BILLING MANAGEMENT SYSTEM

A Major Project -II

Submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

(Computer Science & Engineering)

by

AnushkaSoni :- 0225CS191019
AkshitaVerma :- 0225CS191009

Under the guidance of

Prof. Abhishek Singh

Department of Computer Science & Engineering

BADERIA GLOBAL INSTITUTE OF ENGINEERING & MANAGEMENT, Jabalpur (M.P.)

under

Rajiv Gandhi ProdyogikiVishwavidyalaya, Bhopal (M.P.)

April 2023

BADERIA GLOBAL THE BEST Institute of Engineering & Management	**BADERIA GLOBAL INSTITUTE OF ENGINEERING & MANAGEMENT, Jabalpur (M.P.)**
	Department Of Information Technology

Certificate

This is to certify that the **Major Project - II** report entitled **Billing Management System** submitted by **Anushka Soni and AkshitaVerma** has been carried out under my guidance & supervision. The project report is approved for submission towards partial fulfillment of the requirement for the award of degree of **Bachelor of Engineering** in **Computer Science & Engineering** from "**Rajiv Gandhi Proudyogiki Vishwavidyalaya", Bhopal (M.P).**

Prof. Abhishek Singh
Project Incharge

Prof. Saurabh Sharma
HOD
Dept of CS

Global Nature Care Sangthan's Group of Institutions, Jabalpur (M.P.)

Certificate

This is to certify that the Major Project report entitled "**Billing Management System"** is submitted by **Anushka Soni and Akshita Verma** for the partial fulfillment of the requirement for the award of degree of **Bachelor of Engineering** in **Computer Science & Engineering** from **Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P).**

Internal Examiner
Date :

External Examiner
Date :

Declaration

We hereby declare that the project entitled **"Billing Management System"** which is being submitted in partial fulfillment of the requirement for award of the Degree of Bachelor of Engineering in Computer Science and Engineering to **"RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)"** is an authentic record of our own work done under the guidance of **Prof. Saurabh Sharma**, Department of Computer Science & Engineering, **GLOBAL ENGINEERING COLLEGE, JABALPUR**..

The matter reported in this Project has not been submitted earlier for the award of any other degree.

Dated :

submitted by

Anushka Soni 0225CS191019

Akshita Verma 0225CS191009

Acknowledgment

We sincerely express indebtedness to esteemed and revered guide "Prof. Abhishek Singh", Assistant Professor, for his invaluable guidance, supervision and encouragement throughout the work. Without his kind patronage and guidance the project would not have taken shape.

We take this opportunity to express deep sense of gratitude to "Prof. Saurabh Sharma", Head of "Department of Computer Science & Engineering"for his encouragement and kind approval. Also we thank him in providing the computer lab facility. We would like to express our sincere regards to him for advice and counseling from time to time.

We owe sincere thanks to all the lecturers in "Department of Computer Science & Engineering*"* for their advice and counseling time to time.

Dated :

Student Name:

Anushka Soni 0225CS191030

Akshita Verma 0225CS191009

Table of Contents

Chapter 1 : Introduction

- Background
- Objective
- Purpose
- Scope
- Applicability

Chapter 2: Survey of Technologies

- A Brief Introduction of Python
- A Brief Introduction of Python GUI - tkinter

Chapter 3: Requirement and Analysis

- Problem Definition
- Identification of Need
- Feasibility Study
- Project Planning& Scheduling
- Software Requirement Specification (SRS)
- Software Engineering Paradigm applied.

Chapter 4 : System Design

- Basic Modules
- Data Design
- Database Design
- Data Integrity and Constraints

4.1.Diagrams :

1. Data Flow Diagram
2. Use Case Diagram
3. Sequence diagram
4. Entity Relationship Diagram
5. Object Oriented design
6. Use Case Diagram

Chapter 5 : Results and Discussion

- Cost Estimation of the Project
- User Documentation
- Screenshots of source code
- Report generation

Chapter 6 : Conclusion

Appendex

Chapter-1

Introduction

Background:
In this report, we will discuss on how to run the billing system python. TheBilling System project in Python provides an exacting way for controlling the simplest to most complicated circumstances in which billing amounts and dates do not align with the dates when products and services are actually rendered.

Objectives:

A billing system, in its most basic form, is the mechanism through which a company bills and invoices its customers. Payment software, which automates the process of collecting payments, sending out periodic invoices, expense tracking, and invoice tracking, is frequently included in billing system.The objective of the project is to develop an application that will give service to users, gather user usage records, generate bills for each credit expiration, collect payments, and change customer balances. A project based on a billing that just requires Python and no other libraries for the user interface. The project that we have undertaken aims to develop a billing system that is clean, user-friendly and multi-functional. Development of this application includes a number of fields such that user feels comfortable and the system appears as dynamic to him.

The concept of this Billing Management System Project**,** is quite apparent; it is similar to real-life scenarios and is well-implemented. In this report, we will discuss on how to run the billing system python.The objectives of this project is to provide the systematic billing system to the organization which
includes the following points:
• To develop a better automated billing management system.
• To reduce workload of staff.
• To store data in centralized location.The project that we have undertaken aims to develop a billing system that is clean, user-friendly and multi-functional. Development of this application includes a number of fields such that user feels comfortable and the system appears as dynamic to him.

Purpose, scope and applicability:

- **Purpose:**

Python provides an exacting way for controlling the simplest to most complicated circumstances in which billing amounts and dates do not align with the dates when products and services are actually rendered.A billing system, in its most basic form, is the mechanism through which a company bills and invoices its customers. Payment software, which automates the process of collecting payments, sending out periodic invoices, expense tracking, and invoice tracking, is frequently included in billing systems. The main purpose of the project is to develop an application that will give service to users, gather user usage records, generate bills for each credit expiration, collect payments, and change customer balances. A project based on a billing that just requires Python and no other libraries for the

user interface. The concept of this Billing Management System Project**,** is quite apparent; it is similar to real-life scenarios and is well-implemented.

- **Scope:**

Billing systems aid in the automation of time-consuming tasks such as invoice generation, product tracking, and other accounting documents, among others. Furthermore, such software calculates and applies tax on taxable products automatically, so you don't have to worry about it every time you bill.

