SIX WEEKS SUMMER TRAINING

REPORT

on

(ADVANCE PYTHON PROGRAMMING & MINI PROJECT)

Submitted by

Siddharth Rathore (12100767)

Program Name: B. Tech CSE

Under the Guidance of

EBOX an Amphisoft Product

School of Computer Science & Engineering Lovely Professional University, Phagwara

(June-July 2022)

DECLARATION

I hereby declare that I have completed my six weeks summer training at EBOX an Amphisoft Product an Online learning platform from 27th May 2022 to 10th July 2022 under the guidance of Lovely Professional University. I have declared that I have worked with full dedication during these six weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of B.Tech CSE, Lovely Professional University, Phagwara.

Siddharth Rathore

Name of Student: Siddharth Rathore

Registration no: 12100767

Date: 8th July 2022

ACKNOWLEDGEMENT

A few typewritten words of thanks cannot really express the sincerity of my gratitude. But I am still trying to put into words my gratefulness towards all who have helped and encouraged me in carrying out this Course. This course of mine bears the imprint of many people who have an important impact on my thinking, behavior, and acts during the technology learning and mini project making.

First, I would like to take this opportunity to thank the LOVELY PROFESSIONAL UNIVERSITY for having Summer Training as a part of the B. Tech CSE degree. The accomplishment of this course otherwise would have been painstaking endeavor, for lack of staunch and sincere support of the School of Computer Science and Engineering, LPU. The incessant and undeterred succors extended by the mentors of the EBOX Platform explained the course to the great extent. If this goes unnoticed and unacknowledged it would be selfishness.

Many people have influenced the shape and content of this course, and many supported me throughout. I express my sincere gratitude to EBOX Team, who was available for help whenever I required, their guidance, gentle persuasion and active support has made it possible to complete this course.

In the end, I can say only this much that “ALL ARE NOT MENTIONED BUT NONE IS FORGOTTEN”

Last but not the least I would like to thank GOD, who continues to look after us despite all my flaws.

SUMMER TRAINING CERTIFICATE

(From Training Institute)

TABLE OF CONTENTS

1. INTRODUCTION

EBOX Advance Python Programming

This Report describes about my technical learning and project development from Advance Python Programming by EBOX Learning. EBOX learning is an Amphisoft product which provides various programming courses. In my course "Learn Advance Python", we have 9 topics, and we have total number of 55 hours of video lecture which contains 87 competitive coding-based questions also contains 256 quiz questions, every topic has some subtopics. Each topic has videos, quizzes, and coding-based questions.

Further, this report includes the objective of mine during this technology learning. Implementation of Advance python gave me confidence to work more on mini project. I learnt about rules and methods in advance python.

By learning the topics of advance python by EBOX, I had developed one useful mini project based on billing system for retailer or shopkeeper i.e., Billing software.

This software project is windows-based application. Billing system using python.

Science and technology with all its fascinating advancements has been taking human life standards to the next level. The whole world will be literate with these innovations. This project is an innovation, which makes the way of generating the invoice simple compared to the manual calculations. The main modules available in this project are Payment's modules which manages the functionality of Payments. Transaction History is normally used for managing Transaction History.

Transaction contains all the functionalities related to Transaction, Login functionality, Customer has all the features of Customer and BILLs module manages the functionality of Bills.

As we know python projects are trending topics for academic python project development. so, I had chosen python3 for developing Billing System. In this project we developed features for Billing, Transaction history, Customer etc., which reduces the human efforts and increase the efficiency. Scope of python language is growing day by day.

Billing Software

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and

fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner.

It provides fast, efficient, and convenient manipulation and distribution of works. Somehow, general store and hardware relies in manual approach which results to some disadvantages such as: delayed transactions, consumed much time for calculation and inaccurate sales reports.

For the above-mentioned disadvantages providing a billing system is a solution, to the problem. It provides faster and accurate manipulation that contribute to a much reliable and efficient work.

This billing system focus on the development of an information system that will automate manual transaction in Beatriz Food and Cafe.

However, the study has focused on the following:

- It will generate receipt on every transaction inputted to the system.
- The software will display view of calculations of every transaction.
- For security and privacy of the management, the Billing System comply two log-in users with different access level.
- The system will store and recognize customer reservations.

2. PROFILE OF THE PROBLEM

In the current system, we have employees working for different clients on different projects. The accountant calculates all the working hours of employees. But if he makes any mistake in data entry of calculating the hours then a wrong report about payment is sent to the client and it may over cost the client. The client again must recheck everything and sent it back to the consultancy for re-evaluation or sometimes the employee does not get the correct pay. This helps to generate an invoice which is done manually. The user has identified these problems:

- Generation of the Invoice is very slow and takes a long time to remit to the employee.
- The generation reports took even longer than invoice.
- Delay in all the reports due to manual work and there are possible human errors which may cost the client.
- Manual work is expensive
- There is no way to find a duplicate entry of data.

3. EXISTING SYSTEM

In the existing system, we don't have perfect billing management system as we can see our small vendors who are manually cutting or generating low quality bills which is using papers and by using/wasting lots of paper we are destroying our environment.

Paper bills can be lost by the customer. Also, we have employees working for different clients on different projects. The accountant calculates all the working hours of employees. But if he makes any mistake in data entry of calculating the hours then a wrong report about payment is sent to the client and it may over cost the client. The client again must recheck everything and sent it back to the consultancy for re-evaluation or sometimes the employee does not get the correct pay. This helps to generate an invoice which is done manually. This document presents a detailed explanation of the objectives, features, user interface and application of Billing software System in real life.it will also describe how the system will perform and under which it must operate. In this document it will be also shown user interface. Both the stakeholder and the developer of the system can benefit from this document.

4. PROBLEM ANALYSIS

Product definition

The Billing software System helps the shop manager to manage the shop more effectively and efficiently by computerizing product ordering.

Billing and inventory control. The system processes transaction and stores the resulting data. Reports will be generated from These data which help the manager to make appropriate business decisions for the store.

Scope:

This system will help to manage and run the business systematically. In this Management system, we will provide an app that can be used by the customers to buy product. Customers can also give feedback through this app. So that owner of the shop can evaluate the whole system.

Customers can Also make payment through debit or credit cards using POS which will be integrated with the Management software. Customers can see current discount facility of the shop. Customers can also see the rate chart which will increase consciousness about their budget.

All the information about daily expenses and profit will be saved in the system. Also, their required information's about employees will be saved in the system which can be only accessed by the system admin.

- From customer point of view this is a software which is very easy to use.
- The software is standalone.
- The user interface of the text to speech is simple as we use in python idle, as the GUI based output is not used in the software.
- Generation of bill.
- Detail information about customer.
- Information about product and items sold in shop.
- Less wastage of paper.
- Less time taking for the calculation.

Feasibility Analysis

- **Functional Requirements**
  - Maintain company information:

    The store manager maintains the company information in the system which has details about a customer like its name, phone No.
  - Maintain project data:

    The company should have the proper information about the project such as client id which is provided by the company, project number, its name, start date, and end date, status, the name of the manager, name of the client, and budget for the project completion.
  - Maintain client data:

    The company should the client details such as project id, the name of the client, address of the company, email id, contact information, invoice frequency, billing terms, invoice grouping.
  - Generate Invoice:

    Create an invoice based on the following criteria,

    ✓ Input labour charge within the budget goal.

    ✓ Project data that contains the Client*, Project number, Project name, Start_Date*, End_Date*, Status, Project Manager and Budget*.

- ✓ There are employees who work overtime when needed.

- Generate Reports:
  1. Invoice Report
  2. Project Report
  3. Budget Report

Non-Functional Requirements

a. Usability
   - The system must be easy to use so that user can easily perform any actions.
   - A User should be able to create an invoice without any difficulty.
   - A user should be able to do that inaccurate time.
   - A user should be able to effectively operate the system with less than one hour of training.

b. Availability
   - The system must be highly reliable since, if the system is not available, the user can't easily be able to create an invoice.
   - The system should be available 99.999% of the office open hours.

c. Performance
   - All the actions should be performed in accurate time.
   - All the imports should be performed inaccurate time.
   - Generation of Invoice should be performed inaccurate time.
   - Generation of reports should be performed inaccurate time.

d. Supportability
   - The system should be developed in a common technology that the Accountants in the Consulting company should be able to use to make upgrades to the system
   - The system should be documented and coded in a way that a developer that was not originally on the development team could determine how to make updates.

e. Interface
   - The system must support an interface with some users.
   - The system must support a file exchange interface.

- The system interface should provide multiple accesses.
- Accountants, Project Manager, and Employees will operate and use this system.

f. Operations: In-office personnel will use the system. These users are not trained computer operators.

g. Legal

- The system should meet legal and OC security requirements for people data.
- No one should have access to specific People data.
- Copyright protected.
- Trade protected.

5. SOFTWARE REQUIREMENT ANALYSIS

General Description:

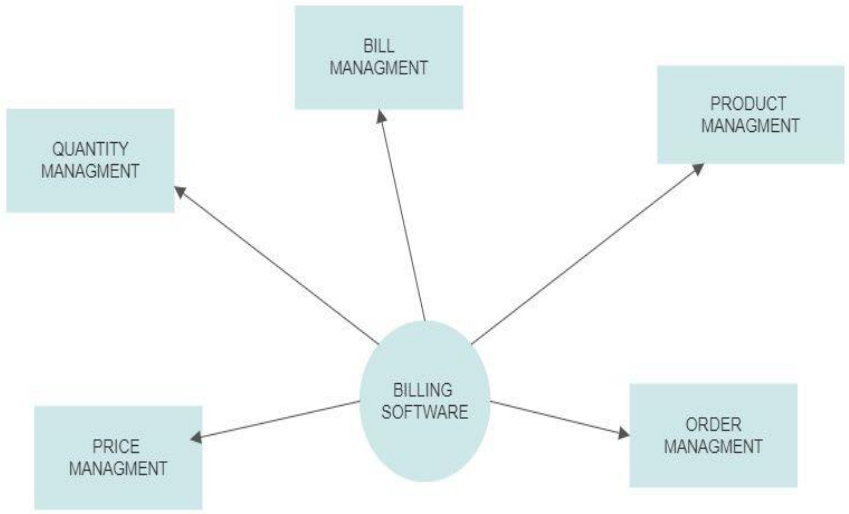
- VS Code: Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.
- Language: python
- It needs to use PC to generate bill according to order in this system. Which will running on Windows Operating System.
- Tkinter: It is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface.

