PROJECT REPORT

Date	**14.11.2022**
Team ID	**PNT2022TMID28752**
Project Name	**Inventory Management System For Retailers**
Team Members	**MOHIT L -411719106032** **BALAKRISHNAN R -411719106006** **MOHAMMED THANVEER S -411719106031** **ARUN M -411719106005**

1. INTRODUCTION

1.1 Project Overview

The objective of this system is to manage the items in an inventory such as tracking orders, placing orders to other suppliers and checking the items in the inventory. The system allows the admin to maintain the items in the inventory.

Whenever the item levels go low, the system places an order to the supplier. The supplier gets the notification of these orders as soon as they are placed and can send the items to the inventory. There are two login pages each for the admin and supplier.

The software has been developed using the most powerful and secured backend Python and IBM Cloud for the databases and most widely accepted frontend JavaScript with HTML and CSS coding

1.2 Purpose

The primary purpose of inventory management is to ensure there is enough goods or materials to meet demand without creating overstock, or excess inventory

Retail management refers to the process of helping customers find products in your store. It includes everything from increasing your customer pool to how products are presented, and how you fulfill a customer's needs. A good store manager helps customers leave the store with a smile.

2. LITERATURE SURVEY

2.1 Existing problem

- The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.
- Good planning and sales forecast before setting optimal inventory levels, appropriate inventory management requires close coordination between the areas of sales, purchasing and finance.

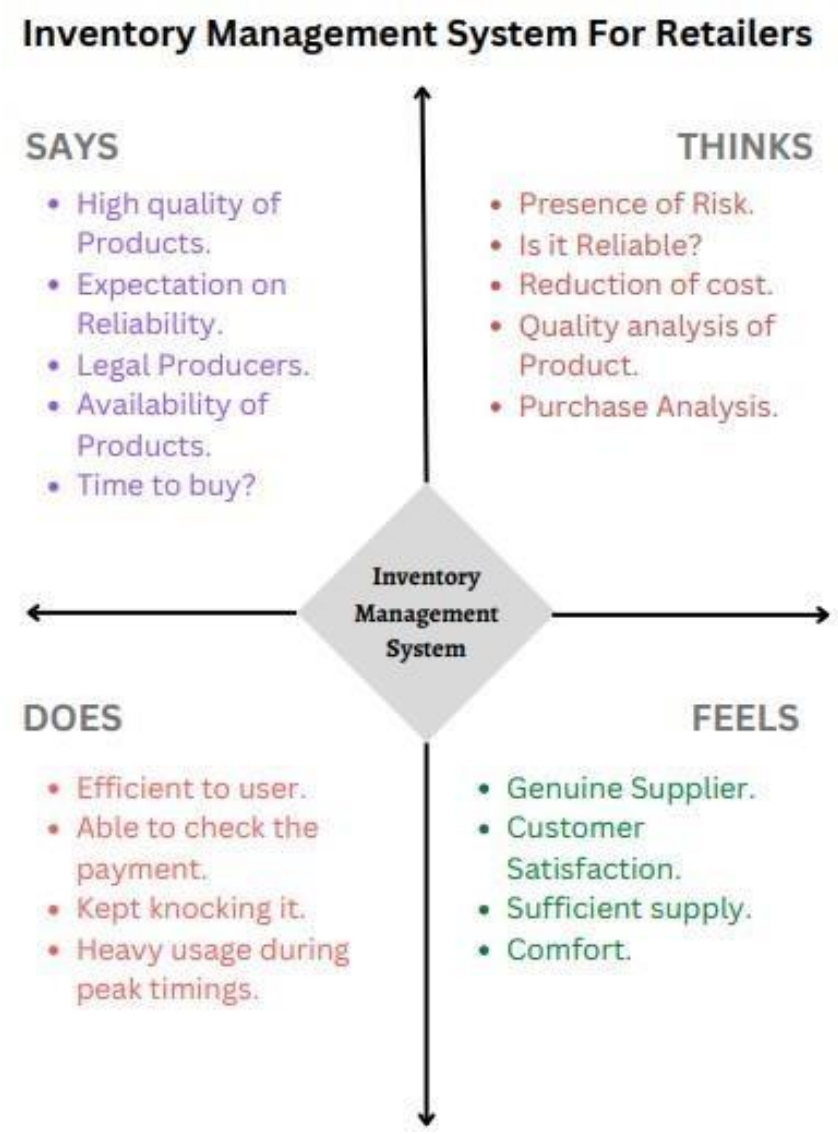
2.2 Problem Statement Definition

Retail inventory management works by creating systems to log products, receive them into inventory, track changes when sales occur, manage the flow of goods from purchasing to final sale and check stock counts.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



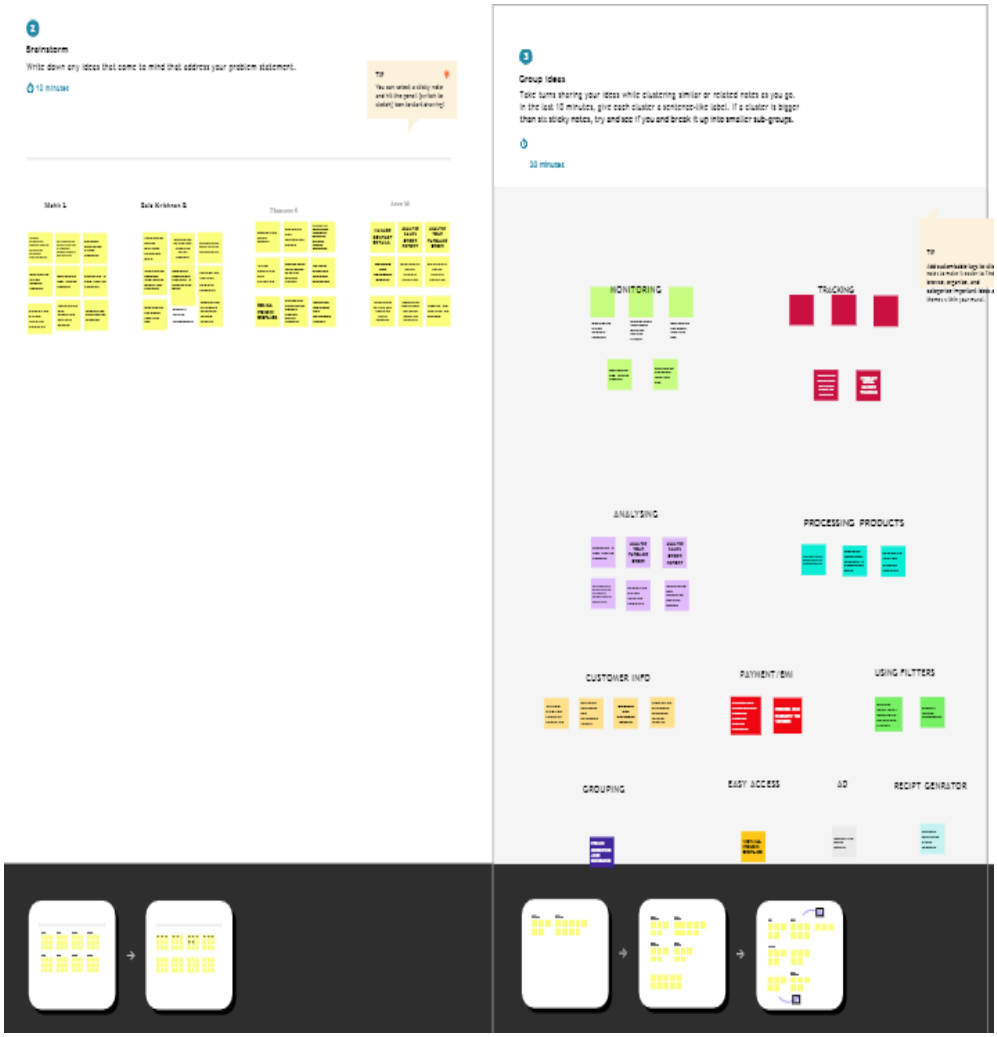
3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

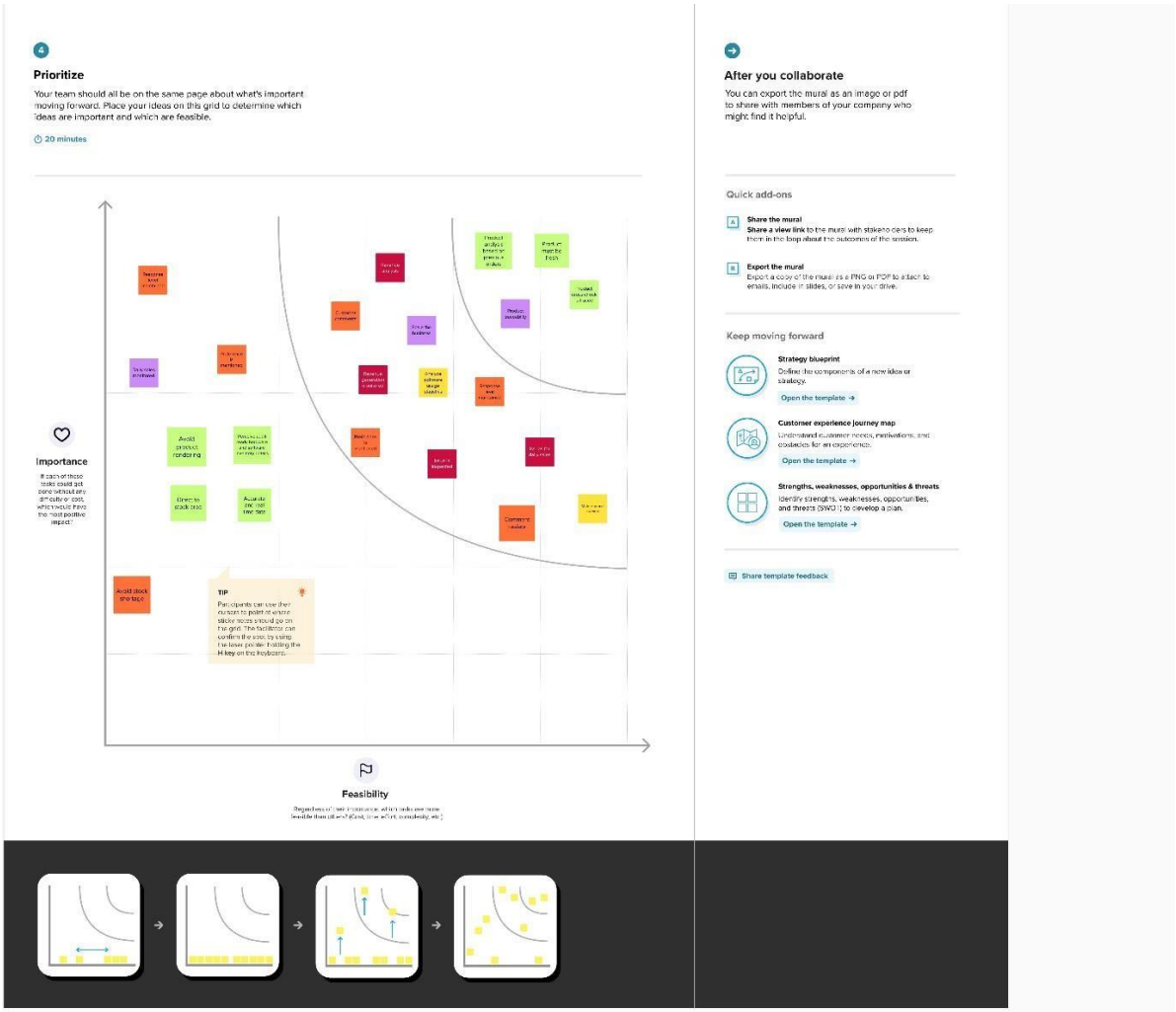
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



3.3 Proposed Solution

The system customizes and only shows recommended jobs based on the user's skill set and preferences (Using graphql api)
Similarly, the same recommendation system helps provide job applicant recommendations to the job recruiters to find the most eligible candidates for their firm.
All important data - job seeker's and hoster's personal information needs to be also stored safely and securely. Using a sql database is the most easiest, safest and convienent way possible.
Data needs to also be private in some cases like when information is shared with the host while applying for a job.

3.4 Problem Solution fit

Project Title: Inventory Management System for Retailers **Project Design Phase-I – Problem Solution Fit** **Team ID:** PNT2022TMID28952

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

- General consumers who are in need of a product.
- They can be of all ages.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- Cost of product.
- Quality product.
- Lack of network connection.
- Delivery cost.
- Product delivery delay.
- Device to order.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

- They can compare the cost of product and purchase their desired choice.
- They can return if the quality does not satisfy their expectation.
- They can see when the delivery date is and they can decide to purchase the product or not.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Maintaining up-to-date products.
- Ordering the right amount and not in excess.
- Purchasing the products in lower price than selling price.
- Having competitive stock pricings.
- Product demand forecasting.
- Not having enough bandwidth to support 'n' number of consumers in the site at a time.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do
this job?
i.e. customers have to do it because of the change in regulations.

