Sales Management System

A COURSE PROJECT REPORT

By

MD Nizamuddin – RA2111030010192

Under the guidance of

Ms. M. Safa

In partial fulfilment for the Course

of

18CSC207J - Advanced Programming Practice

in NWC

FACULTY OF ENGINEERING AND TECHNOLOGY,

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,

Kattankulathur, Chengalpattu District

APRIL 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report
" **Sales Management System**"
is the bonafide work of
MD Nizamuddin – RA2111030010192
Mohommad Adnan - RA2111030010229
Shreyansh Tiwari - RA2111030010249
who carried out the project work under my supervision.

SIGNATURE

Ms. M. Safa
Assistant Professor
NWC
SRM Institute of Science and Technology

TABLE OF CONTENTS

CHAPTERS **CONTENTS**

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honourable **Vice Chancellor**
Dr. C. Muthamizhchelvan, for being the beacon in all our endeavours.
We would like to express my warmth of gratitude to our **Registrar**
Dr. S. Ponnusamy, for his encouragement.
We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.
We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project
We wish to express my sincere thanks to **Course Audit Professor**
Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to my Course project Faculty
Ms. M. Safa, Assistant Professor, NWC, 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 NWC** and our Departmental colleagues for their Support.

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

1. ABSTRACT

- The Sales Management System is a software application designed to manage the sales activities of a company.

- This system is developed using the Tkinter and SQLite3 libraries of Python.

- The system provides a user-friendly interface to manage the sales and inventory of a company.

- The system allows the user to add new products, manage the stock, and generate reports based on the sales data.

- The application is secure and provides data protection with a login system

2. INTRODUCTION

1. The Sales Management System is designed to help small and medium-sized businesses manage their sales activities.

2. The application is developed in Python using the Tkinter and SQLite3 libraries.

3. The system has a user-friendly interface that allows the user to add new products, manage the stock, and generate reports.

4. The application is designed to be secure and protect the data with a login system.

5. The system has several features, including:

- Adding new products to the inventory
- Managing the stock of existing products
- Generating reports based on the sales data
- Tracking the sales of individual products
- Secure login system

6. The system is developed using the Tkinter library of Python, which provides a set of tools for creating graphical user interfaces.

