INVOICEFY: BILLING SOLUTIONS

CONTENTS

ACKNOWLEDGEMENT

Grace towards the accomplishment of the project I thank **God Almighty** for all his timely guidance and work.

I express my deep sense of gratitude to our beloved Principal **Mrs. Sangeetha Ajith Kumar** for her constant guidance and encouragement throughout the curriculum.

I would like to express my sincere thanks to **Mrs. Jaseela K.I**, Head of the Department, for motivating us to undergo the project work.

I express my sincere gratitude towards our school Al Ameen International Public School, Kochi and to entire teaching and non-teaching faculty for encouraging me throughout my work.

I express my profound gratitude to my Project Guide **Mrs. Jaseela K.I**, P.G.T COMPUTER SCIENCE who have shown keen interest and helped me in completing the project work. It was a great experience to work under such qualified guide who made this project work a great success.

I thank my parents for the motivation and encouragement rendered throughout my project work. Finally, I thank each and every one who has assisted me in doing my project work.

INTRODUCTION

Invoicefy is a user-friendly application designed to streamline the billing process for businesses. Built with Python, using Tkinter Module and MySQL, this app aims to simplify invoicing, manage customer data, and keep track of transactions efficiently. It's perfect for small to medium-sized enterprises that need an intuitive solution for handling their billing needs.

Key Features:

1. User-Friendly Interface:

The app features a clean and straightforward interface that allows users to navigate easily. Users can access various functionalities with minimal training, ensuring a smooth onboarding process.

2. Invoice Generation:

The core functionality of the app is its ability to create invoices quickly and also save and check them at a later time. Users can input products or services sold, quantities, and prices. The app calculates totals, generating a professional invoice ready for printing or emailing.

3. Secure Data Storage:

Customer and transaction data is stored securely, ensuring that sensitive information is protected. The app can also provide backup options to prevent data loss.

4. Technology Stack:

The Simple Billing System is developed using Python, leveraging libraries such as Tkinter for the graphical user interface and MySQL for database management. This combination allows for a lightweight application that can run on various operating systems.

SOFTWARE AND HARDWARE REQUIRMENTS

SOFTWARE REQUIRMENTS

- **Python:**

  Python is a high level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

- **MySQL:**

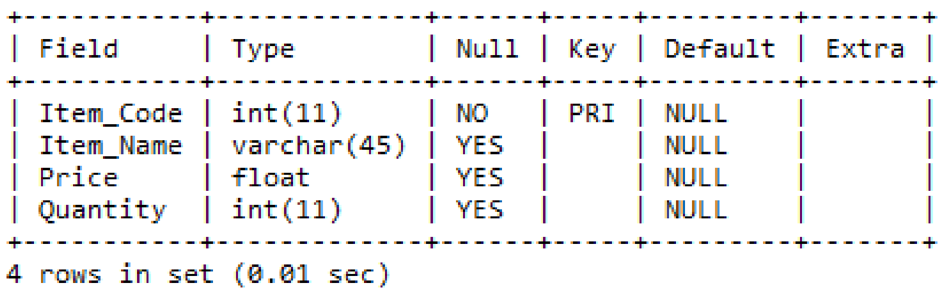
  MySQL is an open-source relational database management system. It may be anything from a simple shopping list to a picture gallery or the vast amount of information in a corporate network. you need a database management system such as MySQL server to add, access, and process data stored in a computer database.

HARDWARE REQUIRMENTS

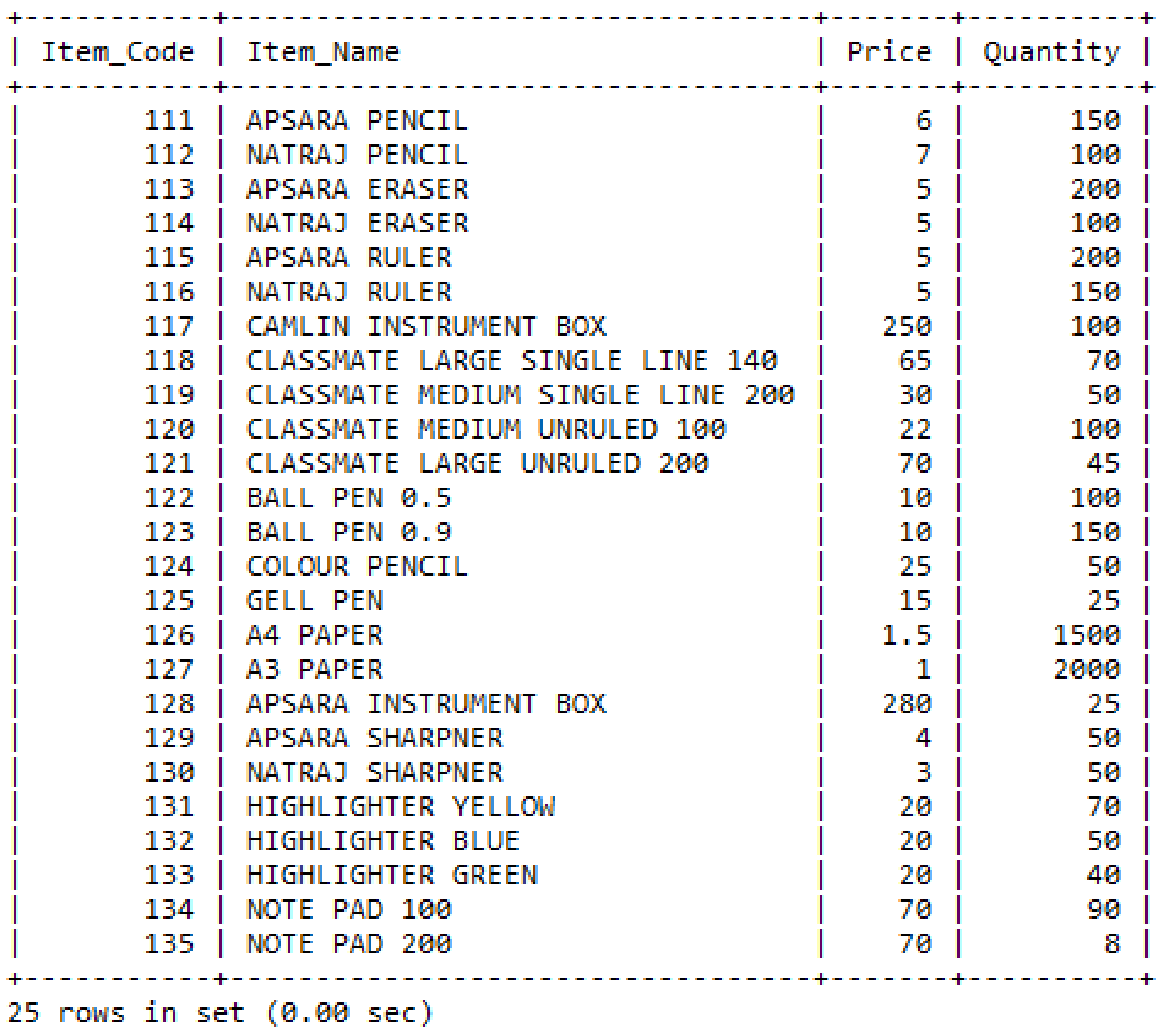
• CPU	- Intel Dual Core
• Hard Disk	- 500 Gigabytes
• RAM	- 4 Gigabytes
• VDA	- Video Display Adaptor

TABLES INCLUDED

Table: Products

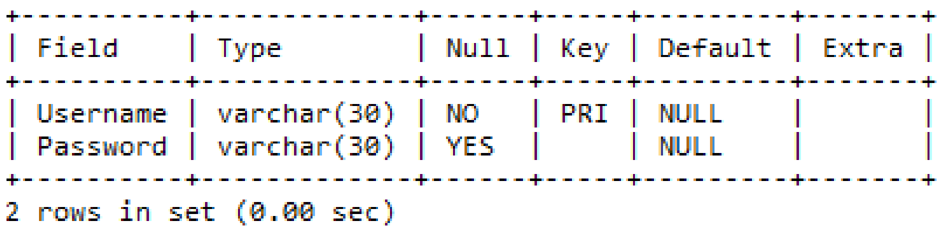


```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Item_Code | int(11)     | NO   | PRI | NULL    |       |
| Item_Name | varchar(45) | YES  |     | NULL    |       |
| Price     | float       | YES  |     | NULL    |       |
| Quantity  | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

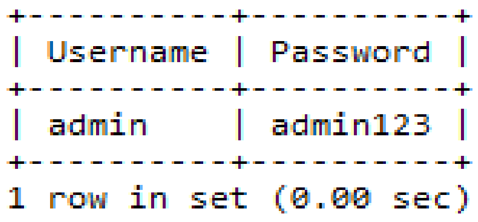


```
+-----------+-----------------------------------+-------+----------+
| Item_Code | Item_Name                         | Price | Quantity |
+-----------+-----------------------------------+-------+----------+
|       111 | APSARA PENCIL                     |     6 |      150 |
|       112 | NATRAJ PENCIL                     |     7 |      100 |
|       113 | APSARA ERASER                     |     5 |      200 |
|       114 | NATRAJ ERASER                     |     5 |      100 |
|       115 | APSARA RULER                      |     5 |      200 |
|       116 | NATRAJ RULER                      |     5 |      150 |
|       117 | CAMLIN INSTRUMENT BOX             |   250 |      100 |
|       118 | CLASSMATE LARGE SINGLE LINE 140   |    65 |       70 |
|       119 | CLASSMATE MEDIUM SINGLE LINE 200  |    30 |       50 |
|       120 | CLASSMATE MEDIUM UNRULED 100      |    22 |      100 |
|       121 | CLASSMATE LARGE UNRULED 200       |    70 |       45 |
|       122 | BALL PEN 0.5                      |    10 |      100 |
|       123 | BALL PEN 0.9                      |    10 |      150 |
|       124 | COLOUR PENCIL                     |    25 |       50 |
|       125 | GELL PEN                          |    15 |       25 |
|       126 | A4 PAPER                          |   1.5 |     1500 |
|       127 | A3 PAPER                          |     1 |     2000 |
|       128 | APSARA INSTRUMENT BOX             |   280 |       25 |
|       129 | APSARA SHARPNER                   |     4 |       50 |
|       130 | NATRAJ SHARPNER                   |     3 |       50 |
|       131 | HIGHLIGHTER YELLOW                |    20 |       70 |
|       132 | HIGHLIGHTER BLUE                  |    20 |       50 |
|       133 | HIGHLIGHTER GREEN                 |    20 |       40 |
|       134 | NOTE PAD 100                      |    70 |       90 |
|       135 | NOTE PAD 200                      |    70 |        8 |
+-----------+-----------------------------------+-------+----------+
25 rows in set (0.00 sec)
```

Table: Admin

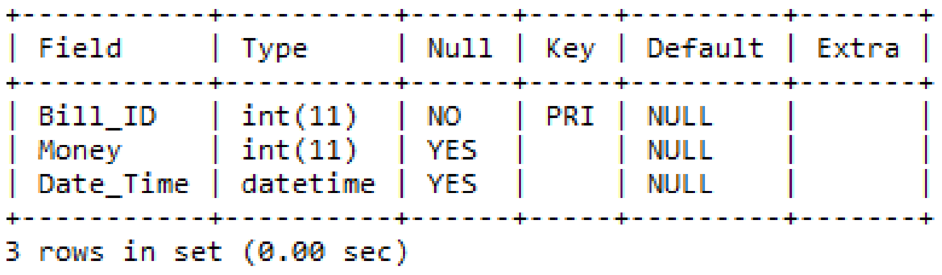


```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Username | varchar(30) | NO   | PRI | NULL    |       |
| Password | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

