VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018.

FILE STRUCTURE MINI PROJECT REPORT

ON

"ONLINE SHOPPING MANAGMENT SYSTEM"

Submitted in partial fulfilment for the requirement of VI Semester for the

Degree of Bachelor of Engineering in

INFORMATION SCIENCE & ENGINEERING

For the Academic Year 2022-23

SUBMITTED BY

VIKAS S H[1DB20IS163]

VISHAL C HALKODU[1DB20IS166]

Under the Guidance of

Mrs. DEEPIKA A B

Assistant Professor

Dept. of ISE

Department of Information Science & Engineering

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysuru Road, Bengaluru – 560074

2022-23

DON BOSCO INSTITUTE OF TECHNOLOGY
Kumbalagodu, Mysuru Road, Bengaluru – 560074

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project Report entitled **"ONLINE SHOPPING MANAGMENT SYSTEM"** is a bonafide Mini Project work carried out by **Vikas S H (1DB20IS163), Vishal C Halkodu(1DB20IS166)** in partial fulfilment of VI semester for the Degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi, during the Academic Year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessments have been incorporated with the degree mentioned.

Signature of Guide	Signature of HoD
______________________	______________________
Mrs. DEEPIKA A B	**Dr. B K RAGHAVENDRA**
Assistant Professor	Professor & Head
Dept. of ISE,	Dept. of ISE,
DBIT, Bengaluru.	DBIT, Bengaluru

External Viva

Name of the Examiners	**Signature with Date**
1. ________________	________________
2. ________________	________________

DON BOSCO INSTITUTE OF TECHNOLOGY Kumbalagodu, Bengaluru -560074

DECLARATION

We **Vikas S H** and **Vishal C Halkodu ,** students of 6th Semester B.E, Information Science and Engineering at **Don Bosco Institute of Technology, Bengaluru** hereby declare that the project work entitled **“Online Shopping Managment System**” has been carried out by us as a part of the course work 18ISL67 – File Structures Laboratory with mini project under the supervision of **Mrs. DEEPIKA A B,** Assistant Professor, Department of Information Science and Engineering, Don Bosco Institute of Technology ,Bengaluru during the academic year 2022-2023. We further declare that the report has not been submitted to any other university for the award of any other degree

Place: Bengaluru

Date:

ACKNOWLEDGEMENT

At the various stages in making the mini project, a number of people have given me invaluable comment on the manuscript. We take this opportunity to express my deepest gratitude and appreciation to all those who helped me directly or indirectly towards the successful completion of this project.

We would like to thank our **Dr. B. S. NAGABHUSHANA, Principal Don Bosco Institute of Technology, Bengaluru** for his support throughout this project.

We express my whole hearted gratitude to **Dr. B. K. RAGHAVENDRA,** who is our respectable **Professor and Head of Department of Information Science and Engineering**. We wish to acknowledge for his valuable help and encouragement.

With this regard We owe a heartfelt gratitude to our guide **Mrs. DEEPIKA A B, Assistant Professor, Department of Information Science and Engineering**, for timely advice on the Mini Project and regular assistance throughout the project work.

We would also like to thank the teaching and non-teaching staff members **of Department of Information Science and Engineering** for their corporation.

VIKAS S H (1DB20IS163)

VISHAL C HALKODU(1DB20IS166)

ABSTRACT

The main aim and objective was to plan and program system application and to get rid of manual entry and to store it in a file so that easily available. We have to apply the best software engineering practice for system application. We developed "Online Shopping System" project, this is a small online shopping application that showcases the concepts of hashing. The application has been designed to handle any number of users and includes several features to ensure a smooth and secure shopping experience. Users are required to verify their username and password before accessing the system. Once logged in, users can search for products and add them to their cart for purchase. The application allows users to maintain a history of their orders and remove items from their cart if needed. Additionally, the application includes an automatic billing feature that enables users to securely make online payments using their preferred payment method. The system has been designed with a check on input data to ensure that all user data is valid and secure. Overall, this small online shopping application provides users with a user-friendly interface for purchasing products online while ensuring that their personal and financial information is protected.

TABLE OF CONTENTS

LISTS OF FIGURES

CHAPTER 1

INTRODUCTION

Online shopping has become an essential part of our daily lives. With the growth of the internet and e-commerce, consumers can now purchase products online from anywhere in the world. To keep up with this trend, businesses must provide their customers with a reliable and efficient online shopping experience. This mini-project aims to develop an online shopping management system that provides customers with a user-friendly interface to purchase products from an online store. The system will include features such as user registration, product search, product details, shopping cart, order tracking, and payment gateway integration. The online shopping management system will be designed to be secure, reliable, and efficient. Users will be required to register before they can access the system, and their personal and financial information will be encrypted to ensure their privacy and security. The system will also include a payment gateway integration to enable users to make secure online payments using their preferred payment method. Overall, this mini-project aims to develop a robust and efficient online shopping management system that provides customers with a convenient and secure way to purchase products online, while also providing businesses with an efficient platform to manage their online sales..