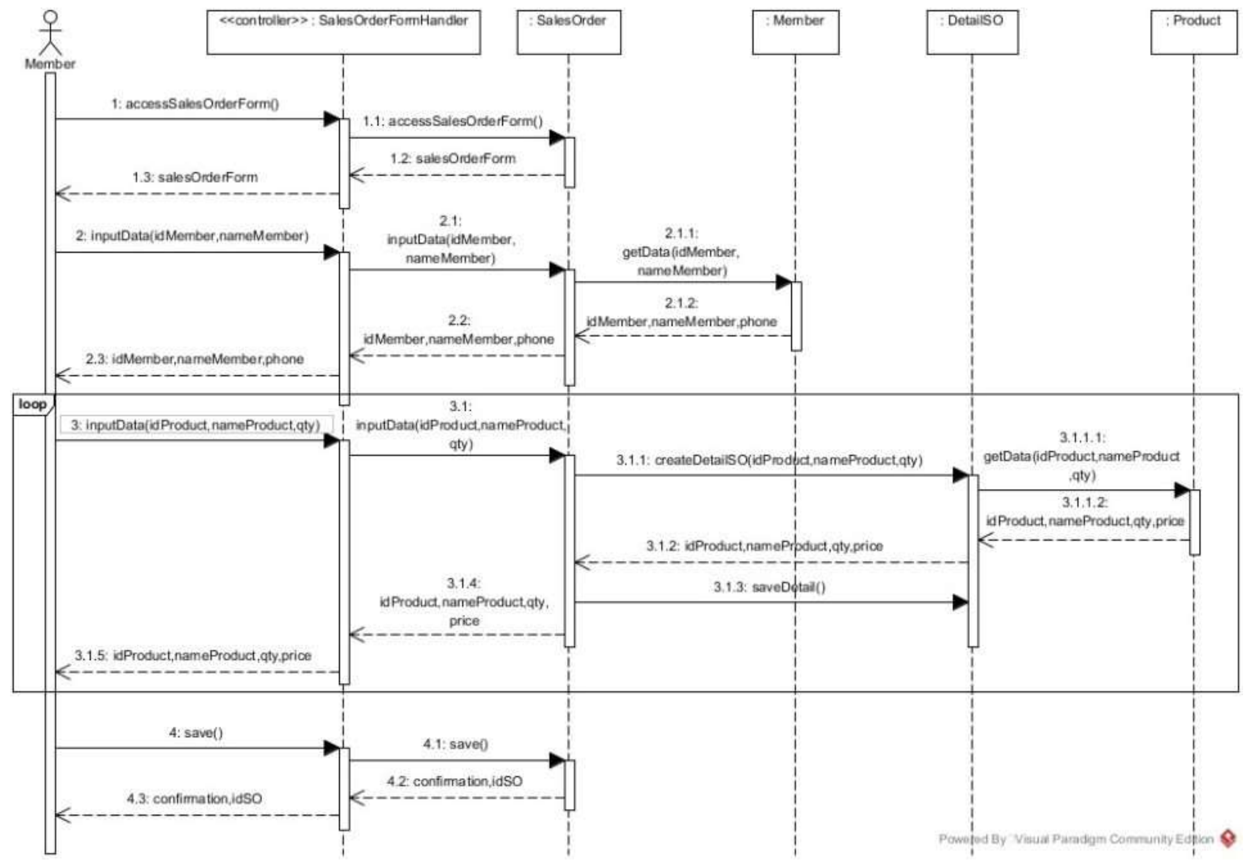
7. The SQLite3 library is used to store the data in a database. The application can run on any platform that supports Python and SQLite3.

