# ABSTRACT

The Minimalist Retail Management System Mini Project represents a pioneering effort to revolutionize retail management solutions, particularly targeting the needs of small and medium-sized enterprises (SME’s). In response to the complexities inherent in existing retail management systems, this project adopts a minimalist design philosophy, streamlining functionalities to essential components. The system aims to offer a user-friendly, cost-effective, and adaptable solution for retailers while avoiding unnecessary complexities.

As the retail landscape undergoes dynamic changes, SME’s face unique challenges in adopting and implementing traditional retail management systems. These challenges include steep learning curves, resource-intensive implementations, and feature-rich systems that may exceed operational requirements.

This project introduces a paradigm shift by embracing a minimalist design philosophy to create an intuitive, efficient, and accessible retail management system. The system prioritizes essential functionalities while eliminating unnecessary complexities, catering specifically to the needs of SME’s.

# 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 are highly thankful to our my Course project Faculty **Dr. M. Manickam (Assistant Professor , Department of Networking and Communications )** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HOD Dr. Annapurani Panaiyappan .K** (**Networking and Communication**) 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 proje

**TABLE OF CONTENTS**

**CHAPTER NO CONTENTS PAGE NO**

1. [INTRODUCTION 5](#_TOC_250005)
   1. [Motivation **5**](#_TOC_250004)
   2. [Objective **6**](#_TOC_250003)
   3. [Problem Statement **6**](#_TOC_250002)
   4. [Challenges **7**](#_TOC_250001)
2. [LITERATURE SURVEY 8](#_TOC_250000)
3. REQUIREMENTS 9
4. ARCHITECTURE & 10

DESIGN

1. IMPLEMENTATION 11
2. RESULTS/OUTPUT 62
3. CONCLUSION 66
4. REFERENCES 67

# INTRODUCTION

## Motivation

The motivation behind the creation of MRMS stems from the inherent challenges faced by SMEs in adopting and leveraging traditional retail management systems. As retail environments evolve, the need for accessible, efficient, and tailored solutions becomes increasingly evident. The motivation is rooted in the desire to empower SMEs with a system that aligns with their scale, operational nuances, and growth aspirations.

Evolving Retail Dynamics: The retail industry is undergoing rapid transformations, propelled by advancements in technology, changing consumer behaviors, and the globalization of markets. SMEs, despite being integral to the retail fabric, often grapple with adapting to these dynamics. The motivation for MRMS arises from the recognition that SMEs require agile and adaptive retail management solutions to thrive in this dynamic landscape.

Pioneering Change: Ultimately, the motivation behind MRMS is to pioneer a change in the way SMEs approach retail management. It envisions a future where small retailers can harness the power of technology without being overwhelmed by unnecessary complexities. By providing a minimalist, yet robust solution, MRMS seeks to contribute to the resilience and growth of SMEs in the ever-evolving and competitive retail landscape.

## OBJECTIVE

The primary objective of the MRMS Mini Project is to pioneer a minimalist retail management solution that addresses the specific needs of SMEs. By prioritizing simplicity and functionality, the system aims to streamline essential retail operations—inventory management, sales tracking, and customer relations. This objective is driven by the vision to provide SMEs with a user-friendly, cost-effective, and easily adaptable retail management tool.

## Problem Statement

The retail landscape is marked by an array of challenges for SMEs, including the overwhelming complexity of traditional retail management systems. These challenges manifest in the form of arduous learning curves, resource-intensive implementations, and the inclusion of features that often surpass the practical needs of smaller retailers. The problem statement crystallizes around the imperative to bridge this gap, offering a solution that is tailored to the scale and requirements of SMEs.

## Challenges

Complexity of Existing Systems:

Traditional retail management systems, while comprehensive, often introduce unnecessary complexities for SMEs. Navigating through intricate interfaces and managing an array of features not pertinent to their operations poses a significant challenge.

Implementation Hurdles:

Resource-intensive implementations of retail management systems can be daunting for SMEs with limited resources. The challenge lies in simplifying the implementation process without compromising the system's effectiveness.

Feature Overload:

Existing systems tend to include an abundance of features that surpass the operational demands of SMEs. This challenge emphasizes the need to discern and prioritize functionalities essential for small retailers.

Training Overhead:

Steep learning curves associated with conventional retail management systems result in increased training overhead for SMEs. The challenge is to create a system that minimizes the learning curve, ensuring swift adaptation and operational efficiency.

