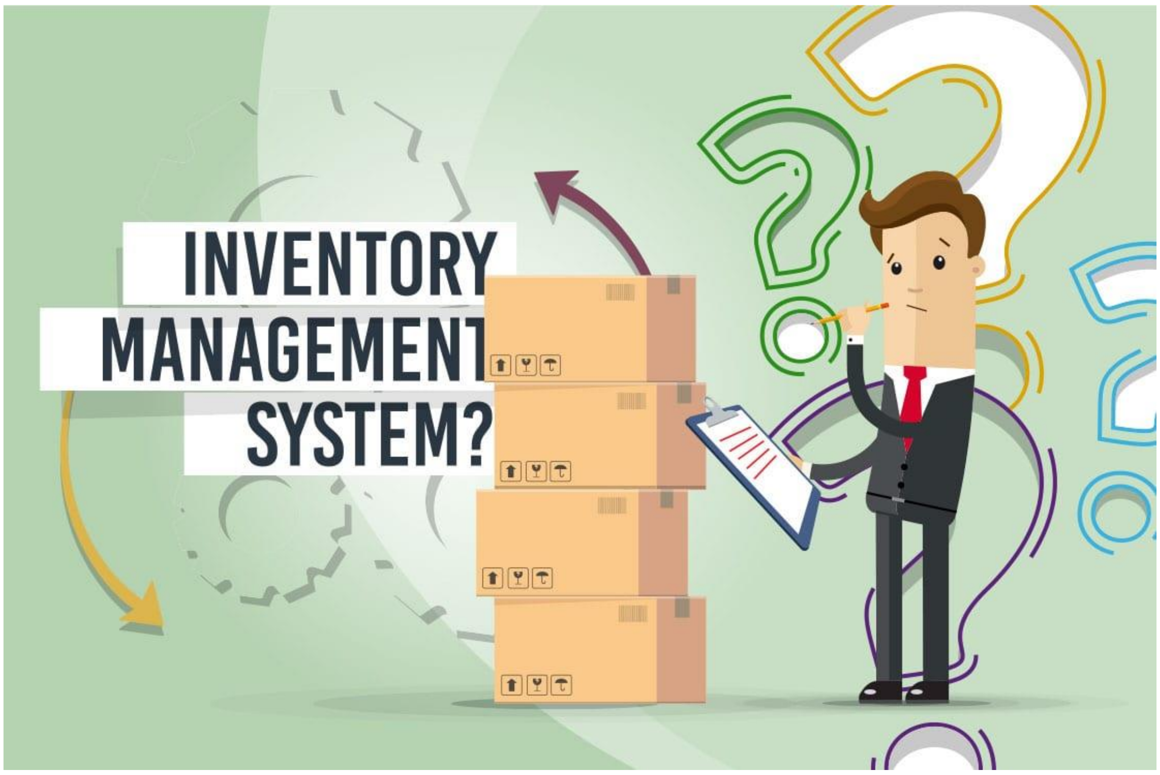
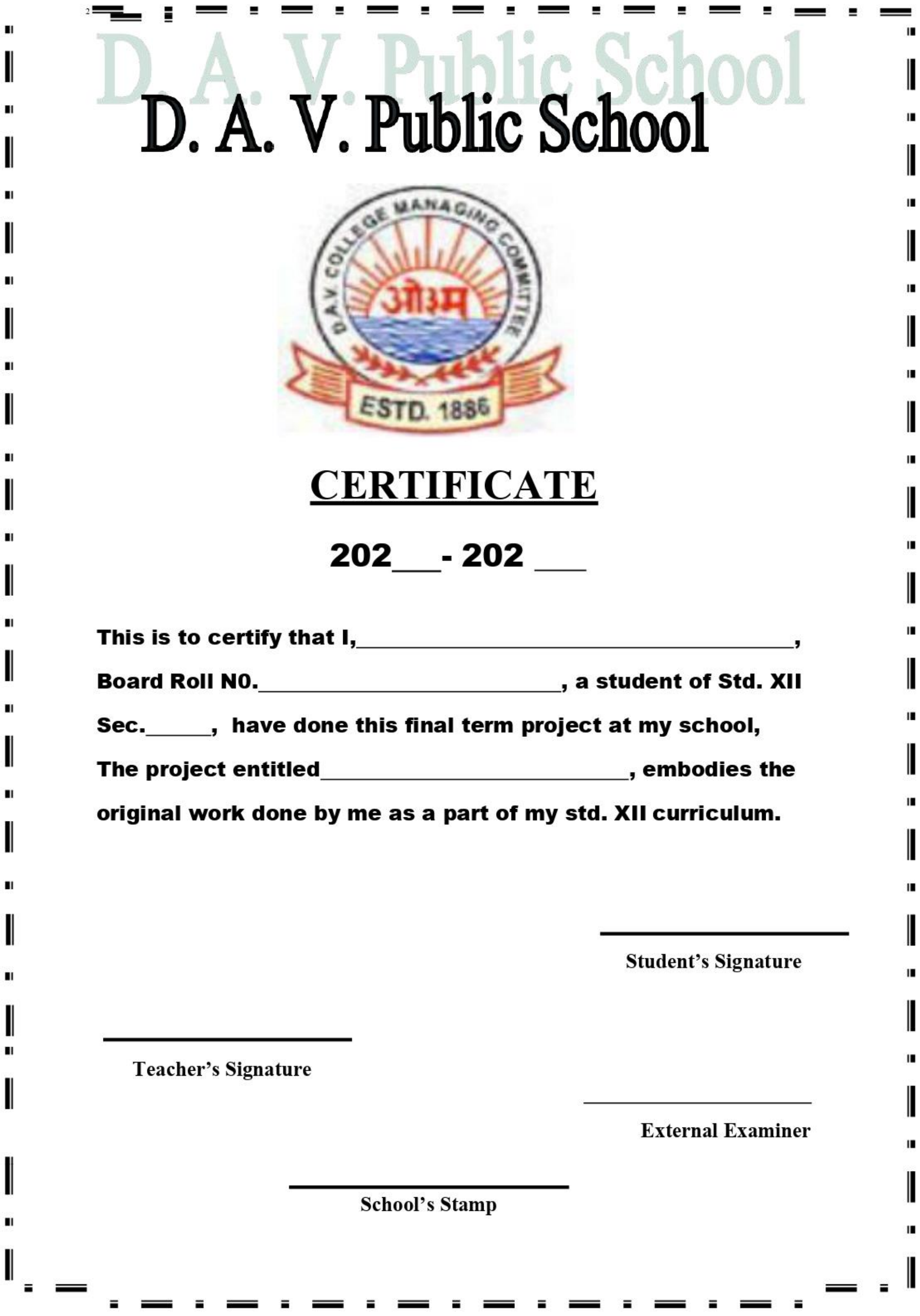


Inventory Management System

Kshitiz Saxena | XII S2 | 16 |

Levin Varghese | XII S2 | 18 |

Tanay Koli | XII S2 | 33 |



D. A. V. Public School

CERTIFICATE

202___- 202 ___

This is to certify that I,____,

Board Roll N0.____________________________, a student of Std. XII

Sec.______, have done this final term project at my school,

The project entitled____________________________, embodies the

original work done by me as a part of my std. XII curriculum.

Student's Signature

Teacher's Signature

External Examiner

School's Stamp

ACKNOWLEDGEMENT

On this great occasion of accomplishment of our project on INVENTORY MANAGEMENT SYSYTEM, we would like to sincerely express our gratitude to Mrs. Sangita Arora, who has been supported through the completion of this project.

We would also be thankful to our principal Mr. Jose Kurian of DAV Public School for providing all the required facilities in completion of this project.

Finally, as one of the team members, I would like to appreciate all my group members for their support and coordination, I hope we will achieve more in our future endeavours.

INDEX

INTRODUCTION TO PYHTON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

I. Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
II. Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
III. Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
IV. Python is a Beginner's Language – Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68,SmallTalk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include –

I. Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
II. Easy-to-read – Python code is more clearly defined and visible to the eyes.
III. Easy-to-maintain – Python's source code is fairly easy-to-maintained.

IV. A broad standard library – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

V. Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

VI. Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

VII. Extendable – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

VIII. Databases – Python provides interfaces to all major commercial databases.

IX. GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

X. Scalable – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

I. It supports functional and structured programming methods as well as OOP.

II. It can be used as a scripting language or can be compiled to byte-code for building

III. large applications.

IV. It provides very high-level dynamic data types and supports dynamic type checking.

V. It supports automatic garbage collection.

VI. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

INTRODUCTION TO MYSQL

A database system is basically a computer-based record keeping system. The collection of data, usually referred to as the database, contains information about one particular enterprise. In a typical file-processing system, permanent records are stored in various file. A number of different application program are written to extract records from files and add records to the appropriate files A data management system is answer to all these problem as it provides a centralized control of the data.

Various advantages of data base system are:

I. Data base system reduce data redundancy (data duplication) to a large extent.
II. Data base system control data inconsistency to a large extent.
III. Database facilitate sharing of data.
IV. Database enforces standards.
V. Centralized data bases can ensure data security.
VI. Integrity can be maintained through databases.

My SQL is a freely available source Relational Database Management System (RDMS) that uses Structured query language (SQL). It is downloadable from site WWW.MYSQL.ORG. In a MYSQL database, information stored in tables. MYSQL provides you with a rich set of features that support a secure environment for storing, maintaining, accessing data. MYSQL is a fast, reliable scalable alternative to many of the commercial RDBMSs available today.

MYSQL was created and is supported by MYSQL AB, a company based in Sweden (ww.mysql.com). This company is now a subsidiary of sun micro systems, which holds the copyright to most of the code base. On APRIL 20, 2009 ORACLE CORP., which develops and sells the proprietary ORACLE DATABASE, announced a deal to acquire sun Microsystems.

INTRODUCTION TO INVENTORY MANAGEMENT SYSTEM

The Inventory Management System is a real-time inventory database capable of connecting multiple stores. This can be used to track the inventory of a single store or to manage the delivery of stock between several branches of a larger franchise. However, the system merely records sales and restocking data and provides warning of low stock at any location through email at a specified interval.

The goal is to reduce the stress of tracking rather than to holder all store maintenance. Further features may consist of the ability to create reports of sales, but again the explanation is left to the management. In addition, since theft does occasionally occur, the system provides solutions for confirming the store inventory and for correcting stock quantities.

Production units use an inventory management system to reduce their transport costs. The system is used to track products and parts as they are transported from a seller to a storeroom, between storerooms, and finally to a retail location or directly to a customer.

The inventory management system is used for various purposes, including:

I. Maintaining and recording the information between too much and too little inventory in the company.
II. Keep track of inventories as it is transported between different locations.
III. Recording product information in a warehouse or other location.
IV. Having a record of Picking, packing, and selling products from a warehouse.
V. Reduction of product obsolescence and decay.
VI. Avoiding out-of-stock situations

SYSTEM DESIGN

I. **Login Page:** Basically, for any software security is a major concern. So, we have developed a secure application. Without being authenticated no user is allowed to view any other interfaces. For the login page, we have a User ID, Password, Profile. After being authenticated user is authorized to perform certain work according to his/her profile.

II. **DASHBOARD:** The dashboard provides flexibility to change quality if any inventory gets damaged. Managing inventory is our main goal so this page is only visible to admin profiles.

III. **Employee Page:** From the admins can change information like addition of employees, correction in name, email id, address, etc.

IV. **Supplier Page:** On this page, users can add supplier's details and store it. It is attached to the products table which registers the given supplier's product in the database and desired data is stored.

V. **Category Page:** On this page, users can add the category of items present in their inventory. It is attached to the products table which registers the given product' category in the database.

VI. **Products Page:** The user can add new items using this page. While adding the items to the database user provides an item description.

VII. **Billing Page:** Using the billing page the employee can place an order and the database would add the item to the order list and the quantity has been decreasing from the products table. Order is attached to the sales table which registers any sales made.

VIII. **Sales Page:** It registers any sale made by the employee and stores the respective sale's bill. This bill can be viewed by the admin.

SOURCE CODE

DashboardFile

```
from tkinter import *

from turtle import width

from employee import EmpClass

from supplier import SupplierClass

from category import CategoryClass

from product import productClass

from sales import salesClass

from PIL import Image, ImageTk

import sqlite3

from tkinter import messagebox

import os

import time



class IMS:

    def __init__(self, root):

        self.root = root

        self.root.geometry("1356x735+0+0")

        self.root.title("Inventory Management System")

        self.root.config(bg="white")


        # title

        self.icon_title = Image.open("images\logo3.png")

        self.icon_title = self.icon_title.resize((150, 125),
Image.ANTIALIAS)

        self.icon_title = ImageTk.PhotoImage(self.icon_title)
        title = Label(self.root, text="Inventory Management System",
image=self.icon_title, compound=LEFT, font=(
            "times new roman", 40, 'bold'), bg="#010c48", fg="white",
anchor="w", padx=20).place(x=0, y=0, relwidth=1, height=70)


        # button_logout
        button_logout = Button(self.root, text="Logout",
command=self.logout, font=("times new roman", 15, "bold"),
                               bg="white", bd=2,
cursor='hand2').place(x=1150, y=10, height=50, width=150)


        # clock
        self.lbl_clock = Label(self.root, text="Welcome To Inventory
Management System\t\t Date : DD-MM-YYYY\t\t Time : HH:MM:SS ",
                               font=("times new roman", 15), bg="black",
fg="white")
        self.lbl_clock.place(x=0, y=70, relwidth=1, height=30)


        # Left Menu
        self.MenuIcon = Image.open("images\logo1.jpg")
        self.MenuIcon = self.MenuIcon.resize((200, 200),
Image.ANTIALIAS)
        self.MenuIcon = ImageTk.PhotoImage(self.MenuIcon)


        LeftMenu = Frame(self.root, bd=2, relief=RIDGE, bg="white")
        LeftMenu.place(x=0, y=102, width=200, height=600)


        lbl_menuLogo = Label(LeftMenu, image=self.MenuIcon)
        lbl_menuLogo.pack(side=TOP, fill=X)
        lftmenu_lbl = Label(LeftMenu, text="Menu", font=("times new
roman", 27, "bold"), bg="cyan", bd=2, relief=RIDGE).pack(side=TOP,
fill=X)

        lftmenu_1 = Button(LeftMenu, text="Employee",
command=self.employee, font=("times new roman", 20, "bold"), bg="white",
bd=4, cursor='hand2').pack(side=TOP, fill=X)

        lftmenu_2 = Button(LeftMenu, text="Supplier",
command=self.supplier, font=("times new roman", 20, "bold"), bg="white",
bd=4, cursor='hand2').pack(side=TOP, fill=X)

        lftmenu_3 = Button(LeftMenu, text="Categories",
command=self.category, font=("times new roman", 20, "bold"), bg="white",
bd=4, cursor='hand2').pack(side=TOP, fill=X)

        lftmenu_4 = Button(LeftMenu, text="Product",
command=self.product, font=("times new roman", 20, "bold"), bg="white",
bd=4, cursor='hand2').pack(side=TOP, fill=X)

        lftmenu_4 = Button(LeftMenu, text="Sales", command=self.sales,
font=("times new roman", 20, "bold"), bg="white", bd=4,
cursor='hand2').pack(side=TOP, fill=X)

        lftmenu_4 = Button(LeftMenu, text="Exit", font=("times new
roman", 20, "bold"), bg="white", bd=4, cursor='hand2').pack(side=TOP,
fill=X)



        # content

        self.lbl_1 = Label(self.root, text="Total Employee\n[0]", bd=5,
relief=RIDGE, bg='blue', fg='white', font=('goudy old style', 28,
'bold'))

        self.lbl_1.place(x=300, y=175, height=150, width=300)



        self.lbl_2 = Label(self.root, text="Total Supplier\n[0]", bd=5,
relief=RIDGE, bg='pink', fg='white', font=('goudy old style', 28,
'bold'))

        self.lbl_2.place(x=650, y=175, height=150, width=300)



        self.lbl_3 = Label(self.root, text="Total Categories\n[0]",
bd=5, relief=RIDGE, bg='orange', fg='white', font=('goudy old style',
28, 'bold'))

        self.lbl_3.place(x=1000, y=175, height=150, width=300)
        self.lbl_4 = Label(self.root, text="Total Products\n[0]", bd=5, 
relief=RIDGE, bg='green', fg='white', font=('goudy old style', 28, 
'bold'))

        self.lbl_4.place(x=300, y=400, height=150, width=300)


        self.lbl_5 = Label(self.root, text="Total Sales\n[0]", bd=5, 
relief=RIDGE, bg='brown', fg='white', font=('goudy old style', 28, 
'bold'))

        self.lbl_5.place(x=650, y=400, height=150, width=300)


        # footer

        lbl_footer = Label(self.root, text=" IMS - Inventory Management 
System ", font=("times new roman", 13, 'bold'), bg="#010c48", 
fg="white").pack(side=BOTTOM, fill=X)


        self.update_content()




# ____




    def employee(self):

        self.new_win = Toplevel(self.root)

        self.new_obj = EmpClass(self.new_win)


# ____

    def supplier(self):

        self.new_win = Toplevel(self.root)

        self.new_obj = SupplierClass(self.new_win)
```

```
# ____
    def category(self):
        self.new_win = Toplevel(self.root)
        self.new_obj = CategoryClass(self.new_win)

# ____
    def product(self):
        self.new_win = Toplevel(self.root)
        self.new_obj = productClass(self.new_win)

# ____
    def sales(self):
        self.new_win = Toplevel(self.root)
        self.new_obj = salesClass(self.new_win)

# ___
    def update_content(self):
        con = sqlite3.connect(database=r"project.db")
        cur = con.cursor()
        try:
            cur.execute("select*from product")
            product = cur.fetchall()
            self.lbl_4.config(
                text=f'Total Products\n[{str(len(product))}]')

            cur.execute("select*from supplier")
            supplier = cur.fetchall()
            self.lbl_2.config(
                text=f'Total Supplier\n[{str(len(supplier))}]')
            cur.execute("select*from category")

            category = cur.fetchall()

            self.lbl_3.config(

                text=f'Total Category\n[{str(len(category))}]')


            cur.execute("select*from employee")

            employee = cur.fetchall()

            self.lbl_1.config(

                text=f'Total Employee\n[{str(len(employee))}]')


            bill = len(os.listdir('bill'))

            self.lbl_5.config(text=f'Total Sales\n[{str(bill)}]')


            time_ = time.strftime("%I:%M:%S")

            date_ = time.strftime("%d-%m:%Y")

            self.lbl_clock.config(text=f"Welcome To Inventory Management
System\t\t Date: {str(date_)}\t\t Time: {str(time_)}", font=(

                "times new roman", 15), bg="black", fg="white")

            self.lbl_clock.after(200, self.update_content)


        except Exception as ex:

            messagebox.showerror("Error", f"Error due to : {str(ex)}")


    # ___

    def logout(self):

        self.root.destroy()

        os.system("login.py")
```

```
if __name__ == "__main__":
    root = Tk()
    obj = IMS(root)
    root.mainloop()
```

