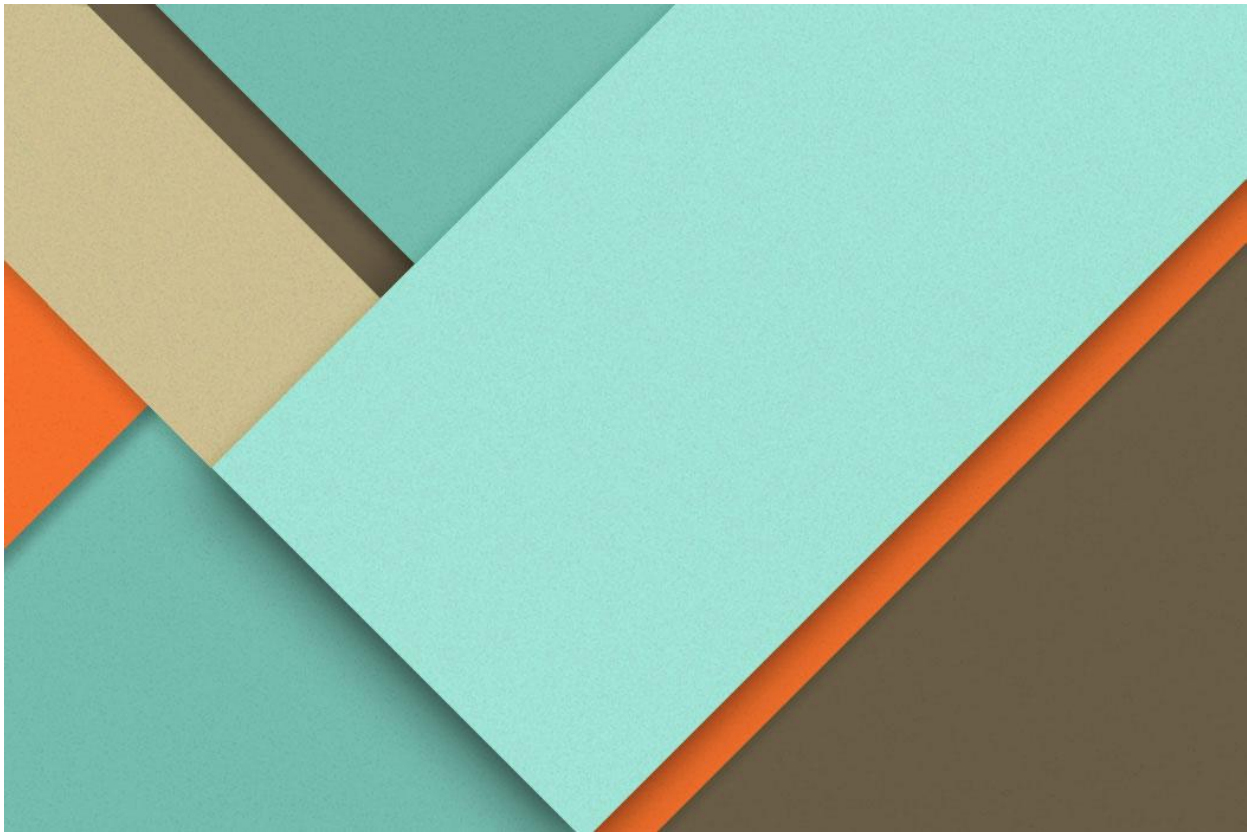

# Retail Management System

SMALL BAZAAR

—

Akshat Kumar Singh
12th M3
( 2022-23 )
Sant Atulanand Residential Academy

## Overview

A python-based retail management system with a sqlite3 database is a handy tool for small business owners or employees responsible for managing the day-to-day operations of a retail store. This system can help streamline and automate various tasks, such as tracking inventory, processing sales transactions, and generating receipts.

It comes with a sleek and modern fluid design for optimal user integration.

## Goals

1. Streamlining and automating various retail management tasks.
2. Improving the accuracy and reliability of data.
3. Providing a secure and convenient way to store and access data.
4. Enhancing customer experience.

## System specifications

Feel free to use it anywhere / anytime / anyhow... ;)

Any operating system - { cross platform }
Can work on less than 200Kb of memory

Python 3+ [ with sqlite3 ]

## Milestones

### I. Admin

> Control and manage employees.

> Add or remove products.

> Check past invoices.

### II. Employee

> Generate bills

> Add or remove items to the cart.

> Search through past bills.

## Features

I. Aesthetic and minimal UI
II. Lovely background
III. Separate login screens for admin and employee
IV. Kills the previous screen upon loading the new one
V. Uses less memory
VI. Easily generate bills with a unique bill number
VII. Updates stock quantity in real-time

## Limitations

- No printer support
- Not much safe for commercial use

## Contributing

Overcome limitations

# CODE

## main.py

```
__author__ = "Akshat"

import os

from tkinter import *

from tkinter import messagebox

main = Tk()

main.geometry("1366x768")

main.title("Small 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()
```

## Admin.py

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", "Logged in successfully.")
                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)
```

INVENTORY-

```
def inventory():

    adm.withdraw()

    global inv

    global page3

    inv = Toplevel()

    page3 = Inventory(inv)

    page3.time()

    inv.protocol("WM_DELETE_WINDOW", exitt)

    inv.mainloop()
```

EMPLOYEE-

```
def employee():

    adm.withdraw()

    global emp

    global page5

    emp = Toplevel()

    page5 = Employee(emp)

    page5.time()

    emp.protocol("WM_DELETE_WINDOW", exitt)

    emp.mainloop()
```

INVOICES-

```
def invoices():

    adm.withdraw()

    global invoice

    invoice = Toplevel()

    page7 = Invoice(invoice)
    page7.time()

    invoice.protocol("WM_DELETE_WINDOW", exitt)

    invoice.mainloop()
```

ADD PRODUCT -

```
def add_product(self):

        global p_add

        global page4

        p_add = Toplevel()

        page4 = add_product(p_add)

        page4.time()

        p_add.mainloop()
```

TIME -

```
def time(self):

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

        self.clock.config(text=string)

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

## Employee.py

BILL NUMBER

```
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)
```

PHONE NUMBER -

```
def valid_phone(phn):

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

        return True

    return False
```

ADD TO 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)
```

REMOVE PRODUCT -

```
    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)
```

SEARCH BILL -

```
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)
```

NOTE : the above given/shown code(s) are parts of the main segment of various files

## BIBLIOGRAPHY -

- Wikipedia
- Pypi
- GPT-3
- Computer Science with python (by Sumita Arora)

**SPECIAL THANKS TO- SATYAJEET & SUNIL sir for their kind guidance which made this project possible.**

OUR TEAM -

- Akshat Kumar Singh
- Ashit Mishra
- Ashutosh Yadav