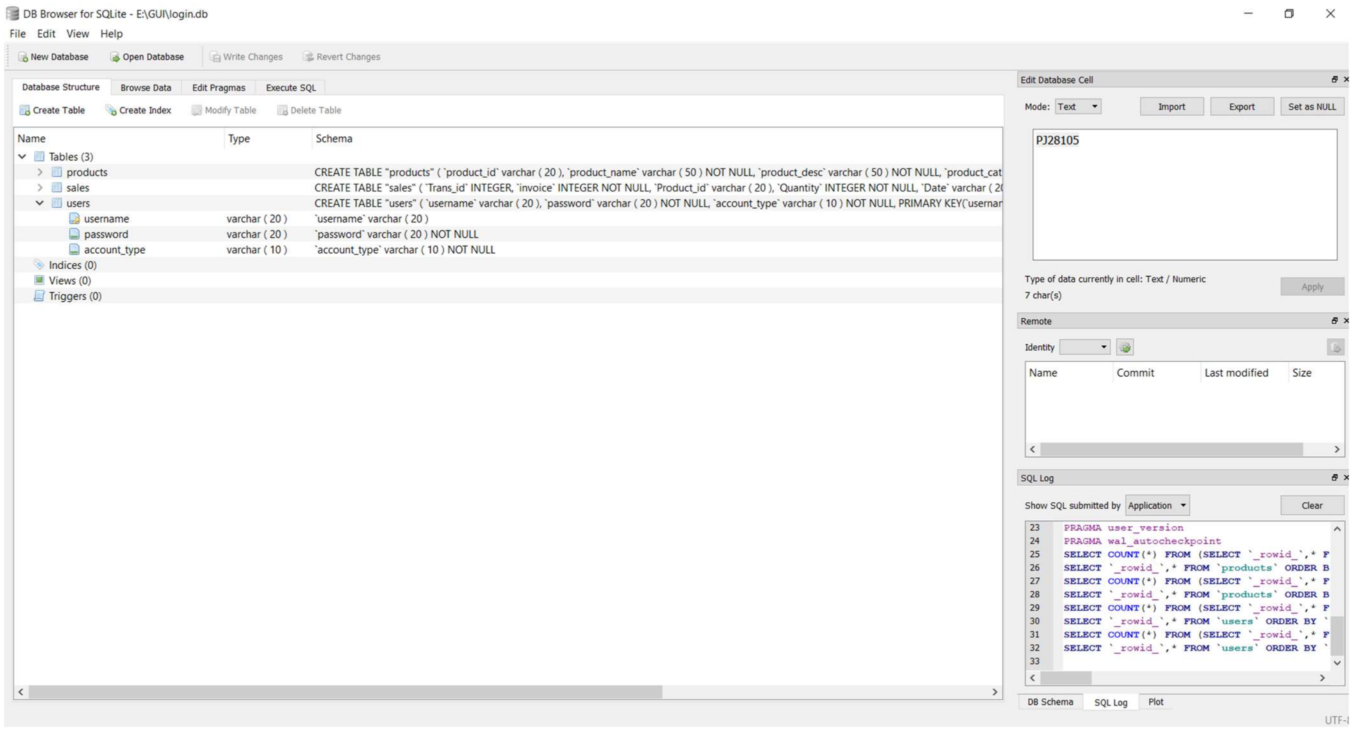
3.Requirement Analysis

1. **User Interface**: The system must have a user-friendly interface that allows users to easily navigate the application. The interface should be intuitive, and users should be able to perform tasks such as adding new products, managing the stock, and generating reports without any difficulty.
2. **Product Management**: The system should allow users to add new products to the inventory. The user should be able to add details about the product, including the name, description, price, and stock quantity.
3. **Stock Management**: The system should allow users to manage the stock of existing products. The user should be able to update the stock quantity, add new stock, or remove existing stock.
4. **Sales Management**: The system should allow users to track the sales of individual products. The system should be able to generate reports based on the sales data, including sales by product, sales by date, and sales by location.
5. **Data Security**: The system should be secure and protect the data with a login system. The user should be required to enter a username and password to access the system.
6. **Platform Compatibility**: The system should be able to run on any platform that supports Python and SQLite3. The system should be compatible with Windows, Linux, and macOS.
7. **Integration with Payment Gateways**: The system should have the ability to integrate with payment gateways to process online payments. This feature would allow customers to make purchases directly from the system.
8. **Export Data**: The system should have the ability to export data to common formats like CSV and Excel. This feature would allow users to analyze the sales data outside of the system.

4. ARCHITECTURE AND DESIGN

1. The architecture of the Sales Management System follows the **Model-View-Controller (MVC) pattern**. This pattern separates the application into three interconnected components: the model, the view, and the controller.
2. The model represents the **data and business logic** of the application. In the case of the Sales Management System, the model will be implemented in a separate file that will handle the database connection and manipulation. The model will interact with the database and provide data to the controller and view.
3. The view represents the user interface of the application. In the Sales Management System, there will be a view: **admin_menu.py** which will display the different functionalities available to the admin and other employees, respectively. The view will receive input from the user and send it to the controller.
4. The **controller acts as an intermediary** between the model and the view. It receives input from the view, processes it using the model, and sends the result back to the view. In the Sales Management System, the controller will be implemented in the main.py file, which will initialize the database connection and manage the login process.
5. The **main.py** file will receive input from the userlogin.py file, which will validate the **user's credentials** and redirect them to the appropriate view.
6. The additional features, such as integration with payment gateways data export functionalities, will be implemented in the **dditional_features.py** file. This file will use third-party libraries and APIs to add these functionalities to the system.
7. Overall, the MVC architecture of the Sales Management System separates the different aspects of the application, making it modular and **easy to maintain**. The model handles the data and business logic, the view represents the user interface, and the controller manages the flow of information between the model and the view.


