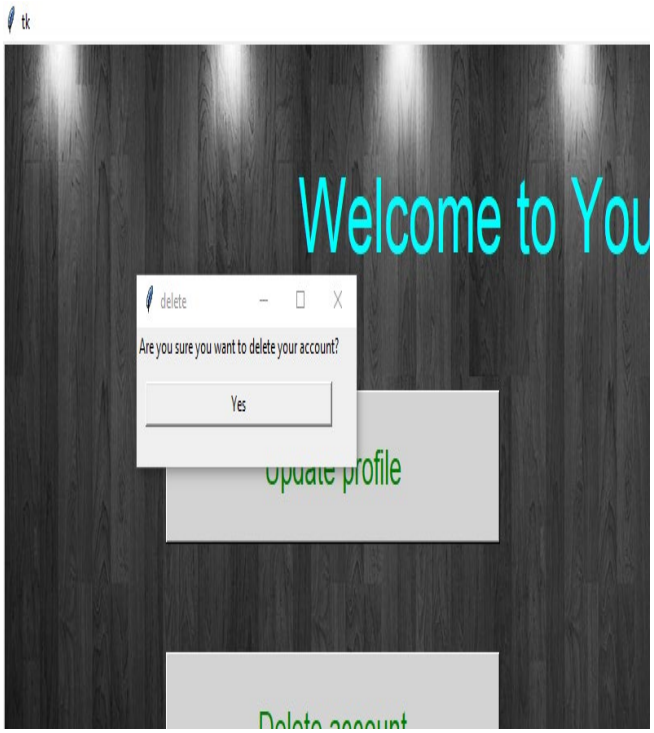
MAIN ACCOUNT WINDOW



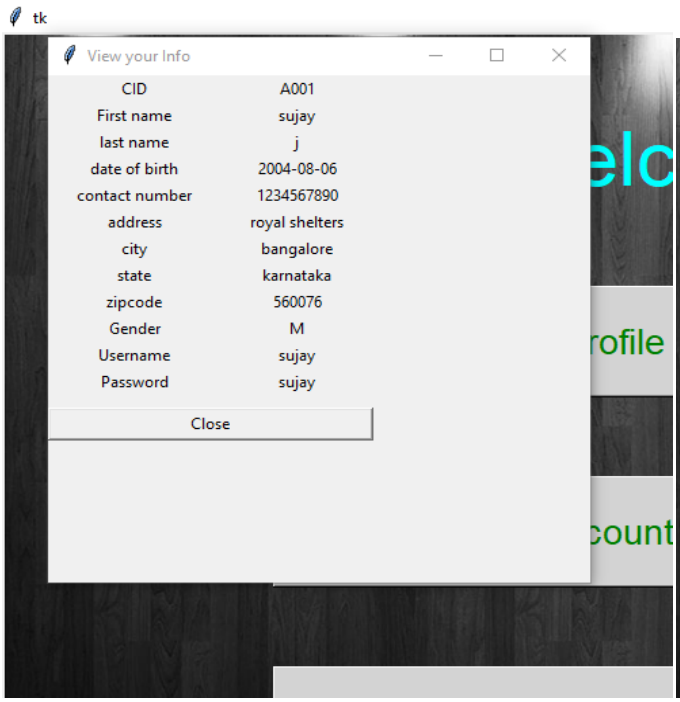
REVIEW AND UPDATE FUNCTIONALITY



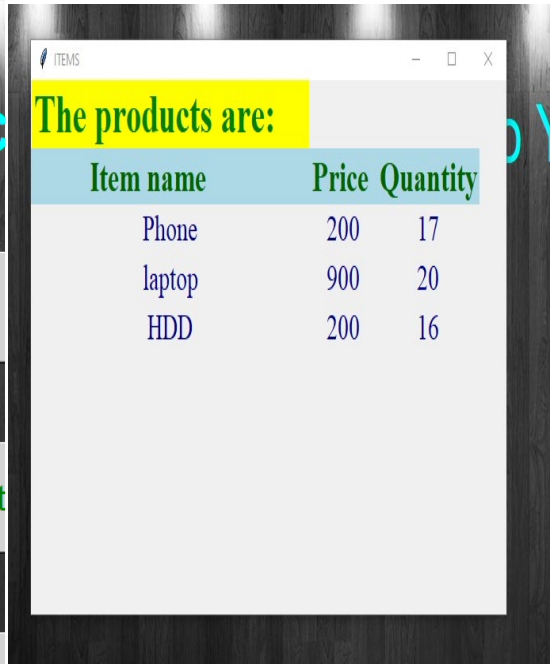
DELETE ACCOUNT



VIEW ACCOUNT CREDENTIALS



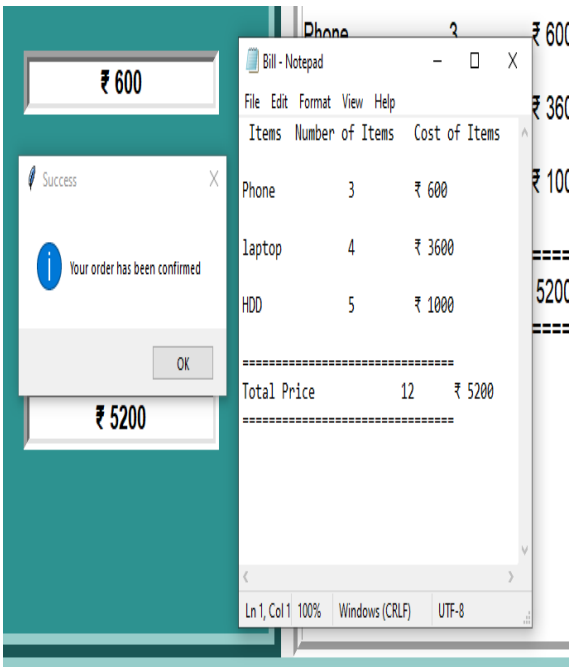
VIEWING THE INVENTORY AND PRODUCTS LIST



