

Inventory Management System

Kshitiz Saxena | XII S2 | 16 |

Levin Varghese | XII S2 | 18 |

Tanay Koli | XII S2 | 33 |

# 

# 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. Kanchan S. Manuja 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

|  |  |  |
| --- | --- | --- |
| SR.  NO. | CONTENTS | PAGE  NO. |
| 1 | COVER PAGE | 1 |
| 2 | CERTIFICATE | 2 |
| 3 | ACKNOWLEGEMENT | 3 |
| 4 | INDEX | 4 |
| 5 | INTRODUCTION TO PYTHON AND MYSQL | 5 |
| 6 | OVERVIEW OF PROJECT | 8 |
| 7 | SYSTEM DESIGN | 9 |
| 8 | SOURCE CODE | 10 |
| 9 | OUTPUT | 82 |
| 10 | LIMITATIONS AND FUTURE SCOPES | 86 |
| 11 | BIBLIOGRAPHY | 87 |

# 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.

1. 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.
2. Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
3. Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
4. 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 −

1. Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
2. Easy-to-read − Python code is more clearly defined and visible to the eyes.
3. Easy-to-maintain − Python's source code is fairly easy-to-maintained.
4. A broad standard library − Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Macintosh.
5. Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
6. Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
7. 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.
8. Databases − Python provides interfaces to all major commercial databases.
9. 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.
10. 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 −

1. It supports functional and structured programming methods as well as OOP.
2. It can be used as a scripting language or can be compiled to byte-code for building
3. large applications.
4. It provides very high-level dynamic data types and supports dynamic type checking.
5. It supports automatic garbage collection.
6. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

# INTRODUCTION TO SQLite3

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

2000 - D. Richard Hipp designed SQLite for the purpose of no administration required for operating a program.

## SQLite3 Features

1. SQLite does not require a separate server process or system to operate (serverless).
2. SQLite comes with zero-configuration, which means no setup or administration needed.
3. A complete SQLite database is stored in a single cross-platform disk file.
4. SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
5. SQLite is self-contained, which means no external dependencies.
6. SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
7. SQLite supports most of the query language features found in SQL92 (SQL2) standard.
8. SQLite is written in ANSI-C and provides simple and easy-to-use API.
9. SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

## 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:**

1. Maintaining and recording the information between too much and too little inventory in the company.
2. Keep track of inventories as it is transported between different locations.
3. Recording product information in a warehouse or other location.
4. Having a record of Picking, packing, and selling products from a warehouse.
5. Reduction of product obsolescence and decay.
6. Avoiding out-of-stock situations

# SYSTEM DESIGN

1. **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.
2. **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.
3. **Employee Page:** From the admins can change information like addition of employees, correction in name, email id, address, etc.
4. **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.
5. **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.
6. **Products Page:** The user can add new items using this page. While adding the items to the database user provides an item description.
7. **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.
8. **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()

EmployeeFile

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,values=("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=("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)

#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,values=("Select","Admin",'Employee'),state="readonly",justify=CENTER,font=("goudy 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=("goudy 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=("goudy old style",15,'bold'),bg="green",fg="black",cursor="hand2").place(x=840,y=305,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.set)

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,Salary) 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=?,Address=?,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()

SupplierFile

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=("goudy old style",15),bg="gold",fg="black",cursor="hand2").place(x=990,y=79,width=100,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=("goudy 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=("goudy old style",15,'bold'),bg="green",fg="black",cursor="hand2").place(x=420,y=345,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=("goudy 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","Price","QTY","Status"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.set)

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()

