AMRITSAR GROUP OF COLLEGES

Autonomous status conferred by UGC under UGC act-1956, (2f), NAAC-A Grade,
(Formerly Known as Amritsar College of Engineering & Technology | Amritsar Pharmacy College)

Project Report

On

"INVENTRY MANAGEMENT SYSTEM"

Submitted in the Partial fulfilment of the requirement for the Award of Degree of
Bachelor of Technology

In

COMPUTER SCIENCE & ENGINEERING (2020-24)

SUBMITTED TO

Er. Ajay Sharma (Associate Professor)

Er. Neha Chadha (Assistant Professor)

SUBMITTED BY

Sanjay Kumar Sah(2000193)

Shahil Kumar (20000196)

Siddharth Kumar Sonu (2000206)

Sumit Kumar Giri (2000213)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Amritsar Group of Colleges, Amritsar

ACKNOWLEDGEMENT

It is our proud privilege to release the feelings of our gratitude to several persons who helped us directly or indirectly to conduct this Analysis Project Work. We express our heart full thanks and owe a deep sense of gratitude to our teacher and my faculty guide Er. Ajay Sharma, Associate Professor, Department of CSE, **Amritsar Group of Colleges**, Amritsar, for his guidance and inspiration in completing this project.

We are extremely thankful to the Dr. Sandeep Kad, Head of Department and all faculty members of CSE Department at **Amritsar Group of Colleges**, Amritsar for their co-ordination and co-operation and for their kind guidance and encouragement.

We also thank all our friends who have more or less contributed to the preparation of this Project Report, we will be always indebted to them.

This project completion has indeed helped us explore more knowledge avenues related to RDBMS/Python and we are sure it will help us in future too.

DECLARATION

We Sanjay, Shahil , Siddharth and Sumit hereby as a team declare that the project work entitled **"INVENTORY MANAGEMENT SYSTEM** is an authentic record of our own work carried out as per the requirements of RDBMS/PYTHON Labs (Part-B) for the award of degree of **B.Tech (CSE)**, **Amritsar Group of Colleges, Amritsar**, under the guidance of **Er. Ajay Sharma (Associate Professor), Er. Neha Chadha (Assistant Professor).**

Sanjay Kumar Sah (2000193)

Shahil Kumar (20000196)

Siddharth Kumar Sonu (2000206)

Sumit Kumar Giri (2000213)

Certified that the above statement made by the students is correct to the best of our knowledge and belief.

Faculty Coordinator

Er. Ajay Sharma (Associate Professor)

Er. Neha Chadha (Assistant Professor)

Introduction About Project

It maintains the information about the personal and official details of the inventory. It is developed to override the problems prevailing in the practicing manual system. Inventory Management System is a distributed application, developed to maintain the details of employees working in any organization. Project: The objective of this project is to provide a comprehensive approach towards the management of inventory information. Provides full functional reports to the management of company. To develop a well designed database to store employee information. This project aims to simplify the task of maintaining records of the employees of Company. Objective: The main objective of the application is to maintain the details of employees,supplier, The working in any organization, objective of this project is to provide a comprehensive approach towards the management of inventory

TABLE OF CONTENTS
(Change it according to your project report)

Introduction to RDBMS

- **What is RDBMS ?**

RDBMS stands for Relational Database Management System.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS.

It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

- **How it works.**

Data is represented in terms of tuples (rows) in RDBMS.

A relational database is the most commonly used database. It contains several tables, and each table has its primary key.

Due to a collection of an organized set of tables, data can be accessed easily in RDBMS.

- **Brief History of RDBMS.**

From 1970 to 1972, E.F. Codd published a paper to propose using a relational database model.

RDBMS is originally based on E.F. Codd's relational model invention.

- **What is table/Relation?**

Everything in a relational database is stored in the form of relations. The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data. Each table represents some real-world objects such as person, place, or event about which information is collected. The organized collection of data into a relational table is known as the logical view of the database.

- **Properties of a Relation:**

    - Each relation has a unique name by which it is identified in the database.
    - Relation does not contain duplicate tuples.
    - The tuples of a relation have no specific order.
    - All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

- **What is a row or record?**

    A row of a table is also called a record or tuple. It contains the specific information of each entry in the table. It is a horizontal entity in the table. For example, The above table contains 5 records.

- **Properties of a row:**

    - No two tuples are identical to each other in all their entries.
    - All tuples of the relation have the same format and the same number of entries.
    - The order of the tuple is irrelevant. They are identified by their content, not by their position.

Introduction to PYTHON

● What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python.

● Features of python:

- Easy to Learn and Use .
- Expressive Language.
- Interpreted Language.
- Cross-platform Language.
- Free and Open Source .
- Object-Oriented Language.
- Extensible.
- Large Standard Library.
- GUI Programming Supporte.
- Integrated.
- Embeddable.
- Dynamic Memory Allocation.

● Flavors of Python

- Cpython
- Jpython
- Ironpython

 PyPy

 Rubypython

 Stackless Python

 Pythonxy

 AnacondaPython

Objectives of the project

Easy management of Inventory.

Handel details of sales, purchase, balance stock .

Make Stock Manageable.

Details with day to day requirement of any production organization.

Material Availability.

Better Level of Customer Service.

Keeping Wastage and Losses to a Minimum.

Cost-Effective Storage.

Optimizing Product Sales.

Modules used in project

- **Frontend Tool Tkinter.**

We used frontend tool Tkinter to give magnificent look

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application. It is however the most commonly used one

CODING

● LOGIN PAGE:

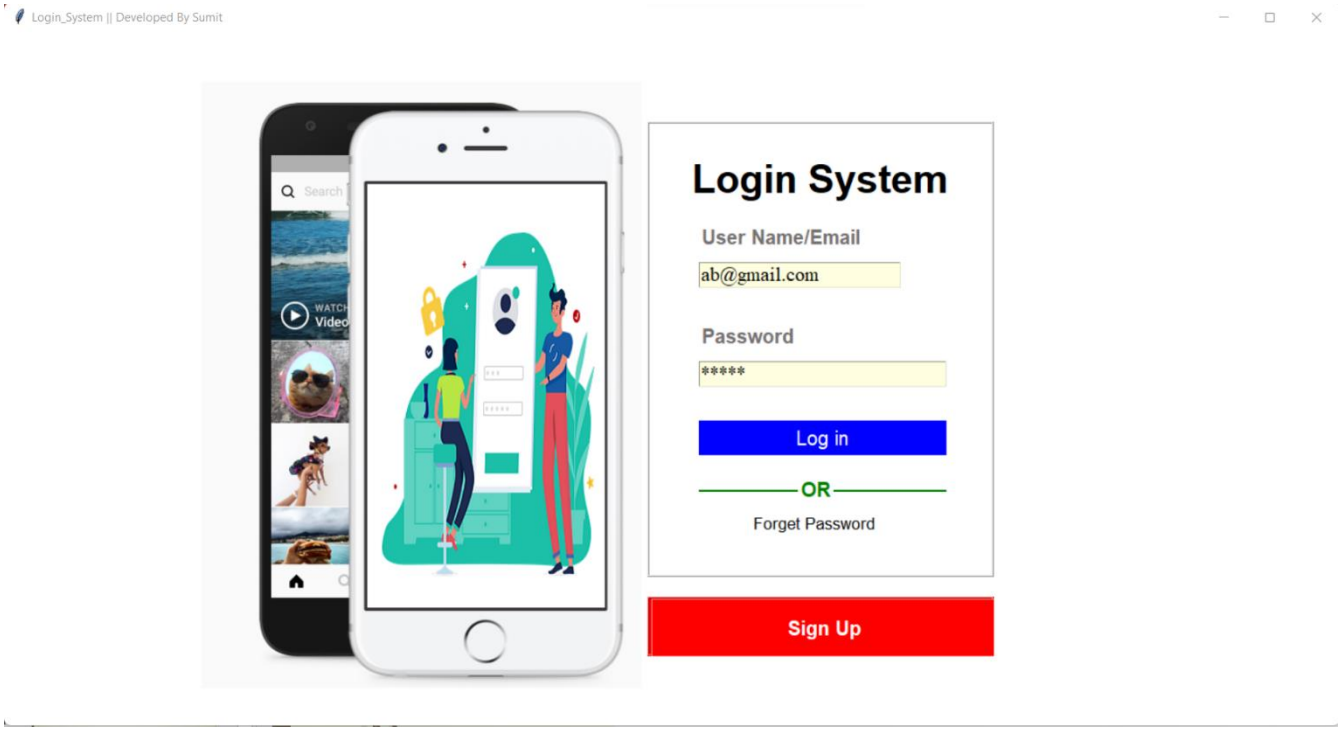
```
from tkinter import*
from tkinter import font
from tkinter import messagebox
import cx_Oracle
import os
from PIL import ImageTk
from singup import singupClass
from Dashboard import IMS
class Login_System:
    def __init__(self,root):
        self.roots=root
        self.roots.title("Login_System || Developed By Sumit")
        self.roots.geometry("1350x700+0+0")
        self.roots.config(bg="Cyan")
        #========== images =================
        self.phone_image=ImageTk.PhotoImage(file="images/phone.png")
        self.lbl_phone_image=Label(self.roots,image=self.phone_image,bd=0).place(x=200,y=50)
        self.roots.config(bg="white")
        #==========Login frame ======================
        self.employee_id=StringVar()
        self.password=StringVar()
        login_Frame=Frame(self.roots,bd=2,relief=RIDGE,bg="white")
        login_Frame.place(x=650,y=90,width=350,height=460)
```

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


    #=========employee_id ====================

    lbl_user=Label(login_Frame,text="User Name/Email",font=("time new
roman",15,"bold"),bg="white",fg="#767171").place(x=50,y=100)

    self.username=StringVar()

    self.password=StringVar()

    self.username=Entry(login_Frame,textvariable=self.username,font=("times new
roman",15),bg="lightyellow")

    self.username.place(x=50,y=140)


    lbl_pass=Label(login_Frame,text="Password",font=("time
new",15,"bold"),bg="white",fg="#767171").place(x=50,y=200)

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

    self.password.place(x=50,y=240)


    btn_login=Button(login_Frame,text="Log in",command=self.login,font=("time new
roman",15),bg="blue",activebackground="blue",fg="white",bd=0,activeforeground="white",cu
rsor="hand2").place(x=50,y=300,width=210,height=35)

    hr=Label(login_Frame,bg="green").place(x=50,y=370,width=250,height=2)

    or_=Label(login_Frame,text="OR",bg="white",fg="green",font=("time new
roman",15,"bold")).place(x=150,y=355)


    btn_forget=Button(login_Frame,text="Forget Password",font=("time new
roman",13),bg="white",fg="black",bd=0,activebackground="white",activeforeground="black"
).place(x=100,y=390)


    register_Frame=Frame(self.roots,bd=2,relief=RIDGE,bg="red")

    register_Frame.place(x=650,y=570,width=350,height=60)