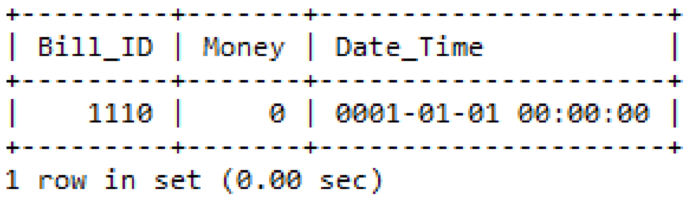


```
+----------+----------+
| Username | Password |
+----------+----------+
| admin    | admin123 |
+----------+----------+
1 row in set (0.00 sec)
```

Table: Bills

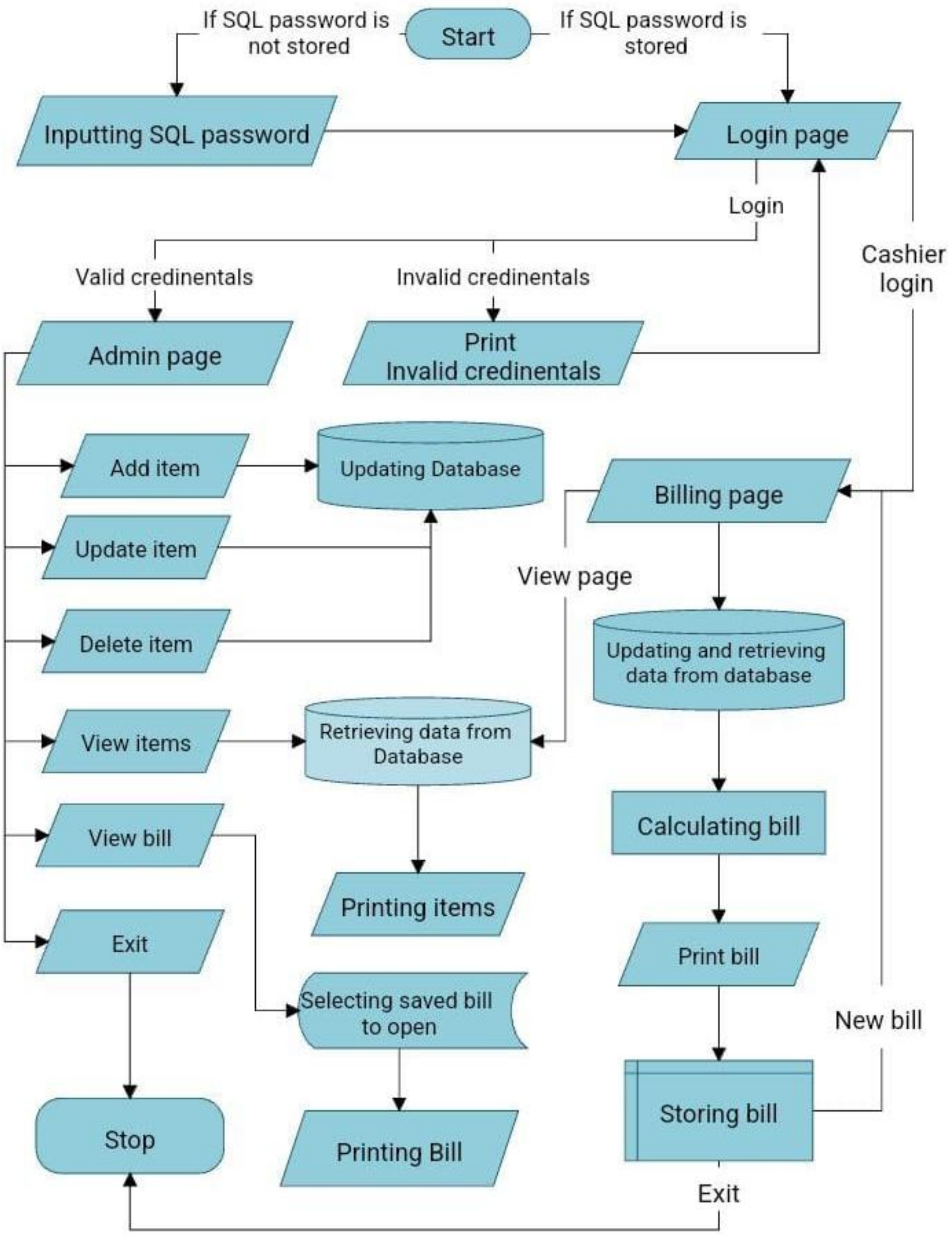


```
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| Bill_ID   | int(11)  | NO   | PRI | NULL    |       |
| Money     | int(11)  | YES  |     | NULL    |       |
| Date_Time | datetime | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```



```
+---------+-------+---------------------+
| Bill_ID | Money | Date_Time           |
+---------+-------+---------------------+
|    1110 |     0 | 0001-01-01 00:00:00 |
+---------+-------+---------------------+
1 row in set (0.00 sec)
```

FLOW CHART



Start
If SQL password is not stored
If SQL password is stored
Inputting SQL password
Login page
Login
Valid credinentals
Invalid credinentals
Cashier login
Admin page
Print Invalid credinentals
Add item
Updating Database
Update item
Delete item
Billing page
View page
Updating and retrieving data from database
View items
Retrieving data from Database
View bill
Calculating bill
Printing items
Exit
Print bill
Selecting saved bill to open
New bill
Stop
Printing Bill
Storing bill
Exit

SOURCE CODE

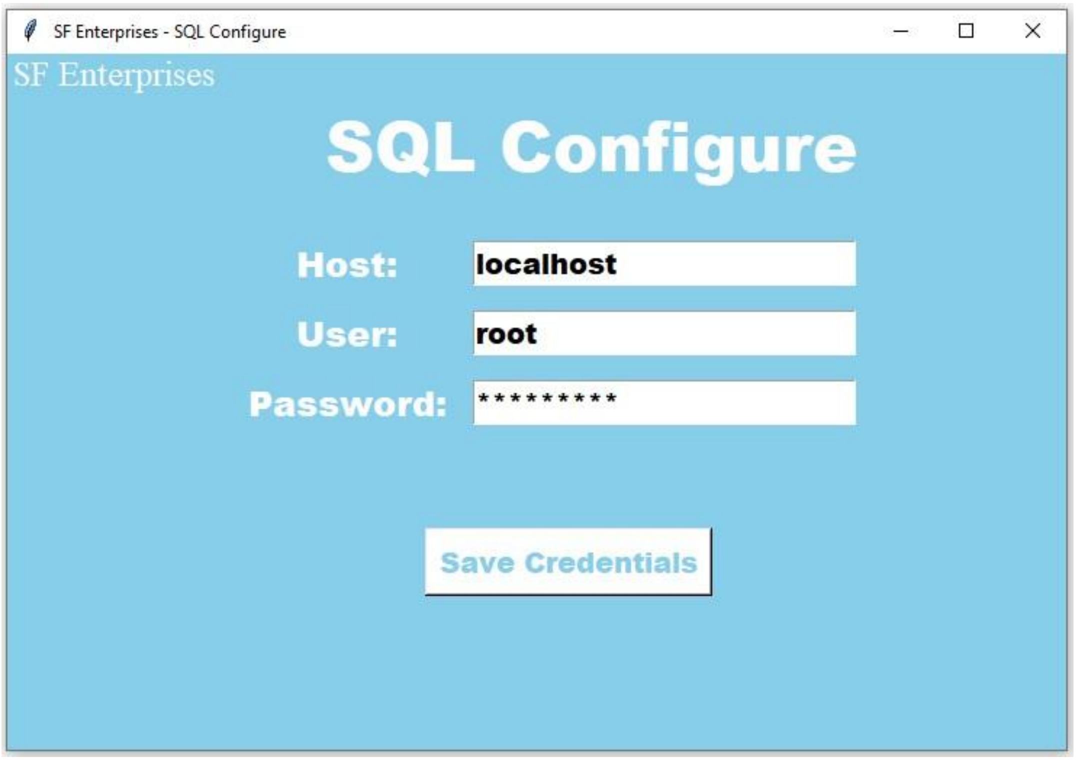
```
#Importing Modules
import mysql.connector as mc
from tkinter import filedialog
import tkinter.font as font
from tkinter import ttk
from tkinter import *
import pickle as pk
import datetime
import os
#Creating Variables
com_name = 'SF Enterprises'
sql_configure_page_open = False
billing_total=0
billing_all_item_names = []
billing_button_value = 'Add'
billing_billed_items = []
last_page = None
deleting_all_item_names = []
updating_all_item_names = []
fonts = [('Arial Black', 35),('Arial Black', 28),('Arial Black', 20),('Arial Black', 19),('Arial Black', 18),('Arial Black', 15),
     ('Arial Black', 14),('Times New Roman', 18)]
colours = ['SkyBlue','LightBlue','White','Red','Green']
##########################################################################################################
#SQL CONFIGURE PAGE
def connection():
    global cur
    global con
    global sql_configure_page_open
    global sql_configure_window
    global sql_configure_label5
    #Creating the Path If It Does't Exist
    path = 'D:/SF Enterprises/Bills'
    if not os.path.exists(path):
        os.makedirs(path)
    try:
        #Checking and Opening SQL File
        with open('D:/SF Enterprises/SQL.dat','rb') as f:
            values = pk.load(f)
        #Establising Connection
        con=mc.connect(host=values[0],user=values[1],password=values[2],database='SF_Enterprises',charset='utf8')
        cur=con.cursor()
        #Distroying SQL Configure Page
        if sql_configure_page_open == True:
            sql_configure_window.destroy()
        login_page()
    except:
        if sql_configure_page_open == False:
            sql_configure_page_open = True
            sql_configure_page()
        else:
          sql_configure_label5.config(text="Wrong Credentials")
def sql_configure(sql_configure_window):
    global sql_configure_text1
    global sql_configure_text2
    global sql_configure_text3
    global sql_configure_page_open
    #Retrieving Inputs
    input1 = StringVar()
    input2 = StringVar()
    input1 = sql_configure_text1.get('1.0', 'end-1c')
    input2 = sql_configure_text2.get('1.0', 'end-1c')
    input3 = sql_configure_text3.get()
    #Saving To File
    with open('D:/SF Enterprises/SQL.dat','wb') as f:
        pk.dump([input1,input2,input3],f)
    connection()
def sql_configure_page():
    global com_name
    global sql_configure_text1
    global sql_configure_text2
    global sql_configure_text3
    global sql_configure_label5
    global sql_configure_window
    #Window Creation
    sql_configure_window = Tk(className = f' {com_name} - SQL Configure')
    sql_configure_window.geometry('730x480+265+100')
    sql_configure_window.configure(bg=colours[0])
    #Widget Creation
    sql_configure_label1 = Label(sql_configure_window, text='SQL Configure',bg=colours[0],fg=colours[2],font=fonts[0])
    sql_configure_label2 = Label(sql_configure_window, text='Host:',bg=colours[0],fg=colours[2],font=fonts[4])
    sql_configure_label3 = Label(sql_configure_window, text='User:',bg=colours[0],fg=colours[2],font=fonts[4])
    sql_configure_label4 = Label(sql_configure_window, text='Password:',bg=colours[0],fg=colours[2],font=fonts[4])
    sql_configure_label5=Label(sql_configure_window,text="",font=fonts[6],bg=colours[0],fg=colours[3])
    sql_configure_label6 = Label(sql_configure_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    sql_configure_text1  = Text(sql_configure_window, width='20',height='1',font=fonts[6])
    sql_configure_text2  = Text(sql_configure_window, width='20',height='1',font=fonts[6])
    sql_configure_text3  = Entry(sql_configure_window, width='20',show='*',font=fonts[6])
    sql_configure_button1 = Button(sql_configure_window,text='Save Credentials',bg=colours[2],fg=colours[0],
        activebackground=colours[1],activeforeground=colours[2],font=fonts[6],
        command=lambda:   sql_configure(sql_configure_window))
    #Widget Position
    sql_configure_label1.place(relx=0.55, rely=0.13, anchor=CENTER)
    sql_configure_label2.place(relx=0.32, rely=0.3, anchor=CENTER)
    sql_configure_label3.place(relx=0.32, rely=0.4, anchor=CENTER)
    sql_configure_label4.place(relx=0.32, rely=0.5, anchor=CENTER)
    sql_configure_label5.place(relx=0.57, rely=0.6, anchor=CENTER)
    sql_configure_label6.place(relx=0.1, rely=0.03, anchor=CENTER)
    sql_configure_text1.place(relx=0.62, rely=0.3, anchor=CENTER)
    sql_configure_text2.place(relx=0.62, rely=0.4, anchor=CENTER)
    sql_configure_text3.place(relx=0.62, rely=0.5, anchor=CENTER)
    sql_configure_button1.place(relx=0.53, rely=0.73, anchor=CENTER)
    #Inserting Value Into Textboxes
    sql_configure_text1.insert(INSERT,'localhost')
    sql_configure_text2.insert(INSERT,'root')
    sql_configure_window.mainloop()
##########################################################################################
#PAGE - LOGIN
def login_button_press(login_window):
    #Navigating To Billing Page
    login_window.destroy()
    billing_page()
```

