BLS INTERNATIONAL SCHOOL

PROJECT REPORT

ON

SHOPPING MANAGEMENT SYSTEM

Submitted By:

Saksham Goyal

XII A

Submitted To:

Mr Vivek Upadhyaya

(PGT Comp. Sci. Dept.)

BONAFIDE CERTIFICATE

This is to certify that this project report on the topic “SHOPPING MANAGEMENT SYSTEM” is the Bonafide work of Saksham Goyal, Student of Class-XII(PCM). This report is submitted as a part of the Practical Board Examination in Computer Science for the session 2024-25.

_____

Signature

Mr Vivek Upadhyaya

(PGT CS Deptt.)

ACKNOWLEDGEMENT

On the auspicious occasion of submitting my Computer Project, I am deeply obliged and thankful to my Computer Science Teacher Mr. Vivek Upadhyaya without the righteous and guided assistance of whom, this project would not have seen the daylight. He devoted his precious time in bringing my work to this stature. I also pay thanks to my parents, our Principal Mam Mrs. Karnika Srivastav for giving me full support throughout the completion of my project.

Saksham Goyal

Class: XII A (PCM)

INDEX

OVERVIEW OF PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted –Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PH P.

- Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Features

Python's features include –

- Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read – Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain – Python's source code is fairly easy-to-maintain.
- A broad standard library – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable –You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases – Python provides interfaces to all major commercial databases.

- GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- Scalable – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.

ABSTRACT

This Shopping Management System is designed to support small-scale vendors by simplifying store management through digital tools. Small businesses often struggle with limited resources and manual processes, which can lead to errors and limit growth. This system addresses these issues by providing features like order tracking, customer management, stock updates, and automated billing, which help reduce paperwork and improve accuracy in cash management.

The intuitive interface makes it easy for vendors to track sales, monitor stock, and maintain customer records without technical expertise. Automated record-keeping allows vendors to manage data quickly and error-free, helping them make informed decisions based on sales trends and customer activity.

In essence, this project empowers small vendors to scale their operations efficiently, enhancing productivity and accuracy while reducing manual workload. It enables them to focus on serving customers better and growing their business in today's digital age.

The Shopping Management System provides an efficient, digital platform for managing a small-scale store, aimed at assisting vendors in tracking inventory, managing orders, and maintaining customer records. Key functionalities include:

1. Product Management: Add, update, or remove products from the system, ensuring inventory is always current.

2. Order Processing: Create and manage customer orders with automated record-keeping, which simplifies sales tracking and reduces manual errors.

3. Customer Database Management: Maintain essential customer details such as names, mobile numbers, and order histories to improve service and loyalty tracking.

4. Cart Management: Customers can add, view, and modify items in their cart, allowing flexibility before finalizing a purchase.

5. Billing and Checkout: Generate detailed bills, including tax calculations and any applied discounts, creating clear, error-free transactions.

6. Daily Sales and Earnings Report: Provides a daily summary of total sales and earnings, helping vendors assess business performance.

7. Order Search and Tracking: Customers can search their previous orders by mobile number, and admins can view order details, especially for today's orders.

8. Admin Authentication: Secure login for the admin interface to protect sensitive business information.

9. User-Friendly Interface: A professional, organized layout that ensures all functions are easy to access and navigate.

10. Order History and Record Keeping: Stores all transactions with customer and order details for future reference, reducing the need for manual paperwork.

USER MANUAL

Admin Functions:

1. Add Product: This function allows the admin to add new items to the store's inventory. When selected, the system prompts for key details like the product name, category, price, and stock quantity. After input validation, the product is saved into the database. A confirmation message displays once the product is successfully added, keeping inventory updated and accessible.

2.Update Product: This feature enables the admin to modify existing product details, such as price or stock quantity. By selecting this option, the admin provides the product ID and the updated information, which the system then validates and updates in the database. A success message confirms the update, ensuring accurate and current inventory records.

3.Delete Product: The admin can remove outdated or unavailable products from the inventory using this function. Upon entering the product ID, the system verifies its existence and deletes the product from the database. This keeps the inventory list clean and relevant, with a confirmation message displayed once the deletion is complete.

4. View Products: This function allows the admin to view all available products in a neatly formatted table. It retrieves product details such as ID, name, category, price, and stock from the database and displays them in a well-organized layout on the screen. This feature enables the admin to get a quick overview of inventory status at any time.

5. View Orders: This function enables the admin to see all orders placed on the current day, providing a clear summary of sales activity. The order details, including customer names, mobile numbers, items, and prices, are displayed in a table format, making it easy to review. At the end of the list, the system calculates and displays the total earnings for the day, assisting with daily financial tracking.

6. Search Orders by Customer: This feature allows the admin to search for specific orders based on a customer's mobile number. The system retrieves all matching orders from the database and displays them in a table format, helping the admin locate order histories quickly for customer service or record-keeping.

7. Admin Login: To secure sensitive functions, the admin must log in using a designated username and password. Upon entering valid credentials, the admin gains access to the full range of

system management options. This feature ensures that only authorized users can modify inventory or view detailed sales data.

Customer Functions:

1. Add to Cart: Customers can select items to add to their cart by entering the product ID and quantity. The system checks availability and adds the item to the cart if the stock is sufficient. After each addition, the customer is prompted to add more items if needed. This feature makes shopping convenient and customizable.

2. View Cart: This function displays the current items in the cart, showing details such as product name, quantity, and price per item. The total price for all items in the cart is calculated and displayed at the bottom, providing the customer with a clear overview before proceeding to checkout.

3. Update Cart: This feature allows customers to make adjustments to their cart items, such as changing the quantity or removing an item entirely. By keeping the cart flexible, the system ensures that customers can finalize their selection

before checkout, improving overall satisfaction and accuracy in orders.

4. Checkout: During checkout, the system generates a detailed bill that includes the itemized list of products, total price, and additional charges like GST and discounts (if applicable). The customer is prompted to enter their name and mobile number, which are saved with the order for future reference. This organized billing process adds professionalism and clarity to the transaction.

5. Order History: Customers can view past orders by entering their mobile number. The system retrieves and displays previous transactions in a list format, making it easy for customers to reference previous purchases or reorder items if needed. This feature enhances convenience and fosters loyalty by keeping a reliable record of customer interactions.

Database Structure

ALL THE TABLES USED:

```
mysql> show tables;
+--------------------+
| Tables_in_shopping |
+--------------------+
| admin_users        |
| order_items        |
| orders             |
| products           |
+--------------------+
4 rows in set (0.01 sec)
```

1. Admin Table: This table stores admin credentials for secure login. It includes:

- admin_id (INT, Primary Key): A unique identifier for the admin.
- username (VARCHAR): Admin username for login.
- password (VARCHAR): Password for secure access.