    btn_singnup=Button(register_Frame,text="Sign Up",command=self.singup,font=("time new
roman",15,"bold"),bg="red",activebackground="red",fg="white",activeforeground="white",cu
rsor="hand2").place(x=2,y=0,width=350,height=60)
```

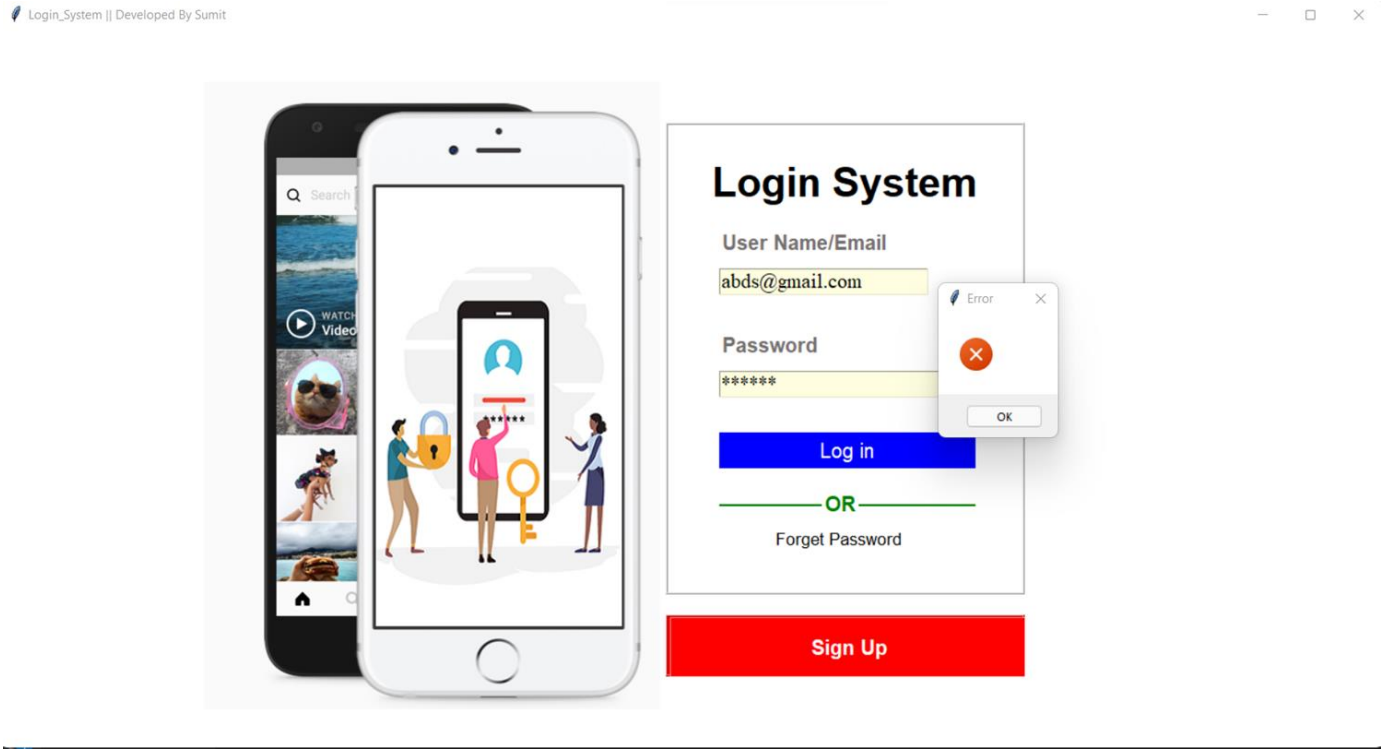
```
        self.im1=ImageTk.PhotoImage(file="images/im1.png")
        self.im2=ImageTk.PhotoImage(file="images/im2.png")
        self.im3=ImageTk.PhotoImage(file="images/im3.png")

        self.lbl_change_image=Label(self.roots,bg="white")
        self.lbl_change_image.place(x=367,y=153,width=240,height=428)
#================ animation ========================================
        self.animate()

    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 singup(self):
        self.new_win=Toplevel(self.roots)
        self.new_win=singupClass(self.new_win)

    def login(self):
        flag=0
        if self.username.get()=="" or self.password.get()=="":
            messagebox.showerror("Error!","All Field are required",parent=self.root)
        elif self.username.get()!='' and self.password.get()!='':
            connection=cx_Oracle.connect('sumit/abhinav')
            cur=connection.cursor()
            cur.execute("select * from singup")
```

```
            row=cur.fetchall()
            for i in row:
                if(self.username.get()==i[3]) and (self.password.get()==i[4]):
                    print('user has accessed the system')
                    flag=1
                    break
            if(flag==1):
                print('user has accessed the system')
                self.redirect_windows()
            else:
                messagebox.showerror("Error")

    def redirect_windows(self):
        self.roots.destroy()
        root=Tk()
        obj=IMS(root)
        root.mainloop()




#=============== forget password ==========================================
#def forget_window(self):

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

□ SNAPSHOTS OF LOGIN PAGE.



□ **If we enter wrong username and password then it will display "Invalid Username/Email and password".**



□ BACKEND OF LOGIN PAGE

```
Run SQL Command Line

SQL> conn
Enter user-name: sumit/abhinav
Connected.
SQL> select *from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADMIN                          TABLE
BIN$2Exg4bghSjWNpBHe7VlXEw==$0 TABLE
CATEGORY                       TABLE
EMPLOYEE                       TABLE
LOGIN                          TABLE
PRODUCT                        TABLE
SALES                          TABLE
SINGUP                         TABLE
SUPPLIER                       TABLE

9 rows selected.

SQL> desc login;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_ID                                        VARCHAR2(20)
 PASSWORD                                           VARCHAR2(30)
```

● CODE OF SINGUP PAGE

□ **If you are new user then Signup firstly.**

```
from tkinter import *
from tkinter import messagebox
from turtle import left, title
from PIL import Image,ImageTk #pip install pillow
import cx_Oracle

#def getvals():
```

```
      # print("Accepted")
class singupClass:
    def __init__(self,root):
        self.roots=root
        self.roots.geometry('1350x700+0+0')
        self.roots.title("singup")
        self.roots.config(bg="Cyan")
#========= singup frame ===============
        frame=Frame(self.roots,bg="orange",bd=10)
        frame.place(x=600,y=60,width=600,height=550)

        self.MenuLogo=Image.open("images/img7.png")
        self.MenuLogo=self.MenuLogo.resize((560,550),Image.ANTIALIAS)
        self.MenuLogo=ImageTk.PhotoImage(self.MenuLogo)

        lbl_menuLogo=Label(self.roots,image=self.MenuLogo)
        lbl_menuLogo.place(x=4,y=60)



        fname=StringVar()
        title=Label(frame,text="SINGUP HERE",font=("time new
roman",30,"bold"),bg="orange",fg="blue").place(x=150,y=10)
        f_name=Label(frame,text="First Name",font=("time new
roman",15,"bold"),bg="orange").place(x=50,y=80)
        self.f_name=Entry(frame,font=("time new roman",15),bg="lightyellow",textvariable=fname)
        self.f_name.place(x=20,y=120)

        lname=StringVar()
        last_name=Label(frame,text="Last Name",font=("time new
roman",15,"bold"),bg="orange").place(x=420,y=80)
        self.last_name=Entry(frame,font=("time new roman",15),bg="lightyellow",textvariable=lname)
```

```
        self.last_name.place(x=350,y=120)


        contact1=StringVar()
        contact=Label(frame,text="Contact",font=("time new
roman",15,"bold"),bg="orange").place(x=50,y=200)
        self.contact=Entry(frame,font=("time new roman",15),bg="lightyellow",textvariable=contact1)
        self.contact.place(x=20,y=240)


        email1=StringVar()
        email=Label(frame,text="Email",font=("time new
roman",15,"bold"),bg="orange").place(x=430,y=200)
        self.email=Entry(frame,font=("time new roman",15),bg="lightyellow",textvariable=email1)
        self.email.place(x=350,y=240)


        password1=StringVar()
        password=Label(frame,text="Password",font=("time new
roman",15,"bold"),bg="orange").place(x=50,y=320)
        self.password=Entry(frame,show="*",font=("time new
roman",15),bg="lightyellow",textvariable=password1)
        self.password.place(x=20,y=360)


        password2=StringVar()
        c_password=Label(frame,text="Conform Password",font=("time new
roman",15,"bold"),bg="orange").place(x=360,y=320)
        self.c_password=Entry(frame,show="*",font=("time new
roman",15),bg="lightyellow",textvariable=password2)
        self.c_password.place(x=350,y=360)



    #============== submit ============================
        btn_submit=Button(frame,text="submit",command=self.sign,font=("times new
roman",20),bg="green",fg="red",bd=5).place(x=230,y=430,height=50,width=120)
```

```
    #====================backend============================
    def sign(self):

        connection = cx_Oracle.connect('sumit/abhinav')
        cur = connection.cursor()



        f=self.f_name.get()
        l=self.last_name.get()
        c=self.contact.get()
        e=self.email.get()
        p=self.password.get()
        c=self.c_password.get()
        cur.execute("insert into singup
values(:fn,:ln,:us,:em,:pa,:c_pa)",{":fn":f,":ln":l,":us":c,":em":e,":pa":p,":c_pa":c})
        cur.close()
        connection.commit()
        connection.close()
        messagebox.showinfo("Register Successful",parent=self.roots)



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

□ SNAPSHOTS OF LOGIN PAGE.

□ BACKEND OF LOGIN PAGE

```
SQL> desc singup
 Name                                              Null?    Type
 ------------------------------------------------- -------- ------------------------------
 F_NAME                                                     VARCHAR2(20)
 LAST_NAME                                                  VARCHAR2(30)
 CONTACT                                                    VARCHAR2(40)
 EMAIL                                                      VARCHAR2(20)
 PASSWORD                                                   VARCHAR2(30)
 C_PASSWORD                                                 VARCHAR2(40)

SQL> select *from singup;

F_NAME               LAST_NAME                      CONTACT
-------------------- ------------------------------ ----------------------------------------
EMAIL                PASSWORD                       C_PASSWORD
-------------------- ------------------------------ ----------------------------------------
dsf                  afs                            ff
dsgfd                dg                             ff

abhinav              anand                          123
ab@gmail.com         123                            123
```

```
sumit                giri                              1234
sumit@gmail.com      1234                              1234

shahil               kumar                             912820
shahil123@gmail.com  912820                            912820

sanjay               sah                               15434
sanjay321@gmail.com  15434                             15434









siddharth            sah                               987546
siddharth@gmail.com  987546                            987546


16 rows selected.
```

● **CODE OF DASHBOARD INVENTORY MANAGEMENT SYSTEM:**

```
 from tkinter import *
from turtle import width
from PIL import Image,ImageTk #pip install pillow
from employee import employeeClass
from supplier import supplierClass
from category import categoryClass
from product import productClass
from sales import salesClass
class IMS:
  def __init__(self,root):
```

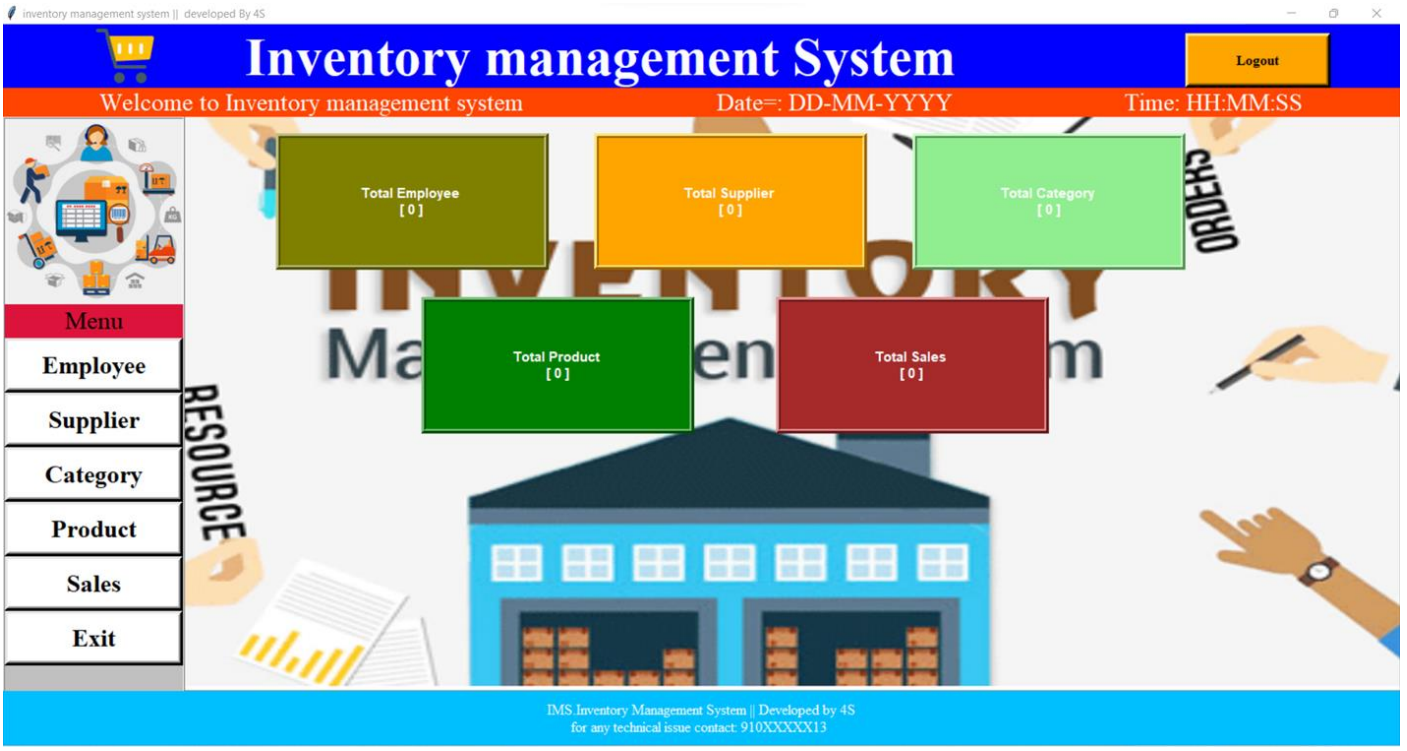
```
    self.root=root
    self.root.geometry('1350x700+0+0')
    self.root.title("inventory management system ||  developed By 4S")
    self.root.config(bg="Cyan")

