PROJECT REPORT

On

"Water Billing System Using GUI"

Submitted By

Preet Tiwari

Guided By: -

Mr. Ratnesh K. Choudhary

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

S. B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR.

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

2021-2022

S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SESSION 2021-2022

CERTIFICATE

This is to certify that the Project titled **"Water Billing System Using GUI"** is a Bonafide work of Preet Tiwari carried out for the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in **Computer Science & Engineering.**

Mr. Ratnesh K. Choudhary
Assistant Professor

Mr. Animesh Tayal
Head of Department

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Billing software is a tool that automates invoice generation for goods and services rendered. The integrated tools create a list of products and services and their related costs per tax calculations and send them to the respective recipient.

Water billing system is an online project which allows the water district management to post the bill of each customer and any other transactions related to water billing such as customer information, service record module and statement of accounts.

The online system can be accessed by the administrators and other staff of the organization and as well as the customers of the water district. The administrator can access all the modules of the project whereas the customers can only view their payment history and billing records.

By employing water billing software, utilities can reduce the size of their labor force, providing immediate savings. They won't need as many utility employees to tackle billing issues if software is doing this job for them. And reducing the number of employees in the billing department doesn't necessarily mean laying off talented workers. Utilities can train billing department staffers to take on other jobs, whether with the municipal utility or elsewhere in the utility.

Water billing software also allows utility employees to focus their efforts on other tasks

They can spend more time in the utility's customer service department answering resident questions. They can analyze the reports created by billing software to hunt for potentially costly leaks. And they can study these same reports to determine new ways to conserve water use and reduce the water utility's overall costs of providing this water to its customers.

Today's water billing programs provide a host of reports and functions that stretch far beyond simply sending payment notices to consumers. Savvy water utility officials recognize this and use these reports to operate their utilities in a more efficient and cost-effective manner.

CHAPTER 2

METHODOLOGY

We created an application for the water billing system with help of python. In this, we imported the main file or packages which are required for it. So that, when we run the code, it should not show any type of error in the compilation.

As, we created a python file to store the code in it. There are some packages and file which are need to be imported as like tkinter, OS, random, messagebox.

1.Tkinter:

Tkinter is the most commonly used library for developing GUI (Graphical User Interface) in Python. It is a standard Python interface to the Tk GUI toolkit shipped with Python. As Tk and Tkinter are available on most of the Unix platforms as well as on the Windows system, developing GUI applications with Tkinter becomes the fastest and easiest.

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

2.Random:

The Python Random module is a **built-in module for generating random numbers in Python**. These are pseudo-random numbers, which do not represent true randomness. This module can be used to do things like generate random numbers, output a random value for a list or string, and so on.

Python **Random module** is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random. This module can be used to perform random actions such as generating random numbers, print random a value for a list or string, etc.

random.random() method is used to generate random floats between 0.0 to 1.

random.choice() function is used to return a random item from a list, tuple, or string.

random.shuffle() method is used to shuffle a sequence (list). Shuffling means changing the position of the elements of the sequence. Here, the shuffling operation is inplace.

3.OS:

The OS module in python **provides functions for interacting with the operating system**. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.
There are different methods available in the OS module for creating a directory. These are –

- os.mkdir()
- os.makedirs()

os.listdir() method in Python is used to get the list of all files and directories in the specified directory. If we don't s in Python. These are –

- Using os.remove()
- Using os.rmdir()

pecify any directory, then the list of files and directories in the current working directory will be returned.

4.Messagebox:

The messagebox module is used to display the message boxes in the python applications. There are the various functions which are used to display the relevant messages depending upon the application requirements. The syntax to use the messagebox is given below. Syntax messagebox.function_name (title, message [, options]) Parameters

- **function_name:** It represents an appropriate message box function.
- **title:** It is a string which is shown as a title of a message box.
- **message:** It is the string to be displayed as a message on the message box.
- **options:** There are various options which can be used to configure the message dialog box.

1. showinfo()

The showinfo() messagebox is used where we need to show some relevant information to the user.

2. showwarning()

This method is used to display the warning to the user

3. showerror()

This method is used to display the error message to the user

4. askquestion()

This method is used to ask some question to the user which can be answered in yes or no.

5. askokcancel()

This method is used to confirm the user's action regarding some application activity.

6. askyesno()

This method is used to ask the user about some action to which, the user can answer is yes or no.

7. askretrycancel()

This method is used to ask the user about doing a particular task again or not.

CHAPTER 3
TOOLS/PLATFORMS

3.1 SOFTWARE REQUREMENT

a) SOFTWARE: Python

b) IDE/FRAMEWORK: VS code

c) LIBARIES: Tkinter, Random, Os, Messagebox

1.Python:

Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

This specially designed Python tutorial will help you learn Python Programming Language in most efficient way, with the topics from basics to advanced (like Web-scraping, Django, Deep-Learning, etc.) with examples.

