STOCK MANAGEMENT SYSTEM

MINOR PROJECT REPORT

By

SAI DINESH (RA2211029010016)
RAHUL CHOWDARY (RA2211029010024)

Under the guidance of

Dr. Sundarrajan M

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING PRACTICE

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC203P** ADVANCED **PROGRAMMING PRACTICE** entitled in "**Stock Management System**" is the bonafide work of **SAI DINESH (RA2211029010016),RAHUL CHOWDARY (RA2211029010024)**who carried out the work under my supervision.

SIGNATURE

Dr. Sundarrajan M

Assistant Professor

CINTEL

SRM Institute of Science and Technology

Kattankulathur

ABSTRACT

A stock management system is a vital component of efficient inventory control in businesses across various industries. It serves as the backbone for tracking and managing inventory levels, ensuring that items are readily available when needed while minimizing excess stock to control costs. This system incorporates various tools, including barcoding, automated replenishment, and demand forecasting, to address the complexities of inventory accuracy and demand prediction. Additionally, it helps organizations navigate challenges such as supply chain disruptions, the management of inventory across multiple sales channels, vendor coordination, and data security. Ultimately, a well-implemented stock management system empowers businesses to streamline their operations, enhance customer satisfaction, and adapt to the dynamic demands of the modern marketplace.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to my Course project Faculty **Dr. Sundarrajan M, Assistant Professor,** Networking & Communications for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD, Dr K. Annapurani Panaiyappan**, Professor, Department of Networking and Communications and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1. INTRODUCTION

In today's dynamic business landscape, effective inventory management is crucial for ensuring business growth and profitability. A robust stock management system plays a pivotal role in streamlining inventory operations, optimizing stock levels, and reducing costs. It serves as a centralized platform for tracking goods throughout the supply chain, from receiving and storing to requisitioning and selling. By automating manual tasks, enhancing data accuracy, and providing real-time insights, stock management systems empower businesses to make informed decisions, minimize stockouts, and maximize profits.

A stock management system is an essential tool for any business that trades or sells stocks. These systems help businesses to track their inventory of stocks, manage their orders, and fulfill their customer orders. There are many different types of stock management systems available, each with its own strengths and weaknesses. This helps us to access and manage the information easily. This also helps to verify the stock currently available with them and to update the stock when necessary. This also reduce the time to search the product from the current available stock. The role of an inventory system is to track your products and supplies. Inventory management is the process of controlling of the ordering, storage, and use of components that a company uses in the production of the products it sells.

.

1.1 MOTIVATION

The motivation for stock management system is to optimize the inventory levels and operations of a business. Some of the benefits of stock management system are:

→**Cost reduction:** Stock management system helps to reduce the costs of holding, ordering, and storing inventory. It also helps to avoid losses due to obsolescence, spoilage, theft, and damage of inventory.

→**Customer satisfaction:** Stock management system ensures that the products are available when the customers want them. It also enables faster and cheaper delivery of products to the customers, thus enhancing customer loyalty and retention.

→**Efficiency improvement:** Stock management system streamlines the processes of inventory planning, tracking, allocation, and replenishment. It also improves the accuracy and reliability of inventory data and reduces the errors and wastage of inventory.

→**Competitive advantage:** Stock management system enables a business to respond quickly and effectively to the changing market demands and customer preferences. It also helps to gain insights into the inventory performance and trends, and make informed decisions for inventory optimization.

1.2 OBJECTIVE

The objective for stock management system is to design and implement a software solution that can help a business to manage its inventory effectively and efficiently. The stock management system should have the following features:

→It should allow the user to input, update, and delete the inventory data, such as product name, quantity, price, category, location, supplier, etc.

→It should provide the user with various reports and analytics on the inventory status, such as inventory level, turnover rate, reorder point, safety stock, etc.

→It should enable the user to set and adjust the inventory policies, such as minimum and maximum inventory levels, reorder quantities, lead times, etc.

→It should support the user in making inventory decisions, such as when to order, how much to order, which supplier to choose, etc.

→It should integrate with other systems, such as accounting, sales, purchasing, etc., to ensure data consistency and accuracy.

The objective for stock management system is to achieve the following goals:

→To reduce the inventory costs and increase the profitability of the business.

→To improve the customer satisfaction and loyalty by meeting their demands and expectations.

→To enhance the efficiency and productivity of the inventory operations and processes.

→To gain a competitive edge in the market by adapting to the changing environment and customer needs.

1.3 PROBLEM STATEMENT

In is very difficult to manage the records of every product manually It is very time-consuming process. In case any problems occur like missing the records which are saved in offline then many problems arises and it becomes difficult to get back the data so managing the details is not preferable.

Overstocking: The company often orders more products than it needs, resulting in high inventory holding costs, wastage of space, and risk of obsolescence.

Understocking: The company sometimes runs out of stock of some products, leading to lost sales, customer dissatisfaction, and damage to reputation.

Inaccuracy: The company relies on manual processes and outdated systems to record and track its inventory, causing errors, discrepancies, and inefficiencies.

Lack of visibility: The company has no real-time information on its inventory levels, demand patterns, and supplier performance, making it difficult to plan and optimize its inventory operations.

1.4 CHALLENGES

A stock management system plays a crucial role in the efficient operation of businesses, particularly in retail, manufacturing, and distribution sectors. However, it faces several challenges that can impact its effectiveness. One significant challenge is inventory accuracy. Maintaining precise inventory records in real-time is essential, but discrepancies can arise due to factors like theft, errors in data entry, and items becoming obsolete. Inaccurate inventory levels can lead to overstocking or understocking, resulting in financial losses and dissatisfied customers.

- To explain that this software is better than manual system.
- To explain the detail process involved in the software.
- To develop a software which easy to use and avoid complexity.
- The software should satisfy the user needs.
- To provide accurate database services.
- To make sure that the software works at the user place (user environment).
- Mis-communication between the sales and the supply chain management team
- Handling multi-store scenarios: In a scenario where a business operates multiple stores, managing stock across these locations can be challenging. Coordinating inventory levels, transportation schedules, and other logistics aspects.
- Accurate tracking and monitoring: One of the most critical aspects of stock management is tracking and monitoring the stock in real-time. Ensuring accuracy and consistency across the entire stock can be challenging, especially in cases of data entry errors or inconsistent inventory tracking.

2. REQUIREMENTS

2.1 Requirement Analysis

The specific requirements of a stock management system will vary depending on the size and type of business, but there are some common requirements that most businesses will need:

- Product information management: The system should be able to track basic product information such as product name, description, SKU, barcode, and price.
- Stock tracking: The system should be able to track stock levels in real time, including stock on hand, stock on order, and stock committed to orders.
- Inventory reporting: The system should be able to generate reports on inventory levels, stock movements, and other inventory-related data.
- Purchase order management: The system should be able to generate purchase orders for vendors based on predetermined stock levels or demand forecasts.
- Demand forecasting: The system should be able to forecast demand for products based on historical sales data and other factors.
- Integration with other systems: The stock management system should be able to integrate with other business systems such as accounting systems, e-commerce platforms, and warehouse management systems.

2.2 Hardware Requirement

The hardware requirements for a stock management system depend on the type and size of the system, as well as the features and functionalities that it offers. However, some of the common hardware components that are needed for a stock management system are:

- Computers: These are the devices that run the software and allow the user to access and manage the inventory data. Computers should have a compatible operating system, browser, and internet connection to run the system smoothly.
- Printers: These are devices that can print the inventory documents, such as invoices, receipts, labels, reports, etc. Printers can help to provide physical records and proofs of the inventory transactions and operations, as well as facilitate the communication and coordination with the customers, suppliers, and other stakeholders. Printers can be connected to the computers or tablets via USB, Bluetooth, or Wi-Fi, depending on the system and the device.
- Servers: These are devices that store and process the inventory data and transactions, and provide the access and functionality of the system to the users. Servers can be either rented or purchased, depending on the system and the business needs.
- PROCESSOR: 64-bit
- ROM: 2GB
- RAM: 4GB

Software Requirements:

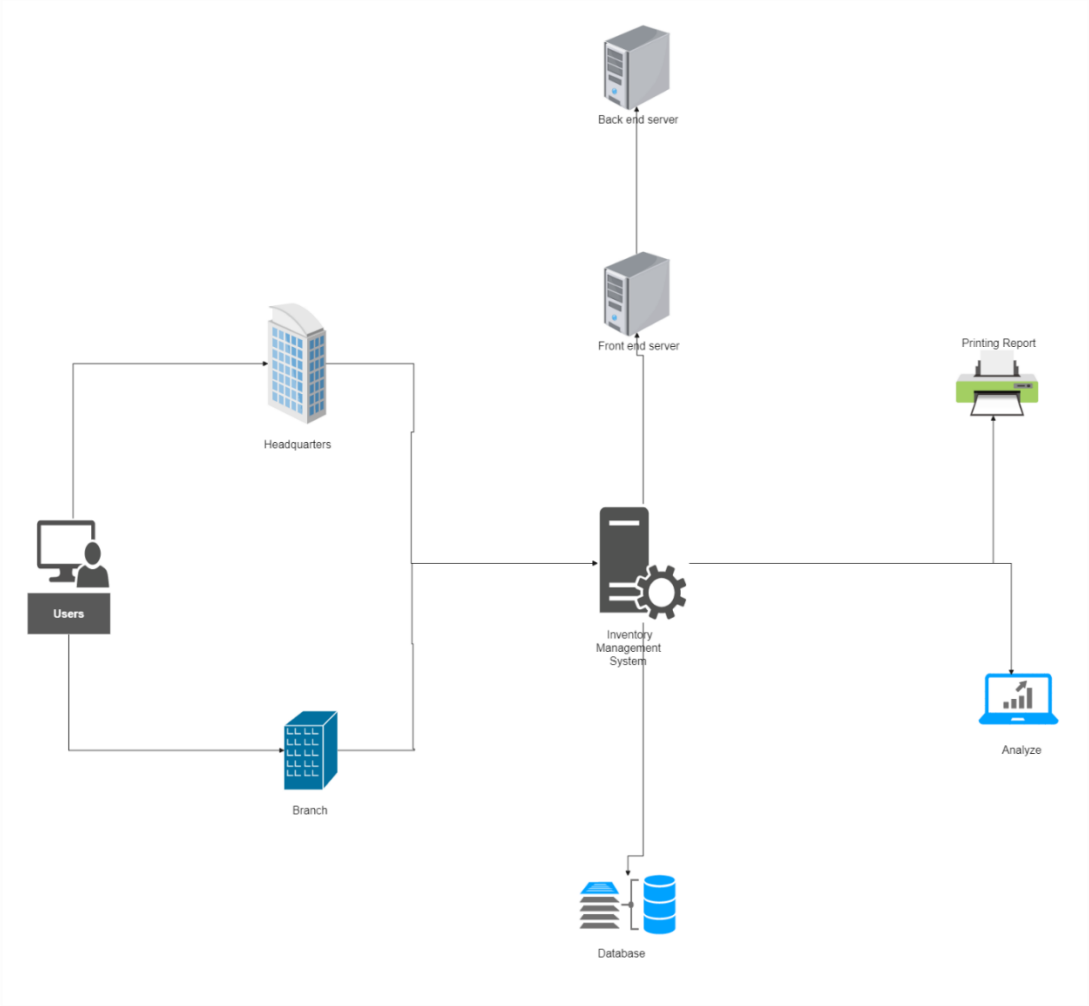
OPERATING SYSTEM: WINDOWS8/10

FRONTEND: JAVANETBEANS

BACKEND: MYSQL

3. ARCHITECTURE AND DESIGN

The stock management system architecture is as follows:

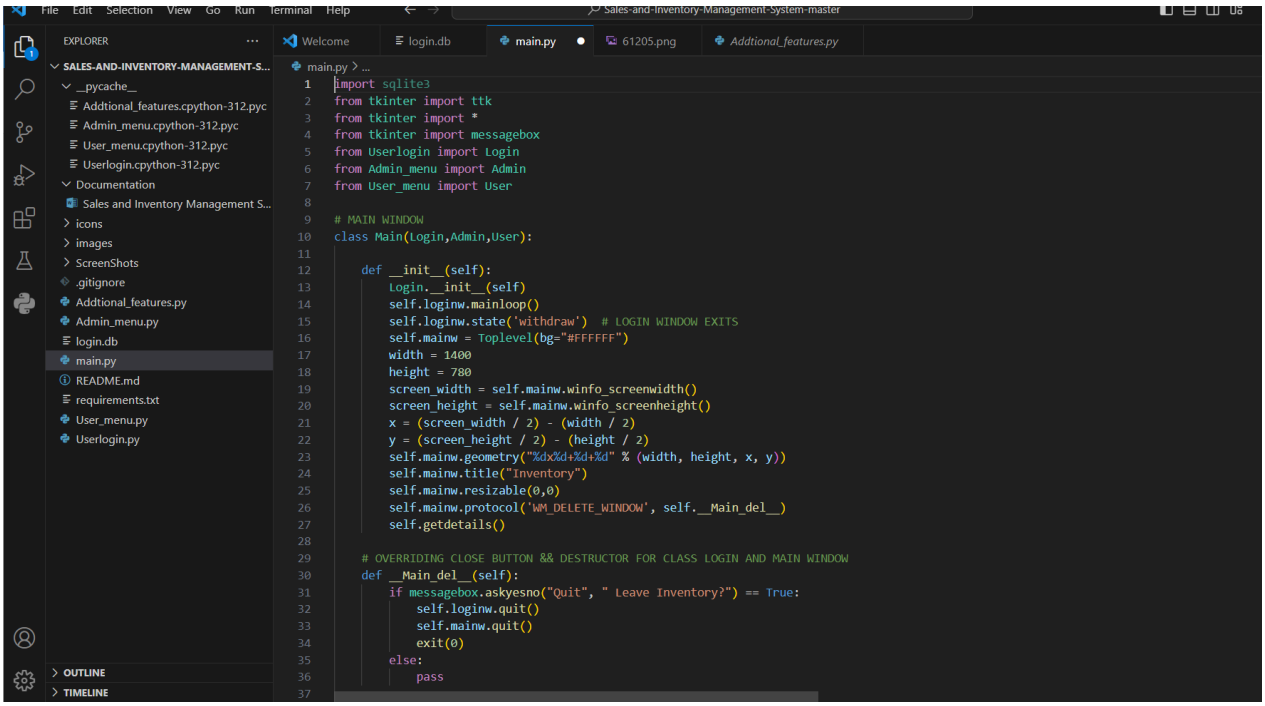


- The architecture of stock management system uses client server model.
- The design or architectural specification for the stock management system is Java since the JSP architecture will be used.
- The Java Database Connectivity (JDBC) will use the MySQL Connector for the server to communicate to the stock database.
- Upon receiving requests from the clients, the server will issue transactions to the MySQL database