## LITERATURE SURVEY

* + 1. User-Centric Design in Retail Management: delves into the importance of user-centric design principles in retail management systems, drawing parallels with MRMS. Utilizing user surveys, usability testing, and heuristic evaluations, the research evaluates how a minimalist approach aligns with user expectations and enhances the overall user experience.
    2. Technical Evaluation of Minimalist Retail Management Systems: research takes a technical stance, conducting an in-depth analysis of the architecture and performance of MRMS. Using benchmarking techniques, the study assesses system responsiveness, scalability, and resource efficiency, providing valuable insights into the technical merits and limitations.
    3. Adoption Challenges and Success Factors of Minimalist Retail Management Systems: investigates the challenges and success factors in the adoption of MRMS among SMEs. Through interviews and case studies, the research

explores organizational readiness, change management strategies, and

contextual factors influencing the successful implementation of

minimalist retail systems.

## REQUIREMENTS

### Hardware

**Server:**

* Processor: Multi-core processor (e.g., Intel Core i5 or equivalent).
* RAM: 8 GB or higher for optimal performance.
* Storage: SSD storage for faster data access.

**Client Machines:**

* Processor: Dual-core processor or higher.
* RAM: 4 GB or higher.
* Storage: SSD or HDD storage

### Software

**Operating System:**

* Server: Linux (e.g., Ubuntu Server) or Windows Server.
* Client Machines: Windows, macOS, or Linux.

**Database Management System MySQL:** For storing and managing data efficiently.

**Server-Side Scripting:** Python: For server-side scripting and application logic.

**User Interface:** Tkinter: For the graphical user interface (GUI) in Python

**Development Environment:** Integrated Development Environment (IDE): Such as VSCode or PyCharm for coding efficiency.

## ARCHITECTURE AND DESIGN

**A diagram of a product

Description automatically generated**

* + - 1. **IMPLEMENTATION**

### Admin File

# ==================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()

**EMPLOYEE FILE**

#==================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")

user = StringVar()

passwd = StringVar()

fname = StringVar()

lname = StringVar()

new\_user = StringVar()

new\_passwd = StringVar()

cust\_name = StringVar()

cust\_num = StringVar()

cust\_new\_bill = StringVar()

cust\_search\_bill = StringVar()

bill\_date = StringVar()

with sqlite3.connect("./Database/store.db") as db:

cur = db.cursor()

def random\_bill\_number(stringLength):

lettersAndDigits = string.ascii\_letters.upper() + string.digits

strr=''.join(random.choice(lettersAndDigits) for i in range(stringLength-2))

return ('BB'+strr)

def valid\_phone(phn):

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

return True

return False

def login(Event=None):

global username

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:

messagebox.showinfo("Login Page", "The login is successful")

page1.entry1.delete(0, END)

page1.entry2.delete(0, END)

root.withdraw()

global biller

global page2

biller = Toplevel()

page2 = bill\_window(biller)

page2.time()

biller.protocol("WM\_DELETE\_WINDOW", exitt)

biller.mainloop()

else:

messagebox.showerror("Error", "Incorrect username or password.")

page1.entry2.delete(0, END)

def logout():

sure = messagebox.askyesno("Logout", "Are you sure you want to logout?", parent=biller)

if sure == True:

biller.destroy()

root.deiconify()

page1.entry1.delete(0, END)

page1.entry2.delete(0, END)

class login\_page:

def \_\_init\_\_(self, top=None):

top.geometry("1366x768")

top.resizable(0, 0)

top.title("Retail Manager")

self.label1 = Label(root)

self.label1.place(relx=0, rely=0, width=1366, height=768)

self.img = PhotoImage(file="./images/employee\_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=login)

class Item:

def \_\_init\_\_(self, name, price, qty):

self.product\_name = name

self.price = price

self.qty = qty

class Cart:

def \_\_init\_\_(self):

self.items = []

self.dictionary = {}

def add\_item(self, item):

self.items.append(item)

def remove\_item(self):

self.items.pop()

def remove\_items(self):

self.items.clear()

def total(self):

total = 0.0

for i in self.items:

total += i.price \* i.qty

return total

def isEmpty(self):

if len(self.items)==0:

return True

def allCart(self):

for i in self.items:

if (i.product\_name in self.dictionary):

self.dictionary[i.product\_name] += i.qty

else:

self.dictionary.update({i.product\_name:i.qty})

def exitt():

sure = messagebox.askyesno("Exit","Are you sure you want to exit?", parent=biller)

if sure == True:

biller.destroy()

root.destroy()

class bill\_window:

def \_\_init\_\_(self, top=None):

top.geometry("1366x768")

top.resizable(0, 0)

top.title("Billing System")

self.label = Label(biller)

self.label.place(relx=0, rely=0, width=1366, height=768)

self.img = PhotoImage(file="./images/bill\_window.png")

self.label.configure(image=self.img)

self.message = Label(biller)

self.message.place(relx=0.038, 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=username)

self.message.configure(anchor="w")

self.clock = Label(biller)