Specific Requirements:

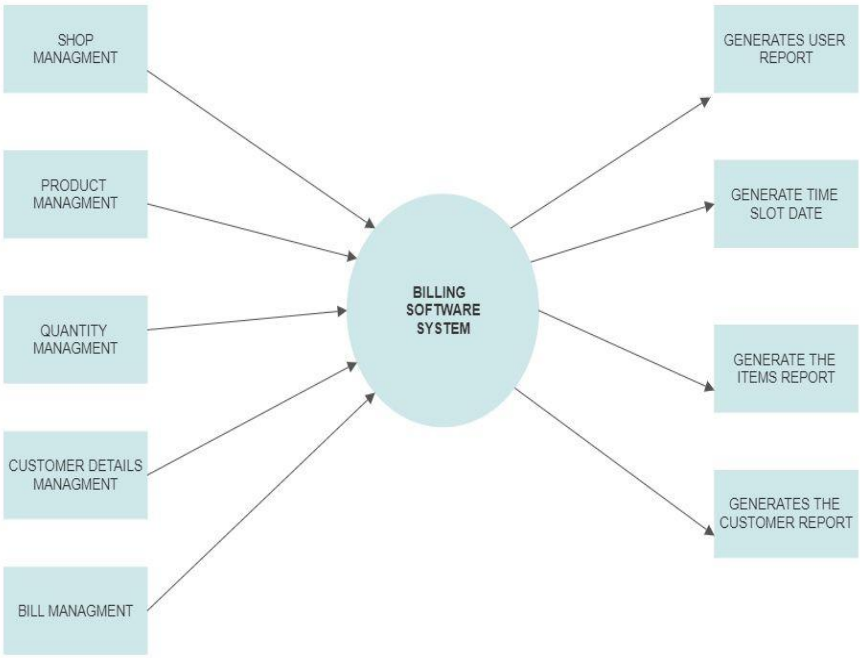
- Operating System: Windows7, 8, 8.1, 10 & 11
- Back end: Random Library
- Front end: Tkinter
- Editor: VS Code (python extension) and Notepad

6. DESIGN

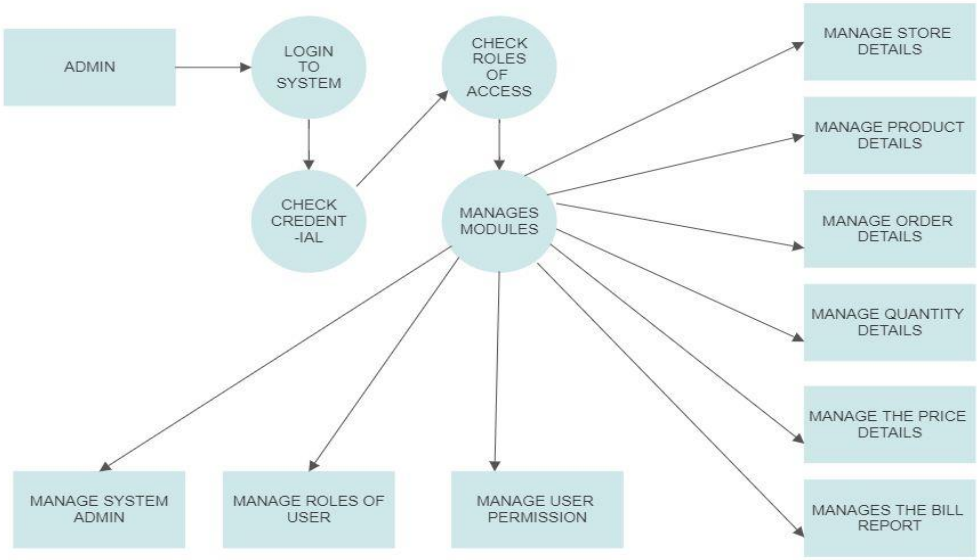
a. DFD (Data Flow Diagram)



ZERO LEVEL DFD - BILLING SOFTWARE

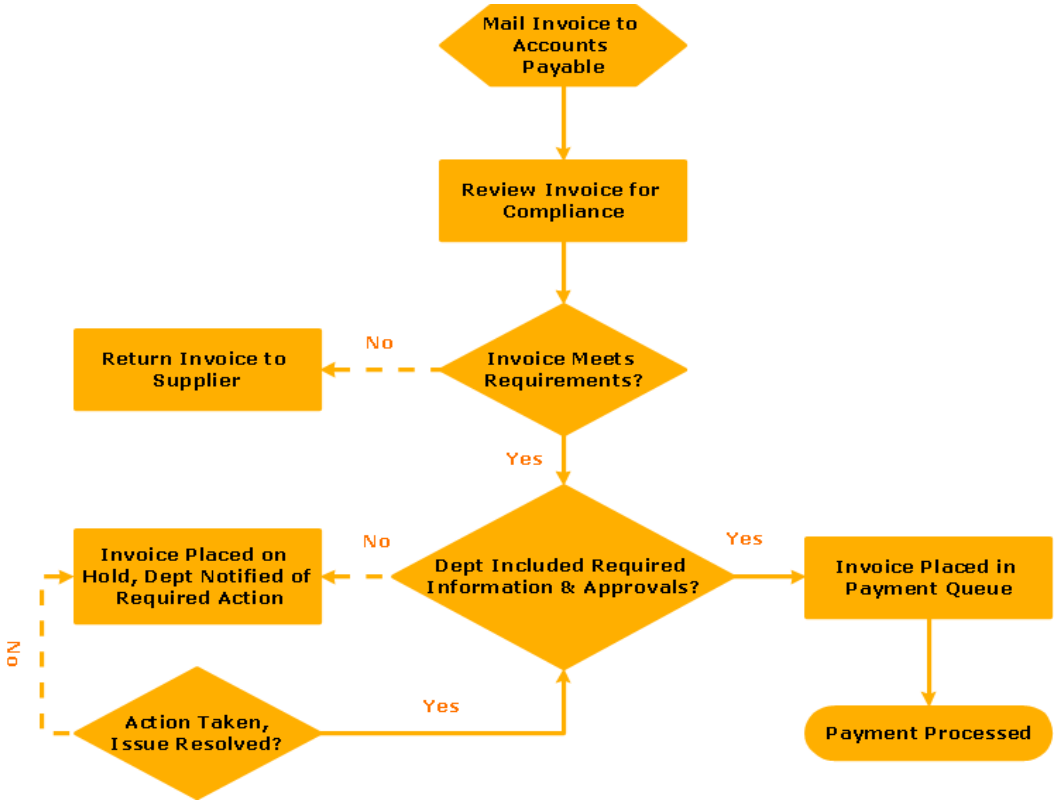


FIRST LEVEL DFD - BILLING SOFTWARE

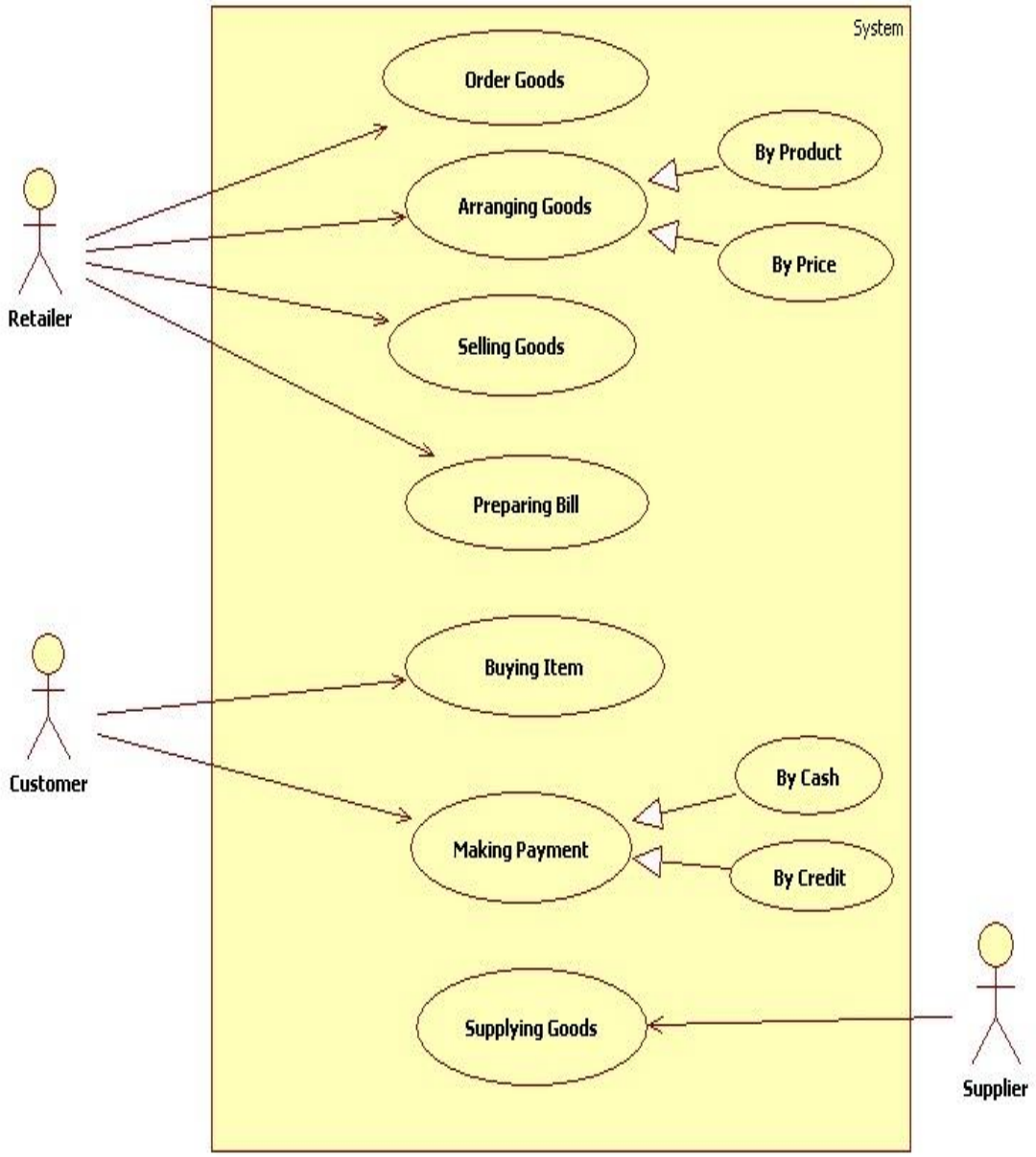


SECOND LEVEL DFD - BILLING SOFTWARE

b. Flowchart



c. **UML Diagram:**



7. TESTING

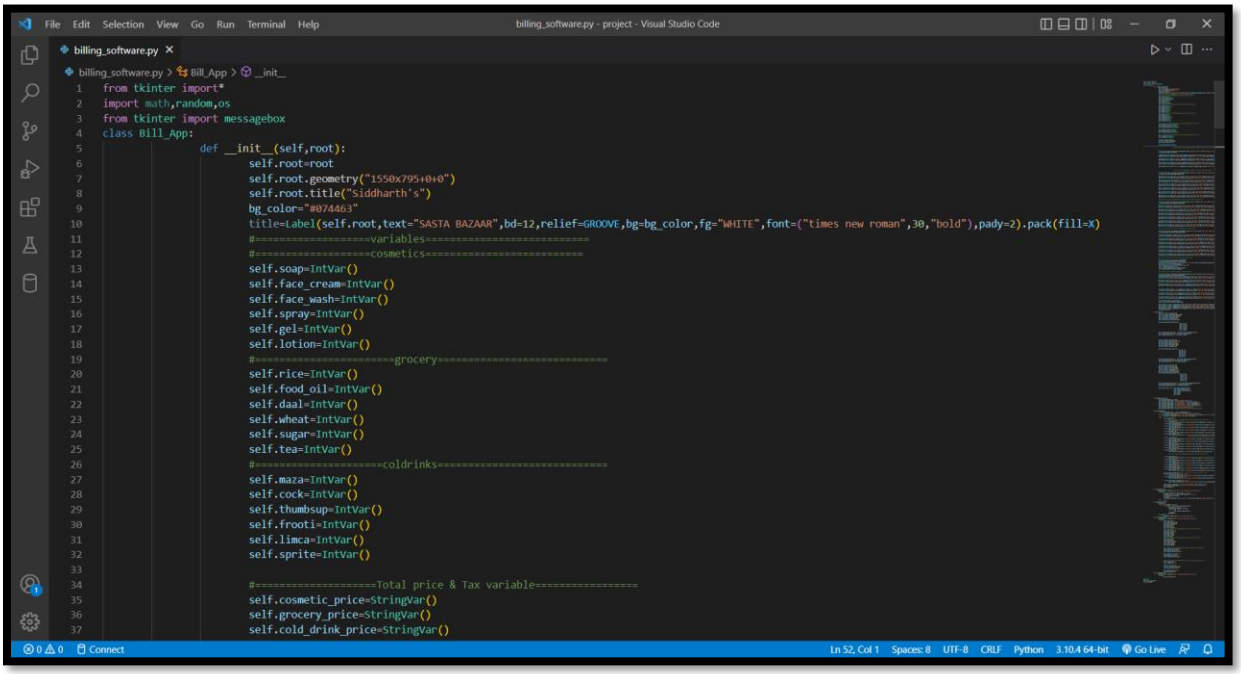
In this phase, when the Retailer will open the software he will be on the interface of billing software where retailer will have to feed Customer Details like Customer Name, Phone Number, Bill Number then feed the data in quantity that which and how many of the products he have bought, Products like Cosmetics, Grocery & Beverages then the retailer will see the application will automatically have entered the data of how much quantity of that product the customer have bought

then he have to press Total Button to get the total amount of the product and then have to click on the generate Bill button so that he'll provide the valid bill to the customer. After this process retailer can check or fetch the older bill of any customers by their phone number as well as bill number.