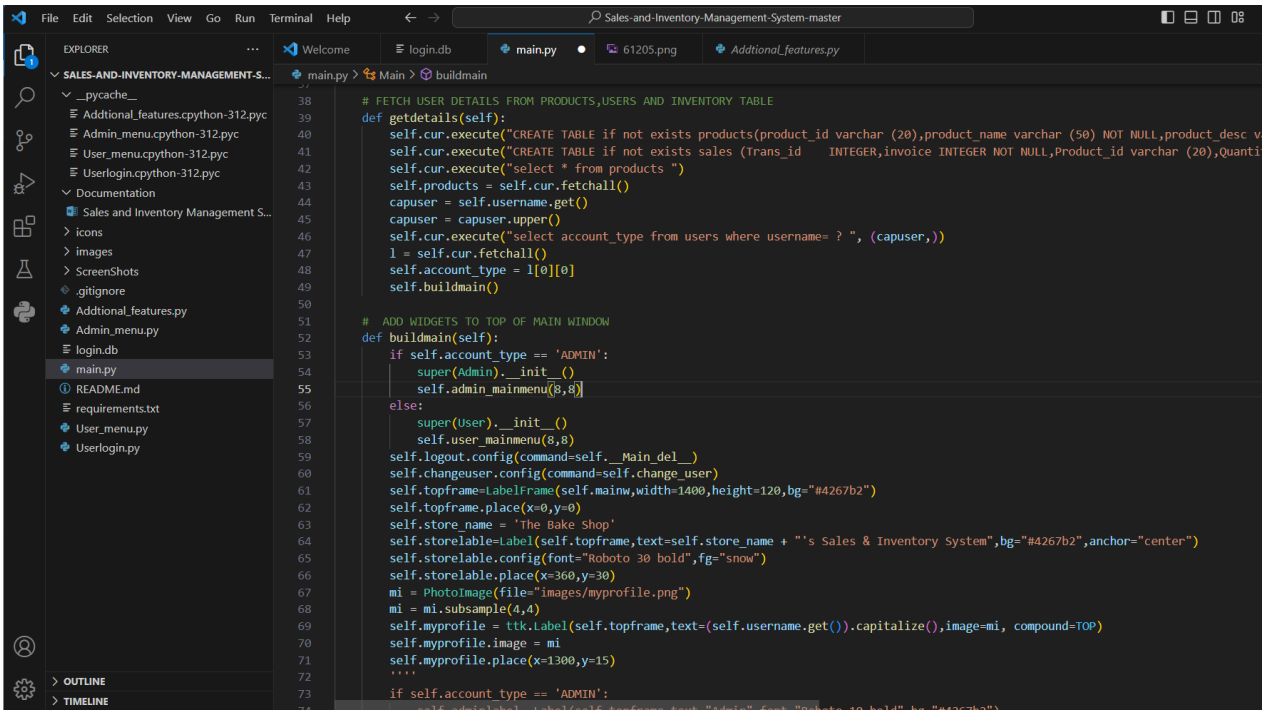
4. PYTHON

5.1 Implementation

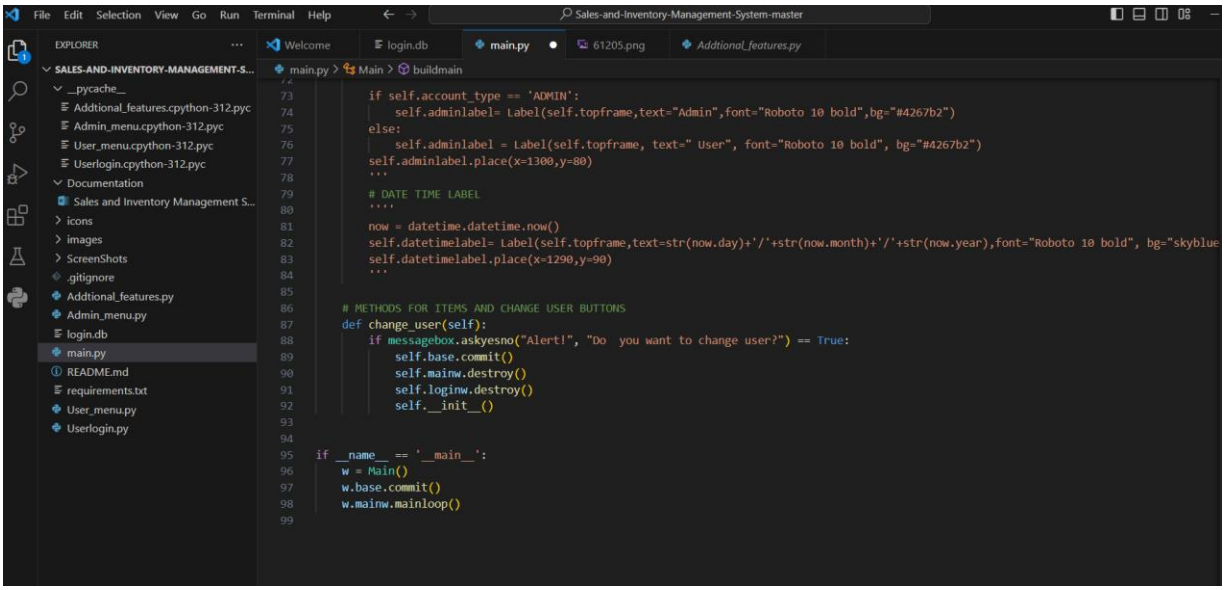
Main.py



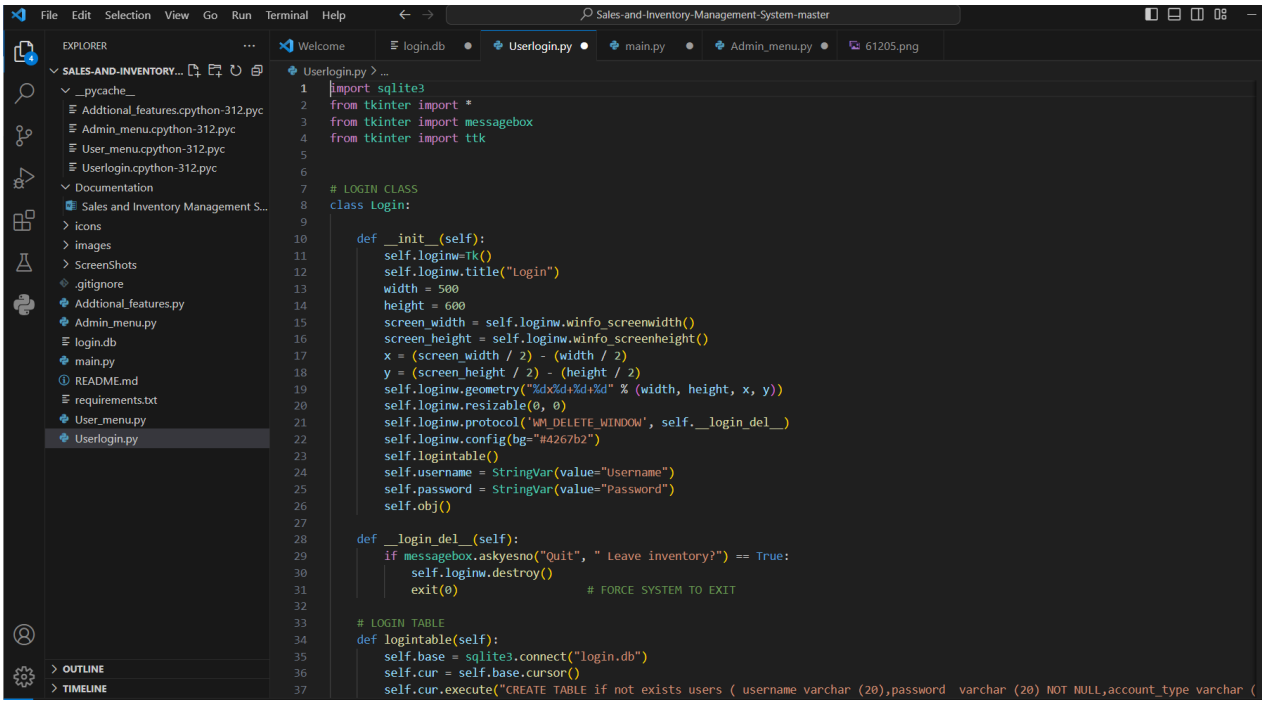
```
import sqlite3
from tkinter import ttk
from tkinter import *
from tkinter import messagebox
from Userlogin import Login
from Admin_menu import Admin
from User_menu import User

# MAIN WINDOW
class Main(Login,Admin,User):

    def __init__(self):
        Login.__init__(self)
        self.loginw.mainloop()
        self.loginw.state('withdraw')  # LOGIN WINDOW EXITS
        self.mainw = Toplevel(bg="#FFFFFF")
        width = 1400
        height = 780
        screen_width = self.mainw.winfo_screenwidth()
        screen_height = self.mainw.winfo_screenheight()
        x = (screen_width / 2) - (width / 2)
        y = (screen_height / 2) - (height / 2)
        self.mainw.geometry("%dx%d+%d+%d" % (width, height, x, y))
        self.mainw.title("Inventory")
        self.mainw.resizable(0,0)
        self.mainw.protocol('WM_DELETE_WINDOW', self.__Main_del__)
        self.getdetails()

    # OVERRIDING CLOSE BUTTON && DESTRUCTOR FOR CLASS LOGIN AND MAIN WINDOW
    def __Main_del__(self):
        if messagebox.askyesno("Quit", " Leave Inventory?") == True:
            self.loginw.quit()
            self.mainw.quit()
            exit(0)
        else:
            pass
```