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(biller)

self.entry1.place(relx=0.509, rely=0.23, width=240, height=24)

self.entry1.configure(font="-family {Poppins} -size 12")

self.entry1.configure(relief="flat")

self.entry1.configure(textvariable=cust\_name)

self.entry2 = Entry(biller)

self.entry2.place(relx=0.791, rely=0.23, width=240, height=24)

self.entry2.configure(font="-family {Poppins} -size 12")

self.entry2.configure(relief="flat")

self.entry2.configure(textvariable=cust\_num)

self.entry3 = Entry(biller)

self.entry3.place(relx=0.102, rely=0.23, width=240, height=24)

self.entry3.configure(font="-family {Poppins} -size 12")

self.entry3.configure(relief="flat")

self.entry3.configure(textvariable=cust\_search\_bill)

self.button1 = Button(biller)

self.button1.place(relx=0.031, rely=0.104, 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=logout)

self.button2 = Button(biller)

self.button2.place(relx=0.315, rely=0.234, 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="""Search""")

self.button2.configure(command=self.search\_bill)

self.button3 = Button(biller)

self.button3.place(relx=0.048, rely=0.885, width=86, height=25)

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 10")

self.button3.configure(borderwidth="0")

self.button3.configure(text="""Total""")

self.button3.configure(command=self.total\_bill)

self.button4 = Button(biller)

self.button4.place(relx=0.141, rely=0.885, width=84, height=25)

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 10")

self.button4.configure(borderwidth="0")

self.button4.configure(text="""Generate""")

self.button4.configure(command=self.gen\_bill)

self.button5 = Button(biller)

self.button5.place(relx=0.230, rely=0.885, width=86, height=25)

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 10")

self.button5.configure(borderwidth="0")

self.button5.configure(text="""Clear""")

self.button5.configure(command=self.clear\_bill)

self.button6 = Button(biller)

self.button6.place(relx=0.322, rely=0.885, width=86, height=25)

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 10")

self.button6.configure(borderwidth="0")

self.button6.configure(text="""Exit""")

self.button6.configure(command=exitt)

self.button7 = Button(biller)

self.button7.place(relx=0.098, rely=0.734, width=86, height=26)

self.button7.configure(relief="flat")

self.button7.configure(overrelief="flat")

self.button7.configure(activebackground="#CF1E14")

self.button7.configure(cursor="hand2")

self.button7.configure(foreground="#ffffff")

self.button7.configure(background="#CF1E14")

self.button7.configure(font="-family {Poppins SemiBold} -size 10")

self.button7.configure(borderwidth="0")

self.button7.configure(text="""Add To Cart""")

self.button7.configure(command=self.add\_to\_cart)

self.button8 = Button(biller)

self.button8.place(relx=0.274, rely=0.734, width=84, height=26)

self.button8.configure(relief="flat")

self.button8.configure(overrelief="flat")

self.button8.configure(activebackground="#CF1E14")

self.button8.configure(cursor="hand2")

self.button8.configure(foreground="#ffffff")

self.button8.configure(background="#CF1E14")

self.button8.configure(font="-family {Poppins SemiBold} -size 10")

self.button8.configure(borderwidth="0")

self.button8.configure(text="""Clear""")

self.button8.configure(command=self.clear\_selection)

self.button9 = Button(biller)

self.button9.place(relx=0.194, rely=0.734, width=68, height=26)

self.button9.configure(relief="flat")

self.button9.configure(overrelief="flat")

self.button9.configure(activebackground="#CF1E14")

self.button9.configure(cursor="hand2")

self.button9.configure(foreground="#ffffff")

self.button9.configure(background="#CF1E14")

self.button9.configure(font="-family {Poppins SemiBold} -size 10")

self.button9.configure(borderwidth="0")

self.button9.configure(text="""Remove""")

self.button9.configure(command=self.remove\_product)

text\_font = ("Poppins", "8")

self.combo1 = ttk.Combobox(biller)

self.combo1.place(relx=0.035, rely=0.408, width=477, height=26)

find\_category = "SELECT product\_cat FROM raw\_inventory"

cur.execute(find\_category)

result1 = cur.fetchall()

cat = []

for i in range(len(result1)):

if(result1[i][0] not in cat):