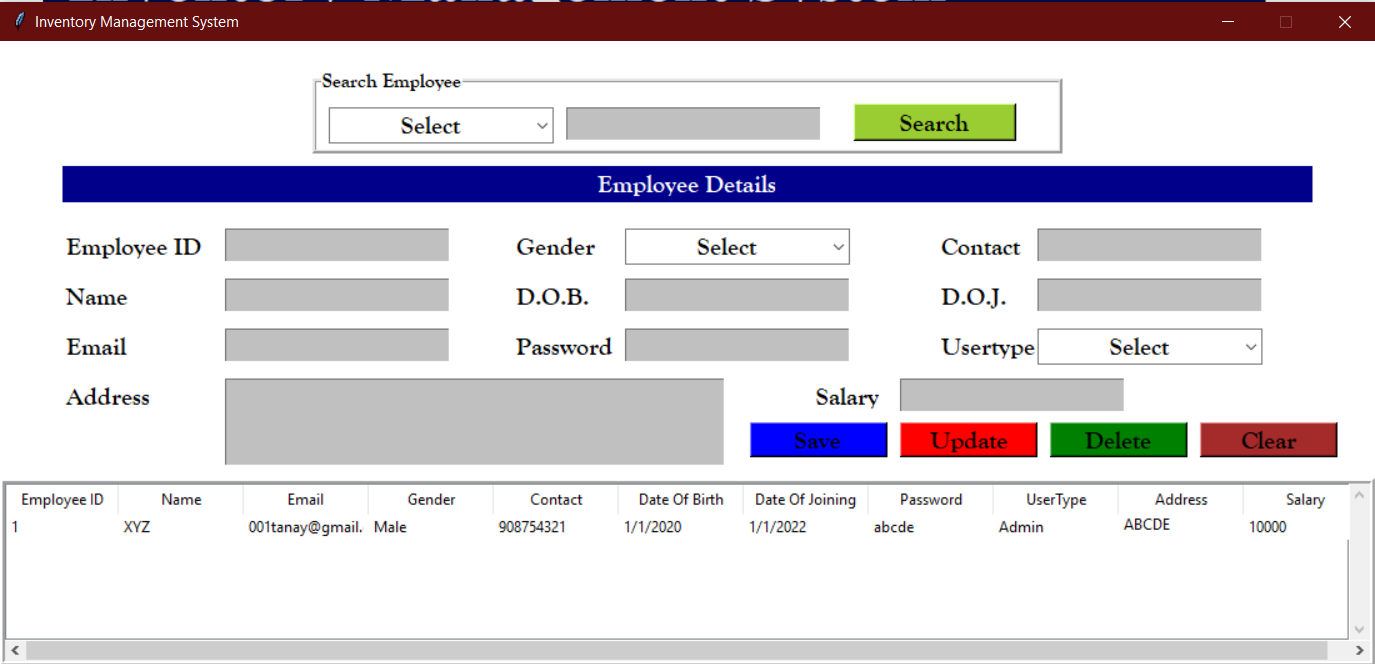
Pass\_emailFile

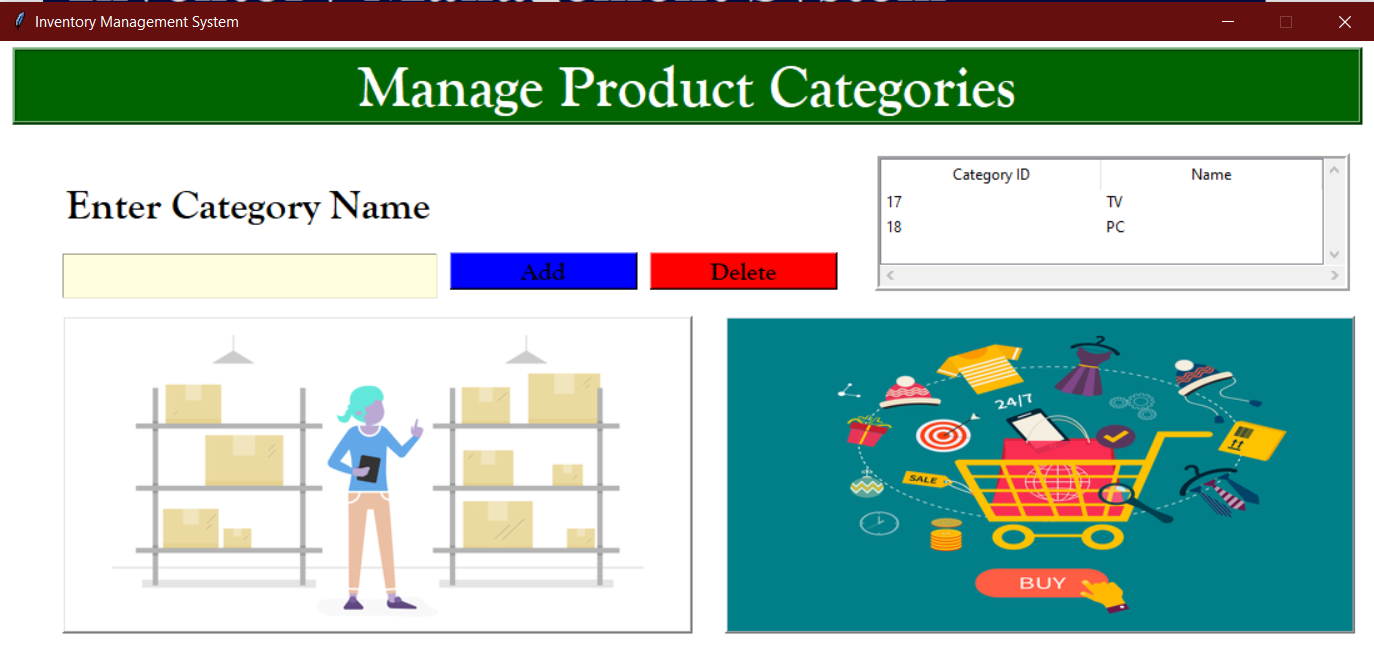
email\_="001tanay@gmail.com"