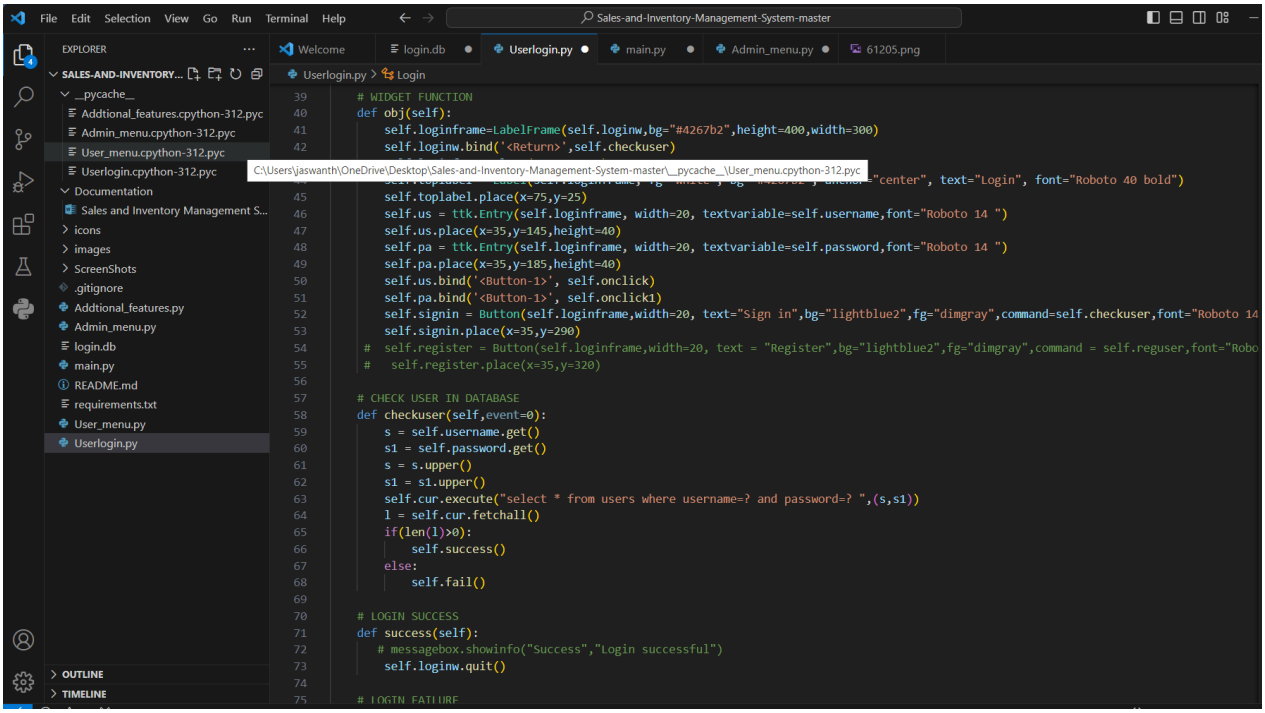


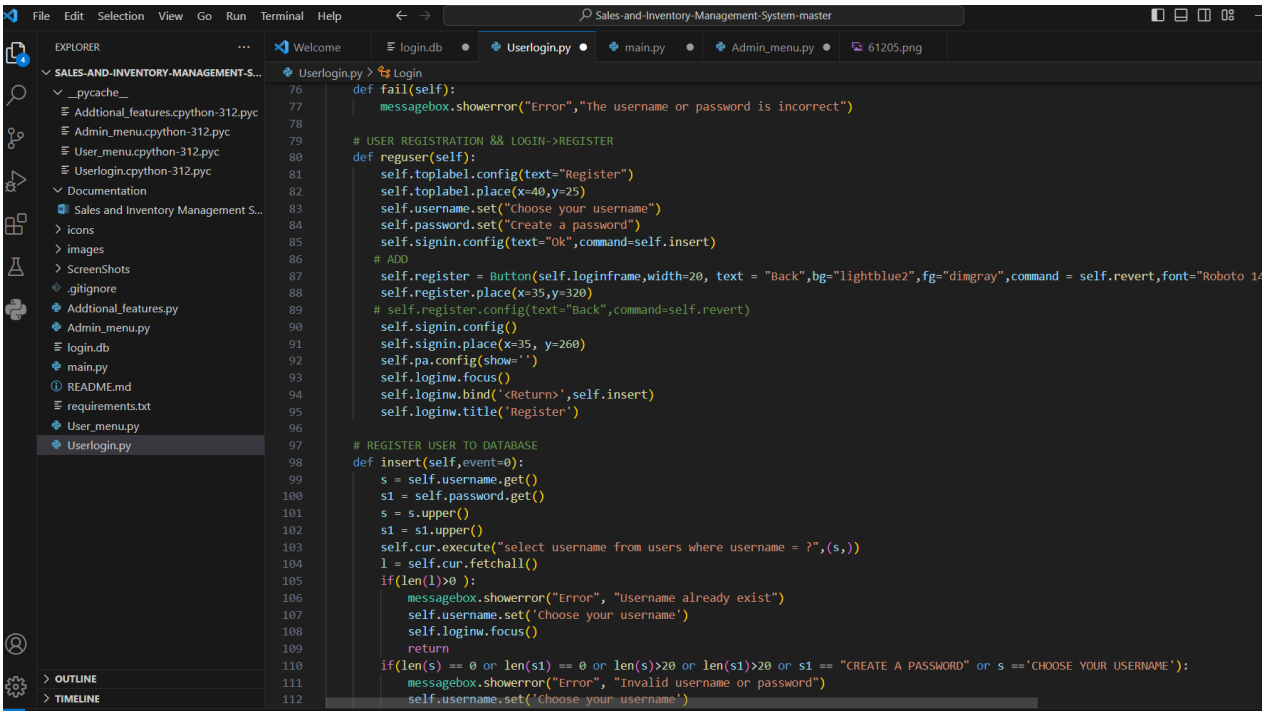
```
    # FETCH USER DETAILS FROM PRODUCTS,USERS AND INVENTORY TABLE
    def getdetails(self):
        self.cur.execute("CREATE TABLE if not exists products(product_id varchar (20),product_name varchar (50) NOT NULL,product_desc v
        self.cur.execute("CREATE TABLE if not exists sales (Trans_id    INTEGER,invoice INTEGER NOT NULL,Product_id varchar (20),Quanti
        self.cur.execute("select * from products ")
        self.products = self.cur.fetchall()
        capuser = self.username.get()
        capuser = capuser.upper()
        self.cur.execute("select account_type from users where username= ? ", (capuser,))
        l = self.cur.fetchall()
        self.account_type = l[0][0]
        self.buildmain()

    #  ADD WIDGETS TO TOP OF MAIN WINDOW
    def buildmain(self):
        if self.account_type == 'ADMIN':
            super(Admin).__init__()
            self.admin_mainmenu(8,8)
        else:
            super(User).__init__()
            self.user_mainmenu(8,8)
        self.logout.config(command=self.__Main_del__)
        self.changeuser.config(command=self.change_user)
        self.topframe=LabelFrame(self.mainw,width=1400,height=120,bg="#4267b2")
        self.topframe.place(x=0,y=0)
        self.store_name = 'The Bake Shop'
        self.storelable=Label(self.topframe,text=self.store_name + "'s Sales & Inventory System",bg="#4267b2",anchor="center")
        self.storelable.config(font="Roboto 30 bold",fg="snow")
        self.storelable.place(x=360,y=30)
        mi = PhotoImage(file="images/myprofile.png")
        mi = mi.subsample(4,4)
        self.myprofile = ttk.Label(self.topframe,text=(self.username.get()).capitalize(),image=mi, compound=TOP)
        self.myprofile.image = mi
        self.myprofile.place(x=1300,y=15)
        ''''
        if self.account_type == 'ADMIN':
```