Below are some facts about Python Programming Language:

1. Python is currently the most widely used multi-purpose, high-level programming language.
2. Python allows programming in Object-Oriented and Procedural paradigms.
3. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
4. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.
5. The biggest strength of Python is huge collection of standard library which can be used for the following:
    - Machine Learning
    - GUI Applications (like Kivy, Tkinter, PyQt etc.)
    - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
    - Image processing (like OpenCV, Pillow)
    - Web scraping (like Scrapy, BeautifulSoup, Selenium)
    - Test frameworks
    - Multimedia
    - Scientific computing
    - Text processing and many more.

2.VS Code:

Visual Studio Code is a code editor in layman's terms. Visual Studio Code is "a free-editor that helps the programmer write code, helps in debugging and corrects the code using the intelli-sense method". In normal terms, it facilitates users to write the code in an easy manner. Many people say that it is half of an IDE and an editor, but the decision is up to the coders. Any program/software that we see or use works on the code that runs in the background. Traditionally coding was used to do in the traditional editors or even in the basic editors like notepad! These editors used to provide basic support to the coders.

Some of them were so basic that it was very difficult in writing basic English level programs in them. As time went by, some programming languages needed a specific framework and support for further coding and development it, which was not possible using these editors. VI Editor, Sublime Text Editor, is one of the many kinds of editors that came into existence. The most prominent and which supports almost every coding language is VISUAL STUDIO CODE. Its features let the user modify the editor as per the usage, which means the user is able to download the libraries from the internet and integrate it with the code as per his requirements.

Tkinter: Is the most commonly used library for developing GUI (Graphical User Interface) in Python. It is a standard Python interface to the Tk GUI toolkit shipped with Python. As Tk and Tkinter are available on most of the Unix platforms as well as on the Windows system, developing GUI applications with Tkinter becomes the fastest and easiest.

Random: First, a prominent disclaimer is necessary. Most random data generated with Python is not fully random in the scientific sense of the word. Rather, it is **pseudorandom**: generated with a pseudorandom number generator (PRNG), which is essentially any algorithm for generating seemingly random but still reproducible data.

"True" random numbers can be generated by, you guessed it, a true random number generator (TRNG). One example is to repeatedly pick up a die off the floor, toss it in the air, and let it land how it may.

CHAPTER 4

DESIGN & IMPLEMENTATION

4.1 ALGORITHM

Step 1: Start.

Step 2: Import required library i.e., thinter, random, os, messagebox, etc .

Step 3: Create class in under which we can store all the variables.

Step 4: Define the functions.

Step 5: Create variables to store the values.

Step 6: Create frames for the different categories.

Step 7: In that frame create entries and labels.

Step 8: Create a Bill Area for displaying the output.

Step 9: Create a button frame to i.e, total, generate bill, clear, exit.

Step10: Define a function for bill area.

Step 11: Assigned all the variables in it and store the result.

Step 12: Define some function which are related with bill i.e., save bill, find bill, clear data, exit.

Step13: Run the code and enter the values.

Step 14: Stop.