- Can't predict customers needs in short period of time. Need data to have an accurate stock prediction.
- Contacting suppliers and getting good deals from them.
- Having low bandwidth to hold sufficient consumers in the site.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- Estimation of sales prediction to stock up by having customer feedback.
- Finding good supplier with low cost of product.
- Customer feedback for improvement of application.
- Having sufficient bandwidth to support on demand consumers.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- Customer unable to reach the application due to high demand.
- Having the stock price high.
- Lack of application service.

4. EMOTIONS: BEFORE / AFTER EM

BEFORE – Untrusted, worried, lack of knowledge of stocks.

AFTER - Trusted, happy, referring to others, having sound knowledge of stocks, etc.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behavior.

- Deploying the application in a cloud server that tracks the real-time inventory and manages them.
- Such as purchase details, sales, sales prediction, etc.
- It sends an email to the retailers when the stocks are low and needs to be restocked.
- Having a chatbot to guide and help the consumers who are having

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

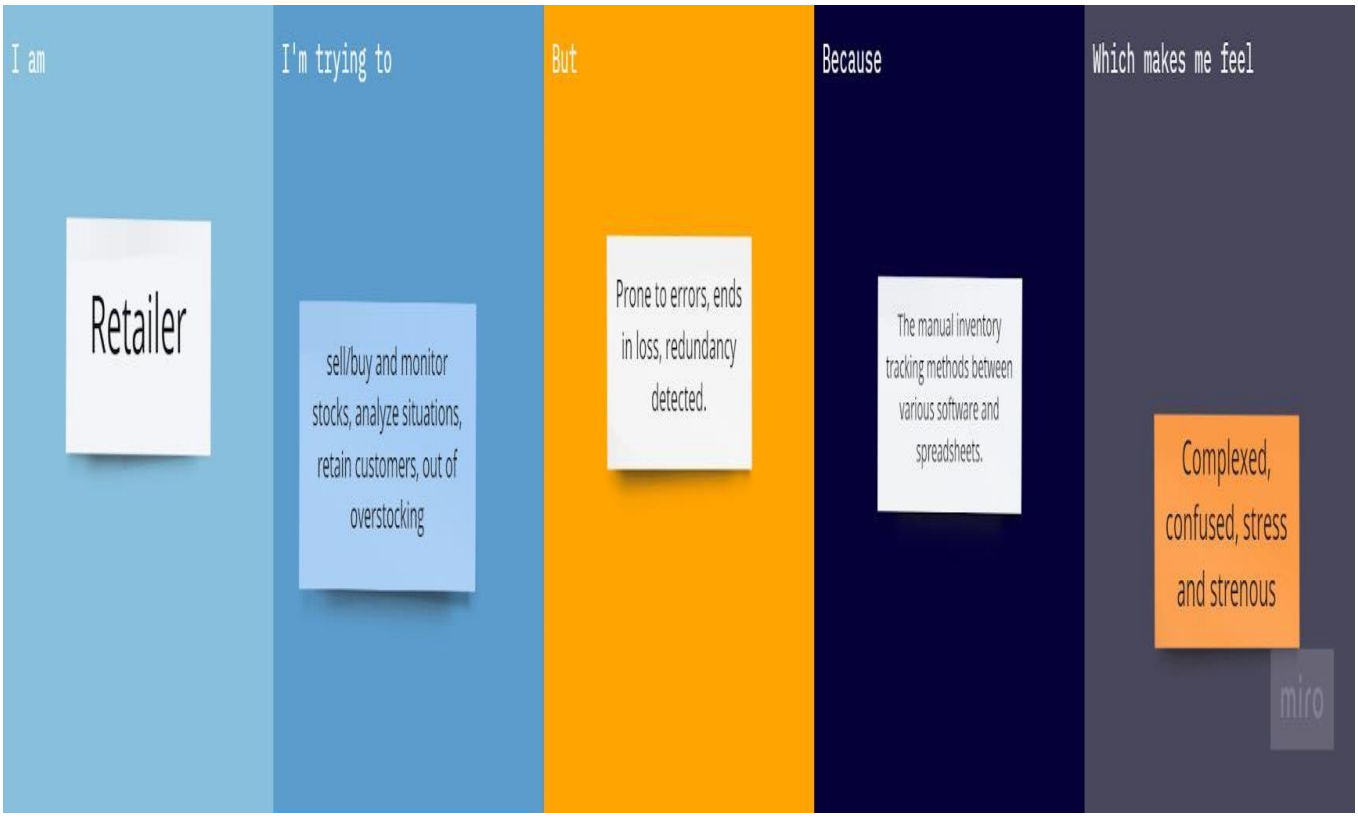
8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

ONLINE – Can access all the services and details.

OFFLINE - SMS notification for detailed list of enquiries.

Identify strong TR & EM

3.5 Customer Problem Statement



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

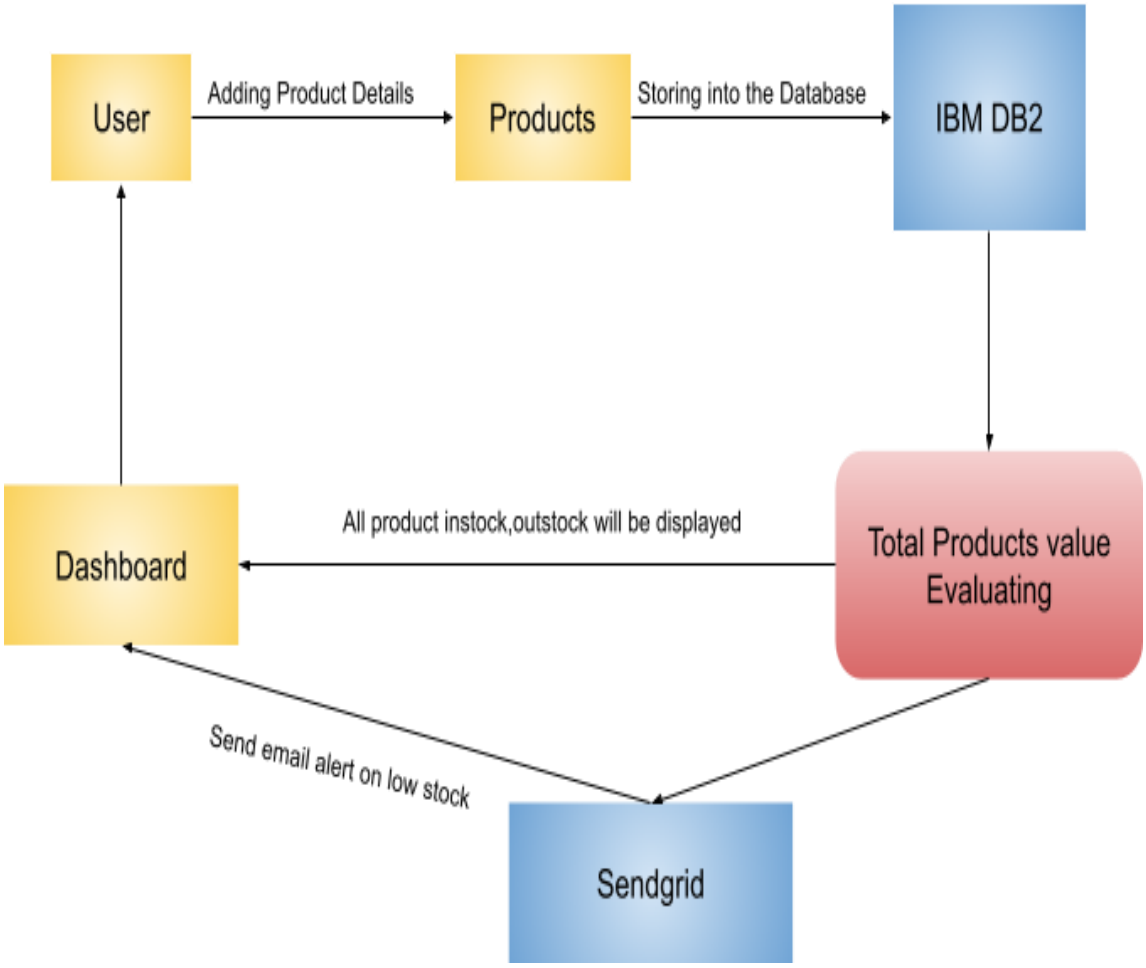
- The System aims at providing an efficient interface to the user for managing of inventory, it shall also provide the user varied options for managing the inventory through various functions at hand. The ingredient levels are continuously monitored based on their usage and are checked for the threshold levels in the inventory and accordingly the user is alerted about low levels of certain ingredients. The design is such that the user does not have to manually update the inventory every time, the System does if for the user.
- The System calculates and predicts the amount of usage for specific set days that are pre-set by the user(admin) , it also alerts the user of an impending action to order ingredients before the specific day set by the user. Therefore the user never has to worry about manually calculating the estimated usage of the ingredients as the System does it for the user.
- The simple interface of the System has functions like adding a recipe, removing or updating the recipe. It also extends to functions such as adding a vendor for an ingredient,, removing the vendor, checking threshold levels, processing orders, altering processed orders etc.

4.2 Non-Functional requirements

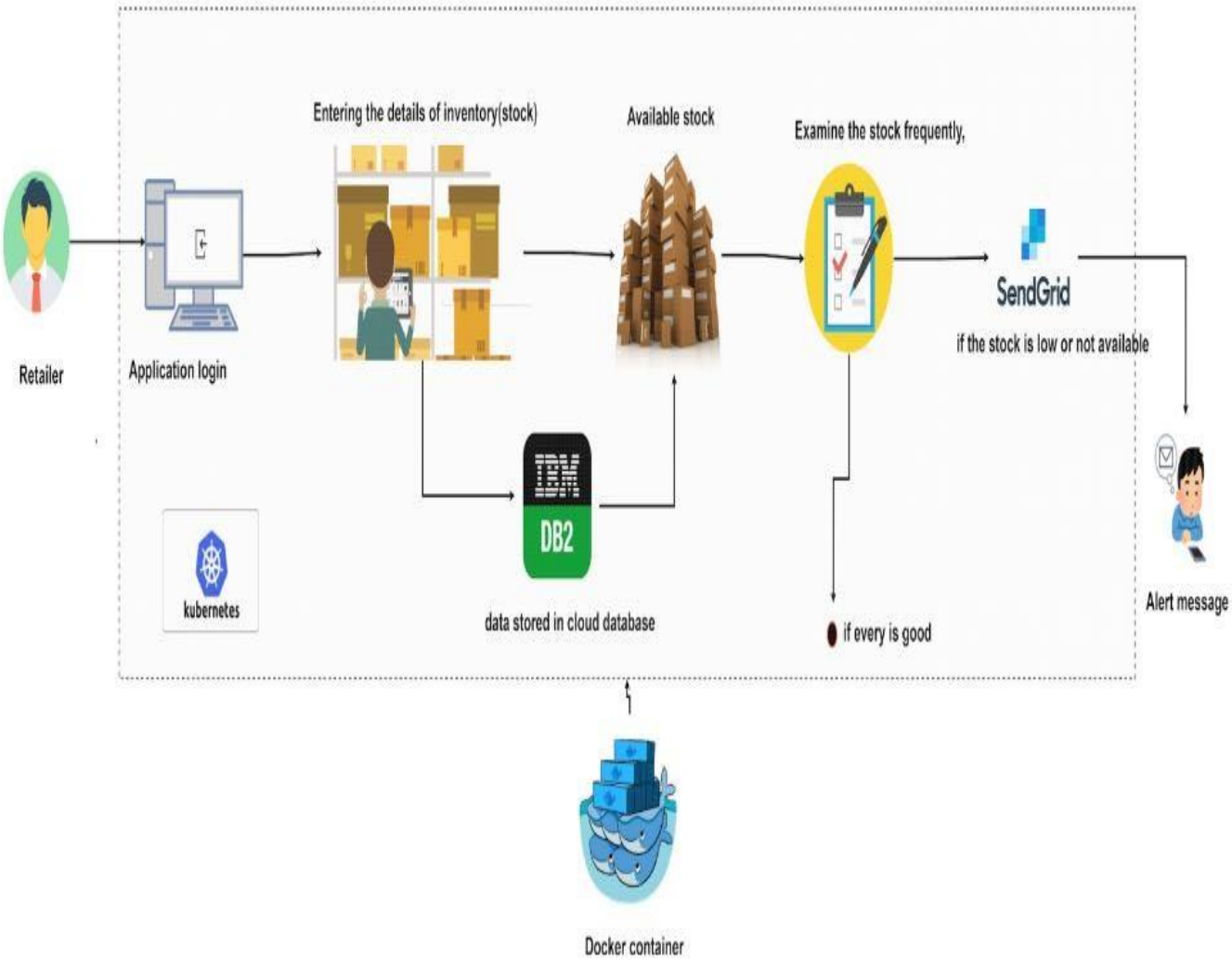
- The system must not lag, because the workers using it don't have down-time to wait for it to complete an action.
- The system must complete updating the databases, adding of recipe, ingredient, vendor and occasions successfully every time the user requests such a process.
- All the functions of the system must be available to the user every time the system is turned on.
- The calculations performed by the system must comply according to the norms set by the user and should not vary unless explicitly changed by the user
- The System must give accurate inventory status to the user continuously. Any inaccuracies are taken care by the regular confirming of the actual levels with the levels displayed in the system.
- The System must successfully add any recipe, ingredients, vendors or special occasions given by the user and provide estimations and inventory status in relevance with the newly updated entities.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Retailer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account /dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password	I can access my account /dashboard	High	Sprint-1
	Dashboard	USN-4	As a user, I can view the stock list and suppliers list	Once I log in to the system, I can able to view the stocks	Medium	Sprint-1
	Items	USN-5	As a user, I can add the items.	I can create a new type of item	High	Sprint-2
		USN-6	As a user, I can see the items	I can be able to see the items that can be added to the inventory	Low	Sprint-2
	Inventory	USN-7	As a user, I can add the items to inventory.	I can add items to the inventory with quantity	High	Sprint-2
		USN-8	As a user, I can see the items in the inventory.	I can see the inventory items with quantity	Low	Sprint-2
	Indication	USN-9	As a user, I can be able to receive indication	I receive a notification when the stock running low	High	Sprint-3
	Location	USN-10	As a user, I can be able to see items from a particular store location	I can be able to make purchase from a particular location	Medium	Sprint-3
		USN-11	As a user, I can add a new location of my store	I can be able to add new store locations	Medium	Sprint - 3
Customer	Purchase	USN -12	As a customer, I can be able to purchase good from the particular location of the store	I can able to purchase from the store	High	Sprint - 4
Retailer & Customer	Deployment	USN-13	As a user, I can access the software in the web	I can access the software in web	High	Sprint -4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	High	Split between all 4
Sprint 1		USN-2	As a user, I will receive confirmation email once I have registered for the application	Medium	Split between all 4
Sprint 1	Login	USN-3	As a user, I can log into the application by enteringemail & password	High	Nithyananthan & Prithiv
Sprint 2	Items	USN-5	As a user, I can add the items.	High	Mohamed Shiham
Sprint 2		USN-6	As a user, I can see the items	Low	Nithyananthan
Sprint 2	Inventory	USN-7	As a user, I can add the items to inventory.	High	Pradeep Sriram
Sprint 2		USN-8	As a user, I can see the items in the inventory.	Low	Prithiv
Sprint 3	Indication	USN-9	As a user, I can be able to receive indication	High	Pradeep Sriram
Sprint 3	Location	USN-10	As a user, I can be able to see items from a particularstore location	Medium	Mohamed Shiham
Sprint 3		USN-11	As a user, I can add a new location of my store	Medium	Nithyananthan
Sprint 4	Purchase	USN -12	As a customer, I can be able to purchase good fromthe particular location of the store	High	Prithiv