```
def login(login_window):
    global login_text1
    global login_text2
    global login_label4
    #Retrieving Values
    input1 = login_text1.get()
    input2 = login_text2.get()
    #Retrieving Usernames and Passwords
    cur.execute("SELECT * FROM ADMINS")
    admins = cur.fetchall()
    #Checking Username and Password
    for i in admins:
        if input1.lower() == i[0] and input2 == i[1]:
            login_window.destroy()
            admin_page()
            break
    else:
            login_label4.config(text="Wrong Credentials")
def login_page():
    global com_name
    global login_text1
    global login_text2
    global login_label4
    #Window Creation
    login_window = Tk(className = f' {com_name} - Login')
    login_window.geometry('730x480+265+100')
    login_window.configure(bg=colours[0])
    #Widget Creation
    login_label1 = Label(login_window, text='LOGIN',bg=colours[0],fg=colours[2],font=fonts[0])
    login_label2 = Label(login_window, text='Username:',bg=colours[0],fg=colours[2],font=fonts[4])
    login_label3 = Label(login_window, text='Password:',bg=colours[0],fg=colours[2],font=fonts[4])
    login_label4=Label(login_window,text="",bg=colours[0],fg=colours[3],font=fonts[6])
    login_label5 = Label(login_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    login_text1  = Entry(login_window, width='20',font=fonts[6])
    login_text2  = Entry(login_window, width='20',show="*",font=fonts[6])
    login_button1 = Button(login_window,text='   Login   ',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[6],command=lambda:   login(login_window))
    login_button2 = Button(login_window,text='Cashier Login',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[6],command=lambda:   login_button_press(login_window))
    #Widget Position
    login_label1.place(relx=0.55, rely=0.13, anchor=CENTER)
    login_label2.place(relx=0.32, rely=0.3, anchor=CENTER)
    login_label3.place(relx=0.32, rely=0.43, anchor=CENTER)
    login_label4.place(relx=0.6, rely=0.52, anchor=CENTER)
    login_label5.place(relx=0.1, rely=0.03, anchor=CENTER)
    login_text1.place(relx=0.62, rely=0.3, anchor=CENTER)
    login_text2.place(relx=0.62, rely=0.43, anchor=CENTER)
    login_button1.place(relx=0.73, rely=0.63, anchor=CENTER)
    login_button2.place(relx=0.46, rely=0.63, anchor=CENTER)
    login_window.mainloop()
##########################################################################################################
#PAGE  - ADMIN
def admin_button_press(admin_window,page):
    global last_page
    #Navigating To Other Pages
    if page == 'add':
        admin_window.destroy()
        adding_page()
    elif page == 'update':
        admin_window.destroy()
        updating_page()
    elif page == 'delete':
        admin_window.destroy()
        deleting_page()
    elif page == 'view':
        last_page = 'admin'
        admin_window.destroy()
        view_page()
    elif page == 'signup':
        admin_window.destroy()
        sign_up_page()
    elif page == 'logout':
        admin_window.destroy()
        login_page()
    elif page =='open':
        bill_opening_page('admin')
def admin_page():
    global com_name
    #Window Creation
    admin_window = Tk(className = f' {com_name} - Admin')
    admin_window.geometry("700x480+300+100")
    admin_window.configure(bg="SkyBlue")
    #Widgets Creation
    admin_label1=Label(admin_window,text="   Admin Page   ",font=fonts[1],bg=colours[0],fg=colours[2])
    admin_label2 = Label(admin_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    admin_button1 = Button(admin_window,text='Add Item',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
        command=lambda:   admin_button_press(admin_window,'add'))
    admin_button2 = Button(admin_window,text='Update Item',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
        command=lambda:   admin_button_press(admin_window,'update'))
    admin_button3 = Button(admin_window,text='Delete Item',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
                command=lambda:   admin_button_press(admin_window,'delete'))
    admin_button4 = Button(admin_window,text='View Item',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
        command=lambda:   admin_button_press(admin_window,'view'))
    admin_button5 = Button(admin_window,text='Sign Up',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
        command=lambda:   admin_button_press(admin_window,'signup'))
    admin_button6 = Button(admin_window,text='Open Bill',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
        command=lambda:   admin_button_press(admin_window,'open'))
    admin_button7 = Button(admin_window,text='Log Out',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,
        command=lambda:   admin_button_press(admin_window,'logout'))
    admin_button8 = Button(admin_window,text='   Exit   ',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[3],height=1,width=11,command=admin_window.destroy)
    #Widget Position
    admin_label1.place(relx=0.5, rely=0.08, anchor=CENTER)
    admin_label2.place(relx=0.11, rely=0.03, anchor=CENTER)
    admin_button1.place(relx=0.34, rely=0.28, anchor=CENTER)
    admin_button2.place(relx=0.66, rely=0.28, anchor=CENTER)
    admin_button3.place(relx=0.34, rely=0.46, anchor=CENTER)
    admin_button4.place(relx=0.66, rely=0.46, anchor=CENTER)
    admin_button5.place(relx=0.34, rely=0.66, anchor=CENTER)
    admin_button6.place(relx=0.66, rely=0.66, anchor=CENTER)
    admin_button7.place(relx=0.34, rely=0.86, anchor=CENTER)
    admin_button8.place(relx=0.66, rely=0.86, anchor=CENTER)
#######################################################################################################
#PAGE - SIGN UP
def sign_up_button_press(sign_up_window):
    #Navigating To Admin Page
    sign_up_window.destroy()
    admin_page()
def sign_up(sign_up_window):
    global sign_up_text1
    global sign_up_text2
    global sign_up_label4
    #Retrieving Values
    input1 = sign_up_text1.get()
    input2 = sign_up_text2.get()
    #Retrieving Usernames and Passwords
    cur.execute("SELECT * FROM admins")
    sign_up_usernames  = []
    for i in cur:
        sign_up_usernames.append(i[0].lower())
    if input1.lower() in sign_up_usernames:
        sign_up_label4.config(text="Username Already Exists!!")
    else:
        #Adding New Username And Password
        cur.execute(f"INSERT INTO admins VALUES('{input1}','{input2}')")
        con.commit()
        sign_up_window.destroy()
        login_page()
def sign_up_page():
    global com_name
    global sign_up_text1
    global sign_up_text2
    global sign_up_label4
    #Window Creation
    sign_up_window = Tk(className=f' {com_name} - Sign Up')
    sign_up_window.geometry('750x480+300+100')
    sign_up_window.configure(bg=colours[0])
    #Widget Creation
    sign_up_label1 = Label(sign_up_window, text='Sign Up',bg=colours[0],fg=colours[2],font=fonts[0])
    sign_up_label2 = Label(sign_up_window, text='Username:',bg=colours[0],fg=colours[2],font=fonts[2])
    sign_up_label3 = Label(sign_up_window, text='Password:',bg=colours[0],fg=colours[2],font=fonts[2])
    sign_up_label4 = Label(sign_up_window, text='',bg=colours[0],fg=colours[3],font=fonts[6])
    sign_up_label5 = Label(sign_up_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    sign_up_text1  = Entry(sign_up_window, width='20',font=fonts[6])
    sign_up_text2  = Entry(sign_up_window, width='20',font=fonts[6],show="*")
    sign_up_button1 = Button(sign_up_window,text='Sign Up',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=lambda:   sign_up(sign_up_window))
    sign_up_button2 = Button(sign_up_window,text='Back',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:   sign_up_button_press(sign_up_window))
    #Widget Position
    sign_up_label1.place(relx=0.5, rely=0.18, anchor=CENTER)
    sign_up_label2.place(relx=0.35, rely=0.33, anchor=CENTER)
    sign_up_label3.place(relx=0.35, rely=0.41, anchor=CENTER)
    sign_up_label4.place(relx=0.6, rely=0.5, anchor=CENTER)
    sign_up_label5.place(relx=0.1, rely=0.03, anchor=CENTER)
    sign_up_text1.place(relx=0.65, rely=0.33, anchor=CENTER)
    sign_up_text2.place(relx=0.65, rely=0.42, anchor=CENTER)
    sign_up_button1.place(relx=0.7, rely=0.6, anchor=CENTER)
    sign_up_button2.place(relx=0.5, rely=0.6, anchor=CENTER)
################################################################################################
#PAGE - ADDING
def adding_button_press(adding_window):
    #Navigating To Admin Page
    adding_window.destroy()
    admin_page()
def adding_item():
    global adding_label5
    #Retrieving Item Details
    cur.execute("SELECT * FROM PRODUCTS")
    code=cur.fetchall()[-1][0]
    try:
        item_name=adding_text1.get()
        item_price=float(adding_text2.get())
        item_quantity=eval(adding_text3.get())
        cur.execute(f'SELECT * FROM PRODUCTS WHERE ITEM_NAME = "{item_name}"')
        rc = cur.fetchall()
        #Checking For Used Usernames
        if rc != []:
            raise ZeroDivisionError
        #Checking For Empty Inputs
        if item_name == '':
            raise AttributeError
    except ValueError:
        adding_label5.config(text="Value Error!")
        adding_text1.delete(0,END)
        adding_text2.delete(0,END)
        adding_text3.delete(0,END)
    except ZeroDivisionError:
        adding_label5.config(text="          Item Already Exists!")
        adding_text1.delete(0,END)
        adding_text2.delete(0,END)
        adding_text3.delete(0,END)
    except AttributeError:
        adding_label5.config(text="          Enter Item Name!")
        adding_text1.delete(0,END)
        adding_text2.delete(0,END)
        adding_text3.delete(0,END)
    else:
        #Inserting The New Item
        code=code+1
        cur.execute("INSERT INTO PRODUCTS VALUES({str(code)},'{item_name}',{str(item_price)},{str(item_quantity)})")
        con.commit()
        adding_text1.delete(0,END)
        adding_text2.delete(0,END)
        adding_text3.delete(0,END)
        adding_label5.config(text="Value Added")
def adding_page():
    global adding_text1
    global adding_text2
    global adding_text3
    global adding_label5
    global com_name
    #Window Creation
    adding_window=Tk(className = f' {com_name} - Adding Items')
    adding_window.geometry("700x400+300+100")
    adding_window.configure(bg="SkyBlue")
    #Widget Creation
    adding_label1=Label(adding_window,text="Adding Items",bg=colours[0],fg=colours[2],font=fonts[1])
    adding_label2=Label(adding_window,text="Item name",bg=colours[0],fg=colours[2],font=fonts[3])
    adding_label3=Label(adding_window,text="Price",bg=colours[0],fg=colours[2],font=fonts[3])
    adding_label4=Label(adding_window,text="Qty",bg=colours[0],fg=colours[2],font=fonts[3])
    adding_label5=Label(adding_window,text="",bg=colours[0],fg=colours[2],font=fonts[3])
    adding_label6 = Label(adding_window, text=com_name,bg=colours[0],fg=colours[2],font=('Times New Roman',18))
    adding_text1=Entry(adding_window,width=18,font=fonts[3])
    adding_text2=Entry(adding_window,width=6,font=fonts[3],justify="right")
    adding_text3=Entry(adding_window,width=6,font=fonts[3],justify="right")
    adding_button1 = Button(adding_window,text='Add',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=adding_item)
    adding_button2 = Button(adding_window,text='Back',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:    adding_button_press(adding_window))
    #Widget Position
    adding_label1.place(relx=0.5, rely=0.125, anchor=CENTER)
    adding_label2.place(relx=0.28, rely=0.3, anchor=CENTER)
    adding_label3.place(relx=0.63, rely=0.3, anchor=CENTER)
    adding_label4.place(relx=0.81, rely=0.3, anchor=CENTER)