The admin table restricts access to sensitive functions and ensures only authorized users can modify inventory and sales data.

```
mysql> desc admin_users;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| admin_id | int         | NO   | PRI | NULL    | auto_increment |
| username | varchar(50) | NO   |     | NULL    |                |
| password | varchar(50) | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

2. Products Table: This table stores all product-related information in the inventory. Each row represents a unique product, with columns including:

- product_id (INT, Primary Key): A unique identifier for each product.
- name (VARCHAR): The name of the product.
- category (VARCHAR): The category to which the product belongs.
- price (FLOAT): The selling price of the product.
- stock (INT): The quantity of each product available in inventory.

This table helps maintain an organized inventory by recording essential product details, enabling easy access and updates.

```
mysql> desc products;
+----------------+---------------+------+-----+---------+----------------+
| Field          | Type          | Null | Key | Default | Extra          |
+----------------+---------------+------+-----+---------+----------------+
| product_id     | int           | NO   | PRI | NULL    | auto_increment |
| name           | varchar(100)  | NO   |     | NULL    |                |
| category       | varchar(50)   | YES  |     | NULL    |                |
| price          | decimal(10,2) | NO   |     | NULL    |                |
| stock_quantity | int           | NO   |     | NULL    |                |
| product_unit   | varchar(15)   | NO   |     | NULL    |                |
+----------------+---------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

3. Cart Table: The cart table temporarily stores products that a customer has added to their cart. It includes:

   - cart_id (INT, Primary Key): A unique identifier for each cart entry.
   - product_id (INT, Foreign Key): Links to the product in the Products table.
   - quantity (INT): The quantity of each product in the cart.

The cart table allows customers to select multiple items and make adjustments before finalizing their purchase.

```
mysql> desc order_items;
+---------------+---------------+------+-----+---------+----------------+
| Field         | Type          | Null | Key | Default | Extra          |
+---------------+---------------+------+-----+---------+----------------+
| order_item_id | int           | NO   | PRI | NULL    | auto_increment |
| order_id      | int           | YES  | MUL | NULL    |                |
| product_name  | varchar(100)  | YES  |     | NULL    |                |
| quantity      | int           | YES  |     | NULL    |                |
| price         | decimal(10,2) | YES  |     | NULL    |                |
+---------------+---------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

4. Orders Table: This table records finalized customer orders, helping track past transactions. Columns include:

   - order_id (INT, Primary Key): Unique identifier for each order.
   - customer_name (VARCHAR): The name of the customer

○ mobile_number (VARCHAR): Contact number for customer identification.

○ total_price (FLOAT): Total amount due, including taxes and discounts. ○ order_date (DATE): The date of the order.

By keeping a record of each completed transaction, this table aids in order tracking, generating daily sales reports, and viewing order history.

```
mysql> desc orders;
+-----------------+---------------+------+-----+-------------------+-------------------+
| Field           | Type          | Null | Key | Default           | Extra             |
+-----------------+---------------+------+-----+-------------------+-------------------+
| order_id        | int           | NO   | PRI | NULL              | auto_increment    |
| customer_name   | varchar(100)  | NO   |     | NULL              |                   |
| total_price     | decimal(10,2) | NO   |     | NULL              |                   |
| order_date      | timestamp     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| customer_mobile | char(10)      | NO   |     | NULL              |                   |
+-----------------+---------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)
```

Modules Used

The following Module are used for different functions used in my project:

Modules	Respective Functions/objects
OS	os.system('cls')
mysql.connector	Mysql.connector.connect() , cursor.execute() , cursor.commit() , cursor.fetchone() , cursor.fetchall()
Date Time	date.today()
Time	time.strftime('%Y-%m-%d %H:%M:%S')

SOURCE CODE:

```
import os
import mysql.connector
from datetime import date
import time

def connect_db():
    return mysql.connector.connect(
        host="localhost",
        username="root",
        password="03april2008",
        database="shopping"
    )

def clear_screen():
    os.system('cls')

def center_print(text):
    print(text.center(60))


#Functions defined for Admin Page

def add_product(cursor, connection):
    center_print("=== Add New Product ===")

    name = input("Enter product name: ")
    category = input("Enter product category: ")
    price = float(input("Enter product price: "))
    product_unit = input("Enter product unit: ")
    stock_qty = int(input("Enter product stock quantity: "))
```

```
    sql = "INSERT INTO products (name, category, price, product_unit,
stock_quantity) VALUES (%s,%s, %s, %s, %s)"
    values = (name, category, price, product_unit, stock_qty)

    cursor.execute(sql, values)
    connection.commit()

    center_print("Product '" + name + "' added successfully.")
    input("Press enter to continue...")

def update_product(cursor, connection):
    center_print("=== Update Product ===")
    product_id = int(input("Enter product ID to update: "))
    cursor.execute("SELECT * FROM products WHERE product_id = %s",
(product_id,))
    product = cursor.fetchone()

    if not product:
        clear_screen()
        center_print("Product not found.")
        return

    center_print("Current details:")
    center_print("Name = " + product[1] + ", Category = " + product[2] +",
Price = " + str(product[3]) + ", Unit = " product[5] + ", Stock = " +
str(product[4]))

    name = input("Enter new name (press enter to skip): ") or product[1]
    category = input("Enter new category (press enter to skip): ") or
product[2]
    price = input("Enter new price (press enter to skip): ") or product[3]
    product_unit = input("Enter new unit (press enter to skip): ") or
product[5]
```

```
    stock_quantity = input("Enter new stock quantity (press enter to skip):
") or product[4]

    sql = "UPDATE products SET name = %s, category = %s, price = %s,
product_unit = %s, stock_quantity = %s WHERE product_id = %s"
    values = (name, category, price, product_unit, stock_quantity,
product_id)

    cursor.execute(sql, values)
    connection.commit()


    center_print("Product ID '" + str(product_id) + "' updated successfully.")
    input("Press enter to continue...")


def delete_product(cursor, connection):
    center_print("=== Delete Product ===")
    product_id = int(input("Enter product ID to delete: "))

    sql = "DELETE FROM products WHERE product_id = %s"

    cursor.execute(sql, (product_id,))
    connection.commit()


    center_print("Product ID '" + str(product_id) + "' deleted successfully.")
    input("Press enter to continue...")


def view_products(cursor):
    center_print("=== Available Products ===")
    print()
```