- **Applicability:**

This Billing System Project in Python is primarily concerned with dealing with customer bill, including purchase quantity and discount. The system allows the user to first enter the customer's name, after which the user can simply enter the item's name and quantity desired. The Simple Billing System Project estimates the total due bill with the predicted discount percentage after selecting all of the product components. Finally, the system generates an invoice receipt in.txt format that includes the customer's name, date, and time of purchase, product item quantity, unit price, and total, discountable amount, and payment amount. The quantity of each sold product is also deducted from the system, which is kept in the products.txt file.

Chapter 2

Survey of Technologies

- **A Brief Introduction of Python:**

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.Python Syntax compared to other programming languages
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.

- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.
- **A Brief Introduction of Python GUI- tkinter:**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

- Importing the module – tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

There are two main methods used which the user needs to remember while creating the Python application with GUI.

- **T**k(screenName=None, baseName=None, className='Tk', useTk=1): To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

  ```
  m=tkinter.Tk() where m is the name of the main window object
  ```

- **mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

  The application 'Billing Management System' being a small,portable and having good GUI, A computer system should have the combination of following hardware and software:

  1. Hardware Requirement:

  - intel I3 Processor
  - Minimum 2 GB RAM
  - Atleast 500GB HDD

  2. Software Requirement:

  - Windows 7/8/10 OS
  - Python 3.X IDLE
  - Tkinter module

Chapter 3

Requirements and Analysis

- **Problem Definition:**

As there is no use of digital technology to manage all the operations of the billing system. Operations of the store is managed manually through paper-based system. Due to the old manual system, the store is facing various problems of time ineffectiveness, lots of paper work, slow data processing and difficulties in finding the specific record due to file management system.
At existing system, the record has been kept manually. Any customers who need to pay bill has to wait for the paper bill as it consumes more time. This often requires a lot of time and effort. In the existing system the person work done manually. The manual work processes were time consuming and hence slow Following are the drawbacks of the existing system:
• The existing system is totally manual thus there are chance of error in processing.
• The basic and major drawbacks in the existing system are the speed of retrieval of data.
• The manual work such as calculation are more error prone.
• There is no central database from where one can get different statistical data at one place.

Problems that a modern billing software can solve:

Modern billing software can solve many problem to manage retail efficiently. Retailers and quick service restaurant (QSR) owners need to take lots of features into consideration in order to make sure that the business runs smoothly. It is not only the everyday sales, but also the customer retention, sales and stock tracking, accounts and receivables, which needs proper management. And being in charge of all of these things is definitely not easy, more so, when you are required to take care of all of these things single handedly. An automated and intuitive billing solution makes things easier in such a situation.

Problems that businesses usually face:

Most of the businesses, retail or QSR, till date use the cash register to tender cash and complete the financial transactions within the store. Here the problem is that a cash register can aid only with the in-store requirements. The back office functionalities like stock and purchase management, payments and receivables, accounting, everyday store difficulties and more cannot be managed with a cash register. It is a smart and modern billing solution that helps in solving all of these problems at one go.

So, what are the everyday issues that a store faces?

- **Maintaining more than one store**– Those who have a chain of stores or even more than one store, product/item lookup, inventory transfer, checking the overall sales report and more becomes difficult.
- **Level of efficiency**– In case of both the retail stores and QSRs, managing customers and billing during the rush hours become tough. Hand billing or billing through a cash register takes time and serving more number of customers in less times becomes an impossibility.

- **Maintaining payments and receivables**– A cash register does not provide you the option to keep a check on your payments and receivables. So, without a billing solution, you need to keep track of these things manually, which again takes up a lot of your time.
- **Keeping track of stock**– Keeping track of stock is one of the most important things for a business. But a cash register cannot provide any kind of help in this regard.
- **Maintaining customer satisfaction**– Customer satisfaction is of foremost importance for any business. However, if billing takes time, if there's a problem with the delivery, etc., it becomes hard for a business to retain the customers.

How a billing app solves these issues?

A billing app is a complete solution that helps you bill, build customer loyalty, manage customers, keep track of stock, payables and receivables, accounting, manage multiple stores, check reports and even more.

- **Generate bill in a matter of few seconds**– Bill generation becomes easier and fuss free, since you are no more required to do hand billing. Furthermore, with a billing system, you can print a bill in a matter of few seconds, thereby cutting down the waiting period for the customers.
- **Improve the level of customer satisfaction**– With a billing system, you can work on customer satisfaction. It is not only the smaller queue, but also the right and on-time delivery, discounts and customer loyalty, which work as customer delights.
- **Cut down on shrinkage**– Since you can keep an eye on the stock, it becomes easier for you to cut down on the shrinkage, thereby improving flow of materials and money. This contributes towards the improvement of the business functioning.
- **Generate reports**– When you start using a billing solution, you are no more required to keep track of any aspect of the business manually. The system does everything for you. You will simply have to check the reports from the system.

A billing solution automates the business in such a way that you can gain control over every aspect of your business. This again helps with the improvement of the ROI over time.

- **Identification of Need:**

This is the first phase in the system development cycle where a reasonable topic that is capable to solve the issue of the organization was selected. Work on the topic was started after defining its objectives and scope. A written document as a project proposal was submitted to the concerned faculty, stating the clear view about the organization, its issue, recommendation and requirement of the organization. To gather all these data required the organization was visited. After the approval of the project proposal, the next phase was started.

- **Feasibility Study:**

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it.
Feasibility study lets the developers for see the future of the project and usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization ability to meet the user needs and effective uses of resources. The document provides the feasibility of the project that is being designed

and list various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities.

- **Planning and Scheduling:**

In this phase the process involved in the overall development of the system and the activities that must be performed as well as the strategies were defined. Various planning activities has been be conducted, which includes the planning of work,
schedule, budget, gathering resources, and etc. Those proper planning activities helps me to complete the project on time and within the budget. the requirement documented were prioritized and the system view was developed. Different types of feasibility were analyzed for the completion of the project with in the estimated time, budget and the resources required. The overall module of the system was developed in this phase. The actual implementation was performed and the testing of the system was also executed. This phase was the longest phase.This is the last phase, in which the project is completed and formally closed. In this phase the overall process and the achievement is documented and presented to the mentor. Project Closure involves handing over the actual implementation view of the project along with the documentation including all the activities involve in project from scratch level to the completion of the project to the concerned External and
Internal supervisor.