cat.append(result1[i][0])

self.combo1.configure(values=cat)

self.combo1.configure(state="readonly")

self.combo1.configure(font="-family {Poppins} -size 8")

self.combo1.option\_add("\*TCombobox\*Listbox.font", text\_font)

self.combo1.option\_add("\*TCombobox\*Listbox.selectBackground", "#D2463E")

self.combo2 = ttk.Combobox(biller)

self.combo2.place(relx=0.035, rely=0.479, width=477, height=26)

self.combo2.configure(font="-family {Poppins} -size 8")

self.combo2.option\_add("\*TCombobox\*Listbox.font", text\_font)

self.combo2.configure(state="disabled")

self.combo3 = ttk.Combobox(biller)

self.combo3.place(relx=0.035, rely=0.551, width=477, height=26)

self.combo3.configure(state="disabled")

self.combo3.configure(font="-family {Poppins} -size 8")

self.combo3.option\_add("\*TCombobox\*Listbox.font", text\_font)

self.entry4 = ttk.Entry(biller)

self.entry4.place(relx=0.035, rely=0.629, width=477, height=26)

self.entry4.configure(font="-family {Poppins} -size 8")

self.entry4.configure(foreground="#000000")

self.entry4.configure(state="disabled")

self.Scrolledtext1 = tkst.ScrolledText(top)

self.Scrolledtext1.place(relx=0.439, rely=0.586, width=695, height=275)

self.Scrolledtext1.configure(borderwidth=0)

self.Scrolledtext1.configure(font="-family {Podkova} -size 8")

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

self.combo1.bind("<<ComboboxSelected>>", self.get\_category)

def get\_category(self, Event):

self.combo2.configure(state="readonly")

self.combo2.set('')

self.combo3.set('')

find\_subcat = "SELECT product\_subcat FROM raw\_inventory WHERE product\_cat = ?"

cur.execute(find\_subcat, [self.combo1.get()])

result2 = cur.fetchall()

subcat = []

for j in range(len(result2)):

if(result2[j][0] not in subcat):

subcat.append(result2[j][0])

self.combo2.configure(values=subcat)

self.combo2.bind("<<ComboboxSelected>>", self.get\_subcat)

self.combo3.configure(state="disabled")

def get\_subcat(self, Event):

self.combo3.configure(state="readonly")

self.combo3.set('')

find\_product = "SELECT product\_name FROM raw\_inventory WHERE product\_cat = ? and product\_subcat = ?"

cur.execute(find\_product, [self.combo1.get(), self.combo2.get()])

result3 = cur.fetchall()

pro = []

for k in range(len(result3)):

pro.append(result3[k][0])

self.combo3.configure(values=pro)

self.combo3.bind("<<ComboboxSelected>>", self.show\_qty)

self.entry4.configure(state="disabled")

def show\_qty(self, Event):

self.entry4.configure(state="normal")

self.qty\_label = Label(biller)

self.qty\_label.place(relx=0.033, rely=0.664, width=82, height=26)

self.qty\_label.configure(font="-family {Poppins} -size 8")

self.qty\_label.configure(anchor="w")

product\_name = self.combo3.get()

find\_qty = "SELECT stock FROM raw\_inventory WHERE product\_name = ?"

cur.execute(find\_qty, [product\_name])

results = cur.fetchone()

self.qty\_label.configure(text="In Stock: {}".format(results[0]))

self.qty\_label.configure(background="#ffffff")

self.qty\_label.configure(foreground="#333333")

cart = Cart()

def add\_to\_cart(self):

self.Scrolledtext1.configure(state="normal")

strr = self.Scrolledtext1.get('1.0', END)

if strr.find('Total')==-1:

product\_name = self.combo3.get()

if(product\_name!=""):

product\_qty = self.entry4.get()

find\_mrp = "SELECT mrp, stock FROM raw\_inventory WHERE product\_name = ?"

cur.execute(find\_mrp, [product\_name])

results = cur.fetchall()

stock = results[0][1]

mrp = results[0][0]

if product\_qty.isdigit()==True:

if (stock-int(product\_qty))>=0:

sp = mrp\*int(product\_qty)

item = Item(product\_name, mrp, int(product\_qty))

self.cart.add\_item(item)

self.Scrolledtext1.configure(state="normal")

bill\_text = "{}\t\t\t\t\t\t{}\t\t\t\t\t {}\n".format(product\_name, product\_qty, sp)

self.Scrolledtext1.insert('insert', bill\_text)