Member
<<controller>> : SalesOrderFormHandler
: SalesOrder
: Member
: DetailSO
: Product
1: accessSalesOrderForm()
1.1: accessSalesOrderForm()
1.2: salesOrderForm
1.3: salesOrderForm
2: inputData(idMember,nameMember)
2.1: inputData(idMember, nameMember)
2.1.1: getData(idMember, nameMember)
2.1.2: idMember,nameMember,phone
2.2: idMember,nameMember,phone
2.3: idMember,nameMember,phone
loop
3: inputData(idProduct,nameProduct,qty)
3.1: inputData(idProduct,nameProduct, qty)
3.1.1: createDetailSO(idProduct,nameProduct,qty)
3.1.1.1: getData(idProduct,nameProduct ,qty)
3.1.1.2: idProduct, nameProduct,qty,price
3.1.2: idProduct,nameProduct,qty,price
3.1.3: saveDetail()
3.1.4: idProduct,nameProduct,qty, price
3.1.5: idProduct,nameProduct,qty,price
4: save()
4.1: save()
4.2: confirmation,idSO
4.3: confirmation,idSO
Powered By Visual Paradigm Community Edition


DB Browser for SQLite - E:\GUI\login.db

File Edit View Help

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Modify Table | Delete Table

Name	Type	Schema
Tables (3)		
products		CREATE TABLE "products" (`product_id` varchar (20), `product_name` varchar (50) NOT NULL, `product_desc` varchar (50) NOT NULL, `product_cat
sales		CREATE TABLE "sales" (`Trans_id` INTEGER, `invoice` INTEGER NOT NULL, `Product_id` varchar (20), `Quantity` INTEGER NOT NULL, `Date` varchar (2(
users		CREATE TABLE "users" (`username` varchar (20), `password` varchar (20) NOT NULL, `account_type` varchar (10) NOT NULL, PRIMARY KEY(`usernar
username	varchar (20)	`username` varchar (20)
password	varchar (20)	`password` varchar (20) NOT NULL
account_type	varchar (10)	`account_type` varchar (10) NOT NULL
Indices (0)		
Views (0)		
Triggers (0)		

Edit Database Cell

Mode: Text | Import | Export | Set as NULL

PJ28105

Type of data currently in cell: Text / Numeric
7 char(s)

Apply

Remote

Identity

Name	Commit	Last modified	Size

SQL Log

Show SQL submitted by Application | Clear

```
PRAGMA user_version
PRAGMA wal_autocheckpoint
SELECT COUNT(*) FROM (SELECT `_rowid_`,* F
SELECT `_rowid_`,* FROM `products` ORDER B
SELECT COUNT(*) FROM (SELECT `_rowid_`,* F
SELECT `_rowid_`,* FROM `products` ORDER B
SELECT COUNT(*) FROM (SELECT `_rowid_`,* F
SELECT `_rowid_`,* FROM `users` ORDER BY `
SELECT COUNT(*) FROM (SELECT `_rowid_`,* F
SELECT `_rowid_`,* FROM `users` ORDER BY `
```

DB Schema | SQL Log | Plot

UTF-

5. IMPLEMENTATION

1. The main file for the application will be called **main.py**. This file will be responsible for initializing the database connection and starting the login process. The system will use SQLite3 for storing data, and the database file will be named **login.db**

2. The **admin_menu.py** file will contain the code for the admin menu. The admin menu will be available to users with admin privileges, and it will allow them to manage products, manage stock, generate reports, and perform other tasks.

3. The **user_menu.py** file will contain the code for the user menu. The user menu will be available to other employees, and it will allow them to perform tasks like adding new products, updating stock levels, and generating reports.

4. The **userlogin.py** file will contain the code for the login screen. This screen will be the first screen that the user sees when they start the application. The user will need to enter their username and password to access the application.