<u>EmployeeFile</u>

```
from ast import Delete
import sqlite3
from tkinter import *
from tkinter import font
from turtle import width
from webbrowser import get
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
class EmpClass:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1100x500+220+130")
        self.root.resizable(width=FALSE,height=FALSE)
        self.root.title("Inventory Management System")
        self.root.config(bg='white')
        self.root.focus_force()
        #AllVariables
        self.var_SearchBy=StringVar()
        self.var_SearchTxt=StringVar()
        self.var_EmpID=StringVar()
        self.var_Gender=StringVar()
        self.var_Contact=StringVar()
        self.var_Name=StringVar()
        self.var_DOB=StringVar()

        self.var_DOJ=StringVar()

        self.var_Email=StringVar()

        self.var_Password=StringVar()

        self.var_UserType=StringVar()

        self.var_Address=StringVar()

        self.var_Salary=StringVar()


#searchframe

        SearchFrame=LabelFrame(self.root,text="Search
Employee",font=("goudy old
style",12,'bold'),bd=3,relief=RIDGE,bg="white")

        SearchFrame.place(x=250,y=20,width=600,height=70)


#options


cmb_searchbox=ttk.Combobox(SearchFrame,textvariable=self.var_SearchBy,va
lues=("Select","Name",'Email',"Contact"),state="readonly",justify=CENTER
,font=("goudy old style",15,'bold'))

        cmb_searchbox.place(x=10,y=10,width=180)

        cmb_searchbox.current(0)



txt_search=Entry(SearchFrame,textvariable=self.var_SearchTxt,font=("goud
y old style",15,'bold'),bg="silver").place(x=200,y=10)


btn_search=Button(SearchFrame,text="Search",command=self.search,font=("g
oudy old style",15,'bold'),bg="yellow
green",fg="black",cursor="hand2").place(x=430,y=7,width=130,height=30)


        #title
        title=Label(self.root,text="Employee Details",font=("goudy old
style",15,"bold"),bg="Dark
Blue",fg="White").place(x=50,y=100,width=1000)


        #content

        #row1

        lbl_EmpID=Label(self.root,text="Employee ID",font=("goudy old
style",15,"bold"),bg="White").place(x=50,y=150)

        lbl_Gender=Label(self.root,text="Gender",font=("goudy old
style",15,"bold"),bg="White").place(x=410,y=150)

        lbl_Contact=Label(self.root,text="Contact",font=("goudy old
style",15,"bold"),bg="White").place(x=750,y=150)



txt_EmpID=Entry(self.root,textvariable=self.var_EmpID,font=("goudy old
style",15,'bold'),bg="silver").place(x=180,y=150,width=180)


#txt_Gender=Entry(self.root,textvariable=self.var_Gender,font=("goudy
old style",15,'bold'),bg="silver").place(x=500,y=150,width=180)


cmb_gender=ttk.Combobox(self.root,textvariable=self.var_Gender,values=("
Select","Male",'Female',"Other"),state="readonly",justify=CENTER,font=("
goudy old style",15,'bold'))

        cmb_gender.place(x=500,y=150,width=180)

        cmb_gender.current(0)


txt_Contact=Entry(self.root,textvariable=self.var_Contact,font=("goudy
old style",15,'bold'),bg="silver").place(x=830,y=150,width=180)


        #row2

        lbl_Name=Label(self.root,text="Name",font=("goudy old
style",15,"bold"),bg="White").place(x=50,y=190)

        lbl_DateOfBirth=Label(self.root,text="D.O.B.",font=("goudy old
style",15,"bold"),bg="White").place(x=410,y=190)
```

```
        lbl_DateOfJoining=Label(self.root,text="D.O.J.",font=("goudy old
style",15,"bold"),bg="White").place(x=750,y=190)


        txt_Name=Entry(self.root,textvariable=self.var_Name,font=("goudy
old style",15,'bold'),bg="silver").place(x=180,y=190,width=180)


txt_DateOfBirth=Entry(self.root,textvariable=self.var_DOB,font=("goudy
old style",15,'bold'),bg="silver").place(x=500,y=190,width=180)


txt_DateOfJoining=Entry(self.root,textvariable=self.var_DOJ,font=("goudy
old style",15,'bold'),bg="silver").place(x=830,y=190,width=180)

#row3

        lbl_Email=Label(self.root,text="Email",font=("goudy old
style",15,"bold"),bg="White").place(x=50,y=230)

        lbl_Password=Label(self.root,text="Password",font=("goudy old
style",15,"bold"),bg="White").place(x=410,y=230)

        lbl_Usertype=Label(self.root,text="Usertype",font=("goudy old
style",15,"bold"),bg="White").place(x=750,y=230)



txt_Email=Entry(self.root,textvariable=self.var_Email,font=("goudy old
style",15,'bold'),bg="silver").place(x=180,y=230,width=180)


txt_Password=Entry(self.root,textvariable=self.var_Password,font=("goudy
old style",15,'bold'),bg="silver").place(x=500,y=230,width=180)


cmb_UserType=ttk.Combobox(self.root,textvariable=self.var_UserType,value
s=("Select","Admin",'Employee'),state="readonly",justify=CENTER,font=("g
oudy old style",15,'bold'))

        cmb_UserType.place(x=830,y=230,width=180)

        cmb_UserType.current(0)


        #row4

        lbl_Address=Label(self.root,text="Address",font=("goudy old
style",15,"bold"),bg="White").place(x=50,y=270)
```

```
        lbl_Salary=Label(self.root,text="Salary",font=("goudy old
style",15,"bold"),bg="White").place(x=650,y=270)


        self.txt_Address=Text(self.root,font=("goudy old
style",15,'bold'),bg="silver")

        self.txt_Address.place(x=180,y=270,width=400,height=70)


txt_Salary=Entry(self.root,textvariable=self.var_Salary,font=("goudy old
style",15,'bold'),bg="silver").place(x=720,y=270,width=180)


        #button


btn_add=Button(self.root,text="Save",command=self.add,font=("goudy old
style",15,'bold'),bg="blue",fg="black",cursor="hand2").place(x=600,y=305
,width=110,height=28)


btn_update=Button(self.root,text="Update",command=self.update,font=("gou
dy old
style",15,'bold'),bg="red",fg="black",cursor="hand2").place(x=720,y=305,
width=110,height=28)


btn_delete=Button(self.root,text="Delete",command=self.delete,font=("gou
dy old
style",15,'bold'),bg="green",fg="black",cursor="hand2").place(x=840,y=30
5,width=110,height=28)


btn_clear=Button(self.root,text="Clear",command=self.clear,font=("goudy
old style",15,'bold'),bg=
"brown",fg="black",cursor="hand2").place(x=960,y=305,width=110,height=28
)


        #EmployeeDetails

        emp_frame=Frame(self.root,bd=4,relief=RIDGE)

        emp_frame.place(x=0,y=350,relwidth=1,height=150)


        scrolly=Scrollbar(emp_frame,orient=VERTICAL)
        scrollx=Scrollbar(emp_frame,orient=HORIZONTAL)


        self.EmpTable=ttk.Treeview(emp_frame,columns=("Employee
ID","Name","Email","Gender","Contact","DOB","DOJ","Password","UserType",
"Address","Salary"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.se
t)

        scrollx.pack(side=BOTTOM,fill=X)

        scrolly.pack(side=RIGHT,fill=Y)

        scrollx.config(command=self.EmpTable.xview)

        scrolly.config(command=self.EmpTable.yview)

        self.EmpTable.heading("Employee ID",text="Employee ID")

        self.EmpTable.heading("Name",text="Name")

        self.EmpTable.heading("Email",text="Email")

        self.EmpTable.heading("Gender",text="Gender")

        self.EmpTable.heading("Contact",text="Contact")

        self.EmpTable.heading("DOB",text="Date Of Birth")

        self.EmpTable.heading("DOJ",text="Date Of Joining")

        self.EmpTable.heading("Password",text="Password")

        self.EmpTable.heading("UserType",text="UserType")

        self.EmpTable.heading("Address",text="Address")

        self.EmpTable.heading("Salary",text="Salary")


        self.EmpTable["show"]="headings"


        self.EmpTable.column("Employee ID",width=90)

        self.EmpTable.column("Name",width=100)

        self.EmpTable.column("Email",width=100)

        self.EmpTable.column("Gender",width=100)

        self.EmpTable.column("Contact",width=100)

        self.EmpTable.column("DOB",width=100)
        self.EmpTable.column("DOJ",width=100)

        self.EmpTable.column("Password",width=100)

        self.EmpTable.column("UserType",width=100)

        self.EmpTable.column("Address",width=100)

        self.EmpTable.column("Salary",width=100)


        self.EmpTable.pack(fill=BOTH,expand=1)

        self.EmpTable.bind('<ButtonRelease-1>',self.get_data)

        self.show()


#____

    def add(self):

        con=sqlite3.connect(database=r"project.db")

        cur=con.cursor()

        try:

            if self.var_EmpID.get()=="":

                messagebox.showerror("Error","Employee ID Must Be 
Required",parent=self.root)

            else:

                cur.execute("Select * from employee where 
EmpID=?",(self.var_EmpID.get(),))

                row=cur.fetchone()

                if row!=None:

                    messagebox.showerror("Error","This Employee ID is 
Already Assigned, Try A Different One",parent=self.root)

                else:

                    cur.execute("Insert into employee 
(EmpID,Name,Email,Gender,Contact,DOB,DOJ,Password,UserType,Address,Salar
y) values(?,?,?,?,?,?,?,?,?,?,?)",(

                                self.var_EmpID.get(),

                                self.var_Name.get(),
                                self.var_Email.get(),
                                self.var_Gender.get(),
                                self.var_Contact.get(),
                                self.var_DOB.get(),
                                self.var_DOJ.get(),
                                self.var_Password.get(),
                                self.var_UserType.get(),
                                self.txt_Address.get("1.0",END),
                                self.var_Salary.get(),
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Employee Added 
Sucessfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")


    def show(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            cur.execute("Select * from employee")
            rows=cur.fetchall()
            self.EmpTable.delete(*self.EmpTable.get_children())
            for row in rows:
                self.EmpTable.insert('',END,values=row)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")
```

```
    def get_data(self,ev):
        f=self.EmpTable.focus()
        content=(self.EmpTable.item(f))
        row=content['values']
        #print(row)
        self.var_EmpID.set(row[0])
        self.var_Name.set(row[1])
        self.var_Email.set(row[2])
        self.var_Gender.set(row[3])
        self.var_Contact.set(row[4])
        self.var_DOB.set(row[5])
        self.var_DOJ.set(row[6])
        self.var_Password.set(row[7])
        self.var_UserType.set(row[8])
        self.txt_Address.delete("1.0",END)
        self.txt_Address.insert(END,row[9])
        self.var_Salary.set(row[10])


    def update(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            if self.var_EmpID.get()=="":
                messagebox.showerror("Error","Employee ID Must Be 
Required",parent=self.root)
            else:
                cur.execute("Select * from employee where 
EmpID=?",(self.var_EmpID.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Invalid Employee ID,
Try A Different One",parent=self.root)
                else:
                    cur.execute("Update employee set
Name=?,Email=?,Gender=?,Contact=?,DOB=?,DOJ=?,Password=?,UserType=?,Addr
ess=?,Salary=? where EmpID=?",(
                                    self.var_Name.get(),
                                    self.var_Email.get(),
                                    self.var_Gender.get(),
                                    self.var_Contact.get(),
                                    self.var_DOB.get(),
                                    self.var_DOJ.get(),
                                    self.var_Password.get(),
                                    self.var_UserType.get(),
                                    self.txt_Address.get("1.0",END),
                                    self.var_Salary.get(),
                                    self.var_EmpID.get()
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Employee Updated
Sucessfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")


    def delete(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            if self.var_EmpID.get()=="":
                messagebox.showerror("Error","Employee ID Must Be 
Required",parent=self.root)
            else:
                cur.execute("Select * from employee where 
EmpID=?",(self.var_EmpID.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Invalid Employee ID, 
Try A Different One",parent=self.root)

                else:
                    op=messagebox.askyesno("Confirm","Do You Really Want 
To Delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from employee where 
EmpID=?",(self.var_EmpID.get(),))
                        con.commit()
                        messagebox.showinfo("Delete","Employee Delted 
Successfully",parent=self.root)
                        self.clear()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")


    def clear(self):
        self.var_Name.set("")
        self.var_Email.set("")
        self.var_Gender.set("Select")
        self.var_Contact.set("")
        self.var_DOB.set("")
        self.var_DOJ.set("")
        self.var_Password.set("")

        self.var_UserType.set("Admin")

        self.txt_Address.delete("1.0",END)

        self.var_Salary.set("")

        self.var_EmpID.set("")

        self.var_SearchTxt.set("")

        self.var_SearchBy.set("Select")

        self.show()



    def search(self):

        con=sqlite3.connect(database=r"project.db")

        cur=con.cursor()

        try:

            if self.var_SearchBy.get()=="Select":

                messagebox.showerror("Error","Select Search By 
Option",parent=self.root)

            elif self.var_SearchTxt.get()=="":

                messagebox.showerror("Error","Search Input Should Be 
Required",parent=self.root)

            else:

                cur.execute("Select * from employee where 
"+self.var_SearchBy.get()+" LIKE '%"+self.var_SearchTxt.get()+"%'")

                rows=cur.fetchall()

                if len(rows)!=0:

                    self.EmpTable.delete(*self.EmpTable.get_children())

                    for row in rows:

                        self.EmpTable.insert('',END,values=row)

                else:
                    messagebox.showerror("Error","No Record
Found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")



if __name__=="__main__":
    root=Tk()
    obj=EmpClass(root)
    root.mainloop()
```