8. IMPLEMENTATION

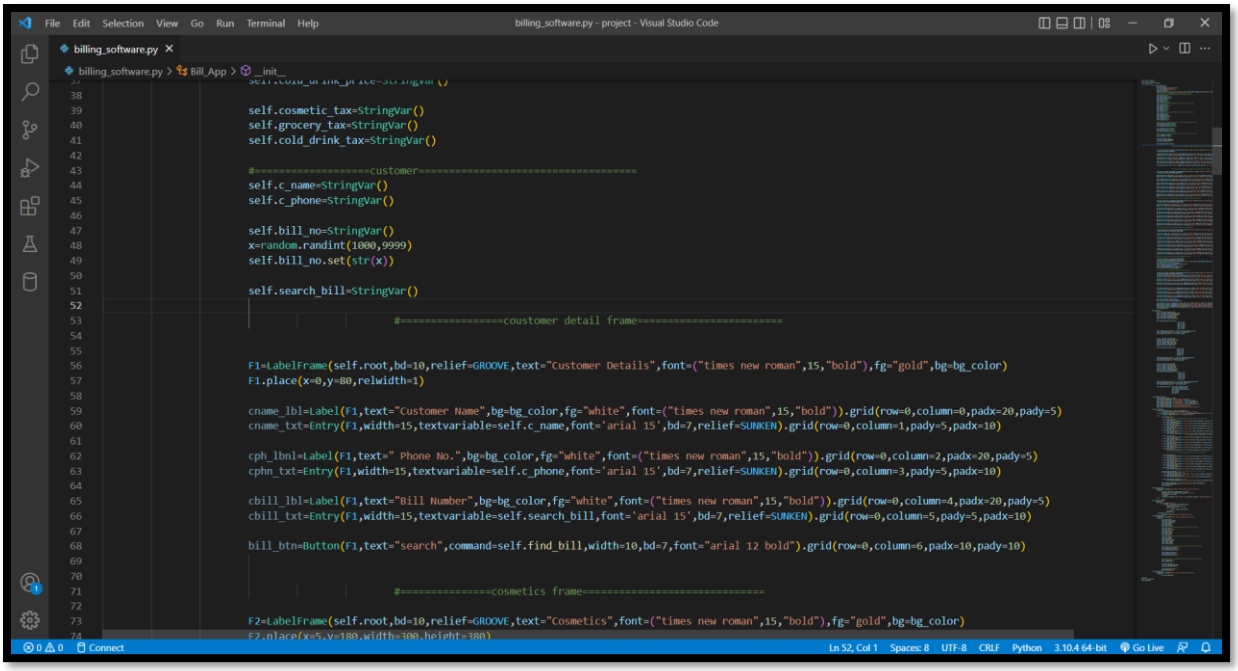
Code

The following is the screenshot of the project name "Billing Software" in python language, the code runs smoothly, and the screenshot of the program output is ahead.



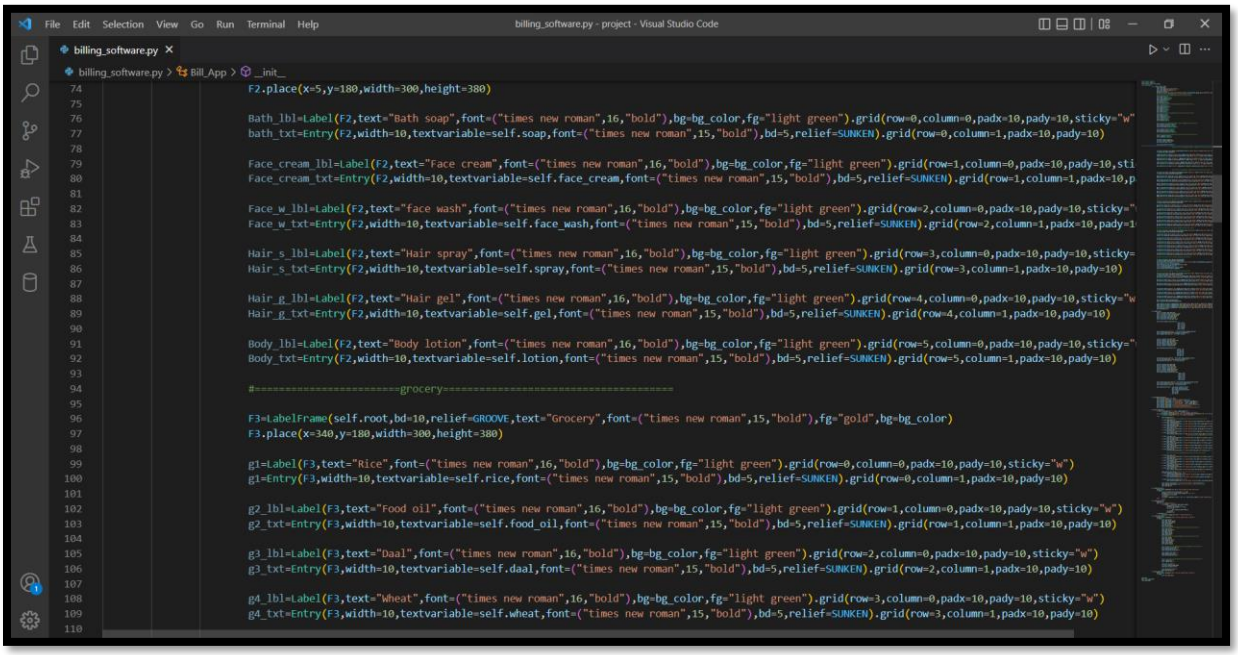
```
from tkinter import*
import math,random,os
from tkinter import messagebox
class Bill_App:
        def __init__(self,root):
                self.root=root
                self.root.geometry("1550x795+0+0")
                self.root.title("Siddharth's")
                bg_color="#074463"
                title=Label(self.root,text="SASTA BAZAAR",bd=12,relief=GROOVE,bg=bg_color,fg="WHITE",font=("times new roman",30,"bold"),pady=2).pack(fill=X)
                #===================variables==========================
                #===================cosmetics==========================
                self.soap=IntVar()
                self.face_cream=IntVar()
                self.face_wash=IntVar()
                self.spray=IntVar()
                self.gel=IntVar()
                self.lotion=IntVar()
                #=======================grocery===========================
                self.rice=IntVar()
                self.food_oil=IntVar()
                self.daal=IntVar()
                self.wheat=IntVar()
                self.sugar=IntVar()
                self.tea=IntVar()
                #=====================coldrinks==========================
                self.maza=IntVar()
                self.cock=IntVar()
                self.thumbsup=IntVar()
                self.frooti=IntVar()
                self.limca=IntVar()
                self.sprite=IntVar()

                #====================Total price & Tax variable================
                self.cosmetic_price=StringVar()
                self.grocery_price=StringVar()
                self.cold_drink_price=StringVar()
```

Fig- 8.1(C)



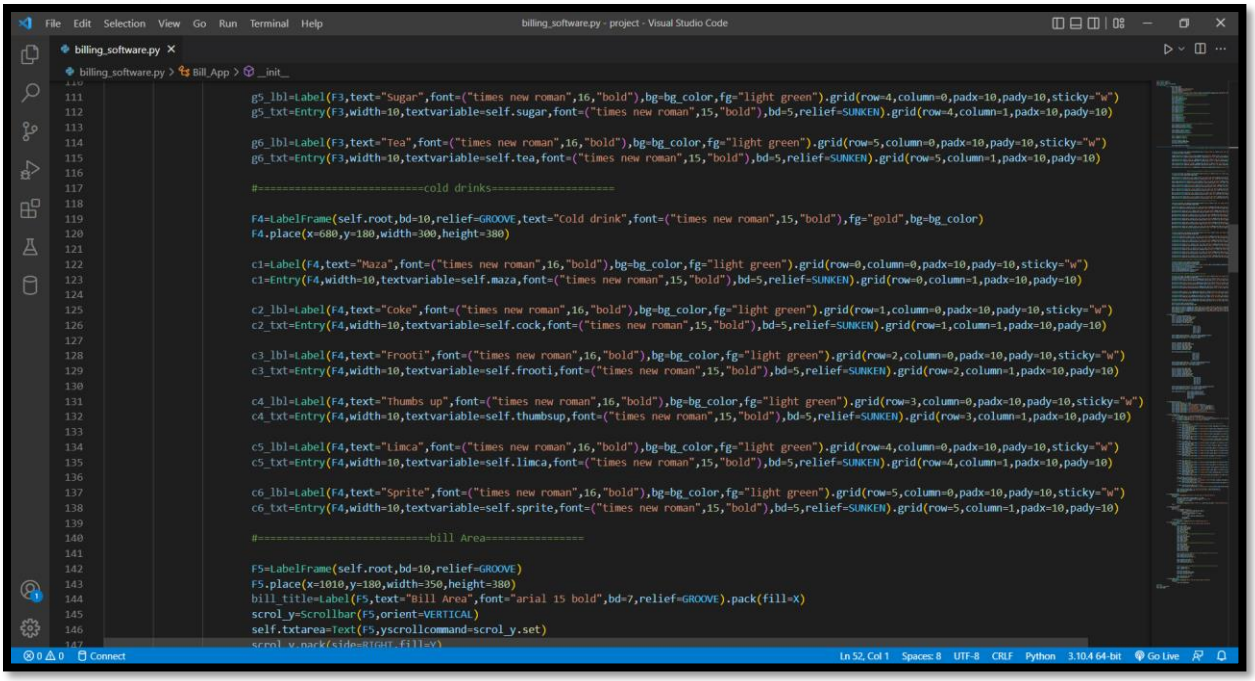
```
billing_software.py > Bill_App > __init__
                self.cosmetic_tax=StringVar()
                self.grocery_tax=StringVar()
                self.cold_drink_tax=StringVar()

                #==================customer==================================
                self.c_name=StringVar()
                self.c_phone=StringVar()

                self.bill_no=StringVar()
                x=random.randint(1000,9999)
                self.bill_no.set(str(x))

                self.search_bill=StringVar()

                        #================coustomer detail frame=======================


                F1=LabelFrame(self.root,bd=10,relief=GROOVE,text="Customer Details",font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
                F1.place(x=0,y=80,relwidth=1)

                cname_lbl=Label(F1,text="Customer Name",bg=bg_color,fg="white",font=("times new roman",15,"bold")).grid(row=0,column=0,padx=20,pady=5)
                cname_txt=Entry(F1,width=15,textvariable=self.c_name,font='arial 15',bd=7,relief=SUNKEN).grid(row=0,column=1,pady=5,padx=10)

                cph_lbnl=Label(F1,text=" Phone No.",bg=bg_color,fg="white",font=("times new roman",15,"bold")).grid(row=0,column=2,padx=20,pady=5)
                cphn_txt=Entry(F1,width=15,textvariable=self.c_phone,font='arial 15',bd=7,relief=SUNKEN).grid(row=0,column=3,pady=5,padx=10)

                cbill_lbl=Label(F1,text="Bill Number",bg=bg_color,fg="white",font=("times new roman",15,"bold")).grid(row=0,column=4,padx=20,pady=5)
                cbill_txt=Entry(F1,width=15,textvariable=self.search_bill,font='arial 15',bd=7,relief=SUNKEN).grid(row=0,column=5,pady=5,padx=10)

                bill_btn=Button(F1,text="search",command=self.find_bill,width=10,bd=7,font="arial 12 bold").grid(row=0,column=6,padx=10,pady=10)


                        #==============cosmetics frame============================

                F2=LabelFrame(self.root,bd=10,relief=GROOVE,text="Cosmetics",font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
                F2.place(x=5,y=180,width=300,height=380)
```