1.1 Overview of the file structure

File Structure is the organization of Data in secondary storage device in such a way that minimize the access time and the storage space. A File structure is a combination of representations for Data in files and of operations for accessing the data. A File structure allows applications to read, write and modify Data.

1.2 Problem statement

In today's fast-paced digital era, there is a growing need for an efficient and user-friendly online shopping management system that can seamlessly handle the entire shopping process, from product selection and inventory management to secure payment processing and timely order fulfillment. The system should address challenges such as improving customer experience by providing personalized recommendations, ensuring accurate product information and availability, streamlining logistics and delivery operations, safeguarding sensitive customer data, and facilitating effective communication between buyers, sellers, and customer support. The ultimate goal is to create a robust online shopping platform that enhances convenience, trust, and satisfaction for both consumers and retailers in a rapidly evolving e-commerce landscape.

CHAPTER 2

STUDY OF AN EXISITING SYSTEM

2.1Existing system

The existing system of online shopping management faces several challenges and shortcomings that hinder its overall efficiency and user experience. Firstly, there is a lack of centralized and synchronized inventory management, resulting in inaccurate product availability information and potential order fulfillment delays. Additionally, the current system often fails to provide personalized recommendations based on user preferences and browsing history, limiting the potential for cross-selling and upselling. Payment processing can be cumbersome and prone to security risks, leading to customer concerns about data privacy. Moreover, the communication channels between buyers, sellers, and customer support are often disjointed, resulting in delayed responses and unsatisfactory resolutions to customer queries or issues. Lastly, the existing system struggles to seamlessly integrate logistics and delivery operations, leading to uncertainties in tracking and delayed shipments. Overall, there is a pressing need to address these limitations and develop an upgraded online shopping management system that can enhance efficiency, security, personalization, and communication, ultimately providing a superior shopping experience for customers.

2.2Disadvantages

- Lack of centralization: The existing system lacks a centralized system for users to maintain a record of their orders, making it difficult to keep track of their purchase history.
- Time-consuming: As users have to visit multiple websites to purchase different products, the process can be time-consuming.
- Limited options: As users are limited to individual online stores, they may have limited options for purchasing products.
- Security risks: There may be security risks involved when users make purchases directly from the store's website, as their personal and financial information may be compromised.

2.3Proposed System

The proposed system for the mini project on online shopping management system would include a user-friendly interface that enables customers to browse products, view details, and add items to their cart for purchase. The system would also include a payment gateway that enables customers to securely make online payments using their preferred payment method. Additionally, the system would allow customers to track their orders and view their purchase history. The system would require users to log in with their credentials before they can access the system. The system administrator would be able to manage product categories, view customer orders, and manage user accounts. The system would also include features such as product search and sorting, reviews and ratings, and order confirmation emails.

Back End Design

In our project we use python as our backed language. All file handling and data retrieval and updates are done using python. Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file.

Front End Design

In this we have explained about the frontend design tools such as PyCharm along with front end language Python.

PyCharm IDE**:** PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCS es), and supports web development with Django as well as data science with Anaconda.

Qt-Designer**:** Qt Designer is a visual tool for designing and building graphical user interfaces (GUIs) from Qt components. It allows you to design and build widgets and dialogs using on-screen forms usingthe same widgets that will be used in your application. Components created with Qt Designer can also take advantage of Qt's signals and slots, and they can be previewed so that you can ensure that they willlook and feel exactly as you intended.

Python: Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear.

2.4 Advantages

• Convenience: Customers can shop for products from the comfort of their homes or offices, eliminating the need to visit physical stores

. • Accessibility: The system is available 24/7, allowing customers to shop at any time of the day or night.

• Secure Transactions: The payment gateway ensures that all transactions are secure and protected from fraud.

• Order Tracking: Customers can track their orders and get updates on delivery status, ensuring that they are always informed about their purchases.

• Efficient Management: The system administrator can efficiently manage product categories, view customer orders, and manage user accounts, ensuring that the system operates smoothly.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

Programming Language: Python

Software used: Visual Studio,PyCharm

3.2 HARDWARE REQUIREMENTS

RAM: 4 GB RAM or above

Processor: Intel Core

Operating System: Windows

CHAPTER 4

IMPLEMENTATION

4.1 File creation

ADD - This option allows the admin to add product to the inventory with its name, price and quantity.

Function:

```
def add_product(self):
        add_product_window = tk.Toplevel(self)
        add_product_window.title("Add Product")
        add_product_window.geometry("300x200")

        name_label = tk.Label(add_product_window, text="Product Name:")
        name_label.pack(padx=5, pady=5)
        name_entry = tk.Entry(add_product_window)
        name_entry.pack(padx=5, pady=5)

        price_label = tk.Label(add_product_window, text="Product Price:")
        price_label.pack(padx=5, pady=5)
        price_entry = tk.Entry(add_product_window)
        price_entry.pack(padx=5, pady=5)

        quantity_label = tk.Label(add_product_window, text="Product Quantity:")
        quantity_label.pack(padx=5, pady=5)
        quantity_entry = tk.Entry(add_product_window)
        quantity_entry.pack(padx=5, pady=5)

        add_button = tk.Button(add_product_window, text="Add", command=lambda:
self.add_product_confirm(
            name_entry.get(), float(price_entry.get()), int(quantity_entry.get()), add_product_window))
        add_button.pack(padx=5, pady=5)
```

SEARCH – This option enables user to search for specific products details by entering the product name.