#     self.bg=ImageTk.PhotoImage(file="images/10158.jpg")
#     lbl_bg=Label(self.root,image=self.bg,padx=100,pady=200)
#     lbl_bg.place(x=190,y=100,width=1360,height=650)

    self.im1=Image.open("images/img1.png")
    self.im1=self.im1.resize((1500,710),Image.ANTIALIAS)
    self.im1=ImageTk.PhotoImage(self.im1)

    self.lbl_im1=Label(self.root,image=self.im1,bd=2,relief=RAISED)
    self.lbl_im1.place(x=50,y=20)

    #====================== Title ====================================

    self.icon_title=PhotoImage(file="images/logo1.png")
    title=Label(self.root,text="Inventory management
System",image=self.icon_title,compound=LEFT,font=("times new
roman",44,"bold"),bg="Blue",fg="white",anchor="w",padx=100).place(x=0,y=0,relwidth=1,height=70)

   #======================btn logout ====================================

    btn_logout=Button(self.root,text="Logout",font=("times new
roman",11,"bold"),bg="orange",bd=5).place(x=1300,y=10,height=60,width=160)
    #================== 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",20),bg="orangered",fg="white",bd=5)
    self.lbl_clock.place(x=0,y=70,relwidth=1,height=32)

    #================ Left menu
================================================
```

```
    self.MenuLogo=Image.open("images/menu_im.png")
    self.MenuLogo=self.MenuLogo.resize((200,200),Image.ANTIALIAS)
    self.MenuLogo=ImageTk.PhotoImage(self.MenuLogo)

    Leftmenu=Frame(self.root,bd=2,relief=RIDGE,bg="Silver")
    Leftmenu.place(x=0,y=102,width=200,height=660)

    lbl_menuLogo=Label(Leftmenu,image=self.MenuLogo)
    lbl_menuLogo.pack(side=TOP,fill=X)

    lbl_menu=Label(Leftmenu,text="Menu",font=("times new
roman",20),bg="crimson").pack(side=TOP,fill=X)
    btn_Employee=Button(Leftmenu,text="Employee",command=self.employee,font=("times new
roman",20,"bold"),bg="white",bd=5,cursor="hand2").pack(side=TOP,fill=X)
    btn_supplier=Button(Leftmenu,text="Supplier",command=self.supplier,font=("times new
roman",20,"bold"),bg="white",bd=5,cursor="hand2").pack(side=TOP,fill=X)
    btn_category=Button(Leftmenu,text="Category",command=self.category,font=("times new
roman",20,"bold"),bg="white",bd=5,cursor="hand2").pack(side=TOP,fill=X)
    btn_product=Button(Leftmenu,text="Product",command=self.product,font=("times new
roman",20,"bold"),bg="white",bd=5,cursor="hand2").pack(side=TOP,fill=X)
    btn_sales=Button(Leftmenu,text="Sales",command=self.sales,font=("times new
roman",20,"bold"),bg="white",bd=5,cursor="hand2").pack(side=TOP,fill=X)
    btn_exit=Button(Leftmenu,text="Exit",font=("times new
roman",20,"bold"),bg="white",bd=5,cursor="hand2").pack(side=TOP,fill=X)

    #================== content
==============================================
    self.lbl_employee=Label(self.root,text="Total Employee \n[ 0
]",bd=5,relief=RIDGE,bg="olive",fg="white",font=("time new roman",11,"bold"))
    self.lbl_employee.place(x=300,y=120,height=150,width=300)

    self.lbl_supplier=Label(self.root,text="Total Supplier \n[ 0
]",bd=5,relief=RIDGE,bg="orange",fg="white",font=("time new roman",11,"bold"))
    self.lbl_supplier.place(x=650,y=120,height=150,width=300)
```

```
    self.lbl_category=Label(self.root,text="Total Category \n[ 0
]",bd=5,relief=RIDGE,bg="lightgreen",fg="white",font=("time new roman",11,"bold"))

    self.lbl_category.place(x=1000,y=120,height=150,width=300)


    self.lbl_product=Label(self.root,text="Total Product \n[ 0
]",bd=5,relief=RIDGE,bg="green",fg="white",font=("time new roman",11,"bold"))

    self.lbl_product.place(x=460,y=300,height=150,width=300)


    self.lbl_sales=Label(self.root,text="Total Sales \n[ 0
]",bd=5,relief=RIDGE,bg="brown",fg="white",font=("time new roman",11,"bold"))

    self.lbl_sales.place(x=850,y=300,height=150,width=300)

    #================= footer ============================================

    lbl_footer=Label(self.root,text="IMS.Inventory Management System || Developed by 4S\nfor any
technical issue contact: 910XXXXX13 ",font=("times new
roman",12),bg="deepskyblue",fg="white",bd=10).pack(side=BOTTOM,fill=X)

    #=====================================================================
=====


  def employee(self):

    self.new_win=Toplevel(self.root)

    self.new_win=employeeClass(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)

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

SNAPSHOTS OF LOGIN PAGE.



● **CODE OF EMPLOYEE DETAIL:**

```
from ast import Delete
from importlib.resources import contents
from multiprocessing import connection
from multiprocessing.sharedctypes import Value
from optparse import Values
from select import select
from webbrowser import get
import cx_Oracle
from tkinter import *
from PIL import Image,ImageTk #pip install pillow
from tkinter import ttk,messagebox
class employeeClass:
 def __init__(self,root):
    self.root=root
    self.root.geometry('1100x500+200+130')
    self.root.title("inventory management system ||  developed By 4S")
    self.root.config(bg="Cyan")
    self.root.focus_force()
    #============== all variable ================================
    self.Var_searchby=StringVar()
    self.Var_searchtxt=StringVar()

    self.Var_emp_id=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_pass=StringVar()
    self.Var_utype=StringVar()
    self.Var_address=StringVar()
    self.Var_salary=StringVar()


    #========== search frame ====================================
    SearchFrame=LabelFrame(self.root,text="Search Employee",font=("time new roman",12,"bold"),bd=5,relief=RIDGE,bg="white",fg="blue")
    SearchFrame.place(x=250,y=20,width=600,height=70)

    #================== option ================================
    cmb_search=ttk.Combobox(SearchFrame,textvariable=self.Var_searchby,values=("select","E mail","Name","contact"),state='readonly',justify=CENTER,font=("time new roman",15))
    cmb_search.place(x=10,y=10,width=180)
    cmb_search.current(0)

    text_search=Entry(SearchFrame,textvariable=self.Var_searchtxt,font=("time new roman",15),bg="lightyellow").place(x=200,y=10)
    btn_search=Button(SearchFrame,text="search",command=self.search,font=("time new roman",15),bg="#4caf50",fg="white",cursor="hand2").place(x=430,y=7,width=150,height=30)

    #============= title ============================================
    title=Label(self.root,text="Employee Detail",font=("time new roman",15),bg="green",fg="white",bd=5,cursor="hand2").place(x=50,y=100,width=1000)

    #============ content ============================================
    #============= row1 ==============================================
```

```
    lbl_emp_id=Label(self.root,text="Emp ID",font=("time new
roman",15,"bold"),bg="cyan").place(x=50,y=150)

    lbl_gender=Label(self.root,text="Gender",font=("time new
roman",15,"bold"),bg="cyan").place(x=400,y=150)

    lbl_contact=Label(self.root,text="Contact",font=("time new
roman",15,"bold"),bg="cyan").place(x=750,y=150)


    self.emp_id=Entry(self.root,textvariable=self.Var_emp_id,font=("time new
roman",15),bg="lightyellow")

    self.emp_id.place(x=150,y=150)

    #txt_gender=Entry(self.root,textvariable=self.Var_gender,font=("time new
roman",15),bg="lightyellow").place(x=500,y=150)

    cmb_gender=ttk.Combobox(self.root,textvariable=self.Var_gender,values=("select","Male","
Female","Other"),state='readonly',justify=CENTER,font=("time new roman",15))

    cmb_gender.place(x=500,y=150,width=220)

    cmb_gender.current(0)

    self.contact=Entry(self.root,textvariable=self.Var_contact,font=("time new
roman",15),bg="lightyellow")

    self.contact.place(x=900,y=150)

    #====================== row2
=====================================================

    lbl_name=Label(self.root,text="Name",font=("time new
roman",15,"bold"),bg="cyan").place(x=50,y=190)

    lbl_dob=Label(self.root,text="D.O.B",font=("time new
roman",15,"bold"),bg="cyan").place(x=400,y=190)

    lbl_doj=Label(self.root,text="D.O.J",font=("time new
roman",15,"bold"),bg="cyan").place(x=750,y=190)


    self.name=Entry(self.root,textvariable=self.Var_name,font=("time new
roman",15),bg="lightyellow")

    self.name.place(x=150,y=190)

    self.dob=Entry(self.root,textvariable=self.Var_dob,font=("time new
roman",15),bg="lightyellow")

    self.dob.place(x=500,y=190)
```

```
    self.doj=Entry(self.root,textvariable=self.Var_doj,font=("time new
roman",15),bg="lightyellow")

    self.doj.place(x=900,y=190)


     #====================== row3
=================================================

    lbl_email=Label(self.root,text="Email",font=("time new
roman",15,"bold"),bg="cyan").place(x=50,y=230)

    lbl_pass=Label(self.root,text="Password",font=("time new
roman",15,"bold"),bg="cyan").place(x=400,y=230)

    lbl_utype=Label(self.root,text="User Type",font=("time new
roman",15,"bold"),bg="cyan").place(x=750,y=230)


    self.email=Entry(self.root,textvariable=self.Var_email,font=("time new
roman",15),bg="lightyellow")

    self.email.place(x=150,y=230,width=220)

    self.password=Entry(self.root,textvariable=self.Var_pass,font=("time new
roman",15),bg="lightyellow")

    self.password.place(x=500,y=230,width=220)

    #txt_utype=Entry(self.root,textvariable=self.Var_utype,font=("time new
roman",15),bg="lightyellow").place(x=900,y=230,width=220)

    cmb_utype=ttk.Combobox(self.root,textvariable=self.Var_utype,values=("select","Admin","E
mployee"),state='readonly',justify=CENTER,font=("time new roman",15))

    cmb_utype.place(x=900,y=230,width=220)

    cmb_utype.current(0)


    #================== row4
=============================================================

    lbl_address=Label(self.root,text="Address",font=("time new
roman",15,"bold"),bg="cyan").place(x=50,y=270)

    lbl_salary=Label(self.root,text="Salary",font=("time new
roman",15,"bold"),bg="cyan").place(x=750,y=270)


    self.address=Entry(self.root,font=("time new roman",15),bg="lightyellow")
```

```
    self.address.place(x=150,y=270,width=320,height=70)


    self.salary=Entry(self.root,textvariable=self.Var_salary,font=("time new
roman",15),bg="lightyellow")
    self.salary.place(x=900,y=270,width=220)
    #================= buttons ==============================
    btn_add=Button(self.root,text="Save",command=self.add,font=("time new
roman",15),bg="#2196f3",fg="white",cursor="hand2").place(x=500,y=305,width=110,height=28
)


    btn_update=Button(self.root,text="Update",command=self.update,font=("time new
roman",15),bg="#4caf50",fg="white",cursor="hand2").place(x=620,y=305,width=110,height=28
)


    btn_delete=Button(self.root,text="Delete",command=self.delete,font=("time new
roman",15),bg="#f44336",fg="white",cursor="hand2").place(x=740,y=305,width=110,height=28
)


    btn_clear=Button(self.root,text="Clear",command=self.clear,font=("time new
roman",15),bg="#607d8b",fg="white",cursor="hand2").place(x=860,y=305,width=110,height=2
8)


    #=========== Employee Detail ===============================================
    emp_frame=Frame(self.root,bd=3,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.EmployeeTable=ttk.Treeview(emp_frame,columns=("eid","name","email","gender","co
ntact","dob","doj","pass","utype","address","salary"),yscrollcommand=scrolly.set,xscrollcom
mand=scrollx.set)
    scrollx.pack(side=BOTTOM,fill=X)
    scrolly.pack(side=RIGHT,fill=Y)
```

```
    scrollx.config(command=self.EmployeeTable.xview)
    scrolly.config(command=self.EmployeeTable.yview)
    #===========heading=======================================
    self.EmployeeTable.heading("eid",text="EMP ID")
    self.EmployeeTable.heading("name",text="Name")
    self.EmployeeTable.heading("email",text="Email")
    self.EmployeeTable.heading("gender",text="Gender")
    self.EmployeeTable.heading("contact",text="Contact")
    self.EmployeeTable.heading("dob",text="D.O.B")
    self.EmployeeTable.heading("doj",text="D.O.J")
    self.EmployeeTable.heading("pass",text="Password")
    self.EmployeeTable.heading("utype",text="User Type")
    self.EmployeeTable.heading("address",text="Address")
    self.EmployeeTable.heading("salary",text="Salary")
    self.EmployeeTable["show"]="headings"
#==============column ============================================
    self.EmployeeTable.column("eid",width=90)
    self.EmployeeTable.column("name",width=100)
    self.EmployeeTable.column("email",width=100)
    self.EmployeeTable.column("gender",width=100)
    self.EmployeeTable.column("contact",width=100)
    self.EmployeeTable.column("dob",width=100)
    self.EmployeeTable.column("doj",width=100)
    self.EmployeeTable.column("pass",width=100)
    self.EmployeeTable.column("utype",width=100)
    self.EmployeeTable.column("address",width=100)
    self.EmployeeTable.column("salary",width=100)
    self.EmployeeTable["show"]="headings"
    self.EmployeeTable.pack(fill=BOTH,expand=1)
    #self.EmployeeTable.bind("<ButtonRelease-1>",self.get_data)
```

```
   #self.show()


#=======================================backend=========================
==========

#====================Added function
=================================================

 def add(self):

     connection=cx_Oracle.connect('sumit/abhinav')

     cur=connection.Cursor()

     try:

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

           messagebox.showerror("Error","Employee ID Must be required",parent=self.root)


       else:

           cur.execute("select *from employee where eid=?",(self.Var_emp_id.get(),))

           row=cur.fetchone()

           if row!=None:

               messagebox.showerror("Error","This Employee ID already assigned,try
different",parent=self.root)

           else:

             cur.execute("Insert into
employee(eid,name,email,gender,contact,dob,doj,pass,utype,address,salary)
values(?,?,?,?,?,?,?,?,?,?,?)",(

                                 self.Var_emp_id.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_pass.get(),
                    self.Var_utype.get(),
                    self.Var_address.get('1.0',END),
                    self.Var_salary.get(),
            ))
            connection.Commit()
            messagebox.showinfo("success","Employee Added Successfully",parent=self.root)
            self.show()

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

def show(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      cur.execute("select *from employee")
      rows=cur.fetchall()
      self.EmployeeTable.delete(*self.EmployeeTable.get_children())
      for row in rows:
          self.EmployeeTable.insert('',END,values=row)
    except Exception as ex:
      messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=self.root)

def get_data(self,ev):
    f=self.EmployeeTable.focus()
    content=(self.EmployeeTable.item(f))
    row=content['values']
    print(row)
```