Fig- 8.2(C)



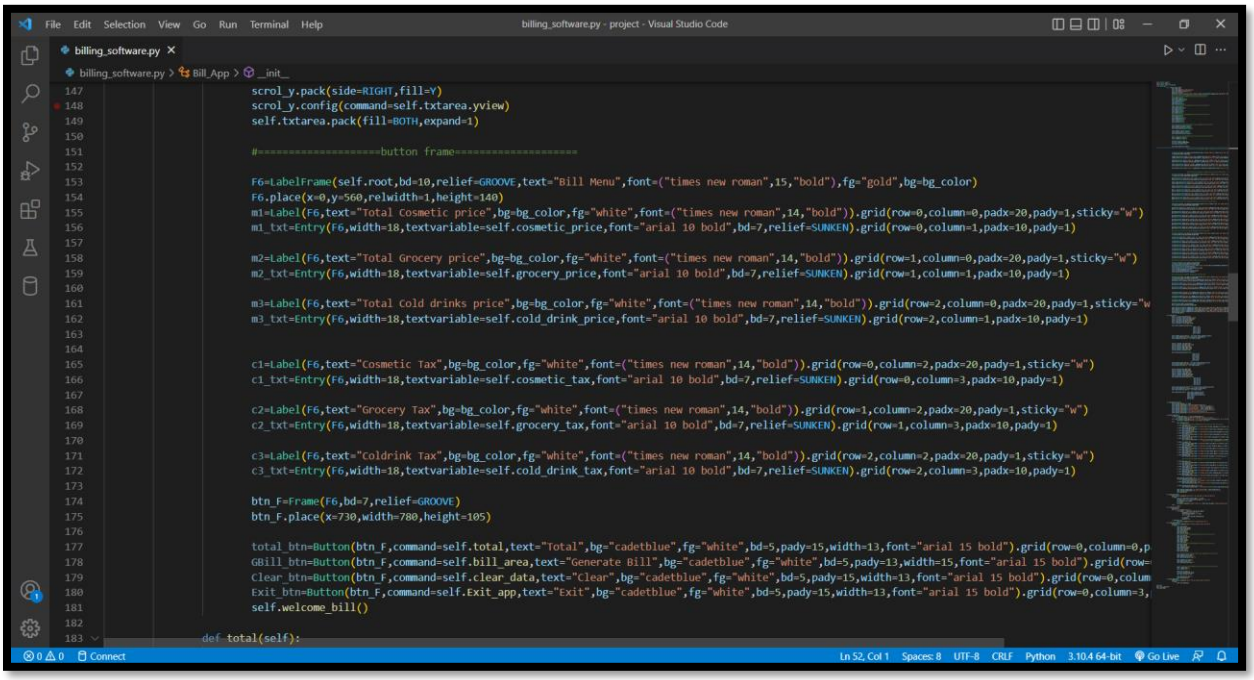
```
billing_software.py > Bill_App > __init__
                F2.place(x=5,y=180,width=300,height=380)

                Bath_lbl=Label(F2,text="Bath soap",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=0,column=0,padx=10,pady=10,sticky="w"
                bath_txt=Entry(F2,width=10,textvariable=self.soap,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=0,column=1,padx=10,pady=10)

                Face_cream_lbl=Label(F2,text="Face cream",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=1,column=0,padx=10,pady=10,sti
                Face_cream_txt=Entry(F2,width=10,textvariable=self.face_cream,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=1,column=1,padx=10,p

                Face_w_lbl=Label(F2,text="face wash",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=2,column=0,padx=10,pady=10,sticky="
                Face_w_txt=Entry(F2,width=10,textvariable=self.face_wash,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=2,column=1,padx=10,pady=1

                Hair_s_lbl=Label(F2,text="Hair spray",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=3,column=0,padx=10,pady=10,sticky=
                Hair_s_txt=Entry(F2,width=10,textvariable=self.spray,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=3,column=1,padx=10,pady=10)

                Hair_g_lbl=Label(F2,text="Hair gel",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=4,column=0,padx=10,pady=10,sticky="w
                Hair_g_txt=Entry(F2,width=10,textvariable=self.gel,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=4,column=1,padx=10,pady=10)

                Body_lbl=Label(F2,text="Body lotion",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=5,column=0,padx=10,pady=10,sticky="
                Body_txt=Entry(F2,width=10,textvariable=self.lotion,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=5,column=1,padx=10,pady=10)

                #=======================grocery==================================

                F3=LabelFrame(self.root,bd=10,relief=GROOVE,text="Grocery",font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
                F3.place(x=340,y=180,width=300,height=380)

                g1=Label(F3,text="Rice",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=0,column=0,padx=10,pady=10,sticky="w")
                g1=Entry(F3,width=10,textvariable=self.rice,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=0,column=1,padx=10,pady=10)

                g2_lbl=Label(F3,text="Food oil",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=1,column=0,padx=10,pady=10,sticky="w")
                g2_txt=Entry(F3,width=10,textvariable=self.food_oil,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=1,column=1,padx=10,pady=10)

                g3_lbl=Label(F3,text="Daal",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=2,column=0,padx=10,pady=10,sticky="w")
                g3_txt=Entry(F3,width=10,textvariable=self.daal,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=2,column=1,padx=10,pady=10)

                g4_lbl=Label(F3,text="Wheat",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=3,column=0,padx=10,pady=10,sticky="w")
                g4_txt=Entry(F3,width=10,textvariable=self.wheat,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=3,column=1,padx=10,pady=10)
```

Fig- 8.3(C)



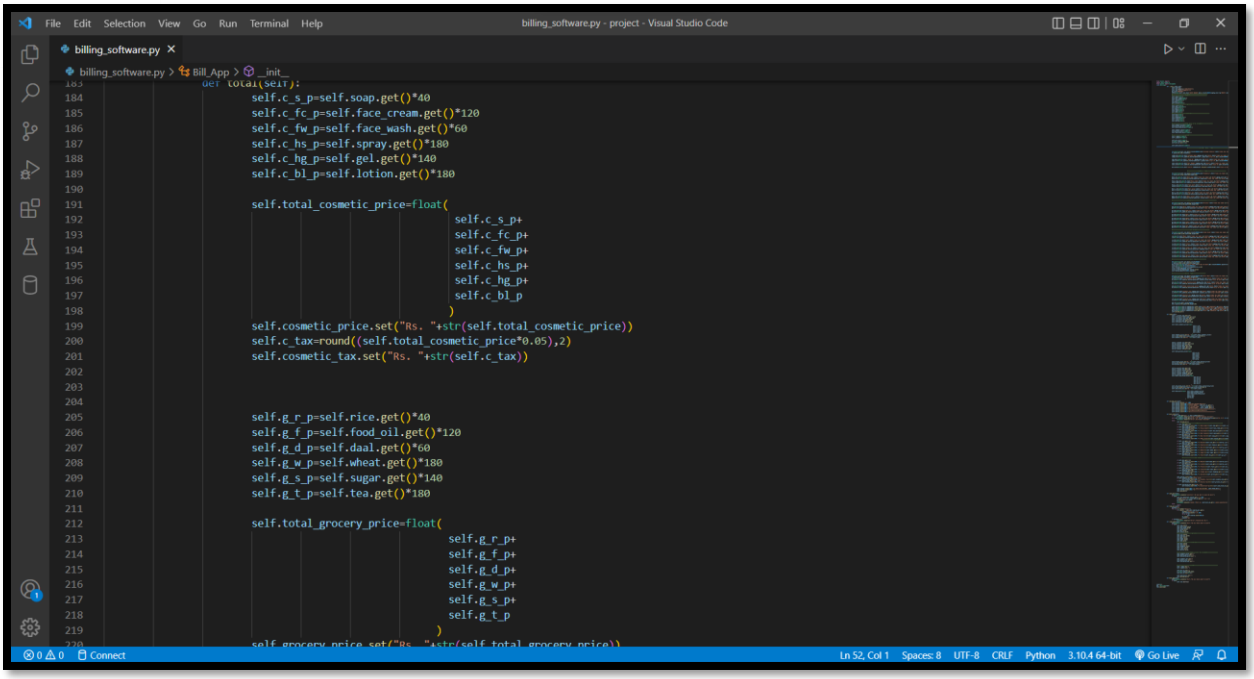
```
g5_lbl=Label(F3,text="Sugar",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=4,column=0,padx=10,pady=10,sticky="w")
g5_txt=Entry(F3,width=10,textvariable=self.sugar,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=4,column=1,padx=10,pady=10)

g6_lbl=Label(F3,text="Tea",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=5,column=0,padx=10,pady=10,sticky="w")
g6_txt=Entry(F3,width=10,textvariable=self.tea,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=5,column=1,padx=10,pady=10)

#==========================cold drinks====================

F4=LabelFrame(self.root,bd=10,relief=GROOVE,text="Cold drink",font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
F4.place(x=680,y=180,width=300,height=380)

c1=Label(F4,text="Maza",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=0,column=0,padx=10,pady=10,sticky="w")
c1=Entry(F4,width=10,textvariable=self.maza,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=0,column=1,padx=10,pady=10)

c2_lbl=Label(F4,text="Coke",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=1,column=0,padx=10,pady=10,sticky="w")
c2_txt=Entry(F4,width=10,textvariable=self.cock,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=1,column=1,padx=10,pady=10)

c3_lbl=Label(F4,text="Frooti",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=2,column=0,padx=10,pady=10,sticky="w")
c3_txt=Entry(F4,width=10,textvariable=self.frooti,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=2,column=1,padx=10,pady=10)

c4_lbl=Label(F4,text="Thumbs up",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=3,column=0,padx=10,pady=10,sticky="w")
c4_txt=Entry(F4,width=10,textvariable=self.thumbsup,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=3,column=1,padx=10,pady=10)

c5_lbl=Label(F4,text="Limca",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=4,column=0,padx=10,pady=10,sticky="w")
c5_txt=Entry(F4,width=10,textvariable=self.limca,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=4,column=1,padx=10,pady=10)

c6_lbl=Label(F4,text="Sprite",font=("times new roman",16,"bold"),bg=bg_color,fg="light green").grid(row=5,column=0,padx=10,pady=10,sticky="w")
c6_txt=Entry(F4,width=10,textvariable=self.sprite,font=("times new roman",15,"bold"),bd=5,relief=SUNKEN).grid(row=5,column=1,padx=10,pady=10)

#==========================bill Area===============

F5=LabelFrame(self.root,bd=10,relief=GROOVE)
F5.place(x=1010,y=180,width=350,height=380)
bill_title=Label(F5,text="Bill Area",font="arial 15 bold",bd=7,relief=GROOVE).pack(fill=X)
scrol_y=Scrollbar(F5,orient=VERTICAL)
self.txtarea=Text(F5,yscrollcommand=scrol_y.set)
```