Function:

```
def search_product(self):
        search_product_window = tk.Toplevel(self)
        search_product_window.title("Search Product")
        search_product_window.geometry("300x100")
```

```
 name_label = tk.Label(search_product_window, text="Product Name to Search:")
        name_label.pack(padx=5, pady=5)
        name_entry = tk.Entry(search_product_window)
        name_entry.pack(padx=5, pady=5)

        search_button = tk.Button(search_product_window, text="Search", command=lambda:
self.search_product_confirm(
            name_entry.get(), search_product_window))
        search_button.pack(padx=5, pady=5)
```

DELETE – Here delete function allows us to delete or remove the account of customeronce after deleting the account it deletes the entire details of customer.

FUNCTION:

```
def remove_product(self):
        remove_product_window = tk.Toplevel(self)
        remove_product_window.title("Remove Product")
        remove_product_window.geometry("300x100")

        name_label = tk.Label(remove_product_window, text="Product Name to Remove:")
        name_label.pack(padx=5, pady=5)
        name_entry = tk.Entry(remove_product_window)
        name_entry.pack(padx=5, pady=5)

        remove_button = tk.Button(remove_product_window, text="Remove", command=lambda:
self.remove_product_confirm(
            name_entry.get(), remove_product_window))
        remove_button.pack(padx=5, pady=5)
```

4.2 Description of methods

INDEXING:

It permits a user to input some pieces of information(such as the name of an individual)and see all other information about the relevant file, such as the case number or the date.

All indexes are based on same basic concepts-keys and reference fields. The types of indexes we see are called simple indexes because they are represented using simple arrays of structures that contain the keys and reference fields.

A Simple Index for Entry-Sequenced Files-Simple indexing can be useful when the entire index can be held in memory.

- Changes (additions and deletions) require both the index and the data file to be changed.
- Updates affect the index if the key field is changed, or if the record is moved.
- An update which moves a record can be handled as the deletion followed by an addition.

FUNCTION:

```
# Create the index file.
index_file_path = "index.txt"
if not os.path.isfile(index_file_path):
    with open(index_file_path, 'w') as index_file:
        index_file.write("ProductID\tName\n")
with open(index_file_path, 'a') as index_file:
    index_file.write(f"{getpid}\t{name}\n")
```

HASHING:

Hashing is the process of indexing and retrieving the data items in a data structure to provide faster way of finding the elements using the hash function. The hashed values are kept in a data structure known as hash tables.

Hashed File-The file that uses the hashing algorithm for distributing records in one or more groups on disks

FUNCTION:

```
getpid = random.randint(0, 10000)
        hashno = hashlib.md5(str(name).encode()).hexdigest()
        product = [getpid, name, price, quantity,hashno]
        #add
        with open('shop.csv', 'a+', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(product)
        messagebox.showinfo("Success", "Product added successfully!")
        window.destroy()
```