- **Software Requirements Specification:**

    - Product Overview:

In this report, we will discuss on how to run the billing system python. TheBilling System project in Python provides an exacting way for controlling the simplest to most complicated circumstances in which billing amounts and dates do not align with the dates when products and services are actually rendered.

    - Purpose:

Python provides an exacting way for controlling the simplest to most complicated circumstances in which billing amounts and dates do not align with the dates when products and services are actually rendered.A billing system, in its most basic form, is the mechanism through which a company bills and invoices its customers. Payment software, which automates the process of collecting payments, sending out periodic invoices, expense tracking, and invoice tracking, is frequently included in billing systems. The main purpose of the project is to develop an application that will give service to users, gather user usage records, generate bills for each credit expiration, collect payments, and change customer balances. A project based on a billing that just requires Python and no other libraries for the user interface.
The concept of this Billing Management System Project**,** is quite apparent; it is similar to real-life scenarios and is well-implemented.

    - Scope:

Billing systems aid in the automation of time-consuming tasks such as invoice generation, product tracking, and other accounting documents, among others. Furthermore, such software calculates and applies tax on taxable products automatically, so you don’t have to worry about it every time you bill.

- Definition And Abbreviation

As there is no use of digital technology to manage all the operations of the billing system. Operations of the store is managed manually through paper-based system. Due to the old manual system, the store is facing various problems of time ineffectiveness, lots of paper work, slow data processing and difficulties in finding the specific record due to file management system.
At existing system, the record has been kept manually. Any customers who need to pay bill has to wait for the paper bill as it consumes more time. This often requires a lot of time and effort. In the existing system the person work done manually. The manual work processes were time consuming and hence slow Following are the drawbacks of the existing system:
• The existing system is totally manual thus there are chance of error in processing.
• The basic and major drawbacks in the existing system are the speed of retrieval of data.
• The manual work such as calculation are more error prone.
• There is no central database from where one can get different statistical data at one place.

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:
• Is there sufficient support for the management from the users?
• Will the system be used and work properly if it is being developed and implemented?
• Will there be any resistance from the user that will undetermined the possible application benefits?
This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

- **Software Engineering Paradigm applied** :

In this project, we have applied waterfall model.

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

1. **Requirements analysis and specification phase:**

The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as

to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called Software Requirement Specification (SRS) document is created which contained a detailed description of what the system will do in the common language.

2. **Design Phase:**

This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

3. **Implementation and unit testing:**

During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

4. **Integration and System Testing:**

This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

5. **Operation and maintenance phase:**

Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Advantages of Waterfall model:-

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.

- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

Chapter 4

System Design

Systems design is the process of defining the architecture, product design, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.
This partof the project consists of all the work performed from the very initial sketching of the outcome of the project to successful designing of the project.

- **Basic Modules:**

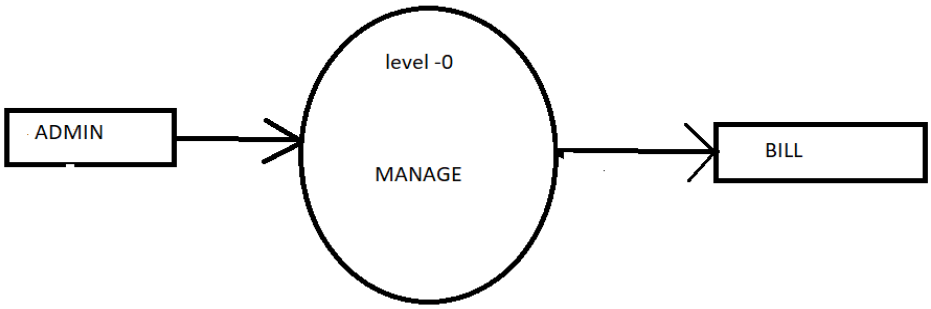
You should follow the divide and conquer theory, so divide the overall problem into more manageable parts and develop each part or module separately. When all modules are ready. You should integrate all the modules into one system. In this phase, you should briefly describe all the modules and the functionality of these modules.

Data Design: data design will consist of how you organize, managing and manipulate the data.

- **Database Design:** define the structure and explanation of schemas used in your project.
- **Data Integrity and Constraints:** define and explain all the validity checks and constraints you are providing to maintain data integrity.

Diagrams:

1.DFD (DATA FLOW DIAGRAM) :-



2.USE CASE DIAGRAM:-

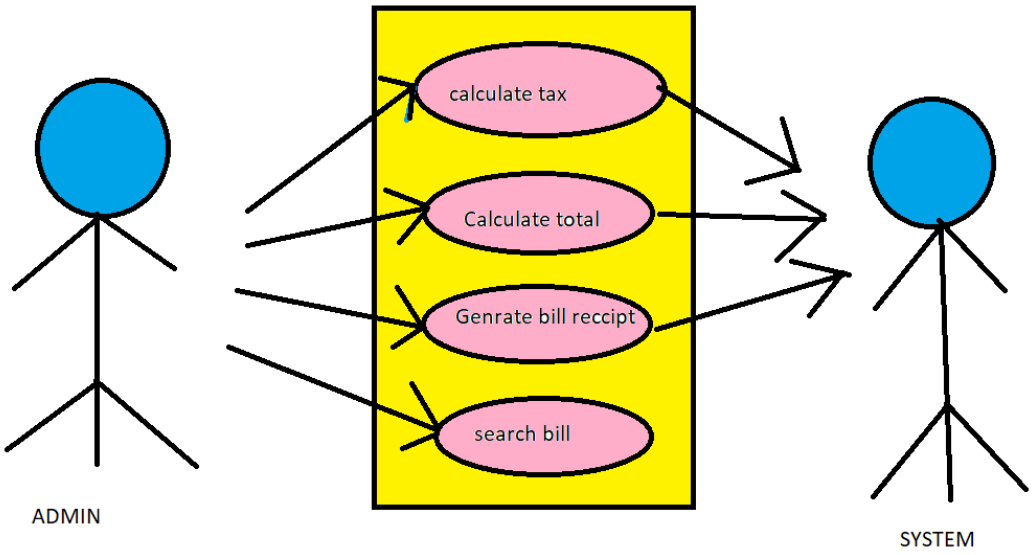
Use case diagram consists of use cases and actors and shows the interaction between them. The key points are:

1.The main purpose is to show the interaction between the use cases and the actor.

