THE HINDU SENIOR SECONDARY SCHOOL

TRIPLICANE, CHENNAI – 600005

COMPUTER SCIENCE PROJECT

2021 – 2022

TOPIC: PHARMACY BILLING APP USING PYTHON AND MySQL CONNECTIVITY

THE HINDU SENIOR SECONDARY SCHOOL,

CHENNAI – 600005

REGISTER NUMBER: | | | | | | | | |

BONAFIDE CERTIFICATE

Certified to be the Bonafide for the project work done by Master/Miss_____ of class XII in **THE HINDU SENIOR SECONDARY SCHOOL, CHENNAI** during the year 2022-2023.

Dated_____ Signature of the teacher

(P.G.T in Computer Science)

School Seal

Submitted for All India Senior Secondary Practical Examination held in _____ at _____ Chennai

Dated_____ Signature of External Examiner

External Examiner number

ACKNOWLEDGEMENT

There are many people who have helped us in making this project a successful one. We would like to thank the school management and principal Smt. Alamelu Raghavan. We extend our gratitude to our beloved computer science teacher, Smt. S. Siva Prabha for her guidance.

We also thank our parents for their support. We cannot forget our sincere thanks to our classmates who have helped us to conduct this project work successfully and for their valuable advice and support which we received from time to time.

We wish to express our deep gratitude and sincere thanks to those helping hands without whom this project would have been completed.

INDEX

GOODWILL PHARMACY

-FOR QUALITY MEDICINAL DRUGS

DONE BY:

Hemanth Kumar

Arudhran

Abhishek

Overview:

This is a Medicine Billing Application by using Python-MYSQL to make this project. We have used Python Module Tkinter to Create GUI and MYSQL Connector to Connect to MYSQL Database.

Here Python-Tkinter Works as a front-end and MYSQL works as a back-end.

Applications:

1. With some Alterations in the code, it can also be used in Hospitals to record Incoming and Outgoing Patients.
2. It can be used in Pharmacies to check the availability of drugs and check the retails of the day. It will help you keep track of the cash flow in your business.

3. It Gets the input medicine name from the Customer and generates a bill and saves a copy in MYSQL DATABASE.

Main Aim:

1. To Establish a Connection between MYSQL and Python.
2. To prepare GUI for Billing Application
3. To Code for the following functions:-
   i. DISPLAYING older bills from Database
   ii. ADDING new bills to Database
   iii. MODIFY the existing bill from Database
   iv. DELETING an Item from the Bill
   v. DELETING an Entire Bill from Database

Front-End: Tkinter

We have used Python Tkinter Module to create GUI & Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

Back-End: MYSQL

It is open source, Non pirated, reliable, compatible with all major hosting providers, cost-effective, and easy to manage. Many organizations are leveraging the data security and strong transactional support offered by MySQL to secure online transactions and enhance customer interactions.

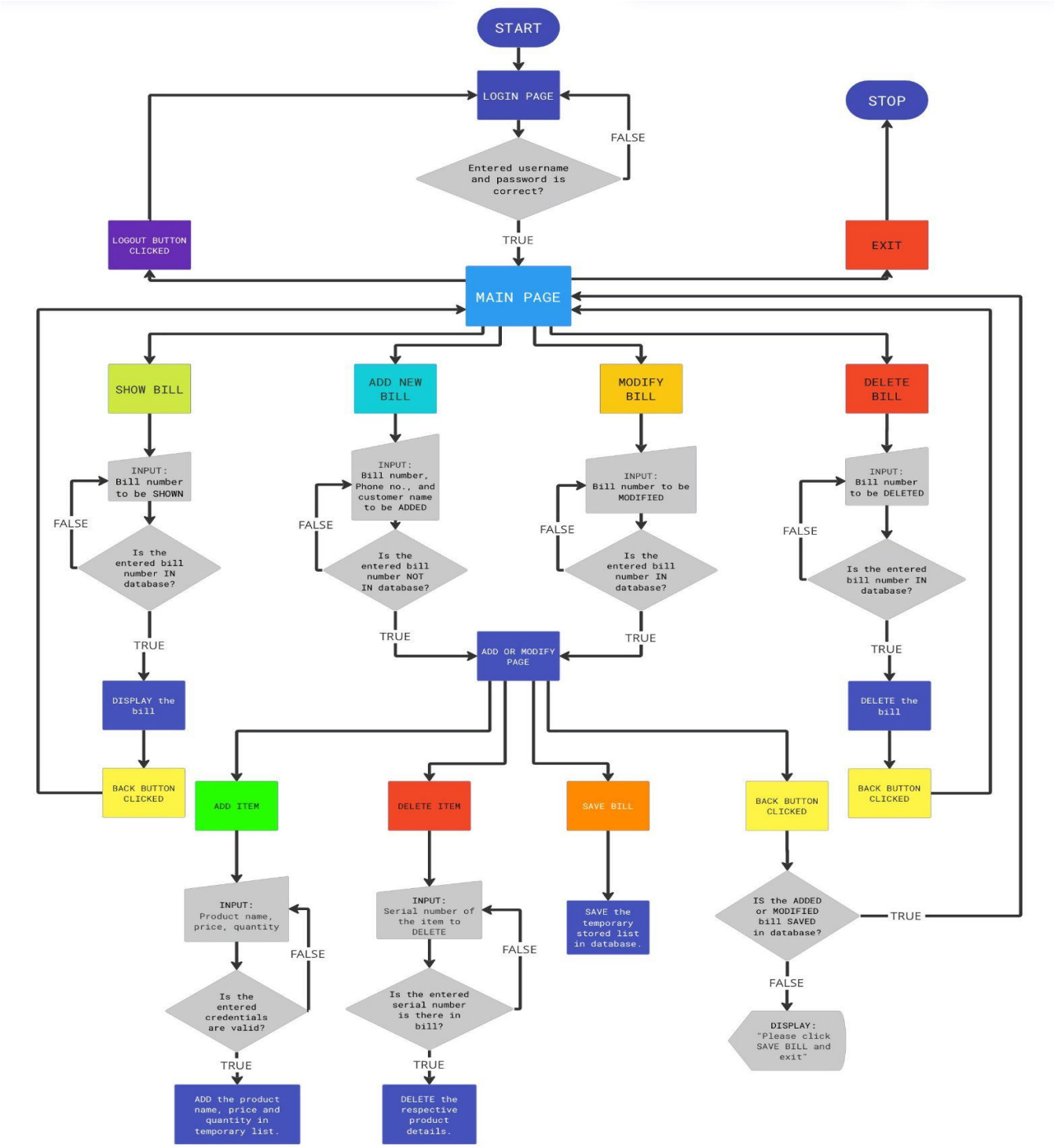
HARDWARE REQUIREMENT:

- CPU compatible with your OS(Arm 64-bit, x86 64-bit, etc. Preferably quad-core processor).
- Minimum of 4GB of RAM. (8GB is recommended).
- Minimum of 16GB of free storage.

SOFTWARE REQUIREMENTS:

- OS such as Linux, MacOS, Windows, etc.
- Python 3.7 version or above.
- Python modules such as 'Tkinter', 'MySQL connector' which must be installed using command prompt(on Windows OS) or terminal(on Linux and MacOS).
- MySQL database version 5.6, 5.7, 8.0 or greater.