<u>SupplierFile</u>

```
from ast import Delete
import sqlite3
from tkinter import *
from tkinter import font
from turtle import width
from webbrowser import get
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
class SupplierClass:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1100x500+220+130")
        self.root.resizable(width=FALSE,height=FALSE)
        self.root.title("Inventory Management System")
        self.root.config(bg='white')

        self.root.focus_force()


        #AllVariables

        self.var_SearchBy=StringVar()

        self.var_SearchTxt=StringVar()

        self.var_SupInvoice=StringVar()

        self.var_SuppName=StringVar()

        self.var_Contact=StringVar()




#options

        lbl_searchbox=Label(self.root,text="Invoice Number",font=("goudy
old style",15,'bold'))

        lbl_searchbox.place(x=670,y=80)



txt_search=Entry(self.root,textvariable=self.var_SearchTxt,font=("goudy
old style",15,'bold'),bg="light yellow").place(x=820,y=80,width=150)


btn_search=Button(self.root,text="Search",command=self.search,font=("gou
dy old
style",15),bg="gold",fg="black",cursor="hand2").place(x=990,y=79,width=1
00,height=27)


        #title

        title=Label(self.root,text="Supplier Details",font=("goudy old
style",25,"bold"),bg="Dark
Blue",fg="White").place(x=50,y=10,width=1000,height=40)


        #content

        #row1
        lbl_SuppInvoice=Label(self.root,text="Invoice
Number",font=("goudy old style",15,"bold"),bg="White").place(x=50,y=70)


txt_SuppInvoice=Entry(self.root,textvariable=self.var_SupInvoice,font=("
goudy old style",15,'bold'),bg="light
yellow").place(x=200,y=70,width=220)



#row2

        lbl_SuppName=Label(self.root,text="Supplier Name",font=("goudy
old style",15,"bold"),bg="White").place(x=50,y=110)


txt_SuppName=Entry(self.root,textvariable=self.var_SuppName,font=("goudy
old style",15,'bold'),bg="light yellow").place(x=200,y=110,width=220)



 #row3

        lbl_Contact=Label(self.root,text="Contact",font=("goudy old
style",15,"bold"),bg="White").place(x=50,y=150)


txt_Contact=Entry(self.root,textvariable=self.var_Contact,font=("goudy
old style",15,'bold'),bg="light yellow").place(x=200,y=150,width=220)

#row4

        lbl_Address=Label(self.root,text="Description",font=("goudy old
style",15,"bold"),bg="White").place(x=50,y=190)

        self.txt_Address=Text(self.root,font=("goudy old
style",15,'bold'),bg="light yellow")

        self.txt_Address.place(x=200,y=190,width=430,height=100)



        #button


btn_add=Button(self.root,text="Save",command=self.add,font=("goudy old
style",15,'bold'),bg="blue",fg="black",cursor="hand2").place(x=200,y=345
,width=100,height=40)


btn_update=Button(self.root,text="Update",command=self.update,font=("gou
dy old
style",15,'bold'),bg="red",fg="black",cursor="hand2").place(x=310,y=345,
width=100,height=40)


btn_delete=Button(self.root,text="Delete",command=self.delete,font=("gou
dy old
style",15,'bold'),bg="green",fg="black",cursor="hand2").place(x=420,y=34
5,width=100,height=40)


btn_clear=Button(self.root,text="Clear",command=self.clear,font=("goudy
old style",15,'bold'),bg=
"brown",fg="black",cursor="hand2").place(x=530,y=345,width=100,height=40
)



        #EmployeeDetails

        emp_frame=Frame(self.root,bd=4,relief=RIDGE)

        emp_frame.place(x=670,y=120,width=420,height=350)



        scrolly=Scrollbar(emp_frame,orient=VERTICAL)

        scrollx=Scrollbar(emp_frame,orient=HORIZONTAL)



        self.SuppTable=ttk.Treeview(emp_frame,columns=("Supplier
Invoice","Supplier
Name","Contact","Description"),yscrollcommand=scrolly.set,xscrollcommand
=scrollx.set)

        scrollx.pack(side=BOTTOM,fill=X)

        scrolly.pack(side=RIGHT,fill=Y)

        scrollx.config(command=self.SuppTable.xview)

        scrolly.config(command=self.SuppTable.yview)

        self.SuppTable.heading("Supplier Invoice",text="Supplier
Invoice")

        self.SuppTable.heading("Supplier Name",text="Supplier Name")

        self.SuppTable.heading("Contact",text="Contact")

        self.SuppTable.heading("Description",text="Description")
        self.SuppTable["show"]="headings"


        self.SuppTable.column("Supplier Invoice",width=100)

        self.SuppTable.column("Supplier Name",width=100)

        self.SuppTable.column("Contact",width=80)

        self.SuppTable.column("Description",width=100)


        self.SuppTable.pack(fill=BOTH,expand=1)

        self.SuppTable.bind('<ButtonRelease-1>',self.get_data)

        self.show()


#____

    def add(self):

        con=sqlite3.connect(database=r"project.db")

        cur=con.cursor()

        try:

            if self.var_SupInvoice.get()=="":

                messagebox.showerror("Error","Supplier Invoice Must Be 
Required",parent=self.root)

            else:

                cur.execute("Select * from Supplier where 
SupInvoice=?",(self.var_SupInvoice.get(),))

                row=cur.fetchone()

                if row!=None:

                    messagebox.showerror("Error","This Supplier Invoice 
is Already Assigned, Try A Different One",parent=self.root)

                else:

                    cur.execute("Insert into 
supplier(SupInvoice,SuppName,Contact,Address) values(?,?,?,?)",(

                                self.var_SupInvoice.get(),
                                self.var_SuppName.get(),
                                self.var_Contact.get(),
                                self.txt_Address.get('1.0',END),

                    ))
                    con.commit()
                    messagebox.showinfo("Success","Supplier Added
Sucessfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")

    def show(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            cur.execute("Select * from supplier")
            rows=cur.fetchall()
            self.SuppTable.delete(*self.SuppTable.get_children())
            for row in rows:
                self.SuppTable.insert('',END,values=row)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")

    def get_data(self,ev):
        f=self.SuppTable.focus()
        content=(self.SuppTable.item(f))
        row=content['values']
        print(row)
        self.var_SupInvoice.set(row[0])

        self.var_SuppName.set(row[1]),

        self.var_Contact.set(row[2]),

        self.txt_Address.delete("1.0",END)

        self.txt_Address.insert(END,row[3]),


    def update(self):

        con=sqlite3.connect(database=r"project.db")

        cur=con.cursor()

        try:

            if self.var_SupInvoice.get()=="":

                messagebox.showerror("Error","Supplier Invoice Must Be 
Required",parent=self.root)

            else:

                cur.execute("Select * from supplier where 
SupInvoice=?",(self.var_SupInvoice.get(),))

                row=cur.fetchone()

                if row==None:

                    messagebox.showerror("Error","Invalid Supplier 
Invoice, Try A Different One",parent=self.root)

                else:

                    cur.execute("Update supplier set 
SuppName=?,Contact=?,Address=? where SupInvoice=?",(

                                    self.var_SuppName.get(),

                                    self.var_Contact.get(),

                                    self.txt_Address.get('1.0',END),

                                    self.var_SupInvoice.get()

                    ))

                    con.commit()

                    messagebox.showinfo("Success","Employee Added 
Sucessfully",parent=self.root)
```

```
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")


    def delete(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            if self.var_SupInvoice.get()=="":
                messagebox.showerror("Error","Supplier Invoice Must Be
Required",parent=self.root)
            else:
                cur.execute("Select * from supplier where
SupInvoice=?",(self.var_SupInvoice.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Invalid Supplier
Invoice, Try A Different One",parent=self.root)


                else:
                    op=messagebox.askyesno("Confirm","Do You Really Want
To Delete?",parent=self.root)
                    if op==True:


                        cur.execute("delete from supplier where
SupInvoice=?",(self.var_SupInvoice.get(),))
                        con.commit()
                        messagebox.showinfo("Delete","Supplier Delted
Successfully",parent=self.root)
                        self.clear()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")


    def clear(self):
        self.var_SuppName.set("")
        self.var_Contact.set("")
        self.txt_Address.delete("1.0",END)
        self.var_SupInvoice.set("")
        self.var_SearchTxt.set("")
        self.show()



    def search(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            if self.var_SearchBy.get()=="Select":
                messagebox.showerror("Error","Select Search By 
Option",parent=self.root)
            else:
                cur.execute("Select * from supplier where 
SupInvoice=?",(self.var_SearchTxt.get(),))
                row=cur.fetchone()
                if row!=0:

self.SuppTable.delete(*self.SuppTable.get_children())
                    self.SuppTable.insert('',END,values=row)
                else:
                    messagebox.showerror("Error","No Record 
Found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")



        if __name__=="__main__":
    root=Tk()
    obj=SupplierClass(root)
    root.mainloop()
```

CategoryFile

```
from ast import Delete
import sqlite3
from tkinter import *
from tkinter import font
from turtle import width
from webbrowser import get
from PIL import Image,ImageTk
from tkinter import ttk,messagebox

import sqlite3

class CategoryClass:

    def __init__(self,root):

        self.root=root

        self.root.geometry("1100x500+220+130")

        self.root.resizable(width=FALSE,height=FALSE)

        self.root.title("Inventory Management System")

        self.root.config(bg='white')

        self.root.focus_force()

#Variable

        self.var_CategoryID=StringVar()

        self.var_Name=StringVar()



#title

        lbl_title=Label(self.root,text="Manage Product
Categories",font=("goudy old style",35,"bold"),bg="Dark
Green",fg="White",bd=3,relief=RIDGE).pack(side=TOP,fill=X,padx=10,pady=5
)

        lbl_name=Label(self.root,text="Enter Category Name",font=("goudy
old style",25,"bold"),bg="white").place(x=50,y=110)

        txt_name=Entry(self.root,textvariable=self.var_Name,font=("goudy
old style",20,"bold"),bg="light yellow").place(x=50,y=170,width=300)




btn_add=Button(self.root,text="Add",command=self.add,font=("goudy old
style",15,'bold'),bg="blue",fg="black",cursor="hand2").place(x=360,y=169
,width=150,height=30)


btn_update=Button(self.root,text="Delete",command=self.delete,font=("gou
dy old
style",15,'bold'),bg="red",fg="black",cursor="hand2").place(x=520,y=169,
width=150,height=30)
```

```
#CategoryDetails
        cat_frame=Frame(self.root,bd=4,relief=RIDGE)
        cat_frame.place(x=700,y=90,width=380,height=110)

        scrolly=Scrollbar(cat_frame,orient=VERTICAL)
        scrollx=Scrollbar(cat_frame,orient=HORIZONTAL)

        self.CatTable=ttk.Treeview(cat_frame,columns=("Category
ID","Name"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.set)
        scrollx.pack(side=BOTTOM,fill=X)
        scrolly.pack(side=RIGHT,fill=Y)
        scrollx.config(command=self.CatTable.xview)
        scrolly.config(command=self.CatTable.yview)
        self.CatTable.heading("Category ID",text="Category ID")
        self.CatTable.heading("Name",text="Name")


        self.CatTable["show"]="headings"

        self.CatTable.column("Category ID",width=100)
        self.CatTable.column("Name",width=100)


        self.CatTable.pack(fill=BOTH,expand=1)
        self.CatTable.bind('<ButtonRelease-1>',self.get_data)

#image
```

```
        self.Image=Image.open("images/image1.png")
        self.Image=self.Image.resize((500,250),Image.ANTIALIAS)
        self.Image=ImageTk.PhotoImage(self.Image)

self.lbl_Image=Label(self.root,image=self.Image,bd=2,relief=RAISED)
        self.lbl_Image.place(x=50,y=220)

        self.Image2=Image.open("images/image1.jpg")
        self.Image2=self.Image2.resize((500,250),Image.ANTIALIAS)
        self.Image2=ImageTk.PhotoImage(self.Image2)

self.lbl_Image2=Label(self.root,image=self.Image2,bd=2,relief=RAISED)
        self.lbl_Image2.place(x=580,y=220)

        self.show()

#functions
    def add(self):
            con=sqlite3.connect(database=r"project.db")
            cur=con.cursor()
            try:
                if self.var_Name.get()=="":
                    messagebox.showerror("Error","Category Name Must Be
Required",parent=self.root)
                else:
                    cur.execute("Select * from Category where
Name=?",(self.var_Name.get(),))
```

```
                    row=cur.fetchone()

                    if row!=None:

                        messagebox.showerror("Error","This Category is
Already Present, Try A Different One",parent=self.root)

                    else:

                        cur.execute("Insert into Category(Name)
values(?)",(self.var_Name.get(),))

                        con.commit()

                        messagebox.showinfo("Success","Category Added
Sucessfully",parent=self.root)

                        self.clear()

            except Exception as ex:

                messagebox.showerror("Error",f"Error due to :
{str(ex)}")

    def show(self):

        con=sqlite3.connect(database=r"project.db")

        cur=con.cursor()

        try:

            cur.execute("Select * from Category")

            rows=cur.fetchall()

            self.CatTable.delete(*self.CatTable.get_children())

            for row in rows:

                self.CatTable.insert('',END,values=row)

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to : {str(ex)}")


    def get_data(self,ev):

        f=self.CatTable.focus()

        content=(self.CatTable.item(f))

        row=content['values']
        #print(row)

        self.var_CategoryID.set(row[0])

        self.var_Name.set(row[1]),


    def delete(self):

            con=sqlite3.connect(database=r"project.db")

            cur=con.cursor()

            try:

                if self.var_CategoryID.get()=="":

                    messagebox.showerror("Error","Category Name Must Be
Required",parent=self.root)

                else:

                    cur.execute("Select * from Category where
CatID=?",(self.var_CategoryID.get(),))

                    row=cur.fetchone()

                    if row==None:

                        messagebox.showerror("Error","Invalid Category ,
Try A Different One",parent=self.root)


                    else:

                        op=messagebox.askyesno("Confirm","Do You Really
Want To Delete?",parent=self.root)

                        if op==True:

                            cur.execute("delete from category where
CatID=?",(self.var_CategoryID.get(),))

                            con.commit()

                            messagebox.showinfo("Delete","Category
Deleted Successfully",parent=self.root)

                            self.clear()

                            self.var_CategoryID.set("")

                            self.var_Name.set("")
            except Exception as ex:
                messagebox.showerror("Error",f"Error due to :
{str(ex)}")

    def clear(self):
        self.var_Name.set("")
        self.var_CategoryID.set("")
        self.show()



if __name__=="__main__":
    root=Tk()
    obj=CategoryClass(root)
    root.mainloop()
```