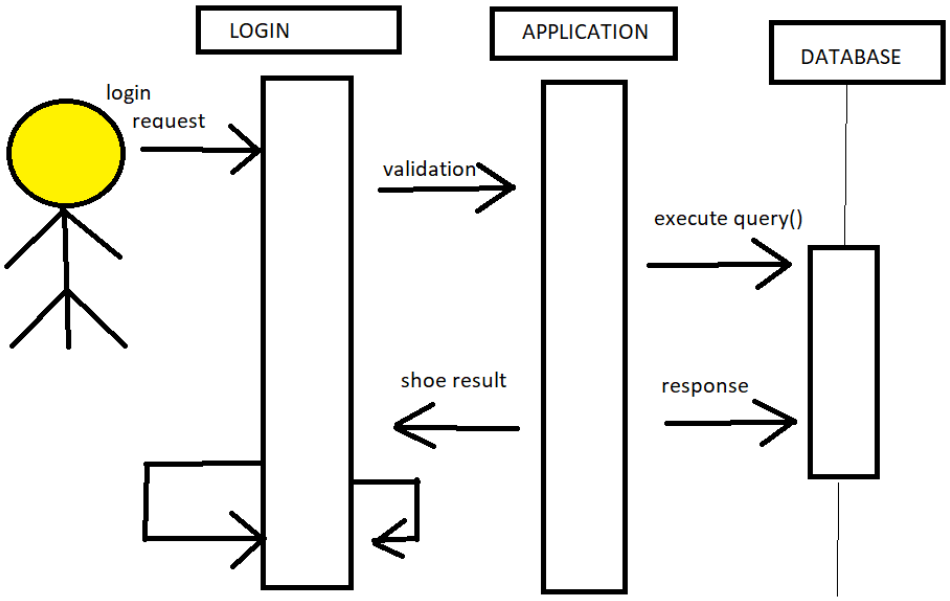
2.To represent the system requirement from user's perspective.

3.The use cases are the functions that are to be performed in the module.

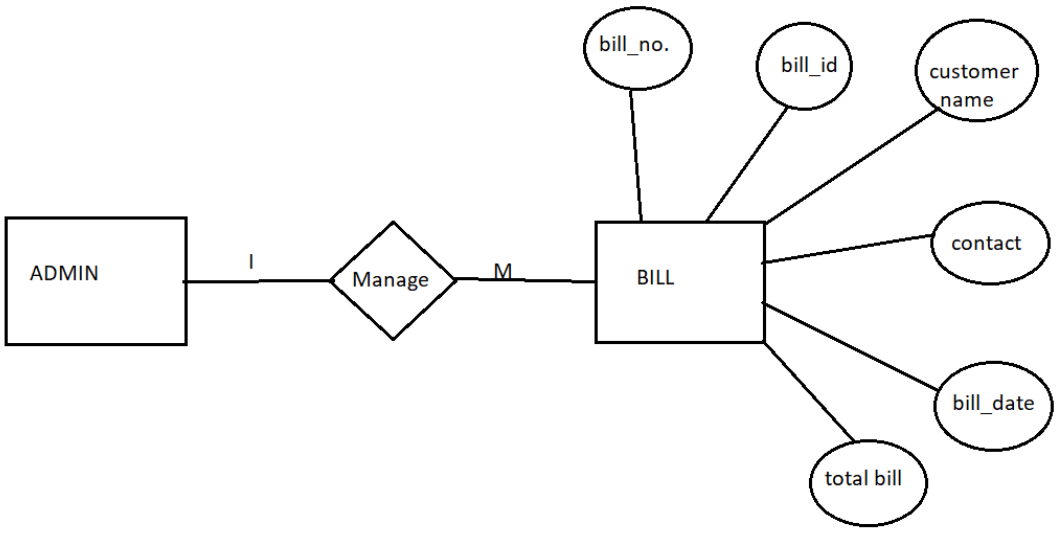
4.An actor could be the end-user of the system or anexternal system.



3. SEQUENCE DIAGRAM:-



4.ER DIAGRAM:-



- **Object Oriented Design:**

We have used classes and objects in this project. 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. This is the snapshot of the class used in our project.