4.3 Code

```
import hashlib
import tkinter as tk
from tkinter import *
from PIL import Image,ImageTk
from tkinter import messagebox
from tkinter import ttk
import csv
import random

# Frontend & Backend implementation
class Inventory:
    def __init__(self):
        pass
class LoginPage(tk.Frame):
    def __init__(self, parent, inventory):
        super().__init__(parent)
        self.inventory = inventory
        self.parent = parent
        self.username_var = tk.StringVar()
        self.password_var = tk.StringVar()
        self.create_widgets()

    def create_widgets(self):

        username_label = tk.Label(self, text="Username:")
        username_label.grid(row=0, column=0, padx=5, pady=5)
        username_entry = tk.Entry(self, textvariable=self.username_var)
        username_entry.grid(row=0, column=1, padx=5, pady=5)

        password_label = tk.Label(self, text="Password:")
        password_label.grid(row=1, column=0, padx=5, pady=5)
        password_entry = tk.Entry(self, textvariable=self.password_var, show="*")
        password_entry.grid(row=1, column=1, padx=5, pady=5)

        login_button = tk.Button(self, text="Login", command=self.login)
        login_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

    def login(self):
        username = self.username_var.get()
        password = self.password_var.get()
        # Add your authentication logic here
        if username == "admin" and password == "password":
            self.parent.show_front_page()
        else:
            messagebox.showerror("Login Failed", "Invalid username or password.")
```

```
class FrontPage(tk.Frame):
    def __init__(self, parent, inventory):
        super().__init__(parent)
        self.inventory = inventory
        self.parent = parent
        self.create_widgets()

    def create_widgets(self):
        add_button = tk.Button(self, text="Add product", command=self.add_product)
        add_button.pack(padx=5, pady=5)

        remove_button = tk.Button(self, text="Remove product", command=self.remove_product)
        remove_button.pack(padx=5, pady=5)

        search_button = tk.Button(self, text="Search product", command=self.search_product)
        search_button.pack(padx=5, pady=5)

        list_button = tk.Button(self, text="List all products", command=self.list_products)
        list_button.pack(padx=5, pady=5)

        exit_button = tk.Button(self, text="Exit", command=self.parent.quit)
        exit_button.pack(padx=5, pady=5)

    def add_product(self):
        add_product_window = tk.Toplevel(self)
        add_product_window.title("Add Product")
        add_product_window.geometry("300x200")

        name_label = tk.Label(add_product_window, text="Product Name:")
        name_label.pack(padx=5, pady=5)
        name_entry = tk.Entry(add_product_window)
        name_entry.pack(padx=5, pady=5)

        price_label = tk.Label(add_product_window, text="Product Price:")
        price_label.pack(padx=5, pady=5)
        price_entry = tk.Entry(add_product_window)
        price_entry.pack(padx=5, pady=5)

        quantity_label = tk.Label(add_product_window, text="Product Quantity:")
        quantity_label.pack(padx=5, pady=5)
        quantity_entry = tk.Entry(add_product_window)
        quantity_entry.pack(padx=5, pady=5)

        add_button = tk.Button(add_product_window, text="Add", command=lambda: self.add_product_confirm(
            name_entry.get(), float(price_entry.get()), int(quantity_entry.get()), add_product_window))
        add_button.pack(padx=5, pady=5)
```

```
def add_product_confirm(self, name, price, quantity, window):
    #check
    flag = 0
    with open('shop.csv', 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            for element in row:
                if element == name:
                    flag = 1
    if flag == 1:
        def update_newlist(j):
            with open('shop.csv', 'w', newline='') as file:
                writer = csv.writer(file)
                writer.writerows(j)

        new_list = []

        with open('shop.csv', 'r') as file:
            reader = csv.reader(file)
            for row in reader:
                new_list.append(row)
                for element in row:
                    if element == name:
                        data = [row[0], name, price, quantity,row[4]]
                        index = new_list.index(row)
                        new_list[index] = data

        update_newlist(new_list)
        messagebox.showinfo("Success", "Product updated successfully!")
        window.destroy()
    else:
        getpid = random.randint(0, 10000)
        hashno = hashlib.md5(str(name).encode()).hexdigest()
        product = [getpid, name, price, quantity,hashno]
        #add
        with open('shop.csv', 'a+', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(product)
        messagebox.showinfo("Success", "Product added successfully!")
        window.destroy()

  def remove_product(self):
    remove_product_window = tk.Toplevel(self)
    remove_product_window.title("Remove Product")
    remove_product_window.geometry("300x100")

    name_label = tk.Label(remove_product_window, text="Product Name to Remove:")
    name_label.pack(padx=5, pady=5)
    name_entry = tk.Entry(remove_product_window)
```

```
 name_entry.pack(padx=5, pady=5)

    remove_button = tk.Button(remove_product_window, text="Remove", command=lambda: self.remove_product_confirm(
        name_entry.get(), remove_product_window))
    remove_button.pack(padx=5, pady=5)

  def remove_product_confirm(self, name, window):
    #self.inventory.remove_product(name)
    def save(j):
        with open('shop.csv', 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(j)

    new_list = []
    with open('shop.csv', 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            new_list.append(row)