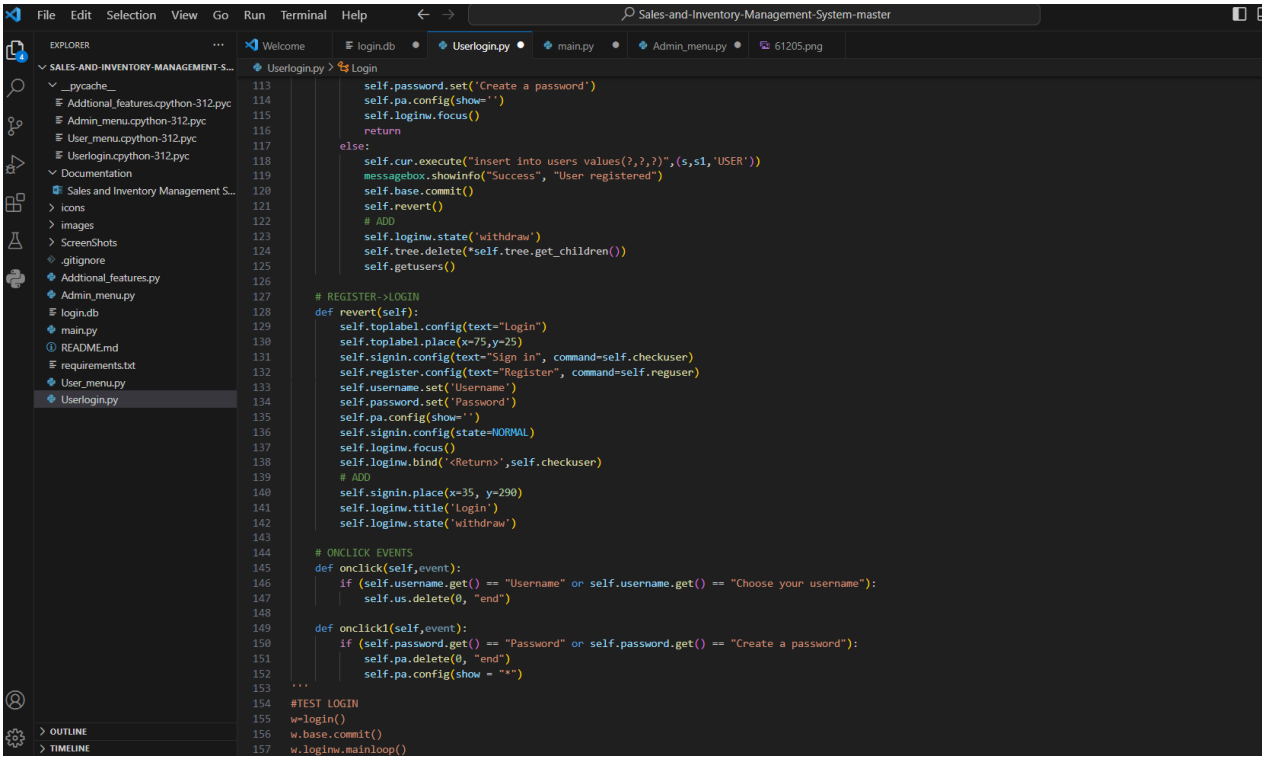


Userlogin.py

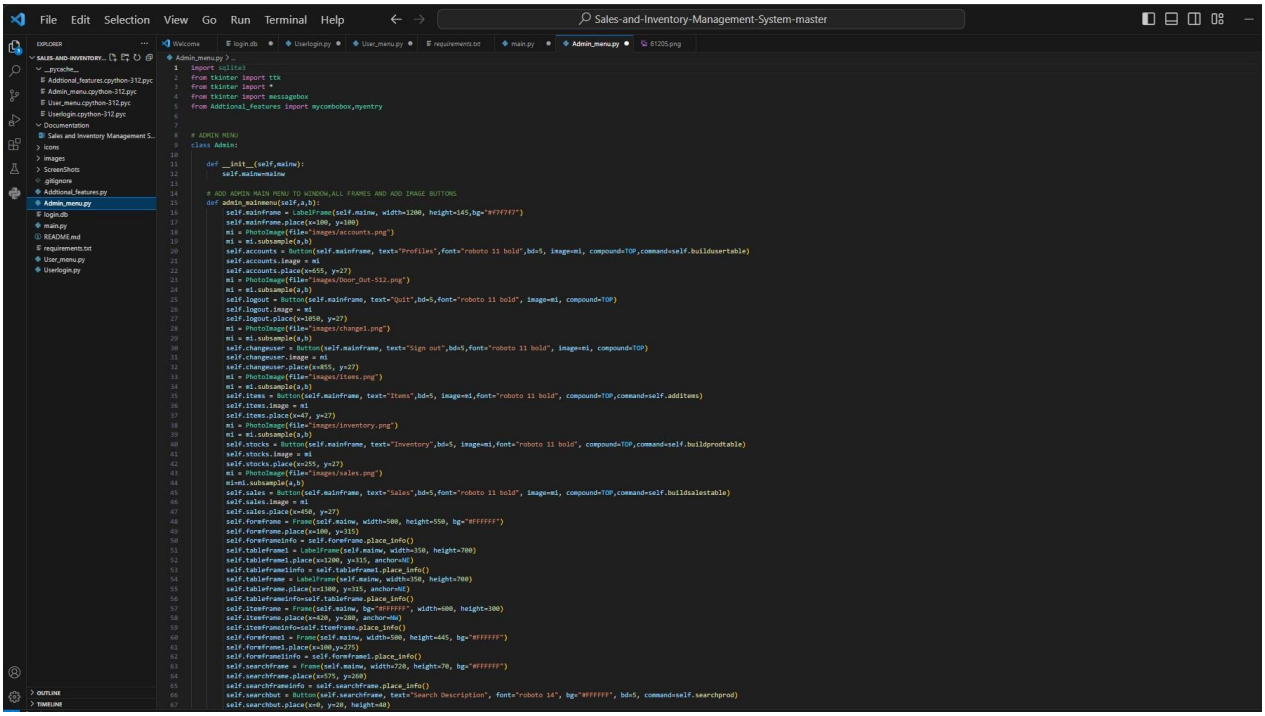


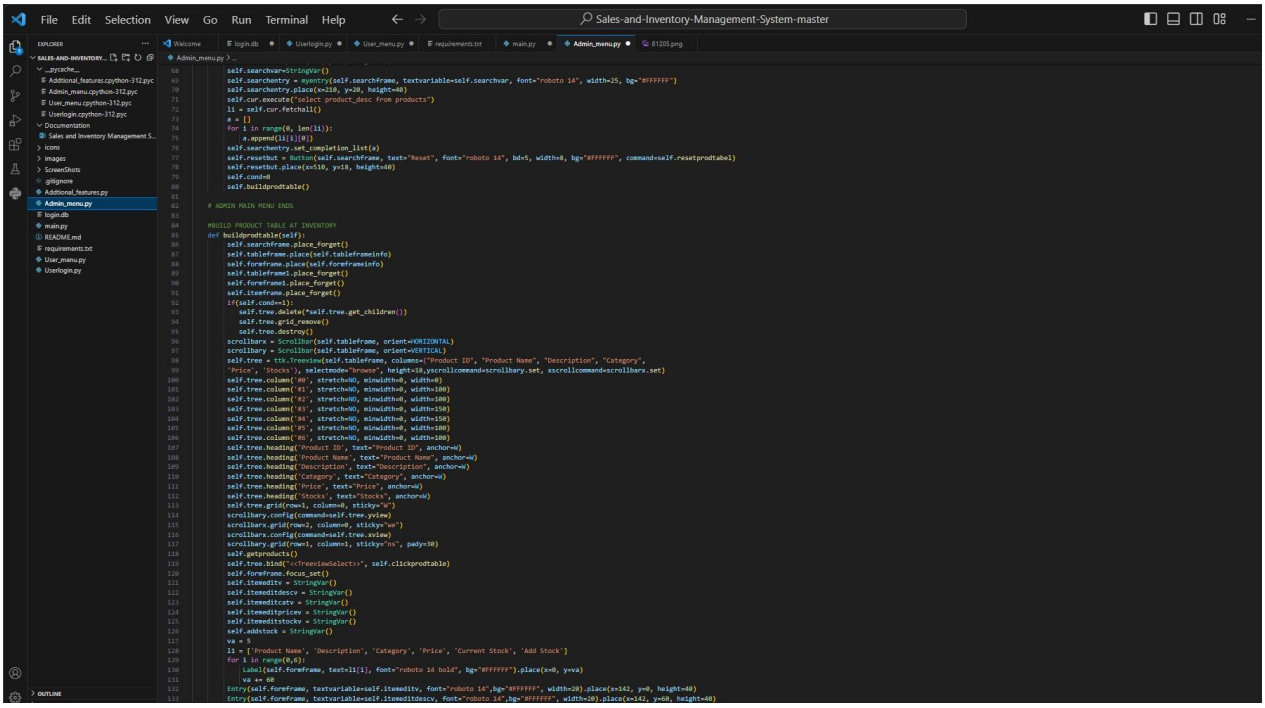

File Edit Selection View Go Run Terminal Help
Sales-and-Inventory-Management-System-master
EXPLORER
SALES-AND-INVENTORY...
Welcome
login.db
Userlogin.py
main.py
Admin_menu.py
61205.png
Userlogin.py > Login
__pycache__
Addtional_features.cpython-312.pyc
Admin_menu.cpython-312.pyc
User_menu.cpython-312.pyc
Userlogin.cpython-312.pyc
Documentation
Sales and Inventory Management S...
icons
images
ScreenShots
.gitignore
Addtional_features.py
Admin_menu.py
login.db
main.py
README.md
requirements.txt
User_menu.py
Userlogin.py
C:\Users\jaswanth\OneDrive\Desktop\Sales-and-Inventory-Management-System-master__pycache__\User_menu.cpython-312.pyc
OUTLINE
TIMELINE
# WIDGET FUNCTION
def obj(self):
self.loginframe=LabelFrame(self.loginw,bg="#4267b2",height=400,width=300)
self.loginw.bind('<Return>',self.checkuser)
center", text="Login", font="Roboto 40 bold")
self.toplabel.place(x=75,y=25)
self.us = ttk.Entry(self.loginframe, width=20, textvariable=self.username,font="Roboto 14 ")
self.us.place(x=35,y=145,height=40)
self.pa = ttk.Entry(self.loginframe, width=20, textvariable=self.password,font="Roboto 14 ")
self.pa.place(x=35,y=185,height=40)
self.us.bind('<Button-1>', self.onclick)
self.pa.bind('<Button-1>', self.onclick1)
self.signin = Button(self.loginframe,width=20, text="Sign in",bg="lightblue2",fg="dimgray",command=self.checkuser,font="Roboto 14
self.signin.place(x=35,y=290)
# self.register = Button(self.loginframe,width=20, text = "Register",bg="lightblue2",fg="dimgray",command = self.reguser,font="Robo
# self.register.place(x=35,y=320)
# CHECK USER IN DATABASE
def checkuser(self,event=0):
s = self.username.get()
s1 = self.password.get()
s = s.upper()
s1 = s1.upper()
self.cur.execute("select * from users where username=? and password=? ",(s,s1))
l = self.cur.fetchall()
if(len(l)>0):
self.success()
else:
self.fail()
# LOGIN SUCCESS
def success(self):
# messagebox.showinfo("Success","Login successful")
self.loginw.quit()
# LOGIN FAILURE


File Edit Selection View Go Run Terminal Help
Sales-and-Inventory-Management-System-master
EXPLORER
SALES-AND-INVENTORY-MANAGEMENT-S...
Welcome
login.db
Userlogin.py
main.py
Admin_menu.py
61205.png
Userlogin.py > Login
__pycache__
Addtional_features.cpython-312.pyc
Admin_menu.cpython-312.pyc
User_menu.cpython-312.pyc
Userlogin.cpython-312.pyc
Documentation
Sales and Inventory Management S...
icons
images
ScreenShots
.gitignore
Addtional_features.py
Admin_menu.py
login.db
main.py
README.md
requirements.txt
User_menu.py
Userlogin.py
OUTLINE
TIMELINE
def fail(self):
messagebox.showerror("Error","The username or password is incorrect")
# USER REGISTRATION && LOGIN->REGISTER
def reguser(self):
self.toplabel.config(text="Register")
self.toplabel.place(x=40,y=25)
self.username.set("Choose your username")
self.password.set("Create a password")
self.signin.config(text="Ok",command=self.insert)
# ADD
self.register = Button(self.loginframe,width=20, text = "Back",bg="lightblue2",fg="dimgray",command = self.revert,font="Roboto 14
self.register.place(x=35,y=320)
# self.register.config(text="Back",command=self.revert)
self.signin.config()
self.signin.place(x=35, y=260)
self.pa.config(show='')
self.loginw.focus()
self.loginw.bind('<Return>',self.insert)
self.loginw.title('Register')
# REGISTER USER TO DATABASE
def insert(self,event=0):
s = self.username.get()
s1 = self.password.get()
s = s.upper()
s1 = s1.upper()
self.cur.execute("select username from users where username = ?",(s,))
l = self.cur.fetchall()
if(len(l)>0):
messagebox.showerror("Error", "Username already exist")
self.username.set('Choose your username')
self.loginw.focus()
return
if(len(s) == 0 or len(s1) == 0 or len(s)>20 or len(s1)>20 or s1 == "CREATE A PASSWORD" or s =='CHOOSE YOUR USERNAME'):
messagebox.showerror("Error", "Invalid username or password")
self.username.set('Choose your username')