Python OOPs Concepts :-

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

Main Concepts of Object-Oriented Programming (OOPs) :-

1.Class

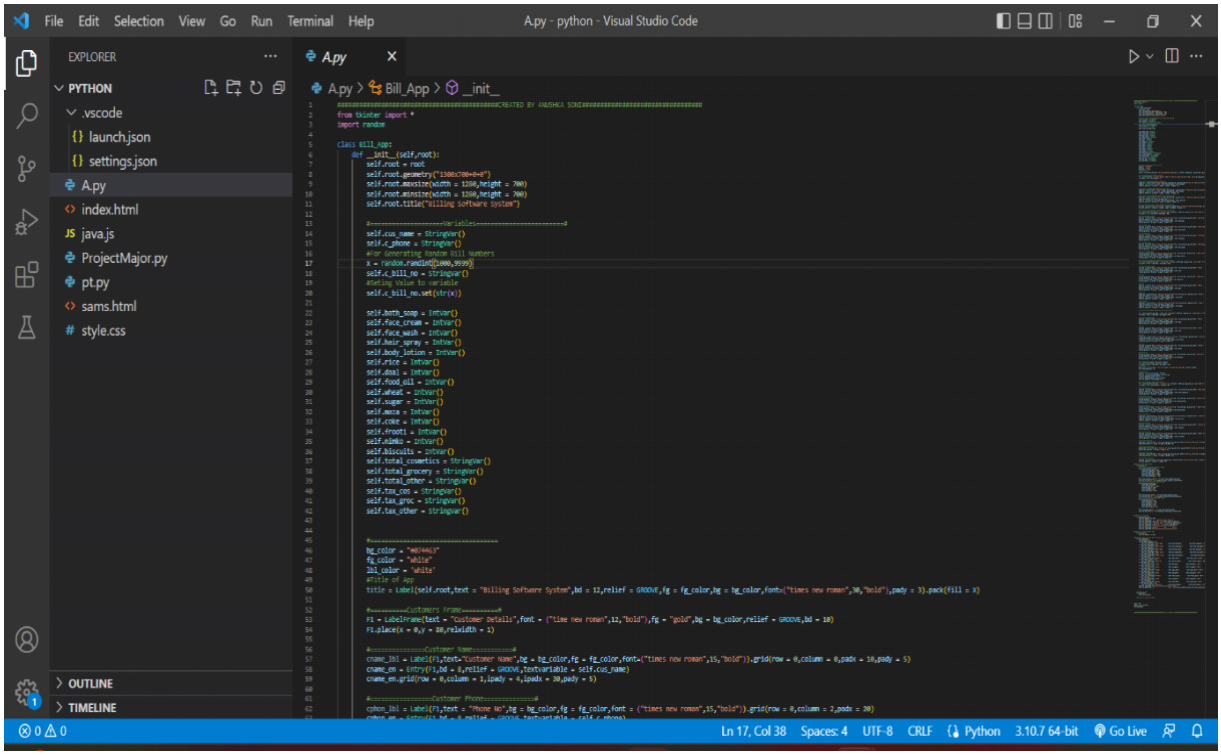
2.Objects

3.Polymorphism

4.Encapsulation

5.Inheritance

6.Data Abstraction



Chapter 5

Results and Discussion

- **Cost Estimation of the Project:**

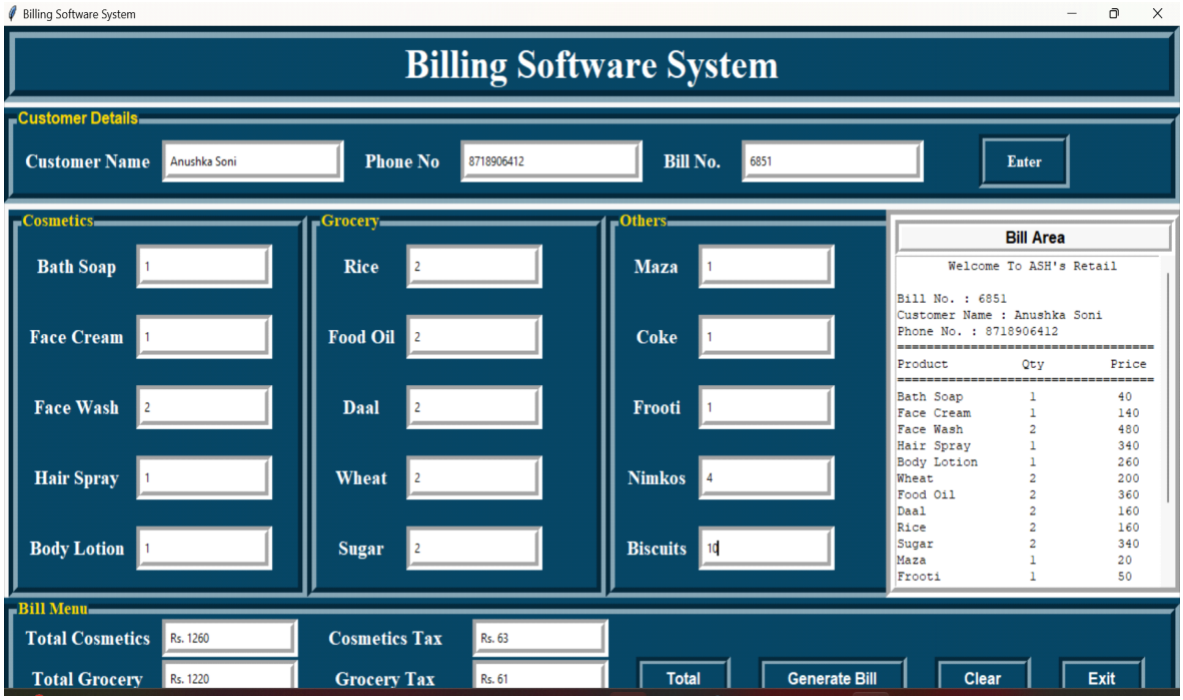
A system can be developed technically and that will be used if needed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.
The system is economically feasible. Since the interface for this system is developed using the existing resources available at the organization, there is nominal expenditure and economic feasibility for certain.

- **User Documentation:**

Billing System Project in Python: Project Details and Technology:

Project Name:	Billing System Project in Python
Abstract	A mini project based on Billing System which only uses Python Language with no any other Libraries for GUI.
Language/s Used:	Python
Python version (Recommended):	3.8/3.9 and so on…
Database:	None
Type:	Desktop Application
Developer:	AnushkaSoni

- **Screenshot of the working project:**



Chapter 6

Conclusions

In this report, we have discussed on how to run the billing system python. The Billing System project in Python provides an exacting way for controlling the simplest to most complicated circumstances in which billing amounts and dates do not align with the dates when products and services are actually rendered. This Billing Management System Project inPython is created in python programming language using PyCharm Community IDE, This Project was developed using Console Based.

- What is Billing System Project in Python?

A billing system, in its most basic form, is the mechanism through which a company bills and invoices its customers. Payment software, which automates the process of collecting payments, sending out periodic invoices, expense tracking, and invoice tracking, is frequently included in billing systems.

The goal of the project is to develop an application that will give service to users, gather user usage records, generate bills for each credit expiration, collect payments, and change customer balances.

A Mini project based on a billing that just requires Python and no other libraries for the user interface.

The concept of this Billing Management System Project, is quite apparent; it is similar to real-life scenarios and is well-implemented.

- What are the different billing systems?

There are three basic types of system : Closed**,** open, and isolated. Medical billing is a big system that is part of the larger healthcare system. Medical billing, best practices for patient care, health institutions, and private clinics are all part of the healthcare network.

- What is the function of billing system?

Billing software, in its most basic form, allows you to keep track of the products and services your customers use, generate and send invoices, and accept payments.

- Why is Billing System in Python Important?

Billing systems aid in the automation of time-consuming tasks such as invoice generation, product tracking, and other accounting documents, among others. Furthermore, such software calculates and applies tax on taxable products automatically, so you don’t have to worry about it every time you bill.

This Billing System Project in Python is primarily concerned with dealing with customer bill, including purchase quantity and discount. The system allows the user to first enter the customer’s name, after which the user can simply enter the item’s name and quantity desired.TheSimple Billing System Project estimates the total due bill with the predicted discount percentage after selecting all of the product components. Finally, the system generates an invoice receipt in.txt format that includes the customer’s name, date, and time of purchase, product item quantity, unit price, and total, discountable amount, and payment amount. The quantity of each sold product is also deducted from the system, which is kept in the products.txt file.