    adding_label5.place(relx=0.18, rely=0.56, anchor=CENTER)
    adding_label6.place(relx=0.11, rely=0.03, anchor=CENTER)
    adding_text1.place(relx=0.28, rely=0.4, anchor=CENTER)
    adding_text2.place(relx=0.63, rely=0.4, anchor=CENTER)
    adding_text3.place(relx=0.81, rely=0.4, anchor=CENTER)
    adding_button1.place(relx=0.81, rely=0.56, anchor=CENTER)
    adding_button2.place(relx=0.81, rely=0.81, anchor=CENTER)
####################################################################################################
#PAGE - UPDATING
def updating_button_press(updating_window):
    #Navigating To Admin Page
    updating_window.destroy()
    admin_page()
def updating_combobox_search(event):
    global updating_text1
    global updating_all_item_names
    #Retrieving Input From Combobox
    value = event.widget.get()
    if value == '':
        updating_text1['values'] = updating_all_item_names
    else:
        updating_item_names = []
        #Changing Value Of Dropdowm Box
        for item in updating_all_item_names:
            if value.lower() in item.lower():
                updating_item_names.append(item)
        updating_text1['values'] = updating_item_names
def updating_search():
    global updating_label3
    global status
    global old_item_name
    #Retrieving Input
    old_item_name=updating_text1.get()
    item_name=updating_text1.get()
    #Retrieving Item Details
    cur.execute("SELECT * FROM PRODUCTS WHERE Item_name LIKE " +"'"+ item_name+"'")
    item=cur.fetchall()
    #Item Not Found
    if item==[]:
        status=False
        updating_label3.config(text="Item not found")
        updating_text1.delete(0,END)
        updating_text2.delete(0,END)
        updating_text3.delete(0,END)
    else:
        updating_label3.config(text="")
        pr=item[0][2]
        qt=item[0][3]
        updating_text2.delete(0,END)
        updating_text3.delete(0,END)
        #Changing Entry Text Values
        updating_text2.insert(0,pr)
        updating_text3.insert(0,qt)
        status=True
def updating_item():
    global old_item_name
    #Retrieving Input
    item_name=updating_text1.get()
    try:
        item_price=float(updating_text2.get())
        item_quantity=int(updating_text3.get())
    except ValueError:
        updating_label3.config(text="Value Error")
    else:
        #Updating Item
        cur.execute(f"UPDATE products SET Item_name ='{item_name}' WHERE Item_name LIKE '{ old_item_name}'")
        cur.execute(f"UPDATE products SET Price ={str(item_price)} WHERE Item_name LIKE '{ old_item_name }'")
        cur.execute(f"UPDATE products SET Quantity ='{str(item_quantity)}' WHERE Item_name LIKE '{old_item_name}'")
        con.commit()
        updating_label3.config(text="Update complete")
        updating_text1.delete(0,END)
        updating_text2.delete(0,END)
        updating_text3.delete(0,END)
def updating_page():
    global updating_text1
    global updating_text2
    global updating_text3
    global updating_label3
    global com_name
    global updating_all_item_names
    status = True
    old_item_name=str()
    #Window Creation
    updating_window=Tk(className = f' {com_name} - Updating Items')
    updating_window.geometry("700x485+300+100")
    updating_window.configure(bg="SkyBlue")
    #Adding Values To Combobox
    cur.execute("SELECT * FROM PRODUCTS WHERE ITEM_CODE > 110 ORDER BY ITEM_NAME")
    updating_all_item_names = []
    for i in cur:
        updating_all_item_names.append(i[1])
    updating_window.option_add("*TCombobox*Listbox*Font", fonts[5])
    #Widget Creation
    updating_label1=Label(updating_window,text="Updating Products",bg=colours[0],fg=colours[2],font=fonts[1])
    updating_label2=Label(updating_window,text="Product name",bg=colours[0],fg=colours[2],font=fonts[3])
    updating_label3=Label(updating_window,text="",bg=colours[0],fg=colours[3],font=fonts[3])
    updating_label4=Label(updating_window,text="Price",bg=colours[0],fg=colours[2],font=fonts[3])
    updating_label5=Label(updating_window,text="Qty",bg=colours[0],fg=colours[2],font=fonts[3])
    updating_label6 = Label(updating_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    updating_text1=ttk.Combobox(updating_window,value=updating_all_item_names,width=23,font=fonts[5])
    updating_text2=Entry(updating_window,width=6,font=fonts[3])
    updating_text3=Entry(updating_window,width=6,font=fonts[3])
    updating_button1 = Button(updating_window,text='Search',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=updating_search)
    updating_button2 = Button(updating_window,text='Set',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=updating_item)
    updating_button3 = Button(updating_window,text='Back',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:       updating_button_press(updating_window))
    #Widget Position
    updating_label1.place(relx=0.5, rely=0.125, anchor=CENTER)
    updating_label2.place(relx=0.28, rely=0.25, anchor=CENTER)
    updating_label3.place(relx=0.28, rely=0.45, anchor=CENTER)
    updating_label4.place(relx=0.15, rely=0.55, anchor=CENTER)
    updating_label5.place(relx=0.15, rely=0.75, anchor=CENTER)
    updating_label6.place(relx=0.11, rely=0.03, anchor=CENTER)
    updating_text1.place(relx=0.28, rely=0.35, anchor=CENTER)
    updating_text2.place(relx=0.15, rely=0.65, anchor=CENTER)
    updating_text3.place(relx=0.15, rely=0.85, anchor=CENTER)
    updating_button1.place(relx=0.65, rely=0.35, anchor=CENTER)
    updating_button2.place(relx=0.67, rely=0.85, anchor=CENTER)
    updating_button3.place(relx=0.85, rely=0.85, anchor=CENTER)
    #Widget Key Binds
    updating_text1.bind('<KeyRelease>',updating_combobox_search)
##########################################################################################################
#PAGE - DELETING
def deleting_button_press(deleting_window):
    #Navigating To Admin Page
    deleting_window.destroy()
    admin_page()
def deleting_combobox_search(event):
    global deleting_text1
    global deleting_all_item_names
    #Retrieving Input From Combobox
    value = event.widget.get()
    if value == '':
        deleting_text1['values'] = deleting_all_item_names
    else:
        deleting_item_names = []
        #Changing Value Of Dropdowm Box
        for item in deleting_all_item_names:
            if value.lower() in item.lower():
                deleting_item_names.append(item)
        deleting_text1['values'] = deleting_item_names
def deleting_remove():
    global deleting_text1
    global deleting_label3
    try:
        #Retrieving Input
        item_name=deleting_text1.get()
    except :
        deleting_label3.config(text="Invalid Argument")
    else:
        #Retrieving Item Details
        cur.execute("SELECT * FROM PRODUCTS WHERE Item_name LIKE " +"'"+ item_name+"'")
        item=cur.fetchall()
        #Item Not Found
        if item==[]:
                deleting_label3.config(text="Item not Found!!")
        elif item!=[]:
            #Item Deletion
            cur.execute("DELETE FROM PRODUCTS WHERE Item_name LIKE " +"'"+ item_name+"'")
            con.commit()
            deleting_label3.config(text="Deletion complete")
def deleting_page():
    global deleting_text1
    global deleting_label3
    global com_name
    global deleting_all_item_names
    #Window Creation
    deleting_window=Tk(className = f' {com_name} - Deleting Items')
    deleting_window.geometry("600x300+300+200")
    deleting_window.configure(bg="SkyBlue")
    #Adding Values To Combobox
    cur.execute("SELECT * FROM PRODUCTS WHERE ITEM_CODE > 110 ORDER BY ITEM_NAME")
    deleting_all_item_names = []
    for i in cur:
        deleting_all_item_names.append(i[1])
    deleting_window.option_add("*TCombobox*Listbox*Font", fonts[5])
    #Widget Creation
    deleting_label1=Label(deleting_window,text="Deleting Items",bg=colours[0],fg=colours[2],font=fonts[1])
    deleting_label2=Label(deleting_window,text="Item Name",bg=colours[0],fg=colours[2],font=fonts[3])
    deleting_label3=Label(deleting_window,text="",bg=colours[0],fg=colours[3],font=fonts[3])
    deleting_label4 = Label(deleting_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    deleting_text1=ttk.Combobox(deleting_window,value=deleting_all_item_names,width=23,font=fonts[5])
    deleting_button1 = Button(deleting_window,text='Delete',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=deleting_remove)
    deleting_button2 = Button(deleting_window,text='Back',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:        deleting_button_press(deleting_window))
    #Widget Position
    deleting_label1.place(relx=0.5, rely=0.175, anchor=CENTER)
    deleting_label2.place(relx=0.44, rely=0.39, anchor=CENTER)
    deleting_label3.place(relx=0.45, rely=0.65, anchor=CENTER)
    deleting_label4.place(relx=0.125, rely=0.05, anchor=CENTER)
    deleting_text1.place(relx=0.42, rely=0.5, anchor=CENTER)
    deleting_button1.place(relx=0.85, rely=0.5, anchor=CENTER)
    deleting_button2.place(relx=0.85, rely=0.7, anchor=CENTER)
    #Widget Key Binds
    deleting_text1.bind('<KeyRelease>',deleting_combobox_search)
##########################################################################################
#PAGE - VIEW
def view_search(event=None):
    global view_table
    global view_text1
    #Retrieving and Checking Search Box Value
    value = view_text1.get()
    if value == '':
        cur.execute("SELECT * FROM PRODUCTS ORDER BY ITEM_NAME ASC")
    else:
        cur.execute(f"SELECT * FROM PRODUCTS WHERE ITEM_NAME LIKE '%{value}%' ORDER BY ITEM_NAME ASC")
    items=cur.fetchall()
    #Clearing Treeview Values
    view_table.delete(*view_table.get_children())
    #Treeview Values Insertion
    sl=1
    for i in items:
        view_table.insert("",END,values=(sl,i[0],i[1].upper(),i[2],i[3]))
        sl+=1
def view_button_press(view_window):
    global last_page
    #Navigating To Admin/Billing Page
    view_window.destroy()
    if last_page == 'admin':
        last_page = None
        admin_page()
def view_page():
    global com_name
    global view_text1
    global view_table
    #Window Creation
    view_window=Tk(className = f' {com_name} - Viewing Items')
    view_window.geometry("1020x450+130+150")
    view_window.configure(bg="SkyBlue")
    #Retrieving Item Details
    '''cur.execute("SELECT * FROM PRODUCTS ORDER BY ITEM_NAME ASC")
    item=cur.fetchall()'''
    #Widget Creation
    view_label1=Label(view_window,text="View Items",font=fonts[1],bg=colours[0],fg=colours[2])
    view_label2 = Label(view_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    view_label3 = Label(view_window, text="Search",bg=colours[0],fg=colours[2],font=fonts[2])
    view_button1 = Button(view_window,text='Back',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:    view_button_press(view_window))
    view_text1 = Entry(view_window,width=20,font=fonts[5])
    view_table = ttk.Treeview(view_window,columns=("Sl no","Item Code","Item Name","Price","Quantity"),
        show='headings')
    view_table.heading("#1",text="Sl No")
    view_table.heading("#2",text="Item_code")
    view_table.heading("#3",text="Item Name")
    view_table.heading("#4",text="Price")
    view_table.heading("#5",text="Quantity")
    #Widget Position
    view_label1.place(relx=0.5,rely=0.1, anchor=CENTER)
    view_label2.place(relx=0.075, rely=0.04, anchor=CENTER)
    view_label3.place(relx=0.78, rely=0.16, anchor=CENTER)
    view_button1.place(relx=0.93, rely=0.9, anchor=CENTER)
    view_text1.place(relx=0.86, rely=0.23, anchor=CENTER)
    view_table.place(relx=0.5,rely=0.55,anchor=CENTER)
    #Widget Key Binds
    view_text1.bind('<KeyRelease>',view_search)
    view_search()
```