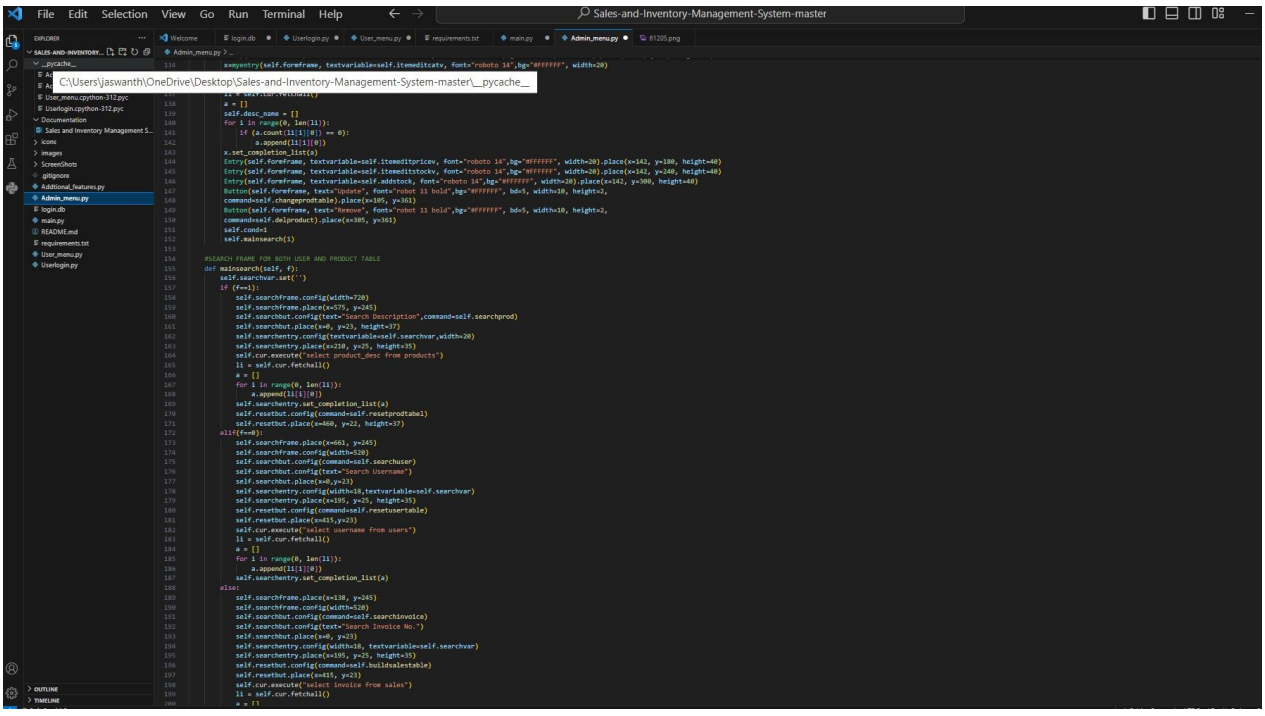


Admin_menu.py

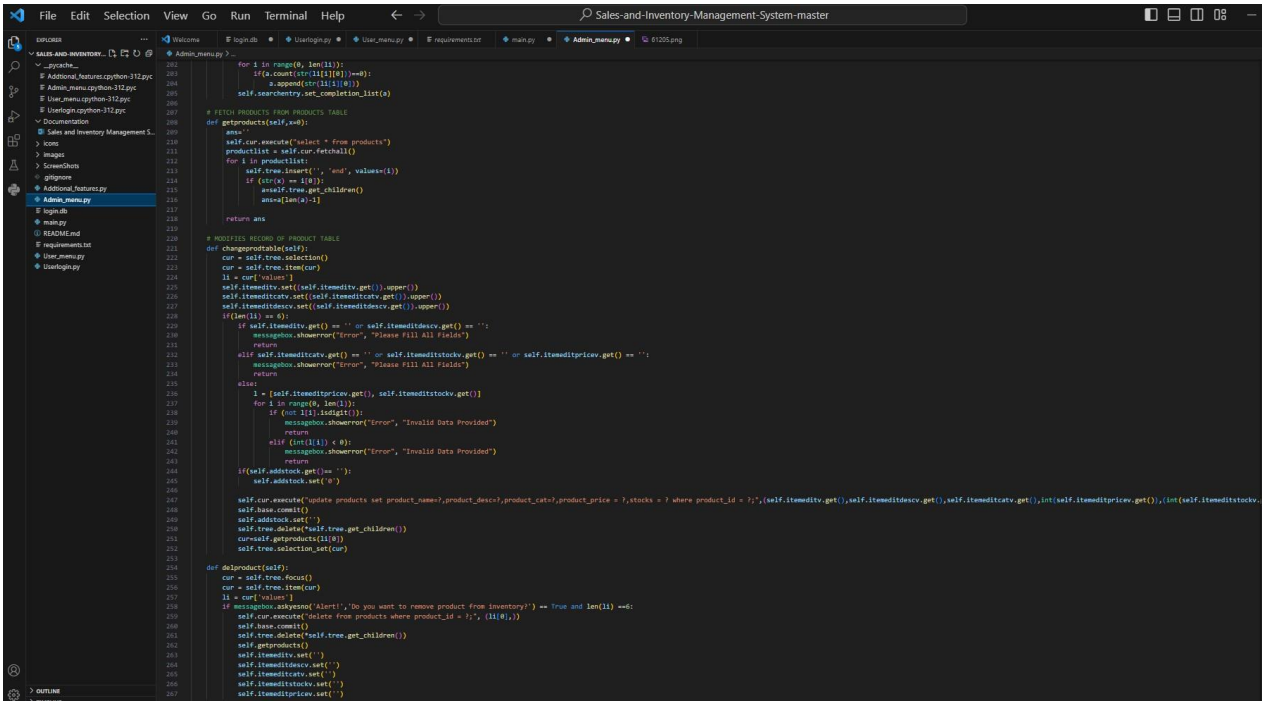




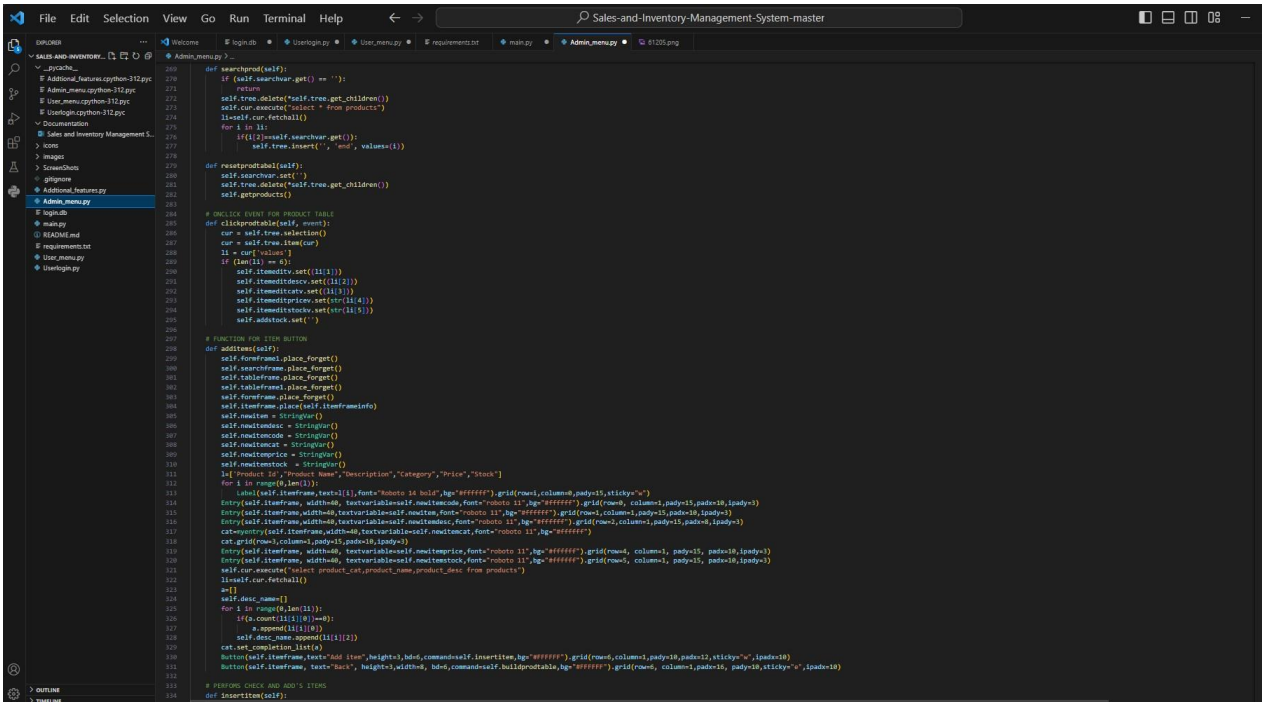
```
        self.searchvar=StringVar()
        self.searchentry = myentry(self.searchframe, textvariable=self.searchvar, font="roboto 14", width=25, bg="#FFFFFF")
        self.searchentry.place(x=210, y=20, height=40)
        self.cur.execute("select product_desc from products")
        li = self.cur.fetchall()
        a = []
        for i in range(0, len(li)):
            a.append(li[i][0])
        self.searchentry.set_completion_list(a)
        self.resetbut = Button(self.searchframe, text="Reset", font="roboto 14", bd=5, width=8, bg="#FFFFFF", command=self.resetprodtable1)
        self.resetbut.place(x=510, y=18, height=40)
        self.cond=0
        self.buildprodtable()

    # ADMIN MAIN MENU ENDS

    #BUILD PRODUCT TABLE AT INVENTORY
    def buildprodtable(self):
        self.searchframe.place_forget()
        self.tableframe.place(self.tableframeinfo)
        self.foreframe.place(self.foreframeinfo)
        self.tableframe1.place_forget()
        self.foreframe1.place_forget()
        self.itemframe.place_forget()
        if(self.cond==1):
            self.tree.delete(*self.tree.get_children())
            self.tree.grid_remove()
            self.tree.destroy()
        scrollbarx = Scrollbar(self.tableframe, orient=HORIZONTAL)
        scrollbary = Scrollbar(self.tableframe, orient=VERTICAL)
        self.tree = ttk.Treeview(self.tableframe, columns=("Product ID", "Product Name", "Description", "Category",
        'Price', 'Stocks'), selectmode="browse", height=18,yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
        self.tree.column('#0', stretch=NO, minwidth=0, width=0)
        self.tree.column('#1', stretch=NO, minwidth=0, width=100)
        self.tree.column('#2', stretch=NO, minwidth=0, width=100)
        self.tree.column('#3', stretch=NO, minwidth=0, width=150)
        self.tree.column('#4', stretch=NO, minwidth=0, width=150)
        self.tree.column('#5', stretch=NO, minwidth=0, width=100)
        self.tree.column('#6', stretch=NO, minwidth=0, width=100)
        self.tree.heading('Product ID', text="Product ID", anchor=W)
        self.tree.heading('Product Name', text="Product Name", anchor=W)
        self.tree.heading('Description', text="Description", anchor=W)
        self.tree.heading('Category', text="Category", anchor=W)
        self.tree.heading('Price', text="Price", anchor=W)
        self.tree.heading('Stocks', text="Stocks", anchor=W)
        self.tree.grid(row=1, column=0, sticky="W")
        scrollbary.config(command=self.tree.yview)
        scrollbarx.grid(row=2, column=0, sticky="we")
        scrollbarx.config(command=self.tree.xview)
        scrollbary.grid(row=1, column=1, sticky="ns", padx=30)
        self.getproducts()
        self.tree.bind("<<TreeviewSelect>>", self.clickprodtable)
        self.foreframe.focus_set()
        self.itemeditv = StringVar()
        self.itemeditdescv = StringVar()
        self.itemeditcatv = StringVar()
        self.itemeditpricev = StringVar()
        self.itemeditstockv = StringVar()
        self.addstock = StringVar()
        va = 5
        li = ['Product Name', 'Description', 'Category', 'Price', 'Current Stock', 'Add Stock']
        for i in range(0,6):
            Label(self.foreframe, text=li[i], font="roboto 14 bold", bg="#FFFFFF").place(x=0, y=va)
            va += 60
        Entry(self.foreframe, textvariable=self.itemeditv, font="roboto 14",bg="#FFFFFF", width=20).place(x=142, y=0, height=40)
        Entry(self.foreframe, textvariable=self.itemeditdescv, font="roboto 14",bg="#FFFFFF", width=20).place(x=142, y=60, height=40)
```



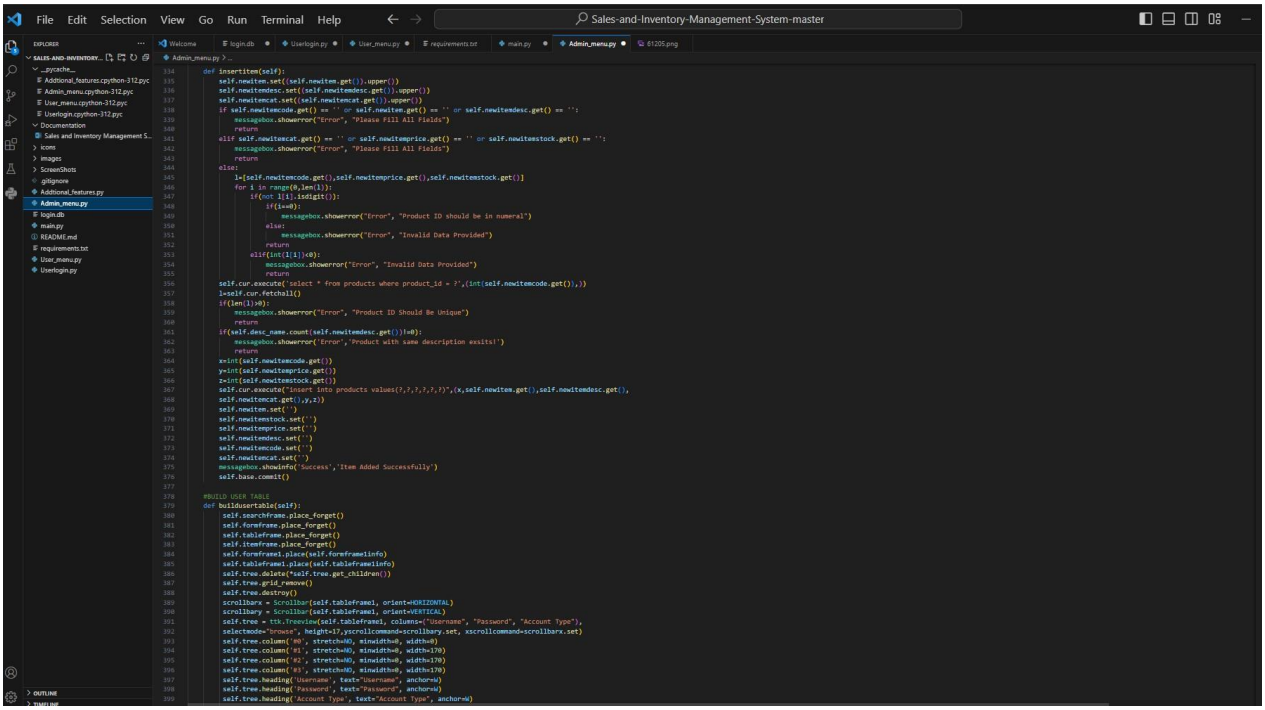
```
        myentry(self.foreframe, textvariable=self.itemeditcatv, font="roboto 14",bg="#FFFFFF", width=20)
        li = self.cur.fetchall()
        a = []
        self.desc_name = []
        for i in range(0, len(li)):
            if (a.count(li[i][0]) == 0):
                a.append(li[i][0])
        x.set_completion_list(a)
        Entry(self.foreframe, textvariable=self.itemeditpricev, font="roboto 14",bg="#FFFFFF", width=20).place(x=142, y=180, height=40)
        Entry(self.foreframe, textvariable=self.itemeditstockv, font="roboto 14",bg="#FFFFFF", width=20).place(x=142, y=240, height=40)
        Entry(self.foreframe, textvariable=self.addstock, font="roboto 14",bg="#FFFFFF", width=20).place(x=142, y=300, height=40)
        Button(self.foreframe, text="Update", font="robot 11 bold",bg="#FFFFFF", bd=5, width=10, height=2,
        command=self.changeprodtable).place(x=105, y=361)
        Button(self.foreframe, text="Remove", font="robot 11 bold",bg="#FFFFFF", bd=5, width=10, height=2,
        command=self.delproduct).place(x=305, y=361)
        self.cond=1
        self.mainsearch(1)

    #SEARCH FRAME FOR BOTH USER AND PRODUCT TABLE
    def mainsearch(self, f):
        self.searchvar.set('')
        if (f==1):
            self.searchframe.config(width=720)
            self.searchframe.place(x=575, y=245)
            self.searchbut.config(text="Search Description",command=self.searchprod)
            self.searchbut.place(x=0, y=23, height=37)
            self.searchentry.config(textvariable=self.searchvar,width=20)
            self.searchentry.place(x=210, y=25, height=35)
            self.cur.execute("select product_desc from products")
            li = self.cur.fetchall()
            a = []
            for i in range(0, len(li)):
                a.append(li[i][0])
            self.searchentry.set_completion_list(a)
            self.resetbut.config(command=self.resetprodtable1)
            self.resetbut.place(x=460, y=22, height=37)
        elif(f==0):
            self.searchframe.place(x=661, y=245)
            self.searchframe.config(width=520)
            self.searchbut.config(command=self.searchuser)
            self.searchbut.config(text="Search Username")
            self.searchbut.place(x=0,y=23)
            self.searchentry.config(width=18,textvariable=self.searchvar)
            self.searchentry.place(x=195, y=25, height=35)
            self.resetbut.config(command=self.resetusertable)
            self.resetbut.place(x=415,y=23)
            self.cur.execute("select username from users")
            li = self.cur.fetchall()
            a = []
            for i in range(0, len(li)):
                a.append(li[i][0])
            self.searchentry.set_completion_list(a)
        else:
            self.searchframe.place(x=138, y=245)
            self.searchframe.config(width=520)
            self.searchbut.config(command=self.searchinvoice)
            self.searchbut.config(text="Search Invoice No.")
            self.searchbut.place(x=0, y=23)
            self.searchentry.config(width=18, textvariable=self.searchvar)
            self.searchentry.place(x=195, y=25, height=35)
            self.resetbut.config(command=self.buildsalestable)
            self.resetbut.place(x=415, y=23)
            self.cur.execute("select invoice from sales")
            li = self.cur.fetchall()
            a = []
```