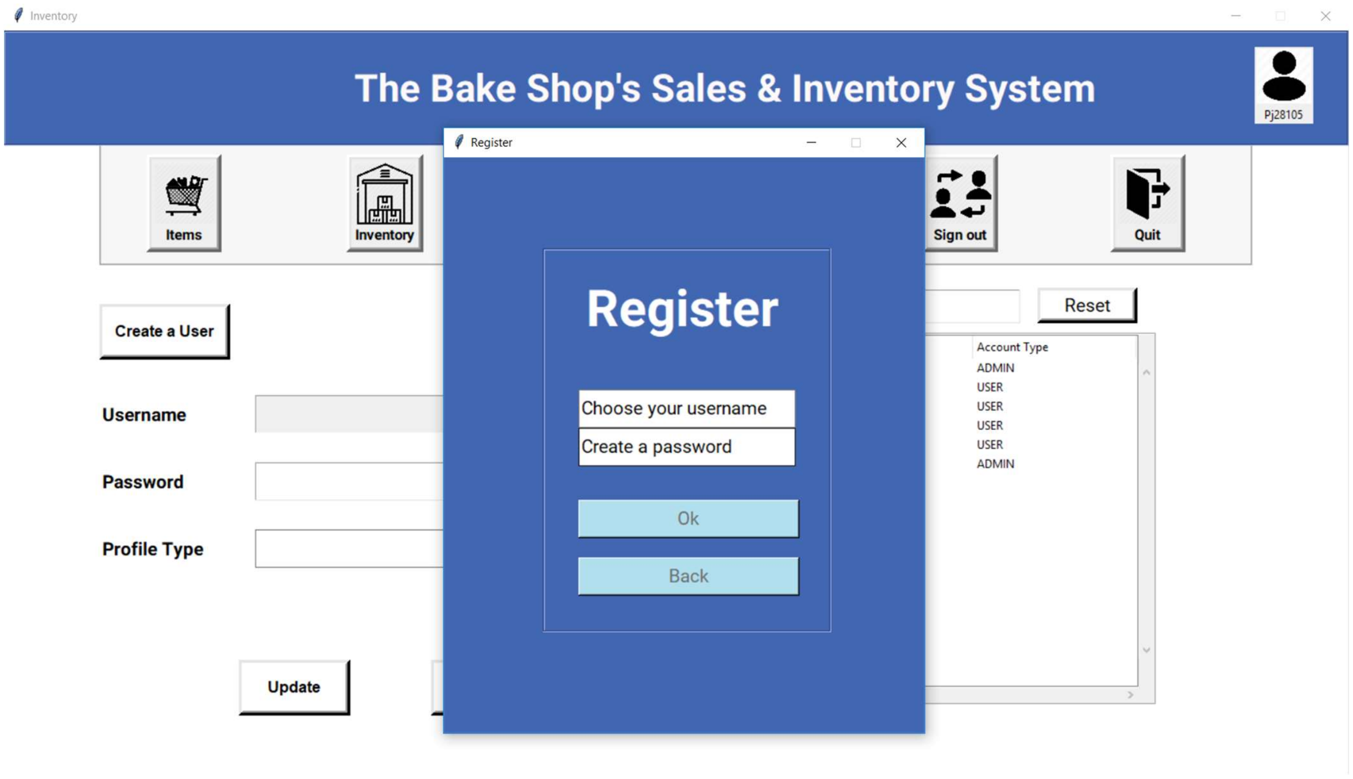
5. The **additional_features.py** file will contain the code for additional features like integration with payment gateways and shipping carriers, as well as the ability to export data to common formats like CSV and Excel.