            for element in row:
                if element == name:
                    new_list.remove(row)
    save(new_list)
    messagebox.showinfo("Success", "Product removed successfully!")
    window.destroy()

  def search_product(self):
    search_product_window = tk.Toplevel(self)
    search_product_window.title("Search Product")
    search_product_window.geometry("300x100")

    name_label = tk.Label(search_product_window, text="Product Name to Search:")
    name_label.pack(padx=5, pady=5)
    name_entry = tk.Entry(search_product_window)
    name_entry.pack(padx=5, pady=5)

    search_button = tk.Button(search_product_window, text="Search", command=lambda: self.search_product_confirm(
        name_entry.get(), search_product_window))
    search_button.pack(padx=5, pady=5)

  def search_product_confirm(self, name, window):
    data = []
    with open('shop.csv', 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            for element in row:
                if (element == name):
                    data.append(row)
    if data == []:
            messagebox.showinfo("Error", "No Data Found!")
            window.destroy()
        else:
            product_list_window = tk.Toplevel(self)
            product_list_window.title("Product List")
            product_list_window.geometry("400x400")

            product_list_label = tk.Label(product_list_window, text="List of Products:")
            product_list_label.pack(padx=5, pady=5)

            product_list_frame = Frame(product_list_window)
            product_list_frame.place(x=0, y=50, width=400, height=350)

            scroll_x = ttk.Scrollbar(product_list_frame, orient=HORIZONTAL)
            scroll_y = ttk.Scrollbar(product_list_frame, orient=VERTICAL)

            product_list = ttk.Treeview(product_list_frame, column=("ProductID","Name", "Price",
"Quantity","HashCode"),
                                        xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
            scroll_x.pack(side=BOTTOM, fill=X)
            scroll_y.pack(side=RIGHT, fill=Y)

            scroll_x.config(command=product_list.xview)
            scroll_y.config(command=product_list.yview)

            product_list.heading("ProductID", text="ProductID")
            product_list.heading("Name", text="Name")
            product_list.heading("Price", text="Price")
            product_list.heading("Quantity", text="Quantity")
            product_list.heading("HashCode", text="HashCode")

            product_list["show"] = "headings"

            product_list.column("ProductID", width=100)
            product_list.column("Name", width=100)
            product_list.column("Price", width=100)
            product_list.column("Quantity", width=100)
            product_list.column("HashCode", width=150)

            product_list.pack(fill=BOTH, side=TOP)
            product_list.delete(*product_list.get_children())
            for i in data:
                product_list.insert("", END, values=i)
            window.destroy()

    def list_products(self):
        product_list_window = tk.Toplevel(self)
        product_list_window.title("Product List")
        product_list_window.geometry("400x400")
```

```
     product_list_label = tk.Label(product_list_window, text="List of Products:")
        product_list_label.pack(padx=5, pady=5)

        product_list_frame =Frame(product_list_window)
        product_list_frame.place(x=0,y=50,width=400,height=350)

        scroll_x = ttk.Scrollbar(product_list_frame, orient=HORIZONTAL)
        scroll_y = ttk.Scrollbar(product_list_frame, orient=VERTICAL)

        product_list                                                                                   =
ttk.Treeview(product_list_frame,column=("ProductID","Name","Price","Quantity","HashCode"),xscrollcommand=scr
oll_x.set, yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM, fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)

        scroll_x.config(command=product_list.xview)
        scroll_y.config(command=product_list.yview)

        product_list.heading("ProductID", text="ProductID")
        product_list.heading("Name",text="Name")
        product_list.heading("Price",text="Price")
        product_list.heading("Quantity",text="Quantity")
        product_list.heading("HashCode", text="HashCode")

        product_list["show"] = "headings"

        product_list.column("ProductID", width=100)
        product_list.column("Name",width=100)
        product_list.column("Price",width=100)
        product_list.column("Quantity",width=100)
        product_list.column("HashCode", width=150)

        product_list.pack(fill=BOTH, side = TOP)
        data = []

        with open('shop.csv', 'r') as file:
            reader = csv.reader(file)
            for row in reader:
                data.append(row)
        product_list.delete(*product_list.get_children())
        for i in data:
            product_list.insert("", END, values=i)


class Application(tk.Tk):
    def __init__(self, inventory):
        super().__init__()
        self.inventory = inventory
        self.title("Online Shopping Management System")
        self.geometry("300x200+200+200")
        """"""
        frame = Frame(self, width=500, height=300)
        frame.pack()
        frame.place(anchor='center', relx=0.5, rely=0.5)
        img = ImageTk.PhotoImage(Image.open("login.png"))
        label = Label(frame, image=img)
        label.pack()
        self.mainloop()
        """"
        self.login_page = LoginPage(self, self.inventory)
        self.front_page = FrontPage(self, self.inventory)
        self.login_page.pack()


    def show_front_page(self):
        self.login_page.pack_forget()
        self.front_page.pack()

if __name__ == "__main__":
    inventory = Inventory()
    app = Application(inventory)
    app.mainloop()
```