```
    print("ID".ljust(5) + "Name".ljust(20) + "Category".ljust(15) +
"Price".ljust(10) + "Unit".ljust(7) + "Stock".ljust(10))
    print("-" * 60)

    cursor.execute("SELECT * FROM products")
    products = cursor.fetchall()

    if not products:
        center_print("No products available.")
        return

    for product in products:
        product_id = str(product[0]).ljust(5)
        name = product[1].ljust(20)
        category = product[2].ljust(15)
        price = str(product[3]).ljust(10)
        product_unit=str(product[5]).ljust(7)
        stock = str(product[4]).ljust(10)

        print(product_id + name + category + price + product_unit + stock)

    input("\nPress Enter to continue...")


def view_orders(cursor):
    print()
    center_print("=== Today's Orders ===")
    print()
    today_date = date.today()
    center_print("Date: "+ str(today_date))
    print()
```

```
    cursor.execute("SELECT order_id, customer_name, customer_mobile, total_price, TIME(order_date) as order_time FROM orders WHERE DATE(order_date)= %s",(today_date,))
    orders = cursor.fetchall()

    if not orders:
        center_print("No orders placed today yet.")
    else:
        print("Order ID".ljust(10) + "Customer Name".ljust(15) + "Mobile".ljust(14) + "Total".ljust(12) + "Order Time")
        print("-" * 60)
        total_earnings = 0
        for order in orders:
            total_earnings += order[3]
            print(str(order[0]).ljust(10) + order[1].ljust(15) + order[2].ljust(14) + "Rs. " + str(order[3]).ljust(8) + str(order[4]).ljust(10))

        print("-" * 60)
        center_print("Total Earnings Today: Rs. " + str(round(total_earnings, 2)))

    input("Press Enter to return to the menu...")
```

```
def admin_menu(cursor, connection):
    while True:
        clear_screen()
        center_print("--- Admin Menu ---")
        center_print("1. Add Product")
        center_print("2. Update Product")
        center_print("3. Delete Product")
        center_print("4. View Products")
        center_print("5. View Orders")
        center_print("6. Search Customer Orders")
        center_print("7. Manage Admins")
        center_print("8. Logout")

        choice = input("\nEnter your choice: ")

        if choice == "1":
            add_product(cursor, connection)
        elif choice == "2":
            update_product(cursor, connection)
        elif choice == "3":
            delete_product(cursor, connection)
        elif choice == "4":
            view_products(cursor)
        elif choice == "5":
            view_orders(cursor)
        elif choice == "6":
            search_orders(cursor)
        elif choice == "7":
            manage_admins(cursor, connection)
        elif choice == "8":
            break
        else:
            center_print("Invalid choice. Please try again."
```

```
 input()

# Functions defined for Customer page

cart = []

def view_products_customer(cursor):
    print()
    center_print("=== Available Products ===")
    print()

    print("ID".ljust(5) + "Name".ljust(20) + "Category".ljust(15) +
"Price".ljust(10) + "Unit".ljust(7) )
    print("-" * 60)

    cursor.execute("SELECT * FROM products")
    products = cursor.fetchall()

    # Display each product
    for product in products:
        product_id = str(product[0]).ljust(5)
        name = product[1].ljust(20)
        category = product[2].ljust(15)
        price = str(product[3]).ljust(10)
        product_unit=str(product[5]).ljust(7)

        print(product_id + name + category + price + product_unit)

    input("\nPress Enter to continue...")


def add_to_cart(cursor):
    clear_screen()
```

```
    view_products_customer(cursor)
    while True:

        product_id = int(input("Enter the product ID to add to cart: "))
        quantity = int(input("Enter the quantity: "))

        cursor.execute("SELECT * FROM products WHERE product_id = %s",
(product_id,))
        product = cursor.fetchone()

        if not product:
            center_print("Product not found.")
            input("Press Enter to continue...")
            continue  # Ask for product ID again

        # Check if product is available in stock
        if quantity > product[4]:
            center_print("Not enough stock available.")
            input("Press Enter to continue...")
            continue  # Ask for product ID again

        for item in cart:
            if item['product_id'] == product_id:
                item['quantity'] += quantity
                center_print("Updated quantity of '" + item['name'] + "' to " +
str(item['quantity']) + ".")
                break
        else:
            # Add product to cart
            cart.append({"product_id": product_id, "name": product[1], "price":
product[3], "product_unit":product[5], "quantity": quantity})
            center_print("Added '" + str(quantity) + "' of '" + product[1] + "' to
cart.")
```

```
        ask = input("Do you want to add more items? (yes/no): ").strip().lower()
        if ask != 'yes':
            break

    input("Press Enter to return to the menu...")


def view_cart(cursor):
    print()
    center_print("=== Your Shopping Cart ===")

    if not cart:
        center_print("Your cart is empty.")
        input("Press Enter to continue...")
        return

    # Column headers
    print("ID".ljust(5) + "Name".ljust(20) + "Price".ljust(10) + "Unit".ljust(7) +
"Quantity".ljust(10))
    print("-" * 55)

    for index, item in enumerate(cart):
        product_id = str(index + 1).ljust(5)
        name = item["name"].ljust(20)
        price = str(item["price"]).ljust(10)
        product_unit=str(item["product_unit"]).ljust(7)
        quantity = str(item["quantity"]).ljust(10)

        print(product_id + name + price + product_unit + quantity)

    input("\nPress Enter to continue...")
```

```
def update_cart(cursor):
    print()
    center_print("=== Update Cart ===")

    if not cart:
        center_print("Your cart is empty.")
        input("Press Enter to continue")
        return

    print("Current items in your cart:")

    print("ID".ljust(5) + "Product Name".ljust(20) + "Quantity".ljust(10) +
"Unit".ljust(7) + "Price")
    print("-" * 45)
    for i, item in enumerate(cart, 1):
        print(str(i).ljust(5) + item['name'].ljust(20) +
str(item['quantity']).ljust(10) + str (item['product_unit']).ljust(7) +
str(item['price']))

    item_number = int(input("Enter the item number to update (or 0 to
cancel): ").strip())

    if item_number == 0:
        return

    if 1 <= item_number <= len(cart):
        selected_item = cart[item_number - 1]

        center_print("Selected: " + selected_item['name'] + ", Quantity: " +
str(selected_item['quantity']))
        print("1. Update Quantity")
        print("2. Remove Item")
```

```
        action_choice = input("Enter your choice: ").strip()

        if action_choice == '1':
            new_quantity = int(
                input("Enter new quantity for '" + selected_item['name'] + "' (or 0 to remove): ").strip())
            if new_quantity == 0:
                cart.remove(selected_item)
                center_print("'" + selected_item['name'] + "' removed from the cart.")
            else:
                selected_item['quantity'] = new_quantity
            center_print("Quantity for '" + selected_item['name'] + "' updated to " + str(new_quantity) + ".")