Fig- 8.4(C)



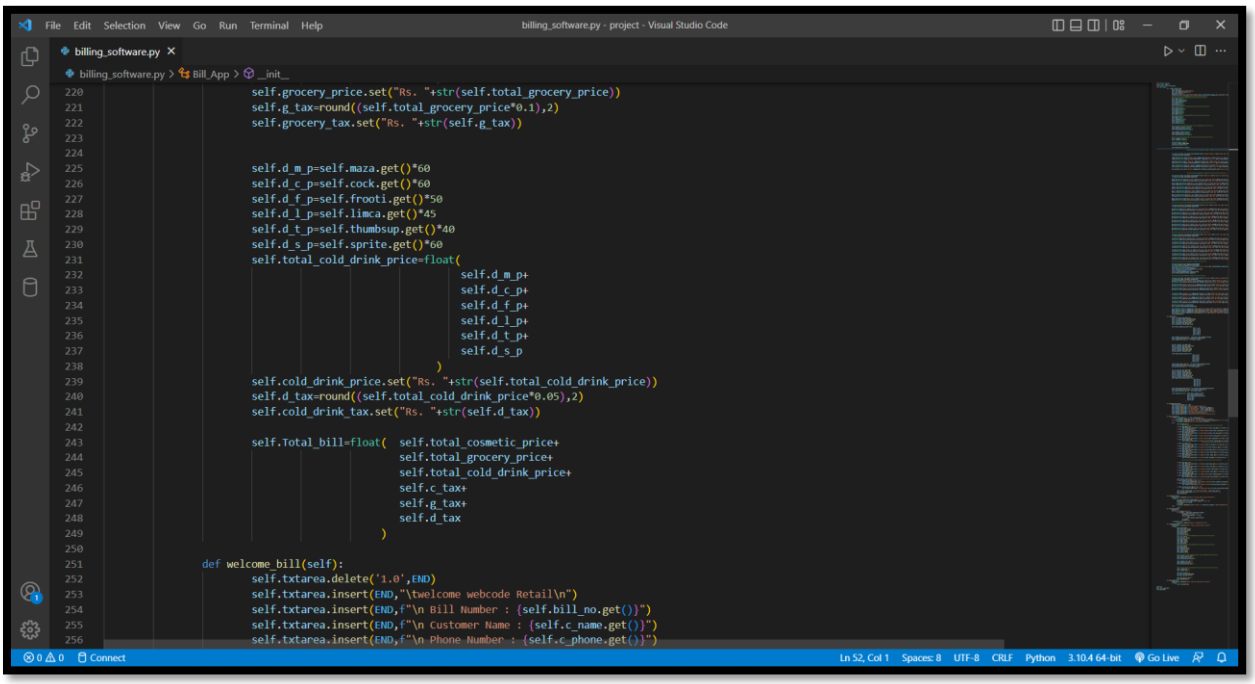
```
scrol_y.pack(side=RIGHT,fill=Y)
scrol_y.config(command=self.txtarea.yview)
self.txtarea.pack(fill=BOTH,expand=1)

#===================button frame===================

F6=LabelFrame(self.root,bd=10,relief=GROOVE,text="Bill Menu",font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
F6.place(x=0,y=560,relwidth=1,height=140)
m1=Label(F6,text="Total Cosmetic price",bg=bg_color,fg="white",font=("times new roman",14,"bold")).grid(row=0,column=0,padx=20,pady=1,sticky="w")
m1_txt=Entry(F6,width=18,textvariable=self.cosmetic_price,font="arial 10 bold",bd=7,relief=SUNKEN).grid(row=0,column=1,padx=10,pady=1)

m2=Label(F6,text="Total Grocery price",bg=bg_color,fg="white",font=("times new roman",14,"bold")).grid(row=1,column=0,padx=20,pady=1,sticky="w")
m2_txt=Entry(F6,width=18,textvariable=self.grocery_price,font="arial 10 bold",bd=7,relief=SUNKEN).grid(row=1,column=1,padx=10,pady=1)

m3=Label(F6,text="Total Cold drinks price",bg=bg_color,fg="white",font=("times new roman",14,"bold")).grid(row=2,column=0,padx=20,pady=1,sticky="w
m3_txt=Entry(F6,width=18,textvariable=self.cold_drink_price,font="arial 10 bold",bd=7,relief=SUNKEN).grid(row=2,column=1,padx=10,pady=1)


c1=Label(F6,text="Cosmetic Tax",bg=bg_color,fg="white",font=("times new roman",14,"bold")).grid(row=0,column=2,padx=20,pady=1,sticky="w")
c1_txt=Entry(F6,width=18,textvariable=self.cosmetic_tax,font="arial 10 bold",bd=7,relief=SUNKEN).grid(row=0,column=3,padx=10,pady=1)

c2=Label(F6,text="Grocery Tax",bg=bg_color,fg="white",font=("times new roman",14,"bold")).grid(row=1,column=2,padx=20,pady=1,sticky="w")
c2_txt=Entry(F6,width=18,textvariable=self.grocery_tax,font="arial 10 bold",bd=7,relief=SUNKEN).grid(row=1,column=3,padx=10,pady=1)

c3=Label(F6,text="Coldrink Tax",bg=bg_color,fg="white",font=("times new roman",14,"bold")).grid(row=2,column=2,padx=20,pady=1,sticky="w")
c3_txt=Entry(F6,width=18,textvariable=self.cold_drink_tax,font="arial 10 bold",bd=7,relief=SUNKEN).grid(row=2,column=3,padx=10,pady=1)

btn_F=Frame(F6,bd=7,relief=GROOVE)
btn_F.place(x=730,width=780,height=105)

total_btn=Button(btn_F,command=self.total,text="Total",bg="cadetblue",fg="white",bd=5,pady=15,width=13,font="arial 15 bold").grid(row=0,column=0,p
GBill_btn=Button(btn_F,command=self.bill_area,text="Generate Bill",bg="cadetblue",fg="white",bd=5,pady=13,width=15,font="arial 15 bold").grid(row=
Clear_btn=Button(btn_F,command=self.clear_data,text="Clear",bg="cadetblue",fg="white",bd=5,pady=15,width=13,font="arial 15 bold").grid(row=0,colum
Exit_btn=Button(btn_F,command=self.Exit_app,text="Exit",bg="cadetblue",fg="white",bd=5,pady=15,width=13,font="arial 15 bold").grid(row=0,column=3,
self.welcome_bill()

def total(self):
```

Fig- 8.5(C)



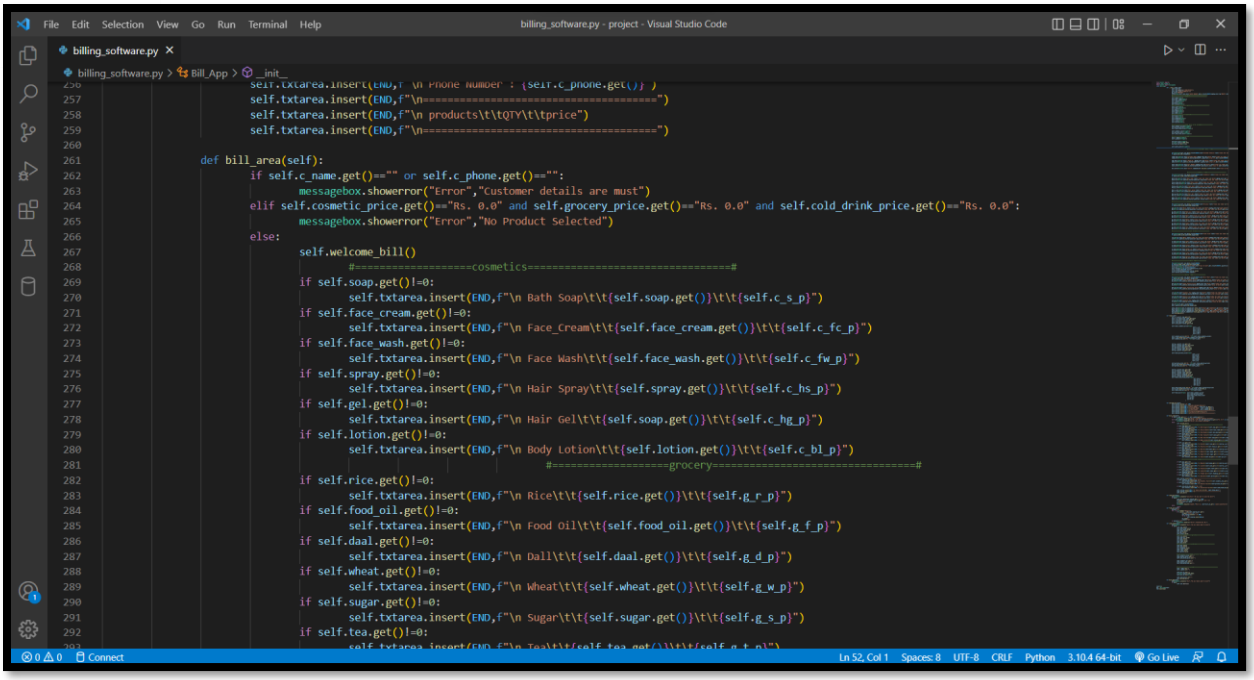
```python
        def total(self):
            self.c_s_p=self.soap.get()*40
            self.c_fc_p=self.face_cream.get()*120
            self.c_fw_p=self.face_wash.get()*60
            self.c_hs_p=self.spray.get()*180
            self.c_hg_p=self.gel.get()*140
            self.c_bl_p=self.lotion.get()*180

            self.total_cosmetic_price=float(
                                            self.c_s_p+
                                            self.c_fc_p+
                                            self.c_fw_p+
                                            self.c_hs_p+
                                            self.c_hg_p+
                                            self.c_bl_p
                                           )
            self.cosmetic_price.set("Rs. "+str(self.total_cosmetic_price))
            self.c_tax=round((self.total_cosmetic_price*0.05),2)
            self.cosmetic_tax.set("Rs. "+str(self.c_tax))



            self.g_r_p=self.rice.get()*40
            self.g_f_p=self.food_oil.get()*120
            self.g_d_p=self.daal.get()*60
            self.g_w_p=self.wheat.get()*180
            self.g_s_p=self.sugar.get()*140
            self.g_t_p=self.tea.get()*180

            self.total_grocery_price=float(
                                           self.g_r_p+
                                           self.g_f_p+
                                           self.g_d_p+
                                           self.g_w_p+
                                           self.g_s_p+
                                           self.g_t_p
                                          )
            self.grocery_price.set("Rs. "+str(self.total_grocery_price))
```

Fig- 8.6(C)