CHAPTER 5

SNAPSHOTS

LOGIN PAGE

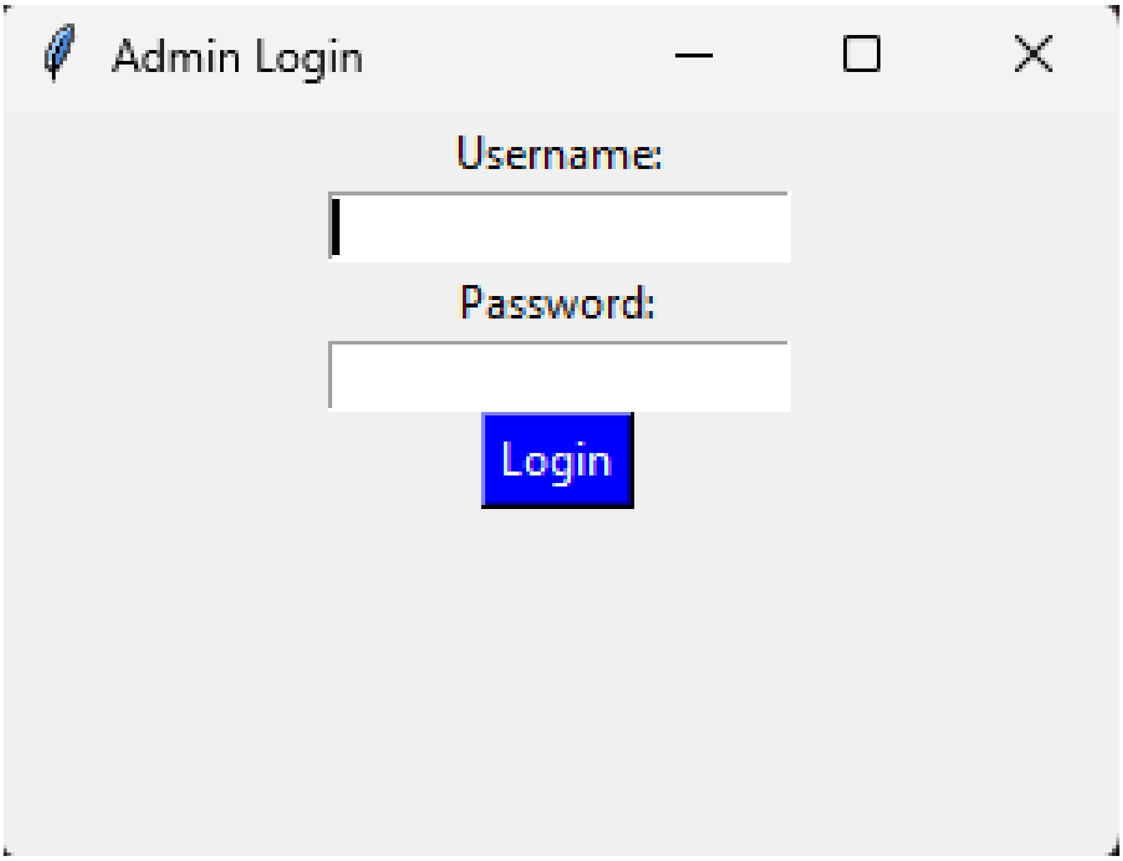


Figure 5.1: Login Page

The figure 5.1 says user should know username and password to login

MENU PAGE

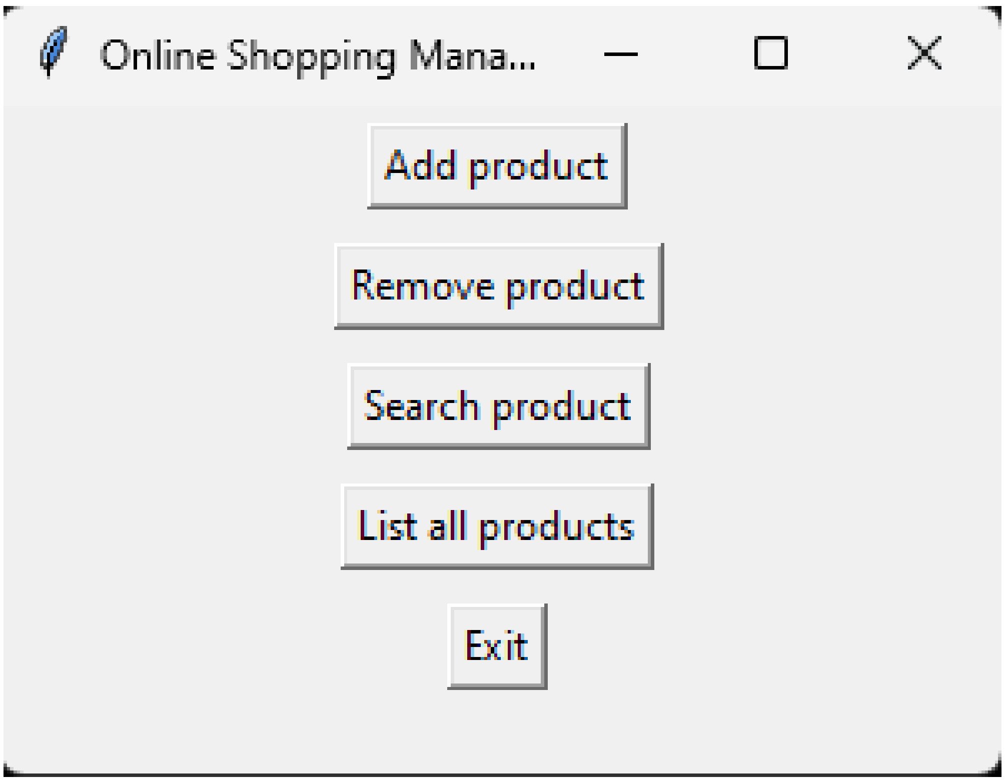


Figure 5.2: Menu Page

The figure 5.2 says user can perform any of these operations .