```
##########################################################################################
#PAGE - BILLING
def billing_button_press(billing_window,page):
    #Navigating To View/Login Page
    if page == 'view':
        view_page()
    elif page == 'login':
        billing_window.destroy()
        login_page()
def billing_combobox_search(event):
    global billing_text1
    global billing_all_item_names
    #Retrieving Input From Combobox
    value = event.widget.get()
    if value == '':
        billing_text1['values'] = billing_all_item_names
    else:
        billing_item_names = []
        #Changing Value Of Dropdowm Box
        for item in billing_all_item_names:
            if value.lower() in item.lower():
                billing_item_names.append(item)
        billing_text1['values'] = billing_item_names
def billing_edit(event):
    global billing_button2
    global billing_text1
    global billing_text2
    global billing_table
    global billing_item
    global billing_selected_item
    global billing_button_value
    try:
        #Selecting and Retrieving Values of Focused Treeview Item
        billing_selected_item = billing_table.focus()
        billing_item = billing_table.item(billing_selected_item, 'values')
        billing_text1.delete(0, END)
        billing_text2.delete(0, END)
        #Changing Entry Text Values
        billing_text1.insert(0, billing_item[1])
        billing_text2.insert(0, billing_item[2])
        billing_button2.config(text='Update')
        billing_button_value = 'Update'
    except:     pass
def billing_removing_from_table():
    global billing_button2
    global billing_text1
    global billing_text2
    global billing_label4
    global billing_table
    global billing_total
    global billing_item
    global billing_button_value
    global billing_billed_items
    try:
        #Selecting and Retrieving Values of Focused Treeview Item
        billing_selected_item = billing_table.focus()
        billing_table.delete(billing_selected_item)
        qty=int(billing_text2.get())
        billing_text1.delete(0, END)
        billing_text2.delete(0, END)
        #Retrieving Item Details
        cur.execute("SELECT * FROM PRODUCTS WHERE Item_name LIKE " +"'"+ billing_item[1] +"'")
        item=cur.fetchall()
        #Updating Item Quantity
        qty = item[0][3]+qty
        cur.execute("UPDATE products SET Quantity = " + str(qty) + " WHERE Item_name LIKE "+"'"+ billing_item[1]+"'")
        #Changing Button Text and Grand Total
        billing_button_value = 'Add'
        billing_button2.config(text='Add')
        billing_total = billing_total-float(billing_item[-1])
        billing_label4.config(text=f"Grand Total: {billing_total}")
        #Checking and Deleting Treeview Item
        for i in range(len(billing_billed_items)):
            if billing_item[1].lower() == billing_billed_items[i][1].lower():
                billing_billed_items.pop(i)
        con.commit()
    except:      pass
def billing_adding_to_table(text):
    global billing_label5
    global billing_label4
    global billing_text1
    global billing_text2
    global billing_table
    global billing_total
    global billing_item
    global billing_selected_item
    global billing_billed_items
    try:
        #Retrieving Inputs
        item_name=billing_text1.get()
        qty=int(billing_text2.get())
    except:
        billing_label5.config(text="Value Error! ")
        billing_text1.delete(0,END)
        billing_text2.delete(0,END)
    else:
        #Retrieving Item Details
        cur.execute("SELECT * FROM PRODUCTS WHERE Item_name LIKE " +"'"+ item_name+"'")
        item=cur.fetchall()
        if text == 'Add':
            old_qty = item[0][3]
        elif text == 'Update':
            old_qty = int(billing_item[2])+item[0][3]
            cur.execute(f"UPDATE products SET Quantity = {str(old_qty)} WHERE Item_name LIKE '{item_name}'")
            billing_total = billing_total-(float(billing_item[-1]))
        if item!=[]:
            #Checking Name and Quantity
            if item[0][1].lower()==item_name.lower() and (item[0][3]-qty) >= 0:
                #Defining Item Details
                item_rate=item[0][2]
                total_rate=item_rate * qty
                billing_total=billing_total+total_rate
                new_qty=old_qty-qty
                #Updating Item
                cur.execute(f"UPDATE products SET Quantity = {str(new_qty)} WHERE Item_name LIKE '{item_name}'")
                con.commit()
                #Changing Widget Values
                billing_label5.config(text="")
                if text == 'Add':
                    billing_table.insert("",END,values=(item[0][0],item[0][1].upper(),qty,item[0][2],total_rate))
                    billing_billed_items.append([item[0][0],item[0][1].upper(),qty,item[0][2],total_rate])
                elif text == 'Update':
                    billing_table.item(billing_selected_item,text="",values=(item[0][0],item[0][1].upper(),qty,item[0][2],
                        total_rate))
                billing_text1.delete(0,END)
                billing_text2.delete(0,END)
                billing_label4.config(text=f"Grand Total: {billing_total}")
            #Checking For Insufficent Quantity
            elif (item[0][3]-qty) <= 0:
                billing_label5.config(text="Insufficent Quantity!")
        else:
            billing_label5.config(text="Item Not Found!")
def billing_page():
    global billing_table
    global billing_text1
    global billing_text2
    global billing_label4
    global billing_label5
    global billing_button1
    global billing_button2
    global com_name
    global billing_button_value
    #Window Creation
    billing_window=Tk(className = f' {com_name} - Billing')
    billing_window.geometry("1020x480+150+120")
    billing_window.configure(bg="SkyBlue")
    #Adding Values To Combobox
    cur.execute("SELECT * FROM PRODUCTS WHERE ITEM_CODE > 110 ORDER BY ITEM_NAME")
    for i in cur:
        billing_all_item_names.append(i[1])
    billing_window.option_add("*TCombobox*Listbox*Font", fonts[5])
    #Widget Creation
    billing_label1=Label(billing_window,text="Billing",bg=colours[0],fg=colours[2],font=fonts[1])
    billing_label2=Label(billing_window,text="Product name",bg=colours[0],fg=colours[2],font=fonts[3])
    billing_label3=Label(billing_window,text="Qty",bg=colours[0],fg=colours[2],font=fonts[3])
    billing_label4=Label(billing_window,text="Grand Total: ",bg=colours[0],fg=colours[4],font=fonts[5])
    billing_label5=Label(billing_window,text="",bg=colours[0],fg=colours[2],font=fonts[5])
    billing_label6=Label(billing_window,text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
    billing_text1=ttk.Combobox(billing_window,value=billing_all_item_names,width=23,font=fonts[5])
    billing_text2=Entry(billing_window,width=6,font=fonts[5],justify="right")
    billing_button1 = Button(billing_window,text='Remove',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=billing_removing_from_table)
    billing_button2 = Button(billing_window,text='Add',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:      billing_adding_to_table(billing_button_value))
    billing_button3 = Button(billing_window,text='Bill',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=lambda:      bill_opening_page('bill'))
    billing_button4 = Button(billing_window,text='Items',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:      billing_button_press(billing_window,'view'))
    billing_button5 = Button(billing_window,text='Back',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,
        command=lambda:      billing_button_press(billing_window,'login'))
    billing_button6 = Button(billing_window,text='Exit',bg=colours[2],fg=colours[0],activebackground=colours[1],
        activeforeground=colours[2],font=fonts[5],height=1,width=7,command=billing_window.destroy)
    billing_table = ttk.Treeview(billing_window,columns=("Sl No","Item","Qty","Price","Item Total"),show='headings')
    billing_table.heading("#1",text="Item_code")
    billing_table.heading("#2",text="Item")
    billing_table.heading("#3",text="Qty")
    billing_table.heading("#4",text="Price")