Flow chart:


START
LOGIN PAGE
FALSE
Entered username and password is correct?
TRUE
STOP
LOGOUT BUTTON CLICKED
EXIT
MAIN PAGE
SHOW BILL
ADD NEW BILL
MODIFY BILL
DELETE BILL
INPUT: Bill number to be SHOWN
INPUT: Bill number, Phone no., and customer name to be ADDED
INPUT: Bill number to be MODIFIED
INPUT: Bill number to be DELETED
FALSE
Is the entered bill number IN database?
Is the entered bill number NOT IN database?
Is the entered bill number IN database?
Is the entered bill number IN database?
TRUE
ADD OR MODIFY PAGE
DISPLAY the bill
DELETE the bill
BACK BUTTON CLICKED
ADD ITEM
DELETE ITEM
SAVE BILL
BACK BUTTON CLICKED
BACK BUTTON CLICKED
INPUT: Product name, price, quantity
INPUT: Serial number of the item to DELETE
SAVE the temporary stored list in database.
IS the ADDED or MODIFIED bill SAVED in database?
Is the entered credentials are valid?
Is the entered serial number is there in bill?
DISPLAY: "Please click SAVE BILL and exit"
ADD the product name, price and quantity in temporary list.
DELETE the respective product details.

SOURCE CODE:

1. GUI and Python :

```
#importing all the required packages
from tkinter import *
from tkinter import messagebox
from database import *

#colors
black = "#000000"
white = "#FFFFFF"
red = "#FF0000"
green = "#00FF00"
blue = "#0066FF"
yellow = "#FFFF00"
cyan = "#00FFFF"
purple = "#A64DFF"
orange = "#FF8533"

#preping a window
root = Tk()
root.title("GOODWILL PHARMACY")
root.configure(background = black)

#setting an icon for the app
icon_image = PhotoImage(file = r"billing app icon.png")
root.tk.call('wm', 'iconphoto', root._w, icon_image)

#clearing all the widgets from the page to make room for the next page
def clear_page():
   for widget in root.grid_slaves():
       widget.destroy()
```

```
#command for the exit button
def exit():
   global exit_window
   exit_window = Tk()
   exit_window.title("EXIT")

   exit_label = Label(exit_window, text = "DO YOU WANT TO EXIT?")
   exit_label.grid(row = 0, column = 0, padx = 2, pady = 2, columnspan = 2)

   yes_button = Button(exit_window, text = "YES", borderwidth = 2, command = lambda :
destroy_app(1))
   yes_button.grid(row = 1, column = 0, padx = 2, pady = 2)

   no_button = Button(exit_window, text = "NO", background = black, foreground = white,
borderwidth = 2, command = lambda : destroy_app(0))
   no_button.grid(row = 1, column = 1, padx = 2, pady = 2)


#this function destroys the entire app by closing the all the windows and quitting
def destroy_app(a):
   global root
   if a == 1:
       root.destroy()
       exit_window.destroy()
       quit()
   else:
       exit_window.destroy()


#destroying all the widgets in the left side frame
def clear_left_frame():
```

```
    for widgets in left_side_frame.winfo_children():
        widgets.destroy()


#we set the output_box state to disabled so that the user can't change anything in the output box
#It is only used to see the output bill
#changing the state from disabled to normal so that the system can edit the content in the output box
def clear_a_line_in_text_box(line):
    output_box.config(state = NORMAL)
    output_box.delete(float(line), float(line + 1))
    output_box.config(state = DISABLED)


def clear_the_entire_text_box():
    output_box.config(state = NORMAL)
    output_box.delete(1.0, END)
    output_box.config(state = DISABLED)


def clear_entire_text_box_from_start_line(line):
    output_box.config(state = NORMAL)
    output_box.delete(float(line), END)
    output_box.config(state = DISABLED)


def display_in_output_box(string, position = -1):
    output_box.config(state = NORMAL)
    output_box.insert(f"end {position} chars", string)
    output_box.config(state = DISABLED)
```

```
#login gui for the user with login button.
def login_gui():
   #entry boxes for username and password and passing the parameters to the button function

   #this is for clearing the page when the login button is pressed
   try:
       clear_page()
   except:
       pass

   username_name_label = Label(root, text = "Username:", foreground = white, background =
black)
   username_name_label.grid(row = 0, column = 0, padx = 4, pady = 4, sticky = EW)

   global username_name_entry
   username_name_entry = Entry(root)
   username_name_entry.grid(row = 0, column = 1, padx = 4, pady = 4)

   password_label = Label(root, text = "Password:", foreground = white, background = black)
   password_label.grid(row = 1, column = 0, padx = 4, pady = 4, sticky = EW)

   global password_entry
   password_entry = Entry(root)
   password_entry.grid(row = 1, column = 1, padx = 4, pady = 4)

   login_button = Button(root, text = "LOGIN", foreground = black, background = green ,command
= lambda : login_button_func(username_name_entry.get(), password_entry.get()))
   login_button.grid(row = 3, column = 1, padx = 4, pady = 4, sticky = E)



#login code to execute to authenticate the username  and the password entered by the user
def login_button_func(username_input, password_input):
```

```
    if authentication(username_input, password_input):
        #savind the username to know who made pa bill
        global username
        username = username_name_entry.get()

        #wiping the entire page and changing the gui to the mainpage
        clear_page()

        #creating two frames to sperate the output box and all the buttons and lables
        global left_side_frame
        global right_side_frame

        left_side_frame = Frame(root, background = black)
        right_side_frame = Frame(root, background = black)

        left_side_frame.grid(row=0, column=0)
        right_side_frame.grid(row=0, column=1)

        main_page()

    else:
        #error pop up,clear the entry widgets ask the user to re-enter all the creadentials
        messagebox.showerror("ACCESS DENIED!!","Wrong username or password entered!")
        password_entry.delete(0, END)
        username_name_entry.delete(0, END)



#creating main page where user can access all the commands like "new bill"
def main_page():

    clear_left_frame()
    #clear_the_entire_text_box()
```

```
    #getting data of main_bill from database.
    global database_bill_no
    database_bill_no = get_bill_no()

    global main_bill_raw
    main_bill_raw = get_main_bill()

    #LEFT FRAME
    home_label = Label(left_side_frame, text = "HOME", width = 30, height = 5, background =
black, foreground = orange, font = (100))
    home_label.grid(row = 0, column = 0 , padx = 10 , pady = 20, sticky = EW)

    show_bill_button = Button(left_side_frame, text = "SHOW A BILL", width = 25, background =
green, foreground = black, command = lambda : show_bill_page())
    show_bill_button.grid(row = 1, column = 0, padx = 10, pady = 20)

    add_bill_button = Button(left_side_frame, text = "ADD A NEW BILL" , width = 25, background
= cyan, foreground = black, command = lambda : add_bill_page())
    add_bill_button.grid(row = 2, column = 0, padx = 10, pady = 20)

    modify_bill_button = Button(left_side_frame, text = "MODIFY A BILL", width = 25, background
= orange, foreground = black, command = lambda : modify_bill_page())
    modify_bill_button.grid(row = 3, column = 0, padx = 10, pady = 20)

    delete_bill_button = Button(left_side_frame, text = "DELETE A BILL", width = 25, background
= red, foreground = black, command = lambda : delete_bill_page())
    delete_bill_button.grid(row = 4, column = 0, padx = 10, pady = 20)

    exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground = black,
command = lambda : exit())
    exit_button.grid(row = 5, column = 0, sticky = W, padx = 10, pady = 20)

    log_out_button = Button(left_side_frame, text = "LOGOUT", background = purple, foreground =
black, command = lambda : login_gui())
    log_out_button.grid(row = 5, column = 0, sticky = E, padx = 10, pady = 20)
```

```
    #RIGHT FRAME
    global output_box

    output_box = Text(right_side_frame, width = 95,height = 30, background = white, foreground
= black, state = DISABLED)
    output_box.grid(column = 0, row = 0, padx = 5, pady = 5)

    #show the main bill in the output box
    text =
"""+-------------+---------------------------+-----------------+--------------+---------------
--+
| BILL NUMBER | CUSTOMER NAME             | PHONE NO.       | TOTAL        | BILLER NAME     |
+-------------+---------------------------+-----------------+--------------+-----------------+
\n"""

    for items in main_bill_raw:
        text += f"| {items[0]}{((11 - len(str(items[0]))) * ' ')} "
        text += f"| {items[1]}{((25 - len(str(items[1]))) * ' ')} "
        text += f"| {items[2]}{((15 - len(str(items[2]))) * ' ')} "
        text += f"| {items[3]}{((12 - len(str(items[3]))) * ' ')} "
        text += f"| {items[4]}{((15 - len(str(items[4]))) * ' ')} "
        text +=
"|\n|-------------+---------------------------+-----------------+--------------+--------------
---|\n"
    text = text[0:len(text)-95]
    text +=
"+-------------+---------------------------+-----------------+--------------+-----------------
+"

    display_in_output_box(text)


#this page is to show the desired bill entered by the user
def show_bill_page():
```

```
    #destroying all widgets in left frame to make room for new widgets
    clear_left_frame()

    input_label=Label(left_side_frame, text= "Enter the bill number to
display",background=black, foreground= white)
    input_label.grid(row=0,column=1, padx = 5,pady = 10, sticky=EW)

    global input_bill_no
    input_bill_no=Entry(left_side_frame)
    input_bill_no.grid(row=1,column=1, padx = 5,pady = 10, sticky=EW)

    display_button=Button(left_side_frame,text = "DISPLAY BILL",background = green ,foreground
= black, command = lambda : show_bill_validation())
    display_button.grid(row=2,column=1, padx = 5,pady = 10, sticky=EW)

    exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground = black,
command = lambda : exit())
    exit_button.grid(row = 3, column = 0, padx = 5, pady = 15, sticky = W)

    back_button = Button(left_side_frame, text = "BACK", background = purple, foreground =
black, command = lambda : main_page())
    back_button.grid(row =3, column = 2, padx = 5, pady = 15, sticky = E)



#validates whether the given bill is there in the database and the entered bill is an integer
def show_bill_validation():
    show_bill_number = input_bill_no.get()

    if show_bill_number.isnumeric():

        show_bill_number = int(show_bill_number)

        if show_bill_number in database_bill_no:
```

```
            clear_left_frame()
            exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground
= black, command = lambda : exit())
            exit_button.grid(row = 1, column = 0, padx = 10, pady = 15, sticky = W)


            back_button = Button(left_side_frame, text = "BACK", background = purple,
foreground = black, command = lambda : main_page())
            back_button.grid(row = 1, column = 1, padx = 10, pady = 15, sticky = E)


            bill = get_customer_bill(show_bill_number)


            #SHOWING THE BILL
            text = """+------+-------------------------------------+--------+-----+-----------+
| S.NO | PRODUCT NAME                        | PRICE  | QTY | TOTAL     |
+------+-------------------------------------+--------+-----+-----------+\n"""
            grand_tot = 0


            for i in range(len(bill)):


                text += f"| {i + 1}.{((3 - len(str(i + 1))) * ' ')} "
                text += f"| {bill[i][0]}{((35 - len(str(bill[i][0]))) * ' ')} "