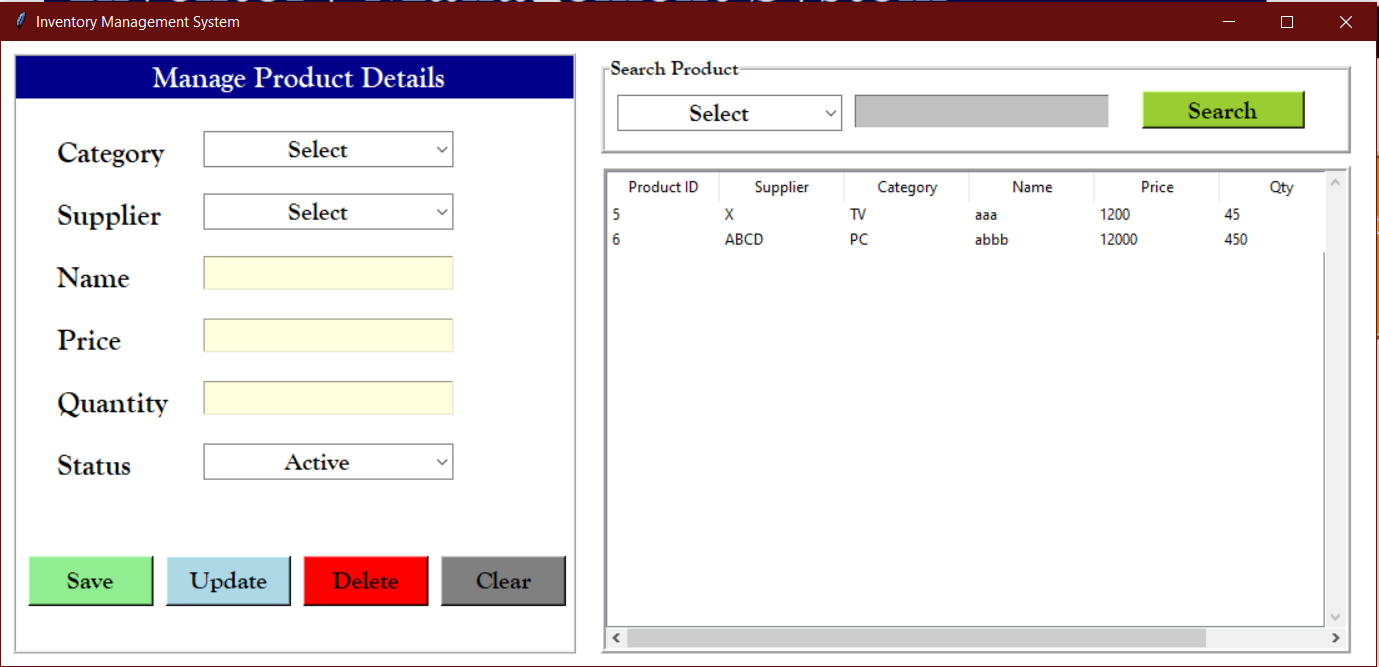
password\_='ifgsxodairrunxts'