6. CODE

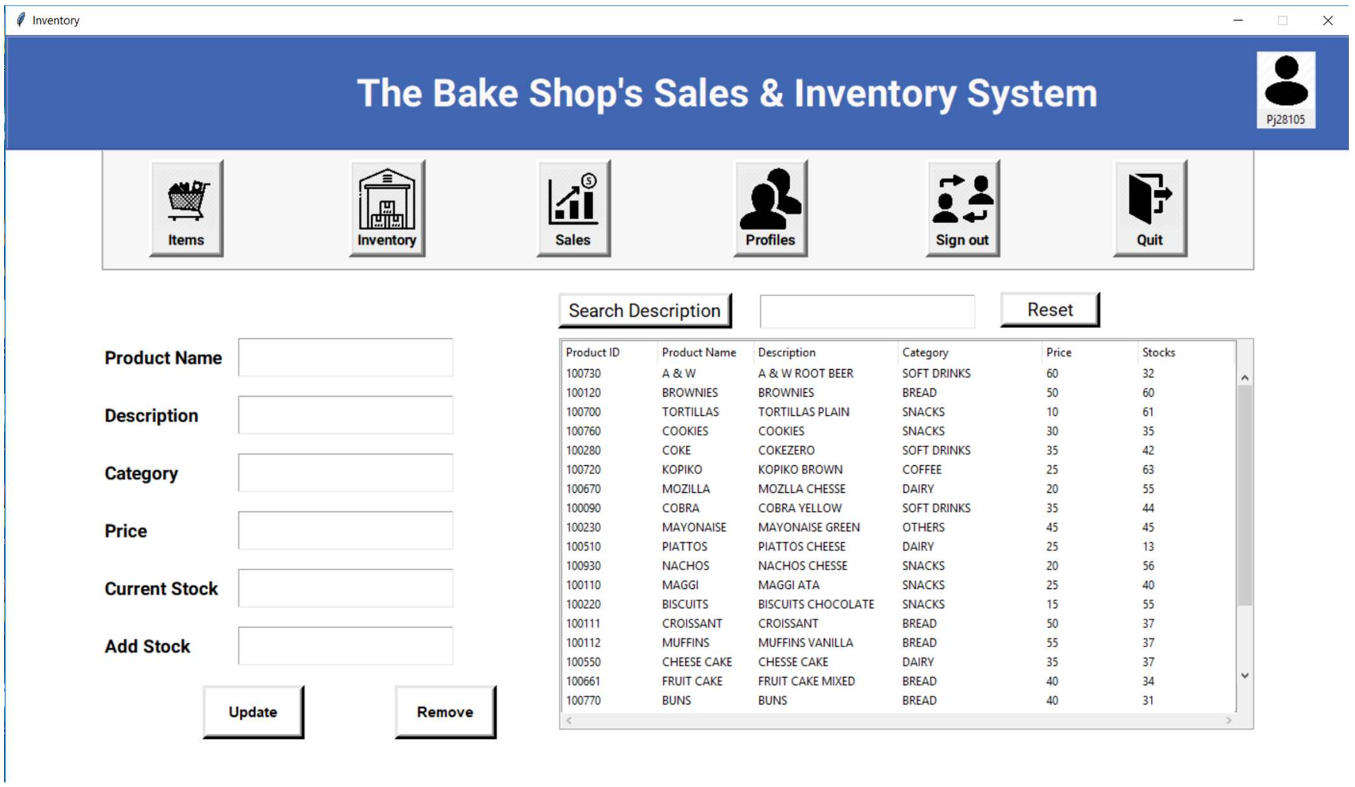
#main.py file only

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

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

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

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

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

    #  ADD WIDGETS TO TOP OF MAIN WINDOW
    def buildmain(self):
        if self.account_type == 'ADMIN':
            super(Admin).__init__()
            self.admin_mainmenu(8,8)
        else:
            super(User).__init__()
            self.user_mainmenu(8,8)
        self.logout.config(command=self.__Main_del__)
        self.changeuser.config(command=self.change_user)