                text += f"| {bill[i][1]}{((6 - len(str(bill[i][1]))) * ' ')} "
                text += f"| {bill[i][2]}{((3 - len(str(bill[i][2]))) * ' ')} "
                text += f"| {bill[i][3]}{((9 - len(str(bill[i][3]))) * ' ')} "
                text +=
"|\n|------+-------------------------------------+--------+-----+-----------|\n"
                grand_tot += bill[i][3]


            text = text[0:len(text)-75]
            text +=
f"""\n+------+-------------------------------------+--------+-----+-----------+
|                                                GRAND TOTAL : {grand_tot}{((10 -
len(str(grand_tot))) * ' ')}|
+------+-------------------------------------+--------+-----+-----------+"""
```

```
            #making room for the desired output
            clear_the_entire_text_box()
            #displaying the bill in the output box
            display_in_output_box(text)

    #showing error to the user when the entered bill no is not integer or
    #when the entered bill no is not there in database
        else:
            messagebox.showerror("BILL DIDN'T EXIST", "Bill no doesn't exist in the database")
            input_bill_no.delete(0, END)
    else:
        messagebox.showerror("BILL DIDN'T EXIST", "Bill no doesn't exist in the database")
        input_bill_no.delete(0, END)



def add_bill_page():

    #destroying all the widgets in the left side frame
    clear_left_frame()

    #displaying all the widgets for getting customer name, phone number, bill number
    add_bill_page_label = Label(left_side_frame, text = "ADD A NEW BILL", background = black,
foreground = orange, font = (100))
    add_bill_page_label.grid(row = 0, column = 0, columnspan = 2, padx = 10, pady = 20, sticky
= N)

    label_customer_name = Label(left_side_frame, text = "Customer Name:", foreground = white,
background = black)
    label_customer_name.grid(row = 1, column = 0, padx = 10, pady = 20, sticky = NS)

    global customer_name_entry
    customer_name_entry = Entry(left_side_frame)
    customer_name_entry.grid(row = 1, column = 1, padx = 10, pady = 20)
```

```
    label_phone_number = Label(left_side_frame, text = "Phone Number:", foreground = white,
background = black)
    label_phone_number.grid(row = 2, column = 0, padx = 10, pady = 20, sticky = NS)

    global phone_number_entry
    phone_number_entry = Entry(left_side_frame)
    phone_number_entry.grid(row = 2, column = 1, padx = 10, pady = 20)

    label_bill_number = Label(left_side_frame, text = "Bill Number:", foreground = white,
background = black)
    label_bill_number.grid(row = 3, column = 0, padx = 10, pady = 20, sticky = NS)

    global bill_number_entry
    bill_number_entry = Entry(left_side_frame)
    bill_number_entry.grid(row = 3, column = 1, padx = 10, pady = 20)

    create_new_bill_button = Button(left_side_frame, text = "CREATE NEW BILL", background =
green, foreground = black, command = lambda : add_validation())
    create_new_bill_button.grid(row = 4, column = 0, columnspan = 2, padx = 10, pady = 20)

    exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground = black,
command = lambda : exit())
    exit_button.grid(row = 5, column = 0, sticky = W, padx = 10, pady = 20)

    back_button = Button(left_side_frame, text = "BACK", background = purple, foreground =
black, command = lambda : main_page())
    back_button.grid(row = 5, column = 1, sticky = E, padx = 10, pady = 20)

def add_validation():
    #this function is to validate whether the given credentials are correct
    global customer_name
    global phone_number
    global bill_number
```

```
    customer_name = customer_name_entry.get().lstrip().rstrip()
    phone_number = phone_number_entry.get().lstrip().rstrip()
    bill_number = bill_number_entry.get().replace(" ","")

    message = ""
    #to chech whether the given bill number is all numeric so that we can type cast it to
integer
    if bill_number.isnumeric() == False:
        message += "# Bill no. can't take alphabets or special chracters!!\n"
    else:
        bill_number = int(bill_number)

    #making sure the given credentials are of correct length and data type
    if str(bill_number).isnumeric() == False or bill_number in database_bill_no or
len(customer_name) > 15 or len(str(bill_number)) > 10 or len(phone_number) > 15 or
customer_name == "" or phone_number == "":

        if bill_number in database_bill_no:
            message += "# Bill no. already exist!\n"
        if len(customer_name) > 15:
            message += "# Customer name has greater than 15 characters!\n"
        if len(str(bill_number)) > 10:
            message += "# Bill no. has greater than 10 characters!\n"
        if len(phone_number) > 15:
            message += "# Phone no. has greater than 15 characters!"

        #informing user about the invalid input given
        messagebox.showwarning(title = "INVALID INPUTS!!", message = message)

    else:
        #next operation page
        customer_name = customer_name.title()
        add_modify_operation_page("add")
```

```
#this page is used by both add and modify section of code
#with few alteration we can change the code by passing different parameter
def add_modify_operation_page(add_or_modify):

    #cleaning the left frame to add new widgets and clearing the right output textbox to
    #display the bill items and list them
    clear_left_frame()
    clear_the_entire_text_box()

    #widgets for left frame
    add_bill_page_label = Label(left_side_frame, text = "ADD A NEW BILL", background = black,
foreground = orange, font = (100))
    add_bill_page_label.grid(row = 0, column = 0, columnspan = 2, padx = 10, pady = 15, sticky
= N)

    label_product_name = Label(left_side_frame, text = "Product name:", foreground = white,
background = black)
    label_product_name.grid(row = 1, column = 0, padx = 10 , pady = 15)

    global entry_product_name
    entry_product_name = Entry(left_side_frame)
    entry_product_name.grid(row = 1, column = 1, padx = 10 , pady = 15)

    label_product_price = Label(left_side_frame, text = "Product Price:", foreground = white,
background = black)
    label_product_price.grid(row = 2, column = 0, padx = 10 , pady = 15)

    global entry_product_price
    entry_product_price = Entry(left_side_frame)
    entry_product_price.grid(row = 2, column = 1, padx = 10 , pady = 15)
```

```
    label_product_quantity = Label(left_side_frame, text = "Quantity:", foreground = white,
background = black)

    label_product_quantity.grid(row = 3, column = 0, padx = 10 , pady = 15)


    global entry_product_quantity

    entry_product_quantity = Entry(left_side_frame)

    entry_product_quantity.grid(row = 3, column = 1, padx = 10 , pady = 15)


    add_product_button = Button(left_side_frame, text = "ADD PRODUCT", foreground = black,
background = green, command = lambda : add_product_function_validation())

    add_product_button.grid(row = 4, column = 0, columnspan = 2, sticky = EW, padx = 10 , pady
= 15)


    delete_product_label = Label(left_side_frame, text = "Enter the S.No to remove :",
foreground = white, background = black)

    delete_product_label.grid(row = 5, column = 0, columnspan = 2, sticky = W, padx = 10 , pady
= 15)


    global delete_product_entry

    delete_product_entry = Entry(left_side_frame, width = 10)

    delete_product_entry.grid(row = 5, column = 0, columnspan = 2, sticky = E, padx = 10 , pady
= 15)


    delete_product_button = Button(left_side_frame, text = "DELETE PRODUCT", foreground =
black, background = cyan, command = lambda : delete_an_item(delete_product_entry.get()))

    delete_product_button.grid(row = 6, column = 0, columnspan = 2, sticky = EW, padx = 10 ,
pady = 15)


    save_button = Button(left_side_frame, text = "SAVE BILL", width = 5, foreground = black,
background = orange, command = lambda : save_bill_database())

    save_button.grid(row = 7, column = 0, columnspan = 2, padx = 10, pady = 15, sticky = EW)


    exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground = black,
command = lambda : exit())

    exit_button.grid(row = 8, column = 0, padx = 10, pady = 15, sticky = W)
```

```
   back_button = Button(left_side_frame, text = "BACK", background = purple, foreground =
black, command = lambda : back_button_validation())
   back_button.grid(row = 8, column = 1, padx = 10, pady = 15, sticky = E)



   #preparing for the output box
   global text
   text = """+------+-------------------------------------+--------+-----+-----------+
| S.NO | PRODUCT NAME                        | PRICE  | QTY | TOTAL     |
+------+-------------------------------------+--------+-----+-----------+\n"""

   global serial_no
   global grand_total
   global products
   global price
   global quantity
   global total
   serial_no, grand_total, products, price, quantity, total = 0, 0, [], [], [], []

   #by passing different parameter we can choose between modify and add functions
   if add_or_modify == "modify":

       customer_bill = get_customer_bill(bill_number)
       for i in range(len(customer_bill)):

           text += f"| {i + 1}.{((3 - len(str(i + 1))) * ' ')} "
           text += f"| {customer_bill[i][0]}{((35 - len(str(customer_bill[i][0]))) * ' ')} "
           text += f"| {customer_bill[i][1]}{((6 - len(str(customer_bill[i][1]))) * ' ')} "
           text += f"| {customer_bill[i][2]}{((3 - len(str(customer_bill[i][2]))) * ' ')} "
           text += f"| {customer_bill[i][3]}{((9 - len(str(customer_bill[i][3]))) * ' ')} "
           text +=
"|\n|------+-------------------------------------+--------+-----+-----------|\n"
```

```
            serial_no += 1
            grand_total += customer_bill[i][1] * customer_bill[i][2]
            products.append(customer_bill[i][0])
            price.append(customer_bill[i][1])
            quantity.append(customer_bill[i][2])
            total.append(customer_bill[i][3])

        text = text[0:len(text)-74]

    text += f"""+------+-------------------------------------+--------+-----+-----------+
|                                                GRAND TOTAL : {grand_total}{((10 -
len(str(grand_total))) * ' ')}|
+------+-------------------------------------+--------+-----+-----------+"""

    display_in_output_box(text)

    #this remainds the user to save the bill before leaving to home page
    #there is a function in save_bill command to implement.
    global remind_save_bill
    remind_save_bill = False