Simple Billing System Project in Python Available Features :

- Invoice Receipt (.txt format)
- Items deductions
- Discount system

To start executing Python Project With Source Code, make sure that you have installed Python **HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/"** HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/"3.9 and **HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/"** HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/" HYPERLINK "https://www.python.org/"PyCharm in your computer. The Bill Management System In Python is a GUI project using python Tkinter which displays the menu of the departmental store, takes orders from the customer, and generates a bill. Thus, this is a digital way of management of the items order and also the payment system. The project file contains a python script (app.py). This Bill Management System is in Python. Talking about the features of this system, it contains only the admin section. The admin manages all the orders management, payments, and report. Here, GUI is created using Python Tkinter. Here, the user can enter customer details and select items from the menu. The software displays the menu items as category wise. Users can generate bills after selecting at least 1 item. The system displays a bill inside the 'Bill Area' section. Also, the design of this system is pretty simple so that the user won't get any difficulties while working on it.

APPENDIX

```
##############################################CREATED BY ANUSHKA
SONI###################################
from tkinter import *
import random


class Bill_App:
    def __init__(self,root):
        self.root = root
        self.root.geometry("1300x700+0+0")
        self.root.maxsize(width = 1280,height = 700)
        self.root.minsize(width = 1280,height = 700)
        self.root.title("Billing Software System")


        #==================Variables======================#
        self.cus_name = StringVar()
        self.c_phone = StringVar()
        #For Generating Random Bill Numbers
        x = random.randint(1000,9999)
        self.c_bill_no = StringVar()
        #Seting Value to variable
        self.c_bill_no.set(str(x))


        self.bath_soap = IntVar()
        self.face_cream = IntVar()
        self.face_wash = IntVar()
        self.hair_spray = IntVar()
```

```
        self.body_lotion = IntVar()
        self.rice = IntVar()
        self.daal = IntVar()
        self.food_oil = IntVar()
        self.wheat = IntVar()
        self.sugar = IntVar()
        self.maza = IntVar()
        self.coke = IntVar()
        self.frooti = IntVar()
        self.nimko = IntVar()
        self.biscuits = IntVar()
        self.total_cosmetics = StringVar()
        self.total_grocery = StringVar()
        self.total_other = StringVar()
        self.tax_cos = StringVar()
        self.tax_groc = StringVar()
        self.tax_other = StringVar()



        #==================================
        bg_color = "#074463"
        fg_color = "white"
        lbl_color = 'white'
        #Title of App
        title = Label(self.root,text = "Billing Software System",bd = 12,relief = GROOVE,fg =
fg_color,bg = bg_color,font=("times new roman",30,"bold"),pady = 3).pack(fill = X)


        #==========Customers Frame==========#
```

```
        F1 = LabelFrame(text = "Customer Details",font = ("time new roman",12,"bold"),fg =
"gold",bg = bg_color,relief = GROOVE,bd = 10)
        F1.place(x = 0,y = 80,relwidth = 1)

        #===============Customer Name===========#
        cname_lbl = Label(F1,text="Customer Name",bg = bg_color,fg = fg_color,font=("times
new roman",15,"bold")).grid(row = 0,column = 0,padx = 10,pady = 5)
        cname_en = Entry(F1,bd = 8,relief = GROOVE,textvariable = self.cus_name)
        cname_en.grid(row = 0,column = 1,ipady = 4,ipadx = 30,pady = 5)

        #================Customer Phone=============#
        cphon_lbl = Label(F1,text = "Phone No",bg = bg_color,fg = fg_color,font = ("times new
roman",15,"bold")).grid(row = 0,column = 2,padx = 20)
        cphon_en = Entry(F1,bd = 8,relief = GROOVE,textvariable = self.c_phone)
        cphon_en.grid(row = 0,column = 3,ipady = 4,ipadx = 30,pady = 5)

        #==================Customer Bill No================#
        cbill_lbl = Label(F1,text = "Bill No.",bg = bg_color,fg = fg_color,font = ("times new
roman",15,"bold"))
        cbill_lbl.grid(row = 0,column = 4,padx = 20)
        cbill_en = Entry(F1,bd = 8,relief = GROOVE,textvariable = self.c_bill_no)
        cbill_en.grid(row = 0,column = 5,ipadx = 30,ipady = 4,pady = 5)

        #==================Bill Search Button==============#
        bill_btn = Button(F1,text = "Enter",bd = 7,relief = GROOVE,font = ("times new
roman",12,"bold"),bg = bg_color,fg = fg_color)
        bill_btn.grid(row = 0,column = 6,ipady = 5,padx = 60,ipadx = 19,pady = 5)

        #================Cosmetics Frame==================#
```

```
        F2 = LabelFrame(self.root,text = 'Cosmetics',bd = 10,relief = GROOVE,bg =
bg_color,fg = "gold",font = ("times new roman",13,"bold"))
        F2.place(x = 5,y = 180,width = 325,height = 380)

        #===========Frame Content
        bath_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Bath Soap")
        bath_lbl.grid(row = 0,column = 0,padx = 10,pady = 20)
        bath_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.bath_soap)
        bath_en.grid(row = 0,column = 1,ipady = 5,ipadx = 5)

        #=======Face Cream
        face_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Face Cream")
        face_lbl.grid(row = 1,column = 0,padx = 10,pady = 20)
        face_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.face_cream)
        face_en.grid(row = 1,column = 1,ipady = 5,ipadx = 5)

        #========Face Wash
        wash_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Face Wash")
        wash_lbl.grid(row = 2,column = 0,padx = 10,pady = 20)
        wash_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.face_wash)
        wash_en.grid(row = 2,column = 1,ipady = 5,ipadx = 5)

        #========Hair Spray
        hair_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Hair Spray")
        hair_lbl.grid(row = 3,column = 0,padx = 10,pady = 20)
```

```
hair_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.hair_spray)