4.2 FLOWCHART

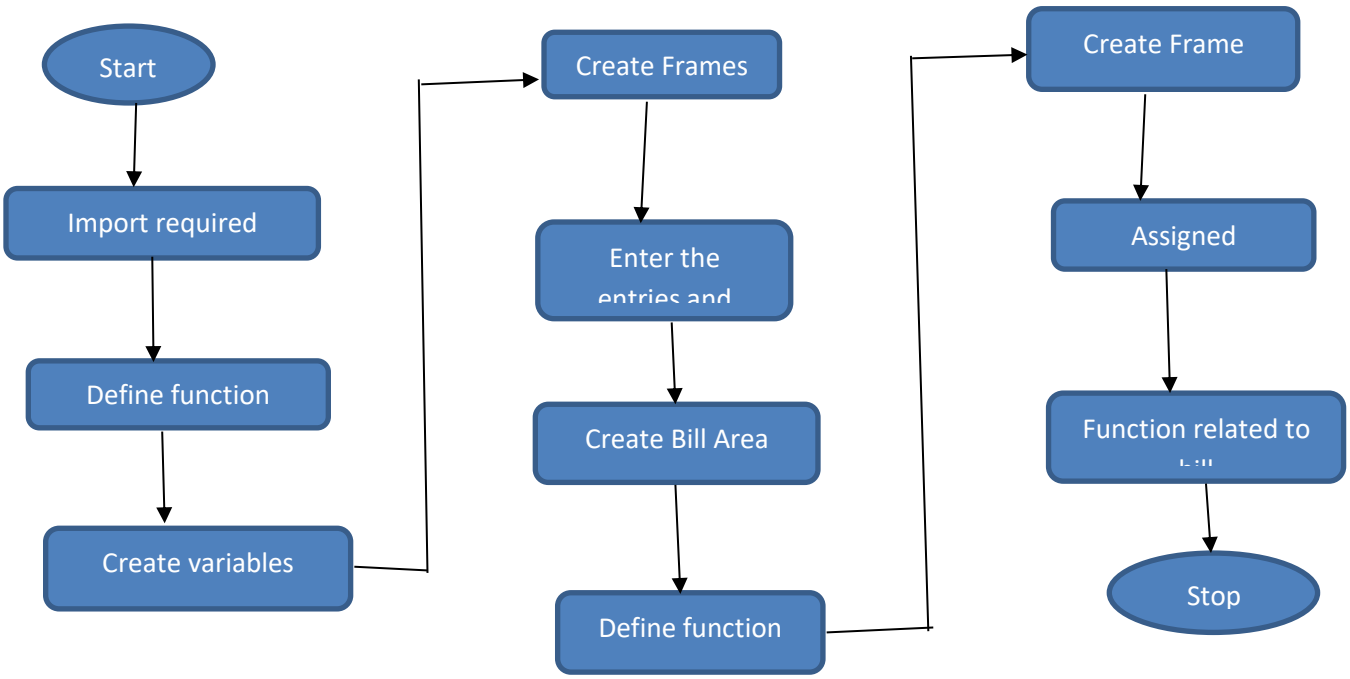

Fig.4.1.1 Flow chart of water billing system

4.3 SOURCE CODE

```
from tkinter import*
import random
import os
from tkinter import messagebox
class Bill_App:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1350x700+0+0")
        self.root.title("Billing System")
        bg_color = "#badc57"
        title = Label(self.root, text="Billing System", font=('times new roman', 30, 'bold'), pady=2,
bd=12, bg="#badc57", fg="Black", relief=GROOVE)
        title.pack(fill=X)
          self.water=IntVar()
        self.meter=IntVar()
        self.sanitary=IntVar()
        self.Dues=IntVar()
        self.other=IntVar()
           self.total_bill = StringVar()
              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()
        self.total=StringVar()
        self.tax=StringVar()
        F1 = LabelFrame(self.root, text="Customer Details", font=('times new roman', 15, 'bold'),
bd=10, fg="Black", bg="#badc57")
```

```
        F1.place(x=0, y=80, relwidth=1)
        cname_lbl = Label(F1, text="Customer Name:", bg=bg_color, font=('times new roman', 15, 'bold'))
        cname_lbl.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=GROOVE)
        cname_txt.grid(row=0, column=1, pady=5, padx=10)
        cphn_lbl = Label(F1, text="Customer Phone:", bg="#badc57", font=('times new roman', 15, 'bold'))
        cphn_lbl.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=GROOVE)
        cphn_txt.grid(row=0, column=3, pady=5, padx=10)
        c_bill_lbl = Label(F1, text="Bill Number:", bg="#badc57", font=('times new roman', 15, 'bold'))
        c_bill_lbl.grid(row=0, column=4, padx=20, pady=5)
        c_bill_txt = Entry(F1, width=15, textvariable=self.search_bill, font='arial 15', bd=7, relief=GROOVE)
        c_bill_txt.grid(row=0, column=5, pady=5, padx=10)
        bil_btn = Button(F1, text="Search", command=self.find_bill, width=10, bd=7, font=('arial', 12, 'bold'), relief=GROOVE)
        bil_btn.grid(row=0, column=6, pady=5, padx=10)

        F3 = LabelFrame(self.root, text="Payment", font=('times new roman', 15, 'bold'), bd=10, fg="Black", bg="#badc57")
        F3.place(x=340, y=180, width=325, height=380)
        p1_lbl = Label(F3, text="Water Charge", font=('times new roman', 16, 'bold'), bg="#badc57", fg="black")
        p1_lbl.grid(row=0, column=0, padx=10, pady=10, sticky='W')
      p1_txt = Entry(F3, width=10, textvariable=self.water, font=('times new roman', 16, 'bold'), bd=5, relief=GROOVE)
        p1_txt.grid(row=0, column=1, padx=10, pady=10)
        p2_lbl = Label(F3, text="Meter Charge", font=('times new roman', 16, 'bold'), bg="#badc57", fg="black")
        p2_lbl.grid(row=1, column=0, padx=10, pady=10, sticky='W')
     p2_txt = Entry(F3, width=10, textvariable=self.meter, font=('times new roman', 16, 'bold'), bd=5, relief=GROOVE)
        p2_txt.grid(row=1, column=1, padx=10, pady=10)
       p3_lbl = Label(F3, text="Sanitary Charge", font=('times new roman', 16, 'bold'), bg="#badc57", fg="black")
        p3_lbl.grid(row=2, column=0, padx=10, pady=10, sticky='W')
        p3_txt = Entry(F3, width=10, textvariable=self.sanitary, font=('times new roman', 16, 'bold'), bd=5, relief=GROOVE)
        p3_txt.grid(row=2, column=1, padx=10, pady=10)
    p4_lbl = Label(F3, text="Dues", font=('times new roman', 16, 'bold'), bg="#badc57", fg="black")
      p4_lbl.grid(row=3, column=0, padx=10, pady=10, sticky='W')
      p4_txt = Entry(F3, width=10, textvariable=self.Dues, font=('times new roman', 16, 'bold'), bd=5, relief=GROOVE)
     p4_txt.grid(row=3, column=1, padx=10, pady=10)
   p5_lbl = Label(F3, text="Other Charge", font=('times new roman', 16, 'bold'), bg="#badc57", fg="black")
     p5_lbl.grid(row=4, column=0, padx=10, pady=10, sticky='W')
```

```
    p5_txt = Entry(F3, width=10, textvariable=self.other, font=('times new roman', 16, 'bold'), bd=5, relief=GROOVE)
     p5_txt.grid(row=4, column=1, padx=10, pady=10)

       F5 = Frame(self.root, bd=10, relief=GROOVE)
       F5.place(x=800, y=180, width=350, height=380)


       bill_title = Label(F5, text="Bill Area", font='arial 15 bold', bd=7, relief=GROOVE)
       bill_title.pack(fill=X)
       scroll_y = Scrollbar(F5, orient=VERTICAL)
       self.txtarea = Text(F5, yscrollcommand=scroll_y.set)
       scroll_y.pack(side=RIGHT, fill=Y)
       scroll_y.config(command=self.txtarea.yview)
       self.txtarea.pack(fill=BOTH, expand=1)