ProductFile

```
from ast import Delete
import sqlite3
from tkinter import *
from tkinter import font
from turtle import width
from webbrowser import get
from PIL import Image, ImageTk

from tkinter import ttk, messagebox

import sqlite3



class productClass:

    def __init__(self, root):

        self.root = root

        self.root.geometry("1100x500+220+130")

        self.root.title("Inventory Management System")

        self.root.config(bg='white')

        self.root.focus_force()

        # All Variables

        self.var_SearchBy = StringVar()

        self.var_SearchTxt = StringVar()

        self.var_pid = StringVar()

        self.var_cat = StringVar()

        self.var_sup = StringVar()

        self.cat_list = []

        self.sup_list = []

        self.var_name = StringVar()

        self.var_price = StringVar()

        self.var_Qty = StringVar()

        self.var_status = StringVar()

        self.fetch_cat_sup()


        #Product Frame

        product_Frame = Frame(self.root, bd=2, relief=RIDGE, bg='white')

        product_Frame.place(x=10, y=10, width=450, height=480)
        # title

        title = Label(product_Frame, text=" Manage Product Details",
font=(

            "goudy old style", 18, "bold"), bg="Dark Blue",
fg="White").pack(side=TOP, fill=X)


        # column 1

        lbl_cat = Label(product_Frame, text="Category", font=(

            "goudy old style", 18, "bold"), bg="white").place(x=30,
y=60)

        lbl_sup = Label(product_Frame, text="Supplier", font=(

            "goudy old style", 18, "bold"), bg="white").place(x=30,
y=110)

        lbl_pn = Label(product_Frame, text="Name", font=(

            "goudy old style", 18, "bold"), bg="white").place(x=30,
y=160)

        lbl_price = Label(product_Frame, text="Price", font=(

            "goudy old style", 18, "bold"), bg="white").place(x=30,
y=210)

        lbl_Qty = Label(product_Frame, text="Quantity", font=(

            "goudy old style", 18, "bold"), bg="white").place(x=30,
y=260)

        lbl_status = Label(product_Frame, text="Status", font=(

            "goudy old style", 18, "bold"), bg="white").place(x=30,
y=310)


        # column 2

        cmb_cat = ttk.Combobox(product_Frame, textvariable=self.var_cat,
values=self.cat_list,

                               state="readonly", justify=CENTER,
font=("goudy old style", 15, 'bold'))

        cmb_cat.place(x=150, y=60, width=200)
        cmb_cat.current(0)


        cmb_sup = ttk.Combobox(product_Frame, textvariable=self.var_sup,
values=self.sup_list,
                               state="readonly", justify=CENTER,
font=("goudy old style", 15, 'bold'))
        cmb_sup.place(x=150, y=110, width=200)
        cmb_sup.current(0)


        txt_name = Entry(product_Frame, textvariable=self.var_name,
font=(
            "goudy old style", 15, 'bold'),
bg='lightyellow').place(x=150, y=160, width=200)
        txt_price = Entry(product_Frame, textvariable=self.var_price,
font=(
            "goudy old style", 15, 'bold'),
bg='lightyellow').place(x=150, y=210, width=200)
        txt_Qty = Entry(product_Frame, textvariable=self.var_Qty, font=(
            "goudy old style", 15, 'bold'),
bg='lightyellow').place(x=150, y=260, width=200)


        cmb_status = ttk.Combobox(product_Frame,
textvariable=self.var_status, values=(
            "Active", "Inactive"), state="readonly", justify=CENTER,
font=("goudy old style", 15, 'bold'))
        cmb_status.place(x=150, y=310, width=200)
        cmb_status.current(0)


        # button
        btn_add = Button(product_Frame, text="Save", command=self.add,
font=(
            "goudy old style", 15, 'bold'), bg="lightgreen", fg="black",
cursor="hand2").place(x=10, y=400, width=100, height=40)
```

```
        btn_update = Button(product_Frame, text="Update",
command=self.update, font=(

            "goudy old style", 15, 'bold'), bg="lightblue", fg="black",
cursor="hand2").place(x=120, y=400, width=100, height=40)

        btn_delete = Button(product_Frame, text="Delete",
command=self.delete, font=(

            "goudy old style", 15, 'bold'), bg="red", fg="black",
cursor="hand2").place(x=230, y=400, width=100, height=40)

        btn_clear = Button(product_Frame, text="Clear",
command=self.clear, font=(

            "goudy old style", 15, 'bold'), bg="grey", fg="black",
cursor="hand2").place(x=340, y=400, width=100, height=40)



        # searchframe

        SearchFrame = LabelFrame(self.root, text="Search Product",
font=(

            "goudy old style", 12, 'bold'), bd=3, relief=RIDGE,
bg="white")

        SearchFrame.place(x=480, y=10, width=600, height=80)



        # options

        cmb_searchbox = ttk.Combobox(SearchFrame,
textvariable=self.var_SearchBy, values=(

            "Select", "Category", "Supplier", "Name"), state="readonly",
justify=CENTER, font=("goudy old style", 15, 'bold'))

        cmb_searchbox.place(x=10, y=10, width=180)

        cmb_searchbox.current(0)



        txt_search = Entry(SearchFrame, textvariable=self.var_SearchTxt,
font=(

            "goudy old style", 15, 'bold'), bg="silver").place(x=200,
y=10)

        btn_search = Button(SearchFrame, text="Search",
command=self.search, font=(
            "goudy old style", 15, 'bold'), bg="yellow green",
fg="black", cursor="hand2").place(x=430, y=7, width=130, height=30)


        # Product Details

        p_Frame = Frame(self.root, bd=4, relief=RIDGE)

        p_Frame.place(x=480, y=100, width=600, height=390)


        scrolly = Scrollbar(p_Frame, orient=VERTICAL)

        scrollx = Scrollbar(p_Frame, orient=HORIZONTAL)


        self.product_table = ttk.Treeview(p_Frame, columns=(

            "pid","Supplier","Category", "Name", "price", "Qty",
"status"), yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

        scrollx.pack(side=BOTTOM, fill=X)

        scrolly.pack(side=RIGHT, fill=Y)

        scrollx.config(command=self.product_table.xview)

        scrolly.config(command=self.product_table.yview)

        self.product_table.heading("pid", text="Product ID")

        self.product_table.heading("Category", text="Category")

        self.product_table.heading("Supplier", text="Supplier")

        self.product_table.heading("Name", text="Name")

        self.product_table.heading("price", text="Price")

        self.product_table.heading("Qty", text="Qty")

        self.product_table.heading("status", text="Status")


        self.product_table["show"] = "headings"


        self.product_table.column("pid", width=90)

        self.product_table.column("Category", width=100)
        self.product_table.column("Supplier", width=100)

        self.product_table.column("Name", width=100)

        self.product_table.column("price", width=100)

        self.product_table.column("Qty", width=100)

        self.product_table.column("status", width=100)


        self.product_table.pack(fill=BOTH, expand=1)

        self.product_table.bind('<ButtonRelease-1>', self.get_data)

        self.show()



# __________


    def fetch_cat_sup(self):

        self.cat_list.append("Empty")

        self.sup_list.append("Empty")

        con = sqlite3.connect(database=r"project.db")

        cur = con.cursor()

        try:

            cur.execute("Select Name from category")

            cat = cur.fetchall()

            if len(cat) > 0:

                del self.cat_list[:]

                self.cat_list.append("Select")

                for i in cat:

                    self.cat_list.append(i[0])

            cur.execute("Select SuppName from supplier")

            sup=cur.fetchall()

            if len(sup)>0:
                del self.sup_list[:]

                self.sup_list.append("Select")

                for i in sup:

                    self.sup_list.append(i[0])

        except Exception as ex:

            messagebox.showerror("Error", f"Error due to : {str(ex)}")




    def add(self):

        con = sqlite3.connect(database=r"project.db")

        cur = con.cursor()

        try:

            if self.var_cat.get() == "Select" or self.var_cat.get() ==
"Empty" or self.var_sup.get() == "Select" or self.var_sup.get() ==
"Empty" or self.var_name.get() == "":

                messagebox.showerror(

                    "Error", "All Fields are Required",
parent=self.root)

            else:

                cur.execute("Select * from product where Name=?",

                            (self.var_name.get(),))

                row = cur.fetchone()

                if row!=None:

                    messagebox.showerror(

                        "Error", "Product already present, Try A
Different One", parent=self.root)

                else:

                    cur.execute("Insert into product(Category, Supplier,
Name,price, Qty, status) values(?,?,?,?,?,?)", (

                                self.var_cat.get(),

                                self.var_sup.get(),
                                self.var_name.get(),
                                self.var_price.get(),
                                self.var_Qty.get(),
                                self.var_status.get(),
                                ))
                    con.commit()
                    messagebox.showinfo(
                        "Success", "Product Added Sucessfully",
parent=self.root)
                    self.show()
                    self.clear()
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")

    def show(self):
        con = sqlite3.connect(database=r"project.db")
        cur = con.cursor()
        try:
            cur.execute("Select * from product")
            rows = cur.fetchall()

self.product_table.delete(*self.product_table.get_children())
            for row in rows:
                self.product_table.insert('', END, values=row)
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")

    def get_data(self, ev):
        f = self.product_table.focus()
        content = (self.product_table.item(f))
        row = content['values']
        self.var_pid.set(row[0])
        self.var_cat.set(row[2])
        self.var_sup.set(row[1])
        self.var_name.set(row[3])
        self.var_price.set(row[4])
        self.var_Qty.set(row[5])
        self.var_status.set(row[6])

    def update(self):
        con = sqlite3.connect(database=r"project.db")
        cur = con.cursor()
        try:
            if self.var_pid.get() == "":
                messagebox.showerror(
                    "Error", "Please Select Product from the list",
parent=self.root)
            else:
                cur.execute("Select * from product where pid=?",
                            (self.var_pid.get(),))
                row = cur.fetchone()
                if row == None:
                    messagebox.showerror(
                        "Error", "Invalid Product, Try A Different One",
parent=self.root)
                else:
                    cur.execute("Update product set
Category=?,Supplier=?,Name=?,price=?,Qty=?,status=? where pid=?", (
                                self.var_cat.get(),
                              self.var_sup.get(),
                              self.var_name.get(),
                              self.var_price.get(),
                              self.var_Qty.get(),
                              self.var_status.get(),
                              self.var_pid.get()
                              ))
                    con.commit()
                    messagebox.showinfo(
                        "Success", "Product Updated Sucessfully",
parent=self.root)
                    self.show()
                    self.clear()
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")


    def delete(self):
        con = sqlite3.connect(database=r"project.db")
        cur = con.cursor()
        try:
            if self.var_pid.get() == "":
                messagebox.showerror(
                    "Error", "Select Product from the list",
parent=self.root)
            else:
                cur.execute("Select * from product where pid=?",
                            (self.var_pid.get(),))
                row = cur.fetchone()
                if row == None:
                    messagebox.showerror(
                        "Error", "Invalid Product, Try A Different One",
parent=self.root)

                else:
                    op = messagebox.askyesno(
                            "Confirm", "Do You Really Want To Delete?",
parent=self.root)
                    if op == True:
                        cur.execute(
                            "delete from product where pid=?",
(self.var_pid.get(),))
                        con.commit()
                        messagebox.showinfo(
                            "Delete", "Product Deleted Successfully",
parent=self.root)
                        self.clear()
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")

    def clear(self):
        self.var_cat.set("Select"),
        self.var_sup.set("Select")
        self.var_name.set("")
        self.var_price.set("")
        self.var_Qty.set("")
        self.var_status.set("Active")
        self.var_pid.set("")
        self.var_SearchTxt.set("")
        self.var_SearchBy.set("Select")
        self.show()


    def search(self):
        con=sqlite3.connect(database=r"project.db")
        cur=con.cursor()
        try:
            if self.var_SearchBy.get()=="Select":
                messagebox.showerror("Error","Select Search By 
Option",parent=self.root)
            elif self.var_SearchTxt.get()=="":
                messagebox.showerror("Error","Search Input Should Be 
Required",parent=self.root)
            else:
                cur.execute("Select * from product where 
"+self.var_SearchBy.get()+" LIKE '%"+self.var_SearchTxt.get()+"%'")
                rows=cur.fetchall()
                if len(rows)!=0:
                    
self.product_table.delete(*self.product_table.get_children())
                    for row in rows:
                        self.product_table.insert('',END,values=row)
                else:
                    messagebox.showerror("Error","No Record 
Found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}")
if __name__ == "__main__":
    root = Tk()
    obj = productClass(root)
    root.mainloop()
```

```
#Create DB

import sqlite3

def create_db():

    con=sqlite3.connect(database=r"project.db")

    cur=con.cursor()

    cur.execute("CREATE TABLE IF NOT EXISTS employee(EmpID INTEGER PRIMARY KEY AUTOINCREMENT,Name text,Email text,Gender text,Contact text,DOB text,DOJ text,Password text,UserType text,Address text,Salary text)")

    con.commit()



    cur.execute("CREATE TABLE IF NOT EXISTS supplier(SupInvoice INTEGER PRIMARY KEY AUTOINCREMENT,SuppName text,Contact text,Address text)")

    con.commit()



    cur.execute("CREATE TABLE IF NOT EXISTS category(CatID INTEGER PRIMARY KEY AUTOINCREMENT,Name text)")

    con.commit()



    cur.execute("CREATE TABLE IF NOT EXISTS product(pid INTEGER PRIMARY KEY AUTOINCREMENT,Supplier text,Category text,Name text,price text,Qty text,status text)")

    con.commit()




create_db()
```