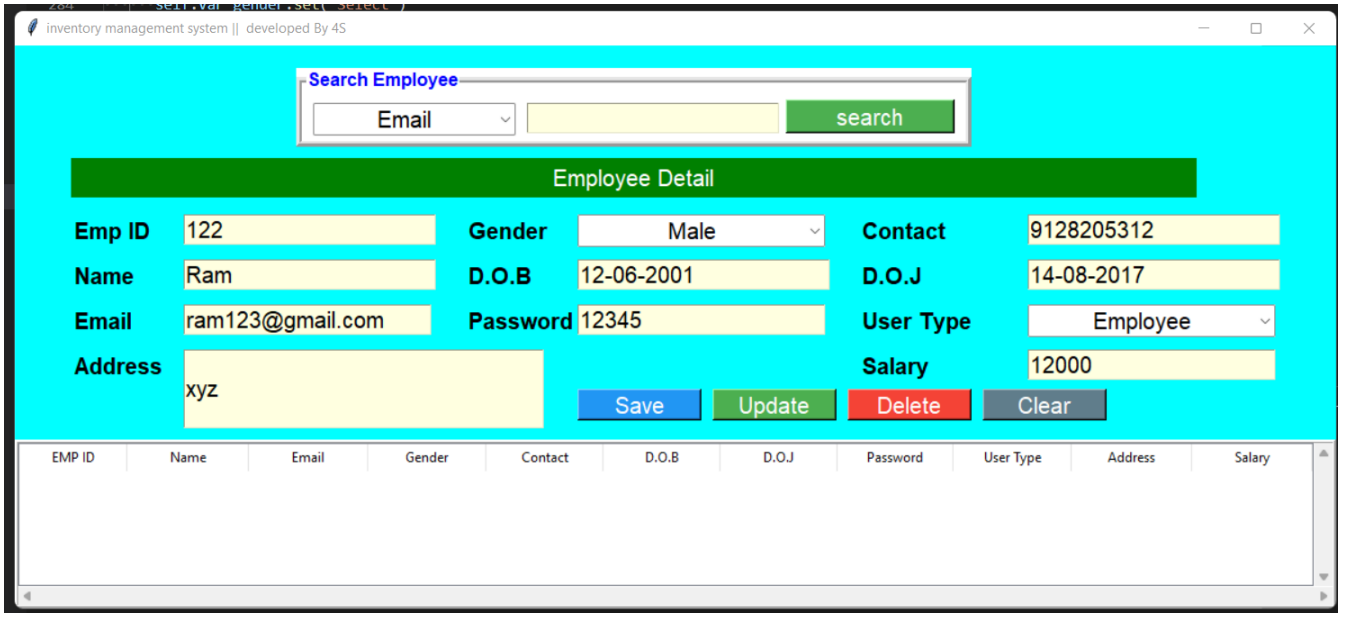
```
    self.Var_emp_id.set(row[0])
    self.Var_name.set(row[1])
    self.Var_email.set(row[2])
    self.Var_gender.set(row[3])
    self.Var_contact.set(row[4])
    self.Var_dob.set(row[5])
    self.Var_doj.set(row[6])
    self.Var_pass.set(row[7])
    self.Var_utype.set(row[8])
    self.Var_address.delete('1.0',END)
    self.Var_address.insert(END,row[9])
    self.Var_salary.set(row[10])
#========== Update function ==============================================
 def update(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      if self.Var_emp_id.get()=="":
          messagebox.showerror("Error","Employee ID Must be required",parent=self.root)

      else:
          cur.execute("select *from employee where eid=?",(self.Var_emp_id.get(),))
          row=cur.fetchone()
          if row==None:
              messagebox.showerror("Error","Invalid Employee ID",parent=self.root)
          else:
            cur.execute("Update employee set
name=?,email=?,gender=?,contact=?,dob,doj=?,pass=?,utype=?,address=?,salary=? where
eid=?",(

                                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_pass.get(),
                        self.Var_utype.get(),
                        self.Var_address.get('1.0',END),
                        self.Var_salary.get(),
                        self.Var_emp_id.get(),
            ))
            connection.Commit()
            messagebox.showinfo("success","Employee Updated Successfully",parent=self.root)
            self.show()
            #con.close()

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

#================ Delete function
================================================
 def delete(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      if self.Var_emp_id.get()=="":
          messagebox.showerror("Error","Employee ID Must be required",parent=self.root)
```

```
        else:
            cur.execute("select *from employee where eid=?",(self.Var_emp_id.get(),))
            row=cur.fetchone()
            if row==None:
                messagebox.showerror("Error","Invalid Employee ID",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 whrere eid=?",(self.Var_emp_id.get(),))
                connection.commit()
                messagebox.showinfo("Delete","Employee Deleted Successfully",parent=self.root)
                self.clear()
    except Exception as ex:
      messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=self.root)

#=============Clear function ====================================
 def clear(self):
    self.Var_emp_id.set("")
    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_pass.set("")
    self.Var_utype.set("Admin")
    self.Var_address.delete('1.0',END)
    self.Var_salary.set("")
    self.Var_searchtxt.get("")
    self.Var_searchby.get("Select")
```

```
        self.show()


    def search(self):
        connection=cx_Oracle.connect('sumit/abhinav')
        cur=connection.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.EmployeeTable.delete(*self.EmployeeTable.get_children())
                    for row in rows:
                        self.EmployeeTable.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)}",parent=self.root)


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

□ SNAPSHOTS OF EMPLOYEE DETAIL.



□ BACKEND OF EMPLOYEE DETAIL

```
SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 EMP_ID                                    NOT NULL VARCHAR2(20)
 GENDER                                             VARCHAR2(20)
 CONTACT                                            VARCHAR2(30)
 NAME                                               VARCHAR2(30)
 DOB                                                DATE
 DOJ                                                DATE
 EMAIL                                              VARCHAR2(40)
 PASS                                               VARCHAR2(40)
 UTYPE                                              VARCHAR2(40)
 ADDRESS                                            VARCHAR2(30)
 SALARY                                             VARCHAR2(20)
```

● **CODE OF SUPPLIER DETAIL:**

```
import cx_Oracle
from tkinter import *
from PIL import Image,ImageTk #pip install pillow
from tkinter import ttk,messagebox
class supplierClass:
 def __init__(self,root):
    self.root=root
    self.root.geometry('1100x500+200+130')
    self.root.title("inventory management system ||  developed By Sumit")
    self.root.config(bg="Cyan")
    self.root.focus_force()
    #============== all variable ================================
    self.Var_searchby=StringVar()
    self.Var_searchtxt=StringVar()

    self.var_sup_invoice=StringVar()
    self.Var_name=StringVar()
    self.Var_contact=StringVar()

    #========== search frame ====================================
    SearchFrame=LabelFrame(self.root,text="Search Supplier",font=("time new
roman",12,"bold"),bd=2,relief=RIDGE,bg="white",fg="blue")
    SearchFrame.place(x=250,y=20,width=600,height=70)