       F6 = LabelFrame(self.root, text="Bill Area", font=('times new roman', 14, 'bold'), bd=10, fg="Black", bg="#badc57")
       F6.place(x=0, y=560, relwidth=1, height=140)

       m2_lbl = Label(F6, text="Total Price", font=('times new roman', 14, 'bold'), bg="#badc57", fg="black")
       m2_lbl.grid(row=1, column=0, padx=20, pady=1, sticky='W')
       m2_txt = Entry(F6, width=18, textvariable=self.total, font='arial 10 bold', bd=7, relief=GROOVE)
       m2_txt.grid(row=1, column=1, padx=18, pady=1)

       m5_lbl = Label(F6, text="Tax", font=('times new roman', 14, 'bold'), bg="#badc57", fg="black")
       m5_lbl.grid(row=1, column=2, padx=20, pady=1, sticky='W')
       m5_txt = Entry(F6, width=18, textvariable=self.tax, font='arial 10 bold', bd=7, relief=GROOVE)
       m5_txt.grid(row=1, column=3, padx=18, pady=1)
       btn_f = Frame(F6, bd=7, relief=GROOVE)
       btn_f.place(x=760, width=580, height=105)
       t_btn = Button(btn_f, command=self.same, text="Total", bg="#535C68", bd=2, fg="white", pady=15, width=12, font='arial 13 bold')
       t_btn.grid(row=0, column=0, padx=5, pady=5)
       t_btn = Button(btn_f, command=self.bill_area, text="Generate Bill", bd=2, bg="#535C68", fg="white", pady=12, width=12, font='arial 13 bold')
       t_btn.grid(row=0, column=1, padx=5, pady=5)
       t_btn = Button(btn_f, command=self.clear_data, text="Clear", bg="#535C68", bd=2, fg="white", pady=15, width=12, font='arial 13 bold')
       t_btn.grid(row=0, column=2, padx=5, pady=5)
       t_btn = Button(btn_f, command=self.exit_app, text="Exit", bd=2, bg="#535C68", fg="white", pady=15, width=12, font='arial 13 bold')
       t_btn.grid(row=0, column=3, padx=5, pady=5)
       self.welcome_bill()
   def same(self):
       self.w_p=self.water.get()*100
       self.m_p=self.meter.get()*75
       self.s_p=self.sanitary.get()*150
       self.d_p=self.Dues.get()*5
```

```
        self.o_p=self.other.get()*250
        self.price_total=float(
                        self.w_p+
                        self.m_p+
                        self.s_p+
                        self.d_p+
                        self.o_p
                        )
        self.total.set("Rs." +str(self.price_total))
        self.t_tax=round((self.price_total*0.05),2)
        self.tax.set("Rs."+str(self.t_tax))
        self.total_bill = float(self.price_total+self.tax)
    def welcome_bill(self):
        self.txtarea.delete('1.0', END)
        self.txtarea.insert(END, "\tWelcome to WBS")
        self.txtarea.insert(END, f"\nBill Number:{self.bill_no.get()}")
        self.txtarea.insert(END, f"\nCustomer Name:{self.c_name.get()}")
        self.txtarea.insert(END, f"\nPhone Number{self.c_phone.get()}")
        self.txtarea.insert(END, f"\n======================================")
        self.txtarea.insert(END,f"\n Particular \t\tUnit \t\tPrice")
    def bill_area(self):
        if self.c_name.get() == " " or self.c_phone.get() == " ":
            messagebox.showerror("Error", "Customer Details Are Must")
        elif self.total_bill.get() == "Rs. 0.0" :
            messagebox.showerror("Error", "No Product Purchased")
        else:
            self.welcome_bill()
        if self.water.get():
            self.txtarea.insert(END, f"\n Water\t\t{self.water.get()}\t\t{self.w_p}")
        if self.meter.get() != 0:
            self.txtarea.insert(END, f"\n Meter\t\t{self.meter.get()}\t\t{self.m_p}")
        if self.sanitary.get() != 0:
            self.txtarea.insert(END, f"\n Sanitary\t\t{self.sanitary.get()}\t\t{self.s_p}")
        if self.Dues.get() != 0:
            self.txtarea.insert(END, f"\n Dues\t\t{self.Dues.get()}\t\t{self.d_p}")
        if self.other.get() != 0:
            self.txtarea.insert(END, f"\n Other\t\t{self.other.get()}\t\t{self.o_p}")
             if self.grocery_tax.get() != '0.0':
            self.txtarea.insert(END, f"\n Tax\t\t\t{self.grocery_tax.get()}")
        self.txtarea.insert(END, f"\n Total Bill:\t\t\t Rs.{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()
            messagebox.showinfo("Saved", f"Bill no:{self.bill_no.get()} Saved Successfully")
        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("Clear", "Do you really want to Clear?")
        if op > 0:
            self.water.set(0)
            self.meter.set(0)
            self.sanitary.set(0)
            self.Dues.set(0)
            self.other.set(0)
             self.total_bill.set("")
            self.t_tax.set("")
            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()
```