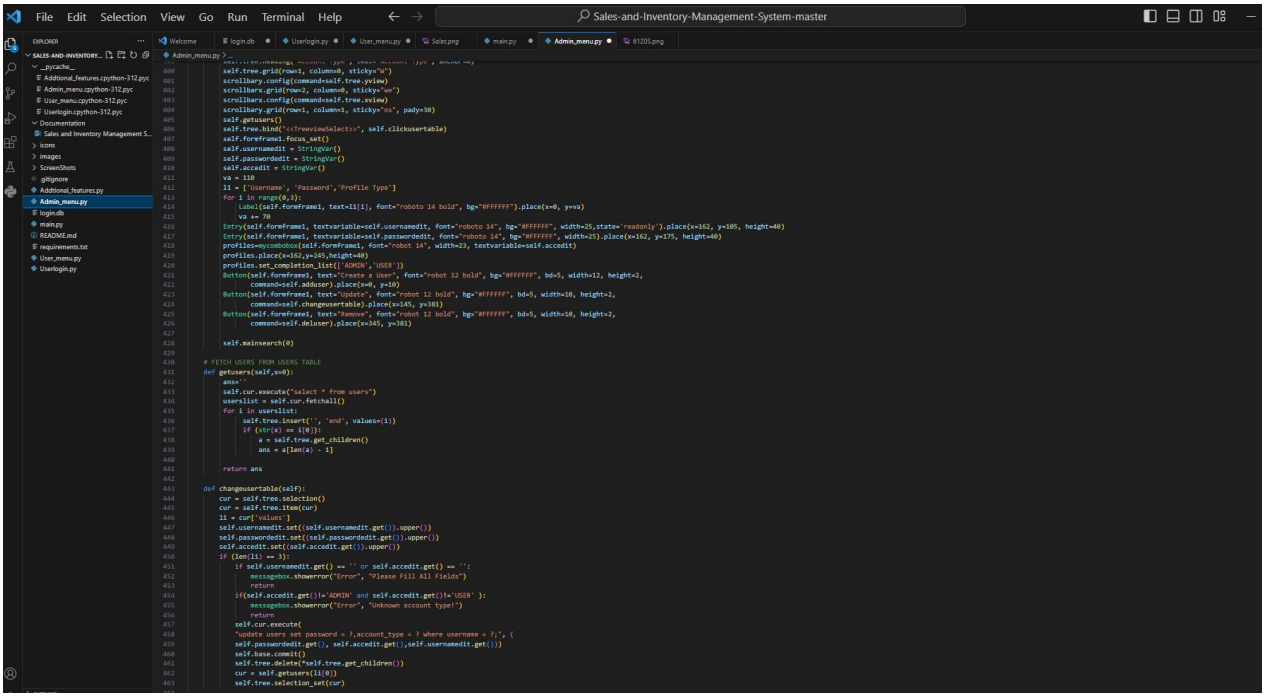
```
        for i in range(0, len(li)):
            if(a.count(str(li[i][0]))==0):
                a.append(str(li[i][0]))
        self.searchentry.set_completion_list(a)

# FETCH PRODUCTS FROM PRODUCTS TABLE
def getproducts(self,x=0):
    ans=''
    self.cur.execute("select * from products")
    productlist = self.cur.fetchall()
    for i in productlist:
        self.tree.insert('', 'end', values=(i))
        if (str(x) == i[0]):
            a=self.tree.get_children()
            ans=a[len(a)-1]

    return ans

# MODIFIES RECORD OF PRODUCT TABLE
def changeprodtable(self):
    cur = self.tree.selection()
    cur = self.tree.item(cur)
    li = cur['values']
    self.itemeditv.set((self.itemeditv.get()).upper())
    self.itemeditcatv.set((self.itemeditcatv.get()).upper())
    self.itemeditdescv.set((self.itemeditdescv.get()).upper())
    if(len(li) == 6):
        if self.itemeditv.get() == '' or self.itemeditdescv.get() == '':
            messagebox.showerror("Error", "Please Fill All Fields")
            return
        elif self.itemeditcatv.get() == '' or self.itemeditstockv.get() == '' or self.itemeditpricev.get() == '':
            messagebox.showerror("Error", "Please Fill All Fields")
            return
        else:
            l = [self.itemeditpricev.get(), self.itemeditstockv.get()]
            for i in range(0, len(l)):
                if (not l[i].isdigit()):
                    messagebox.showerror("Error", "Invalid Data Provided")
                    return
                elif (int(l[i]) < 0):
                    messagebox.showerror("Error", "Invalid Data Provided")
                    return
        if(self.addstock.get()== ''):
            self.addstock.set('0')

        self.cur.execute("update products set product_name=?,product_desc=?,product_cat=?,product_price = ?,stocks = ? where product_id = ?;",(self.itemeditv.get(),self.itemeditdescv.get(),self.itemeditcatv.get(),int(self.itemeditpricev.get()),(int(self.itemeditstockv.
        self.base.commit()
        self.addstock.set('')
        self.tree.delete(*self.tree.get_children())
        cur=self.getproducts(li[0])
        self.tree.selection_set(cur)

def delproduct(self):
    cur = self.tree.focus()
    cur = self.tree.item(cur)
    li = cur['values']
    if messagebox.askyesno('Alert!','Do you want to remove product from inventory?') == True and len(li) ==6:
        self.cur.execute("delete from products where product_id = ?;", (li[0],))
        self.base.commit()
        self.tree.delete(*self.tree.get_children())
        self.getproducts()
        self.itemeditv.set('')
        self.itemeditdescv.set('')
        self.itemeditcatv.set('')
        self.itemeditstockv.set('')
        self.itemeditpricev.set('')
```



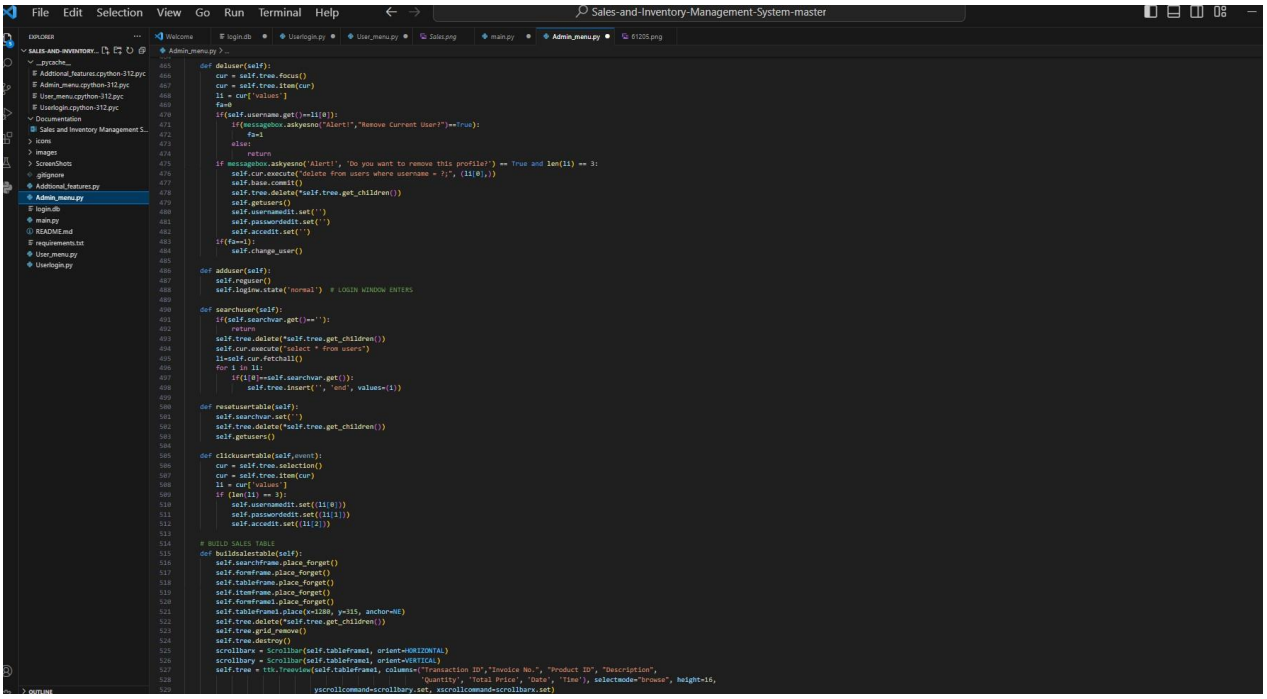
```
def searchprod(self):
    if (self.searchvar.get() == ''):
        return
    self.tree.delete(*self.tree.get_children())
    self.cur.execute("select * from products")
    li=self.cur.fetchall()
    for i in li:
        if(i[2]==self.searchvar.get()):
            self.tree.insert('', 'end', values=(i))

def resetprodtabel(self):
    self.searchvar.set('')
    self.tree.delete(*self.tree.get_children())
    self.getproducts()

# ONCLICK EVENT FOR PRODUCT TABLE
def clickprodtable(self, event):
    cur = self.tree.selection()
    cur = self.tree.item(cur)
    li = cur['values']
    if (len(li) == 6):
        self.itemeditv.set((li[1]))
        self.itemeditdescv.set((li[2]))
        self.itemeditcatv.set((li[3]))
        self.itemeditpricev.set(str(li[4]))
        self.itemeditstockv.set(str(li[5]))
        self.addstock.set('')

# FUNCTION FOR ITEM BUTTON
def additems(self):
    self.formframe1.place_forget()
    self.searchframe.place_forget()
    self.tableframe.place_forget()
    self.tableframe1.place_forget()
    self.formframe.place_forget()
    self.itemframe.place(self.itemframeinfo)
    self.newitem = StringVar()
    self.newitemdesc = StringVar()
    self.newitemcode = StringVar()
    self.newitemcat = StringVar()
    self.newitemprice = StringVar()
    self.newitemstock  = StringVar()
    la=['Product Id',"Product Name","Description","Category","Price","Stock"]
    for i in range(0,len(li)):
        Label(self.itemframe,text=la[i],font="Roboto 14 bold",bg="#ffffff").grid(row=i,column=0,padx=15,sticky="w")
    Entry(self.itemframe, width=60, textvariable=self.newitemcode,font="roboto 11",bg="#ffffff").grid(row=0, column=1,padx=15,padx=10,ipadx=3)
    Entry(self.itemframe,width=60,textvariable=self.newitem,font="roboto 11",bg="#ffffff").grid(row=1,column=1,padx=15,padx=10,ipadx=3)
    Entry(self.itemframe,width=60,textvariable=self.newitemdesc,font="roboto 11",bg="#ffffff").grid(row=2,column=1,padx=15,padx=8,ipadx=3)
    cat=myentry(self.itemframe,width=60,textvariable=self.newitemcat,font="roboto 11",bg="#ffffff")
    cat.grid(row=3,column=1,padx=15,padx=10,ipadx=3)
    Entry(self.itemframe, width=60, textvariable=self.newitemprice,font="roboto 11",bg="#ffffff").grid(row=4, column=1, padx=15, padx=10,ipadx=3)
    Entry(self.itemframe, width=60, textvariable=self.newitemstock,font="roboto 11",bg="#ffffff").grid(row=5, column=1, padx=15, padx=10,ipadx=3)
    self.cur.execute("select product_cat,product_name,product_desc from products")
    li=self.cur.fetchall()
    a=[]
    self.desc_name=[]
    for i in range(0,len(li)):
        if(a.count(li[i][0])==0):
            a.append(li[i][0])
            self.desc_name.append(li[i][2])
    cat.set_completion_list(a)
    Button(self.itemframe,text="Add item",height=3,bd=6,command=self.insertitem,bg="#ffffff").grid(row=6,column=1,padx=10,padx=12,sticky="w",ipadx=10)
    Button(self.itemframe, text="Back", height=3,width=8, bd=6,command=self.buildprodtable,bg="#ffffff").grid(row=6, column=1,padx=15, padx=10,sticky="e",ipadx=10)

# PERFORMS CHECK AND ADD'S ITEMS
def insertitem(self):
```