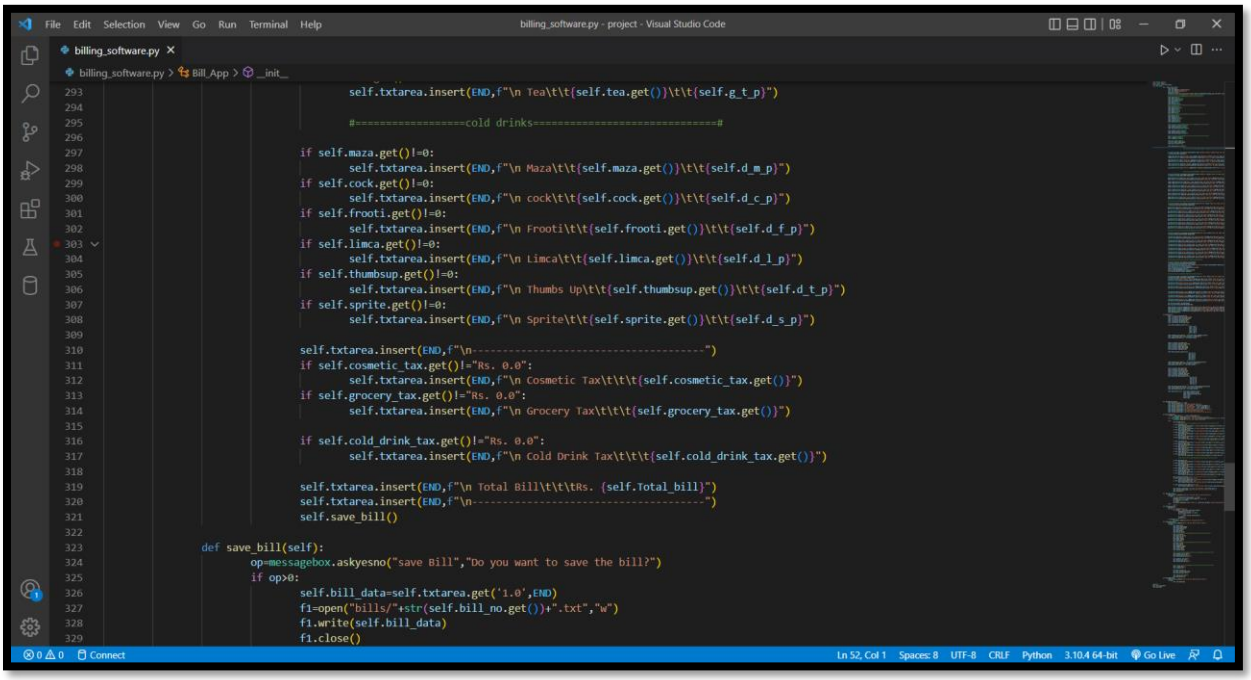
```python
            self.grocery_price.set("Rs. "+str(self.total_grocery_price))
            self.g_tax=round((self.total_grocery_price*0.1),2)
            self.grocery_tax.set("Rs. "+str(self.g_tax))


            self.d_m_p=self.maza.get()*60
            self.d_c_p=self.cock.get()*60
            self.d_f_p=self.frooti.get()*50
            self.d_l_p=self.limca.get()*45
            self.d_t_p=self.thumbsup.get()*40
            self.d_s_p=self.sprite.get()*60
            self.total_cold_drink_price=float(
                                              self.d_m_p+
                                              self.d_c_p+
                                              self.d_f_p+
                                              self.d_l_p+
                                              self.d_t_p+
                                              self.d_s_p
                                             )
            self.cold_drink_price.set("Rs. "+str(self.total_cold_drink_price))
            self.d_tax=round((self.total_cold_drink_price*0.05),2)
            self.cold_drink_tax.set("Rs. "+str(self.d_tax))

            self.Total_bill=float(  self.total_cosmetic_price+
                                    self.total_grocery_price+
                                    self.total_cold_drink_price+
                                    self.c_tax+
                                    self.g_tax+
                                    self.d_tax
                                  )

        def welcome_bill(self):
            self.txtarea.delete('1.0',END)
            self.txtarea.insert(END,"\twelcome webcode Retail\n")
            self.txtarea.insert(END,f"\n Bill Number : {self.bill_no.get()}")
            self.txtarea.insert(END,f"\n Customer Name : {self.c_name.get()}")
            self.txtarea.insert(END,f"\n Phone Number : {self.c_phone.get()}")
```

Fig- 8.7(C)



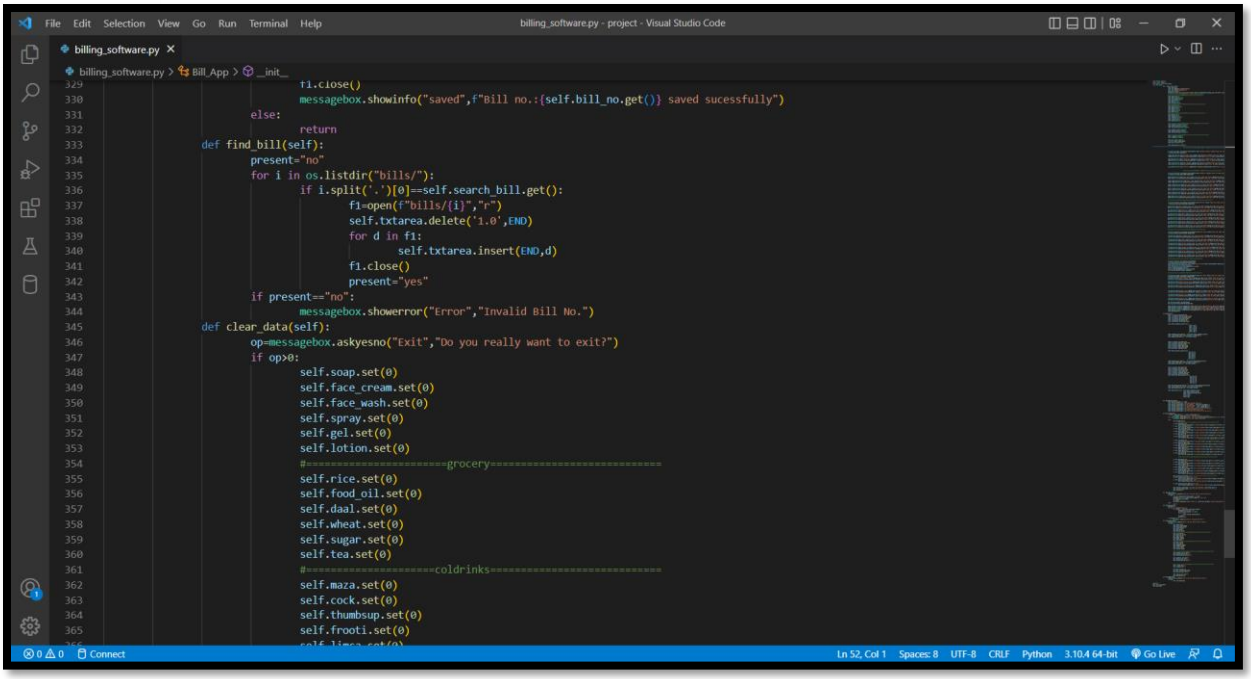
```
billing_software.py > Bill_App > __init__
                self.txtarea.insert(END,f"\n Phone Number : {self.c_phone.get()} )
                self.txtarea.insert(END,f"\n====================================")
                self.txtarea.insert(END,f"\n products\t\tQTY\t\tprice")
                self.txtarea.insert(END,f"\n====================================")

        def bill_area(self):
                if self.c_name.get()== "" or self.c_phone.get()== "":
                        messagebox.showerror("Error","Customer details are must")
                elif self.cosmetic_price.get()=="Rs. 0.0" and self.grocery_price.get()=="Rs. 0.0" and self.cold_drink_price.get()=="Rs. 0.0":
                        messagebox.showerror("Error","No Product Selected")
                else:
                        self.welcome_bill()
                                #==================cosmetics==============================#
                        if self.soap.get()!=0:
                                self.txtarea.insert(END,f"\n Bath Soap\t\t{self.soap.get()}\t\t{self.c_s_p}")
                        if self.face_cream.get()!=0:
                                self.txtarea.insert(END,f"\n Face_Cream\t\t{self.face_cream.get()}\t\t{self.c_fc_p}")
                        if self.face_wash.get()!=0:
                                self.txtarea.insert(END,f"\n Face Wash\t\t{self.face_wash.get()}\t\t{self.c_fw_p}")
                        if self.spray.get()!=0:
                                self.txtarea.insert(END,f"\n Hair Spray\t\t{self.spray.get()}\t\t{self.c_hs_p}")
                        if self.gel.get()!=0:
                                self.txtarea.insert(END,f"\n Hair Gel\t\t{self.soap.get()}\t\t{self.c_hg_p}")
                        if self.lotion.get()!=0:
                                self.txtarea.insert(END,f"\n Body Lotion\t\t{self.lotion.get()}\t\t{self.c_bl_p}")
                                                #==================grocery==============================#
                        if self.rice.get()!=0:
                                self.txtarea.insert(END,f"\n Rice\t\t{self.rice.get()}\t\t{self.g_r_p}")
                        if self.food_oil.get()!=0:
                                self.txtarea.insert(END,f"\n Food Oil\t\t{self.food_oil.get()}\t\t{self.g_f_p}")
                        if self.daal.get()!=0:
                                self.txtarea.insert(END,f"\n Dall\t\t{self.daal.get()}\t\t{self.g_d_p}")
                        if self.wheat.get()!=0:
                                self.txtarea.insert(END,f"\n Wheat\t\t{self.wheat.get()}\t\t{self.g_w_p}")
                        if self.sugar.get()!=0:
                                self.txtarea.insert(END,f"\n Sugar\t\t{self.sugar.get()}\t\t{self.g_s_p}")
                        if self.tea.get()!=0:
```

Fig- 8.8(C)