Sprint	Functional Requirement (Epic)	User Stor y Number	User Story / Task	Priority	Team Members
Sprint 4	Deployment	USN-13	As a user, I can access the software in the web	High	Pradeep Sriram

6.2 Milestone And Activities

Milestones and Activities:

Milestones	Activities
Registration	• Retailer Registration
Login	• Retailer Login
Item Actions	• Add item type • Display item types
Inventory Actions	• Add stock to inventory • Display stock from inventory
Notification	• Sent notification when threshold of stock reached
Location	• Add store location • Update operations to handle location
Customer	• Able to see stocks • Able to make purchase
Deployment	• Project deployment in IBM Cloud

7. CODING&SOLUTIONING

(Explain the features added in the project along with code)

7.1 Feature 1

Complete insights into key products and service drivers. With the help of tables and symbols, marketers can effectively track and analyse factors that have an effect on important bottom lines like profitability. Store Managers can also effectively optimise product mix across channels, lines and brands with the product scorecards available. Some of the different KPIs that managers can avail of from product performance metrics are product sales by region, change in sales and margin per product, ROI per product, top competitor by product category and much more..

7.2 Feature 2

The entire organisation can access the same store data simultaneously and thus everyone has an understanding of what the customer wants. Managers can better monitor progress, respond immediately to customer needs, adjust parameters for continuous improvement, and exercise greater control over the organisation.

One can record and analyze inventory results and merchandise processes daily to know whether business decisions are based on timely, accurate information.

7.3 Code

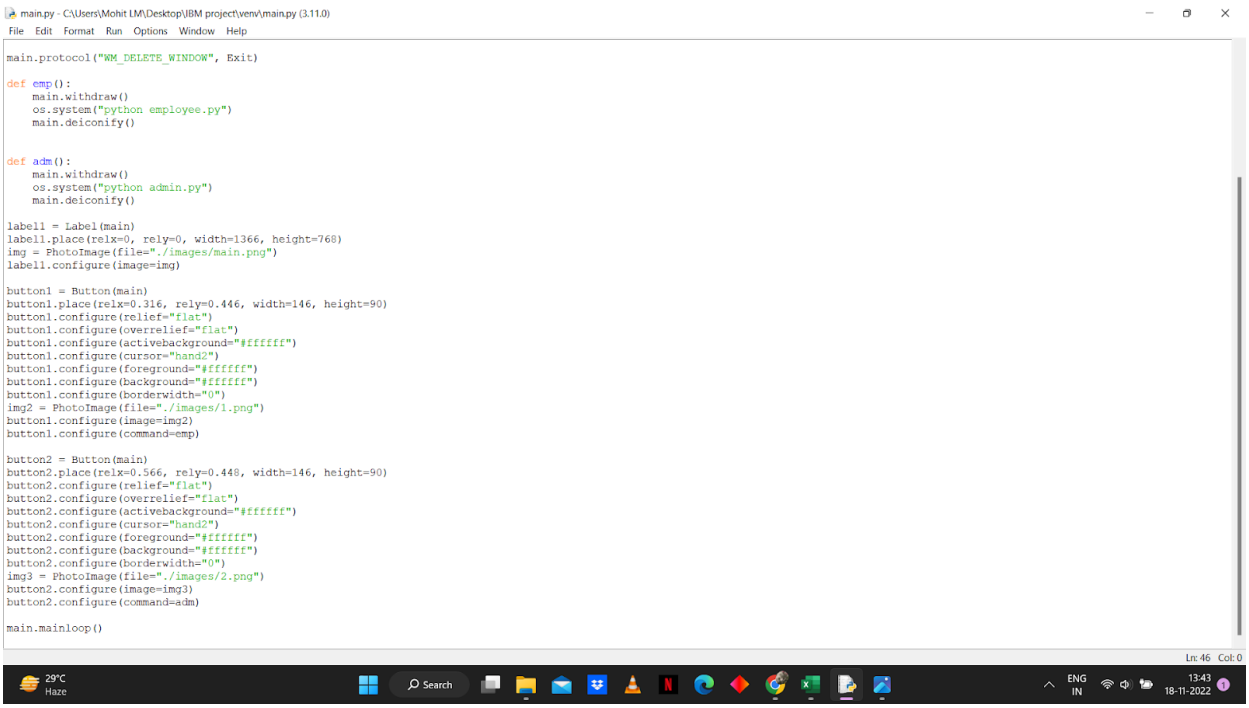
main.py - C:\Users\Mohit LM\Desktop\IBM project\venv\main.py (3.11.0)

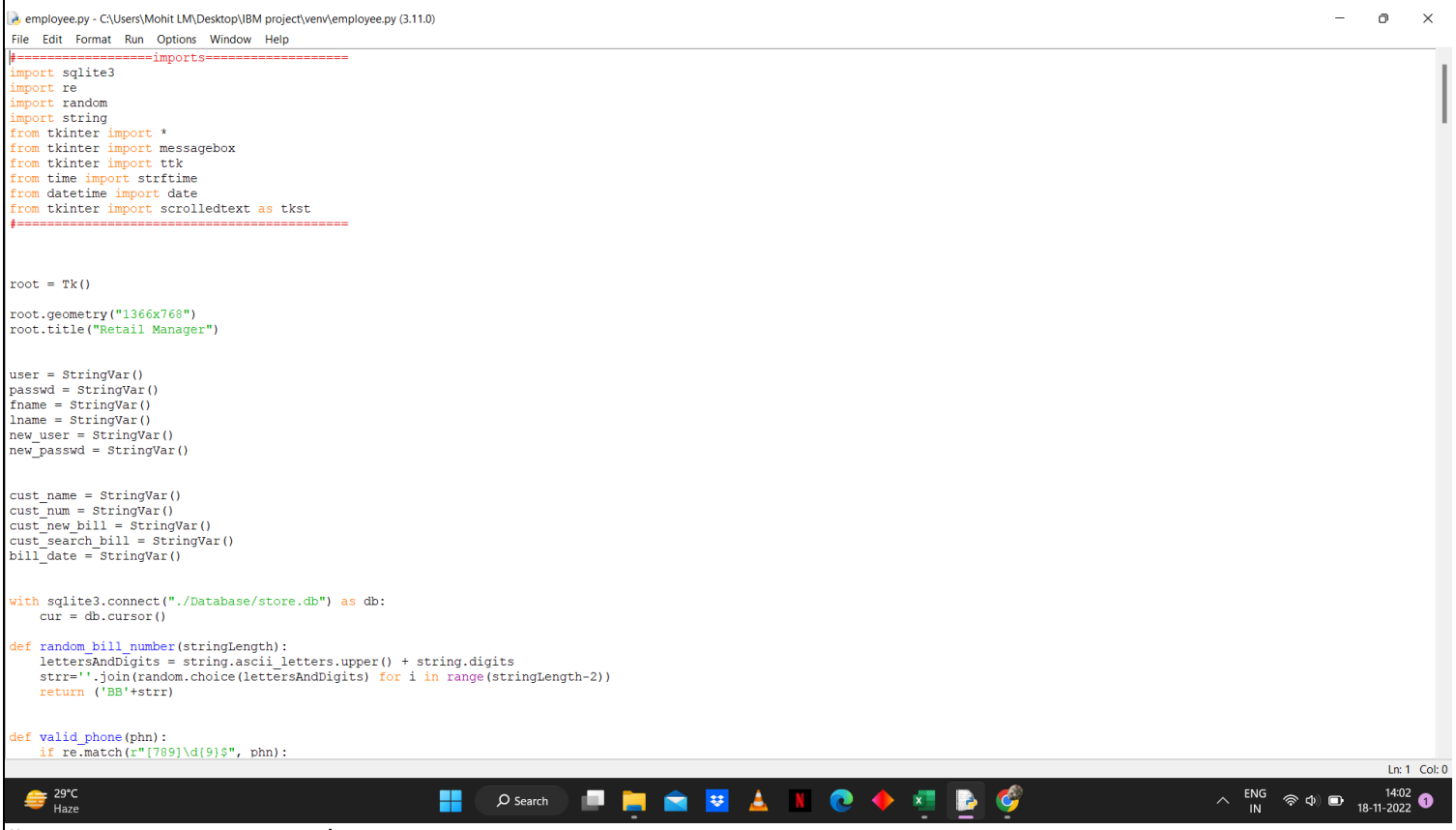
File Edit Format Run Options Window Help

```
__author__ = "macaw"
import os
from tkinter import *
from tkinter import messagebox

main = Tk()
main.geometry("1366x768")
main.title("BIG Bazaar")
main.resizable(0, 0)
def Exit():
    sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=main)
    if sure == True:
        main.destroy()

main.protocol("WM_DELETE_WINDOW", Exit)

def emp():
    main.withdraw()
    os.system("python employee.py")
    main.deiconify()


def adm():
    main.withdraw()
    os.system("python admin.py")
    main.deiconify()

label1 = Label(main)
label1.place(relx=0, rely=0, width=1366, height=768)
img = PhotoImage(file="./images/main.png")
label1.configure(image=img)

button1 = Button(main)
button1.place(relx=0.316, rely=0.446, width=146, height=90)
button1.configure(relief="flat")
button1.configure(overrelief="flat")
button1.configure(activebackground="#ffffff")
button1.configure(cursor="hand2")
button1.configure(foreground="#ffffff")
button1.configure(background="#ffffff")
button1.configure(borderwidth="0")
img2 = PhotoImage(file="./images/1.png")
button1.configure(image=img2)
button1.configure(command=emp)

button2 = Button(main)
button2.place(relx=0.566, rely=0.448, width=146, height=90)
```