CHAPTER 5

RESULT & DISCUSSION

5.1 OUTPUT

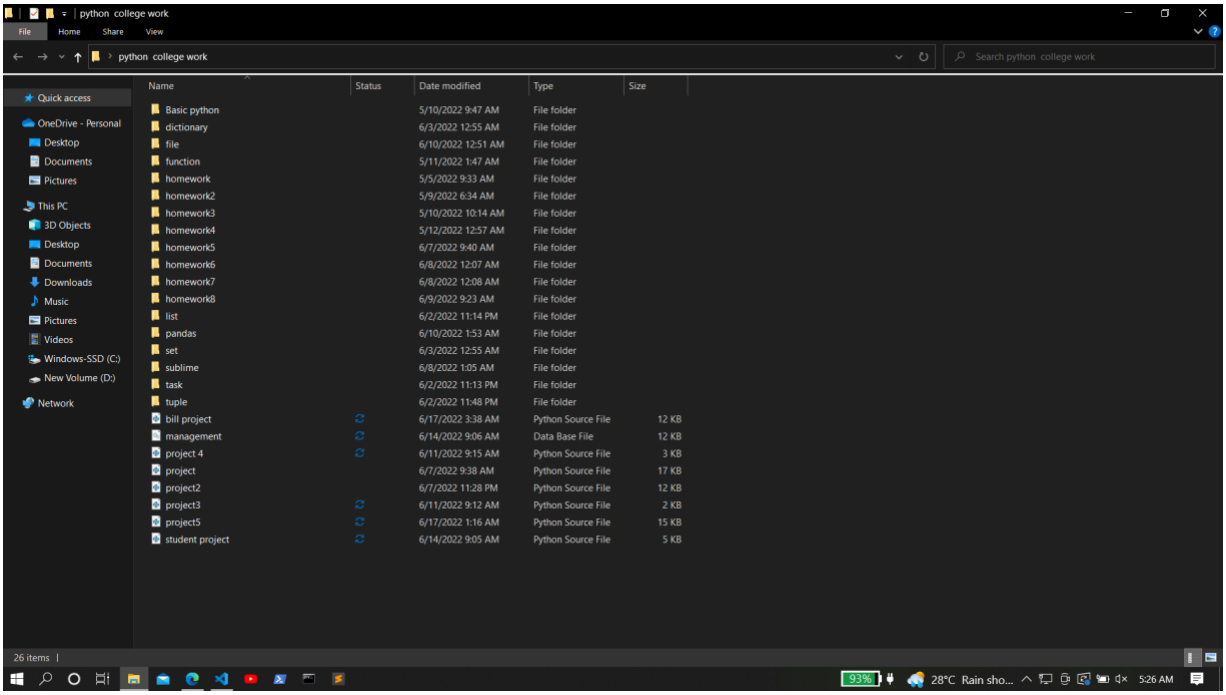

Fig 5.1.1. File Before Execution of code

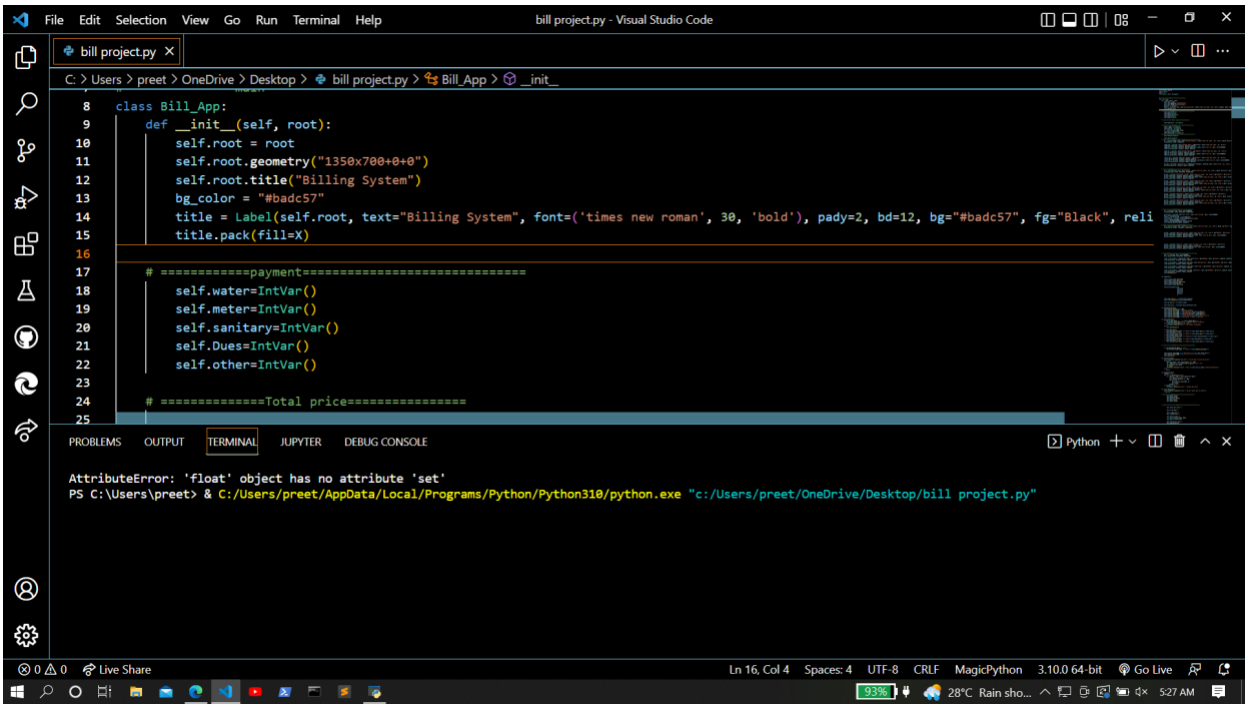

Fig 5.1.2. Execution of code

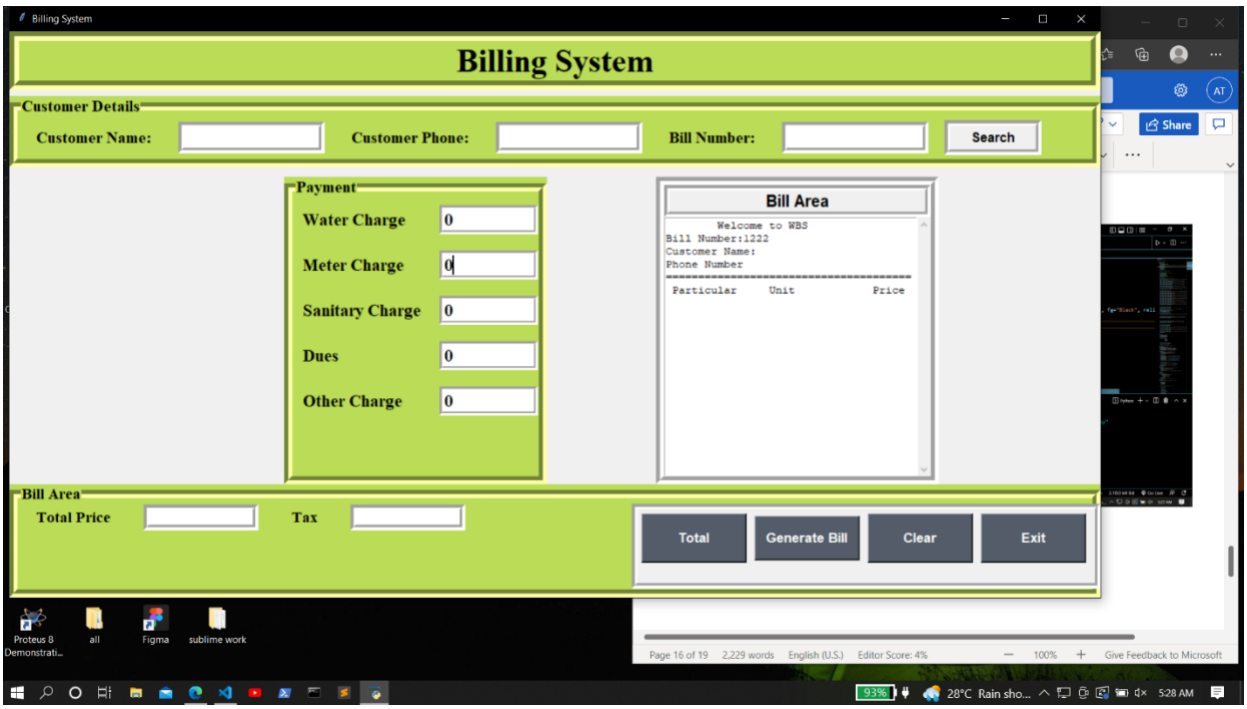

Fig 5.1.3. Output Screen

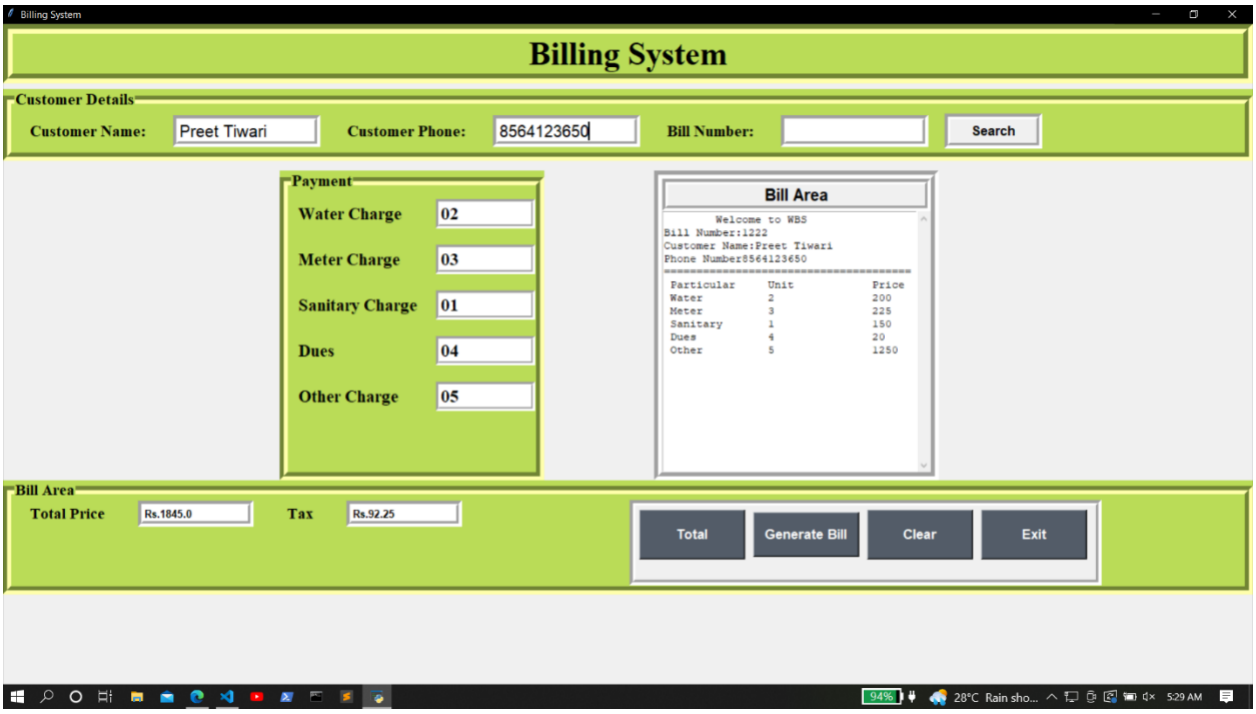

Fig 5.1.4. Saving the Data

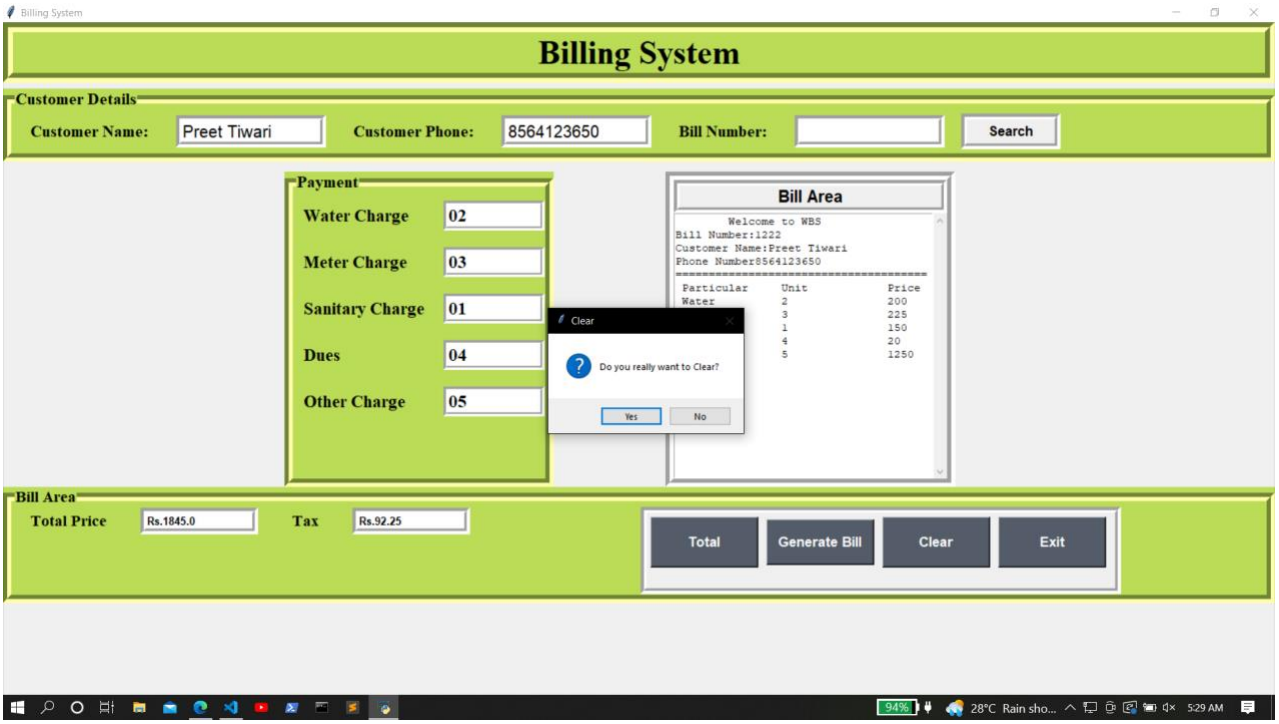

Fig 5.1.5. Clear the Data

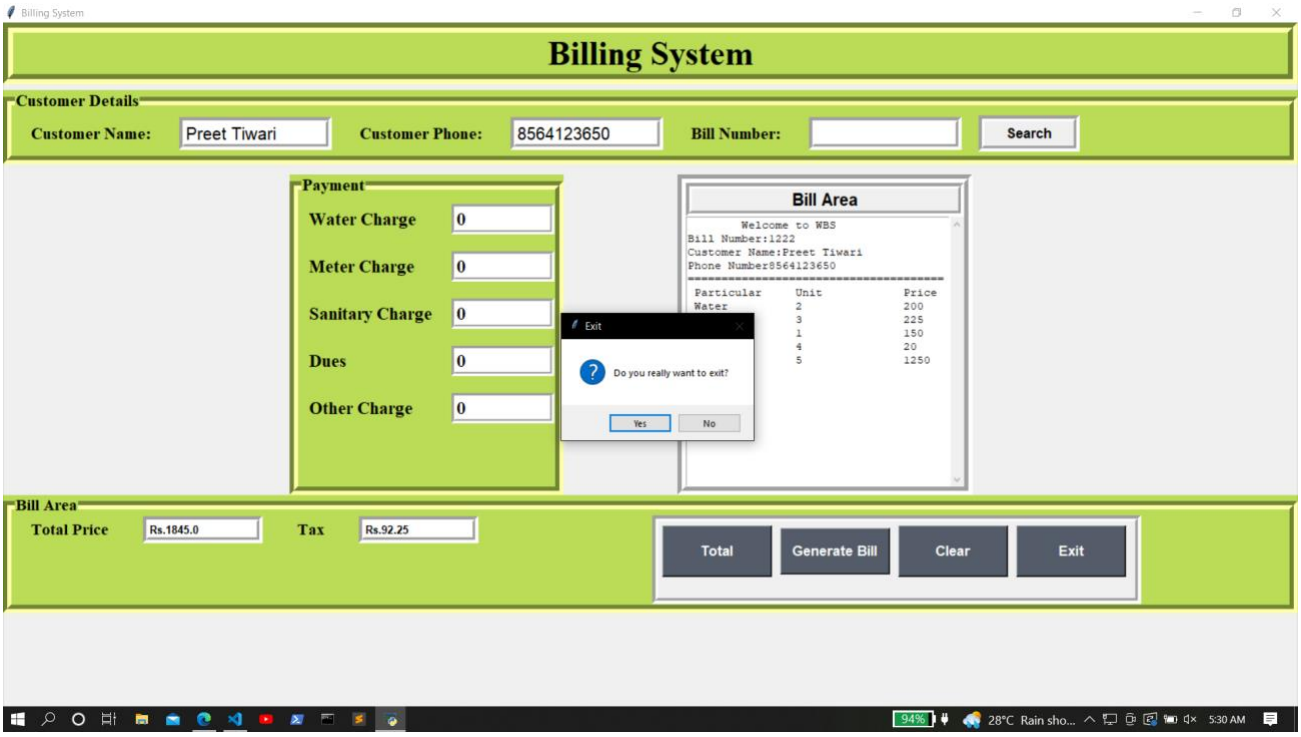

Fig 5.1.6. Exit the system

5.2 DISCUSSION

In this project the **Fig.5.1.1** shows the python file before the execution of the code. The **Fig.5.1.2** shows the code which we have written in that file or the execution of the program the **Fig.5.1.2** shows that the code which we have written is working. After the execution is done