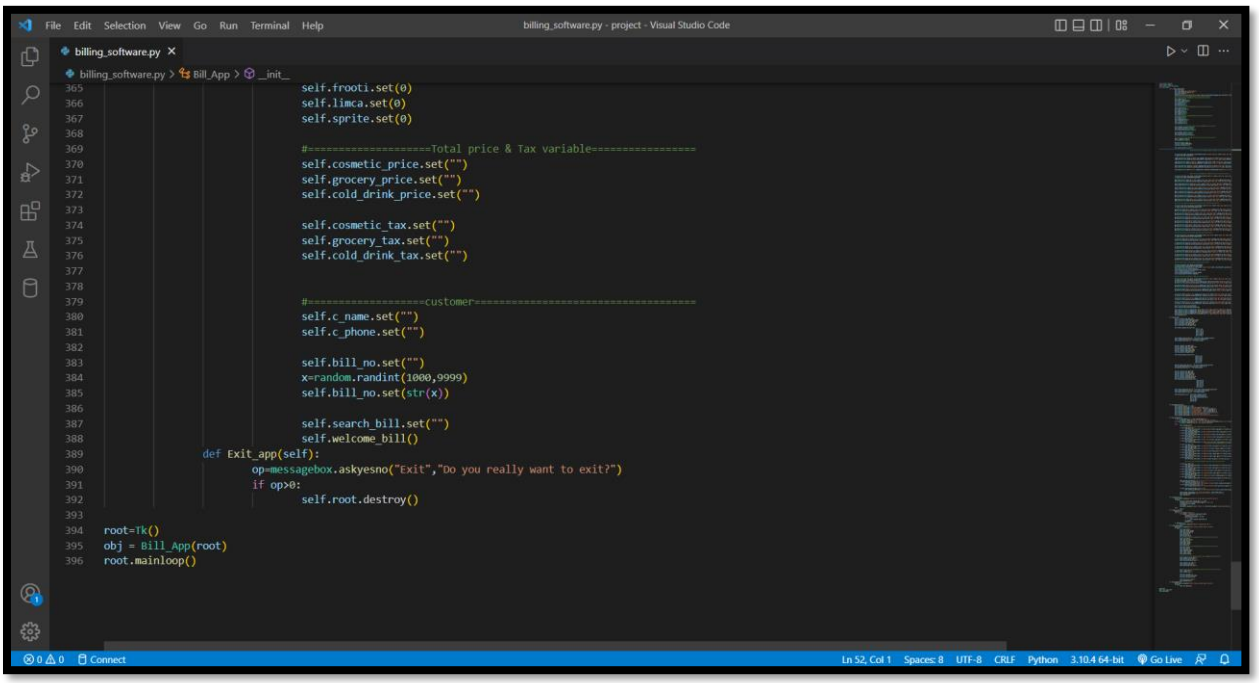
```
billing_software.py > Bill_App > __init__
                        self.txtarea.insert(END,f"\n Tea\t\t{self.tea.get()}\t\t{self.g_t_p}")

                        #==================cold drinks==============================#

                        if self.maza.get()!=0:
                                self.txtarea.insert(END,f"\n Maza\t\t{self.maza.get()}\t\t{self.d_m_p}")
                        if self.cock.get()!=0:
                                self.txtarea.insert(END,f"\n cock\t\t{self.cock.get()}\t\t{self.d_c_p}")
                        if self.frooti.get()!=0:
                                self.txtarea.insert(END,f"\n Frooti\t\t{self.frooti.get()}\t\t{self.d_f_p}")
                        if self.limca.get()!=0:
                                self.txtarea.insert(END,f"\n Limca\t\t{self.limca.get()}\t\t{self.d_l_p}")
                        if self.thumbsup.get()!=0:
                                self.txtarea.insert(END,f"\n Thumbs Up\t\t{self.thumbsup.get()}\t\t{self.d_t_p}")
                        if self.sprite.get()!=0:
                                self.txtarea.insert(END,f"\n Sprite\t\t{self.sprite.get()}\t\t{self.d_s_p}")

                        self.txtarea.insert(END,f"\n------------------------------------")
                        if self.cosmetic_tax.get()!="Rs. 0.0":
                                self.txtarea.insert(END,f"\n Cosmetic Tax\t\t\t{self.cosmetic_tax.get()}")
                        if self.grocery_tax.get()!="Rs. 0.0":
                                self.txtarea.insert(END,f"\n Grocery Tax\t\t\t{self.grocery_tax.get()}")

                        if self.cold_drink_tax.get()!="Rs. 0.0":
                                self.txtarea.insert(END,f"\n Cold Drink Tax\t\t\t{self.cold_drink_tax.get()}")

                        self.txtarea.insert(END,f"\n Total Bill\t\t\tRs. {self.Total_bill}")
                        self.txtarea.insert(END,f"\n------------------------------------")
                        self.save_bill()

        def save_bill(self):
                op=messagebox.askyesno("save Bill","Do you want to save the bill?")
                if op>0:
                        self.bill_data=self.txtarea.get('1.0',END)
                        f1=open("bills/"+str(self.bill_no.get())+".txt","w")
                        f1.write(self.bill_data)
                        f1.close()
```

Fig- 8.9(C)



```
                        f1.close()
                        messagebox.showinfo("saved",f"Bill no.:{self.bill_no.get()} saved sucessfully")
                else:
                        return
        def find_bill(self):
                present="no"
                for i in os.listdir("bills/"):
                        if i.split('.')[0]==self.search_bill.get():
                                f1=open(f"bills/{i}","r")
                                self.txtarea.delete('1.0',END)
                                for d in f1:
                                        self.txtarea.insert(END,d)
                                f1.close()
                                present="yes"
                if present=="no":
                        messagebox.showerror("Error","Invalid Bill No.")
        def clear_data(self):
                op=messagebox.askyesno("Exit","Do you really want to exit?")
                if op>0:
                        self.soap.set(0)
                        self.face_cream.set(0)
                        self.face_wash.set(0)
                        self.spray.set(0)
                        self.gel.set(0)
                        self.lotion.set(0)
                        #=======================grocery==========================
                        self.rice.set(0)
                        self.food_oil.set(0)
                        self.daal.set(0)
                        self.wheat.set(0)
                        self.sugar.set(0)
                        self.tea.set(0)
                        #====================coldrinks==========================
                        self.maza.set(0)
                        self.cock.set(0)
                        self.thumbsup.set(0)
                        self.frooti.set(0)
```

Fig- 8.10(C)



```
                        self.frooti.set(0)
                        self.limca.set(0)
                        self.sprite.set(0)

                        #==================Total price & Tax variable================
                        self.cosmetic_price.set("")
                        self.grocery_price.set("")
                        self.cold_drink_price.set("")

                        self.cosmetic_tax.set("")
                        self.grocery_tax.set("")
                        self.cold_drink_tax.set("")


                        #==================customer==================================
                        self.c_name.set("")
                        self.c_phone.set("")

                        self.bill_no.set("")
                        x=random.randint(1000,9999)
                        self.bill_no.set(str(x))

                        self.search_bill.set("")
                        self.welcome_bill()
        def Exit_app(self):
                op=messagebox.askyesno("Exit","Do you really want to exit?")
                if op>0:
                        self.root.destroy()

root=Tk()
obj = Bill_App(root)
root.mainloop()
```

Fig- 8.11(C)

Layout:

Fig- 8.12(L)

Fig- 8.13(L)

Fig- 8.14(L)

Fig- 8.15(L)

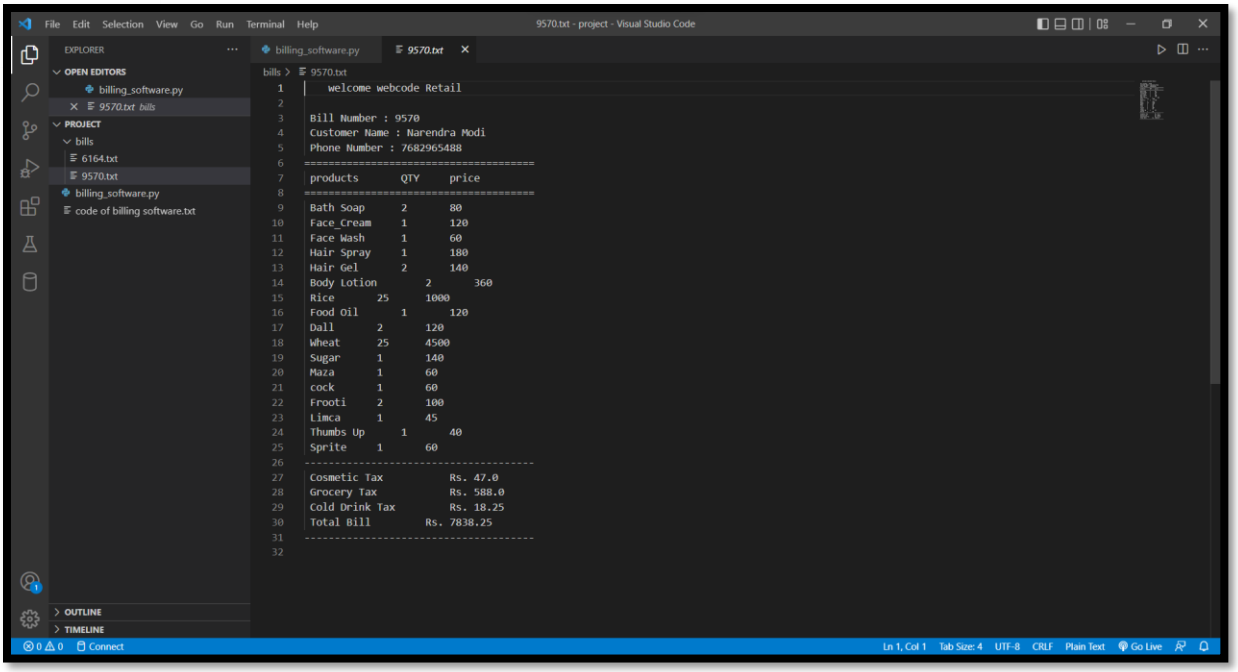


Fig- 8.16(L)

9. GANTT CHART

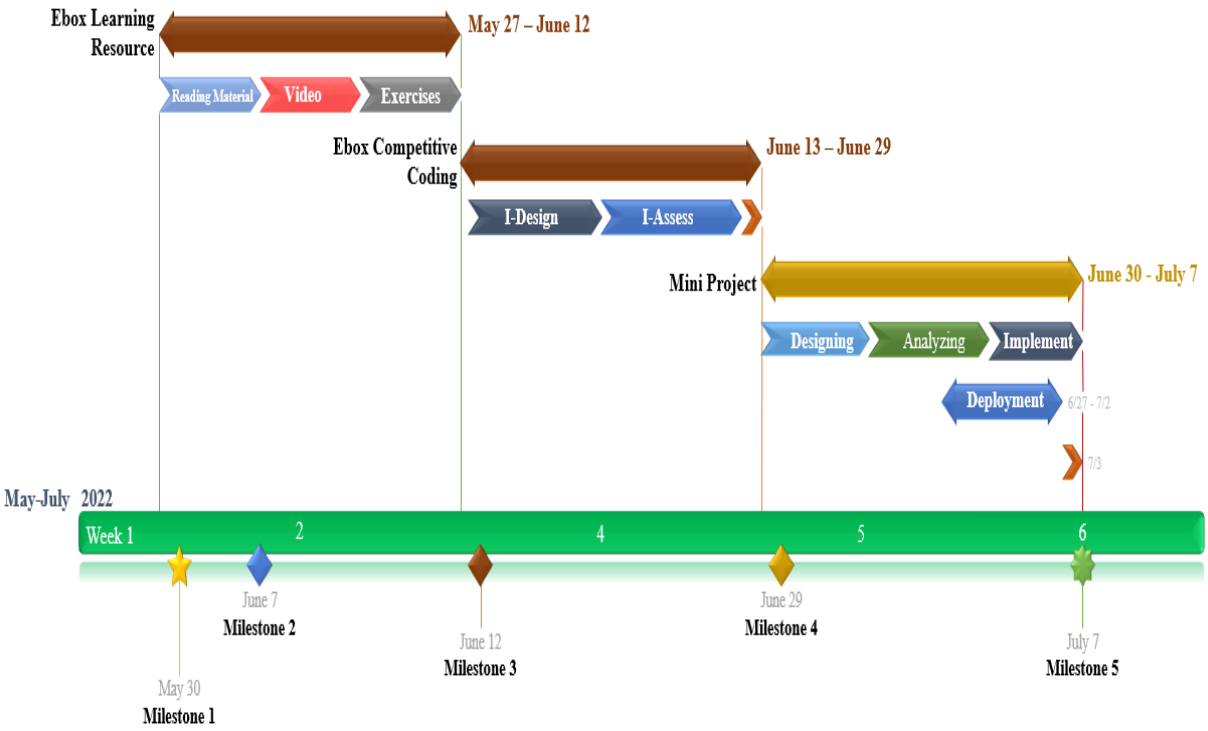


Fig- 9(G)

10. PROJECT LEGACY

Technical Lesson Learnt

Object-oriented programming (OOP) is a programming paradigm that organizes software design around data, rather than functions and logic. An object is a data field with its own set of properties and behavior.

Python is a high-level, object-oriented programming language that has lately gained popularity among students and professionals because of its adaptability, dynamic nature, resilience, and ease of learning. Not only that, but it is now the second most popular and preferred programming language after JavaScript, and it can be utilized in practically any technical industry, including machine learning, data science, web development, analytics, automation, testing, artificial intelligence, and many more.

Learning Python is easy as compared to other high-level, object-oriented programming languages such as Java or C++, but it has a few advanced concepts that come in handy when developing code that is robust, crisp, highly optimized, efficient, and normalized. Using these concepts in our code, I will be able to reduce bugs in my code as well as increase its efficiency thereby making me a seasoned Python programmer. So let us look at these concepts one by one and understand them in detail!