BillingFile

```
from cProfile import label
from cgitb import text
from http.client import EXPECTATION_FAILED
from operator import index
import sqlite3
from sqlite3 import Cursor
from sys import float_repr_style
from tkinter import *
from tkinter import font
from tkinter import messagebox
from tokenize import String
from tkinter import ttk
from turtle import width
from unittest import result
from PIL import Image, ImageTk
import time
import os
import tempfile


class BI:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1400x735+0+0")
        self.root.title("Inventory Management System")
        self.root.config(bg="white")
        self.cart_list = []
        self.chk_print = 0
        # title
        self.icon_title = Image.open("images\logo1.jpg")
        self.icon_title = self.icon_title.resize((150, 125),
Image.ANTIALIAS)
        self.icon_title = ImageTk.PhotoImage(self.icon_title)
        title = Label(self.root, text="Inventory Management System",
image=self.icon_title, compound=LEFT, font=(
            "times new roman", 40, 'bold'), bg="#010c48", fg="white",
anchor="w", padx=20).place(x=0, y=0, relwidth=1, height=70)

        # button_logout
        button_logout = Button(self.root, text="Logout",
command=self.logout, font=("times new roman", 17, "bold"),
                               bg="white", bd=2,
cursor='hand2').place(x=1150, y=10, height=50, width=150)

        # clock
        self.lbl_clock = Label(self.root, text="Welcome To Inventory
Management System\t\t Date : DD-MM-YYYY\t\t Time : HH:MM:SS ",
                               font=("times new roman", 15), bg="black",
fg="white")
        self.lbl_clock.place(x=0, y=70, relwidth=1, height=30)

        # product frame
        ProductFrame = Frame(self.root, bd=4, relief=RIDGE, bg="white")
        ProductFrame.place(x=10, y=110, width=410, height=550)

        pTitle = Label(ProductFrame, text="All Products", font=(
            "goudy old style", 20, "bold"), bg="black",
fg="white").pack(side=TOP, fill=X)



        ProductFrame3 = Frame(ProductFrame, bd=3, relief=RIDGE)
        ProductFrame3.place(x=2, y=50, width=398, height=500)

        scrolly = Scrollbar(ProductFrame3, orient=VERTICAL)
        scrollx = Scrollbar(ProductFrame3, orient=HORIZONTAL)

        self.product_Table = ttk.Treeview(ProductFrame3, columns=(
            "PID", "Name", "Price", "QTY", "Status"),
yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

        scrollx.pack(side=BOTTOM, fill=X)
        scrolly.pack(side=RIGHT, fill=Y)
        scrollx.config(command=self.product_Table.xview)
        scrolly.config(command=self.product_Table.yview)

        self.product_Table.heading("PID", text="PID No.")
        self.product_Table.heading("Name", text="Name")
        self.product_Table.heading("Price", text="Price")
        self.product_Table.heading("QTY", text="QTY")
        self.product_Table.heading("Status", text="Status")
        self.product_Table["show"] = "headings"

        self.product_Table.column("PID", width=50)
        self.product_Table.column("Name", width=100)
        self.product_Table.column("Price", width=80)
        self.product_Table.column("QTY", width=50)
        self.product_Table.column("Status", width=100)
        self.product_Table.pack(fill=BOTH, expand=1)
        self.product_Table.bind('<ButtonRelease-1>', self.get_data)

        lbl_note = Label(ProductFrame3, text="Note: 'Enter 0 Quantity to
remove product from the Cart'", font=(
            "goudy old style", 10), bg="white",
fg="red").pack(side=BOTTOM, fill=Y)

        # Customer Frame
        self.var_cname = StringVar()
        self.var_contact = StringVar()
        CustomerFrame = Frame(self.root, bd=4, relief=RIDGE, bg="white")
        CustomerFrame.place(x=420, y=110, width=530, height=70)

        cTitle = Label(CustomerFrame, text="Customer Details", font=(
            "goudy old style", 15), bg="Lightgray").pack(side=TOP,
fill=X)
        lbl_search = Label(CustomerFrame, text="Name", font=(
            "times new roman", 15,), bg="white").place(x=2, y=35)
        txt_name = Entry(CustomerFrame, textvariable=self.var_cname,
font=(
            "times new roman", 15), bg="light yellow",
cursor="hand2").place(x=60, y=35, width=180)

        lbl_contact = Label(CustomerFrame, text="Contact No.", font=(
            "times new roman", 15,), bg="white").place(x=260, y=35)
        txt_contact = Entry(CustomerFrame,
textvariable=self.var_contact, font=(
            "times new roman", 15), bg="light yellow",
cursor="hand2").place(x=370, y=35, width=140)

        calc_cartFrame = Frame(self.root, bd=2, relief=RIDGE,
bg="white")
        calc_cartFrame.place(x=420, y=190, width=530, height=360)

        self.var_calc_input = StringVar()
        CalcFrame = Frame(calc_cartFrame, bd=4, relief=RIDGE,
bg="white")
        CalcFrame.place(x=5, y=10, width=268, height=340)

        self.txt_calc_input = Entry(CalcFrame,
textvariable=self.var_calc_input, font=(
            "times new roman", 15, "bold"), width=22, bd=10,
relief=GROOVE, justify=RIGHT)
        self.txt_calc_input.grid(row=0, columnspan=4)

        btn_7 = Button(CalcFrame, text=7, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            7), bd=5, width=4, pady=10, cursor='hand2').grid(row=1,
column=0)
        btn_8 = Button(CalcFrame, text=8, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            8), bd=5, width=4, pady=10, cursor='hand2').grid(row=1,
column=1)
        btn_9 = Button(CalcFrame, text=9, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            9), bd=5, width=4, pady=10, cursor='hand2').grid(row=1,
column=2)
        btn_sum = Button(CalcFrame, text='+', font=('times new roman',
15, 'bold'), command=lambda: self.get_input(
            '+'), bd=5, width=4, pady=10, cursor='hand2').grid(row=1,
column=3)

        btn_4 = Button(CalcFrame, text=4, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            4), bd=5, width=4, pady=10, cursor='hand2').grid(row=2,
column=0)
        btn_5 = Button(CalcFrame, text=5, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            5), bd=5, width=4, pady=10, cursor='hand2').grid(row=2,
column=1)
        btn_6 = Button(CalcFrame, text=6, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            6), bd=5, width=4, pady=10, cursor='hand2').grid(row=2,
column=2)
        btn_subtract = Button(CalcFrame, text='-', font=('times new
roman', 15, 'bold'), command=lambda: self.get_input(
            '-'), bd=5, width=4, pady=10, cursor='hand2').grid(row=2,
column=3)

        btn_1 = Button(CalcFrame, text=1, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            1), bd=5, width=4, pady=10, cursor='hand2').grid(row=3,
column=0)
        btn_2 = Button(CalcFrame, text=2, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            2), bd=5, width=4, pady=10, cursor='hand2').grid(row=3,
column=1)
        btn_3 = Button(CalcFrame, text=3, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            3), bd=5, width=4, pady=10, cursor='hand2').grid(row=3,
column=2)
        btn_multiply = Button(CalcFrame, text='*', font=('times new
roman', 15, 'bold'), command=lambda: self.get_input(
            '*'), bd=5, width=4, pady=10, cursor='hand2').grid(row=3,
column=3)

        btn_0 = Button(CalcFrame, text=0, font=('times new roman', 15,
'bold'), command=lambda: self.get_input(
            0), bd=5, width=4, pady=15, cursor='hand2').grid(row=4,
column=0)
        btn_c = Button(CalcFrame, text='C', font=('times new roman', 15,
'bold'), command=self.clear,
                       bd=5, width=4, pady=15,
cursor='hand2').grid(row=4, column=1)
        btn_divide = Button(CalcFrame, text='/', font=('times new
roman', 15, 'bold'), command=lambda: self.get_input(
            '/'), bd=5, width=4, pady=15, cursor='hand2').grid(row=4,
column=2)
        btn_isto = Button(CalcFrame, text='=', font=('times new roman',
15, 'bold'), command=self.perform,
                          bd=5, width=4, pady=15,
cursor='hand2').grid(row=4, column=3)

        cart_Frame = Frame(calc_cartFrame, bd=3, relief=RIDGE)
        cart_Frame.place(x=280, y=8, width=245, height=342)
        self.cartTitle = Label(cart_Frame, text="Cart \t Total Product:
[0]", font=(
            "goudy old style", 15), bg="Lightgray")
        self.cartTitle.pack(side=TOP, fill=X)

        scrolly = Scrollbar(cart_Frame, orient=VERTICAL)
        scrollx = Scrollbar(cart_Frame, orient=HORIZONTAL)

        self.cartTable = ttk.Treeview(cart_Frame, columns=(
            "PID", "Name", "Price", "QTY"), yscrollcommand=scrolly.set,
xscrollcommand=scrollx.set)

        scrollx.pack(side=BOTTOM, fill=X)
        scrolly.pack(side=RIGHT, fill=Y)
        scrollx.config(command=self.cartTable.xview)
        scrolly.config(command=self.cartTable.yview)

        self.cartTable.heading("PID", text="PID")
        self.cartTable.heading("Name", text="Name")
        self.cartTable.heading("Price", text="Price")
        self.cartTable.heading("QTY", text="QTY")

        self.cartTable["show"] = "headings"

        self.cartTable.column("PID", width=40)
        self.cartTable.column("Name", width=90)
        self.cartTable.column("Price", width=90)
        self.cartTable.column("QTY", width=40)

        self.cartTable.pack(fill=BOTH, expand=1)
        self.cartTable.bind('<ButtonRelease-1>', self.get_data_cart)

        # ADD CART BUTTON
        self.var_pid = StringVar()
        self.var_pname = StringVar()
        self.var_QTY = StringVar()
        self.var_Price = StringVar()
        self.var_Stock = StringVar()

        Add_calc_cartFrame = Frame(self.root, bd=2, relief=RIDGE,
bg="white")
        Add_calc_cartFrame.place(x=420, y=550, width=530, height=110)

        lbl_p_name = Label(Add_calc_cartFrame, text="Product Name",
font=(
            "Times new roman", 15), bg="white").place(x=5, y=5)
        txt_p_name = Entry(Add_calc_cartFrame,
textvariable=self.var_pname, font=(
            "Times new roman", 15), bg="light yellow").place(x=5, y=35,
width=190, height=22)

        lbl_p_price = Label(Add_calc_cartFrame, text="Product Price",
font=(
            "Times new roman", 15), bg="white").place(x=210, y=5)
        txt_p_price = Entry(Add_calc_cartFrame,
textvariable=self.var_Price, font=(
            "Times new roman", 15), bg="light yellow").place(x=210,
y=35, width=150, height=22)

        lbl_p_qty = Label(Add_calc_cartFrame, text="Quantity", font=(
            "Times new roman", 15), bg="white").place(x=380, y=5)
        txt_p_qty = Entry(Add_calc_cartFrame, textvariable=self.var_QTY,
font=(
            "Times new roman", 15), bg="light yellow").place(x=380,
y=35, width=120, height=22)

        self.lbl_p_stock = Label(Add_calc_cartFrame, text="In Stock",
font=(
            "Times new roman", 15), bg="white")
        self.lbl_p_stock.place(x=5, y=70)

        btn_clear_cart = Button(Add_calc_cartFrame, text="Clear",
command=self.clear_cart, font=(
            "times new roman", 15, 'bold'), bg="lightgray",
cursor="hand2").place(x=180, y=70, width=150, height=30)

        btn_add_cart = Button(Add_calc_cartFrame, text="Add/Update
Cart", command=self.add_update_cart, font=(
            "times new roman", 15, 'bold'), bg="Orange",
cursor="hand2").place(x=340, y=70, width=180, height=30)

        billframe1 = Frame(self.root, bd=2, relief=RIDGE, bg='white')
        billframe1.place(x=958, y=110, width=430, height=410)
        bTitle = Label(billframe1, text="Customer Bill Area", font=(
            "goudy old style", 20, "bold"), bg="#f44336",
fg="white").pack(side=TOP, fill=X)
        scrolly = Scrollbar(billframe1, orient=VERTICAL)
        scrolly.pack(side=RIGHT, fill=Y)

        self.txt_bill_area1 = Text(billframe1,
yscrollcommand=scrolly.set)
        self.txt_bill_area1.pack(fill=BOTH, expand=1)
        scrolly.config(command=self.txt_bill_area1.yview)

        billMenuFrame = Frame(self.root, bd=2, relief=RIDGE,
bg='white').place(
            x=958, y=520, width=430, height=140)
        self.lbl_amount = Label(billMenuFrame, text='Bill Amount\n0',
font=(
            "goudy old style", 15, 'bold'), bg="Blue", fg='white')
        self.lbl_amount.place(x=963, y=525, width=120, height=70)
        self.lbl_discount = Label(billMenuFrame, text='Discount\n[5%]',
font=(
            "goudy old style", 15, 'bold'), bg="Green", fg='white')
        self.lbl_discount.place(x=1100, y=525, width=120, height=70)
        self.lbl_net_pay = Label(billMenuFrame, text='Net Pay\n0',
font=(
            "goudy old style", 15, 'bold'), bg="Orange", fg='white')
        self.lbl_net_pay.place(x=1230, y=525, width=140, height=70)

        btn_amount = Button(billMenuFrame, text='Print',
command=self.print_bill, font=("goudy old style", 15, 'bold'),
                            bg="lightYellow", fg='gray',
cursor='hand2').place(x=963, y=595, width=120, height=60)
        btn_clear_all = Button(billMenuFrame, text='Clear All',
command=self.clear_all, font=(
            "goudy old style", 15, 'bold'), bg="red", fg='white',
cursor='hand2').place(x=1100, y=595, width=120, height=60)
        btn_generate = Button(billMenuFrame, text='Generate Bill',
command=self.generate_bill, font=(
            "goudy old style", 15, 'bold'), bg="Light Green",
fg='white', cursor='hand2').place(x=1230, y=595, width=140, height=60)

####FOOOOOOOTTTTTEEEEERRRRRR#######
        # footer
        lbl_footer = Label(self.root, text=" IMS - Inventory Management
System ", font=(
            "times new roman", 15, 'bold'), bg="#010c48",
fg="white").pack(side=BOTTOM, fill=X)

        self.show()
        self.date_time()

    def get_input(self, num):
        xnum = self.var_calc_input.get()+str(num)
        self.var_calc_input.set(xnum)

    def clear(self):
        self.var_calc_input.set('')

    def perform(self):
        result = self.var_calc_input.get()
        self.var_calc_input.set(eval(result))

    def show(self):
        con = sqlite3.connect(database=r"project.db")
        cur = con.cursor()
        try:
            #
self.product_Table=ttk.Treeview(ProductFrame3,columns=("PID","Name","Pri
ce","QTY","Status"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.se
t)
            cur.execute(
                "Select pid,name,price,qty,status from product where
status='Active' ")
            rows = cur.fetchall()

self.product_Table.delete(*self.product_Table.get_children())
            for row in rows:
                self.product_Table.insert('', END, values=row)
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")



    def get_data(self, ev):
        f = self.product_Table.focus()
        content = (self.product_Table.item(f))
        row = content['values']
        self.var_pid.set(row[0])
        self.var_pname.set(row[1])
        self.var_Price.set(row[2])
        self.lbl_p_stock.config(text=f'In Stock [{str(row[3])}]')
        self.var_Stock.set(row[4])
        self.var_QTY.set('1')

    def get_data_cart(self, ev):
        f = self.cartTable.focus()
        content = (self.cartTable.item(f))
        row = content['values']
        # pid,name,price,qty,status
        self.var_pid.set(row[0])
        self.var_pname.set(row[1])
        self.var_Price.set(row[2])
        self.var_QTY.set(row[3])
        self.lbl_p_stock.config(text=f'In Stock [{int(row[4])}]')
        self.var_Stock.set(row[4])

    def add_update_cart(self):
        if self.var_pid.get() == '':
            messagebox.showerror(
                'Error', 'Please select product from the list',
parent=self.root)
        elif self.var_QTY.get() == '':
            messagebox.showerror(
                'Error', "Quantity is required", parent=self.root)
        elif (self.var_QTY.get()) > (self.var_Stock.get()):
            messagebox.showerror('error', "Invalid Quantity",
parent=self.root)
        else:
            #price_cal=float(
int(self.var_QTY.get())*float(self.var_Price.get()))
            # print(price_cal)
            price_cal = self.var_Price.get()
            # pid,name,price,qty,status
            cart_data = [self.var_pid.get(), self.var_pname.get(
            ), price_cal, self.var_QTY.get(), self.var_Stock.get()]

            # update cart
            present = 'no'
            index_ = 0
            for row in self.cart_list:
                if self.var_pid.get() == row[0]:
                    present = 'yes'
                    break
                index_ += 1
            if present == 'yes':
                op = messagebox.askyesno(
                    'Confirm', "Product Already present\nDo you want to
update/remove from Cart list", parent=self.root)
                if op == True:
                    if self.var_QTY.get() == '0':
                        self.cart_list.pop(index_)
                    else:
                        # pid,name,price,qty,status
                        # self.cart_list[index_][2]=price_cal #price
                        self.cart_list[index_][3] = self.var_QTY.get()
# qty
            else:
                self.cart_list.append(cart_data)

            self.show_cart()
            self.bill_update()

    def bill_update(self):
        self.bill_amount = 0
        self.net_pay = 0
        self.discount = 0
        for row in self.cart_list:
            self.bill_amount =
self.bill_amount+(float(row[2])*int(row[3]))

        self.discount = (self.bill_amount*5)/100
        self.net_pay = self.bill_amount-self.discount
        self.lbl_amount.config(text=f'Bill
Amount\n[{str(self.bill_amount)}]')
        self.lbl_net_pay.config(text=f'Net
pay(Rs.)\n[{str(self.net_pay)}]')
        self.cartTitle.config(
            text=f"Cart \t Total Product: [{str(len(self.cart_list))}]")

    def show_cart(self):
        try:
            #
self.product_Table=ttk.Treeview(ProductFrame3,columns=("PID","Name","Price","QTY","Status"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.set)
            self.cartTable.delete(*self.cartTable.get_children())
            for row in self.cart_list:
                self.cartTable.insert('', END, values=row)
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")

    def generate_bill(self):
        if self.var_cname.get() == '' or self.var_contact.get() == '':
            messagebox.showerror(
                "Error", f'Customer Details are required',
parent=self.root)
        elif len(self.cart_list) == 0:
            messagebox.showerror(
                "error", f"Please Add Product in the cart!!",
parent=self.root)
        else:
            # ======Bill Top======
            self.bill_top()
            # ======Bill Middle====
            self.bill_middle()
            # ======Bill Bottom=====
            self.bill_bottom()

            fp = open(f'bill/{str(self.invoice)}.txt', 'w')
            fp.write(self.txt_bill_area1.get('1.0', END))
            fp.close()
            messagebox.showinfo(
                "Saved", "Bill has been Generated", parent=self.root)
            self.chk_print = 1

    def bill_top(self):
        self.invoice = int(time.strftime("%H%M%S")) + \
            int(time.strftime("%d%m%Y"))
        bill_top_temp = f'''
\t\tXYZ-Inventory
\t Phone No. 98725***** , Mumbai-410208
{str("="*47)}
 Customer Name: {self.var_cname.get()}
 Ph no. :{self.var_contact.get()}
 Bill No. {str(self.invoice)}\t\t\tDate:
{str(time.strftime("%d/%m/%Y"))}
{str("="*47)}
 Product Name\t\t\tQTY\tPrice
{str("="*47)}
        '''
        self.txt_bill_area1.delete('1.0', END)
        self.txt_bill_area1.insert('1.0', bill_top_temp)
    def bill_bottom(self):
        bill_bottom_temp = f'''
{str("="*47)}
 Bill Amount\t\t\t\tRs.{self.bill_amount}
 Discount\t\t\t\tRs.{self.discount}
 Net Pay\t\t\t\tRs.{self.net_pay}
{str("="*47)}\n
        '''
        self.txt_bill_area1.insert(END, bill_bottom_temp)

    def bill_middle(self):
        for row in self.cart_list:
        # pid,name,price,qty,stock
            name=row[1]
            qty=row[3]
            price=float(row[2])*int(row[3])
            price=str(price)
            self.txt_bill_area1.insert(END,"\n
"+name+"\t\t\t"+qty+"\tRs."+price)

    def clear_cart(self):
        self.var_pid.set('')
        self.var_pname.set('')
        self.var_Price.set('')
        self.var_QTY.set('')
        self.lbl_p_stock.config(text=f'In Stock')
        self.var_Stock.set('')

    def clear_all(self):
        del self.cart_list[:]
        self.var_cname.set('')
        self.var_contact.set('')
        self.txt_bill_area1.delete('1.0', END)
        self.cartTitle.config(text=f"Cart \t Total Product: [0]")
        self.var_search.set('')
        self.clear_cart()
        self.show()
        self.show_cart()
        self.chk_print = 0

    def date_time(self):
        time_ = time.strftime("%I:%M:%S")
        date_ = time.strftime("%d-%m:%Y")
        self.lbl_clock.config(text=f"Welcome To Inventory Management
System\t\t Date: {str(date_)}\t\t Time: {str(time_)}", font=(
            "times new roman", 15), bg="black", fg="white")
        self.lbl_clock.after(200, self.date_time)

    def print_bill(self):
        if self.chk_print == 1:
            messagebox.showinfo(
                'Print', 'Please wait while printing', parent=self.root)
            new_file = tempfile.mktemp('.txt')
            open(new_file, 'w').write(self.txt_bill_area1.get(1.0, END))
            os.startfile(new_file, 'print')
        else:
            messagebox.showerror(
                'Print', 'Please Generate bill to print the receipt',
parent=self.root)

    def logout(self):
        self.root.destroy()
        os.system("login.py")


if __name__ == "__main__":
    root = Tk()
    obj = BI(root)
    root.mainloop()
```