    billing_table.heading("#5",text="Item total")
    #Widget Position
    billing_label1.place(relx=0.5, rely=0.075, anchor=CENTER)
    billing_label2.place(relx=0.24, rely=0.25, anchor=CENTER)
    billing_label3.place(relx=0.55, rely=0.25, anchor=CENTER)
    billing_label4.place(relx=0.81, rely=0.96, anchor=CENTER)
    billing_label5.place(relx=0.15,rely=0.42, anchor=CENTER)
    billing_label6.place(relx=0.072, rely=0.03, anchor=CENTER)
    billing_button1.place(relx=0.70, rely=0.23, anchor=CENTER)
    billing_button2.place(relx=0.70, rely=0.35, anchor=CENTER)
    billing_button3.place(relx=0.82, rely=0.35, anchor=CENTER)
    billing_button4.place(relx=0.82, rely=0.23, anchor=CENTER)
    billing_button5.place(relx=0.94, rely=0.35, anchor=CENTER)
    billing_button6.place(relx=0.94, rely=0.23, anchor=CENTER)
    billing_text1.place(relx=0.24, rely=0.35, anchor=CENTER)
    billing_text2.place(relx=0.55, rely=0.35, anchor=CENTER)
    billing_table.place(relx=0.01,rely=0.45)
    #Widget Key Binds
    billing_text1.bind('<KeyRelease>',billing_combobox_search)
    billing_table.bind('<Button-1>',billing_edit)
###############################################################################################
#PAGE - BILL OPENING
def bill_opening_bill():
    global billing_total
    global billing_billed_items
    #Retrieving Bill ID
    cur.execute("SELECT bill_id FROM Bills")
    bill_id=int(cur.fetchall()[-1][0])
    bill_id+=1
    #Getting Time and Date
    time_full=datetime.datetime.now()
    time_full=str(time_full).split()
    current_date_and_time = time_full[0]+' '+time_full[1][0:8]
    #Inserting Value Into Table
    cur.execute(f"INSERT INTO BILLS VALUES({bill_id},{billing_total},'{current_date_and_time}')")
    con.commit()
    #Saving to File
    with open('D:/SF Enterprises/Bills/bill_id - {bill_id}.dat',"wb") as bf:
        ls=[bill_id,billing_total,current_date_and_time]
        pk.dump(ls,bf)
        pk.dump(billing_billed_items,bf)
def bill_opening_page(page):
    global com_name
    global billing_billed_items
    global billing_total
    if page ==  'admin':
        #Opening Bill
        filename = filedialog.askopenfilename(initialdir = "D:/SF Enterprises/Bills",title = "Select a Bill",
            filetypes = (("Binary Files","*.dat*"),("All Files","*.*")))
        with open('D:/SF Enterprises/Bills/bill_id - 1111.dat','rb') as bf:
            ls=pk.load(bf)
            values = pk.load(bf)
    elif page == 'bill':
        values = billing_billed_items
    #Retrieving Bill ID
    cur.execute("SELECT bill_id FROM Bills")
    bill_id=int(cur.fetchall()[-1][0])
    bill_id+=1
    #Getting Time and Date
    time_full=datetime.datetime.now()
    time_full=str(time_full).split()
    current_date_and_time = time_full[0]+' '+time_full[1][0:8]
    ls = [bill_id,billing_total,current_date_and_time]
#Window Creation
bill_opening_window=Tk(className = f' {com_name} - Open Bill')
bill_opening_window.geometry("1020x480+150+120")
bill_opening_window.configure(bg=colours[0])
#Widget Creation
bill_opening_label1=Label(bill_opening_window,text="Bill Viewer",font=fonts[1],bg=colours[0],fg=colours[2])
bill_opening_label2=Label(bill_opening_window,text=f"Bill Id - {ls[0]}",font=fonts[5],bg=colours[0],fg=colours[2])
bill_opening_label3=Label(bill_opening_window,text=f"Date - {ls[2][0:11]}",font=fonts[5],bg=colours[0],fg=colours[2])
bill_opening_label4=Label(bill_opening_window,text=f"Time - {ls[2][11:]}",font=fonts[5],bg=colours[0],fg=colours[2])
bill_opening_label5=Label(bill_opening_window,text=f"Grand Total :  {ls[1]}",font=fonts[5],bg=colours[0],fg=colours[4]
bill_opening_label6 = Label(bill_opening_window, text=com_name,bg=colours[0],fg=colours[2],font=fonts[7])
bill_opening_button1 = Button(bill_opening_window,text='Close',bg=colours[2],fg=colours[0],
    activebackground=colours[1],activeforeground=colours[2],
    font=fonts[5],height=1,width=7,command=bill_opening_window.destroy)
bill_opening_button2 = Button(bill_opening_window,text='Save',bg=colours[2],fg=colours[0],
    activebackground=colours[1],activeforeground=colours[2],