        elif action_choice == '2':
            cart.remove(selected_item)
            center_print("'" + selected_item['name'] + "' removed from the cart.")

        else:
            center_print("Invalid choice. Returning to the menu.")
    else:
        center_print("Invalid item number.")

    input("Press Enter to return to the menu...")



def checkout(cursor,connection):
    print()
    center_print("=== Checkout ===")
```

```
    if not cart:
        center_print("Your cart is empty. Cannot checkout.")
        input("Press Enter to continue...")
        return

    view_cart(cursor)

    customer_name = input("Enter your name: ").strip()
    customer_mobile = input("Enter your mobile number: ").strip()

    if customer_name =="" or customer_mobile == "":
        center_print("Enter proper details and content")
        input("Please enter to continue .. ")
        return


    total_price = sum(item["price"] * item["quantity"] for item in cart)
    gst= float(total_price) * float(0.18)
    discount= float(total_price) * float(0.10)

    final_total= float(total_price) + float(gst) - float(discount)
    cursor.execute("INSERT INTO orders (customer_name,
customer_mobile, total_price)VALUES (%s, %s, %s)", (customer_name,
customer_mobile, final_total))

    connection.commit()

    order_id = cursor.lastrowid

    for item in cart:
```

```
        cursor.execute("INSERT INTO order_items (order_id, product_name,
quantity, price)VALUES (%s, %s, %s, %s)", (order_id, item['name'],
item['quantity'], item['price']))

    connection.commit()

    center_print("Your Total: " + str(final_total) + " Rs")
    confirm = input("Confirm checkout? (yes/no): ").strip().lower()

    if confirm == "yes":
        center_print("Checkout successful. Thank you for your purchase!")
        print_bill(customer_name, customer_mobile, total_price, gst,
discount, final_total)
        cart.clear()
    else:
        center_print("Checkout cancelled.")

    input("Press Enter to return to the menu...")


def print_bill(customer_name, customer_mobile, total_price, gst,
discount, final_total):
    clear_screen()
    center_print("=" * 60)

    center_print("SHOPPING MANAGEMENT SYSTEM")
    center_print("=" * 60)

    center_print(("Customer: " + customer_name))
    center_print(("Mobile: " + customer_mobile))
    center_print(("Date: " + time.strftime('%Y-%m-%d %H:%M:%S')))
    center_print("-" * 60)
```

```
    print("Product Name".ljust(20) + "Quantity".ljust(10) + "Price".ljust(10) + "Total".ljust(10))
    center_print("-" * 60)

    for item in cart:
        total_item_price = item['quantity'] * item['price']
        print(item['name'].ljust(20) + str(item['quantity']).ljust(3) + str(item["product_unit"]).ljust(7) + "Rs. " + str(item['price']).ljust(10) + "Rs. " + str(total_item_price).ljust(8))

    print("-" * 60)

    print("Subtotal:".ljust(40) + "Rs. " + str(round(total_price, 2)).ljust(10))
    print("GST (18%):".ljust(40) + "Rs. " + str(round(gst, 2)).ljust(10))
    print("Discount:".ljust(40) + "Rs. " + str(round(discount, 2)).ljust(10))

    print("=" * 60)
    print("Total Amount Payable: Rs. " + str(round(final_total, 2)))
    print("=" * 60)

    center_print("\nThank you for shopping with us!")

def search_orders(cursor):
    clear_screen()
    center_print("=== Search Customer Orders ===")
    print()
    mobile_number = input("Enter mobile number: ").strip()

    cursor.execute("SELECT order_id, customer_name, total_price, DATE(order_date) FROM orders WHERE customer_mobile = %s", (mobile_number,))

    orders = cursor.fetchall()
```

```
    if not orders:
        center_print("No orders found for this mobile number.")
        input("Press Enter to return to the menu...")
        return
    print()
    center_print("=== Order History ===")
    print("Order ID".ljust(15) + "Customer Name".ljust(20) + "Total
Price".ljust(15) + "Order Date")
    print("-" * 60)

    for order in orders:
        print(str(order[0]).ljust(15) + order[1].ljust(20) + ("Rs. " +
str(order[2])).ljust(15) + str(order[3]))
    print("-" * 60)
    print()
    order_id = input("Enter Order ID to view details (or press Enter to go
back): ").strip()

    if order_id:
        view_order_details(cursor, order_id)
    else:
        input()


def view_order_details(cursor, order_id):
    print()
    center_print("=== Order Details ===")
    print()
    cursor.execute("SELECT order_items.product_name,
order_items.quantity, order_items.price FROM order_items WHERE
order_items.order_id = %s", (order_id,))
    items = cursor.fetchall()
```

```
    print("Product Name".ljust(20) + "Quantity".ljust(10) + "Price".ljust(10) +
"Total".ljust(10))
    print("-" * 60)

    for item in items:
        total_item_price = item[1] * item[2]
        print(item[0].ljust(20) + str(item[1]).ljust(10) +
              "Rs. " + str(item[2]).ljust(8) + "Rs. " + str(total_item_price).ljust(8))

    print("-" * 60)
    print()
    input("Press Enter to return to the menu...")


def customer_menu(cursor,connection):
    while True:
        clear_screen()
        center_print("--- Customer Menu ---")
        center_print("1. View Products")
        center_print("2. Add to Cart")
        center_print("3. View Cart")
        center_print("4. Update Cart")
        center_print("5. Checkout")
        center_print("6. Search Previous Orders")
        center_print("7. Exit")

        choice = input("\nEnter your choice: ")

        if choice == "1":
            view_products_customer(cursor)
        elif choice == "2":
            add_to_cart(cursor)
```

```
        elif choice == "3":
            view_cart(cursor)
        elif choice == "4":
            update_cart(cursor)
        elif choice == "5":
            checkout(cursor,connection)
        elif choice=="6":
            search_orders(cursor)
        elif choice =="7":
            break
        else:
            center_print("Invalid choice. Please try again.")
            input()