SalesFile

```
from ast import Delete

import sqlite3

from tkinter import *

from PIL import Image, ImageTk

from tkinter import ttk, messagebox

import os




class salesClass:

    def __init__(self, root):

        self.root = root

        self.root.geometry("1100x500+220+130")

        self.root.title("Inventory Management System")

        self.root.config(bg='white')

        self.root.focus_force()
        # All Variables

        self.bill_list = []

        self.var_Invoice = StringVar()


    # Title

        lbl_title = Label(self.root, text=" View Customer Bill",
font=("goudy old style", 35, "bold"),

                             bg="Dark Green", fg="White", bd=3,
relief=RIDGE).pack(side=TOP, fill=X, padx=10, pady=20)

        lbl_invoice = Label(self.root, text="Invoice No.", font=(

            "times new roman", 15), bg="white").place(x=50, y=100)

        txt_invoice = Entry(self.root, textvariable=self.var_Invoice,
font=(

            "times new roman", 15), bg="lightyellow").place(x=160,
y=100, width=180, height=28)



        btn_search = Button(self.root, text="Search",
command=self.search, font=("times new roman", 15, "bold"),

                               bg="#2196f3", fg="white",
cursor="hand2").place(x=360, y=100, width=120, height=28)

        btn_clear = Button(self.root, text="Clear", command=self.clear,
font=("times new roman", 15, "bold"),

                              bg="lightgrey", cursor="hand2").place(x=490,
y=100, width=120, height=28)


    # Bill List

        sales_Frame = Frame(self.root, bd=3, relief=RIDGE)

        sales_Frame.place(x=50, y=140, width=200, height=330)


        scrolly = Scrollbar(sales_Frame, orient=VERTICAL)

        self.Sales_List = Listbox(sales_Frame, font=(
            "goudy old style", 15), bg="white",
yscrollcommand=scrolly.set)

        scrolly.pack(side=RIGHT, fill=Y)

        scrolly.config(command=self.Sales_List.yview)

        self.Sales_List.pack(fill=BOTH, expand=1)

        self.Sales_List.bind("<ButtonRelease-1>", self.get_data)


    # Bill Area

        bill_Frame = Frame(self.root, bd=3, relief=RIDGE)

        bill_Frame.place(x=280, y=140, width=410, height=330)


        lbl_title2 = Label(bill_Frame, text="Customer Bill Area", font=(

            "goudy old style", 20, "bold"),
bg="lightblue").pack(side=TOP, fill=X)


        scrolly2 = Scrollbar(bill_Frame, orient=VERTICAL)

        self.bill_area = Text(bill_Frame, font=(

            "goudy old style", 15), bg="lightyellow",
yscrollcommand=scrolly2.set)

        scrolly2.pack(side=RIGHT, fill=Y)

        scrolly2.config(command=self.bill_area.yview)

        self.bill_area.pack(fill=BOTH, expand=1)


    # Images

        self.BillPhoto = Image.open("images/pic.jpg")

        self.BillPhoto = self.BillPhoto.resize((380, 370),
Image.ANTIALIAS)

        self.BillPhoto = ImageTk.PhotoImage(self.BillPhoto)


        lbl_image = Label(self.root, image=self.BillPhoto, bd=0)
        lbl_image.place(x=700, y=100)

        self.show()

# _____________

    def show(self):
        del self.bill_list[:]
        self.Sales_List.delete(0, END)
        for i in os.listdir('bill'):
            if i.split('.')[-1] == 'txt':
                self.Sales_List.insert(END, i)
                self.bill_list.append(i.split('.')[0])

    def get_data(self, ev):
        index_ = self.Sales_List.curselection()
        file_name = self.Sales_List.get(index_)
        self.bill_area.delete('1.0', END)
        fp = open(f'bill/{file_name}', 'r')
        for i in fp:
            self.bill_area.insert(END, i)
        fp.close()

    def search(self):
        if self.var_Invoice.get() == "":
            messagebox.showerror(
                "Error", "Invoice No. is required", parent=self.root)
        else:
            if self.var_Invoice.get() in self.bill_list:
                fp = open(f'bill/{self.var_Invoice.get()}.txt', 'r')
                self.bill_area.delete('1.0', END)
                for i in fp:
                    self.bill_area.insert(END, i)
                fp.close()
            else:
                messagebox.showerror(
                    "Error", " Invalid Invoice No.", parent=self.root)

    def clear(self):
        self.show()
        self.var_Invoice.set("")
        self.bill_area.delete('1.0', END)


if __name__ == "__main__":
    root = Tk()
    obj = salesClass(root)
    root.mainloop()
```