        hair_en.grid(row = 3,column = 1,ipady = 5,ipadx = 5)

        #============Body Lotion
        lot_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Body Lotion")
        lot_lbl.grid(row = 4,column = 0,padx = 10,pady = 20)
        lot_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.body_lotion)
        lot_en.grid(row = 4,column = 1,ipady = 5,ipadx = 5)

        #==================Grocery Frame====================#
        F2 = LabelFrame(self.root,text = 'Grocery',bd = 10,relief = GROOVE,bg = bg_color,fg
= "gold",font = ("times new roman",13,"bold"))
        F2.place(x = 330,y = 180,width = 325,height = 380)

        #===========Frame Content
        rice_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Rice")
        rice_lbl.grid(row = 0,column = 0,padx = 10,pady = 20)
        rice_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.rice)
        rice_en.grid(row = 0,column = 1,ipady = 5,ipadx = 5)

        #=======
        oil_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Food Oil")
        oil_lbl.grid(row = 1,column = 0,padx = 10,pady = 20)
        oil_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.food_oil)
        oil_en.grid(row = 1,column = 1,ipady = 5,ipadx = 5)
```

```
        #=======
        daal_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Daal")
        daal_lbl.grid(row = 2,column = 0,padx = 10,pady = 20)
        daal_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.daal)
        daal_en.grid(row = 2,column = 1,ipady = 5,ipadx = 5)

        #========
        wheat_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Wheat")
        wheat_lbl.grid(row = 3,column = 0,padx = 10,pady = 20)
        wheat_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.wheat)
        wheat_en.grid(row = 3,column = 1,ipady = 5,ipadx = 5)

        #============
        sugar_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Sugar")
        sugar_lbl.grid(row = 4,column = 0,padx = 10,pady = 20)
        sugar_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.sugar)
        sugar_en.grid(row = 4,column = 1,ipady = 5,ipadx = 5)

        #==================Other Stuff====================#

        F2 = LabelFrame(self.root,text = 'Others',bd = 10,relief = GROOVE,bg = bg_color,fg =
"gold",font = ("times new roman",13,"bold"))
        F2.place(x = 655,y = 180,width = 325,height = 380)

        #===========Frame Content
```

```
        maza_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Maza")
        maza_lbl.grid(row = 0,column = 0,padx = 10,pady = 20)
        maza_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.maza)
        maza_en.grid(row = 0,column = 1,ipady = 5,ipadx = 5)

        #=======
        cock_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Coke")
        cock_lbl.grid(row = 1,column = 0,padx = 10,pady = 20)
        cock_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.coke)
        cock_en.grid(row = 1,column = 1,ipady = 5,ipadx = 5)

        #=======
        frooti_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Frooti")
        frooti_lbl.grid(row = 2,column = 0,padx = 10,pady = 20)
        frooti_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.frooti)
        frooti_en.grid(row = 2,column = 1,ipady = 5,ipadx = 5)

        #========
        cold_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Nimkos")
        cold_lbl.grid(row = 3,column = 0,padx = 10,pady = 20)
        cold_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.nimko)
        cold_en.grid(row = 3,column = 1,ipady = 5,ipadx = 5)

        #============
```

```
        bis_lbl = Label(F2,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Biscuits")
        bis_lbl.grid(row = 4,column = 0,padx = 10,pady = 20)
        bis_en = Entry(F2,bd = 8,relief = GROOVE,textvariable = self.biscuits)
        bis_en.grid(row = 4,column = 1,ipady = 5,ipadx = 5)

        #==================Bill Aera================#
        F3 = Label(self.root,bd = 10,relief = GROOVE)
        F3.place(x = 960,y = 180,width = 325,height = 380)
        #===========
        bill_title = Label(F3,text = "Bill Area",font = ("Lucida",13,"bold"),bd= 7,relief =
GROOVE)
        bill_title.pack(fill = X)

        #============
        scroll_y = Scrollbar(F3,orient = VERTICAL)
        self.txt = Text(F3,yscrollcommand = scroll_y.set)
        scroll_y.pack(side = RIGHT,fill = Y)
        scroll_y.config(command = self.txt.yview)
        self.txt.pack(fill = BOTH,expand = 1)

        #===========Buttons Frame============#
        F4 = LabelFrame(self.root,text = 'Bill Menu',bd = 10,relief = GROOVE,bg =
bg_color,fg = "gold",font = ("times new roman",13,"bold"))
        F4.place(x = 0,y = 560,relwidth = 1,height = 145)

        #==================
        cosm_lbl = Label(F4,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Total Cosmetics")
```

```
        cosm_lbl.grid(row = 0,column = 0,padx = 10,pady = 0)
        cosm_en = Entry(F4,bd = 8,relief = GROOVE,textvariable = self.total_cosmetics)
        cosm_en.grid(row = 0,column = 1,ipady = 2,ipadx = 5)

        #==================
        gro_lbl = Label(F4,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Total Grocery")
        gro_lbl.grid(row = 1,column = 0,padx = 10,pady = 5)
        gro_en = Entry(F4,bd = 8,relief = GROOVE,textvariable = self.total_grocery)
        gro_en.grid(row = 1,column = 1,ipady = 2,ipadx = 5)

        #================
        oth_lbl = Label(F4,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Others Total")
        oth_lbl.grid(row = 2,column = 0,padx = 10,pady = 5)
        oth_en = Entry(F4,bd = 8,relief = GROOVE,textvariable = self.total_other)
        oth_en.grid(row = 2,column = 1,ipady = 2,ipadx = 5)

        #================
        cosmt_lbl = Label(F4,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Cosmetics Tax")
        cosmt_lbl.grid(row = 0,column = 2,padx = 30,pady = 0)
        cosmt_en = Entry(F4,bd = 8,relief = GROOVE,textvariable = self.tax_cos)
        cosmt_en.grid(row = 0,column = 3,ipady = 2,ipadx = 5)

        #=================
        grot_lbl = Label(F4,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Grocery Tax")
        grot_lbl.grid(row = 1,column = 2,padx = 30,pady = 5)
```

```
        grot_en = Entry(F4,bd = 8,relief = GROOVE,textvariable = self.tax_groc)