    font=fonts[5],height=1,width=7,command=bill_opening_bill)
bill_opening_table = ttk.Treeview(bill_opening_window,columns=("Sl No","Item","Qty","Price","Item Total"),
    show='headings')
bill_opening_table.heading("#1",text="Item_code")
bill_opening_table.heading("#2",text="Item")
bill_opening_table.heading("#3",text="Qty")
bill_opening_table.heading("#4",text="Price")
bill_opening_table.heading("#5",text="Item total")
#Widget Position
bill_opening_label1.place(relx=0.5, rely=0.075, anchor=CENTER)
bill_opening_label2.place(relx=0.076, rely=0.15, anchor=CENTER)
bill_opening_label3.place(relx=0.105, rely=0.21, anchor=CENTER)
bill_opening_label4.place(relx=0.089, rely=0.27, anchor=CENTER)
bill_opening_label5.place(relx=0.115, rely=0.93, anchor=CENTER)
bill_opening_label6.place(relx=0.072, rely=0.03, anchor=CENTER)
bill_opening_table.place(relx=0.5,rely=0.6,anchor=CENTER)
bill_opening_button1.place(relx=0.94, rely=0.93, anchor=CENTER)
if page == 'bill':
    bill_opening_button2.place(relx=0.83, rely=0.93, anchor=CENTER)
 #Inserting Into Treeview
for item in values:
    bill_opening_table.insert("",END,values=(item[0],item[1],item[2],item[3],item[4]))
```

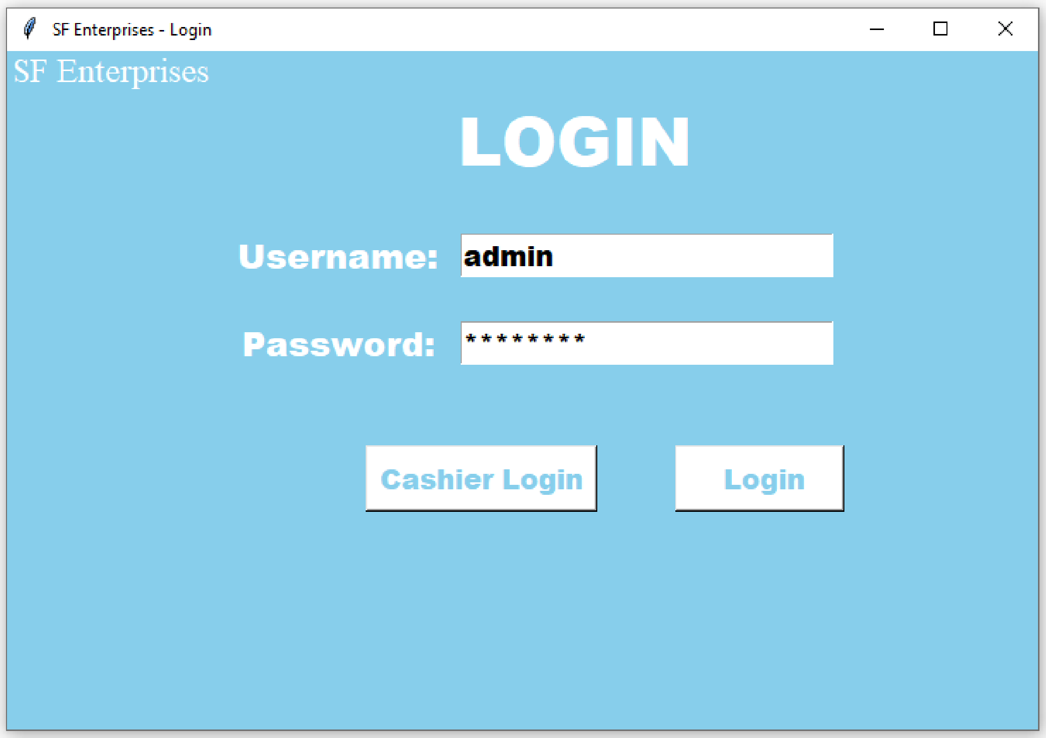
OUTPUT

1.SQL Configure Page



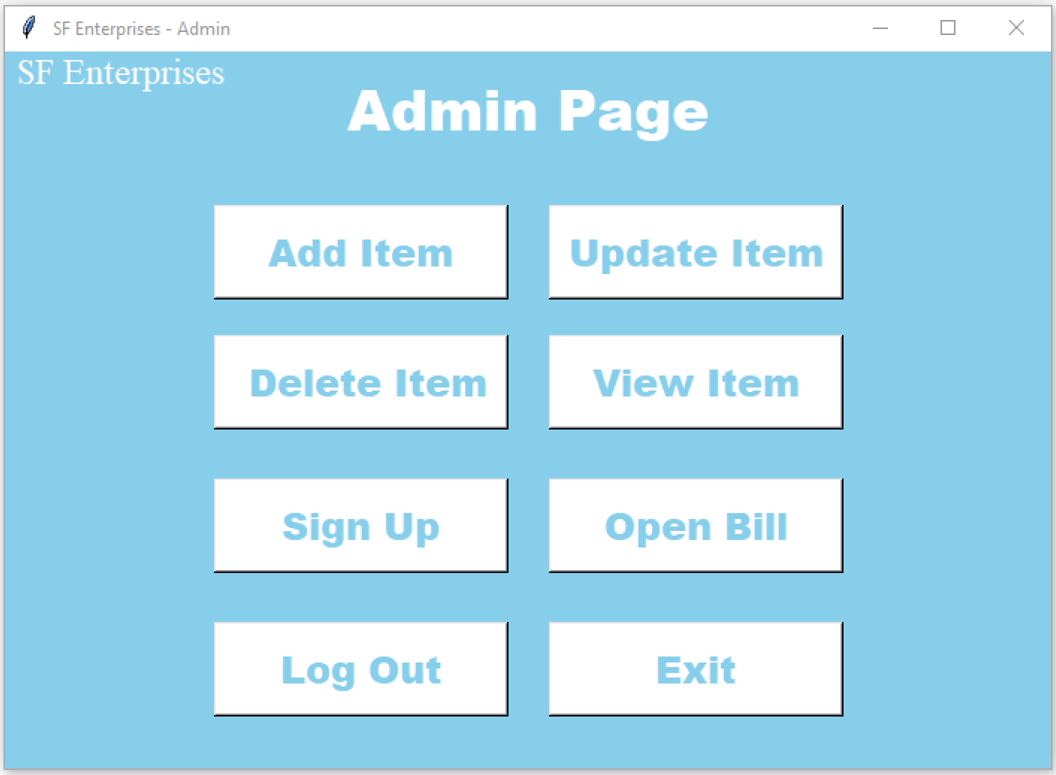
Host: localhost
User: root
Password: *********

2.Login Page

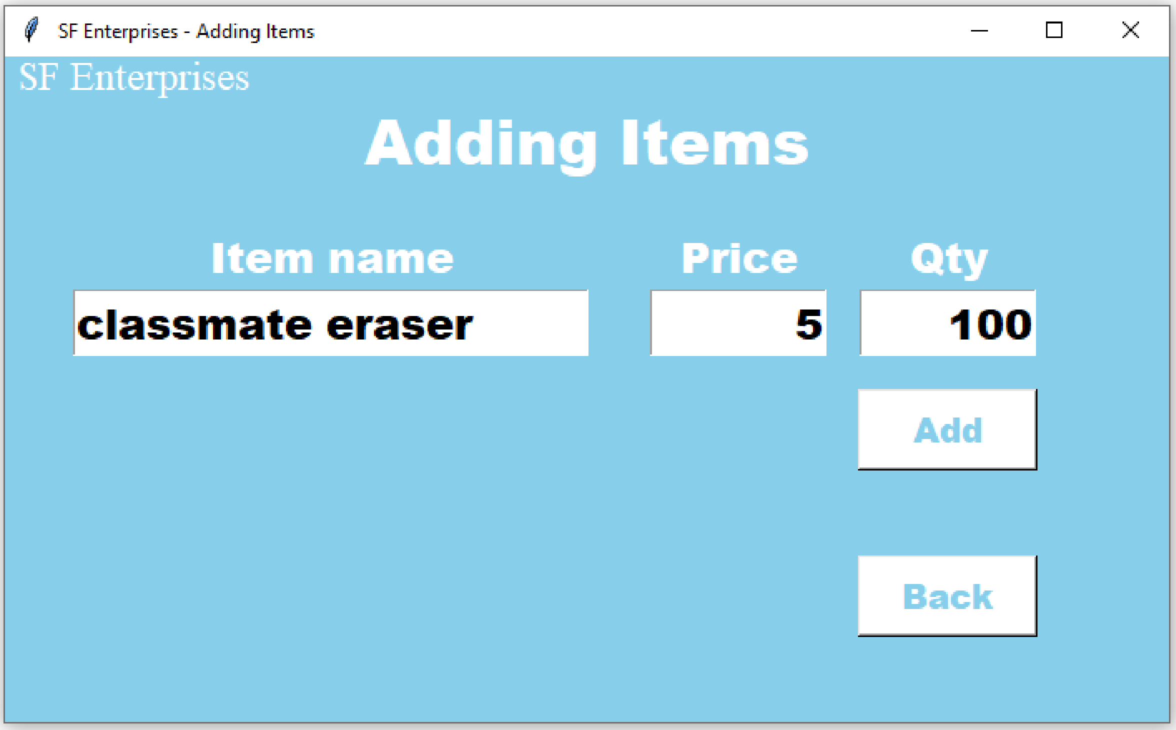


Username: admin
Password: ********

3. Admin page

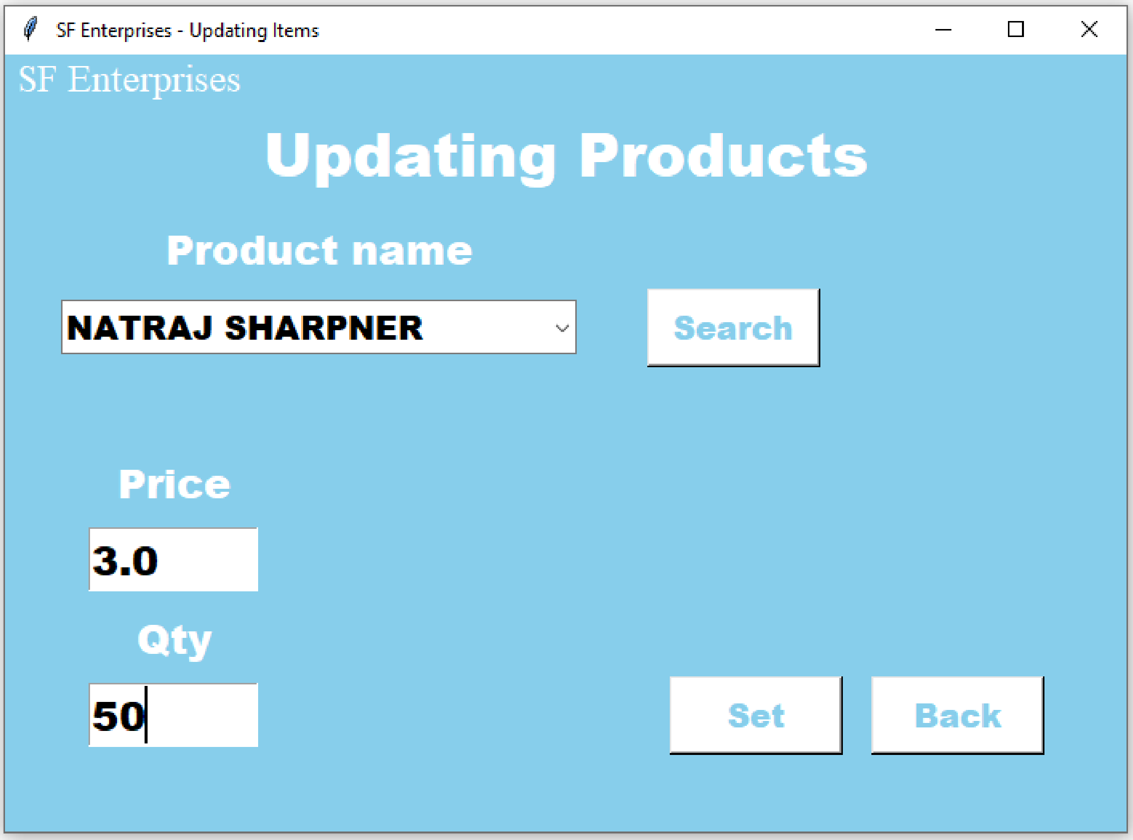


4. Add Item

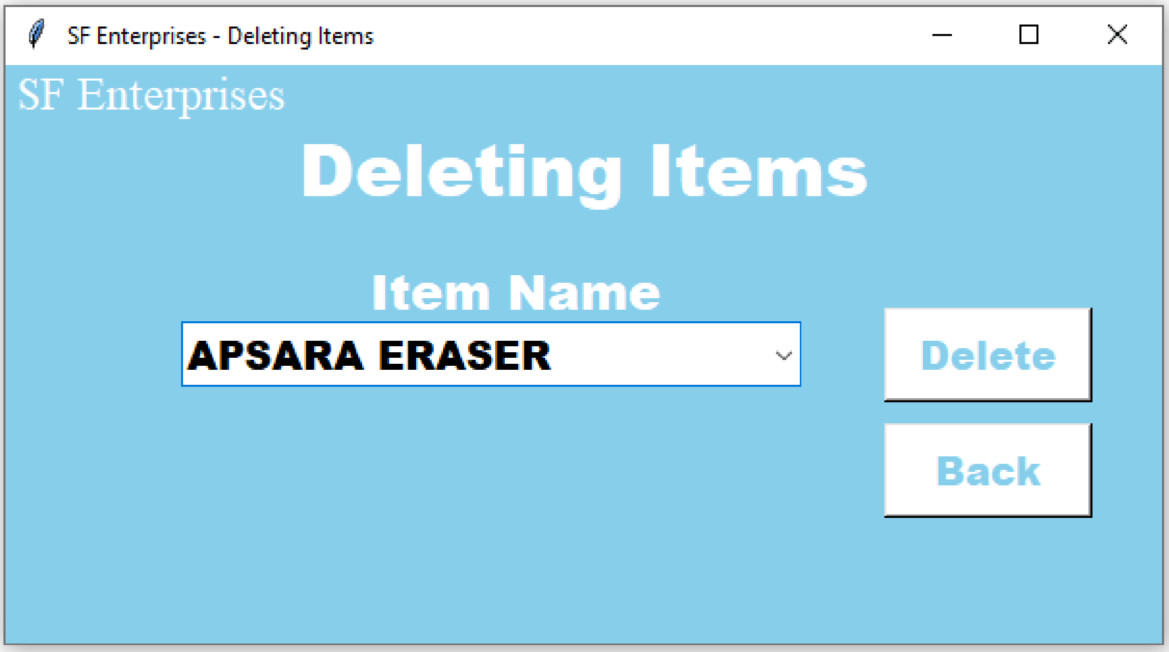


Item name: classmate eraser
Price: 5
Qty: 100

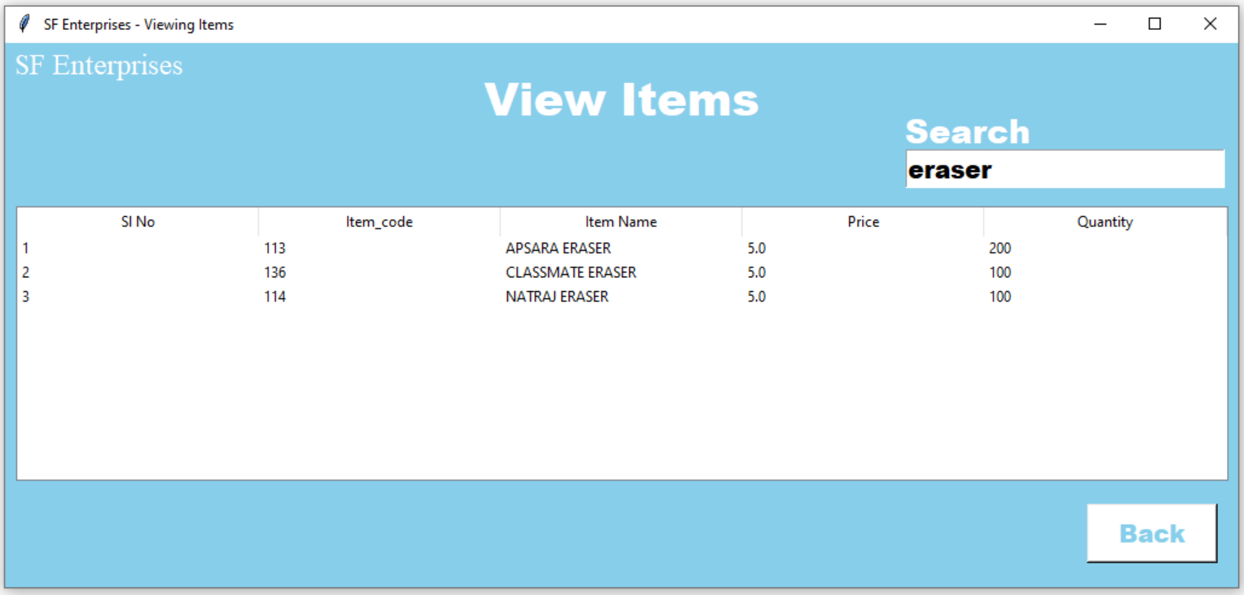
5.Update Item



6.Delete Item



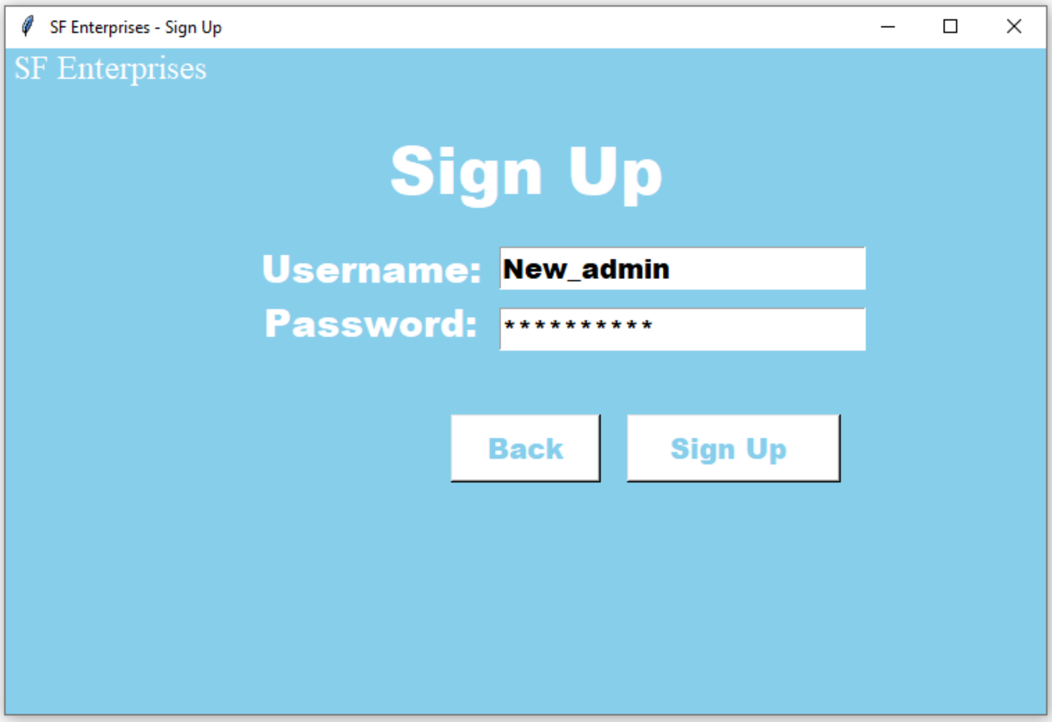
7. View Items



eraser

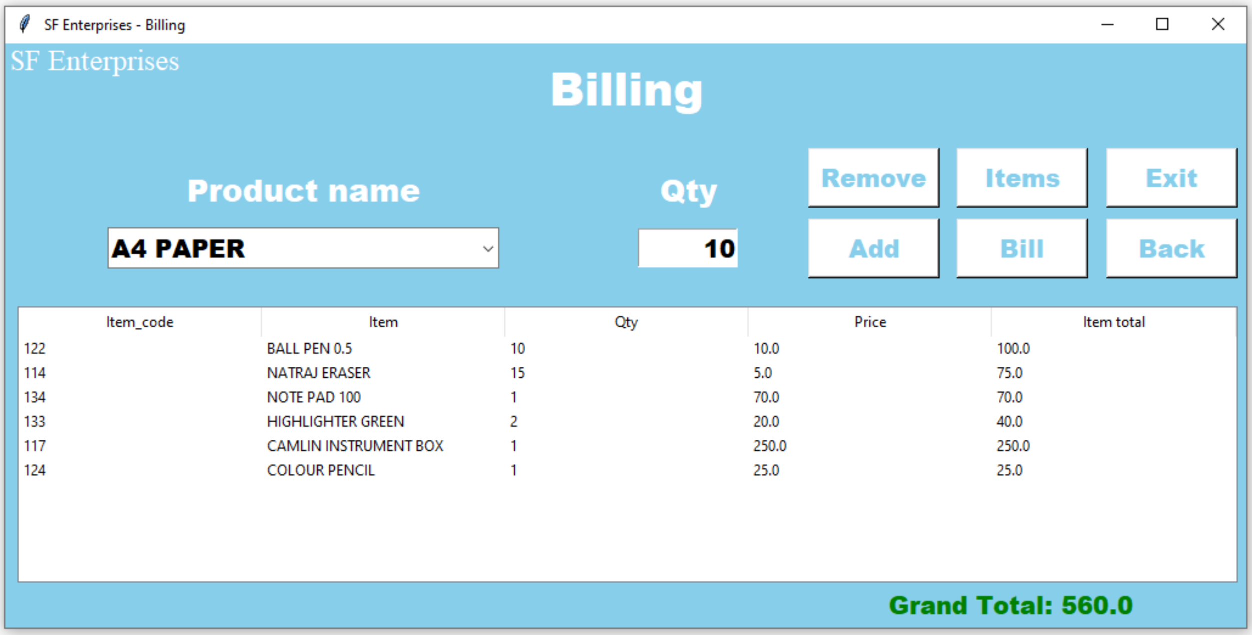
Sl No	Item_code	Item Name	Price	Quantity
1	113	APSARA ERASER	5.0	200
2	136	CLASSMATE ERASER	5.0	100
3	114	NATRAJ ERASER	5.0	100

8. Sign Up



Username: New_admin
Password: **********

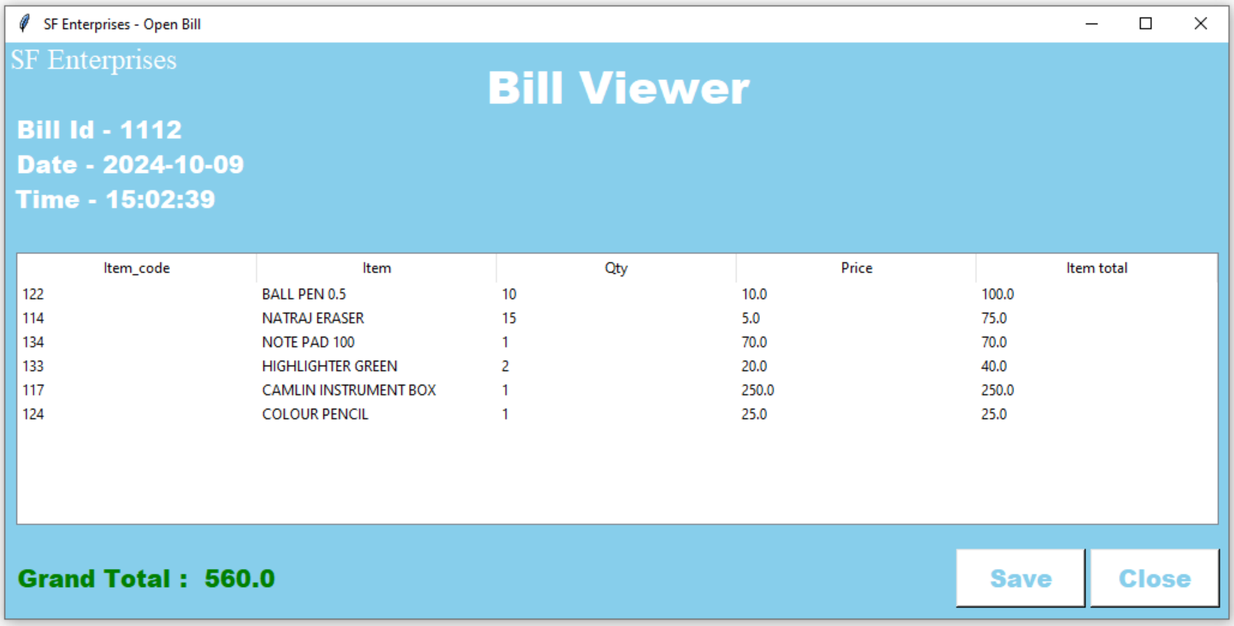
9. Cashier login



Item_code	Item	Qty	Price	Item total
122	BALL PEN 0.5	10	10.0	100.0
114	NATRAJ ERASER	15	5.0	75.0
134	NOTE PAD 100	1	70.0	70.0
133	HIGHLIGHTER GREEN	2	20.0	40.0
117	CAMLIN INSTRUMENT BOX	1	250.0	250.0
124	COLOUR PENCIL	1	25.0	25.0

Grand Total: 560.0

10. Bill

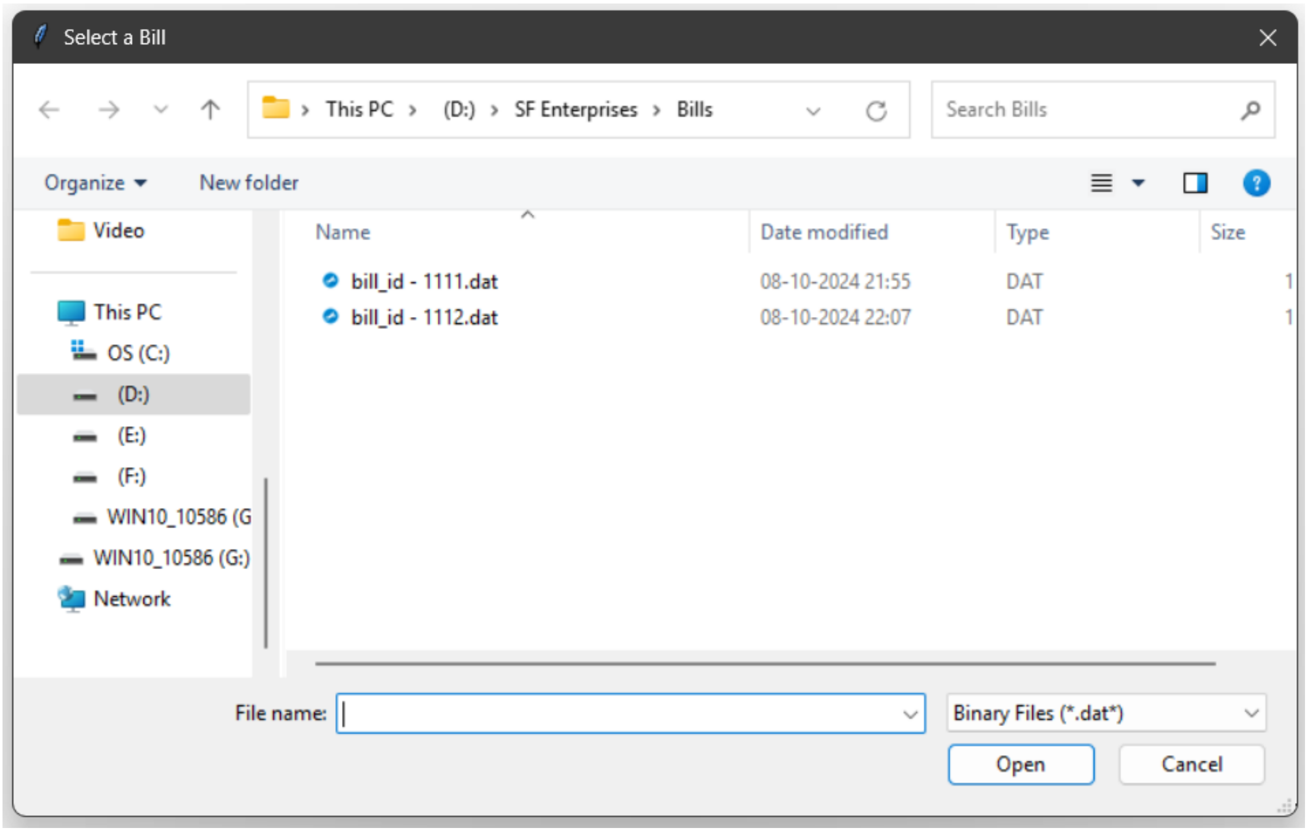


Bill Id - 1112
Date - 2024-10-09
Time - 15:02:39

Item_code	Item	Qty	Price	Item total
122	BALL PEN 0.5	10	10.0	100.0
114	NATRAJ ERASER	15	5.0	75.0
134	NOTE PAD 100	1	70.0	70.0
133	HIGHLIGHTER GREEN	2	20.0	40.0
117	CAMLIN INSTRUMENT BOX	1	250.0	250.0
124	COLOUR PENCIL	1	25.0	25.0

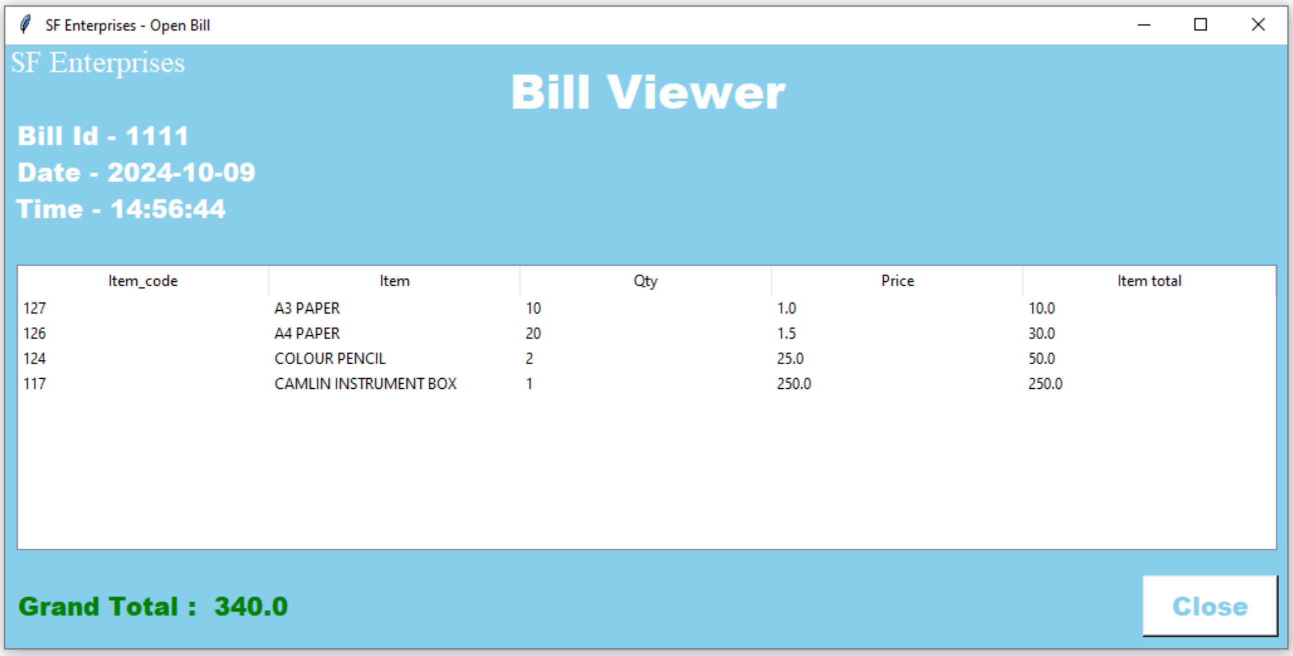
Grand Total : 560.0

11.Open Bill



Name	Date modified	Type	Size
bill_id - 1111.dat	08-10-2024 21:55	DAT	1
bill_id - 1112.dat	08-10-2024 22:07	DAT	1

12. Bill Viewer



Bill Id - 1111
Date - 2024-10-09
Time - 14:56:44

Item_code	Item	Qty	Price	Item total
127	A3 PAPER	10	1.0	10.0
126	A4 PAPER	20	1.5	30.0
124	COLOUR PENCIL	2	25.0	50.0
117	CAMLIN INSTRUMENT BOX	1	250.0	250.0

Grand Total : 340.0

CONCLUSION

- **Efficient Billing:** Invoicefy simplifies invoicing and transaction tracking for businesses.
- **User-Friendly:** Designed with a clean interface, ensuring minimal training for users.
- **Invoice Management:** Allows easy creation, saving, and retrieval of invoices for future reference.
- **Secure Data:** Customer and transaction data is safely stored with backup options to prevent loss.
- **Cross-Platform:** Developed with Python and MySQL, it can operate on multiple operating systems.
- **Time-Saving:** Automates calculations and invoice generation, reducing manual effort.
- **Cost-Effective:** Ideal for small to medium businesses looking for an affordable billing solution.
- **Scalable:** The system is flexible for growing businesses, handling more customers and transactions efficiently.
- **Reliable:** Ensures consistency and accuracy in billing processes.
- **Future Enhancements:** Can be expanded with additional features to cater to evolving business needs.

BIBLIOGRAPHY

- **Preeti Arora Text Book**
- **https://dev.mysql.com/**
- **https://www.w3school.com/**