#validates whether the entered credentials is valid
def add_product_function_validation():
    global product_name
    global product_price
    global product_quantity

    product_name = entry_product_name.get().lstrip().rstrip().title()
    product_price = entry_product_price.get().lstrip().rstrip()
    product_quantity = entry_product_quantity.get().lstrip().rstrip()

    global grand
```

```
    grand = True


    text = ""

    if len(str(serial_no)) > 4 or len(product_name) > 35 or product_price.isnumeric() == False
or len(str(product_price)) > 6 or product_quantity.isnumeric() == False or
len(str(product_quantity)) > 3 or (int(grand_total) + (int(product_price) *
(int(product_quantity)))) > 9999999999:

        if len(str(serial_no)) > 4:

            text += "# Only 9999 products can be stored in a bill.\n"


        if len(product_name) > 35:

            text += f"# Reduce the name of product by {len(product_name) - 35} characters.\n"


        if product_price.isnumeric() == False:

            text += "# Price cannot take alphabets of any special characters.\n"

            grand = False

        else:

            product_price = int(product_price)


        if len(str(product_price)) > 6:

            text += "# Price can take value upto 9,99,999.\n"


        if product_quantity.isnumeric() == False:

            text += "# Price cannot take alphabets of any special characters.\n"

            grand = False

        else:

            product_quantity = int(product_quantity)


        if len(str(product_quantity)) > 3:

            text += "# Price can take value upto 999.\n"


        if grand_total + (product_price * product_quantity) > 9999999999:

            text += "# Grand total cannot take value greater than 999,99,99,999. Click save
bill and make a new bill.\n"
```

```
        #informing the user about the error
        messagebox.showerror("ERROR", text)
    else:
        add_product_function()#checking whether the given bill no is there in database

def add_product_function():
    global serial_no, grand_total, product_quantity, product_price

    #type casting the required variables
    product_quantity = int(product_quantity)
    product_price = int(product_price)

    #adding all the necessary details to the list
    products.append(product_name)
    price.append(product_price)
    quantity.append(product_quantity)
    total_price = product_price * product_quantity
    total.append(total_price)

    serial_no += 1
    grand_total += total_price

    #making text for displaying output
    text = ""

    if serial_no > 1:

        text += "\n|------+-------------------------------------+--------+-----+-----------|"

    text += f"\n| {serial_no}.{((3 - len(str(serial_no))) * ' ')} "
    text += f"| {product_name}{((35 - len(str(product_name))) * ' ')} "
    text += f"| {product_price}{((6 - len(str(product_price))) * ' ')} "
```

```
    text += f"| {product_quantity}{((3 - len(str(product_quantity))) * ' ')} "
    text += f"| {total_price}{((9 - len(str(total_price))) * ' ')} |"

    #clearing the required lines to accommodate for the added item
    if serial_no > 2:

        clear_a_line_in_text_box(serial_no * 2 + 1)
        clear_a_line_in_text_box(serial_no * 2 + 1)
        clear_a_line_in_text_box(serial_no * 2 + 1)
    else:
        clear_a_line_in_text_box(serial_no + 3)
        clear_a_line_in_text_box(serial_no + 3)
        clear_a_line_in_text_box(serial_no + 3)

    text = f"""\n+------+-------------------------------------+--------+-----+-----------+
|                                                GRAND TOTAL : {grand_total}{((10 -
len(str(grand_total))) * ' ')}|
+------+-------------------------------------+--------+-----+-----------+"""

    display_in_output_box(text)

    #changing the value to true to remind the user to save bill
    global remind_save_bill
    remind_save_bill = True

    entry_product_name.delete(0, END)
    entry_product_price.delete(0, END)
    entry_product_quantity.delete(0, END)


def delete_an_item(s_number):
    global serial_no
    global grand_total
```

```
try:
    #making the entered number positive even if the entered number is negative
    s_number = int(s_number).__abs__()
    if s_number <= serial_no:

        #clearing the line according to the serial no entered by the user
        if s_number < 2 :
            clear_entire_text_box_from_start_line(3 + s_number)
        else:
            clear_entire_text_box_from_start_line((s_number * 2) + 2)

        #deleting the items from the lists
        del products[s_number - 1]
        del price[s_number - 1]
        del quantity[s_number - 1]
        grand_total -= total[s_number - 1]
        del total[s_number - 1]
        serial_no -= 1

        #preparing new test for output
        text = "\n"
        try:
            for i in range(s_number - 1, len(products)):
                text += f"| {i + 1}.{((3 - len(str(i + 1))) * ' ')} "
                text += f"| {products[i]}{((35 - len(str(products[i]))) * ' ')} "
                text += f"| {price[i]}{((6 - len(str(price[i]))) * ' ')} "
                text += f"| {quantity[i]}{((3 - len(str(quantity[i]))) * ' ')} "
                text += f"| {total[i]}{((9 - len(str(total[i]))) * ' ')} "
                text +=
"|\n|------+-------------------------------------+--------+-----+-----------|\n"

        except:
```

```
                pass
            text += f"""|                                                GRAND TOTAL :
{grand_total}{((10 - len(str(grand_total))) * ' ')}|
+------+-------------------------------------+--------+-----+-----------+"""

            display_in_output_box(text)

            global remind_save_bill
            remind_save_bill = True
        else:
            messagebox.showerror("ERROR", "ENTERED NUMBER EXCEEDS SERIAL NUMBER")
            delete_product_entry.delete(0, END)

    except:
        messagebox.showerror("ERROR", "CANNOT FIND THE SERIAL NUMBER TO DELETE")
        delete_product_entry.delete(0, END)
    delete_product_entry.delete(0, END)


#validate for back button
def back_button_validation():

    global remind_save_bill

    if remind_save_bill:
        messagebox.showwarning("WARNING", "BILL IS NOT SAVED IN THE DATABASE!\nCLICK 'SAVE
BILL' TO SAVE")
    else:
        main_page()


def save_bill_database():
```

```
    global database_bill_no

    if bill_number not in database_bill_no:

        create_a_new_bill(bill_number, customer_name, phone_number, grand_total, username)
        database_bill_no = get_bill_no()
    else:
        delete_records(bill_number)

    save_bill(bill_number, products, price, quantity, total, grand_total)

    global remind_save_bill
    remind_save_bill = False



def modify_bill_page():

    #clearing left frame for new widgets
    clear_left_frame()
    modify_input_label=Label(left_side_frame, text= "Enter the bill number to
modify",background=black, foreground= white)
    modify_input_label.grid(row=0,column=1, padx = 5,pady = 10, sticky=EW)

    global modify_input_bill_no
    modify_input_bill_no=Entry(left_side_frame)
    modify_input_bill_no.grid(row=1,column=1, padx = 5,pady = 10, sticky=EW)

    modify_button=Button(left_side_frame,text = "MODIFY BILL",background = orange ,foreground =
black, command = lambda : modify_bill_validation())
    modify_button.grid(row=2,column=1, padx = 5,pady = 10, sticky=EW)

    exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground = black,
command = lambda : exit())
    exit_button.grid(row = 3, column = 0, padx = 5, pady = 15, sticky = W)
```

```
    back_button = Button(left_side_frame, text = "BACK", background = purple, foreground =
black, command = lambda : main_page())
    back_button.grid(row =3, column = 2, padx = 5, pady = 15, sticky = E)



#validation for modify page
def modify_bill_validation():

    modify_bill_number = modify_input_bill_no.get()

    if modify_bill_number.isnumeric():

        modify_bill_number = int(modify_bill_number)
        #checking whether the given bill no is there in database
        if modify_bill_number in database_bill_no:

            global bill_number
            bill_number = modify_bill_number
            add_modify_operation_page("modify")
        else:
            messagebox.showerror("BILL DOESN'T EXIST", "Bill no. doesn't exist in the
database")
            modify_input_bill_no.delete(0, END)
    else:
        messagebox.showerror("BILL DOESN'T EXIST", "Bill no. doesn't exist in the database")
        modify_input_bill_no.delete(0, END)



def delete_bill_page():
    #clearing left frame for new widgets
    clear_left_frame()
```

```
    del_input_label=Label(left_side_frame, text= "Enter the bill number to
delete",background=black, foreground= white)

    del_input_label.grid(row=0,column=1, padx = 5,pady = 10, sticky=EW)


    global del_input_bill_no

    del_input_bill_no=Entry(left_side_frame)

    del_input_bill_no.grid(row=1,column=1, padx = 5,pady = 10, sticky=EW)


    delete_button=Button(left_side_frame,text = "DELETE BILL",background = cyan ,foreground =
black, command = lambda : delete_bill_validation())

    delete_button.grid(row=2,column=1, padx = 5,pady = 10, sticky=EW)


    exit_button = Button(left_side_frame, text = "EXIT", background = red , foreground = black,
command = lambda : exit())

    exit_button.grid(row = 3, column = 0, padx = 5, pady = 15, sticky = W)


    back_button = Button(left_side_frame, text = "BACK", background = purple, foreground =
black, command = lambda : main_page())

    back_button.grid(row =3, column = 2, padx = 5, pady = 15, sticky = E)



def delete_bill_validation():

    delete_bill_number = del_input_bill_no.get()


    if delete_bill_number.isnumeric():


        delete_bill_number = int(delete_bill_number)

        #checking whether the given bill no is there in database

        if delete_bill_number in database_bill_no:


            delete_customer_bill(delete_bill_number)

            #clearing the output box for main page

            clear_the_entire_text_box()

            main_page()
```

```
        else:
            messagebox.showerror("BILL DIDN'T EXIST", "Bill no doesn't exist in the database")
            del_input_bill_no.delete(0, END)
    else:
        messagebox.showerror("BILL DIDN'T EXIST", "Bill no doesn't exist in the database")
        del_input_bill_no.delete(0, END)



login_gui()
root.mainloop()
```