Ln: 18 Col: 0

30°C Haze | Search | ENG IN | 12:57 18-11-2022


main.py - C:\Users\Mohit LM\Desktop\IBM project\venv\main.py (3.11.0)
File Edit Format Run Options Window Help
main.protocol("WM_DELETE_WINDOW", Exit)
def emp():
main.withdraw()
os.system("python employee.py")
main.deiconify()
def adm():
main.withdraw()
os.system("python admin.py")
main.deiconify()
label1 = Label(main)
label1.place(relx=0, rely=0, width=1366, height=768)
img = PhotoImage(file="./images/main.png")
label1.configure(image=img)
button1 = Button(main)
button1.place(relx=0.316, rely=0.446, width=146, height=90)
button1.configure(relief="flat")
button1.configure(overrelief="flat")
button1.configure(activebackground="#ffffff")
button1.configure(cursor="hand2")
button1.configure(foreground="#ffffff")
button1.configure(background="#ffffff")
button1.configure(borderwidth="0")
img2 = PhotoImage(file="./images/1.png")
button1.configure(image=img2)
button1.configure(command=emp)
button2 = Button(main)
button2.place(relx=0.566, rely=0.448, width=146, height=90)
button2.configure(relief="flat")
button2.configure(overrelief="flat")
button2.configure(activebackground="#ffffff")
button2.configure(cursor="hand2")
button2.configure(foreground="#ffffff")
button2.configure(background="#ffffff")
button2.configure(borderwidth="0")
img3 = PhotoImage(file="./images/2.png")
button2.configure(image=img3)
button2.configure(command=adm)
main.mainloop()
Ln: 46 Col: 0
29°C
Haze
Search
ENG
IN
13:43
18-11-2022




```
# ==================imports==================
import sqlite3
import re
import random
import string
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from time import strftime
from datetime import date
from tkinter import scrolledtext as tkst
# ============================================

root = Tk()
root.geometry("1366x768")
root.title("Retail Manager(ADMIN)")


user = StringVar()
passwd = StringVar()
fname = StringVar()
lname = StringVar()


with sqlite3.connect("./Database/store.db") as db:
    cur = db.cursor()
```

```
def random_emp_id(stringLength):
    Digits = string.digits
    strr=''.join(random.choice(Digits) for i in range(stringLength-3))
    return ('EMP'+strr)

def valid_phone(phn):
    if re.match(r"[789]\d{9}$", phn):
        return True
    return False

def valid_aadhar(aad):
    if aad.isdigit() and len(aad)==12:
        return True
    return False


class login_page:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("Retail Manager(ADMIN)")

        self.label1 = Label(root)
        self.label1.place(relx=0, rely=0, width=1366, height=768)
        self.img = PhotoImage(file="./images/admin_login.png")
        self.label1.configure(image=self.img)


        self.entry1 = Entry(root)
        self.entry1.place(relx=0.373, rely=0.273, width=374, height=24)
        self.entry1.configure(font="-family {Poppins} -size 10")
        self.entry1.configure(relief="flat")
        self.entry1.configure(textvariable=user)

        self.entry2 = Entry(root)
        self.entry2.place(relx=0.373, rely=0.384, width=374, height=24)
        self.entry2.configure(font="-family {Poppins} -size 10")
        self.entry2.configure(relief="flat")
        self.entry2.configure(show="*")
        self.entry2.configure(textvariable=passwd)

        self.button1 = Button(root)
        self.button1.place(relx=0.366, rely=0.685, width=356, height=43)
        self.button1.configure(relief="flat")
        self.button1.configure(overrelief="flat")
        self.button1.configure(activebackground="#D2463E")
        self.button1.configure(cursor="hand2")
        self.button1.configure(foreground="#ffffff")
        self.button1.configure(background="#D2463E")
        self.button1.configure(font="-family {Poppins SemiBold} -size 20")
        self.button1.configure(borderwidth="0")
```

```
        self.button1.configure(text="""LOGIN""")
        self.button1.configure(command=self.login)

    def login(self, Event=None):
        username = user.get()
        password = passwd.get()

        with sqlite3.connect("./Database/store.db") as db:
            cur = db.cursor()
        find_user = "SELECT * FROM employee WHERE emp_id = ? and password = ?"
        cur.execute(find_user, [username, password])
        results = cur.fetchall()
        if results:
            if results[0][6]=="Admin":
                messagebox.showinfo("Login Page", "The login is successful.")
                page1.entry1.delete(0, END)
                page1.entry2.delete(0, END)

                root.withdraw()
                global adm
                global page2
                adm = Toplevel()
                page2 = Admin_Page(adm)
                #page2.time()
                adm.protocol("WM_DELETE_WINDOW", exitt)
                adm.mainloop()
            else:
                messagebox.showerror("Oops!!", "You are not an admin.")

        else:
            messagebox.showerror("Error", "Incorrect username or password.")
            page1.entry2.delete(0, END)


def exitt():
    sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=root)
    if sure == True:
        adm.destroy()
        root.destroy()

def inventory():
    adm.withdraw()
    global inv
    global page3
    inv = Toplevel()
    page3 = Inventory(inv)
    page3.time()
    inv.protocol("WM_DELETE_WINDOW", exitt)
    inv.mainloop()
```

```
def employee():
    adm.withdraw()
    global emp
    global page5
    emp = Toplevel()
    page5 = Employee(emp)
    page5.time()
    emp.protocol("WM_DELETE_WINDOW", exitt)
    emp.mainloop()


def invoices():
    adm.withdraw()
    global invoice
    invoice = Toplevel()
    page7 = Invoice(invoice)
    page7.time()
    invoice.protocol("WM_DELETE_WINDOW", exitt)
    invoice.mainloop()

def about():
    pass



class Admin_Page:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("ADMIN Mode")

        self.label1 = Label(adm)
        self.label1.place(relx=0, rely=0, width=1366, height=768)
        self.img = PhotoImage(file="./images/admin.png")
        self.label1.configure(image=self.img)

        self.message = Label(adm)
        self.message.place(relx=0.046, rely=0.056, width=62, height=30)
        self.message.configure(font="-family {Poppins} -size 12")
        self.message.configure(foreground="#ffffff")
        self.message.configure(background="#FE6B61")
        self.message.configure(text="""ADMIN""")
        self.message.configure(anchor="w")

        self.button1 = Button(adm)
        self.button1.place(relx=0.035, rely=0.106, width=76, height=23)
        self.button1.configure(relief="flat")
        self.button1.configure(overrelief="flat")
        self.button1.configure(activebackground="#CF1E14")
        self.button1.configure(cursor="hand2")
        self.button1.configure(foreground="#ffffff")
        self.button1.configure(background="#CF1E14")
```

```
self.button1.configure(font="-family {Poppins SemiBold} -size 12")
self.button1.configure(borderwidth="0")
self.button1.configure(text="""Logout""")
self.button1.configure(command=self.Logout)

self.button2 = Button(adm)
self.button2.place(relx=0.14, rely=0.508, width=146, height=63)
self.button2.configure(relief="flat")
self.button2.configure(overrelief="flat")
self.button2.configure(activebackground="#ffffff")
self.button2.configure(cursor="hand2")
self.button2.configure(foreground="#333333")
self.button2.configure(background="#ffffff")
self.button2.configure(font="-family {Poppins SemiBold} -size 12")
self.button2.configure(borderwidth="0")
self.button2.configure(text="""Inventory""")
self.button2.configure(command=inventory)

self.button3 = Button(adm)
self.button3.place(relx=0.338, rely=0.508, width=146, height=63)
self.button3.configure(relief="flat")
self.button3.configure(overrelief="flat")
self.button3.configure(activebackground="#ffffff")
self.button3.configure(cursor="hand2")
self.button3.configure(foreground="#333333")
self.button3.configure(background="#ffffff")
self.button3.configure(font="-family {Poppins SemiBold} -size 12")
self.button3.configure(borderwidth="0")
self.button3.configure(text="""Employees""")
self.button3.configure(command=employee)


self.button4 = Button(adm)
self.button4.place(relx=0.536, rely=0.508, width=146, height=63)
self.button4.configure(relief="flat")
self.button4.configure(overrelief="flat")
self.button4.configure(activebackground="#ffffff")
self.button4.configure(cursor="hand2")
self.button4.configure(foreground="#333333")
self.button4.configure(background="#ffffff")
self.button4.configure(font="-family {Poppins SemiBold} -size 12")
self.button4.configure(borderwidth="0")
self.button4.configure(text="""Invoices""")
self.button4.configure(command=invoices)


self.button5 = Button(adm)
self.button5.place(relx=0.732, rely=0.508, width=146, height=63)
self.button5.configure(relief="flat")
self.button5.configure(overrelief="flat")
self.button5.configure(activebackground="#ffffff")
self.button5.configure(cursor="hand2")
```

```
        self.button5.configure(foreground="#333333")
        self.button5.configure(background="#ffffff")
        self.button5.configure(font="-family {Poppins SemiBold} -size 12")
        self.button5.configure(borderwidth="0")
        self.button5.configure(text="""About Us""")
        self.button5.configure(command=about)

    def Logout(self):
        sure = messagebox.askyesno("Logout", "Are you sure you want to logout?", parent=adm)
        if sure == True:
            adm.destroy()
            root.deiconify()
            page1.entry1.delete(0, END)
            page1.entry2.delete(0, END)


class Inventory:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("Inventory")

        self.label1 = Label(inv)
        self.label1.place(relx=0, rely=0, width=1366, height=768)
        self.img = PhotoImage(file="./images/inventory.png")
        self.label1.configure(image=self.img)

        self.message = Label(inv)
        self.message.place(relx=0.046, rely=0.055, width=136, height=30)
        self.message.configure(font="-family {Poppins} -size 10")
        self.message.configure(foreground="#000000")
        self.message.configure(background="#ffffff")
        self.message.configure(text="""ADMIN""")
        self.message.configure(anchor="w")

        self.clock = Label(inv)
        self.clock.place(relx=0.9, rely=0.065, width=102, height=36)
        self.clock.configure(font="-family {Poppins Light} -size 12")
        self.clock.configure(foreground="#000000")
        self.clock.configure(background="#ffffff")

        self.entry1 = Entry(inv)
        self.entry1.place(relx=0.040, rely=0.286, width=240, height=28)
        self.entry1.configure(font="-family {Poppins} -size 12")
        self.entry1.configure(relief="flat")

        self.button1 = Button(inv)
        self.button1.place(relx=0.229, rely=0.289, width=76, height=23)
        self.button1.configure(relief="flat")
        self.button1.configure(overrelief="flat")
        self.button1.configure(activebackground="#CF1E14")
```

```
self.button1.configure(cursor="hand2")
self.button1.configure(foreground="#ffffff")
self.button1.configure(background="#CF1E14")
self.button1.configure(font="-family {Poppins SemiBold} -size 10")
self.button1.configure(borderwidth="0")
self.button1.configure(text="""Search""")
self.button1.configure(command=self.search_product)

self.button2 = Button(inv)
self.button2.place(relx=0.035, rely=0.106, width=76, height=23)
self.button2.configure(relief="flat")
self.button2.configure(overrelief="flat")
self.button2.configure(activebackground="#CF1E14")
self.button2.configure(cursor="hand2")
self.button2.configure(foreground="#ffffff")
self.button2.configure(background="#CF1E14")
self.button2.configure(font="-family {Poppins SemiBold} -size 12")
self.button2.configure(borderwidth="0")
self.button2.configure(text="""Logout""")
self.button2.configure(command=self.Logout)

self.button3 = Button(inv)
self.button3.place(relx=0.052, rely=0.432, width=306, height=28)
self.button3.configure(relief="flat")
self.button3.configure(overrelief="flat")
self.button3.configure(activebackground="#CF1E14")
self.button3.configure(cursor="hand2")
self.button3.configure(foreground="#ffffff")
self.button3.configure(background="#CF1E14")
self.button3.configure(font="-family {Poppins SemiBold} -size 12")
self.button3.configure(borderwidth="0")
self.button3.configure(text="""ADD PRODUCT""")
self.button3.configure(command=self.add_product)

self.button4 = Button(inv)
self.button4.place(relx=0.052, rely=0.5, width=306, height=28)
self.button4.configure(relief="flat")
self.button4.configure(overrelief="flat")
self.button4.configure(activebackground="#CF1E14")
self.button4.configure(cursor="hand2")
self.button4.configure(foreground="#ffffff")
self.button4.configure(background="#CF1E14")
self.button4.configure(font="-family {Poppins SemiBold} -size 12")
self.button4.configure(borderwidth="0")
self.button4.configure(text="""UPDATE PRODUCT""")
self.button4.configure(command=self.update_product)

self.button5 = Button(inv)
self.button5.place(relx=0.052, rely=0.57, width=306, height=28)
self.button5.configure(relief="flat")
self.button5.configure(overrelief="flat")
self.button5.configure(activebackground="#CF1E14")
```

```
self.button5.configure(cursor="hand2")
self.button5.configure(foreground="#ffffff")
self.button5.configure(background="#CF1E14")
self.button5.configure(font="-family {Poppins SemiBold} -size 12")
self.button5.configure(borderwidth="0")
self.button5.configure(text="""DELETE PRODUCT""")
self.button5.configure(command=self.delete_product)

self.button6 = Button(inv)
self.button6.place(relx=0.135, rely=0.885, width=76, height=23)
self.button6.configure(relief="flat")
self.button6.configure(overrelief="flat")
self.button6.configure(activebackground="#CF1E14")
self.button6.configure(cursor="hand2")
self.button6.configure(foreground="#ffffff")
self.button6.configure(background="#CF1E14")
self.button6.configure(font="-family {Poppins SemiBold} -size 12")
self.button6.configure(borderwidth="0")
self.button6.configure(text="""EXIT""")
self.button6.configure(command=self.Exit)

self.scrollbarx = Scrollbar(inv, orient=HORIZONTAL)
self.scrollbary = Scrollbar(inv, orient=VERTICAL)
self.tree = ttk.Treeview(inv)
self.tree.place(relx=0.307, rely=0.203, width=880, height=550)
self.tree.configure(
    yscrollcommand=self.scrollbary.set, xscrollcommand=self.scrollbarx.set
)
self.tree.configure(selectmode="extended")

self.tree.bind("<<TreeviewSelect>>", self.on_tree_select)

self.scrollbary.configure(command=self.tree.yview)
self.scrollbarx.configure(command=self.tree.xview)

self.scrollbary.place(relx=0.954, rely=0.203, width=22, height=548)
self.scrollbarx.place(relx=0.307, rely=0.924, width=884, height=22)

self.tree.configure(
    columns=(
        "Product ID",
        "Name",
        "Category",
        "Sub-Category",
        "In Stock",
        "MRP",
        "Cost Price",
        "Vendor No.",
    )
)
```

```
    self.tree.heading("Product ID", text="Product ID", anchor=W)
    self.tree.heading("Name", text="Name", anchor=W)
    self.tree.heading("Category", text="Category", anchor=W)
    self.tree.heading("Sub-Category", text="Sub-Category", anchor=W)
    self.tree.heading("In Stock", text="In Stock", anchor=W)
    self.tree.heading("MRP", text="MRP", anchor=W)
    self.tree.heading("Cost Price", text="Cost Price", anchor=W)
    self.tree.heading("Vendor No.", text="Vendor No.", anchor=W)

    self.tree.column("#0", stretch=NO, minwidth=0, width=0)
    self.tree.column("#1", stretch=NO, minwidth=0, width=80)
    self.tree.column("#2", stretch=NO, minwidth=0, width=260)
    self.tree.column("#3", stretch=NO, minwidth=0, width=100)
    self.tree.column("#4", stretch=NO, minwidth=0, width=120)
    self.tree.column("#5", stretch=NO, minwidth=0, width=80)
    self.tree.column("#6", stretch=NO, minwidth=0, width=80)
    self.tree.column("#7", stretch=NO, minwidth=0, width=80)
    self.tree.column("#8", stretch=NO, minwidth=0, width=100)

    self.DisplayData()

def DisplayData(self):
    cur.execute("SELECT * FROM raw_inventory")
    fetch = cur.fetchall()
    for data in fetch:
        self.tree.insert("", "end", values=(data))

def search_product(self):
    val = []
    for i in self.tree.get_children():
        val.append(i)
        for j in self.tree.item(i)["values"]:
            val.append(j)

    try:
        to_search = int(self.entry1.get())
    except ValueError:
        messagebox.showerror("Oops!!", "Invalid Product Id.", parent=inv)
    else:
        for search in val:
            if search==to_search:
                self.tree.selection_set(val[val.index(search)-1])
                self.tree.focus(val[val.index(search)-1])
                messagebox.showinfo("Success!!", "Product ID: {} found.".format(self.entry1.get()), parent=inv)
                break
        else:
            messagebox.showerror("Oops!!", "Product ID: {} not found.".format(self.entry1.get()), parent=inv)

sel = []
def on_tree_select(self, Event):
    self.sel.clear()
    for i in self.tree.selection():
```

```
        if i not in self.sel:
            self.sel.append(i)

def delete_product(self):
    val = []
    to_delete = []

    if len(self.sel)!=0:
        sure = messagebox.askyesno("Confirm", "Are you sure you want to delete selected products?", parent=inv)
        if sure == True:
            for i in self.sel:
                for j in self.tree.item(i)["values"]:
                    val.append(j)

            for j in range(len(val)):
                if j%8==0:
                    to_delete.append(val[j])

            for k in to_delete:
                delete = "DELETE FROM raw_inventory WHERE product_id = ?"
                cur.execute(delete, [k])
                db.commit()

            messagebox.showinfo("Success!!", "Products deleted from database.", parent=inv)
            self.sel.clear()
            self.tree.delete(*self.tree.get_children())

            self.DisplayData()
    else:
        messagebox.showerror("Error!!","Please select a product.", parent=inv)

def update_product(self):
    if len(self.sel)==1:
        global p_update
        p_update = Toplevel()
        page9 = Update_Product(p_update)
        page9.time()
        p_update.protocol("WM_DELETE_WINDOW", self.ex2)
        global valll
        valll = []
        for i in self.sel:
            for j in self.tree.item(i)["values"]:
                valll.append(j)

        page9.entry1.insert(0, valll[1])
        page9.entry2.insert(0, valll[2])
        page9.entry3.insert(0, valll[4])
        page9.entry4.insert(0, valll[5])
        page9.entry6.insert(0, valll[3])
        page9.entry7.insert(0, valll[6])
        page9.entry8.insert(0, valll[7])
```

```
        elif len(self.sel)==0:
            messagebox.showerror("Error","Please choose a product to update.", parent=inv)
        else:
            messagebox.showerror("Error","Can only update one product at a time.", parent=inv)

        p_update.mainloop()



    def add_product(self):
        global p_add
        global page4
        p_add = Toplevel()
        page4 = add_product(p_add)
        page4.time()
        p_add.mainloop()

    def time(self):
        string = strftime("%H:%M:%S %p")
        self.clock.config(text=string)
        self.clock.after(1000, self.time)

    def Exit(self):
        sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=inv)
        if sure == True:
            inv.destroy()
            adm.deiconify()

    def ex2(self):
        sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=p_update)
        if sure == True:
            p_update.destroy()
            inv.deiconify()



    def Logout(self):
        sure = messagebox.askyesno("Logout", "Are you sure you want to logout?")
        if sure == True:
            root.deiconify()
            page1.entry1.delete(0, END)
            page1.entry2.delete(0, END)


class add_product:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("Add Product")
```

```
self.label1 = Label(p_add)
self.label1.place(relx=0, rely=0, width=1366, height=768)
self.img = PhotoImage(file="./images/add_product.png")
self.label1.configure(image=self.img)

self.clock = Label(p_add)
self.clock.place(relx=0.84, rely=0.065, width=102, height=36)
self.clock.configure(font="-family {Poppins Light} -size 12")
self.clock.configure(foreground="#000000")
self.clock.configure(background="#ffffff")

self.entry1 = Entry(p_add)
self.entry1.place(relx=0.132, rely=0.296, width=996, height=30)
self.entry1.configure(font="-family {Poppins} -size 12")
self.entry1.configure(relief="flat")

self.entry2 = Entry(p_add)
self.entry2.place(relx=0.132, rely=0.413, width=374, height=30)
self.entry2.configure(font="-family {Poppins} -size 12")
self.entry2.configure(relief="flat")

self.r2 = p_add.register(self.testint)

self.entry3 = Entry(p_add)
self.entry3.place(relx=0.132, rely=0.529, width=374, height=30)
self.entry3.configure(font="-family {Poppins} -size 12")
self.entry3.configure(relief="flat")
self.entry3.configure(validate="key", validatecommand=(self.r2, '%P'))

self.entry4 = Entry(p_add)
self.entry4.place(relx=0.132, rely=0.646, width=374, height=30)
self.entry4.configure(font="-family {Poppins} -size 12")
self.entry4.configure(relief="flat")


self.entry6 = Entry(p_add)
self.entry6.place(relx=0.527, rely=0.413, width=374, height=30)
self.entry6.configure(font="-family {Poppins} -size 12")
self.entry6.configure(relief="flat")


self.entry7 = Entry(p_add)
self.entry7.place(relx=0.527, rely=0.529, width=374, height=30)
self.entry7.configure(font="-family {Poppins} -size 12")
self.entry7.configure(relief="flat")


self.entry8 = Entry(p_add)
self.entry8.place(relx=0.527, rely=0.646, width=374, height=30)
self.entry8.configure(font="-family {Poppins} -size 12")
```

```
    self.entry8.configure(relief="flat")
    self.entry8.configure(validate="key", validatecommand=(self.r2, "%P"))


    self.button1 = Button(p_add)
    self.button1.place(relx=0.408, rely=0.836, width=96, height=34)
    self.button1.configure(relief="flat")
    self.button1.configure(overrelief="flat")
    self.button1.configure(activebackground="#CF1E14")
    self.button1.configure(cursor="hand2")
    self.button1.configure(foreground="#ffffff")
    self.button1.configure(background="#CF1E14")
    self.button1.configure(font="-family {Poppins SemiBold} -size 14")
    self.button1.configure(borderwidth="0")
    self.button1.configure(text="""ADD""")
    self.button1.configure(command=self.add)

    self.button2 = Button(p_add)
    self.button2.place(relx=0.526, rely=0.836, width=86, height=34)
    self.button2.configure(relief="flat")
    self.button2.configure(overrelief="flat")
    self.button2.configure(activebackground="#CF1E14")
    self.button2.configure(cursor="hand2")
    self.button2.configure(foreground="#ffffff")
    self.button2.configure(background="#CF1E14")
    self.button2.configure(font="-family {Poppins SemiBold} -size 14")
    self.button2.configure(borderwidth="0")
    self.button2.configure(text="""CLEAR""")
    self.button2.configure(command=self.clearr)

def add(self):
    pqty = self.entry3.get()
    pcat = self.entry2.get()
    pmrp = self.entry4.get()
    pname = self.entry1.get()
    psubcat = self.entry6.get()
    pcp = self.entry7.get()
    pvendor = self.entry8.get()


    if pname.strip():
        if pcat.strip():
            if psubcat.strip():
                if pqty:
                    if pcp:
                        try:
                            float(pcp)
                        except ValueError:
                            messagebox.showerror("Oops!", "Invalid cost price.", parent=p_add)
                        else:
                            if pmrp:
                                try:
```

```
                            float(pmrp)
                        except ValueError:
                            messagebox.showerror("Oops!", "Invalid MRP.", parent=p_add)
                        else:
                            if valid_phone(pvendor):
                                with sqlite3.connect("./Database/store.db") as db:
                                    cur = db.cursor()
                                insert = (
                                        "INSERT INTO raw_inventory(product_name, product_cat, product_subcat,
stock, mrp, cost_price, vendor_phn) VALUES(?,?,?,?,?,?,?)"
                                    )
                                cur.execute(insert, [pname, pcat, psubcat, int(pqty), float(pmrp), float(pcp), pvendor])
                                db.commit()
                                messagebox.showinfo("Success!!", "Product successfully added in inventory.",
parent=p_add)
                                p_add.destroy()
                                page3.tree.delete(*page3.tree.get_children())
                                page3.DisplayData()
                                p_add.destroy()
                            else:
                                messagebox.showerror("Oops!", "Invalid phone number.", parent=p_add)
                    else:
                        messagebox.showerror("Oops!", "Please enter MRP.", parent=p_add)
                else:
                    messagebox.showerror("Oops!", "Please enter product cost price.", parent=p_add)
            else:
                messagebox.showerror("Oops!", "Please enter product quantity.", parent=p_add)
        else:
            messagebox.showerror("Oops!", "Please enter product sub-category.", parent=p_add)
    else:
        messagebox.showerror("Oops!", "Please enter product category.", parent=p_add)
else:
    messagebox.showerror("Oops!", "Please enter product name", parent=p_add)

def clearr(self):
    self.entry1.delete(0, END)
    self.entry2.delete(0, END)
    self.entry3.delete(0, END)
    self.entry4.delete(0, END)
    self.entry6.delete(0, END)
    self.entry7.delete(0, END)
    self.entry8.delete(0, END)

def testint(self, val):
    if val.isdigit():
        return True
    elif val == "":
        return True
    return False

def time(self):
    string = strftime("%H:%M:%S %p")
```

```
        self.clock.config(text=string)
        self.clock.after(1000, self.time)


class Update_Product:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("Add Product")

        self.label1 = Label(p_update)
        self.label1.place(relx=0, rely=0, width=1366, height=768)
        self.img = PhotoImage(file="./images/update_product.png")
        self.label1.configure(image=self.img)

        self.clock = Label(p_update)
        self.clock.place(relx=0.84, rely=0.065, width=102, height=36)
        self.clock.configure(font="-family {Poppins Light} -size 12")
        self.clock.configure(foreground="#000000")
        self.clock.configure(background="#ffffff")

        self.entry1 = Entry(p_update)
        self.entry1.place(relx=0.132, rely=0.296, width=996, height=30)
        self.entry1.configure(font="-family {Poppins} -size 12")
        self.entry1.configure(relief="flat")

        self.entry2 = Entry(p_update)
        self.entry2.place(relx=0.132, rely=0.413, width=374, height=30)
        self.entry2.configure(font="-family {Poppins} -size 12")
        self.entry2.configure(relief="flat")

        self.r2 = p_update.register(self.testint)

        self.entry3 = Entry(p_update)
        self.entry3.place(relx=0.132, rely=0.529, width=374, height=30)
        self.entry3.configure(font="-family {Poppins} -size 12")
        self.entry3.configure(relief="flat")
        self.entry3.configure(validate="key", validatecommand=(self.r2, "%P"))

        self.entry4 = Entry(p_update)
        self.entry4.place(relx=0.132, rely=0.646, width=374, height=30)
        self.entry4.configure(font="-family {Poppins} -size 12")
        self.entry4.configure(relief="flat")


        self.entry6 = Entry(p_update)
        self.entry6.place(relx=0.527, rely=0.413, width=374, height=30)
        self.entry6.configure(font="-family {Poppins} -size 12")
        self.entry6.configure(relief="flat")
```

```
    self.entry7 = Entry(p_update)
    self.entry7.place(relx=0.527, rely=0.529, width=374, height=30)
    self.entry7.configure(font="-family {Poppins} -size 12")
    self.entry7.configure(relief="flat")


    self.entry8 = Entry(p_update)
    self.entry8.place(relx=0.527, rely=0.646, width=374, height=30)
    self.entry8.configure(font="-family {Poppins} -size 12")
    self.entry8.configure(relief="flat")


    self.button1 = Button(p_update)
    self.button1.place(relx=0.408, rely=0.836, width=96, height=34)
    self.button1.configure(relief="flat")
    self.button1.configure(overrelief="flat")
    self.button1.configure(activebackground="#CF1E14")
    self.button1.configure(cursor="hand2")
    self.button1.configure(foreground="#ffffff")
    self.button1.configure(background="#CF1E14")
    self.button1.configure(font="-family {Poppins SemiBold} -size 14")
    self.button1.configure(borderwidth="0")
    self.button1.configure(text="""UPDATE""")
    self.button1.configure(command=self.update)

    self.button2 = Button(p_update)
    self.button2.place(relx=0.526, rely=0.836, width=86, height=34)
    self.button2.configure(relief="flat")
    self.button2.configure(overrelief="flat")
    self.button2.configure(activebackground="#CF1E14")
    self.button2.configure(cursor="hand2")
    self.button2.configure(foreground="#ffffff")
    self.button2.configure(background="#CF1E14")
    self.button2.configure(font="-family {Poppins SemiBold} -size 14")
    self.button2.configure(borderwidth="0")
    self.button2.configure(text="""CLEAR""")
    self.button2.configure(command=self.clearr)

def update(self):
    pqty = self.entry3.get()
    pcat = self.entry2.get()
    pmrp = self.entry4.get()
    pname = self.entry1.get()
    psubcat = self.entry6.get()
    pcp = self.entry7.get()
    pvendor = self.entry8.get()


    if pname.strip():
        if pcat.strip():
            if psubcat.strip():
                if pqty:
```

```
                if pcp:
                    try:
                        float(pcp)
                    except ValueError:
                        messagebox.showerror("Oops!", "Invalid cost price.", parent=p_update)
                    else:
                        if pmrp:
                            try:
                                float(pmrp)
                            except ValueError:
                                messagebox.showerror("Oops!", "Invalid MRP.", parent=p_update)
                            else:
                                if valid_phone(pvendor):
                                    product_id = valll[0]
                                    with sqlite3.connect("./Database/store.db") as db:
                                        cur = db.cursor()
                                    update = (
                                    "UPDATE raw_inventory SET product_name = ?, product_cat = ?, product_subcat = ?,
stock = ?, mrp = ?, cost_price = ?, vendor_phn = ? WHERE product_id = ?"
                                    )
                                    cur.execute(update, [pname, pcat, psubcat, int(pqty), float(pmrp), float(pcp), pvendor,
product_id])
                                    db.commit()
                                    messagebox.showinfo("Success!!", "Product successfully updated in inventory.",
parent=p_update)
                                    valll.clear()
                                    Inventory.sel.clear()
                                    page3.tree.delete(*page3.tree.get_children())
                                    page3.DisplayData()
                                    p_update.destroy()
                                else:
                                    messagebox.showerror("Oops!", "Invalid phone number.", parent=p_update)
                            else:
                                messagebox.showerror("Oops!", "Please enter MRP.", parent=p_update)
                        else:
                            messagebox.showerror("Oops!", "Please enter product cost price.", parent=p_update)
                else:
                    messagebox.showerror("Oops!", "Please enter product quantity.", parent=p_update)
            else:
                messagebox.showerror("Oops!", "Please enter product sub-category.", parent=p_update)
        else:
            messagebox.showerror("Oops!", "Please enter product category.", parent=p_update)
    else:
        messagebox.showerror("Oops!", "Please enter product name", parent=p_update)

  def clearr(self):
    self.entry1.delete(0, END)
    self.entry2.delete(0, END)
    self.entry3.delete(0, END)
    self.entry4.delete(0, END)
    self.entry6.delete(0, END)