ADD PRODUCTS

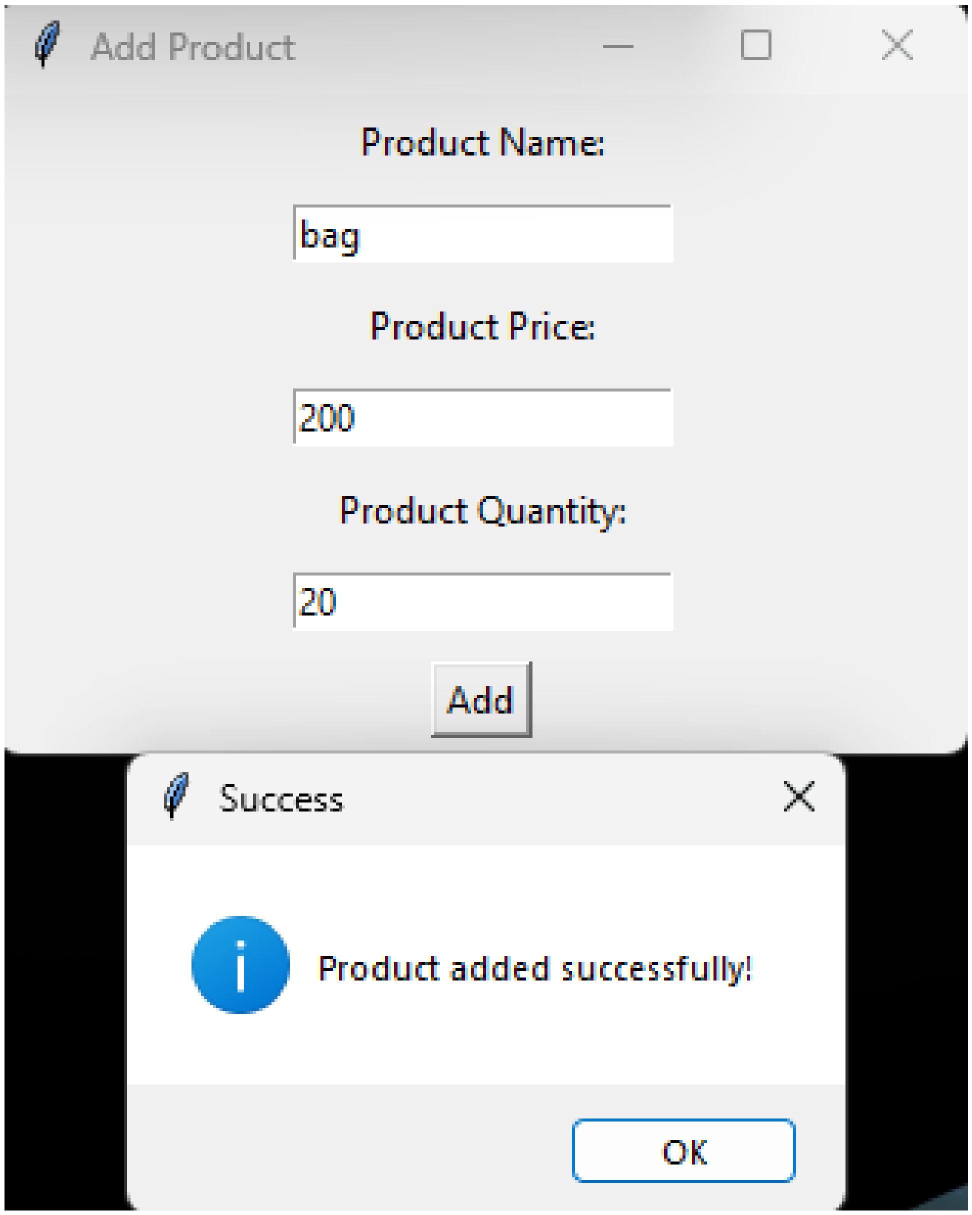
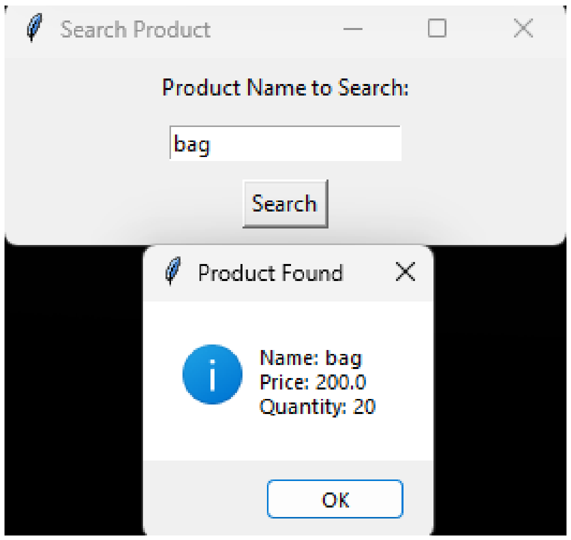


Figure 5.3: Insertion of Records

The figure 5.3 says the user can insert all records.

SEARCH PRODUCTS

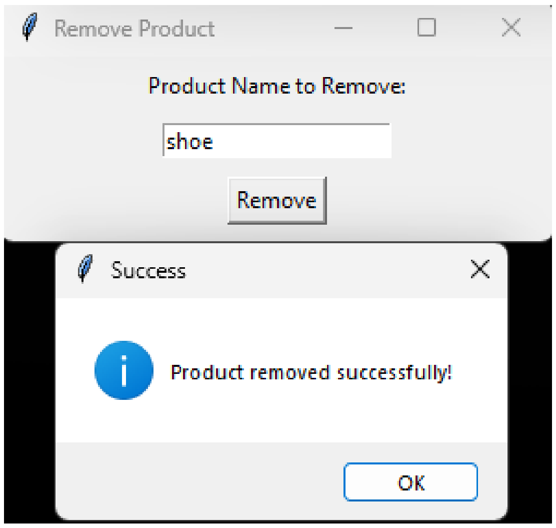


Name: bag
Price: 200.0
Quantity: 20

Figure 5.4: Search Products

The figure 5.4 says when the user click this option then user can search product.