BILLING APPLICATION

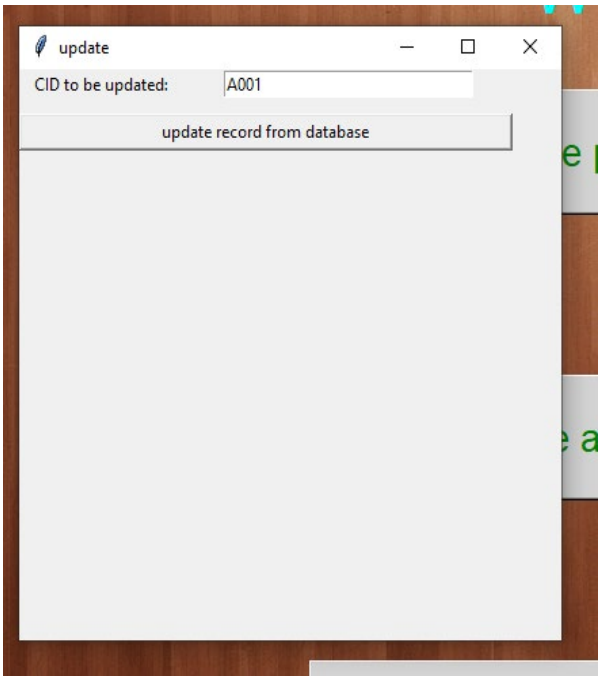
TOTAL AND RECEIPT FUNCTIONALITY

SAVING BILL AS NOTEPAD

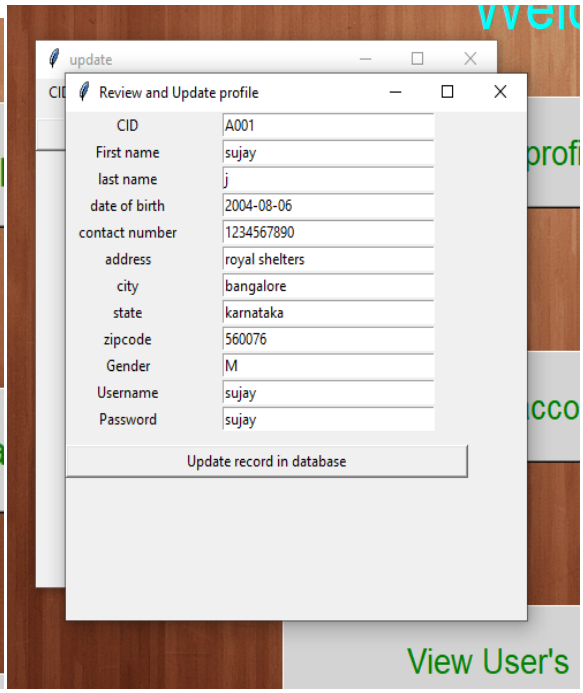


ADMIN WINDOW

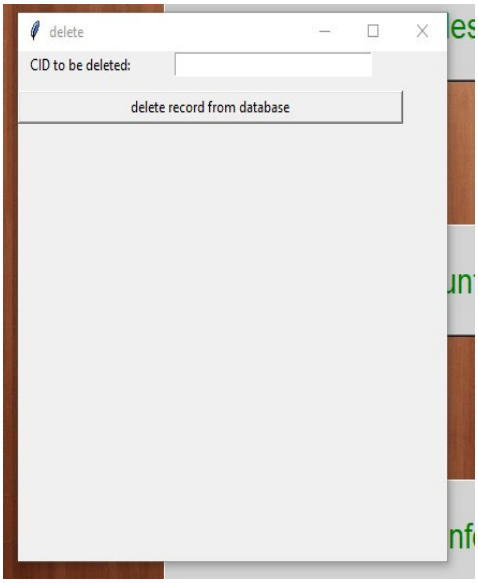
UPDATE CUSTOMER ACCOUNT WITH ID



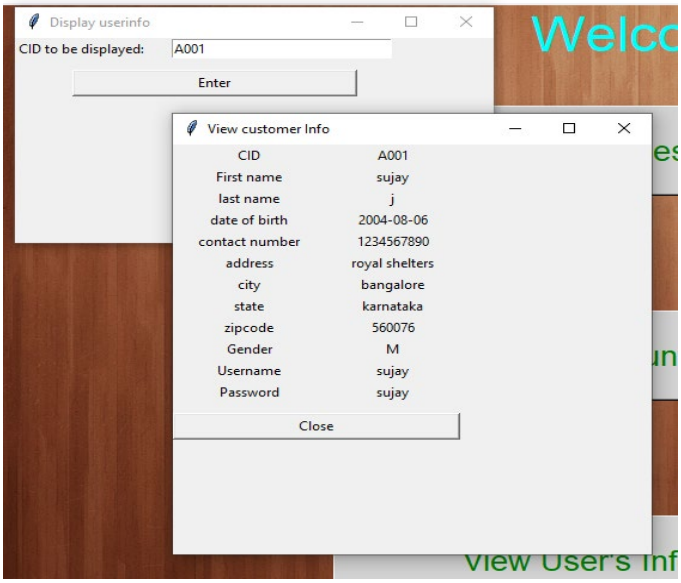
UPDATE WINDOW