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

else:

messagebox.showerror("Oops!", "Out of stock. Check quantity.", parent=biller)

else:

messagebox.showerror("Oops!", "Invalid quantity.", parent=biller)

else:

messagebox.showerror("Oops!", "Choose a product.", parent=biller)

else:

self.Scrolledtext1.delete('1.0', END)

new\_li = []

li = strr.split("\n")

for i in range(len(li)):

if len(li[i])!=0:

if li[i].find('Total')==-1:

new\_li.append(li[i])

else:

break

for j in range(len(new\_li)-1):

self.Scrolledtext1.insert('insert', new\_li[j])

self.Scrolledtext1.insert('insert','\n')

product\_name = self.combo3.get()

if(product\_name!=""):

product\_qty = self.entry4.get()

find\_mrp = "SELECT mrp, stock, product\_id FROM raw\_inventory WHERE product\_name = ?"

cur.execute(find\_mrp, [product\_name])

results = cur.fetchall()

stock = results[0][1]

mrp = results[0][0]

if product\_qty.isdigit()==True:

if (stock-int(product\_qty))>=0:

sp = results[0][0]\*int(product\_qty)

item = Item(product\_name, mrp, int(product\_qty))

self.cart.add\_item(item)

self.Scrolledtext1.configure(state="normal")

bill\_text = "{}\t\t\t\t\t\t{}\t\t\t\t\t {}\n".format(product\_name, product\_qty, sp)

self.Scrolledtext1.insert('insert', bill\_text)

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

else:

messagebox.showerror("Oops!", "Out of stock. Check quantity.", parent=biller)

else:

messagebox.showerror("Oops!", "Invalid quantity.", parent=biller)

else:

messagebox.showerror("Oops!", "Choose a product.", parent=biller)

def remove\_product(self):

if(self.cart.isEmpty()!=True):

self.Scrolledtext1.configure(state="normal")

strr = self.Scrolledtext1.get('1.0', END)

if strr.find('Total')==-1:

try:

self.cart.remove\_item()

except IndexError:

messagebox.showerror("Oops!", "Cart is empty", parent=biller)

else:

self.Scrolledtext1.configure(state="normal")

get\_all\_bill = (self.Scrolledtext1.get('1.0', END).split("\n"))

new\_string = get\_all\_bill[:len(get\_all\_bill)-3]

self.Scrolledtext1.delete('1.0', END)

for i in range(len(new\_string)):

self.Scrolledtext1.insert('insert', new\_string[i])

self.Scrolledtext1.insert('insert','\n')

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

else:

try:

self.cart.remove\_item()

except IndexError:

messagebox.showerror("Oops!", "Cart is empty", parent=biller)

else:

self.Scrolledtext1.delete('1.0', END)

new\_li = []

li = strr.split("\n")

for i in range(len(li)):

if len(li[i])!=0:

if li[i].find('Total')==-1:

new\_li.append(li[i])

else:

break

new\_li.pop()

for j in range(len(new\_li)-1):

self.Scrolledtext1.insert('insert', new\_li[j])

self.Scrolledtext1.insert('insert','\n')

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

else:

messagebox.showerror("Oops!", "Add a product.", parent=biller)

def wel\_bill(self):

self.name\_message = Text(biller)

self.name\_message.place(relx=0.514, rely=0.452, 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(biller)

self.num\_message.place(relx=0.894, rely=0.452, 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(biller)

self.bill\_message.place(relx=0.499, rely=0.477, 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(biller)

self.bill\_date\_message.place(relx=0.852, rely=0.477, 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")

def total\_bill(self):

if self.cart.isEmpty():

messagebox.showerror("Oops!", "Add a product.", parent=biller)

else:

self.Scrolledtext1.configure(state="normal")

strr = self.Scrolledtext1.get('1.0', END)

if strr.find('Total')==-1:

self.Scrolledtext1.configure(state="normal")

divider = "\n\n\n"+("─"\*61)

self.Scrolledtext1.insert('insert', divider)

total = "\nTotal\t\t\t\t\t\t\t\t\t\t\tRs. {}".format(self.cart.total())

self.Scrolledtext1.insert('insert', total)

divider2 = "\n"+("─"\*61)

self.Scrolledtext1.insert('insert', divider2)