#Function for welcome page and login

def admin_login(cursor):

    clear_screen()
    center_print("=== Admin Login ===")

    username = input("Enter admin username: ")
    password = input("Enter admin password: ")

    cursor.execute("SELECT * FROM admin_users WHERE username = %s
AND password = %s", (username, password))
    admin = cursor.fetchone()

    if admin:
        center_print("Login successful!")
        input("Press Enter to continue...")
        return True
```

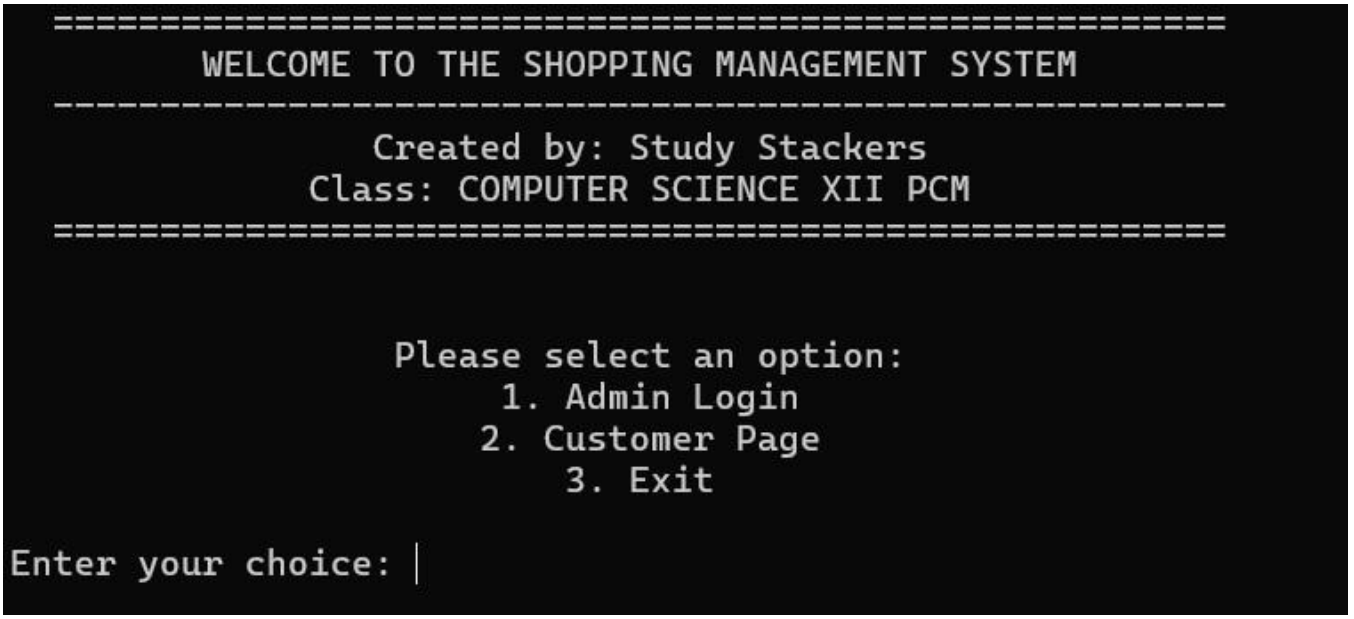
```
    else:
        center_print("Invalid username or password.")
        input("Press Enter to continue...")
        return False

def manage_admins(cursor, db):
    while True:
        clear_screen()
        center_print("=== Manage Admins ===")
        center_print("1. Add Admin")
        center_print("2. Change Password")
        center_print("3. Delete Admin")
        center_print("4. Return to Admin Menu")

        choice = input("Enter your choice: ").strip()

        if choice == '1':
            add_admin(cursor, db)
        elif choice == '2':
            change_admin_password(cursor, db)
        elif choice == '3':
            delete_admin(cursor, db)
        elif choice == '4':
            break
        else:
            center_print("Invalid choice. Please try again.")
            input()

def delete_admin(cursor, db):
    print()
    username = input("Enter the admin username to delete: ")
```

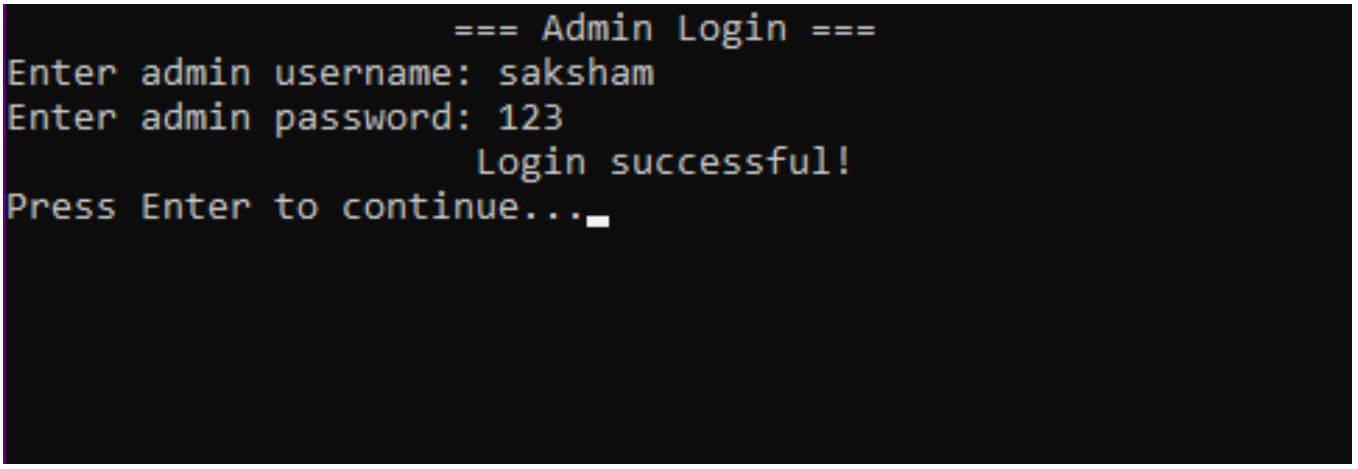
```
    cursor.execute("DELETE FROM admin_users WHERE username = %s",
(username,))
    db.commit()
    center_print("Admin user '" + username + "' deleted successfully.")
    input("Press enter to continue...")

def change_admin_password(cursor, db):
    print()
    username = input("Enter admin username: ")
    old_password = input("Enter current password: ")

    # Check if the admin exists
    cursor.execute("SELECT * FROM admin_users WHERE username = %s
AND password = %s", (username, old_password))
    admin = cursor.fetchone()

    if admin:
        new_password = input("Enter new password: ")

        # Update the password in the database
        cursor.execute("UPDATE admin_users SET password = %s WHERE
username = %s", (new_password, username))
        db.commit()
        center_print("Password changed successfully.")
        input("Press enter to continue...")
    else:
        center_print("Invalid username or current password.")
        input("Press enter to continue...")