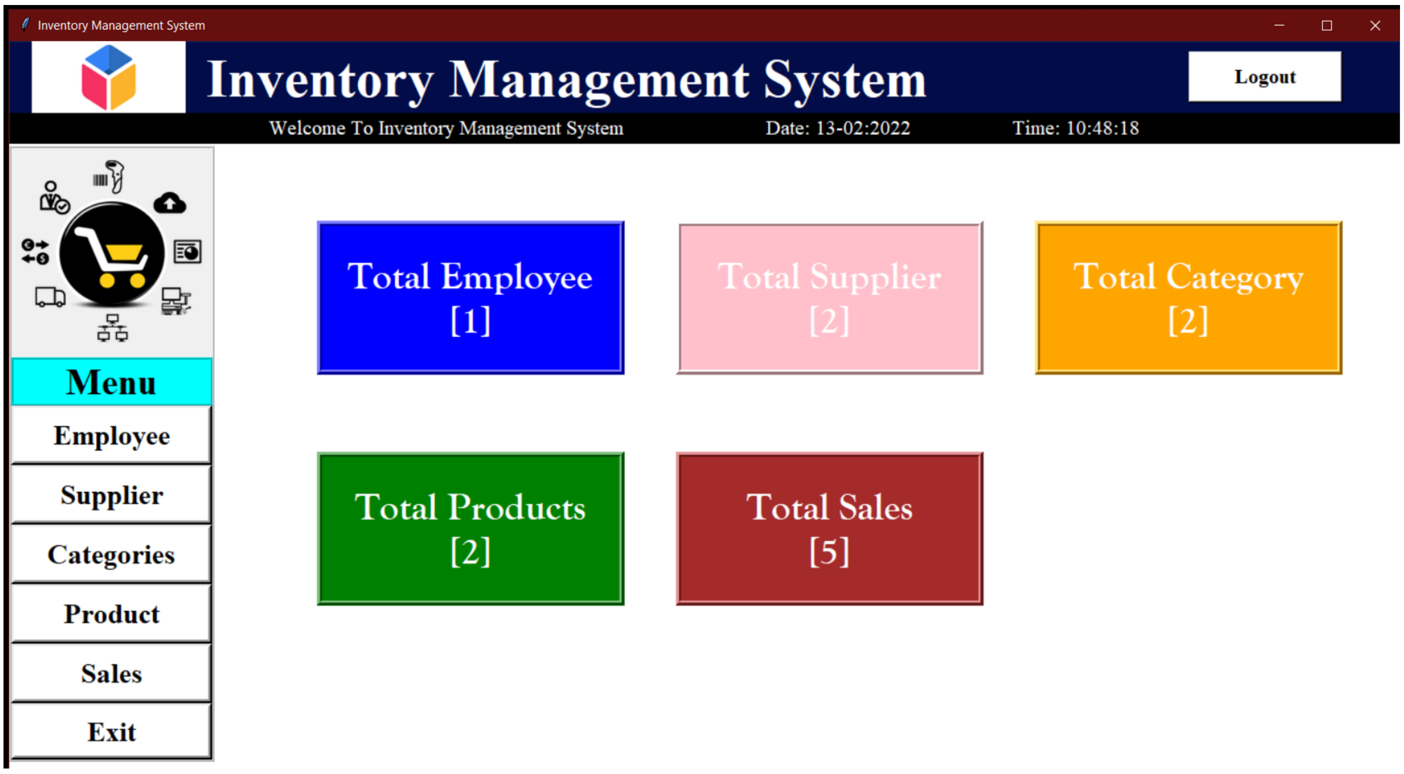
LoginFile

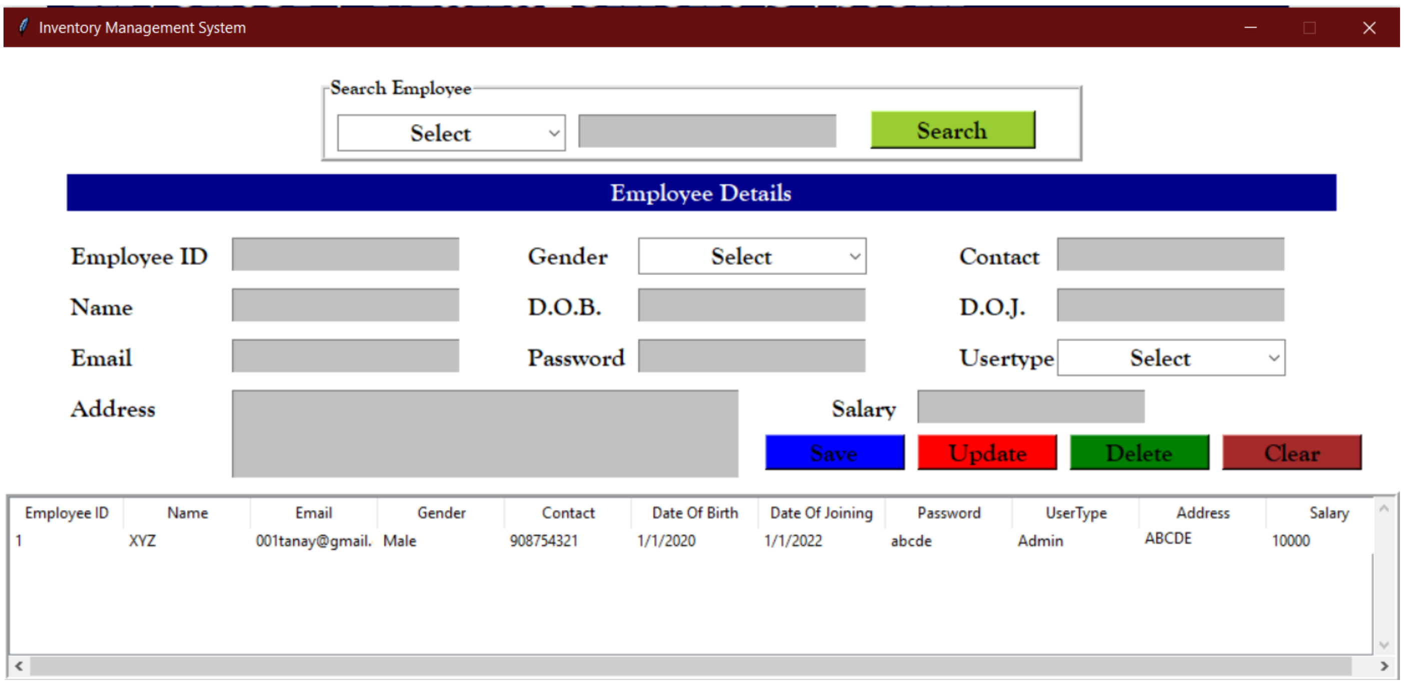
```
import email
from tkinter import*
from tkinter import messagebox
import sqlite3
import os
import pass_email
```

```
from PIL import Image, ImageTk

import smtplib #pip install smtplib

import time



class Login_System:

    def __init__(self, root):

        self.root = root

        self.root.title("Login System")

        self.root.geometry("1350x700+0+0")

        self.root.config(bg="#fafafa")


        self.otp=''


        #images

        self.laptop=ImageTk.PhotoImage(file="images/laptop.jpg")

self.lbl_laptop=Label(self.root,image=self.laptop,bd=0).place(x=0,y=0)



        # Login_Frame

        self.emp_ID = StringVar()

        self.password = StringVar()


        login_Frame = Frame(self.root, bd=2, relief=RIDGE, bg="white")

        login_Frame.place(x=950, y=110, width=350, height=460)


        title = Label(login_Frame, text="Login System",
                      font=("Elephant", 30, "bold"),
bg="white").place(x=0, y=30, relwidth=1)


        lbl_emp_ID = Label(login_Frame, text="Employee ID", font=(

            "goudy old style", 15), bg="white",
fg="#767171").place(x=25, y=110)


        txt_emp_ID = Entry(login_Frame, textvariable=self.emp_ID,
font=("times new roman", 15),

                           bg="#ECECEC").place(x=50, y=140, width=250)


        lbl_pass = Label(login_Frame, text="Password", font=(

            "goudy old style", 15), bg="white",
fg="#767171").place(x=25, y=210)

        txt_pass = Entry(login_Frame, textvariable=self.password,
show='*', font=("times new roman", 15),

                         bg="#ECECEC").place(x=50, y=240, width=250)


        btn_login = Button(login_Frame, command=self.login, text="Log
In", font=(

            "times new roman", 17,'bold'), bg="#237cdb",
activebackground="#237cdb", fg="white", activeforeground="white",
cursor="hand2").place(x=50, y=300, width=250, height=35)


        hr = Label(login_Frame, bg='lightgrey').place(

            x=50, y=370, width=250, height=2)

        or_ = Label(login_Frame, text='OR', bg='white', fg="lightgrey",
font=("times new roman", 15, "bold")).place(

            x=150, y=355)


        btn_forget = Button(login_Frame, text="Forgot Password?",
command=self.forget_p, font=(
            "times new roman", 13), bg="white", fg="#1650d9", bd=0,
activebackground="white", activeforeground="#1650d9").place(x=100,
y=390)


        # Animation Images

        self.im1 = PhotoImage(file="images\login7.png")

        self.im2 = PhotoImage(file="images\image1.png")

        self.im3 = PhotoImage(file="images\login3.png")


        self.lbl_change_image = Label(self.root, bg="white")

        self.lbl_change_image.place(x=250, y=150, width=520, height=380)


        self.animate()


# All Functions


    def animate(self):

        self.im = self.im1

        self.im1 = self.im2

        self.im2 = self.im3

        self.im3 = self.im

        self.lbl_change_image.config(image=self.im)

        self.lbl_change_image.after(2000, self.animate)


    def login(self):

        con = sqlite3.connect(database=r"project.db")

        cur = con.cursor()

        try:

            if self.emp_ID.get() == "" or self.password.get() == "":
                messagebox.showerror(
                    'Error', "All fields are required",
parent=self.root)
            else:
                cur.execute(
                    "select UserType from employee where EmpID=? and
Password=?", (self.emp_ID.get(), self.password.get()))
                user = cur.fetchone()
                if user == None:
                    messagebox.showerror(
                        'Error', "Invalid USERNAME/PASSWORD",
parent=self.root)
                else:
                    if user[0] == "Admin":
                        self.root.destroy()
                        os.system(" dashboard.py")
                    else:
                        self.root.destroy()
                        os.system("dashboard.py")


        except Exception as ex:
            messagebox.showerror("Error", f"Error due to : {str(ex)}")


    def forget_p(self):
        con = sqlite3.connect(database=r"project.db")
        cur = con.cursor()
        try:
            if self.emp_ID.get() == "":
                messagebox.showerror(
                    'Error', "Employee Id must be required",
parent=self.root)
            else:
                cur.execute(
                    "select Email from employee where EmpID=?",
(self.emp_ID.get()))
                email = cur.fetchone()
                if email == None:
                    messagebox.showerror(
                        'Error', "Invalid Employee ID, Try again",
parent=self.root)
                else:
                    # ___Forget Window
                    self.var_otp = StringVar()
                    self.var_new_pass = StringVar()
                    self.var_conf_pass = StringVar()
                    # call send_email_function()
                    check=self.send_email(email[0])
                    if check=='f':
                        messagebox.showerror("Error","Connection Error,
Try Again",parent=self.root)
                    else:


                        self.forget_win = Toplevel(self.root)
                        self.forget_win.title('RESET PASSWORD')
                        self.forget_win.geometry('400x350+500+100')
                        self.forget_win.focus_force()


                        title = Label(self.forget_win, text='Reset
Password', font=(
                                    'goudy old style', 15, 'bold'),
bg="#3f51b5", fg="white").pack(side=TOP, fill=X)

                        lbl_reset = Label(self.forget_win, text="Enter
OTP Sent on Regisitered Email", font=(

                            'times new roman', 15)).place(x=20, y=60)

                        txt_reset = Entry(self.forget_win,
textvariable=self.var_otp, font=(

                            'times new roman', 15),
bg='lightyellow').place(x=20, y=100, width=250, height=30)

                        self.btn_reset = Button(self.forget_win,
text='SUBMIT',command=self.validate_otp, font=(

                            'times new roman', 15), bg='lightblue')

                        self.btn_reset.place(x=280, y=100, width=100,
height=30)



                        lbl_new_pass = Label(self.forget_win, text="New
Password", font=(

                            'times new roman', 15)).place(x=20, y=160)

                        txt_new_pass = Entry(self.forget_win,
textvariable=self.var_new_pass, font=(

                            'times new roman', 15),
bg='lightyellow').place(x=20, y=190, width=250, height=30)



                        lbl_c_pass = Label(self.forget_win,
text="Confirm Password", font=(

                            'times new roman', 15)).place(x=20, y=225)

                        txt_c_pass = Entry(self.forget_win,
textvariable=self.var_conf_pass, font=(

                            'times new roman', 15),
bg='lightyellow').place(x=20, y=255, width=250, height=30)

                        self.btn_update = Button(self.forget_win,
text='UPDATE', command=self.update_password,state=DISABLED, font=(

                            'times new roman', 15), bg='lightblue')
                        self.btn_update.place(x=150, y=300, width=100,
height=30)


        except Exception as ex:

            messagebox.showerror("Error", f"Error due to : {str(ex)}")


    def update_password(self):

        if self.var_new_pass.get()=="" or self.var_conf_pass.get()=="":

            messagebox.showerror("Error, Password Is
Required",parent=self.forget_win)

        elif self.var_new_pass.get()!=  self.var_conf_pass.get():

            messagebox.showerror("Error, New Password & Confirm Password
Must Be Same",parent=self.forget_win)

        else:

            con = sqlite3.connect(database=r"project.db")

            cur = con.cursor()

            try:

                cur.execute("Update employee SET Password=? where
EmpID=?",(self.var_new_pass.get(),self.emp_ID.get()))

                con.commit()

                messagebox.showinfo("Success","Password Updated
Succesfully",parent=self.forget_win)

                self.forget_win.destroy()

            except Exception as ex:

                messagebox.showerror("Error", f"Error due to :
{str(ex)}")




    def validate_otp(self):

        if int(self.otp)==int(self.var_otp.get()):

            self.btn_update.config(state=NORMAL)
            self.btn_reset.config(state=DISABLED)
        else:
            messagebox.showerror("Error","INVALID OTP, Try
Again",parent=self.forget_win)


    def send_email(self,to_):
        s=smtplib.SMTP('smtp.gmail.com',587)
        s.starttls()
        email_=pass_email.email_
        password_=pass_email.password_


        s.login(email_,password_)
        self.otp=str(time.strftime("%H%M%S"))+str(time.strftime('%S'))


        subject="Inventory Management System Password Reset"
        message=f"Dear Sir/Ma'am, \n\n Password Reset OTP
:{str(self.otp)}.\n\n "
        message="subject:{}\n\n{}".format(subject,message)
        s.sendmail(email_,to_,message)
        check=s.ehlo()
        if check[0]==250:
            return 's'
        else:
            return 'f'




root = Tk()
obj = Login_System(root)
```

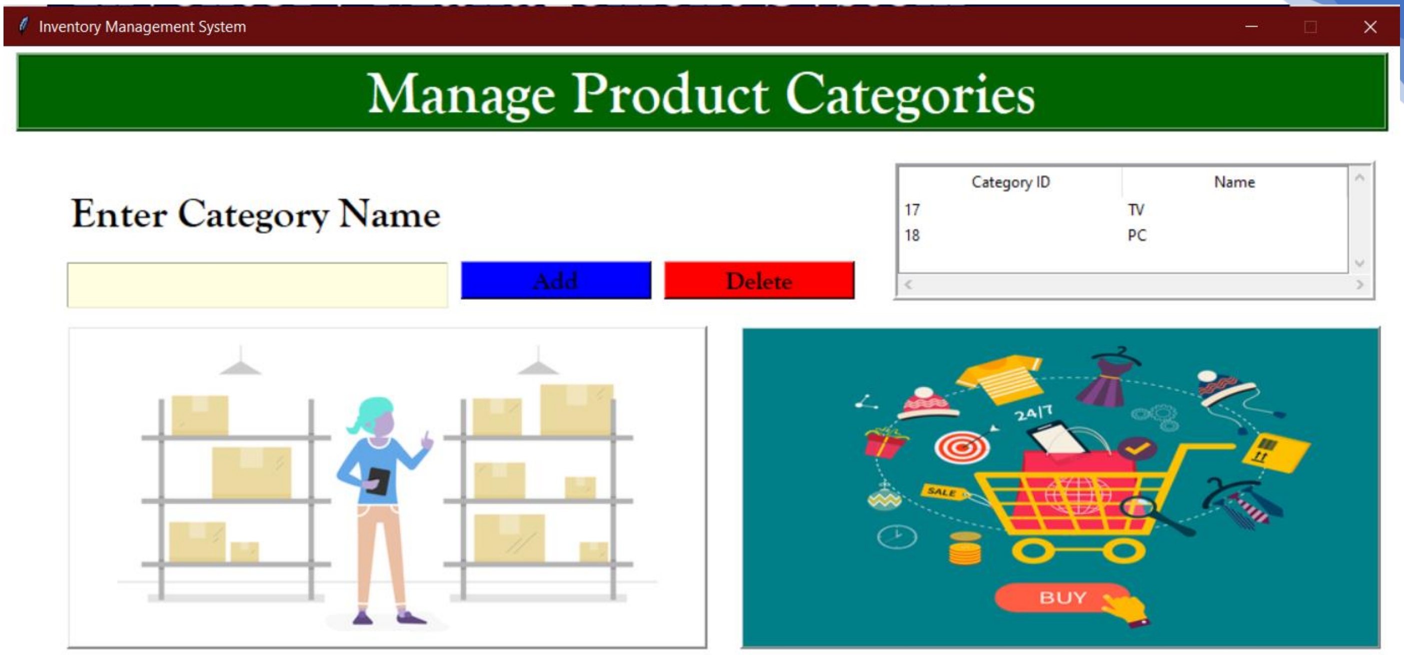
```
root.mainloop()



Pass_emailFile
email_="001tanay@gmail.com"
password_='ifgsxodairrunxts'
```

OUTPUT





Employee ID	Name	Email	Gender	Contact	Date Of Birth	Date Of Joining	Password	UserType	Address	Salary
1	XYZ	001tanay@gmail.	Male	908754321	1/1/2020	1/1/2022	abcde	Admin	ABCDE	10000