self.topframe=LabelFrame(self.mainw,width=1400,height=120,bg="#4267b2")
        self.topframe.place(x=0,y=0)
        self.store_name = 'The Bake Shop'
        self.storelable=Label(self.topframe,text=self.store_name + "'s 
Sales & Inventory System",bg="#4267b2",anchor="center")
        self.storelable.config(font="Roboto 30 bold",fg="snow")
        self.storelable.place(x=360,y=30)
        mi = PhotoImage(file="images/myprofile.png")
        mi = mi.subsample(4,4)
        self.myprofile = 
ttk.Label(self.topframe,text=(self.username.get()).capitalize(),image=m
i, compound=TOP)
        self.myprofile.image = mi
        self.myprofile.place(x=1300,y=15)
        ''''
        if self.account_type == 'ADMIN':
            self.adminlabel=
Label(self.topframe,text="Admin",font="Roboto 10 bold",bg="#4267b2")
        else:
            self.adminlabel = Label(self.topframe, text=" User",
font="Roboto 10 bold", bg="#4267b2")
        self.adminlabel.place(x=1300,y=80)
        '''
        # DATE TIME LABEL
        ''''
        now = datetime.datetime.now()
        self.datetimelabel=
Label(self.topframe,text=str(now.day)+'/'+str(now.month)+'/'+str(now.ye
ar),font="Roboto 10 bold", bg="skyblue2")
        self.datetimelabel.place(x=1290,y=90)
        '''

    # METHODS FOR ITEMS AND CHANGE USER BUTTONS
    def change_user(self):
        if messagebox.askyesno("Alert!", "Do  you want to change
user?") == True:
            self.base.commit()
            self.mainw.destroy()
            self.loginw.destroy()
            self.__init__()


if __name__ == '__main__':
    w = Main()
    w.base.commit()
    w.mainw.mainloop()_time}")
```

7. RESULT & OUTPUTS



Account Type
ADMIN
USER
USER
USER
USER
ADMIN



Product ID	Product Name	Description	Category	Price	Stocks
100730	A & W	A & W ROOT BEER	SOFT DRINKS	60	32
100120	BROWNIES	BROWNIES	BREAD	50	60
100700	TORTILLAS	TORTILLAS PLAIN	SNACKS	10	61
100760	COOKIES	COOKIES	SNACKS	30	35
100280	COKE	COKEZERO	SOFT DRINKS	35	42
100720	KOPIKO	KOPIKO BROWN	COFFEE	25	63
100670	MOZILLA	MOZLLA CHESSE	DAIRY	20	55
100090	COBRA	COBRA YELLOW	SOFT DRINKS	35	44
100230	MAYONAISE	MAYONAISE GREEN	OTHERS	45	45
100510	PIATTOS	PIATTOS CHEESE	DAIRY	25	13
100930	NACHOS	NACHOS CHEESE	SNACKS	20	56
100110	MAGGI	MAGGI ATA	SNACKS	25	40
100220	BISCUITS	BISCUITS CHOCOLATE	SNACKS	15	55
100111	CROISSANT	CROISSANT	BREAD	50	37
100112	MUFFINS	MUFFINS VANILLA	BREAD	55	37
100550	CHEESE CAKE	CHESSE CAKE	DAIRY	35	37
100661	FRUIT CAKE	FRUIT CAKE MIXED	BREAD	40	34
100770	BUNS	BUNS	BREAD	40	31

8. CONCLUSION & FUTURE ENHANCEMENT

Conclusion:

The Sales Management System is a useful tool for managing sales activities in small and medium-sized businesses. The system is user-friendly and provides several features to manage the sales and inventory of a company. The system is secure and protects the data with a login system.

Future Enhancements:

1. Integration with online marketplaces to manage online sales
2. Integration with payment gateways to process online payments
3. Integration with shipping carriers to manage shipping activities
4. Ability to export data to common formats like CSV, Excel

9. REFERENCES

1. **Python Documentation - https://www.python.org/**

2. **Sales example - https://www.zoho.com/crm/sales-management-system.html**