    #================== option ================================
    lbl_search=Label(SearchFrame,text="Search By Invoice No.",bg="white",font=("time new
roman",12,"bold"))
    lbl_search.place(x=10,y=10)
```

```
    text_search=Entry(SearchFrame,textvariable=self.Var_searchtxt,font=("time new
roman",15),bg="lightyellow").place(x=200,y=10)

    btn_search=Button(SearchFrame,text="search",command=self.search,font=("time new
roman",15),bg="#4caf50",fg="white",cursor="hand2").place(x=430,y=7,width=150,height=30)


    #============= title ================================================

    title=Label(self.root,text="Supplier Detail",font=("time new
roman",15,"bold"),bg="green",fg="white",bd=5,cursor="hand2").place(x=50,y=100,width=100
0)


    #============ content ================================================

    #============= row1
=====================================================

    lbl_supplier_invoice=Label(self.root,text="Invoice No.",font=("time new
roman",15,"bold"),bg="cyan").place(x=50,y=150)


    txt_supplier_invoice=Entry(self.root,textvariable=self.var_sup_invoice,font=("time new
roman",15),bg="lightyellow").place(x=170,y=150)

    #====================== row2
=====================================================

    lbl_name=Label(self.root,text="Name",font=("time new
roman",15,"bold"),bg="cyan").place(x=460,y=150)

    txt_name=Entry(self.root,textvariable=self.Var_name,font=("time new
roman",15),bg="lightyellow").place(x=550,y=150)


     #====================== row3
=====================================================

    lbl_contact=Label(self.root,text="Contact",font=("time new
roman",15,"bold"),bg="cyan").place(x=50,y=200)


    txt_contact=Entry(self.root,textvariable=self.Var_contact,font=("time new
roman",15),bg="lightyellow").place(x=170,y=200,width=220)

    #================== row4
=============================================================
```

```
    lbl_desc=Label(self.root,text="Description",font=("time new
roman",15,"bold"),bg="cyan").place(x=440,y=200)

    self.txt_desc=Text(self.root,font=("time new roman",15),bg="lightyellow")

    self.txt_desc.place(x=570,y=200,width=350,height=70)

    #================= buttons ==============================

    btn_add=Button(self.root,text="Save",command=self.add,font=("time new
roman",15),bg="#2196f3",fg="white",cursor="hand2").place(x=500,y=305,width=110,height=28
)


    btn_update=Button(self.root,text="Update",command=self.update,font=("time new
roman",15),bg="orange",fg="white",cursor="hand2").place(x=620,y=305,width=110,height=28)


    btn_delete=Button(self.root,text="Delete",command=self.delete,font=("time new
roman",15),bg="red",fg="white",cursor="hand2").place(x=740,y=305,width=110,height=28)


    btn_clear=Button(self.root,text="Clear",command=self.clear,font=("time new
roman",15),bg="gray",fg="white",cursor="hand2").place(x=860,y=305,width=110,height=28)


    #=========== supplier Detail ===============================================

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

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


    scrolly=Scrollbar(sup_frame,orient=VERTICAL)

    scrollx=Scrollbar(sup_frame,orient=HORIZONTAL)


    self.supplierTable=ttk.Treeview(sup_frame,columns=("invoice","name","contact","desc"),ysc
rollcommand=scrolly.set,xscrollcommand=scrollx.set)

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

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

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

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

    self.supplierTable.heading("invoice",text="Invoice No.")
```

```
   self.supplierTable.heading("name",text="Name")
   self.supplierTable.heading("contact",text="Contact")
   self.supplierTable.heading("desc",text="Description")
   self.supplierTable["show"]="headings"

   self.supplierTable.column("invoice",width=90)
   self.supplierTable.column("name",width=100)
   self.supplierTable.column("contact",width=100)
   self.supplierTable.column("desc",width=100)
   self.supplierTable.pack(fill=BOTH,expand=1)
#   self.supplierTable.bind("<ButtonRelease-1>",self.get_data)

  #  self.show()

#=========================================backend=========================
==========
#====================Added function
=================================================
 def add(self):
     connection=cx_Oracle.connect('sumit/abhinav')
     cur=connection.Cursor()
     try:
       if self.var_sup_invoice.get()=="":
           messagebox.showerror("Error","Invoice Must be required",parent=self.root)

       else:
           cur.execute("select *from supplier where invoice=?",(self.var_sup_invoice.get(),))
           row=cur.fetchone()
           if row!=None:
               messagebox.showerror("Error","INvoice no. already assigned,try
different",parent=self.root)
```

```
            else:
                cur.execute("Insert into supplier (invoice,name,contact,desc) values(?,?,?,?)",(
                                    self.var_sup_invoice.get(),
                                    self.Var_name.get(),
                                    self.Var_contact.get(),

                                    self.txt_desc.get('1.0',END),
                 ))
                connection.Commit()
                messagebox.showinfo("success","Supplier Added Successfully",parent=self.root)
                self.show()

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

    def show(self):
        connection=cx_Oracle.connect('sumit/abhinav')
        cur=connection.Cursor()
        try:
            cur.execute("select *from supplier")
            rows=cur.fetchall()
            self.supplierTable.delete(*self.supplierTable.get_children())
            for row in rows:
                self.supplierTable.insert('',END,values=row)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=self.root)

    def get_data(self,ev):
        f=self.supplierTable.focus()
        content=(self.supplierTable.item(f))
```

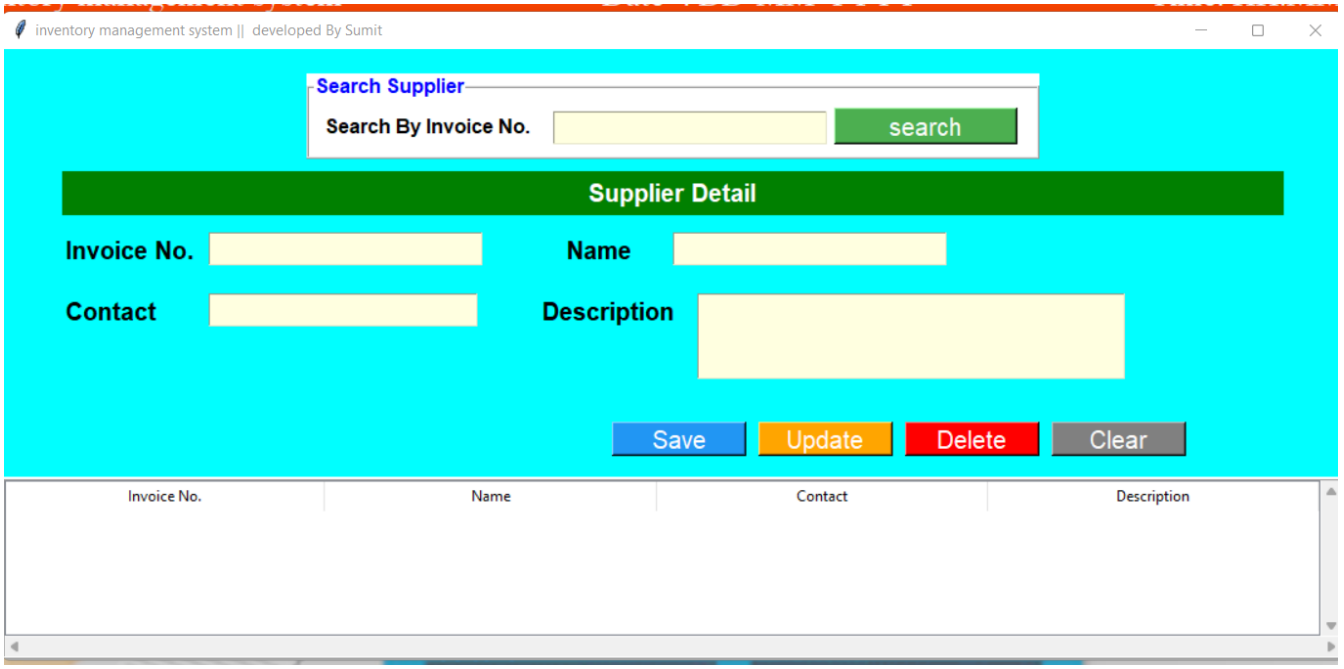
```
    row=content['values']
    print(row)

    self.var_sup_invoice.set(row[0])
    self.Var_name.set(row[1])

    self.Var_contact.set(row[2])

    self.txt_desc.delete('1.0',END)
    self.txt_desc.insert(END,row[3])
#========== Update function ==============================================
 def update(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      if self.var_sup_invoice.get()=="":
          messagebox.showerror("Error","Invoice no. Must be required",parent=self.root)

      else:
          cur.execute("select *from supplier where invoice=?",(self.var_sup_invoice.get(),))
          row=cur.fetchone()
          if row==None:
              messagebox.showerror("Error","Invalid Invoice no.",parent=self.root)
          else:
            cur.execute("Update supplier set name=?,contact=?,desc=? where invoice=?",(

                                self.Var_name.get(),

                                self.Var_contact.get(),
                                self.txt_desc.get('1.0',END),
```

```
                    self.var_sup_invoice.get(),
            ))
            connection.Commit()
            messagebox.showinfo("success","Supplier Updated Successfully",parent=self.root)
            self.show()
            #con.close()

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

#================ Delete function
================================================
 def delete(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      if self.var_sup_invoice.get()=="":
          messagebox.showerror("Error","Invoice no. Must be required",parent=self.root)

      else:
          cur.execute("select *from supplier where invoice=?",(self.var_sup_invoice.get(),))
          row=cur.fetchone()
          if row==None:
              messagebox.showerror("Error","Invalid Invoice No.",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 invoice=?",(self.var_sup_invoice.get(),))
              connection.commit()
              messagebox.showinfo("Delete","Supplier Deleted Successfully",parent=self.root)
```

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

#=============Clear function ====================================
    def clear(self):
        self.var_sup_invoice.set("")
        self.Var_name.set("")
        self.Var_contact.set("")

        self.txt_desc.delete('1.0',END)
        self.Var_searchtxt.get("")
        self.show()

    def search(self):
        connection=cx_Oracle.connect('sumit/abhinav')
        cur=connection.Cursor()
        try:
            if self.Var_searchtxt.get()=="":
                 messagebox.showerror("error","Invoice No. should be required",parent=self.root)
            else:
                cur.execute("select *from supplier where invoice=?",(self.Var_searchtxt.get(),))
                row=cur.fetchone()
                if row!=None:

                    self.supplierTable.delete(*self.supplierTable.get_children())
                    self.supplierTable.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)}",parent=self.root)


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

 SNAPSHOTS OF SUPPLIER DETAIL.



 BACKEND OF SUPPLIER DETAIL

```
SQL> desc supplier
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 SEARCHBY                                           VARCHAR2(20)
 SEARCHTXT                                          VARCHAR2(30)
 SUP_INVOICE                                        VARCHAR2(20)
 NAME                                               VARCHAR2(40)
 CONTACT                                            VARCHAR2(30)
```

● **CODE OF CATEGORY DETAIL:**

```
import cx_Oracle
from tkinter import *
from PIL import Image,ImageTk #pip install pillow
from tkinter import ttk,messagebox
class categoryClass:
 def __init__(self,root):
   self.root=root
   self.root.geometry('1100x500+200+130')
   self.root.title("inventory management system ||  developed By Sumit")
   self.root.config(bg="Cyan")
   self.root.focus_force()
   #============== variable ====================================
   self.var_cat_id=StringVar()
   self.var_name=StringVar()
   #=========== title ========================================
   lbl_title=Label(self.root,text="Manage Product Category",font=("time new
roman",30),bg="#184a45",bd=3,relief=RIDGE,fg="white").pack(side=TOP,fill=X,padx
=10,pady=10)

   lbl_name=Label(self.root,text="Enter Category Name",font=("time new
roman",20,"bold"),bg="blue",fg="orange",bd=10).place(x=50,y=100)
   txt_name=Entry(self.root,textvariable=self.var_name,font=("time new
roman",18),bg="lightyellow").place(x=50,y=170,width=300)

   btn_add=Button(self.root,text="ADD",command=self.add,font=("time new
roman",15),bg="#4caf50",fg="white",cursor="hand2").place(x=360,y=170,width=150,h
eight=30)

   btn_delete=Button(self.root,text="Delete",command=self.delete,font=("time new
roman",15),bg="red",fg="white",cursor="hand2").place(x=520,y=170,width=150,heigh
t=30)
```