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

else:

return

state = 1

def gen\_bill(self):

if self.state == 1:

strr = self.Scrolledtext1.get('1.0', END)

self.wel\_bill()

if(cust\_name.get()==""):

messagebox.showerror("Oops!", "Please enter a name.", parent=biller)

elif(cust\_num.get()==""):

messagebox.showerror("Oops!", "Please enter a number.", parent=biller)

elif valid\_phone(cust\_num.get())==False:

messagebox.showerror("Oops!", "Please enter a valid number.", parent=biller)

elif(self.cart.isEmpty()):

messagebox.showerror("Oops!", "Cart is empty.", parent=biller)

else:

if strr.find('Total')==-1:

self.total\_bill()

self.gen\_bill()

else:

self.name\_message.insert(END, cust\_name.get())

self.name\_message.configure(state="disabled")

self.num\_message.insert(END, cust\_num.get())

self.num\_message.configure(state="disabled")

cust\_new\_bill.set(random\_bill\_number(8))

self.bill\_message.insert(END, cust\_new\_bill.get())

self.bill\_message.configure(state="disabled")

bill\_date.set(str(date.today()))

self.bill\_date\_message.insert(END, bill\_date.get())

self.bill\_date\_message.configure(state="disabled")

with sqlite3.connect("./Database/store.db") as db:

cur = db.cursor()

insert = (

"INSERT INTO bill(bill\_no, date, customer\_name, customer\_no, bill\_details) VALUES(?,?,?,?,?)"

)

cur.execute(insert, [cust\_new\_bill.get(), bill\_date.get(), cust\_name.get(), cust\_num.get(), self.Scrolledtext1.get('1.0', END)])

db.commit()

#print(self.cart.items)

print(self.cart.allCart())

for name, qty in self.cart.dictionary.items():

update\_qty = "UPDATE raw\_inventory SET stock = stock - ? WHERE product\_name = ?"

cur.execute(update\_qty, [qty, name])

db.commit()

messagebox.showinfo("Success!!", "Bill Generated", parent=biller)

self.entry1.configure(state="disabled", disabledbackground="#ffffff", disabledforeground="#000000")

self.entry2.configure(state="disabled", disabledbackground="#ffffff", disabledforeground="#000000")

self.state = 0

else:

return

def clear\_bill(self):

self.wel\_bill()

self.entry1.configure(state="normal")

self.entry2.configure(state="normal")

self.entry1.delete(0, END)

self.entry2.delete(0, END)

self.entry3.delete(0, END)

self.name\_message.configure(state="normal")

self.num\_message.configure(state="normal")

self.bill\_message.configure(state="normal")

self.bill\_date\_message.configure(state="normal")

self.Scrolledtext1.configure(state="normal")

self.name\_message.delete(1.0, END)

self.num\_message.delete(1.0, END)

self.bill\_message.delete(1.0, END)

self.bill\_date\_message.delete(1.0, END)

self.Scrolledtext1.delete(1.0, END)

self.name\_message.configure(state="disabled")

self.num\_message.configure(state="disabled")

self.bill\_message.configure(state="disabled")

self.bill\_date\_message.configure(state="disabled")

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

self.cart.remove\_items()

self.state = 1

def clear\_selection(self):

self.entry4.delete(0, END)

self.combo1.configure(state="normal")

self.combo2.configure(state="normal")

self.combo3.configure(state="normal")

self.combo1.delete(0, END)

self.combo2.delete(0, END)

self.combo3.delete(0, END)

self.combo2.configure(state="disabled")

self.combo3.configure(state="disabled")

self.entry4.configure(state="disabled")

try:

self.qty\_label.configure(foreground="#ffffff")

except AttributeError:

pass

def search\_bill(self):

find\_bill = "SELECT \* FROM bill WHERE bill\_no = ?"

cur.execute(find\_bill, [cust\_search\_bill.get().rstrip()])

results = cur.fetchall()

if results:

self.clear\_bill()

self.wel\_bill()

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")

self.entry1.configure(state="disabled", disabledbackground="#ffffff", disabledforeground="#000000")

self.entry2.configure(state="disabled", disabledbackground="#ffffff", disabledforeground="#000000")

self.state = 0

else:

messagebox.showerror("Error!!", "Bill not found.", parent=biller)

self.entry3.delete(0, END)

def time(self):

string = strftime("%H:%M:%S %p")

self.clock.config(text=string)

self.clock.after(1000, self.time)

page1 = login\_page(root)

root.bind("<Return>", login)

root.mainloop()

**MAIN FILE**

\_\_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)

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")