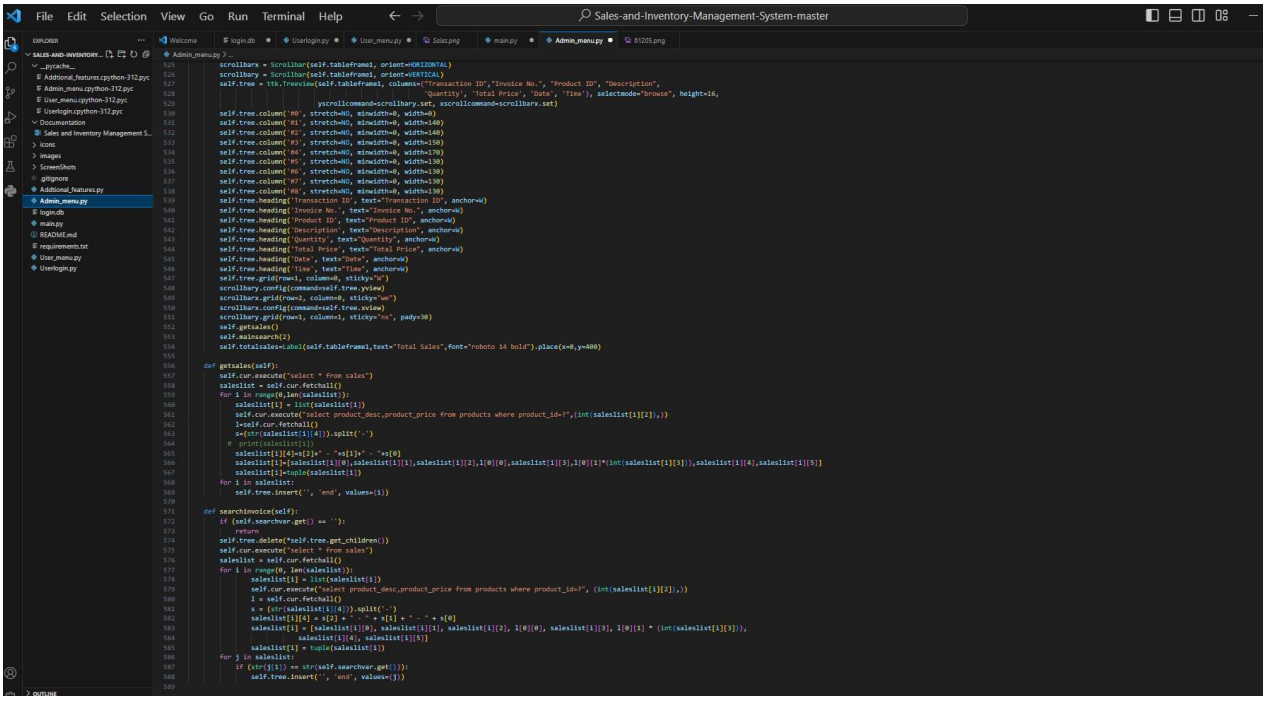
```
    def insertitem(self):
        self.newitem.set((self.newitem.get()).upper())
        self.newitemdesc.set((self.newitemdesc.get()).upper())
        self.newitemcat.set((self.newitemcat.get()).upper())
        if self.newitemcode.get() == '' or self.newitem.get() == '' or self.newitemdesc.get() == '':
            messagebox.showerror("Error", "Please Fill All Fields")
            return
        elif self.newitemcat.get() == '' or self.newitemprice.get() == '' or self.newitemstock.get() == '':
            messagebox.showerror("Error", "Please Fill All Fields")
            return
        else:
            l=[self.newitemcode.get(),self.newitemprice.get(),self.newitemstock.get()]
            for i in range(0,len(l)):
                if(not l[i].isdigit()):
                    if(i==0):
                        messagebox.showerror("Error", "Product ID should be in numeral")
                    else:
                        messagebox.showerror("Error", "Invalid Data Provided")
                    return
                elif(int(l[i])<0):
                    messagebox.showerror("Error", "Invalid Data Provided")
                    return
        self.cur.execute('select * from products where product_id = ?',(int(self.newitemcode.get()),))
        l=self.cur.fetchall()
        if(len(l)>0):
            messagebox.showerror("Error", "Product ID Should Be Unique")
            return
        if(self.desc_name.count(self.newitemdesc.get())!=0):
            messagebox.showerror('Error','Product with same description exsits!')
            return
        x=int(self.newitemcode.get())
        y=int(self.newitemprice.get())
        z=int(self.newitemstock.get())
        self.cur.execute("insert into products values(?,?,?,?,?,?)",(x,self.newitem.get(),self.newitemdesc.get(),
        self.newitemcat.get(),y,z))
        self.newitem.set('')
        self.newitemstock.set('')
        self.newitemprice.set('')
        self.newitemdesc.set('')
        self.newitemcode.set('')
        self.newitemcat.set('')
        messagebox.showinfo('Success','Item Added Successfully')
        self.base.commit()

    #BUILD USER TABLE
    def buildusertable(self):
        self.searchframe.place_forget()
        self.formframe.place_forget()
        self.tableframe.place_forget()
        self.itemframe.place_forget()
        self.formframe1.place(self.formframeinfo)
        self.tableframe1.place(self.tableframeinfo)
        self.tree.delete(*self.tree.get_children())
        self.tree.grid_remove()
        self.tree.destroy()
        scrollbarx = Scrollbar(self.tableframe1, orient=HORIZONTAL)
        scrollbary = Scrollbar(self.tableframe1, orient=VERTICAL)
        self.tree = ttk.Treeview(self.tableframe1, columns=("Username", "Password", "Account Type"),
        selectmode="browse", height=17,yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
        self.tree.column('#0', stretch=NO, minwidth=0, width=0)
        self.tree.column('#1', stretch=NO, minwidth=0, width=170)
        self.tree.column('#2', stretch=NO, minwidth=0, width=170)
        self.tree.column('#3', stretch=NO, minwidth=0, width=170)
        self.tree.heading('Username', text="Username", anchor=W)
        self.tree.heading('Password', text="Password", anchor=W)
        self.tree.heading('Account Type', text="Account Type", anchor=W)
```



```
        self.tree.grid(row=1, column=0, sticky='W')
        scrollbary.config(command=self.tree.yview)
        scrollbary.grid(row=2, column=0, sticky="we")
        scrollbarx.config(command=self.tree.xview)
        scrollbary.grid(row=1, column=1, sticky="ns", pady=30)
        self.getusers()
        self.tree.bind("<ttreeviewSelect>>", self.clickusertable)
        self.foreframe1.focus_set()
        self.usernameedit = StringVar()
        self.passwordedit = StringVar()
        self.accedit = StringVar()
        va = 110
        l1 = ['Username', 'Password','Profile Type']
        for i in range(0,3):
            Label(self.formframe1, text=l1[i], font="roboto 14 bold", bg="#FFFFFF").place(x=0, y=va)
            va += 70
        Entry(self.formframe1, textvariable=self.usernameedit, font="roboto 14", bg="#FFFFFF", width=25,state='readonly').place(x=162, y=105, height=40)
        Entry(self.formframe1, textvariable=self.passwordedit, font="roboto 14", bg="#FFFFFF", width=25).place(x=162, y=175, height=40)
        profiles=mycombobox(self.formframe1, font="robot 14", width=23, textvariable=self.accedit)
        profiles.place(x=162,y=245,height=40)
        profiles.set_completion_list(['ADMIN','USER'])
        Button(self.formframe1, text="Create a User", font="robot 12 bold", bg="#FFFFFF", bd=5, width=12, height=2,
               command=self.adduser).place(x=0, y=10)
        Button(self.formframe1, text="Update", font="robot 12 bold", bg="#FFFFFF", bd=5, width=10, height=2,
               command=self.changeusertable).place(x=145, y=381)
        Button(self.formframe1, text="Remove", font="robot 12 bold", bg="#FFFFFF", bd=5, width=10, height=2,
               command=self.deluser).place(x=345, y=381)

        self.mainsearch(0)

    # FETCH USERS FROM USERS TABLE
    def getusers(self,x=0):
        ans=''
        self.cur.execute("select * from users")
        userslist = self.cur.fetchall()
        for i in userslist:
            self.tree.insert('', 'end', values=(i))
            if (str(x) == i[0]):
                a = self.tree.get_children()
                ans = a[len(a) - 1]

        return ans

    def changeusertable(self):
        cur = self.tree.selection()
        cur = self.tree.item(cur)
        l1 = cur['values']
        self.usernameedit.set((self.usernameedit.get()).upper())
        self.passwordedit.set((self.passwordedit.get()).upper())
        self.accedit.set((self.accedit.get()).upper())
        if (len(l1) == 3):
            if self.usernameedit.get() == '' or self.accedit.get() == '':
                messagebox.showerror("Error", "Please Fill All Fields")
                return
            if(self.accedit.get()!='ADMIN' and self.accedit.get()!='USER' ):
                messagebox.showerror("Error", "Unknown account type!")
                return
            self.cur.execute(
            "update users set password = ?,account_type = ? where username = ?;", (
            self.passwordedit.get(), self.accedit.get(),self.usernameedit.get()))
            self.base.commit()
            self.tree.delete(*self.tree.get_children())
            cur = self.getusers(l1[0])
            self.tree.selection_set(cur)
```



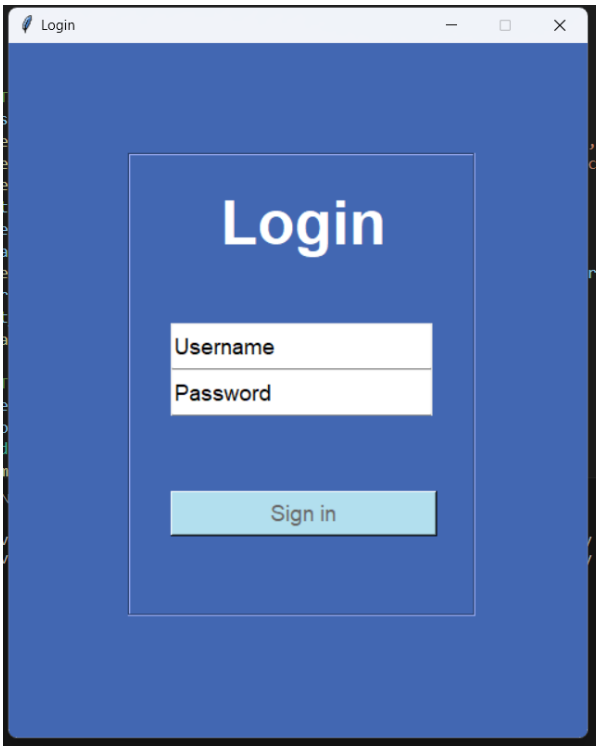
```
    def deluser(self):
        cur = self.tree.focus()
        cur = self.tree.item(cur)
        li = cur['values']
        fa=0
        if(self.username.get()==li[0]):
            if(messagebox.askyesno("Alert!","Remove Current User?")==True):
                fa=1
            else:
                return
        if messagebox.askyesno('Alert!', 'Do you want to remove this profile?') == True and len(li) == 3:
            self.cur.execute("delete from users where username = ?;", (li[0],))
            self.base.commit()
            self.tree.delete(*self.tree.get_children())
            self.getusers()
            self.usernamedit.set('')
            self.passwordedit.set('')
            self.accedit.set('')
        if(fa==1):
            self.change_user()

    def adduser(self):
        self.reguser()
        self.loginw.state('normal')  # LOGIN WINDOW ENTERS

    def searchuser(self):
        if(self.searchvar.get()==''):
            return
        self.tree.delete(*self.tree.get_children())
        self.cur.execute("select * from users")
        li=self.cur.fetchall()
        for i in li:
            if(i[0]==self.searchvar.get()):
                self.tree.insert('', 'end', values=(i))

    def resetusertable(self):
        self.searchvar.set('')
        self.tree.delete(*self.tree.get_children())
        self.getusers()

    def clickusertable(self,event):
        cur = self.tree.selection()
        cur = self.tree.item(cur)
        li = cur['values']
        if (len(li) == 3):
            self.usernamedit.set((li[0]))
            self.passwordedit.set((li[1]))
            self.accedit.set((li[2]))

    # BUILD SALES TABLE
    def buildsalestable(self):
        self.searchframe.place_forget()
        self.formframe.place_forget()
        self.tableframe.place_forget()
        self.itemframe.place_forget()
        self.formframe1.place_forget()
        self.tableframe1.place(x=1280, y=115, anchor=NE)
        self.tree.delete(*self.tree.get_children())
        self.tree.grid_remove()
        self.tree.destroy()
        scrollbarx = Scrollbar(self.tableframe1, orient=HORIZONTAL)
        scrollbary = Scrollbar(self.tableframe1, orient=VERTICAL)
        self.tree = ttk.Treeview(self.tableframe1, columns=("Transaction ID","Invoice No.", "Product ID", "Description",
                                                            'Quantity', 'Total Price', 'Date', 'Time'), selectmode="browse", height=16,
                                 yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
```



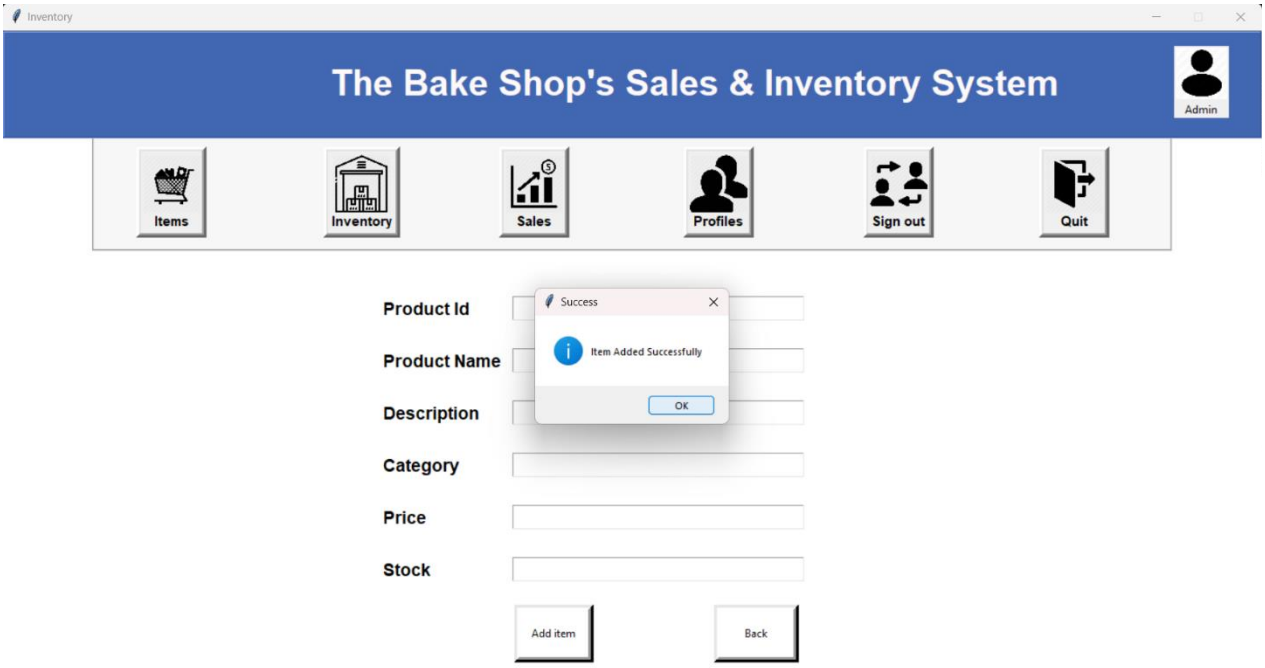
```
        scrollbarx = Scrollbar(self.tableframe1, orient=HORIZONTAL)
        scrollbary = Scrollbar(self.tableframe1, orient=VERTICAL)
        self.tree = ttk.Treeview(self.tableframe1, columns=("Transaction ID","Invoice No.", "Product ID", "Description",
                                                            'Quantity', 'Total Price', 'Date', 'Time'), selectmode="browse", height=16,
                                 yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
        self.tree.column('#0', stretch=NO, minwidth=0, width=0)
        self.tree.column('#1', stretch=NO, minwidth=0, width=140)
        self.tree.column('#2', stretch=NO, minwidth=0, width=140)
        self.tree.column('#3', stretch=NO, minwidth=0, width=150)
        self.tree.column('#4', stretch=NO, minwidth=0, width=170)
        self.tree.column('#5', stretch=NO, minwidth=0, width=130)
        self.tree.column('#6', stretch=NO, minwidth=0, width=130)
        self.tree.column('#7', stretch=NO, minwidth=0, width=130)
        self.tree.column('#8', stretch=NO, minwidth=0, width=130)
        self.tree.heading('Transaction ID', text="Transaction ID", anchor=W)
        self.tree.heading('Invoice No.', text="Invoice No.", anchor=W)
        self.tree.heading('Product ID', text="Product ID", anchor=W)
        self.tree.heading('Description', text="Description", anchor=W)
        self.tree.heading('Quantity', text="Quantity", anchor=W)
        self.tree.heading('Total Price', text="Total Price", anchor=W)
        self.tree.heading('Date', text="Date", anchor=W)
        self.tree.heading('Time', text="Time", anchor=W)
        self.tree.grid(row=1, column=0, sticky="W")
        scrollbary.config(command=self.tree.yview)
        scrollbarx.grid(row=2, column=0, sticky="we")
        scrollbarx.config(command=self.tree.xview)
        scrollbary.grid(row=1, column=1, sticky="ns", pady=30)
        self.getsales()
        self.mainsearch(2)
        self.totalsales=Label(self.tableframe1,text="Total Sales",font="roboto 14 bold").place(x=0,y=480)

    def getsales(self):
        self.cur.execute("select * from sales")
        saleslist = self.cur.fetchall()
        for i in range(0,len(saleslist)):
            saleslist[i] = list(saleslist[i])
            self.cur.execute("select product_desc,product_price from products where product_id=?",(int(saleslist[i][2]),))
            l=self.cur.fetchall()
            s=(str(saleslist[i][4])).split('-')
          #  print(saleslist[i])
            saleslist[i][4]=s[2]+ "-" +s[1]+ "-" +s[0]
            saleslist[i]=[saleslist[i][0],saleslist[i][1],saleslist[i][2],l[0][0],saleslist[i][3],l[0][1]*(int(saleslist[i][3])),saleslist[i][4],saleslist[i][5]]
            saleslist[i]=tuple(saleslist[i])
        for i in saleslist:
            self.tree.insert('', 'end', values=(i))

    def searchinvoice(self):
        if (self.searchvar.get() == ''):
            return
        self.tree.delete(*self.tree.get_children())
        self.cur.execute("select * from sales")
        saleslist = self.cur.fetchall()
        for i in range(0, len(saleslist)):
                saleslist[i] = list(saleslist[i])
                self.cur.execute("select product_desc,product_price from products where product_id=?", (int(saleslist[i][2]),))
                l = self.cur.fetchall()
                s = (str(saleslist[i][4])).split('-')
                saleslist[i][4] = s[2] + "-" + s[1] + "-" + s[0]
                saleslist[i] = [saleslist[i][0], saleslist[i][1], saleslist[i][2], l[0][0], saleslist[i][3], l[0][1] * (int(saleslist[i][3])),
                                saleslist[i][4], saleslist[i][5]]
                saleslist[i] = tuple(saleslist[i])
        for j in saleslist:
            if (str(j[1]) == str(self.searchvar.get())):
                self.tree.insert('', 'end', values=(j))
```