def add_admin(cursor, db):
    print()
    username = input("Enter new admin username: ")
    password = input("Enter new admin password: ")
```

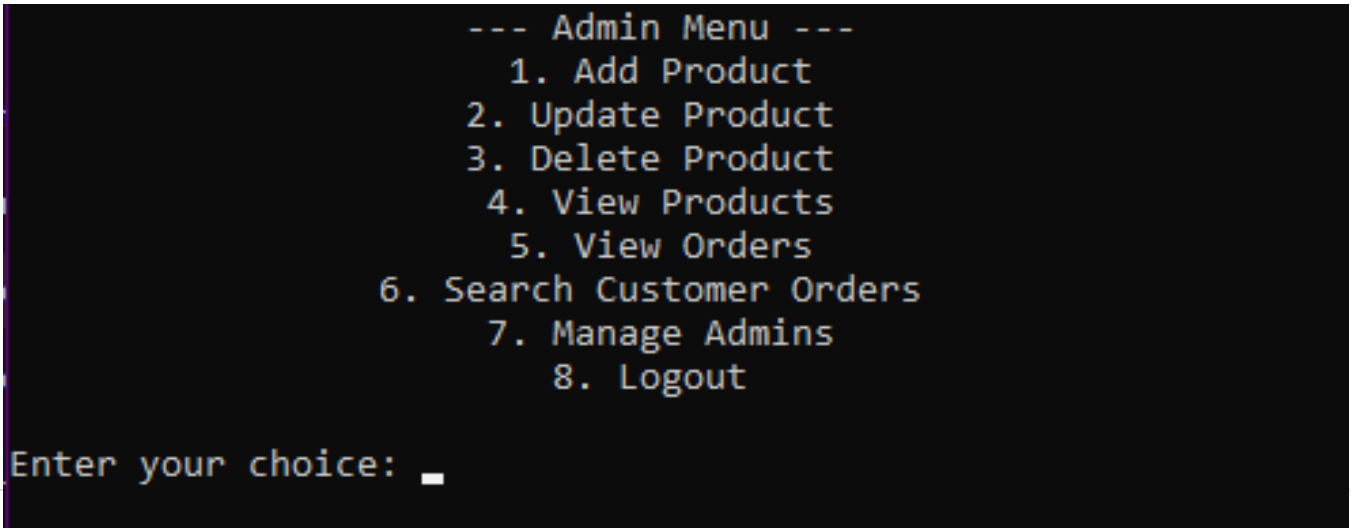
```
    cursor.execute("INSERT INTO admin_users (username, password)
VALUES (%s, %s)", (username, password))
    db.commit()
    center_print("Admin user '" + username + "' added successfully.")
    input("Press enter to continue...")

def welcome_page():
    clear_screen()
    center_print("=" * 55)

    center_print(" WELCOME TO THE SHOPPING MANAGEMENT SYSTEM ")
    center_print("-" * 55)  # Separator line
    center_print(" Created by: Study Stackers")
    center_print(" Class: COMPUTER SCIENCE XII PCM ")
    center_print("=" * 55)

    time.sleep(1)
    center_print("\n")
    center_print("Please select an option:")
    center_print("1. Admin Login")
    center_print("2. Customer Page")
    center_print("3. Exit")

    choice = input("\nEnter your choice: ")
    return choice

def main():
    # Connect to MySQL
    connection = connect_db()
    cursor = connection.cursor()

    while True:
```

```
    choice = welcome_page()

    if choice == "1":
        if admin_login(cursor):
            admin_menu(cursor, connection)
    elif choice == "2":
        customer_menu(cursor,connection)
    elif choice == "3":
        center_print("Thank you for using the system!")
        time.sleep(5)
        break
    else:
        center_print("Invalid choice. Please try again.")
        input("Press Enter to continue...")

  cursor.close()
  connection.close()