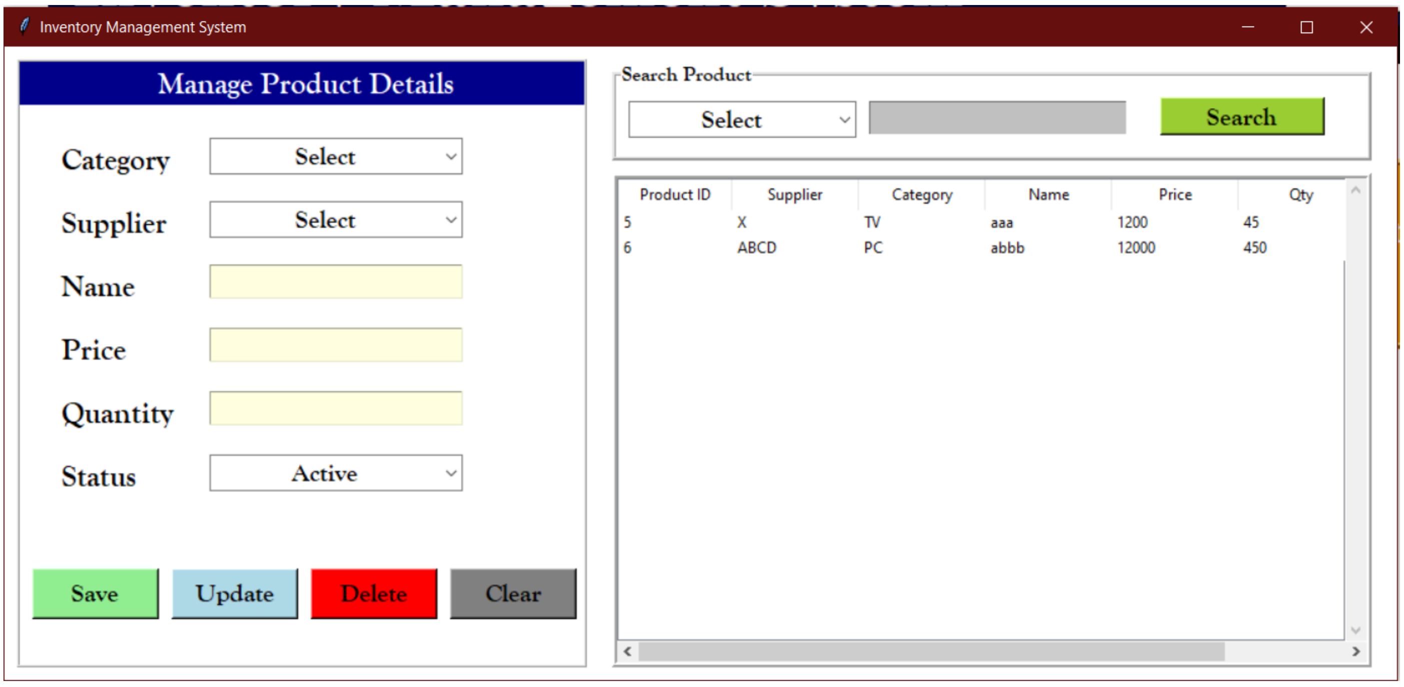


Inventory Management System

Manage Product Categories

Enter Category Name

Category ID	Name
17	TV
18	PC

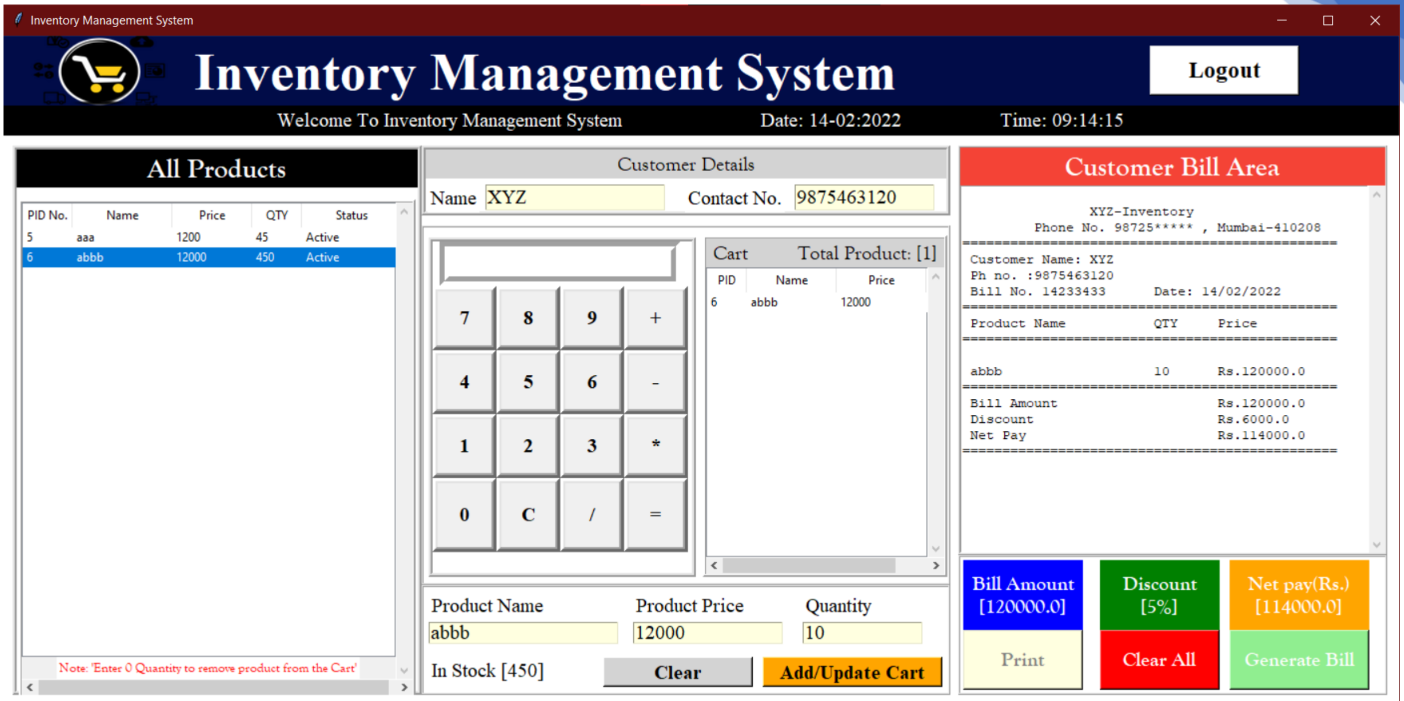


Inventory Management System

Manage Product Details

Category	Select
Supplier	Select
Name	
Price	
Quantity	
Status	Active

Product ID	Supplier	Category	Name	Price	Qty
5	X	TV	aaa	1200	45
6	ABCD	PC	abbb	12000	450



Date: 14-02:2022 Time: 09:14:15

PID No.	Name	Price	QTY	Status
5	aaa	1200	45	Active
6	abbb	12000	450	Active

Note: 'Enter 0 Quantity to remove product from the Cart'

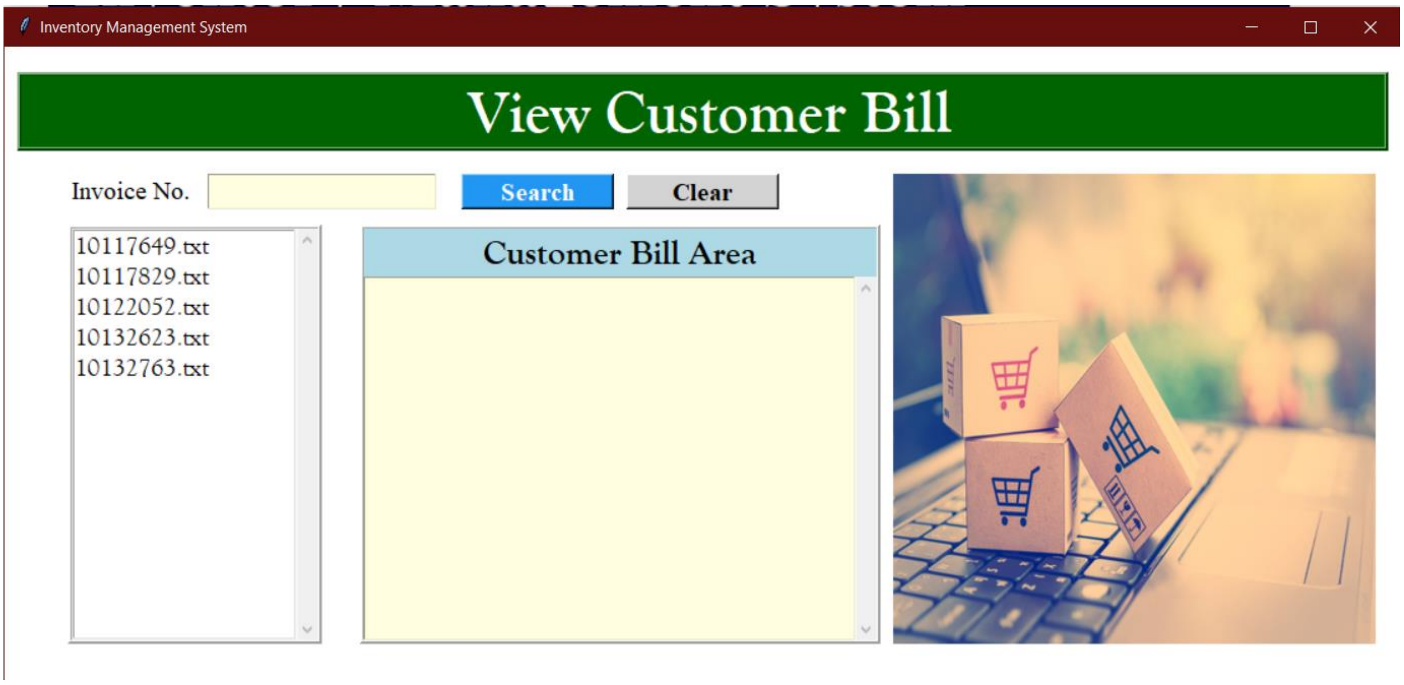
Name XYZ Contact No. 9875463120

Cart Total Product: [1]

PID	Name	Price
6	abbb	12000

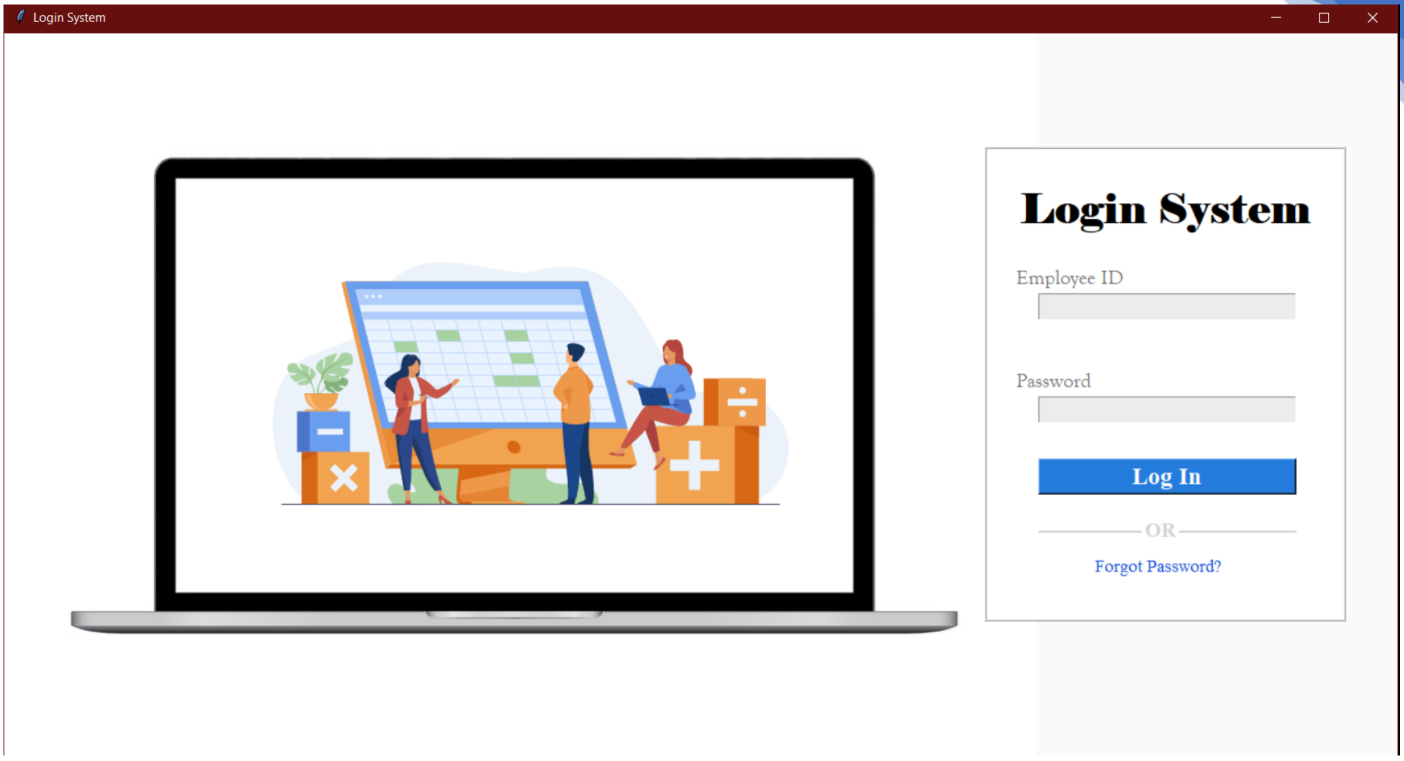
Product Name: abbb
Product Price: 12000
Quantity: 10
In Stock [450]

```
               XYZ-Inventory
       Phone No. 98725***** , Mumbai-410208
===============================================
 Customer Name: XYZ
 Ph no. :9875463120
 Bill No. 14233433      Date: 14/02/2022
===============================================
 Product Name           QTY     Price
===============================================


 abbb                   10      Rs.120000.0
===============================================
 Bill Amount                    Rs.120000.0
 Discount                       Rs.6000.0
 Net Pay                        Rs.114000.0
===============================================
```



Invoice No.

10117649.txt
10117829.txt
10122052.txt
10132623.txt
10132763.txt


Login System

Login System

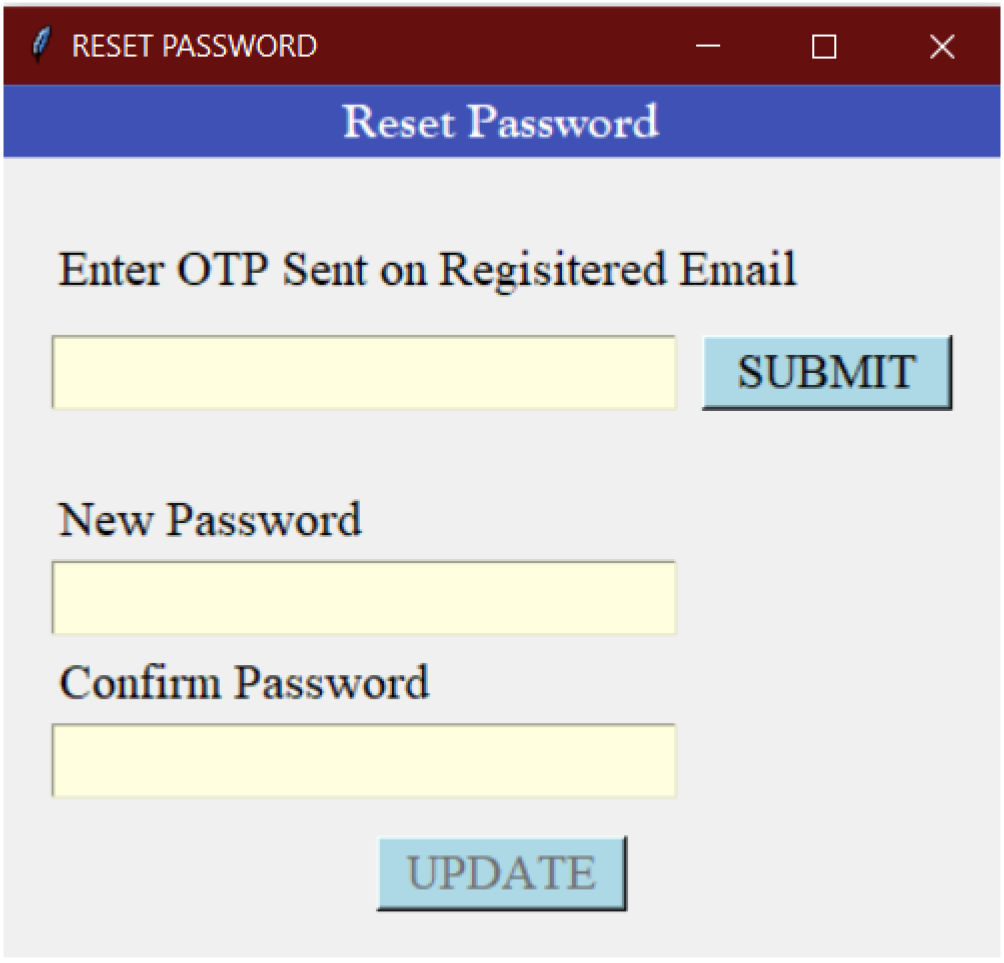
Employee ID

Password

Log In

OR

Forgot Password?


RESET PASSWORD

Reset Password

Enter OTP Sent on Regisitered Email

SUBMIT

New Password

Confirm Password

UPDATE

LIMITATIONS AND FUTURE SCOPES

The limitations of the system include not knowing an exact inventory count in the middle of the period and running the risk of stockouts. With this system, the company knows the inventory level with certainty only when it physically counts the inventory at the end of each period.

It also includes a false sense of reliability and dependence on human entry. The company remains unaware of the theft or waste, known as shrinkage, until it performs a physical count at least once per year. The other limitation is that an employee might enter data incorrectly, introducing inaccurate information that can compromise decision-making.

Regardless of which type of inventory system a company uses, the scope of the inventory may change based on the strategic targets of the business. Scope may refer to different aspects of how inventory counts are conducted or to the way inventory information is used.

The value of the inventory at the end of each period provides a basis for financial reporting on the balance sheet. Measuring the change in inventory allows the company to determine the cost of inventory sold during the period. This allows the company to plan for future inventory needs.

The importance of inventory counts in those examples may require having staff who are dedicated to inventory management. On the other hand, a small cleaning business may not need more than an occasional and rather informal scan of its cleaning supplies inventory to function efficiently.

BIBLIOGRAPHY

COMPUTER SCIENCE WITH PYTHON

XII (By: SUMITA ARORA).

2. COMPUTER SCIENCE WITH PYTHON

XII (By: PREETI ARORA).

3. NCERT Computer Science for Class 12-

Latest edition as per NCERT/CBSE

4. www.google.com

5. www.google.com/Python project

6. www.data.world

7. www.youTube.com

8. Class notes.