```
    #========== category Detail
==============================================
    cat_frame=Frame(self.root,bd=3,relief=RIDGE)
    cat_frame.place(x=700,y=100,width=380,height=100)

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

    self.category_table=ttk.Treeview(cat_frame,columns=("cid","name"),yscrollcomman
d=scrolly.set,xscrollcommand=scrollx.set)
    scrollx.pack(side=BOTTOM,fill=X)
    scrolly.pack(side=RIGHT,fill=Y)
    scrollx.config(command=self.category_table.xview)
    scrolly.config(command=self.category_table.yview)
    self.category_table.heading("cid",text="C id")
    self.category_table.heading("name",text="Name")
    self.category_table["show"]="headings"

    self.category_table.column("cid",width=90)
    self.category_table.column("name",width=100)
    self.category_table.pack(fill=BOTH,expand=1)
    #self.category_table.bind("<ButtonRelease-1>",self.get_data)

    #============ Images ============================
    self.im1=Image.open("images/cat.jpg")
    self.im1=self.im1.resize((500,250),Image.ANTIALIAS)
    self.im1=ImageTk.PhotoImage(self.im1)

    self.lbl_im1=Label(self.root,image=self.im1,bd=2,relief=RAISED)
    self.lbl_im1.place(x=50,y=220)
```

```
   self.im2=Image.open("images/category.jpg")
   self.im2=self.im2.resize((500,250),Image.ANTIALIAS)
   self.im2=ImageTk.PhotoImage(self.im2)

   self.lbl_im2=Label(self.root,image=self.im2,bd=2,relief=RAISED)
   self.lbl_im2.place(x=580,y=220)

   #self.show()

#=================== Backend ==================================
#==================add function ================================
 def add(self):
     connection=cx_Oracle.connect('sumit/abhinav')
     cur=connection.Cursor()
     try:
       if self.var_name.get()=="":
           messagebox.showerror("Error","Category name should 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","Category already present,try
different",parent=self.root)
           else:
             cur.execute("Insert into category (name) values(?)",(self.var_name.get(),))
             connection.Commit()
             messagebox.showinfo("success","Category Added
Successfully",parent=self.root)
             self.show()
```

```
    except Exception as ex:
      messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=self.root)
 def show(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      cur.execute("select *from category")
      rows=cur.fetchall()
      self.category_table.delete(*self.category_table.get_children())
      for row in rows:
          self.category_table.insert('',END,values=row)
    except Exception as ex:
      messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=self.root)

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

    self.var_cat_id.set(row[0])
    self.var_name.set(row[1])

 def delete(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
      if self.var_cat_id.get()=="":
          messagebox.showerror("Error","please select category from the
list",parent=self.root)

      else:
```

```
            cur.execute("select *from category where cid=?",(self.var_cat_id.get(),))
            row=cur.fetchone()
            if row==None:
                messagebox.showerror("Error","Error,please try again",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 cid=?",(self.var_cat_id.get(),))
                connection.commit()
                messagebox.showinfo("Delete","category Deleted
Successfully",parent=self.root)
                self.show()
                self.var_cat_id.set("")
                self.var_name.set("")
     except Exception as ex:
       messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=self.root)



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

□ SNAPSHOTS OF CATEGORY DETAIL.

□ BACKEND OF CATEGORY DETAIL

```
SQL> desc category
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CAT_ID                                             VARCHAR2(30)
 NAME                                               VARCHAR2(40)
```

●CODE OF PRODUCT DETAIL:

```
import cx_Oracle
from tkinter import *
from PIL import Image,ImageTk #pip install pillow
from tkinter import ttk,messagebox
class productClass:
 def __init__(self,root):
    self.root=root
    self.root.geometry('1100x500+200+130')
    self.root.title("inventory management system ||  developed By sumit")
    self.root.config(bg="Cyan")
    self.root.focus_force()
#=================================================
    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.fetch_cat_sup()

    self.var_name=StringVar()
    self.var_price=StringVar()
    self.var_qty=StringVar()
    self.var_status=StringVar()

    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 Detail",font=("time new
roman",15),bg="#4caf50",fg="white").pack(side=TOP,fill=X)
    #============ column1 ==========================================
    lbl_category=Label(product_Frame,text="category",font=("time new
roman",15),bg="white").place(x=30,y=60)
    lbl_supplier=Label(product_Frame,text="supplier",font=("time new
roman",15),bg="white").place(x=30,y=110)
    lbl_product=Label(product_Frame,text="product",font=("time new
roman",15),bg="white").place(x=30,y=160)
    lbl_price=Label(product_Frame,text="price",font=("time new
roman",15),bg="white").place(x=30,y=210)
    lbl_quntity=Label(product_Frame,text="Quantity",font=("time new
roman",15),bg="white").place(x=30,y=260)
    lbl_status=Label(product_Frame,text="Status",font=("time new
roman",15),bg="white").place(x=30,y=310)

   # txt_category=Label(product_Frame,text="category",font=("time new
roman",15),bg="white").place(x=30,y=60)
    #============ column2 ==========================================
    cmb_cat=ttk.Combobox(product_Frame,textvariable=self.var_cat,values=self.cat_list,state=
'readonly',justify=CENTER,font=("time new roman",15))
    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=("time new roman",15))
    cmb_sup.place(x=150,y=110,width=200)
    #cmb_sup.current(0)

    txt_name=Entry(product_Frame,textvariable=self.var_name,font=("time new
roman",15),bg="lightyellow").place(x=150,y=160,width=200)
    txt_price=Entry(product_Frame,textvariable=self.var_price,font=("time new
roman",15),bg="lightyellow").place(x=150,y=210,width=200)
    txt_qty=Entry(product_Frame,textvariable=self.var_qty,font=("time new
roman",15),bg="lightyellow").place(x=150,y=260,width=200)
```

```
    cmb_status=ttk.Combobox(product_Frame,textvariable=self.var_status,values=("Active","I
nactive"),state='readonly',justify=CENTER,font=("time new roman",15))
    cmb_status.place(x=150,y=310,width=200)
    cmb_status.current(0)

    #================= buttons ==============================
    btn_add=Button(product_Frame,text="Save",command=self.add,font=("time new
roman",15),bg="#2196f3",fg="white",cursor="hand2").place(x=5,y=400,width=100,height=2
8)

    btn_update=Button(product_Frame,text="Update",font=("time new
roman",15),bg="#4caf50",fg="white",cursor="hand2").place(x=110,y=400,width=100,height=
28)

    btn_delete=Button(product_Frame,text="Delete",font=("time new
roman",15),bg="#f44336",fg="white",cursor="hand2").place(x=224,y=400,width=100,height
=28)

    btn_clear=Button(product_Frame,text="Clear",font=("time new
roman",15),bg="#607d8b",fg="white",cursor="hand2").place(x=338,y=400,width=100,height
=28)

    #========== search frame ====================================
    SearchFrame=LabelFrame(self.root,text="Search Product",font=("time new
roman",12,"bold"),bd=2,relief=RIDGE,bg="white",fg="blue")
    SearchFrame.place(x=520,y=10,width=600,height=80)

    #================== option ================================
    cmb_search=ttk.Combobox(SearchFrame,textvariable=self.Var_searchby,values=("select","
Category","Supplier","Name"),state='readonly',justify=CENTER,font=("time new
roman",15))
    cmb_search.place(x=10,y=10,width=180)
    cmb_search.current(0)

    text_search=Entry(SearchFrame,textvariable=self.Var_searchtxt,font=("time new
roman",15),bg="lightyellow").place(x=200,y=10)
    btn_search=Button(SearchFrame,text="search",font=("time new
roman",15),bg="#4caf50",fg="white",cursor="hand2").place(x=430,y=7,width=150,height=30
)

    #=========== Product Detail
============================================
    p_frame=Frame(self.root,bd=3,relief=RIDGE)
    p_frame.place(x=520,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","p
rice","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="P 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["show"]="headings"
    self.product_table.pack(fill=BOTH,expand=1)

    #self.show()

#=========================================backend==================
===============
#====================Added function
=================================================
 def fetch_cat_sup(self):
    self.cat_list.append("Empty")
    self.sup_list.append("Empty")

    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.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])
```

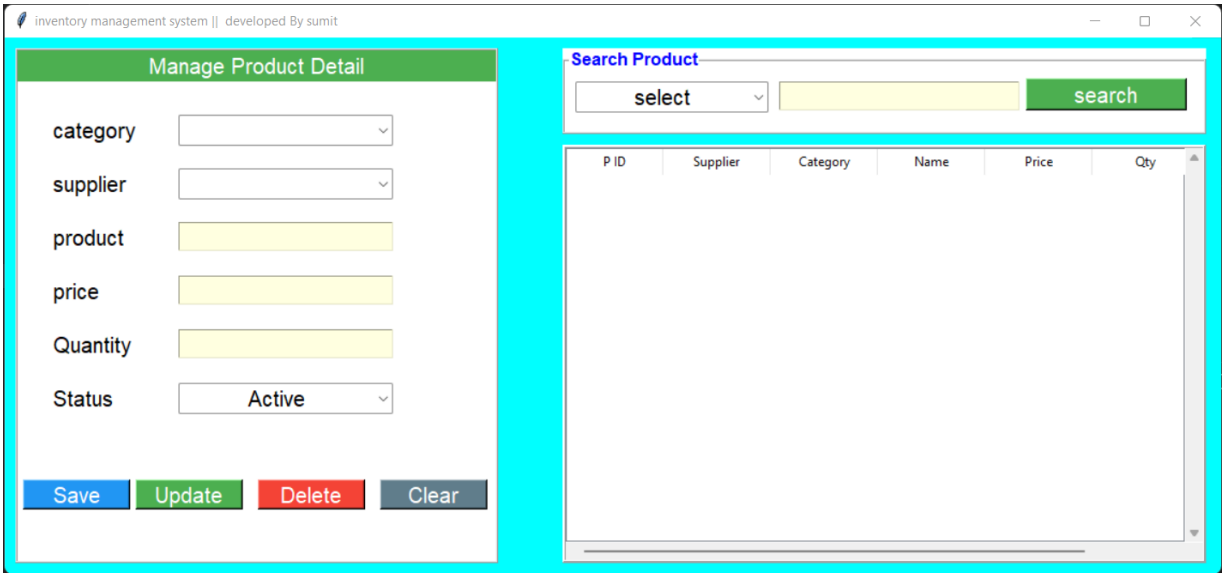
```
        cur.execute("select name from supplier")
        sup=cur.fetchall()
        if len(sup)>0:
            del self.sup_list[:]
            self.sup_list.append("Select")
            for i in sup:
                self.sup_list.append(i[0])

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

 def add(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
        if self.var_cat.get()=="select" or self.var_cat.get()=="Empty" or
self.var_sup.get()=="select" or self.var_name.get()=="":
            messagebox.showerror("Error","All field 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
different",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(),
                  ))
                connection.Commit()
                messagebox.showinfo("success","Product Added Successfully",parent=self.root)
                self.show()

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

 def show(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
```

```
        cur.execute("select *from supplier")
        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)}",parent=self.root)

def get_data(self,ev):
    f=self.product_table.focus()
    content=(self.product_table.item(f))
    row=content['values']
    print(row)
    self.Var_pid.set(row[0])
    self.var_sup.set(row[1])
    self.var_cat.set(row[2])

    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):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
        if self.Var_pid.get()=="":
            messagebox.showerror("Error","please select product from 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",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()
                    ))
                connection.Commit()
                messagebox.showinfo("success","Product Updated Successfully",parent=self.root)
```

```
            self.show()
            #con.close()

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

  def delete(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.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",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 whrere pid=?",(self.Var_pid.get(),))
                    connection.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)}",parent=self.root)

  def clear(self):
    self.var_cat.set("")
    self.var_sup.set("")
    self.var_name.set("")
    self.var_price.set("")
    self.var_qty.set("")

    self.var_status.set("")
    self.Var_pid.set("")
    self.Var_searchtxt.set("")
    self.Var_searchby.set("Select")
    self.show()

  def search(self):
      connection=cx_Oracle.connect('sumit/abhinav')
      cur=connection.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)}",parent=self.root)

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

□ SNAPSHOTS OF PRODUCT DETAIL.



□ BACKEND OF PRODUCT DETAIL