if __name__ == "__main__":
  main()
```

SCREENSHOTS





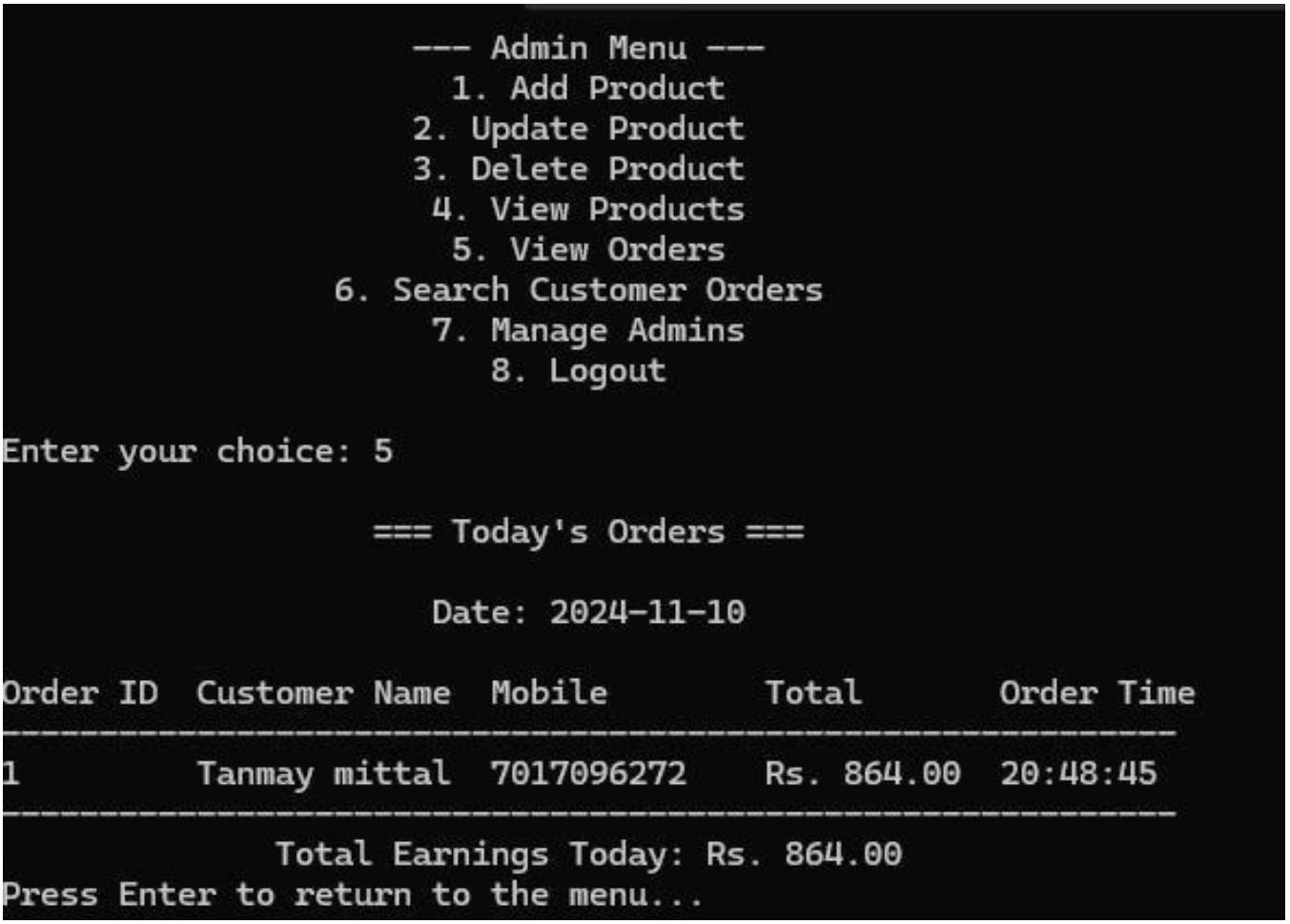


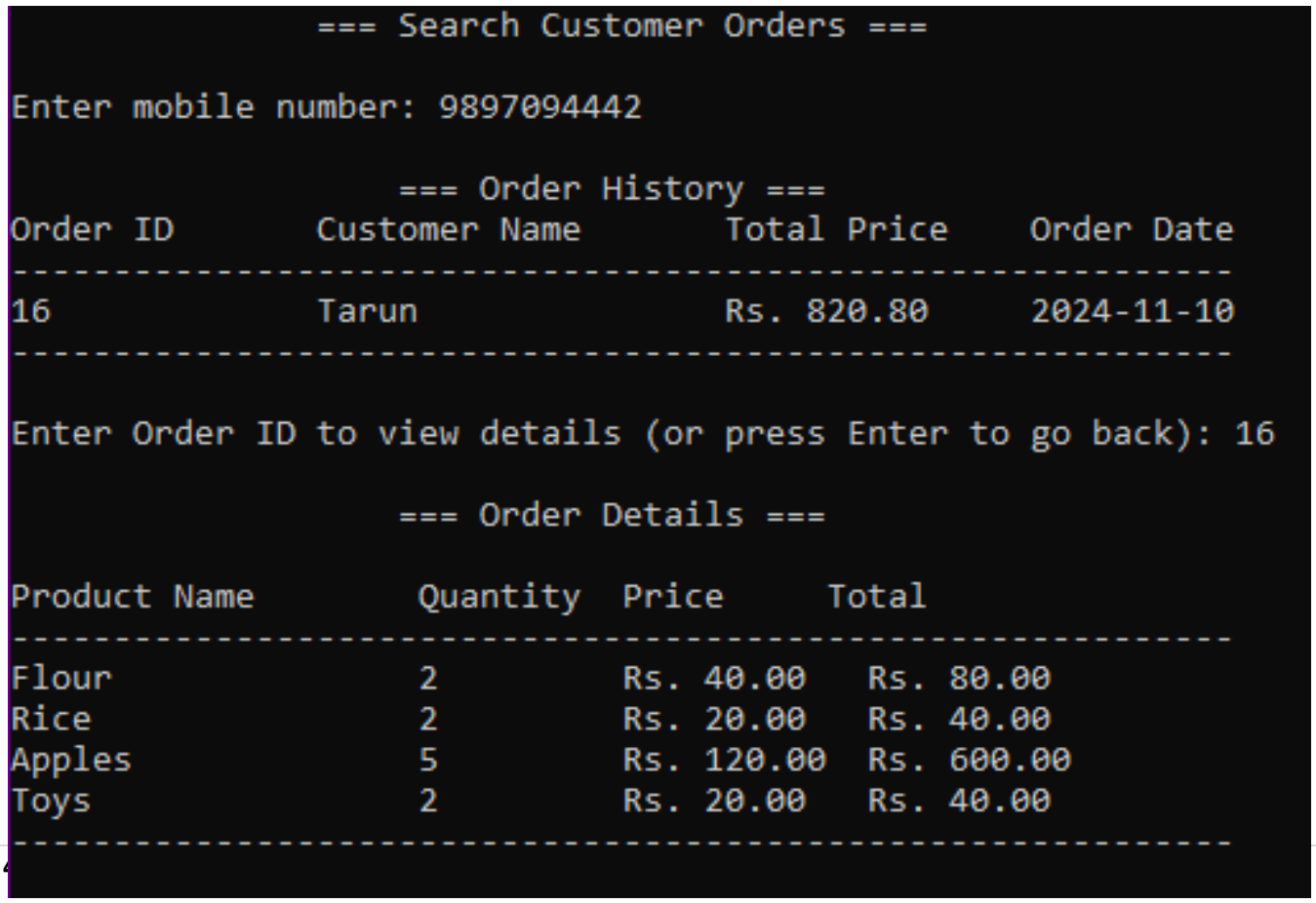
```
                    --- Admin Menu ---
                      1. Add Product
                     2. Update Product
                     3. Delete Product
                      4. View Products
                      5. View Orders
                  6. Search Customer Orders
                      7. Manage Admins
                         8. Logout

Enter your choice: 1
                  === Add New Product ===
Enter product name: Bun
Enter product category: bread
Enter product price: 71
Enter product unit: each
Enter product stock quantity: 200
```

```
                    --- Admin Menu ---
                      1. Add Product
                     2. Update Product
                     3. Delete Product
                      4. View Products
                      5. View Orders
                  6. Search Customer Orders
                      7. Manage Admins
                         8. Logout

Enter your choice: 2
                  === Update Product ===
Enter product ID to update: 1
                    Current details:
Name = Honey, Category = byproduct, Price = 100.00, Unit = , Stock = 200
Enter new name (press enter to skip): bread
Enter new category (press enter to skip): raw
Enter new price (press enter to skip): 60
Enter new unit (press enter to skip): each
Enter new stock quantity (press enter to skip): 100
           Product ID '1' updated successfully.
Press enter to continue...
```

```
                    --- Admin Menu ---
                      1. Add Product
                     2. Update Product
                     3. Delete Product
                      4. View Products
                      5. View Orders
                  6. Search Customer Orders
                      7. Manage Admins
                         8. Logout

Enter your choice: 3
                   === Delete Product ===
Enter product ID to delete: 1
            Product ID '1' deleted successfully.
Press enter to continue...
```

```
                    --- Admin Menu ---
                      1. Add Product
                     2. Update Product
                     3. Delete Product
                      4. View Products
                      5. View Orders
                  6. Search Customer Orders
                      7. Manage Admins
                         8. Logout

Enter your choice: 4
                 === Available Products ===

ID   Name               Category       Price     Unit   Stock
-------------------------------------------------------------
2    Banana             Food           40.00     kg     20
5    Glasses            Utensils       50.00     pcs    20
6    Apples             Food           120.00    kg     10
7    Washing Powder     Household      200.00    kg     5
12   Toothpaste         Household      40.00     pcs    20
13   Soap               Household      20.00     pcs    20
14   Toys               Household      20.00     pcs    2
16   Milk               Drinks         40.00     litre  5
17   Rice               Food           20.00     kg     10
19   Pulses             Food           50.00     kg     10
21   Facewash           Household      200.00    pcs    10
23   Flour              Food           40.00     kg     10