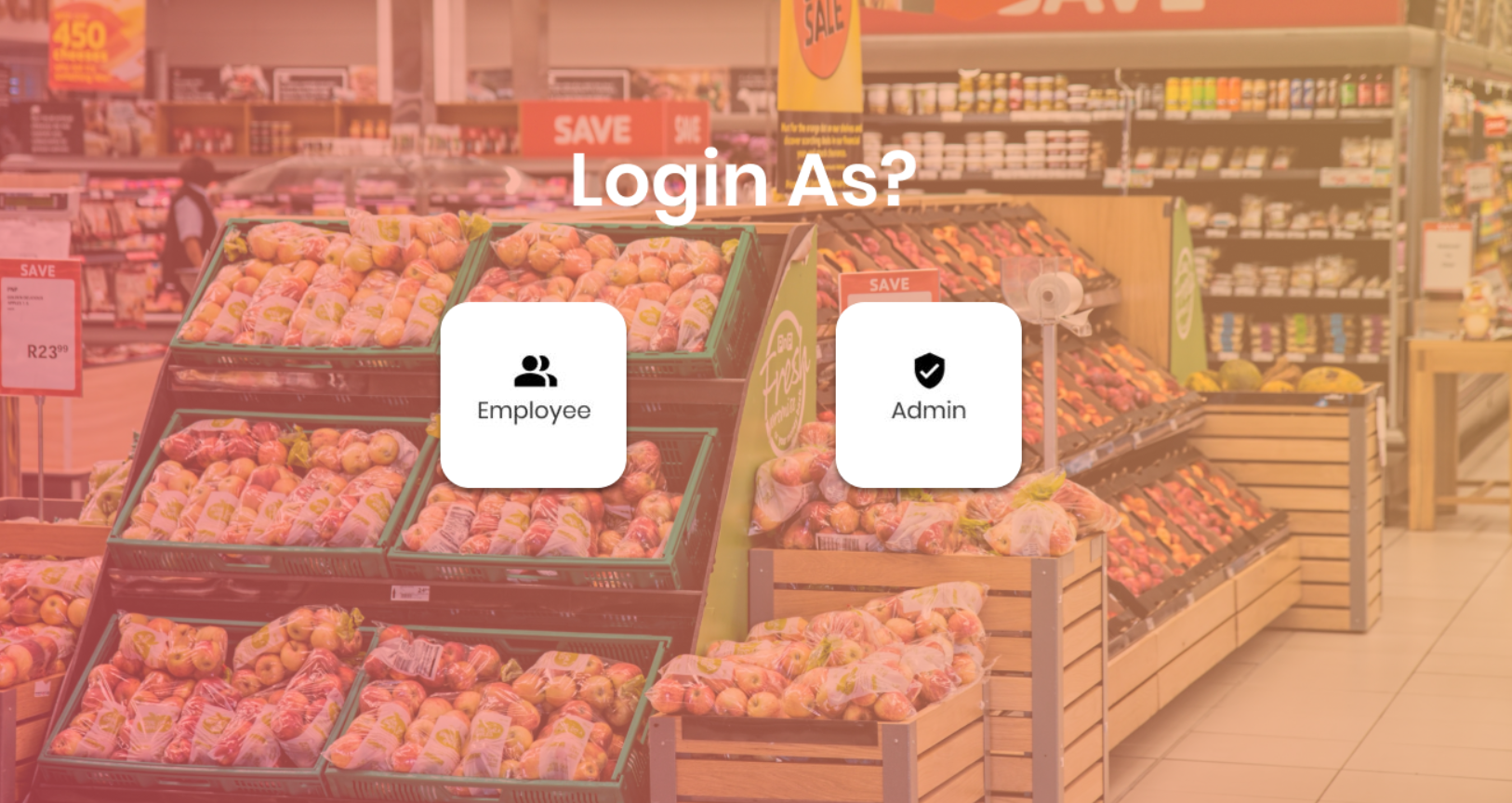
img3 = PhotoImage(file="./images/2.png")

button2.configure(image=img3)

button2.configure(command=adm)

main.mainloop()