```
SQL> desc product
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 SEARCHBY                                           VARCHAR2(20)
 SEARCHTXT                                          VARCHAR2(30)
 SUPPLIER                                           VARCHAR2(20)
 NAME                                               VARCHAR2(40)
 PRICE                                              VARCHAR2(30)
 QUNTITY                                            VARCHAR2(30)
 STATUS                                             VARCHAR2(40)
```

● **CODE OF SALES DETAIL:**

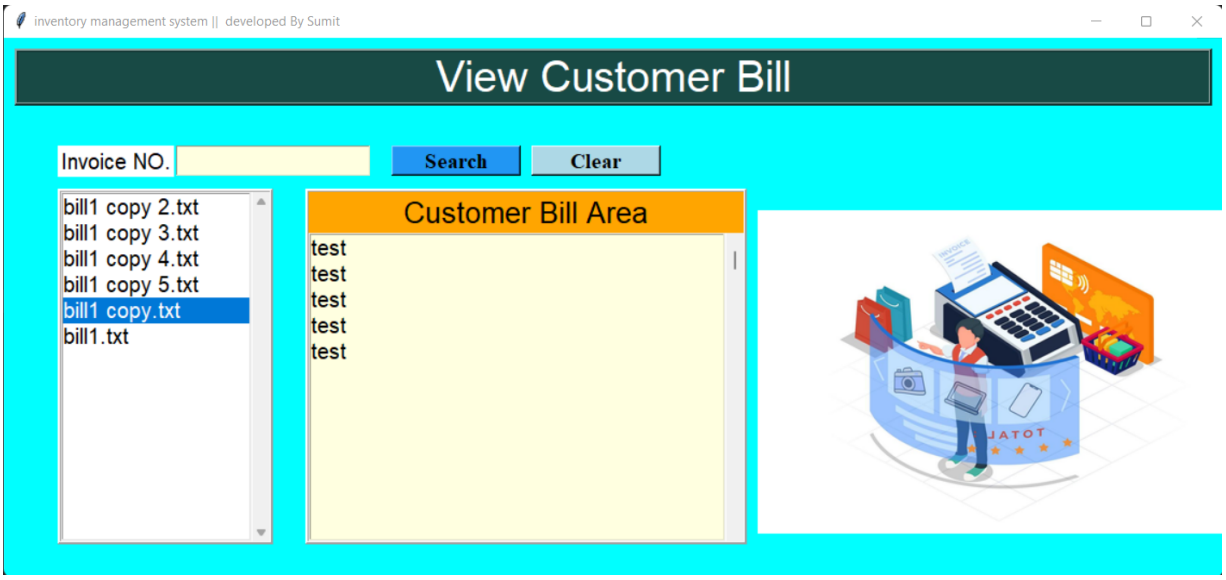
```
from tkinter import*
from PIL import Image,ImageTk #pip install pillow
from tkinter import ttk,messagebox
import cx_Oracle
import os
class salesClass:
 def __init__(self,root):
   #  super(root, self).__init__()
   self.root=root
   self.root.geometry('1100x500+200+130')
   self.root.title("inventory management system ||  developed By Sumit")
   self.root.config(bg="Cyan")
   self.root.focus_force()
   self.bill_list=[]
   self.var_invoice=StringVar()
   #====== title ========================================
   lbl_title=Label(self.root,text="View Customer Bill",font=("goudy old
style",30),bg="#184a45",bd=3,relief=RIDGE,fg="white").pack(side=TOP,fill=X,padx=1
0,pady=10)


   lbl_invoice=Label(self.root,text="Invoice NO.",font=("goudy old
style",15),bg="white").place(x=50,y=100)


   txt_invoice=Entry(self.root,textvariable=self.var_invoice,font=("goudy old
style",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="black",cursor="hand2").place(x=360,y=100,widt
h=120,height=28)
```

```
    btn_clear=Button(self.root,text="Clear",command=self.clear,font=("times new roman",15,"bold"),bg="lightblue",fg="black",cursor="hand2").place(x=490,y=100,width=120,height=28)


    #======================= Bills 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=("time new roman",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)

#======================= Bills 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),bg="orange").pack(side=TOP,fill=X)

    scrolly2=Scrollbar(bill_Frame,orient=VERTICAL)

    self.bill_area=Listbox(bill_Frame,font=("time new roman",15),bg="lightyellow",yscrollcommand=scrolly.set)

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

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

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


    #======== image ==================================

    self.bill_photo=Image.open("images/cat2.jpg")

    self.bill_photo=self.bill_photo.resize((450,300),Image.ANTIALIAS)

    self.bill_photo=ImageTk.PhotoImage(self.bill_photo)
```

```
    lbl_image=Label(self.root,image=self.bill_photo,bd=0)
    lbl_image.place(x=700,y=160)

    self.show()
    #============================================================
==
  def show(self):
    del self.bill_list[:]
    self.Sales_List.delete(0,END)
    #print(os.listdir('../IMS'))
    for i in os.listdir('bill'):
      # print(i.split('.'),i.split('.')[-1])
      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_)
    print(file_name)
    #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. should be 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.bill_area.delete('1.0',END)
if __name__=="__main__":
    root=Tk()
    obj=salesClass(root)
    root.mainloop()
```

SNAPSHOTS OF SALES DETAIL.



BACKEND OF SALES DETAIL

```
SQL> desc sales
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 INVOICE                                            VARCHAR2(40)
```

● **CODE OF BILLING PARTS:**

```
from tkinter import *
import time
from tkinter import ttk,messagebox
from unicodedata import name
from unittest import result
from webbrowser import get
from PIL import Image,ImageTk #pip install pillow
import cx_Oracle
class BillClass:
  def __init__(self,root):
   self.root=root
   self.root.geometry('1350x700+0+0')
   self.root.title("inventory management system ||  developed By Sumit")
   self.root.config(bg="Cyan")
   self.cart_list=[]

   #======================= Title
=====================================
   self.icon_title=PhotoImage(file="images/logo1.png")
   title=Label(self.root,text="Inventory management
System",image=self.icon_title,compound=LEFT,font=("times new
roman",48,"bold"),bg="Blue",fg="white",anchor="w",padx=20).place(x=0,y=0,relwidt
h=1,height=70)

  #======================btn logout
=======================================
   btn_logout=Button(self.root,text="Logout",font=("times new
roman",20,"bold"),bg="orange").place(x=1300,y=10,height=60,width=160)
   #================= 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",20),bg="#4d636d",fg="white")

    self.lbl_clock.place(x=0,y=70,relwidth=1,height=30)


    #============ product frame ==========================


    #============ Search Frame ============================

    productFrame1=Frame(self.root,bd=4,relief=RIDGE,bg="white")

    productFrame1.place(x=6,y=110,width=410,height=550)


    pTitle=Label(productFrame1,text="All Product",font=("time new roman",20,"bold"),bg="black",fg="white").pack(side=TOP,fill=X)

#================ product search Frame ============================

    self.var_search=StringVar()

    productFrame2=Frame(productFrame1,bd=2,relief=RIDGE,bg="white")

    productFrame2.place(x=2,y=42,width=398,height=90)


    lbl_Search=Label(productFrame2,text="Search Product || By Name",font=("time new roman",15,"bold"),bg="white",fg="red").place(x=2,y=5)


    lbl_name=Label(productFrame2,text="Product Name",font=("time new roman",15,"bold"),bg="white",fg="red").place(x=0,y=45)


    lbl_Search=Label(productFrame2,text="Search Product || By Name",font=("time new roman",15,"bold"),bg="white",fg="red").place(x=2,y=5)

    txt_Search=Entry(productFrame2,textvariable=self.var_search,font=("time new roman",15),bg="lightyellow",fg="red").place(x=140,y=47,width=150,height=22)

    btn_search=Button(productFrame2,text="Search",command=self.search,font=("time new
roman",15),bg="blue",fg="white",cursor="hand2").place(x=295,y=45,width=90,height=25)


    btn_show_all=Button(productFrame2,text="Show All",command=self.show,font=("time new
```

```
roman",15),bg="orange",fg="white",cursor="hand2").place(x=295,y=10,width=90,heig
ht=25)

   #=========== Product Detail Frame
=============================================

   ProductFrame3=Frame(productFrame1,bd=3,relief=RIDGE)

   ProductFrame3.place(x=2,y=140,width=398,height=385)


   scrolly=Scrollbar(ProductFrame3,orient=VERTICAL)

   scrollx=Scrollbar(ProductFrame3,orient=HORIZONTAL)


   self.product_table=ttk.Treeview(ProductFrame3,columns=("pid","name","price","qt
y","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")

   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("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)


   lbl_note=Label(productFrame1,text="Note:Enter 0 quantity to remove product from
the cart",font=("time new
roman",12),anchor='w',bg="white",fg="red").pack(side=BOTTOM,fill=X)
```

```
    #=================== 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=("time new
roman",15),bg="lightgray").pack(side=TOP,fill=X)

    lbl_name=Label(CustomerFrame,text="Name",font=("time new
roman",15),bg="white",fg="red").place(x=5,y=35)

    txt_name=Entry(CustomerFrame,textvariable=self.var_cname,font=("time new
roman",13),bg="lightyellow").place(x=80,y=35,width=180)


    lbl_contact=Label(CustomerFrame,text="Conatct No.",font=("time new
roman",15),bg="white",fg="red").place(x=270,y=35)

    txt_contact=Entry(CustomerFrame,textvariable=self.var_contact,font=("time new
roman",13),bg="lightyellow").place(x=380,y=35,width=140)

#====================Cal Cart button ==============

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

    Cal_Cart_Frame.place(x=420,y=190,width=530,height=360)

#=================== Calculatore Frame ==============

    self.var_cal_input=StringVar()

    Cal_Frame=Frame(Cal_Cart_Frame,bd=9,relief=RIDGE,bg="white")

    Cal_Frame.place(x=5,y=10,width=268,height=340)

    txt_cal_input=Entry(Cal_Frame,textvariable=self.var_cal_input,font=("time new
roman",15,"bold"),width=21,bd=10,relief=GROOVE,state="readonly",justify=RIGHT)

    txt_cal_input.grid(row=0,columnspan=4)


    btn_7=Button(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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_sub=Button(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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(Cal_Frame,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_mul=Button(Cal_Frame,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(Cal_Frame,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(Cal_Frame,text='c',font=("times new
roman",15,"bold"),command=self.clear_cal,bd=5,width=4,pady=15,cursor="hand2").gr
id(row=4,column=1)
```

```
    btn_eq=Button(Cal_Frame,text='=',font=("times new roman",15,"bold"),command=self.perform_cal,bd=5,width=4,pady=15,cursor="hand2").grid(row=4,column=2)

    btn_div=Button(Cal_Frame,text='/',font=("times new roman",15,"bold"),command=lambda:self.get_input('/'),bd=5,width=4,pady=15,cursor="hand2").grid(row=4,column=3)


#==================== Cart Frame ==============

    cart_Frame=Frame(Cal_Cart_Frame,bd=3,relief=RIDGE)

    cart_Frame.place(x=280,y=8,width=245,height=342)

    self.cartTitle=Label(cart_Frame,text="Cart Totalproduct:[0]",font=("time new roman",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=100)