and the code is compiled a screen is shown which display the format or the interface of the water billing system. **Fig.5.1.3**

shows the interface of the billing system in this we can find all the data of the customer. **Fig.5.1.4** shows the data of the customer as the data is saved in the program as like water charge, meter charge, sanitary charge, dues, other charge, name, phone number, total, tax and the it shows the whole data in the bill area. After clicking Generate bill it will display the data in bill area. **Fig 5.1.5** shows as we want to clear the data of the system and reset as we click on clear button it will display message "Do you want to clear data?". **Fig 5.1.6** shows the exit from the program after clicking on exit button it will display "Do you want to exit?"

5.3 APPLICATION

The system for water billing using **flow sensor is proposed to tackle the problem of water monitoring and saving**. The system is also integrated with automated GSM message service if the flow sensors send high end flow data.

Water billing system is a necessary tool to assist small municipalities, utility providers, bookkeepers, operators, managers, and auditors in unifying their water billing services, in order to provide consistent and accurate billing information to clients and service recipients.

CHAPTER 6

CONCLUSION

The significance of the study is that it will provide a more accurate and reliable approach to the computation of water bills as well as the storage and retrieval of bill records of consumers. The study will also serve as a useful reference material for other researchers seeking related information on **water billing.**

In this we can manage all the data of customer and find which customer have how much bill. The software allows user or the customer to find all him/herself.

The water billing system allows user to manage information.The Computerize **Water billing system** to be develop would answer all the limitation of the present manual system. It would increase accuracy and reliability of system by automating and incorporating formulas to computation to make transaction and computation accurate and faster. Records would be updated and manage efficiently.

REFERENCE

https://www.w3schools.com

https://www.geeksforgeeks.org

https://www.jetbrains.com/pycharm

https://code.visualstudio.com