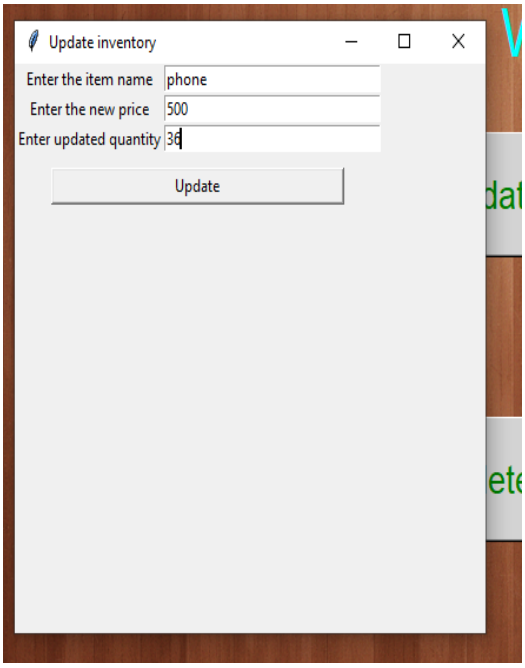
DELETING ID INPUT



VIEW CUSTOMER DETALS



UPDATE INVENTORY WINDOW



BIBLIOGRAPHY

- https://www.tutorialspoint.com/python/
- https://www.geeksforgeeks.org
- http://stackoverflow.com/
- https://www.javatpoint.com
- https://www.google.co.in/
- https://pythonexamples.org
- https://effbot.org/