    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 Frame===========================
    self.var_pid=StringVar()
    self.var_pname=StringVar()
    self.var_price=StringVar()
    self.var_qty=StringVar()
    self.var_stock=StringVar()

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

    lbl_p_name=Label(Add_CarrdwidgetsFrame,text="Product Name",font=("time new roman",15),bg="white").place(x=5,y=5)
    txt_p_name=Entry(Add_CarrdwidgetsFrame,textvariable=self.var_pname,font=("time new
roman",15),bg="lightyellow",state="readonly").place(x=5,y=35,width=190,height=22)

    lbl_p_price=Label(Add_CarrdwidgetsFrame,text="Price per quanity",font=("time new roman",15),bg="white").place(x=230,y=5)
    txt_p_price=Entry(Add_CarrdwidgetsFrame,textvariable=self.var_price,font=("time new
roman",15),bg="lightyellow",state="readonly").place(x=230,y=35,width=150,height=22)

    lbl_p_qty=Label(Add_CarrdwidgetsFrame,text="quanity",font=("time new roman",15),bg="white").place(x=390,y=5)
    txt_p_qty=Entry(Add_CarrdwidgetsFrame,textvariable=self.var_qty,font=("time new roman",15),bg="lightyellow").place(x=390,y=35,width=120,height=22)

    self.lbl_instock=Label(Add_CarrdwidgetsFrame,text="In Stock ",font=("time new roman",15),bg="white")
    self.lbl_instock.place(x=5,y=70)

    btn_clear_cart=Button(Add_CarrdwidgetsFrame,text="Clear",font=("time new roman",15),bg="green",cursor="hand2").place(x=180,y=70,width=150,height=30)
```

```
    btn_add_cart=Button(Add_CarrdwidgetsFrame,text="Add || Update",command=self.add_uppdate_cart,font=("time new roman",15),bg="orange",cursor="hand2").place(x=340,y=70,width=180,height=30)


    #============== billing ================================================

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

    billFrame.place(x=953,y=110,width=410,height=410)

    BTitle=Label(billFrame,text="Customer Bill Area",font=("time new roman",20,"bold"),bg="red",fg="white").pack(side=TOP,fill=X)

    scrolly=Scrollbar(billFrame,orient=VERTICAL)

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

    self.txt_bill_area=Text(billFrame,yscrollcommand=scrolly.set)

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

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

    #================= billing buttons ==========================

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

    billMenuFrame.place(x=953,y=520,width=410,height=140)


    self.lbl_amnt=Label(billMenuFrame,text="Bill Amount\n[0]",font=("time new roman",15,"bold"),bg="#3f51b5",fg="white")

    self.lbl_amnt.place(x=2,y=5,width=120,height=70)

    self.lbl_discount=Label(billMenuFrame,text="Discount\n[5%]",font=("time new roman",15,"bold"),bg="#8bc34a",fg="white")

    self.lbl_discount.place(x=124,y=5,width=120,height=70)

    self.lbl_net_pay=Label(billMenuFrame,text="Net Pay\n[0]",font=("time new roman",15,"bold"),bg="#607d8b",fg="white")

    self.lbl_net_pay.place(x=246,y=5,width=160,height=70)


    btn_print=Button(billMenuFrame,text="Print",cursor="hand2",font=("time new roman",15,"bold"),bg="lightgreen",fg="white")

    btn_print.place(x=2,y=80,width=120,height=50)

    btn_clear=Button(billMenuFrame,text="Clear All",cursor="hand2",font=("time new roman",15,"bold"),bg="gray",fg="white")

    btn_clear.place(x=124,y=80,width=120,height=50)
```

```
   btn_generate=Button(billMenuFrame,text="Generate/save
Bill",command=self.generate_bill,cursor="hand2",font=("time new
roman",12,"bold"),bg="#009688",fg="white")

   btn_generate.place(x=246,y=80,width=160,height=50)

#=================== footer ===================

   lbl_footer=Label(self.root,text="IMS.Inventory Management System || Developed by
Sumit\nfor any technical issue contact: 910XXXXX13 ",font=("times new
roman",12),bg="#4d636d",fg="white").pack(side=BOTTOM,fill=X)

   #self.show()

   #self.bill_top()

#==================== All Function
=====================================

#===================== get function ================================

  def get_input(self,num):

   xnum=self.var_cal_input.get()+str(num)

   self.var_cal_input.set(xnum)


  def clear_cal(self):

   self.var_cal_input.set('')


  def perform_cal(self):

   result=self.var_cal_input.get()

   self.var_cal_input.set(eval(result))

#===================== Show function
==============================

  def show(self):

   connection=cx_Oracle.connect('sumit/abhinav')

   cur=connection.Cursor()

   try:

      cur.execute("select pid,name,price,qty,status 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)}",parent=self.root)
 #=============search function
================================================
   def search(self):
    connection=cx_Oracle.connect('sumit/abhinav')
    cur=connection.Cursor()
    try:
        if self.var_search.get()=="":
            messagebox.showerror("error","Search input should be
required",parent=self.root)
        else:
            cur.execute("select pid,name,price,qty,status from product where name LIKE
'%"+self.var_search.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)}",parent=self.root)



   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])
```

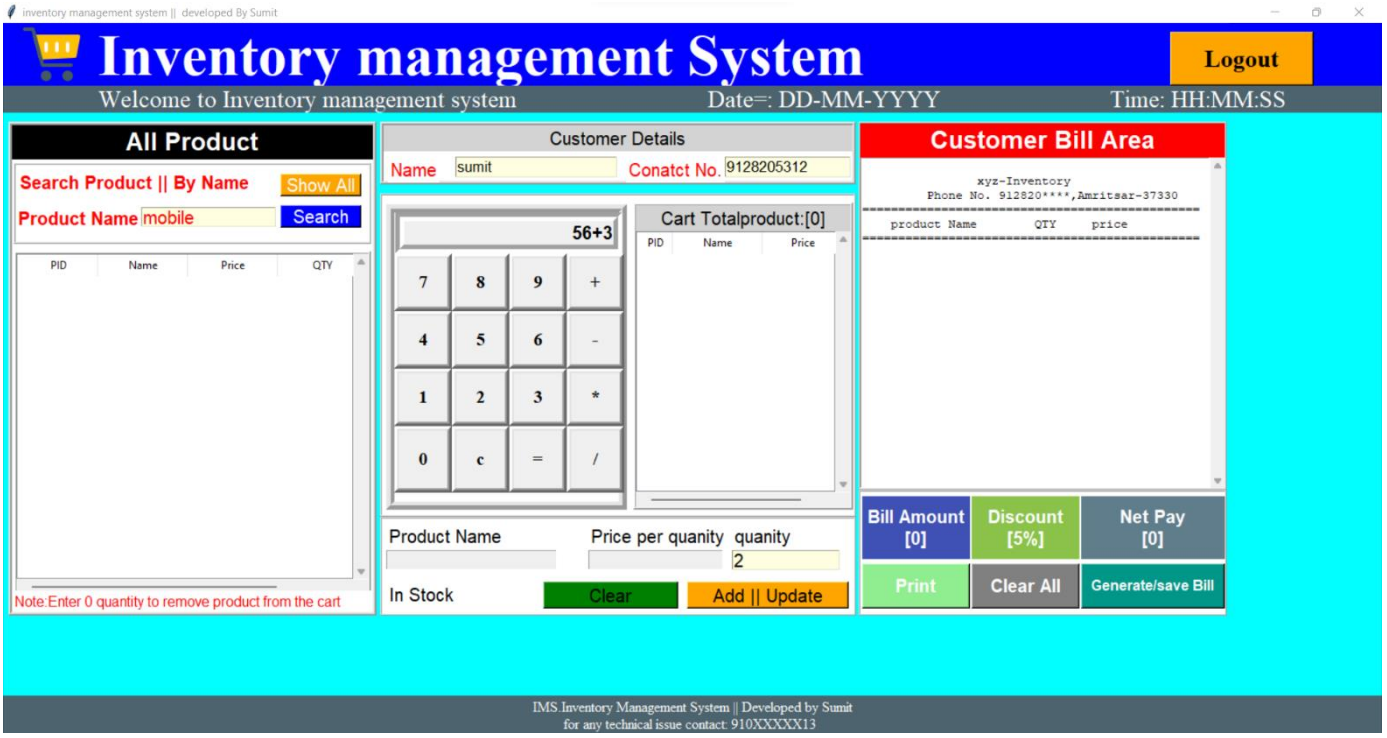
```
    self.var_price.set(row[2])
    self.lbl_instock.config(text=f"In Stock [{str(row[3])}]")
    self.var_stock.set(row[3])
    self.var_qty.set('1')


   def get_data_cart(self,ev):
    f=self.carttable.focus()
    content=(self.carttable.item(f))
    row=content['values']
    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_instock.config(text=f"In Stock [{str(row[4])}]")
    self.var_stock.set(row[4])

#=======================add and update function ======================
   def add_uppdate_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 int(self.var_qty.get())>int(self.var_stock.get()):
           messagebox.showerror('Error',"Invalid Quantity",parent=self.root)
       else:
          # price_cal=int(self.var_qty.get())*float(self.var_price.get())
           price_cal=self.var_price.get()
           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_=-1
        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\n Do you want to update| Remove from the Cart List",parent=self.root)
            if op==True:
                if self .var_qty.get()=="0":
                    self.cart_list.pop(index_)
                else:
                    self.cart_list[index_][2]=price_cal
                    self.cart_list[index_][3]=self.var_qty.get()

        else:
            self.cart_list.append(cart_data)
        self.show_cart()
        self.bill_updates()
#============bill update function =========
    def bill_updates(self):
        self.bill_amnt=0
        self.net_pay=0
        self.discount=0
        for row in self.cart_list:
            self.bill_amnt=self.bill_amnt+(float(row[2])*int(row[3]))
        self.discount=(self.bill_amnt*5)/100
        self.net_pay=self.bill_amnt- self.discount
        self.lbl_amnt.config(text=f'Bill Amnt\n{str(self.bill_amnt)}')
```

```
        self.lbl_amnt.config(text=f'Net pay\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.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)}",parent=self.root)
#===== generate bill function ============
    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)
     else:
        #=====Bill Top======
        self.bill_top()
        #======Bill Middle =======
        self.bill_middle()
        #=====Bill Button =========
        self.bill_bottom()
        #pass

    def bill_top(self):
     invoice=int(time.strftime("%H%M%S"))+int(time.strftime("%d%m%y"))
     #print(invoice)
     bill_top_temp=f'''
\t\txyz-Inventory
\t Phone No. 912820****,Amritsar-37330
{str("="*47)}
```

```
    product Name\t\t\tQTY\tprice
{str("="*47)}
    '''
    self.txt_bill_area.delete('1.0',END)
    self.txt_bill_area.insert('1.0',bill_top_temp)


   def bill_bottom(self):
        bill_bottom_temp=f'''
{str("="*47)}
Bill Amount\t\t\t\tRs.{self.bill_amnt}
Discount\t\t\t\tRs.{self.discount}
Net Pay\t\t\t\tRs.{self.net_pay}
{str("="*47)}\n
    '''
        self.txt_bill_area.insert('1.0',bill_bottom_temp)


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




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

 SNAPSHOTS OF BILLING DETAIL.



Bibliography or references

●BOOKS

●Programming and Problem Solving With PYTHON BY Ashok Namdev Kamthane And Amit Ashok Kamthane.

●Fundamentails of DBMS By Anshuman Sharma,Anurag Gupta and Jagmohan Mago.

●Web URLS

https://www.Programming-Problem-Solving-Python-Kamthane-ebook/dp/B07XH3BVWG

https://www.ebooknetworking.net/ebooks/fundamentals-of-dbms-by-lakhanpal-publishers.html

https://github.com/SumitKumargiri/Inventory-management-system-python-with-oracle/edit/main/README.md