    self.entry7.delete(0, END)
```

```
        self.entry8.delete(0, END)

    def testint(self, val):
        if val.isdigit():
            return True
        elif val == "":
            return True
        return False

    def time(self):
        string = strftime("%H:%M:%S %p")
        self.clock.config(text=string)
        self.clock.after(1000, self.time)



class Employee:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("Employee Management")

        self.label1 = Label(emp)
        self.label1.place(relx=0, rely=0, width=1366, height=768)
        self.img = PhotoImage(file="./images/employee.png")
        self.label1.configure(image=self.img)

        self.message = Label(emp)
        self.message.place(relx=0.046, rely=0.055, width=136, height=30)
        self.message.configure(font="-family {Poppins} -size 10")
        self.message.configure(foreground="#000000")
        self.message.configure(background="#ffffff")
        self.message.configure(text="""ADMIN""")
        self.message.configure(anchor="w")

        self.clock = Label(emp)
        self.clock.place(relx=0.9, rely=0.065, width=102, height=36)
        self.clock.configure(font="-family {Poppins Light} -size 12")
        self.clock.configure(foreground="#000000")
        self.clock.configure(background="#ffffff")

        self.entry1 = Entry(emp)
        self.entry1.place(relx=0.040, rely=0.286, width=240, height=28)
        self.entry1.configure(font="-family {Poppins} -size 12")
        self.entry1.configure(relief="flat")

        self.button1 = Button(emp)
        self.button1.place(relx=0.229, rely=0.289, width=76, height=23)
        self.button1.configure(relief="flat")
        self.button1.configure(overrelief="flat")
        self.button1.configure(activebackground="#CF1E14")
```

```
self.button1.configure(cursor="hand2")
self.button1.configure(foreground="#ffffff")
self.button1.configure(background="#CF1E14")
self.button1.configure(font="-family {Poppins SemiBold} -size 10")
self.button1.configure(borderwidth="0")
self.button1.configure(text="""Search""")
self.button1.configure(command=self.search_emp)

self.button2 = Button(emp)
self.button2.place(relx=0.035, rely=0.106, width=76, height=23)
self.button2.configure(relief="flat")
self.button2.configure(overrelief="flat")
self.button2.configure(activebackground="#CF1E14")
self.button2.configure(cursor="hand2")
self.button2.configure(foreground="#ffffff")
self.button2.configure(background="#CF1E14")
self.button2.configure(font="-family {Poppins SemiBold} -size 12")
self.button2.configure(borderwidth="0")
self.button2.configure(text="""Logout""")
self.button2.configure(command=self.Logout)

self.button3 = Button(emp)
self.button3.place(relx=0.052, rely=0.432, width=306, height=28)
self.button3.configure(relief="flat")
self.button3.configure(overrelief="flat")
self.button3.configure(activebackground="#CF1E14")
self.button3.configure(cursor="hand2")
self.button3.configure(foreground="#ffffff")
self.button3.configure(background="#CF1E14")
self.button3.configure(font="-family {Poppins SemiBold} -size 12")
self.button3.configure(borderwidth="0")
self.button3.configure(text="""ADD EMPLOYEE""")
self.button3.configure(command=self.add_emp)

self.button4 = Button(emp)
self.button4.place(relx=0.052, rely=0.5, width=306, height=28)
self.button4.configure(relief="flat")
self.button4.configure(overrelief="flat")
self.button4.configure(activebackground="#CF1E14")
self.button4.configure(cursor="hand2")
self.button4.configure(foreground="#ffffff")
self.button4.configure(background="#CF1E14")
self.button4.configure(font="-family {Poppins SemiBold} -size 12")
self.button4.configure(borderwidth="0")
self.button4.configure(text="""UPDATE EMPLOYEE""")
self.button4.configure(command=self.update_emp)

self.button5 = Button(emp)
self.button5.place(relx=0.052, rely=0.57, width=306, height=28)
self.button5.configure(relief="flat")
self.button5.configure(overrelief="flat")
self.button5.configure(activebackground="#CF1E14")
```

```
self.button5.configure(cursor="hand2")
self.button5.configure(foreground="#ffffff")
self.button5.configure(background="#CF1E14")
self.button5.configure(font="-family {Poppins SemiBold} -size 12")
self.button5.configure(borderwidth="0")
self.button5.configure(text="""DELETE EMPLOYEE""")
self.button5.configure(command=self.delete_emp)

self.button6 = Button(emp)
self.button6.place(relx=0.135, rely=0.885, width=76, height=23)
self.button6.configure(relief="flat")
self.button6.configure(overrelief="flat")
self.button6.configure(activebackground="#CF1E14")
self.button6.configure(cursor="hand2")
self.button6.configure(foreground="#ffffff")
self.button6.configure(background="#CF1E14")
self.button6.configure(font="-family {Poppins SemiBold} -size 12")
self.button6.configure(borderwidth="0")
self.button6.configure(text="""EXIT""")
self.button6.configure(command=self.Exit)

self.scrollbarx = Scrollbar(emp, orient=HORIZONTAL)
self.scrollbary = Scrollbar(emp, orient=VERTICAL)
self.tree = ttk.Treeview(emp)
self.tree.place(relx=0.307, rely=0.203, width=880, height=550)
self.tree.configure(
    yscrollcommand=self.scrollbary.set, xscrollcommand=self.scrollbarx.set
)
self.tree.configure(selectmode="extended")

self.tree.bind("<<TreeviewSelect>>", self.on_tree_select)

self.scrollbary.configure(command=self.tree.yview)
self.scrollbarx.configure(command=self.tree.xview)

self.scrollbary.place(relx=0.954, rely=0.203, width=22, height=548)
self.scrollbarx.place(relx=0.307, rely=0.924, width=884, height=22)

self.tree.configure(
    columns=(
        "Employee ID",
        "Employee Name",
        "Contact No.",
        "Address",
        "Aadhar No.",
        "Password",
        "Designation"
    )
)

self.tree.heading("Employee ID", text="Employee ID", anchor=W)
```

```
    self.tree.heading("Employee Name", text="Employee Name", anchor=W)
    self.tree.heading("Contact No.", text="Contact No.", anchor=W)
    self.tree.heading("Address", text="Address", anchor=W)
    self.tree.heading("Aadhar No.", text="Aadhar No.", anchor=W)
    self.tree.heading("Password", text="Password", anchor=W)
    self.tree.heading("Designation", text="Designation", anchor=W)

    self.tree.column("#0", stretch=NO, minwidth=0, width=0)
    self.tree.column("#1", stretch=NO, minwidth=0, width=80)
    self.tree.column("#2", stretch=NO, minwidth=0, width=260)
    self.tree.column("#3", stretch=NO, minwidth=0, width=100)
    self.tree.column("#4", stretch=NO, minwidth=0, width=198)
    self.tree.column("#5", stretch=NO, minwidth=0, width=80)
    self.tree.column("#6", stretch=NO, minwidth=0, width=80)
    self.tree.column("#7", stretch=NO, minwidth=0, width=80)

    self.DisplayData()

def DisplayData(self):
    cur.execute("SELECT * FROM employee")
    fetch = cur.fetchall()
    for data in fetch:
        self.tree.insert("", "end", values=(data))

def search_emp(self):
    val = []
    for i in self.tree.get_children():
        val.append(i)
        for j in self.tree.item(i)["values"]:
            val.append(j)

    to_search = self.entry1.get()
    for search in val:
        if search==to_search:
            self.tree.selection_set(val[val.index(search)-1])
            self.tree.focus(val[val.index(search)-1])
            messagebox.showinfo("Success!!", "Employee ID: {} found.".format(self.entry1.get()), parent=emp)
            break
    else:
        messagebox.showerror("Oops!!", "Employee ID: {} not found.".format(self.entry1.get()), parent=emp)

sel = []
def on_tree_select(self, Event):
    self.sel.clear()
    for i in self.tree.selection():
        if i not in self.sel:
            self.sel.append(i)

def delete_emp(self):
    val = []
    to_delete = []
```

```
    if len(self.sel)!=0:
        sure = messagebox.askyesno("Confirm", "Are you sure you want to delete selected employee(s)?",
parent=emp)
        if sure == True:
            for i in self.sel:
                for j in self.tree.item(i)["values"]:
                    val.append(j)

            for j in range(len(val)):
                if j%7==0:
                    to_delete.append(val[j])

            flag = 1

            for k in to_delete:
                if k=="EMP0000":
                    flag = 0
                    break
                else:
                    delete = "DELETE FROM employee WHERE emp_id = ?"
                    cur.execute(delete, [k])
                    db.commit()

            if flag==1:
                messagebox.showinfo("Success!!", "Employee(s) deleted from database.", parent=emp)
                self.sel.clear()
                self.tree.delete(*self.tree.get_children())
                self.DisplayData()
            else:
                messagebox.showerror("Error!!","Cannot delete master admin.")
    else:
        messagebox.showerror("Error!!","Please select an employee.", parent=emp)

  def update_emp(self):

    if len(self.sel)==1:
        global e_update
        e_update = Toplevel()
        page8 = Update_Employee(e_update)
        page8.time()
        e_update.protocol("WM_DELETE_WINDOW", self.ex2)
        global vall
        vall = []
        for i in self.sel:
            for j in self.tree.item(i)["values"]:
                vall.append(j)

        page8.entry1.insert(0, vall[1])
        page8.entry2.insert(0, vall[2])
        page8.entry3.insert(0, vall[4])
        page8.entry4.insert(0, vall[6])
```

```
            page8.entry5.insert(0, vall[3])
            page8.entry6.insert(0, vall[5])
            e_update.mainloop()
        elif len(self.sel)==0:
            messagebox.showerror("Error","Please select an employee to update.")
        else:
            messagebox.showerror("Error","Can only update one employee at a time.")




def add_emp(self):
    global e_add
    e_add = Toplevel()
    page6 = add_employee(e_add)
    page6.time()
    e_add.protocol("WM_DELETE_WINDOW", self.ex)
    e_add.mainloop()


def ex(self):
    e_add.destroy()
    self.tree.delete(*self.tree.get_children())
    self.DisplayData()

def ex2(self):
    e_update.destroy()
    self.tree.delete(*self.tree.get_children())
    self.DisplayData()



def time(self):
    string = strftime("%H:%M:%S %p")
    self.clock.config(text=string)
    self.clock.after(1000, self.time)

def Exit(self):
    sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=emp)
    if sure == True:
        emp.destroy()
        adm.deiconify()


def Logout(self):
    sure = messagebox.askyesno("Logout", "Are you sure you want to logout?")
    if sure == True:
        emp.destroy()
        root.deiconify()

        page1.entry1.delete(0, END)
        page1.entry2.delete(0, END)
```

```
class add_employee:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
        top.title("Add Employee")

        self.label1 = Label(e_add)
        self.label1.place(relx=0, rely=0, width=1366, height=768)
        self.img = PhotoImage(file="./images/add_employee.png")
        self.label1.configure(image=self.img)

        self.clock = Label(e_add)
        self.clock.place(relx=0.84, rely=0.065, width=102, height=36)
        self.clock.configure(font="-family {Poppins Light} -size 12")
        self.clock.configure(foreground="#000000")
        self.clock.configure(background="#ffffff")

        self.r1 = e_add.register(self.testint)
        self.r2 = e_add.register(self.testchar)

        self.entry1 = Entry(e_add)
        self.entry1.place(relx=0.132, rely=0.296, width=374, height=30)
        self.entry1.configure(font="-family {Poppins} -size 12")
        self.entry1.configure(relief="flat")


        self.entry2 = Entry(e_add)
        self.entry2.place(relx=0.132, rely=0.413, width=374, height=30)
        self.entry2.configure(font="-family {Poppins} -size 12")
        self.entry2.configure(relief="flat")
        self.entry2.configure(validate="key", validatecommand=(self.r1, "%P"))

        self.entry3 = Entry(e_add)
        self.entry3.place(relx=0.132, rely=0.529, width=374, height=30)
        self.entry3.configure(font="-family {Poppins} -size 12")
        self.entry3.configure(relief="flat")
        self.entry3.configure(validate="key", validatecommand=(self.r1, "%P"))

        self.entry4 = Entry(e_add)
        self.entry4.place(relx=0.527, rely=0.296, width=374, height=30)
        self.entry4.configure(font="-family {Poppins} -size 12")
        self.entry4.configure(relief="flat")
        self.entry4.configure(validate="key", validatecommand=(self.r2, "%P"))

        self.entry5 = Entry(e_add)
        self.entry5.place(relx=0.527, rely=0.413, width=374, height=30)
        self.entry5.configure(font="-family {Poppins} -size 12")
        self.entry5.configure(relief="flat")
```

```
    self.entry6 = Entry(e_add)
    self.entry6.place(relx=0.527, rely=0.529, width=374, height=30)
    self.entry6.configure(font="-family {Poppins} -size 12")
    self.entry6.configure(relief="flat")
    self.entry6.configure(show="*")

    self.button1 = Button(e_add)
    self.button1.place(relx=0.408, rely=0.836, width=96, height=34)
    self.button1.configure(relief="flat")
    self.button1.configure(overrelief="flat")
    self.button1.configure(activebackground="#CF1E14")
    self.button1.configure(cursor="hand2")
    self.button1.configure(foreground="#ffffff")
    self.button1.configure(background="#CF1E14")
    self.button1.configure(font="-family {Poppins SemiBold} -size 14")
    self.button1.configure(borderwidth="0")
    self.button1.configure(text="""ADD""")
    self.button1.configure(command=self.add)

    self.button2 = Button(e_add)
    self.button2.place(relx=0.526, rely=0.836, width=86, height=34)
    self.button2.configure(relief="flat")
    self.button2.configure(overrelief="flat")
    self.button2.configure(activebackground="#CF1E14")
    self.button2.configure(cursor="hand2")
    self.button2.configure(foreground="#ffffff")
    self.button2.configure(background="#CF1E14")
    self.button2.configure(font="-family {Poppins SemiBold} -size 14")
    self.button2.configure(borderwidth="0")
    self.button2.configure(text="""CLEAR""")
    self.button2.configure(command=self.clearr)



def testint(self, val):
    if val.isdigit():
        return True
    elif val == "":
        return True
    return False

def testchar(self, val):
    if val.isalpha():
        return True
    elif val == "":
        return True
    return False

def time(self):
    string = strftime("%H:%M:%S %p")
    self.clock.config(text=string)
    self.clock.after(1000, self.time)
```

```
    def add(self):
        ename = self.entry1.get()
        econtact = self.entry2.get()
        eaddhar = self.entry3.get()
        edes = self.entry4.get()
        eadd = self.entry5.get()
        epass = self.entry6.get()

        if ename.strip():
            if valid_phone(econtact):
                if valid_aadhar(eaddhar):
                    if edes:
                        if eadd:
                            if epass:
                                emp_id = random_emp_id(7)
                                insert = (
                                        "INSERT INTO employee(emp_id, name, contact_num, address, aadhar_num,
password, designation) VALUES(?,?,?,?,?,?,?)"
                                    )
                                cur.execute(insert, [emp_id, ename, econtact, eadd, eaddhar, epass, edes])
                                db.commit()
                                messagebox.showinfo("Success!!", "Employee ID: {} successfully added in
database.".format(emp_id), parent=e_add)
                                self.clearr()
                            else:
                                messagebox.showerror("Oops!", "Please enter a password.", parent=e_add)
                        else:
                            messagebox.showerror("Oops!", "Please enter address.", parent=e_add)
                    else:
                        messagebox.showerror("Oops!", "Please enter designation.", parent=e_add)
                else:
                    messagebox.showerror("Oops!", "Invalid Aadhar number.", parent=e_add)
            else:
                messagebox.showerror("Oops!", "Invalid phone number.", parent=e_add)
        else:
            messagebox.showerror("Oops!", "Please enter employee name.", parent=e_add)

    def clearr(self):
        self.entry1.delete(0, END)
        self.entry2.delete(0, END)