Press Enter to continue...
```


--- Admin Menu ---
1. Add Product
2. Update Product
3. Delete Product
4. View Products
5. View Orders
6. Search Customer Orders
7. Manage Admins
8. Logout
Enter your choice: 5
=== Today's Orders ===
Date: 2024-11-10
Order ID Customer Name Mobile Total Order Time
1 Tanmay mittal 7017096272 Rs. 864.00 20:48:45
Total Earnings Today: Rs. 864.00
Press Enter to return to the menu...


=== Search Customer Orders ===
Enter mobile number: 9897094442
=== Order History ===
Order ID Customer Name Total Price Order Date
16 Tarun Rs. 820.80 2024-11-10
Enter Order ID to view details (or press Enter to go back): 16
=== Order Details ===
Product Name Quantity Price Total
Flour 2 Rs. 40.00 Rs. 80.00
Rice 2 Rs. 20.00 Rs. 40.00
Apples 5 Rs. 120.00 Rs. 600.00
Toys 2 Rs. 20.00 Rs. 40.00

```
                    === Manage Admins ===
                          1. Add Admin
                       2. Change Password
                        3. Delete Admin
                    4. Return to Admin Menu
Enter your choice: 1

Enter new admin username: Vishal
Enter new admin password: 7890
          Admin user 'Vishal' added successfully.
Press enter to continue...
```

```
                    === Manage Admins ===
                          1. Add Admin
                       2. Change Password
                        3. Delete Admin
                    4. Return to Admin Menu
Enter your choice: 2

Enter admin username: Vishal
Enter current password: 7890
Enter new password: 4563
                Password changed successfully.
Press enter to continue...
```

```
                    --- Customer Menu ---
                       1. View Products
                        2. Add to Cart
                         3. View Cart
                       4. Update Cart
                         5. Checkout
                   6. Search Previous Orders
                           7. Exit

Enter your choice:
```

```
                --- Customer Menu ---
                   1. View Products
                    2. Add to Cart
                     3. View Cart
                    4. Update Cart
                     5. Checkout
               6. Search Previous Orders
                        7. Exit

Enter your choice: 1

               === Available Products ===

ID   Name                Category       Price     Unit
-------------------------------------------------------
2    Banana              Food           40.00     kg
5    Glasses             Utensils       50.00     pcs
6    Apples              Food           120.00    kg
7    Washing Powder      Household      200.00    kg
12   Toothpaste          Household      40.00     pcs
13   Soap                Household      20.00     pcs
14   Toys                Household      20.00     pcs
16   Milk                Drinks         40.00     litre
17   Rice                Food           20.00     kg
19   Pulses              Food           50.00     kg
21   Facewash            Household      200.00    pcs
23   Flour               Food           40.00     kg

Press Enter to continue...
```

```
Press Enter to continue...
Enter the product ID to add to cart: 12
Enter the quantity: 2
            Added '2' of 'Toothpaste' to cart.
Do you want to add more items? (yes/no): yes
Enter the product ID to add to cart: 16
Enter the quantity: 4
               Added '4' of 'Milk' to cart.
Do you want to add more items? (yes/no): no
Press Enter to return to the menu...
```

```
                    --- Customer Menu ---
                       1. View Products
                       2. Add to Cart
                       3. View Cart
                       4. Update Cart
                       5. Checkout
                    6. Search Previous Orders
                             7. Exit

Enter your choice: 3

                    === Your Shopping Cart ===
ID   Name                 Price     Unit   Quantity
-------------------------------------------------
1    Toothpaste           40.00     pcs    2
2    Milk                 40.00     litre  4

Press Enter to continue...
```

```
Enter your choice: 4

                      === Update Cart ===
Current items in your cart:
ID   Product Name         Quantity  Unit   Price
-----------------------------------------
1    Toothpaste           2         pcs    40.00
2    Milk                 4         litre  40.00
Enter the item number to update (or 0 to cancel): 2
                  Selected: Milk, Quantity: 4
1. Update Quantity
2. Remove Item
Enter your choice: 1
Enter new quantity for 'Milk' (or 0 to remove): 5
              Quantity for 'Milk' updated to 5.
Press Enter to return to the menu...
```

```
Enter your choice: 5

                    === Checkout ===

               === Your Shopping Cart ===
ID   Name                Price     Unit   Quantity
-----------------------------------------------------
1    Toothpaste          40.00     pcs    2
2    Milk                40.00     litre  5

Press Enter to continue...
Enter your name: Tarun
Enter your mobile number: 9897094442
                    Your Total: 302.4 Rs
Confirm checkout? (yes/no): yes
```

```
              === Search Customer Orders ===

Enter mobile number: 9897094442

                  === Order History ===
Order ID       Customer Name       Total Price    Order Date
-----------------------------------------------------------
16             Tarun               Rs. 820.80     2024-11-10
17             Tarun               Rs. 302.40     2024-11-11
18             Tarun               Rs. 302.40     2024-11-11
-----------------------------------------------------------

Enter Order ID to view details (or press Enter to go back): 17

                  === Order Details ===

Product Name        Quantity  Price     Total
-----------------------------------------------------------
Toothpaste          2         Rs. 40.00   Rs. 80.00
Milk                5         Rs. 40.00   Rs. 200.00
-----------------------------------------------------------

Press Enter to return to the menu...
```

```
============================================================
                SHOPPING MANAGEMENT SYSTEM
============================================================
                     Customer: Tarun
                    Mobile: 9897094442
                 Date: 2024-11-11 09:47:43
------------------------------------------------------------
Product Name        Quantity  Price     Total
------------------------------------------------------------
Toothpaste          2  pcs    Rs. 40.00     Rs. 80.00
Milk                5  litre  Rs. 40.00     Rs. 200.00
------------------------------------------------------------
Subtotal:                                  Rs. 280.00
GST (18%):                                 Rs. 50.4
Discount:                                  Rs. 28.0
============================================================
Total Amount Payable: Rs. 302.4
============================================================

Thank you for shopping with us!
Press Enter to return to the menu...
```

BIBLIOGRAPHY

The above project work is my own testimony prepared with the help of the following sources. This source acted as a source of inspiration to provide a better insightful knowledge about my project.

- A Textbook of Python by Sumita Arora
- Online Access of code (Google Search)
- Mr. Vivek Upadhyaya (HOD Comp. Sci. deptt. -Project Instructor)
- http://en.wikipedia.org