5.2 RESULTS

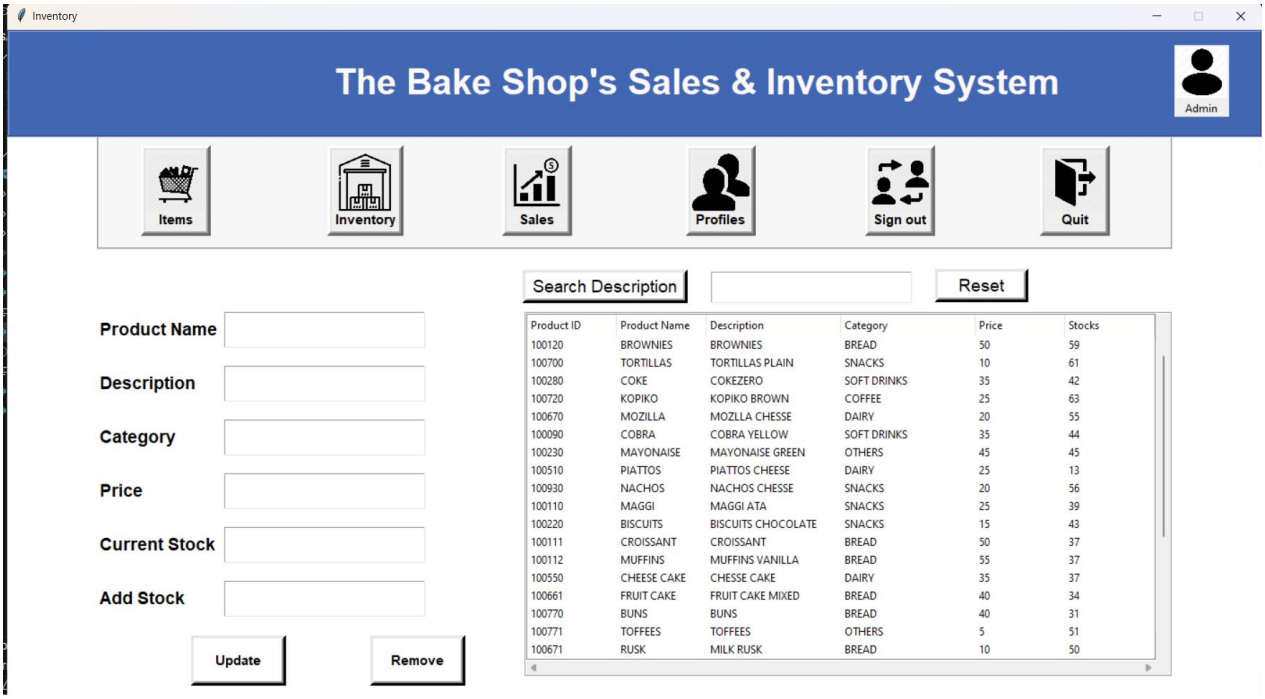
Login for admin



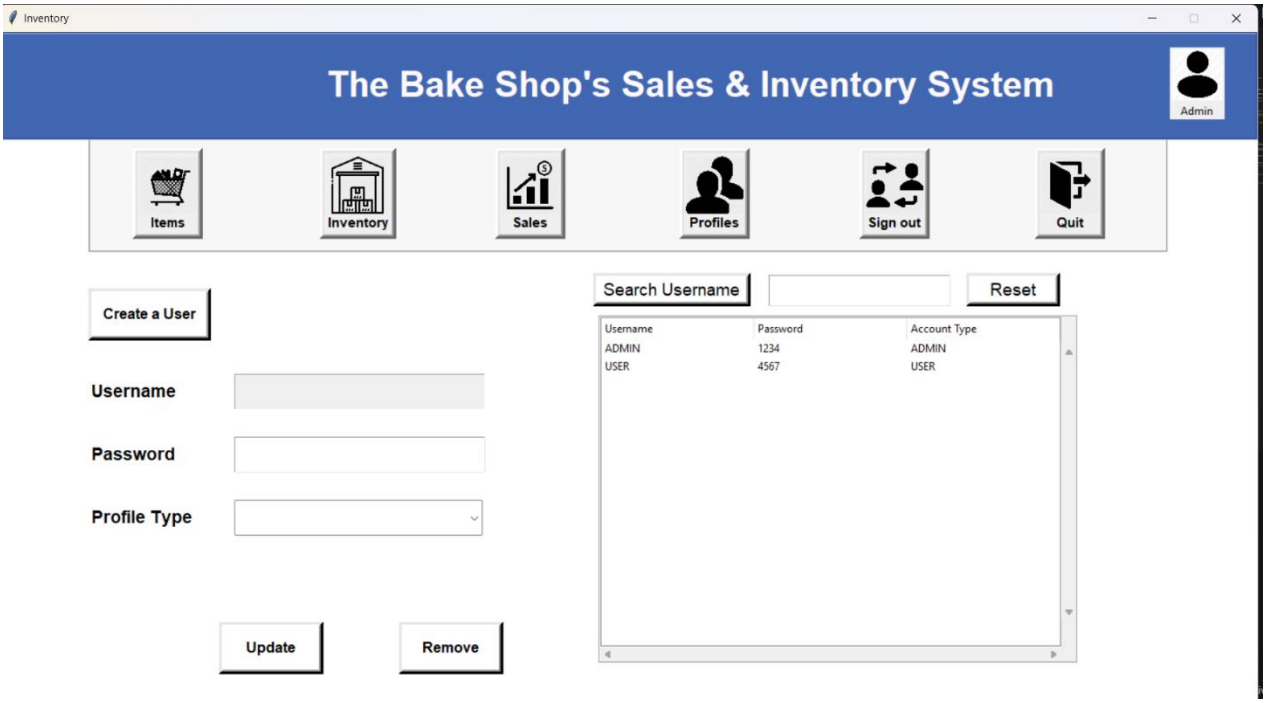
Products can be added to the inventory



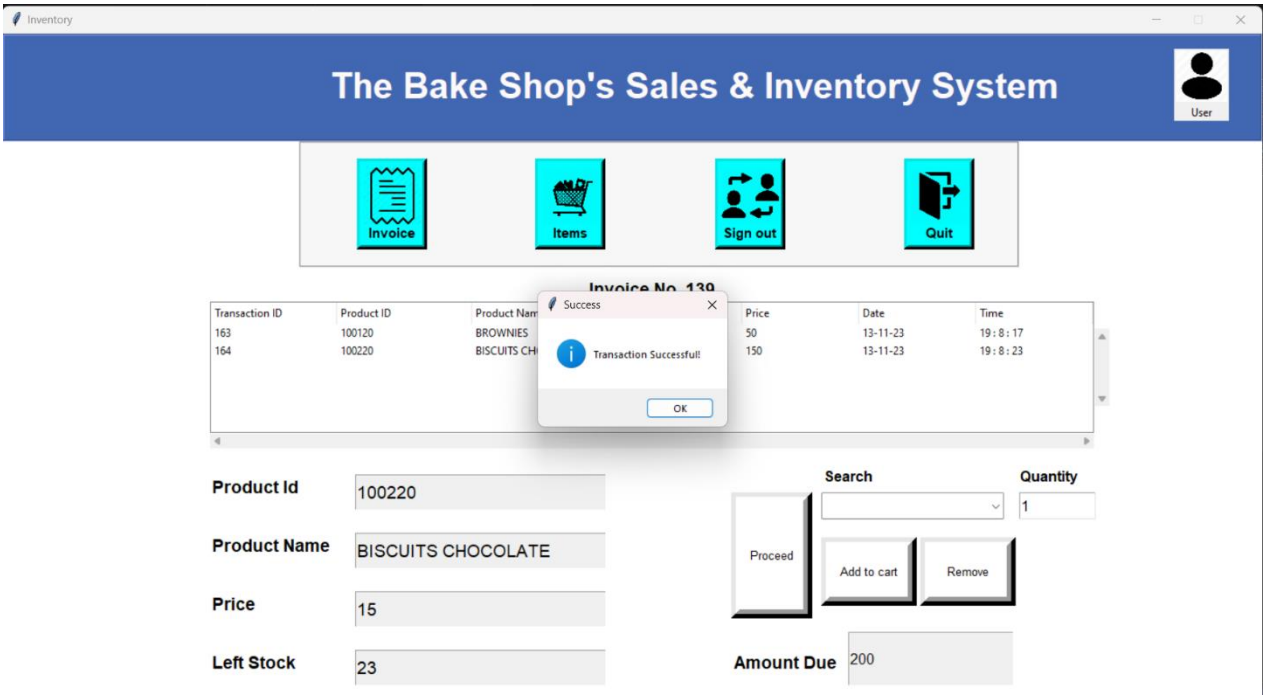
It will be displaying the products saved in the inventory



Creating and displaying the username



Product has been successful purchased and the invoice has been printed



5. CONCLUSION

In this project we have developed a system which helps the retailers to sell and manage their products easily. It covers the functional areas of ERP such as Marketing and sales, Supply chain management, Accounting and Finance and Human Resources. So, this can help in increasing the sales of the retailer through the help of the stock management. So, the required products can be bought based on the demand. In future the products can be scanned with the help of barcode scanner. A system can be developed to take order from the customers online and deliver them. The customer relationship can be built with the help of feedback.

Stock Management System is a simple desktop-based application basically suitable for small organization. It has every basic item which are used for the small organization. Our team is successful in making the application where we can update, insert and delete the item as per the requirement. This application also provides a simple report on daily basis to know the daily sales and purchase details. This application matches for small organization where there small limited if godwoms. Through it has some limitations, our team strongly believes that the implementation of this system will surely benefit the organization.

In conclusion, a well-implemented stock management system is essential for businesses to optimize their inventory, enhance operational efficiency, and deliver a superior customer experience. Such systems play a pivotal role in addressing the challenges of inventory accuracy, demand forecasting, supply chain disruptions, multichannel sales, vendor management, and data security. By leveraging advanced technology, data analytics, and proactive strategies, organizations can navigate these challenges effectively, ultimately leading to reduced costs, improved customer satisfaction, and increased competitiveness. A robust stock management system not only streamlines operations but also empowers businesses to adapt to dynamic market conditions and make informed decisions, ensuring long-term success in today's highly competitive business landscape.

6. REFERENCES

Books:

- "Inventory Management: A Practical Approach" by David W. Fogarty
- "Production and Inventory Management" by Donald W. Fogarty and Steven T. Rosenfield

Websites:

- Inventory Control and Management by Supply Chain Management Review
- Inventory Management by Oracle
- Inventory Management by SAP
- Inventory Management by JDA Software
- Inventory Management by Infor