Topic 1- Class and objects:

Python is an object-oriented programming language. Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

- **__init__() Method:** To understand the meaning of classes we have to understand the built-in __init__() function. All classes have a function called __init__(), which is always executed when the class is being initiated. Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:
- **Object Method:** Objects can also contain methods. Methods in objects are functions that belong to the object.
- **self Parameter:** The self parameter is a reference to the current instance of the class and is used to access variables that belongs to the class. It does not have to

be named self, we can call it whatever we like, but it must be the first parameter of any function in the class.

- **Encapsulation:** Encapsulation in Python describes the concept of bundling data and methods within a single unit. So, for example, when you create a class, it means you are implementing encapsulation. A class is an example of encapsulation as it binds all the data members (instance variables) and methods into a single unit.
- **Access Specifier/Modifier:** Python access modifiers are used to modify the default scope of a variable. There are three types of access modifiers in python, and they are – Public, Private, and Protected. In python, we use underscores "_" symbol to specify the access modifier for specific data members and member functions in the class.
- **Getter and Setter:** Getters are the methods that are used in Object-Oriented Programming (OOPS) to access a class's private attributes. The setattr() function in Python corresponds to the getattr() function in Python. It alters an object's attribute values.The setter is a method that is used to set the property's value. It is very useful in object-oriented programming to set the value of private attributes in a class.

Topic 2- Relationship with classes:

Code reuse is one of the benefits of object-oriented programming languages. The link between the classes makes this reusability possible. In general, inheritance, association, composition, and aggregation are the four forms of relationships that object-oriented programming supports. All of these relationships are founded on the concepts of "is a," "has a," and "part of."

1. **Association:** It is a relationship of the "has-a" variety. Through their objects, associations provide a connection between two classes. One-to-one, one-to-many, many-to-one, and many-to-many associations are all possible.
2. **Composition:** It is a "part-of" relationship. Composition, in its simplest form, refers to the use of instance variables that contain references to other objects. As in "the heart is part of the body," both entities are depending upon one another in a composition relationship. It defines a strong type of relationship.

3. **Aggregation:** The "has-a" relationship is the foundation of aggregate. A unique kind of relationship is aggregation. In association, only one entity serves as the owner; in aggregation, multiple entities serve as the owner. In aggregation, the two entities come together for a task and then separate.

❖ **Relationships:**

One to Many: A one-to-many database relationship is one that exists between two database tables and in which a record in one table can refer to several records in another. In a blogging application.

One to One: A One-to-One relationship is a type of Relationship where both tables can have only one record on either side. A one-to-one relationship is like a relationship between a husband and a wife.

Topic 3- Inheritance:

In object-oriented programming, inheritance is a powerful feature. It refers to creating a new class with minimal or no changes to an existing class. The new class is known as the derived (or child) class, and the one it inherits from is known as the base (or parent) class.

In this course we have two types of Inheritance:

1. **Multiple inheritance:** This inheritance allows a child class to inherit from multiple parent classes. This type of inheritance is not supported by Java classes, but it is supported by Python. It has a significant advantage if we need to collect multiple characteristics from different classes.
2. **Multilevel inheritance:** The transfer of characteristics' properties to more than one class is done hierarchically in multilevel inheritance. To get a better picture, think of it as an ancestor-grandchild relationship or a root-to-leaf relationship in a multi-level tree.
3. **Inheritance Using Super:** In Python, the super() function makes class inheritance more manageable and extensible. The function returns a temporary object that can be referenced by the keyword super to a parent class.

   The super() function has two primary applications:

   - To avoid explicitly using the super (parent) class.
   - To allow for multiple inheritances.

Topic 4- Multi-threading:

Python may be used to create prototypes, and because it is so simple to use and read, it can be done rapidly. In the current context, data processing and its results are crucial to every application. Multithreading is useful when processing large amounts of data. Automation and scheduled jobs both benefit from threading.

Python's multithreading feature allows CPUs to run many components (threads) of a task concurrently to maximize CPU usage.

1. **What is Thread?**

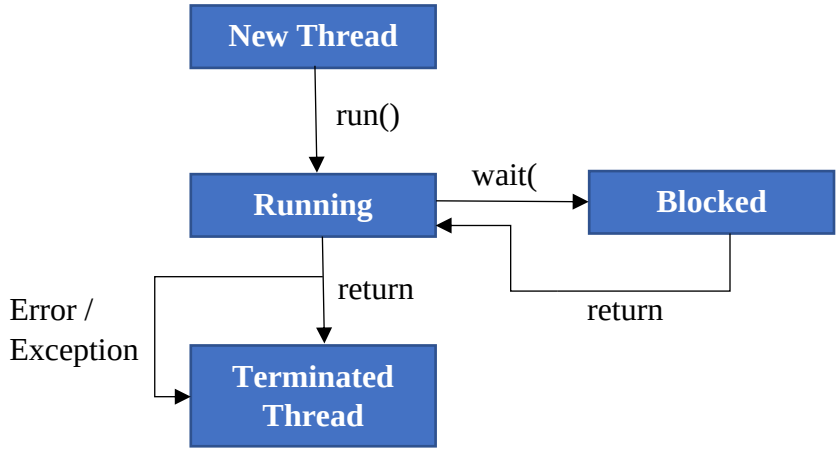
   Python threads are a type of process entity that can have their execution planned. A thread is, to put it simply, a computing process that a computer will carry out. It is a series of these instructions inside of a program that can run separately from other codes.

2. **The lifecycle of a Thread:**

   A Python thread's life cycle can be divided into three stages: creation, execution, and termination.The thread may be running, executing code, or it may be halted, awaiting a resource like another thread or an external one.

   A thread may also come to an end by issuing an error or exception after it has finished running its code.

   The following figure summarizes the states of the thread life cycle and how the thread may transition through these states:



LIFE CYCLE OF A THREAD

Fig- 10.4.1(T)

Topic 5- Exception:

An event that occurs during the execution of a programme and obstructs the regular flow of the program's instructions is an exception. A Python script typically raises an exception when it comes into a scenario that it cannot handle. A Python object that describes an error is called an exception. When a Python script encounters an exception, it must either deal with it right away or quit and stop working.

1. **What is Exception handling?**

   Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

   The try-expect statement: If the Python program contains suspicious code that may throw the exception, we must place that code in the try block. The try block must be followed with the except statement, which contains a block of code that will be executed if there is some exception in the try block.

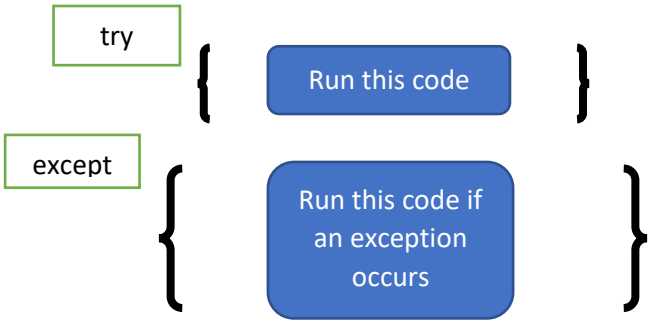


Fig- 10.5.1(T-E)

2. **try - catch - finally blocks:**

   - **try statement:** A try statement consists of the word try, a colon (:), and a group of code that may contain exceptions. It contains a clause or clauses.

     If no exceptions arise during the try statement's execution, the interpreter ignores the try statement's specified exception handlers.
   - **catch statement:** The type of exception that it is likely to catch is one that is taken one argument at a time by catch blocks. These justifications could be anything from a particular sort of exception that can be changed to a general category of exceptions.

- **finally blocks statement:** Finally block always executes irrespective of an exception being thrown or not. The final keyword allows you to create a block of code that follows a try-catch block.

Topic 6- DB Connection:

1. **What is database programming?**

   Every organisation, from a building company to a stock exchange, depends on massive databases. These are basically groups of tables that are linked to one another through columns. The Structured Query Language (SQL), which is used to create, read, and alter data, is supported by these database systems. In addition to creating and utilising the relationships between the stored data, SQL is utilised to access data. These databases also enable database normalisation rules to prevent data duplication. The database programming capabilities of the Python programming language are strong. Numerous databases, including MySQL, Oracle, Sybase, PostgreSQL, etc., are supported by Python. Python additionally offers Data Query Statements, Data Manipulation Language, and Data Definition Language.The Python DB API is a popular module that offers a database application programming interface for database development.

2. **Benefits of DB:**

   - Python programming is arguably faster and more effective than other languages.
   - Python is renowned for being portable.
   - It works on any platform.
   - SQL cursors are supported by Python.

Topic 7- Abstract Class:

1. **What is Abstract Class?**

   If a class includes one or more abstract methods, it is referred to as an abstract class. A method that is stated but lacks an implementation is said to be abstract. Subclasses must implement the abstract methods of abstract classes since they cannot be instantiated.

   The numbers module defines numeric tower which is a collection of base classes for numeric data types.

2. **Code level implementation of Abstract classes:**

   By subclassing directly from the base, we can avoid the need to register the class explicitly. In this case, the Python class management is used to recognize PluginImplementation as implementing the abstract PluginBase.

   e.g.: # implementation of abstract class through subclassing

```
import abc
class parent:
    def geeks(self):
        pass
class child(parent):
    def geeks(self):
        print("child class")
# Driver code
print( issubclass(child, parent))
print( isinstance(child(), parent))
```

   Output:

   True True

   A side-effect of using direct subclassing is, it is possible to find all the implementations of your plugin by asking the base class for the list of known classes derived from it.

Topic 8- Streams:

Streams are high-level async/await-ready primitives to work with network connections. Streams allow sending and receiving data without using callbacks or low-level protocols and transports.

1. **What are StreamWriters?**

   class asyncio.StreamWriter

   Represents a writer object that provides APIs to write data to the IO stream.

   It is not recommended to instantiate StreamWriter objects directly; use open_connection() and start_server() instead.

2. **What are StreamReaders?**

class asyncio.StreamReader

Represents a reader object that provides APIs to read data from the IO stream.

It is not recommended to instantiate StreamReader objects directly; use open_connection() and start_server() instead.

3. **What is StreamReaderWriter?**

   StreamReaderWriter instances allow wrapping streams which work in both read and write modes.

   The design is such that one can use the factory functions returned by the codec.lookup() function to construct the instance.

Topic 9- Lambda:

In Python, an anonymous function is a function that is defined without a name. While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword. Hence, anonymous functions are also called lambda functions.

- **How to use lambda Functions in Python?**

  A lambda function in python has the following syntax.

  Syntax: lambda arguments: expression

  Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

  e.g.: # Program to show the use of lambda functions

```
double = lambda x: x * 2
print(double(5))
```

<u>Managerial Lesson Learnt</u>

- Learned how to use assets in an optimized manner.
- Learned how to reduce the size of an existing application by modifying some of the code.
- Learned how to divide time between designing and coding.
- Learned how to handle frequent crashes in applications.
- How to reduce the use of extra loops to increase application speed.

11. BIBLIOGRAPHY

I was able to get all the data I required to create the programme, some help was taken from various Websites and YouTube for design inspiration for my application. I came up with my own ideas and tried several methods on my own to complete the Python coding using the VS editor. However, we did use certain concepts from www.javatpoint.com, www.github.com, and www.geeksforgeeks.org, for specific coding components.

Flow Charts: https://www.conceptdraw.com/

Geeks for geeks: https://www.geeksforgeeks.org/

Course Hero: https://www.coursehero.com/

GIT Link: https://github.com/s1dbugs/Billing-Management-System-

_____THE END_____