* + - 1. **RESULTS**

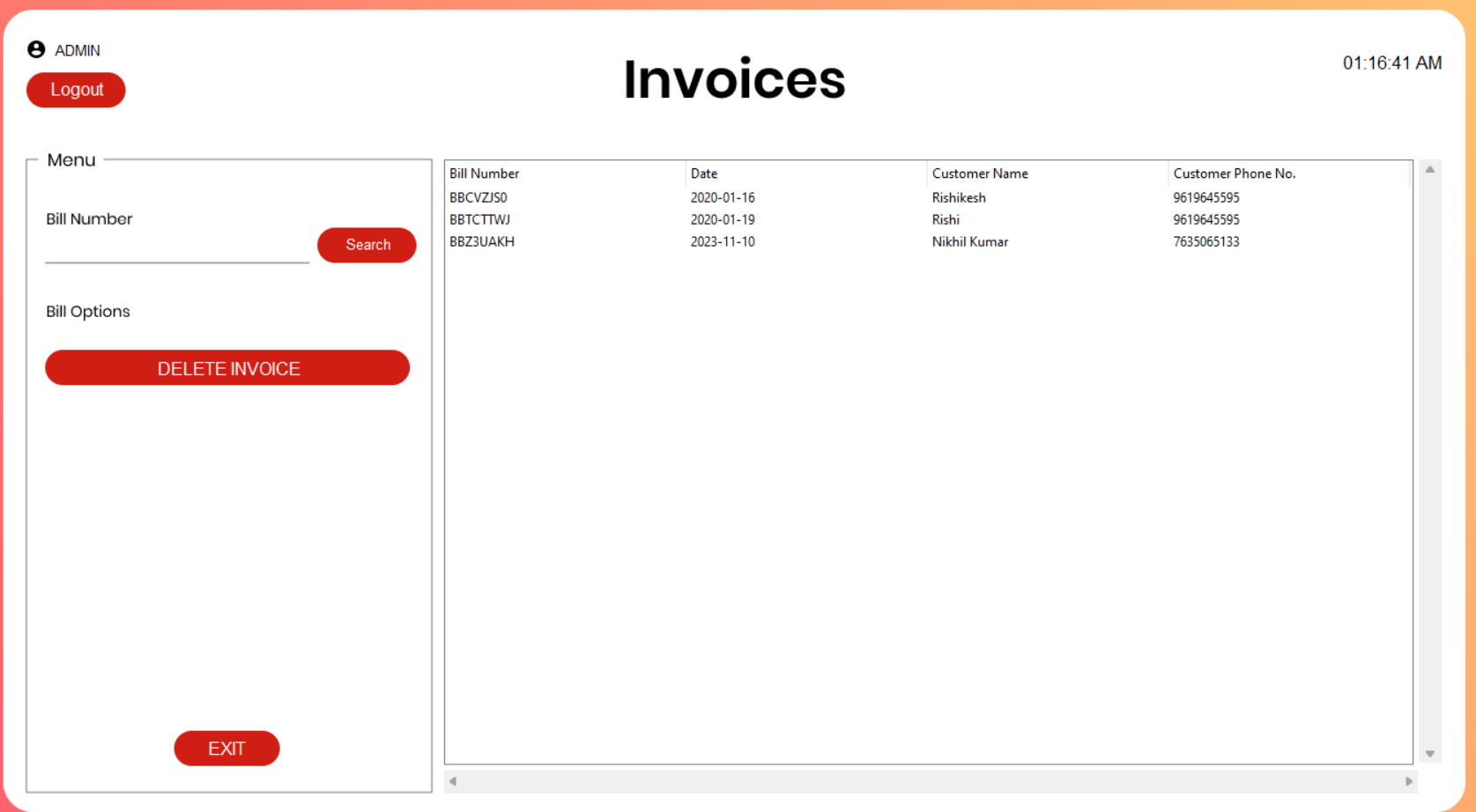
**A screenshot of a login screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a inventory management

Description automatically generatedA screenshot of a computer

Description automatically generated**

* + - 1. **CONCLUSION**

The development of the Minimalist Retail Management System (MRMS) has been a journey aimed at redefining the landscape of retail management, particularly for small and medium-sized enterprises (SMEs). The project's foundation was rooted in the acknowledgment of the challenges faced by SMEs in adopting traditional retail management systems, and the conclusion marks the culmination of efforts to address these challenges through a minimalist approach.

In conclusion, the minimalist design has proven to be a pivotal aspect of the project's success. By distilling functionalities to their essence, MRMS provides SMEs with an intuitive and accessible retail management solution. The streamlined functionality ensures that users can navigate the system with ease, focusing on essential tasks without being encumbered by unnecessary features.

The Minimalist Retail Management System stands as a testament to the transformative power of minimalist design in the realm of retail technology. The conclusion invites reflection on the impact of MRMS in revolutionizing retail management for SMEs and underscores the project's commitment to fostering a more inclusive, efficient, and adaptive future for small retailers in the competitive and ever-evolving retail landscape.

## REFERENCES

* <https://www.geeksforgeeks.org/python-programming-language/>
* <https://www.tutorialspoint.com/python/index.htm>
* <https://www.ieee.org/>
* <https://www.studocu.com/in/document/bharati-vidyapeeth-university/bachler-of-computer-application/tkinter-brief-intro-about-thinter/42719190>
* <https://www.w3schools.com/sql/>