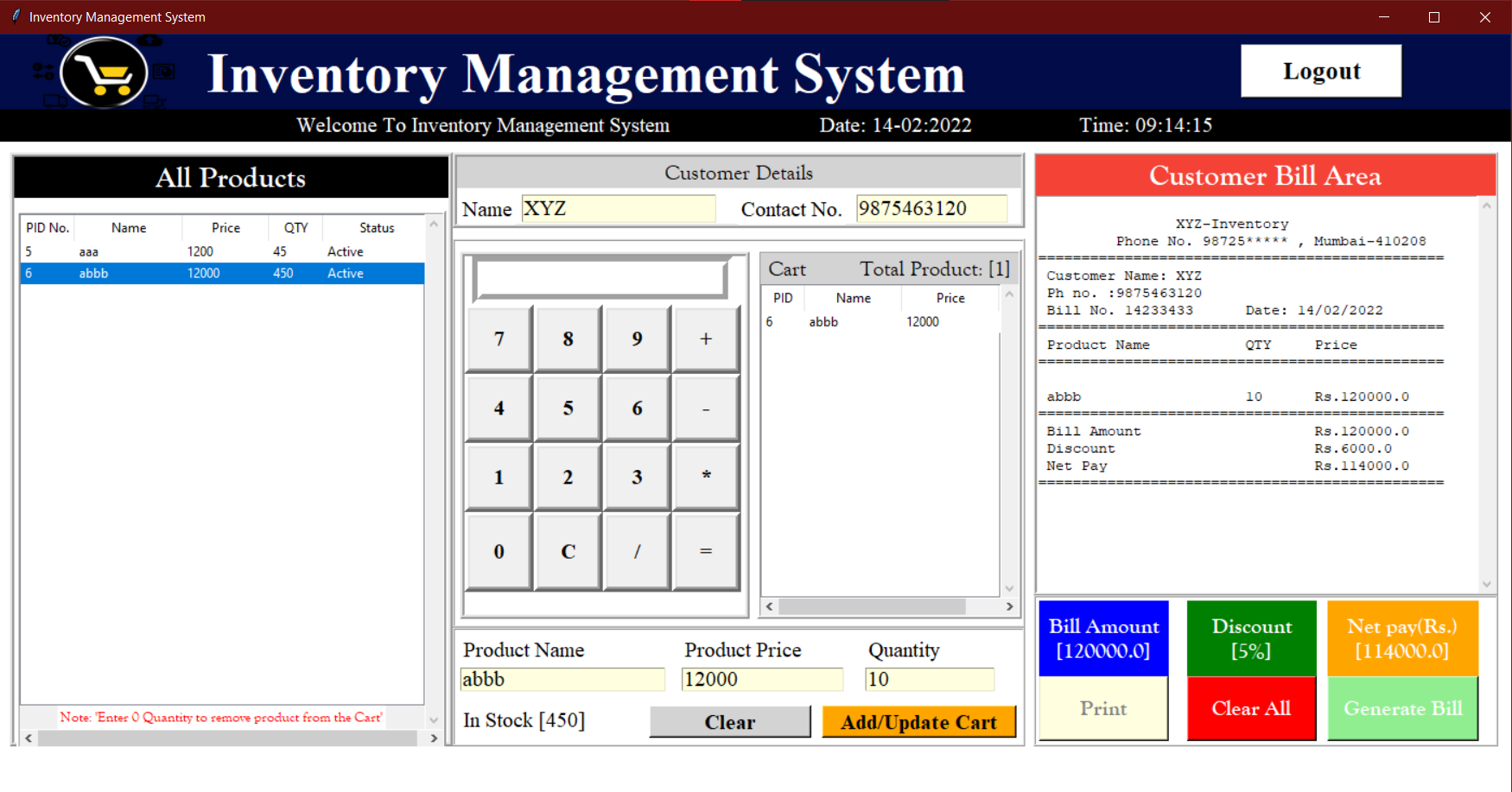
# OUTPUT



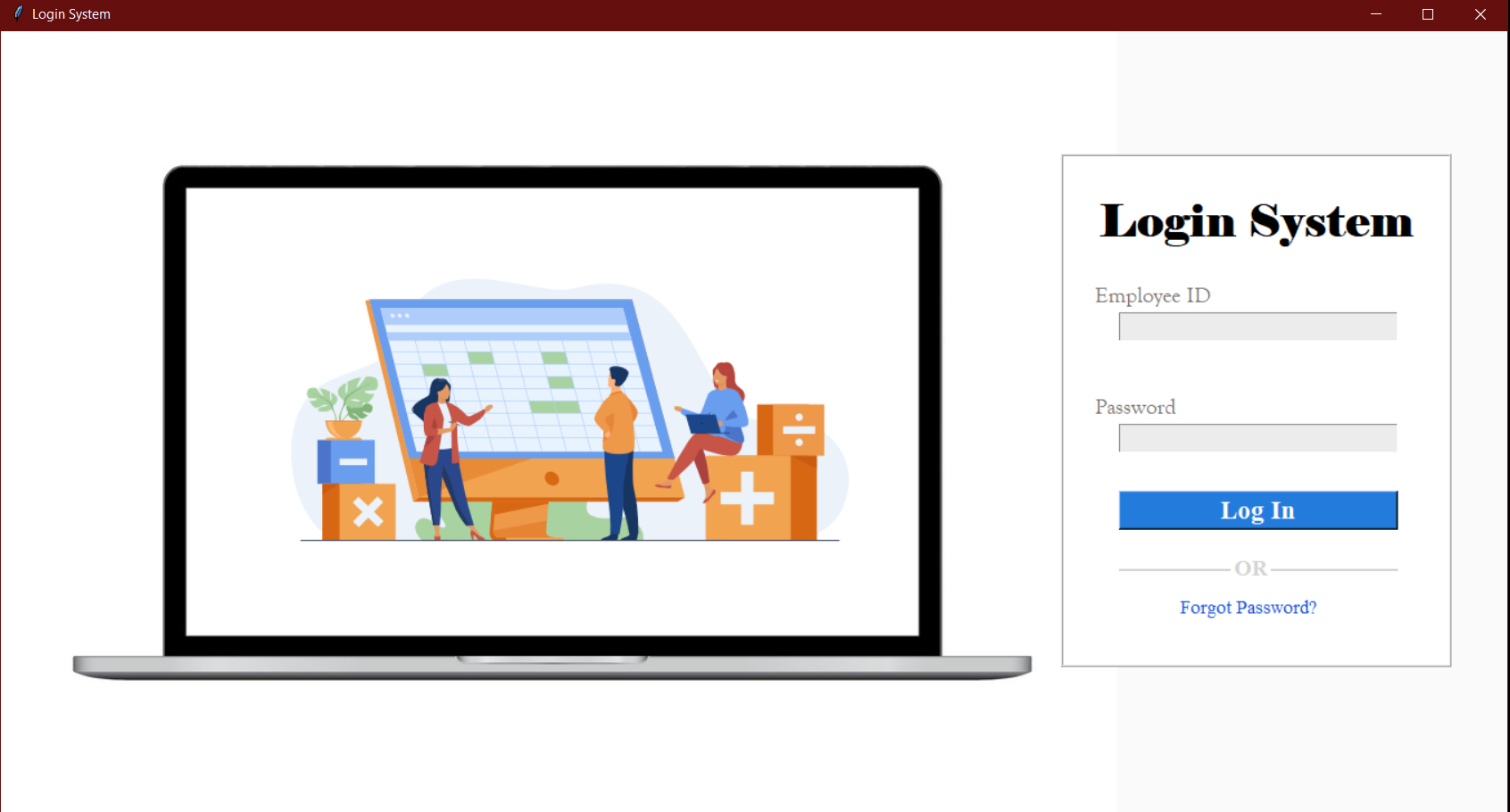


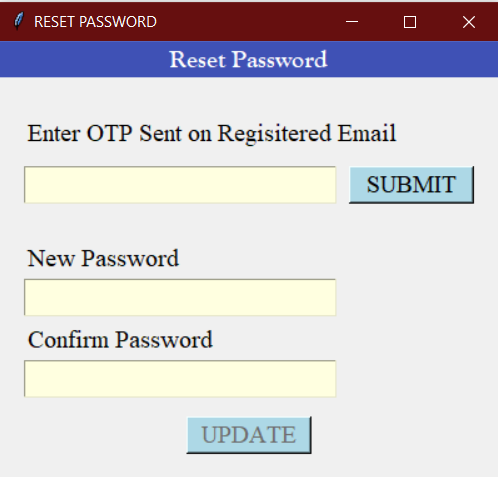












# LIMITATIONS AND FUTURE SCOPES

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.