        grot_en.grid(row = 1,column = 3,ipady = 2,ipadx = 5)

        #=================
        otht_lbl = Label(F4,font = ("times new roman",15,"bold"),fg = lbl_color,bg =
bg_color,text = "Others Tax")
        otht_lbl.grid(row = 2,column = 2,padx = 10,pady = 5)
        otht_en = Entry(F4,bd = 8,relief = GROOVE,textvariable = self.tax_other)
        otht_en.grid(row = 2,column = 3,ipady = 2,ipadx = 5)

        #===================
        total_btn = Button(F4,text = "Total",bg = bg_color,fg =
fg_color,font=("lucida",12,"bold"),bd = 7,relief = GROOVE,command = self.total)
        total_btn.grid(row = 1,column = 4,ipadx = 20,padx = 30)

        #========================
        genbill_btn = Button(F4,text = "Generate Bill",bg = bg_color,fg =
fg_color,font=("lucida",12,"bold"),bd = 7,relief = GROOVE,command = self.bill_area)
        genbill_btn.grid(row = 1,column = 5,ipadx = 20)

        #===================
        clear_btn = Button(F4,text = "Clear",bg = bg_color,fg =
fg_color,font=("lucida",12,"bold"),bd = 7,relief = GROOVE,command = self.clear)
        clear_btn.grid(row = 1,column = 6,ipadx = 20,padx = 30)

        #======================
        exit_btn = Button(F4,text = "Exit",bg = bg_color,fg =
fg_color,font=("lucida",12,"bold"),bd = 7,relief = GROOVE,command = self.exit)
        exit_btn.grid(row = 1,column = 7,ipadx = 20)
```

```
#Function to get total prices
    def total(self):
        #================Total Cosmetics Prices
        self.total_cosmetics_prices = (
            (self.bath_soap.get() * 40)+
            (self.face_cream.get() * 140)+
            (self.face_wash.get() * 240)+
            (self.hair_spray.get() * 340)+
            (self.body_lotion.get() * 260)
        )
        self.total_cosmetics.set("Rs. "+str(self.total_cosmetics_prices))
        self.tax_cos.set("Rs. "+str(round(self.total_cosmetics_prices*0.05)))
        #==================Total Grocery Prices
        self.total_grocery_prices = (
            (self.wheat.get()*100)+
            (self.food_oil.get() * 180)+
            (self.daal.get() * 80)+
            (self.rice.get() *80)+
            (self.sugar.get() * 170)


        )
        self.total_grocery.set("Rs. "+str(self.total_grocery_prices))
        self.tax_groc.set("Rs. "+str(round(self.total_grocery_prices*0.05)))
        #====================Total Other Prices
        self.total_other_prices = (
            (self.maza.get() * 20)+
            (self.frooti.get() * 50)+
            (self.coke.get() * 60)+
```

```
            (self.nimko.get() * 20)+
            (self.biscuits.get() * 20)
        )
        self.total_other.set("Rs. "+str(self.total_other_prices))
        self.tax_other.set("Rs. "+str(round(self.total_other_prices*0.05)))



#Function For Text Area
    def welcome_soft(self):
        self.txt.delete('1.0',END)
        self.txt.insert(END,"       Welcome To ASH's Retail\n")
        self.txt.insert(END,f"\nBill No. : {str(self.c_bill_no.get())}")
        self.txt.insert(END,f"\nCustomer Name : {str(self.cus_name.get())}")
        self.txt.insert(END,f"\nPhone No. : {str(self.c_phone.get())}")
        self.txt.insert(END,"\n==================================")
        self.txt.insert(END,"\nProduct          Qty         Price")
        self.txt.insert(END,"\n==================================")

#Function to clear the bill area
    def clear(self):
        self.txt.delete('1.0',END)

#Add Product name , qty and price to bill area
    def bill_area(self):
        self.welcome_soft()
        if self.bath_soap.get() != 0:
            self.txt.insert(END,f"\nBath
Soap         {self.bath_soap.get()}           {self.bath_soap.get() * 40}")
        if self.face_cream.get() != 0:
```

```
            self.txt.insert(END,f"\nFace
Cream        {self.face_cream.get()}          {self.face_cream.get() * 140}")
        if self.face_wash.get() != 0:
            self.txt.insert(END,f"\nFace
Wash         {self.face_wash.get()}          {self.face_wash.get() * 240}")
        if self.hair_spray.get() != 0:
            self.txt.insert(END,f"\nHair
Spray        {self.hair_spray.get()}          {self.hair_spray.get() * 340}")
        if self.body_lotion.get() != 0 :
            self.txt.insert(END,f"\nBody
Lotion       {self.body_lotion.get()}          {self.body_lotion.get() * 260}")
        if self.wheat.get() != 0:
            self.txt.insert(END,f"\nWheat           {self.wheat.get()}          {self.wheat.get() *
100}")
        if self.food_oil.get() != 0:
            self.txt.insert(END,f"\nFood Oil         {self.food_oil.get()}          {self.food_oil.get()
* 180}")
        if self.daal.get() != 0:
            self.txt.insert(END,f"\nDaal            {self.daal.get()}          {self.daal.get() * 80}")
        if self.rice.get() != 0:
            self.txt.insert(END,f"\nRice            {self.rice.get()}          {self.rice.get() * 80}")
        if self.sugar.get() != 0:
            self.txt.insert(END,f"\nSugar           {self.sugar.get()}          {self.sugar.get() *
170}")
        if self.maza.get() != 0:
            self.txt.insert(END,f"\nMaza            {self.maza.get()}          {self.maza.get() * 20}")
        if self.frooti.get() != 0:
            self.txt.insert(END,f"\nFrooti          {self.frooti.get()}          {self.frooti.get() * 50}")
        if self.coke.get() != 0:
```

```
            self.txt.insert(END,f"\nCoke             {self.coke.get()}          {self.coke.get() * 60}")
        if self.nimko.get() != 0:
            self.txt.insert(END,f"\nNimko            {self.nimko.get()}          {self.nimko.get() *
20}")
        if self.biscuits.get() != 0:
            self.txt.insert(END,f"\nBiscuits         {self.biscuits.get()}          {self.biscuits.get() *
20}")
        self.txt.insert(END,"\n================================")
        self.txt.insert(END,f"\n                    Total :
{self.total_cosmetics_prices+self.total_grocery_prices+self.total_other_prices+self.total_cos
metics_prices * 0.05+self.total_grocery_prices * 0.05+self.total_other_prices * 0.05}")


    #Function to exit
    def exit(self):
        self.root.destroy()

    #Function To Clear All Fields



root = Tk()
object = Bill_App(root)
root.mainloop()



##############################################CREATED BY ANUSHKA
SONI###################################
```