2. DATABASE:

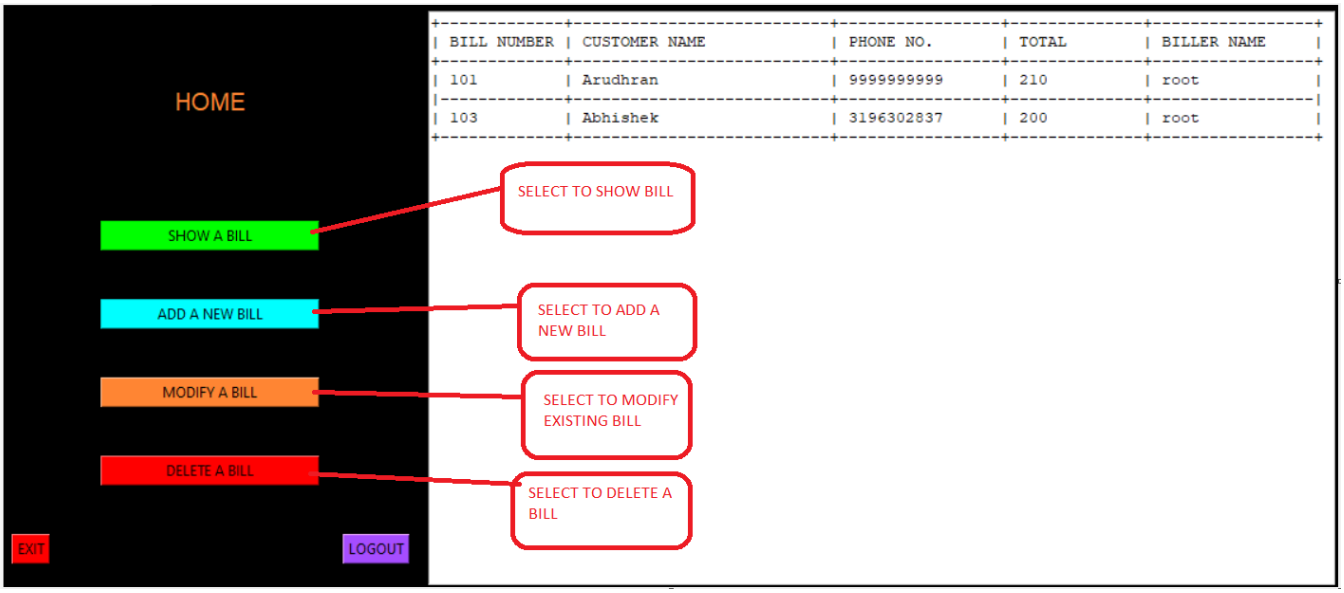
```
import mysql.connector


#authentication for the billing person
def authentication(username, password):
    try:
        global mysql_con
        mysql_con = mysql.connector.connect(host = "localhost", user = username, password = password, db
= "billing_app")

        global mysql_cursor
        mysql_cursor = mysql_con.cursor()
        return True

    except:
        return False

def get_bill_no():
    mysql_cursor.execute("SELECT bill_no FROM main_bill")
    tmp_list = []

    for number in mysql_cursor.fetchall():
        tmp_list.append(number[0])
```

```
    return tmp_list


def get_main_bill():
    mysql_cursor.execute("SELECT * FROM main_bill")
    return mysql_cursor.fetchall()


def create_a_new_bill(bill_no, customer, phone, total, biller):
    mysql_cursor.execute(f"CREATE TABLE bill_{bill_no}(item_name varchar(35), price int, quantity int,
total int)")
    mysql_con.commit()


    mysql_cursor.execute(f"INSERT INTO main_bill VALUES({bill_no}, '{customer}', '{phone}', '{total}',
'{biller}')")
    mysql_con.commit()


def get_customer_bill(bill_no):
    mysql_cursor.execute(f"SELECT * FROM bill_{bill_no}")
    return mysql_cursor.fetchall()


def delete_customer_bill(bill_no):
    mysql_cursor.execute(f"DROP TABLE bill_{bill_no}")
    mysql_cursor.execute(f"DELETE FROM main_bill where bill_no = {bill_no}")
    mysql_con.commit()


def save_bill(bill_no, products, price, quantity, total, grand_total):


    for i in range(len(products)):
        mysql_cursor.execute(f"INSERT INTO bill_{bill_no} VALUES('{products[i]}', {price[i]},
{quantity[i]}, {total[i]})")
        mysql_con.commit()


    mysql_cursor.execute(f"UPDATE main_bill SET total = {grand_total} WHERE bill_no = {bill_no}")
    mysql_con.commit()


def delete_records(bill_no):
    mysql_cursor.execute(f"DELETE FROM bill_{bill_no}")
    mysql_con.commit()
```

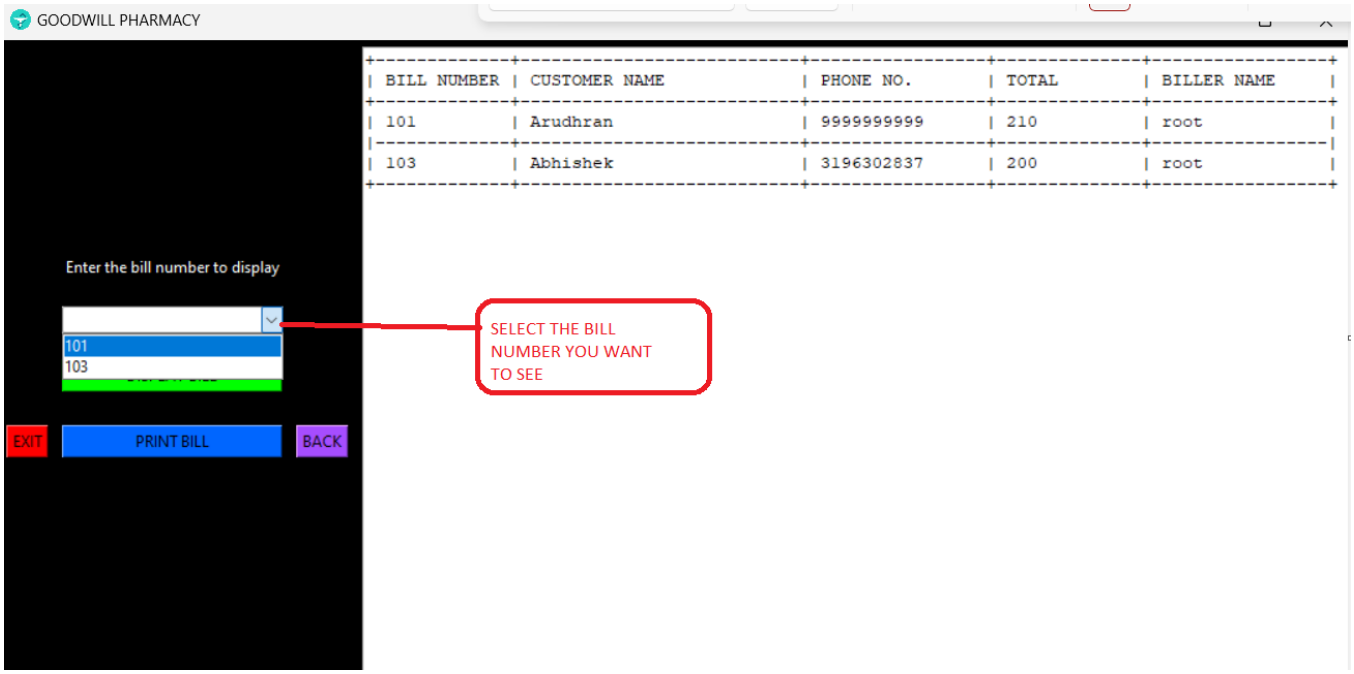
OUTPUT:

i) Enter the Username and Password

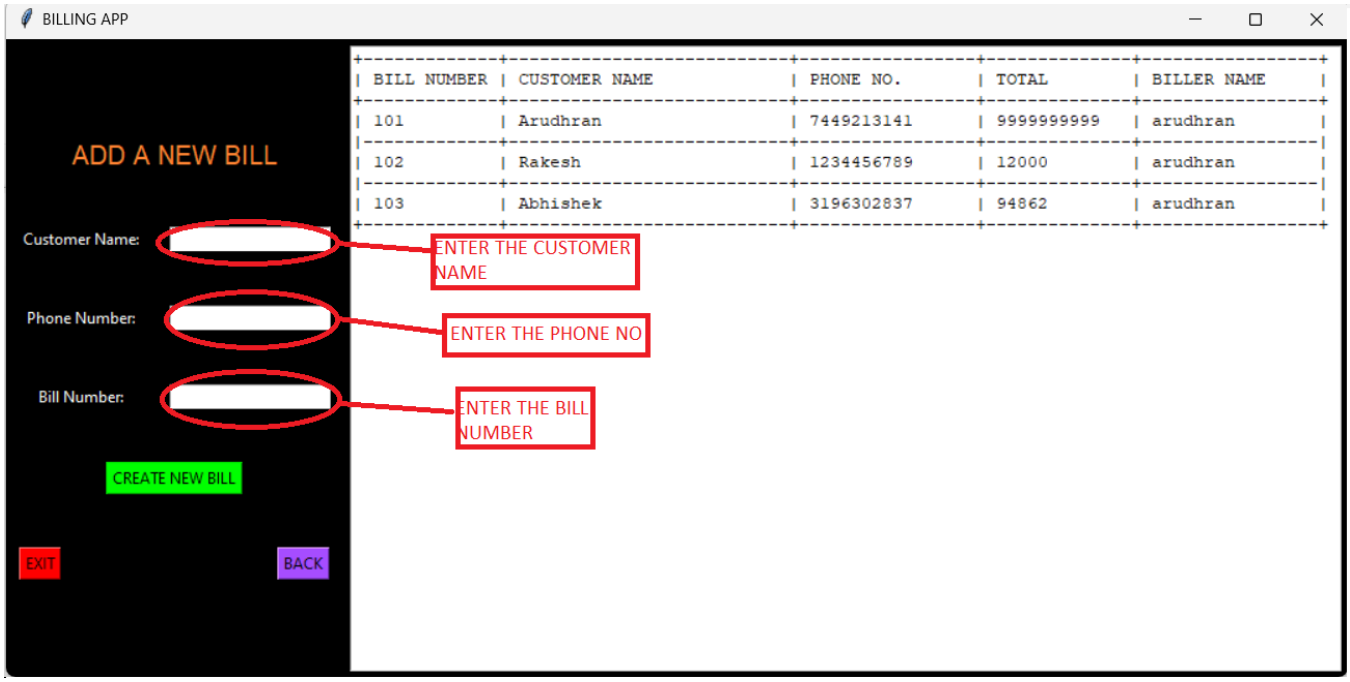
ii) The window here , is used to access the bill, i.e To DELETE, ADD, SHOW & MODIFY



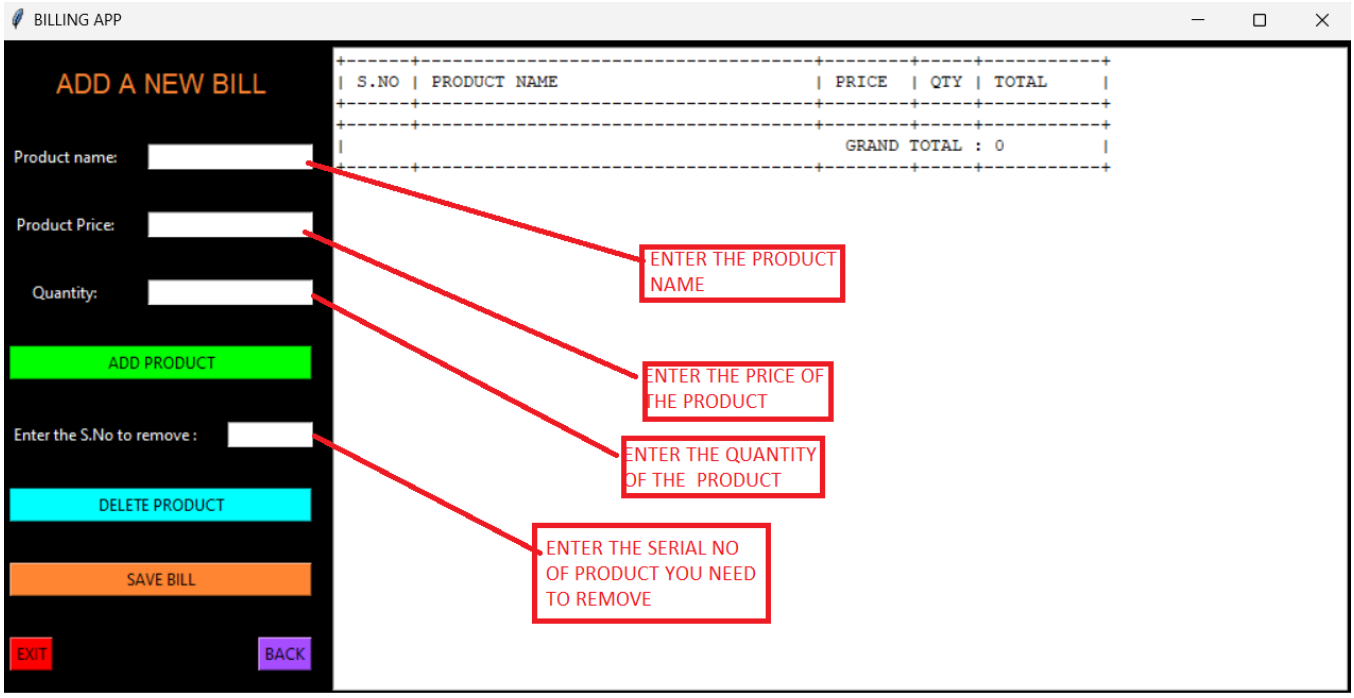
iii) To Show bills:



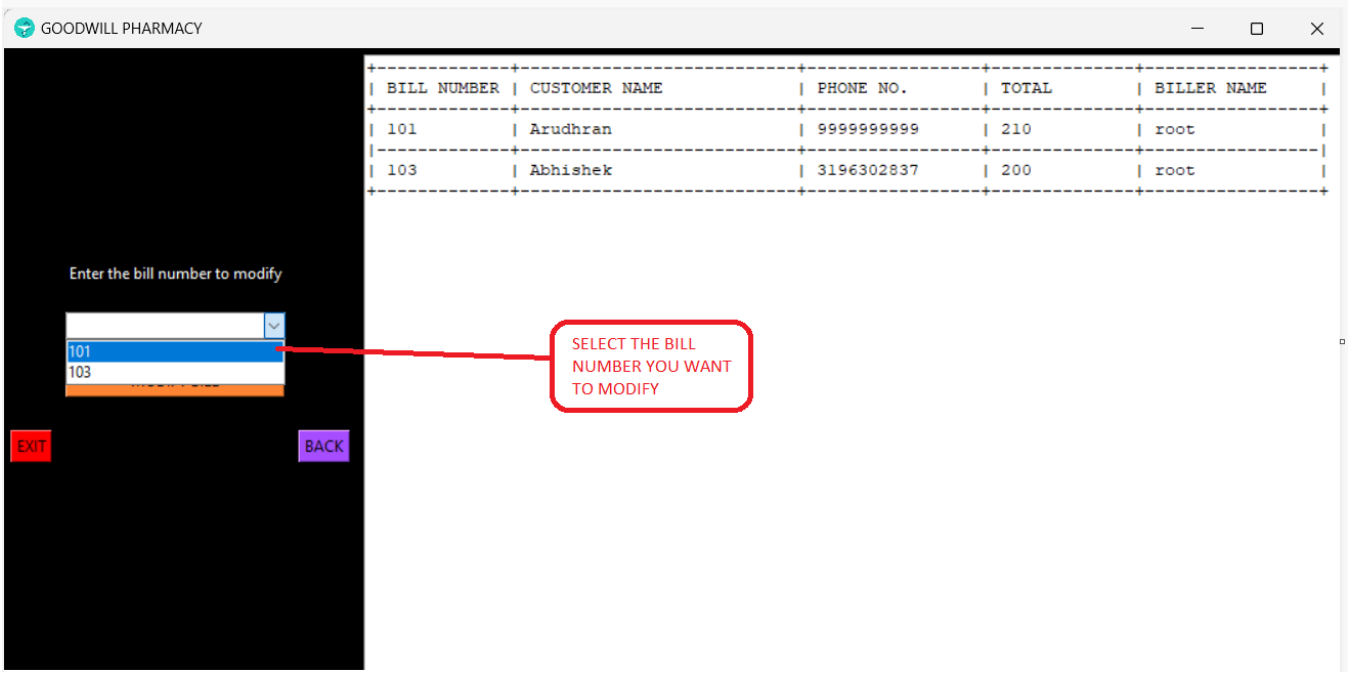
iv) a) To Add a new bill:



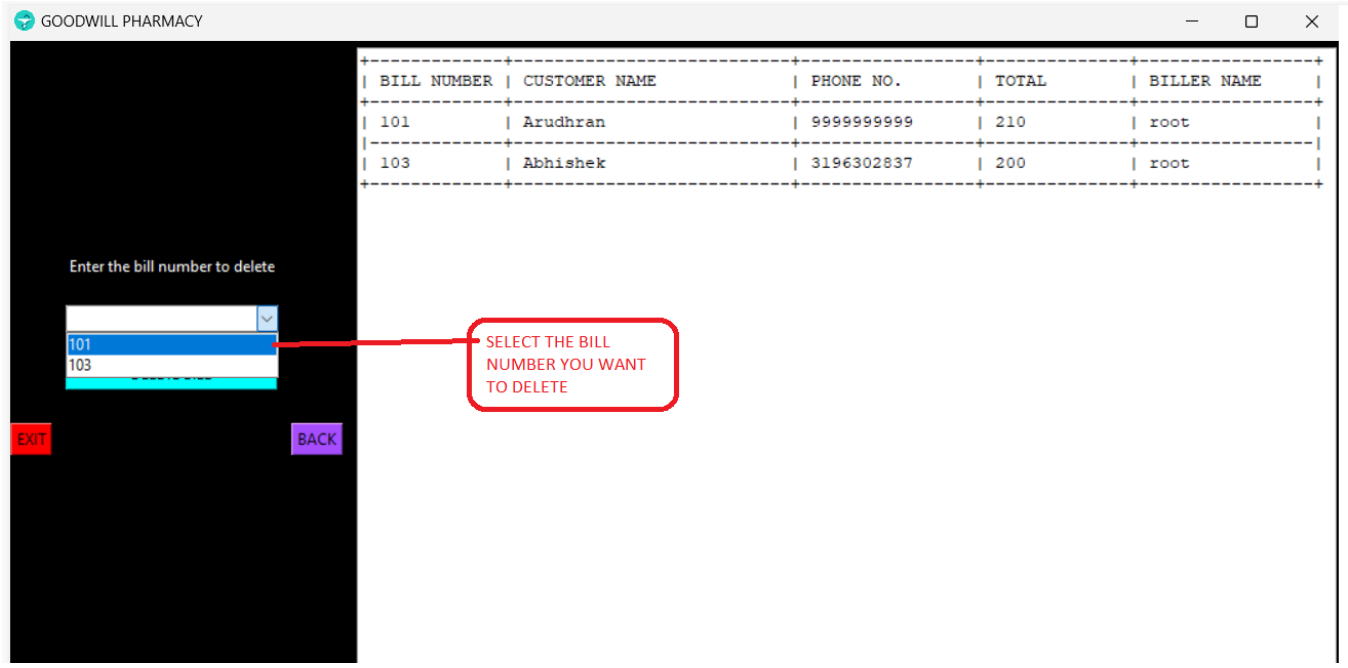
b) Add the product details:



v) To Modify the Bill:



vi) To delete the bill:



DATABASE OUTPUT:

```
mysql> use billing_app;
Database changed
mysql> show tables;
+-----------------------+
| Tables_in_billing_app |
+-----------------------+
| bill_101              |
| bill_102              |
| bill_103              |
| bill_104              |
| main_bill             |
+-----------------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from bill_104;
+-----------+-------+----------+-------+
| item_name | price | quantity | total |
+-----------+-------+----------+-------+
| Shampoo   |     2 |       20 |    40 |
| Soap      |    45 |        2 |    90 |
+-----------+-------+----------+-------+
2 rows in set (0.00 sec)

mysql> select * from bill_103;
+-----------+-------+----------+-------+
| item_name | price | quantity | total |
+-----------+-------+----------+-------+
| Potato    |    20 |        5 |   100 |
| Tomato    |    20 |        5 |   100 |
+-----------+-------+----------+-------+
2 rows in set (0.00 sec)

mysql> select * from bill_102;
+-----------+-------+----------+-------+
| item_name | price | quantity | total |
+-----------+-------+----------+-------+
| Soup      |    20 |        2 |    40 |
| Rice      |    60 |        2 |   120 |
+-----------+-------+----------+-------+
2 rows in set (0.00 sec)

mysql> select * from bill_101;
+-----------+-------+----------+-------+
| item_name | price | quantity | total |
+-----------+-------+----------+-------+
| Maggi     |    85 |        2 |   170 |
| Brush     |    20 |        2 |    40 |
+-----------+-------+----------+-------+
2 rows in set (0.00 sec)
```

BIBLIOGRAPHY:

- John Elder - (codemy.com)
- Geeks For Geeks - (geeksforgeeks.org)
- Stack Overflow - (stackoverflow.com)

CONCLUSION:

We would like to take this opportunity to thank our teachers for their guidance and letting us accomplish this feet. We also learned the values that other members taught us in this process. Thank you!!