REMOVE PRODUCT

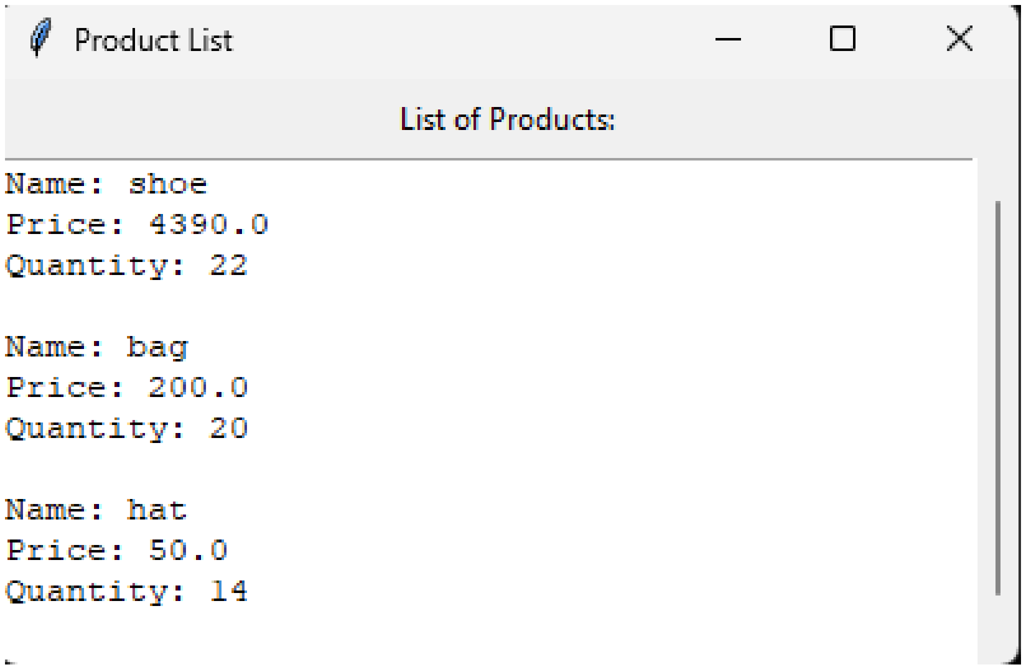


Product removed successfully!

Figure 5.5: Remove Product

The figure 5.5 says that product chose can be removed.

DISPLAY PRODUCTS



```
Name: shoe
Price: 4390.0
Quantity: 22

Name: bag
Price: 200.0
Quantity: 20

Name: hat
Price: 50.0
Quantity: 14
```

Figure 5.6: Display records

The figure 5.6 says that all the products are displayed

CHAPTER 6

CONCLUSION

The implementation of a well-designed and comprehensive online shopping management system is crucial for modern businesses to thrive in the competitive e-commerce market. Such a system not only enhances the overall shopping experience for customers but also improves operational efficiency and enables businesses to effectively manage their inventory, streamline logistics, and ensure secure transactions.

By providing personalized recommendations, accurate product information, and reliable customer support, the system fosters trust and loyalty among consumers, leading to increased sales and customer satisfaction. Moreover, the system's ability to facilitate seamless communication between buyers, sellers, and customer support further strengthens the customer-business relationship.

With the continuous advancements in technology and the growing popularity of online shopping, investing in a robust online shopping management system is a strategic move for businesses to stay relevant, drive growth, and remain competitive in the dynamic e-commerce landscape.

FUTURE ENHANCEMENT

In the future, the Online Shopping Management System can be enhanced in various ways to provide an even better shopping experience. Integration of Artificial Intelligence (AI) can enable personalized recommendations and predictive analytics to anticipate customer preferences and offer tailored product suggestions. The system could also benefit from augmented reality (AR) and virtual reality (VR) integration, allowing customers to virtually try on clothes or visualize products in their own space. Voice-activated shopping capabilities can provide a convenient hands-free experience, while seamless integration with social media platforms can enable direct purchasing and leverage social influence for product discovery..

REFERENCES

[1] Michael J. Folk, Bill Zoellick, Greg Ricchardi: File Structures -An Object-Oriented Approach with C++

[2] https://www.google.com

[3] https://www.youtube.com

[4] https://github.com

[5] https://www.geeksforgeeks.org/file-handling-c-classes/?ref=lbp

[6] https://www.python.com

[7] "Python GUI Programming with Tkinter" by Alan D. Moore

[8] "Python Crash Course" by Eric Matthes