        self.entry3.delete(0, END)
        self.entry4.delete(0, END)
        self.entry5.delete(0, END)
        self.entry6.delete(0, END)


class Update_Employee:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
```

```
top.title("Update Employee")

self.label1 = Label(e_update)
self.label1.place(relx=0, rely=0, width=1366, height=768)
self.img = PhotoImage(file="./images/update_employee.png")
self.label1.configure(image=self.img)

self.clock = Label(e_update)
self.clock.place(relx=0.84, rely=0.065, width=102, height=36)
self.clock.configure(font="-family {Poppins Light} -size 12")
self.clock.configure(foreground="#000000")
self.clock.configure(background="#ffffff")

self.r1 = e_update.register(self.testint)
self.r2 = e_update.register(self.testchar)

self.entry1 = Entry(e_update)
self.entry1.place(relx=0.132, rely=0.296, width=374, height=30)
self.entry1.configure(font="-family {Poppins} -size 12")
self.entry1.configure(relief="flat")


self.entry2 = Entry(e_update)
self.entry2.place(relx=0.132, rely=0.413, width=374, height=30)
self.entry2.configure(font="-family {Poppins} -size 12")
self.entry2.configure(relief="flat")
self.entry2.configure(validate="key", validatecommand=(self.r1, '%P'))

self.entry3 = Entry(e_update)
self.entry3.place(relx=0.132, rely=0.529, width=374, height=30)
self.entry3.configure(font="-family {Poppins} -size 12")
self.entry3.configure(relief="flat")
self.entry3.configure(validate="key", validatecommand=(self.r1, '%P'))

self.entry4 = Entry(e_update)
self.entry4.place(relx=0.527, rely=0.296, width=374, height=30)
self.entry4.configure(font="-family {Poppins} -size 12")
self.entry4.configure(relief="flat")
self.entry4.configure(validate="key", validatecommand=(self.r2, '%P'))

self.entry5 = Entry(e_update)
self.entry5.place(relx=0.527, rely=0.413, width=374, height=30)
self.entry5.configure(font="-family {Poppins} -size 12")
self.entry5.configure(relief="flat")

self.entry6 = Entry(e_update)
self.entry6.place(relx=0.527, rely=0.529, width=374, height=30)
self.entry6.configure(font="-family {Poppins} -size 12")
self.entry6.configure(relief="flat")
self.entry6.configure(show="*")
```

```
        self.button1 = Button(e_update)
        self.button1.place(relx=0.408, rely=0.836, width=96, height=34)
        self.button1.configure(relief="flat")
        self.button1.configure(overrelief="flat")
        self.button1.configure(activebackground="#CF1E14")
        self.button1.configure(cursor="hand2")
        self.button1.configure(foreground="#ffffff")
        self.button1.configure(background="#CF1E14")
        self.button1.configure(font="-family {Poppins SemiBold} -size 14")
        self.button1.configure(borderwidth="0")
        self.button1.configure(text="""UPDATE""")
        self.button1.configure(command=self.update)

        self.button2 = Button(e_update)
        self.button2.place(relx=0.526, rely=0.836, width=86, height=34)
        self.button2.configure(relief="flat")
        self.button2.configure(overrelief="flat")
        self.button2.configure(activebackground="#CF1E14")
        self.button2.configure(cursor="hand2")
        self.button2.configure(foreground="#ffffff")
        self.button2.configure(background="#CF1E14")
        self.button2.configure(font="-family {Poppins SemiBold} -size 14")
        self.button2.configure(borderwidth="0")
        self.button2.configure(text="""CLEAR""")
        self.button2.configure(command=self.clearr)

    def update(self):
        ename = self.entry1.get()
        econtact = self.entry2.get()
        eaddhar = self.entry3.get()
        edes = self.entry4.get()
        eadd = self.entry5.get()
        epass = self.entry6.get()

        if ename.strip():
            if valid_phone(econtact):
                if valid_aadhar(eaddhar):
                    if edes:
                        if eadd:
                            if epass:
                                emp_id = vall[0]
                                update = (
                                        "UPDATE employee SET name = ?, contact_num = ?, address = ?, aadhar_num = ?,
password = ?, designation = ? WHERE emp_id = ?"
                                    )
                                cur.execute(update, [ename, econtact, eadd, eaddhar, epass, edes, emp_id])
                                db.commit()
                                messagebox.showinfo("Success!!", "Employee ID: {} successfully updated in
database.".format(emp_id), parent=e_update)
                                vall.clear()
                                page5.tree.delete(*page5.tree.get_children())
                                page5.DisplayData()
```

```
                        Employee.sel.clear()
                        e_update.destroy()
                    else:
                        messagebox.showerror("Oops!", "Please enter a password.", parent=e_add)
                else:
                    messagebox.showerror("Oops!", "Please enter address.", parent=e_add)
            else:
                messagebox.showerror("Oops!", "Please enter designation.", parent=e_add)
        else:
            messagebox.showerror("Oops!", "Invalid Aadhar number.", parent=e_add)
    else:
        messagebox.showerror("Oops!", "Invalid phone number.", parent=e_add)
else:
    messagebox.showerror("Oops!", "Please enter employee name.", parent=e_add)


def clearr(self):
    self.entry1.delete(0, END)
    self.entry2.delete(0, END)
    self.entry3.delete(0, END)
    self.entry4.delete(0, END)
    self.entry5.delete(0, END)
    self.entry6.delete(0, END)



def testint(self, val):
    if val.isdigit():
        return True
    elif val == "":
        return True
    return False

def testchar(self, val):
    if val.isalpha():
        return True
    elif val == "":
        return True
    return False

def time(self):
    string = strftime("%H:%M:%S %p")
    self.clock.config(text=string)
    self.clock.after(1000, self.time)




class Invoice:
    def __init__(self, top=None):
        top.geometry("1366x768")
        top.resizable(0, 0)
```

```
top.title("Invoices")

self.label1 = Label(invoice)
self.label1.place(relx=0, rely=0, width=1366, height=768)
self.img = PhotoImage(file="./images/invoices.png")
self.label1.configure(image=self.img)

self.message = Label(invoice)
self.message.place(relx=0.046, rely=0.055, width=136, height=30)
self.message.configure(font="-family {Poppins} -size 10")
self.message.configure(foreground="#000000")
self.message.configure(background="#ffffff")
self.message.configure(text="""ADMIN""")
self.message.configure(anchor="w")

self.clock = Label(invoice)
self.clock.place(relx=0.9, rely=0.065, width=102, height=36)
self.clock.configure(font="-family {Poppins Light} -size 12")
self.clock.configure(foreground="#000000")
self.clock.configure(background="#ffffff")

self.entry1 = Entry(invoice)
self.entry1.place(relx=0.040, rely=0.286, width=240, height=28)
self.entry1.configure(font="-family {Poppins} -size 12")
self.entry1.configure(relief="flat")

self.button1 = Button(invoice)
self.button1.place(relx=0.229, rely=0.289, width=76, height=23)
self.button1.configure(relief="flat")
self.button1.configure(overrelief="flat")
self.button1.configure(activebackground="#CF1E14")
self.button1.configure(cursor="hand2")
self.button1.configure(foreground="#ffffff")
self.button1.configure(background="#CF1E14")
self.button1.configure(font="-family {Poppins SemiBold} -size 10")
self.button1.configure(borderwidth="0")
self.button1.configure(text="""Search""")
self.button1.configure(command=self.search_inv)

self.button2 = Button(invoice)
self.button2.place(relx=0.035, rely=0.106, width=76, height=23)
self.button2.configure(relief="flat")
self.button2.configure(overrelief="flat")
self.button2.configure(activebackground="#CF1E14")
self.button2.configure(cursor="hand2")
self.button2.configure(foreground="#ffffff")
self.button2.configure(background="#CF1E14")
self.button2.configure(font="-family {Poppins SemiBold} -size 12")
self.button2.configure(borderwidth="0")
self.button2.configure(text="""Logout""")
self.button2.configure(command=self.Logout)
```

```
self.button3 = Button(invoice)
self.button3.place(relx=0.052, rely=0.432, width=306, height=28)
self.button3.configure(relief="flat")
self.button3.configure(overrelief="flat")
self.button3.configure(activebackground="#CF1E14")
self.button3.configure(cursor="hand2")
self.button3.configure(foreground="#ffffff")
self.button3.configure(background="#CF1E14")
self.button3.configure(font="-family {Poppins SemiBold} -size 12")
self.button3.configure(borderwidth="0")
self.button3.configure(text="""DELETE INVOICE""")
self.button3.configure(command=self.delete_invoice)

self.button4 = Button(invoice)
self.button4.place(relx=0.135, rely=0.885, width=76, height=23)
self.button4.configure(relief="flat")
self.button4.configure(overrelief="flat")
self.button4.configure(activebackground="#CF1E14")
self.button4.configure(cursor="hand2")
self.button4.configure(foreground="#ffffff")
self.button4.configure(background="#CF1E14")
self.button4.configure(font="-family {Poppins SemiBold} -size 12")
self.button4.configure(borderwidth="0")
self.button4.configure(text="""EXIT""")
self.button4.configure(command=self.Exit)

self.scrollbarx = Scrollbar(invoice, orient=HORIZONTAL)
self.scrollbary = Scrollbar(invoice, orient=VERTICAL)
self.tree = ttk.Treeview(invoice)
self.tree.place(relx=0.307, rely=0.203, width=880, height=550)
self.tree.configure(
    yscrollcommand=self.scrollbary.set, xscrollcommand=self.scrollbarx.set
)
self.tree.configure(selectmode="extended")

self.tree.bind("<<TreeviewSelect>>", self.on_tree_select)
self.tree.bind("<Double-1>", self.double_tap)

self.scrollbary.configure(command=self.tree.yview)
self.scrollbarx.configure(command=self.tree.xview)

self.scrollbary.place(relx=0.954, rely=0.203, width=22, height=548)
self.scrollbarx.place(relx=0.307, rely=0.924, width=884, height=22)

self.tree.configure(
    columns=(
        "Bill Number",
        "Date",
        "Customer Name",
        "Customer Phone No.",
```

```
        )
    )

    self.tree.heading("Bill Number", text="Bill Number", anchor=W)
    self.tree.heading("Date", text="Date", anchor=W)
    self.tree.heading("Customer Name", text="Customer Name", anchor=W)
    self.tree.heading("Customer Phone No.", text="Customer Phone No.", anchor=W)


    self.tree.column("#0", stretch=NO, minwidth=0, width=0)
    self.tree.column("#1", stretch=NO, minwidth=0, width=219)
    self.tree.column("#2", stretch=NO, minwidth=0, width=219)
    self.tree.column("#3", stretch=NO, minwidth=0, width=219)
    self.tree.column("#4", stretch=NO, minwidth=0, width=219)


    self.DisplayData()


def DisplayData(self):
    cur.execute("SELECT * FROM bill")
    fetch = cur.fetchall()
    for data in fetch:
        self.tree.insert("", "end", values=(data))

sel = []
def on_tree_select(self, Event):
    self.sel.clear()
    for i in self.tree.selection():
        if i not in self.sel:
            self.sel.append(i)

def double_tap(self, Event):
    item = self.tree.identify('item', Event.x, Event.y)
    global bill_num
    bill_num = self.tree.item(item)['values'][0]


    global bill
    bill = Toplevel()
    pg = open_bill(bill)
    #bill.protocol("WM_DELETE_WINDOW", exitt)
    bill.mainloop()




def delete_invoice(self):
    val = []
    to_delete = []
```

```
        if len(self.sel)!=0:
            sure = messagebox.askyesno("Confirm", "Are you sure you want to delete selected invoice(s)?",
parent=invoice)
            if sure == True:
                for i in self.sel:
                    for j in self.tree.item(i)["values"]:
                        val.append(j)

                for j in range(len(val)):
                    if j%5==0:
                        to_delete.append(val[j])

                for k in to_delete:
                    delete = "DELETE FROM bill WHERE bill_no = ?"
                    cur.execute(delete, [k])
                    db.commit()

                messagebox.showinfo("Success!!", "Invoice(s) deleted from database.", parent=invoice)
                self.sel.clear()
                self.tree.delete(*self.tree.get_children())

                self.DisplayData()
        else:
            messagebox.showerror("Error!!","Please select an invoice", parent=invoice)

    def search_inv(self):
        val = []
        for i in self.tree.get_children():
            val.append(i)
            for j in self.tree.item(i)["values"]:
                val.append(j)

        to_search = self.entry1.get()
        for search in val:
            if search==to_search:
                self.tree.selection_set(val[val.index(search)-1])
                self.tree.focus(val[val.index(search)-1])
                messagebox.showinfo("Success!!", "Bill Number: {} found.".format(self.entry1.get()), parent=invoice)
                break
        else:
            messagebox.showerror("Oops!!", "Bill NUmber: {} not found.".format(self.entry1.get()), parent=invoice)


    def Logout(self):
        sure = messagebox.askyesno("Logout", "Are you sure you want to logout?")
        if sure == True:
            invoice.destroy()
            root.deiconify()
            page1.entry1.delete(0, END)
            page1.entry2.delete(0, END)

    def time(self):
```

```
        string = strftime("%H:%M:%S %p")
        self.clock.config(text=string)
        self.clock.after(1000, self.time)

    def Exit(self):
        sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=invoice)
        if sure == True:
            invoice.destroy()
            adm.deiconify()


class open_bill:
    def __init__(self, top=None):

        top.geometry("765x488")
        top.resizable(0, 0)
        top.title("Bill")

        self.label1 = Label(bill)
        self.label1.place(relx=0, rely=0, width=765, height=488)
        self.img = PhotoImage(file="./images/bill.png")
        self.label1.configure(image=self.img)

        self.name_message = Text(bill)
        self.name_message.place(relx=0.178, rely=0.205, width=176, height=30)
        self.name_message.configure(font="-family {Podkova} -size 10")
        self.name_message.configure(borderwidth=0)
        self.name_message.configure(background="#ffffff")

        self.num_message = Text(bill)
        self.num_message.place(relx=0.854, rely=0.205, width=90, height=30)
        self.num_message.configure(font="-family {Podkova} -size 10")
        self.num_message.configure(borderwidth=0)
        self.num_message.configure(background="#ffffff")

        self.bill_message = Text(bill)
        self.bill_message.place(relx=0.150, rely=0.243, width=176, height=26)
        self.bill_message.configure(font="-family {Podkova} -size 10")
        self.bill_message.configure(borderwidth=0)
        self.bill_message.configure(background="#ffffff")

        self.bill_date_message = Text(bill)
        self.bill_date_message.place(relx=0.780, rely=0.243, width=90, height=26)
        self.bill_date_message.configure(font="-family {Podkova} -size 10")
        self.bill_date_message.configure(borderwidth=0)
        self.bill_date_message.configure(background="#ffffff")


        self.Scrolledtext1 = tkst.ScrolledText(top)
        self.Scrolledtext1.place(relx=0.044, rely=0.41, width=695, height=284)
        self.Scrolledtext1.configure(borderwidth=0)
        self.Scrolledtext1.configure(font="-family {Podkova} -size 8")
```

```
        self.Scrolledtext1.configure(state="disabled")

        find_bill = "SELECT * FROM bill WHERE bill_no = ?"
        cur.execute(find_bill, [bill_num])
        results = cur.fetchall()
        if results:
            self.name_message.insert(END, results[0][2])
            self.name_message.configure(state="disabled")

            self.num_message.insert(END, results[0][3])
            self.num_message.configure(state="disabled")

            self.bill_message.insert(END, results[0][0])
            self.bill_message.configure(state="disabled")

            self.bill_date_message.insert(END, results[0][1])
            self.bill_date_message.configure(state="disabled")

            self.Scrolledtext1.configure(state="normal")
            self.Scrolledtext1.insert(END, results[0][4])
            self.Scrolledtext1.configure(state="disabled")

page1 = login_page(root)
root.bind("<Return>", login_page.login)
root.mainloop()
```

8. RESULTS

8.1 Performance Metrics

Inventory Performance is a measure of how effectively and efficiently inventory is used and replenished. The goal of inventory performance metrics is to compare actual on-hand dollars versus forecasted cost of goods sold. Many Lean practitioners claim that inventory performance is the single best indicator of the overall operational performance of a facility.

9. ADVANTAGES & DISADVANTAGES

- Paper-based retail inventory management can take a lot of time and effort. The retail inventory management software can cut short your in-store inventory process cycles through automation. Automation would give you time to focus on other productive business tasks.
- Inventory management is one of the crucial retail processes. Thus, any discrepancy in the inventory control would impact all other operations in your company. The retail inventory software can streamline the inventory processes, which would, in turn, improve the efficiency of your entire business
- Manual inventory control would increase your labor and process costs. The software would not only help you save time, but it would also help you reduce costs. As a result, the profitability of your business would improve. Also, you can invest the excess funds in activities that promote your business growth.
- One of the biggest problems with any computerized system is the potential for a system crash. A corrupt hard drive, power outages and other technical issues can result in the loss of needed data. At the least, businesses are interrupted when they are unable to access data they need. Business owners should back up data regularly to protect against data loss.
- Hackers look for any way to get company or consumer information. An inventory system connected to point-of-sale devices and accounting is a valuable resource to hack into in search of potential financial information or personal details of owners, vendors or clients. Updating firewalls and anti-virus software can mitigate this potential issue.
- When everything is automated, it is easy to forego time-consuming physical inventory audits. They may no longer seem necessary when the computers are doing their work. However, it is important to continue to do regular audits to identify loss such as spoilage or breakage. Audits also help business owners identify potential internal theft and manipulation of the computerized inventory system.

10. CONCLUSION

Inventory management is a very complex but essential part of the supply chain. An effective inventory management system helps to reduce stock-related costs such as warehousing, carrying, and ordering costs. As you have read above, there are different techniques that businesses can utilize to simplify and optimize stock management processes and control systems.

11. FUTURE SCOPE

In summary, successful companies will embrace the challenges of inventory management in the 21st century by levering the technology that is being offered through the Fourth Industrial Revolution. More important, companies will look at inventory as a strategic asset, that when properly deployed will deliver increased value and competitive advantage. Effective collaboration between supply chain partners will take on increased importance. The intensifying risks inherent with global sourcing in combination with a better appreciation of TCO will motivate companies to rethink their global inventory strategies.

12. REFERENCES

- Aggarwal, S.: A review of current inventory theory and its applications. International Journal of Production Research 12, 443–472 (1974)
- Anily, S., Federgruen, A.: One warehouse multiple retailer systems with vehicle routing costs. Management Science 36, 92–114 (1990)
- Beckmann, M.: An inventory model for arbitrary interval and quantity distributions of demand. Management Science 8, 35–57 (1961)
- Hamann, T., Proth, J.: Inventory control of repairable tools with incomplete information. International Journal of Production Economics 31, 543–550 (1993)

GitHub link

https://github.com/IBM-EPBL/IBM-Project-9694-1659068514.git

THANK YOU