Inventory & Order Management System

Session: 2025 – 2029

Submitted by:

Mirza Talal Ahmed 2025(S)-CS-67

Supervised by:

Mr. Laeeq Khan Niazi

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Abstract

This project presents an Inventory & Order Management System designed to efficiently manage product stock levels and customer orders in a simplified environment. The system provides basic functionalities such as adding, updating, and viewing inventory items, as well as processing customer orders while ensuring stock availability. Developed using C++, the application operates through a console-based interface and uses structured data and modular functions to simulate real-world inventory operations. While designed with limited scope for academic purposes, the system demonstrates core principles of software engineering such as modularity, input handling, and data-driven operations. Key features include inventory tracking, order validation based on stock, and basic reporting. The project also emphasizes user experience through simple navigation and functional flow. Although the current implementation lacks persistent storage and advanced features like multi-user access and GUI, it lays a strong foundation for further development. This system serves as a practical demonstration of how software can streamline and automate inventory and order management in small-scale business environments

Table of Contents

Contents

📌 Short Description of the Project 4

👤 Users of the Application 4

✅ Functional Requirements 4

🧩 Wireframes 5

🗃️ Data Structures 13

🛠️ Function Prototypes 13

🔁 Functions Working Flow 14

💻 Complete Code 15

🧪 Test Cases 33

📌 Limitations 36

🚀 Future Work 36

📌 Short Description of the Project

The **Inventory & Order Management System** is a desktop-based application developed to streamline inventory tracking, order processing, and stock management. It allows businesses to manage items, monitor stock levels, handle customer orders, and keep a basic history of orders. The project was developed using C++.

👤 Users of the Application

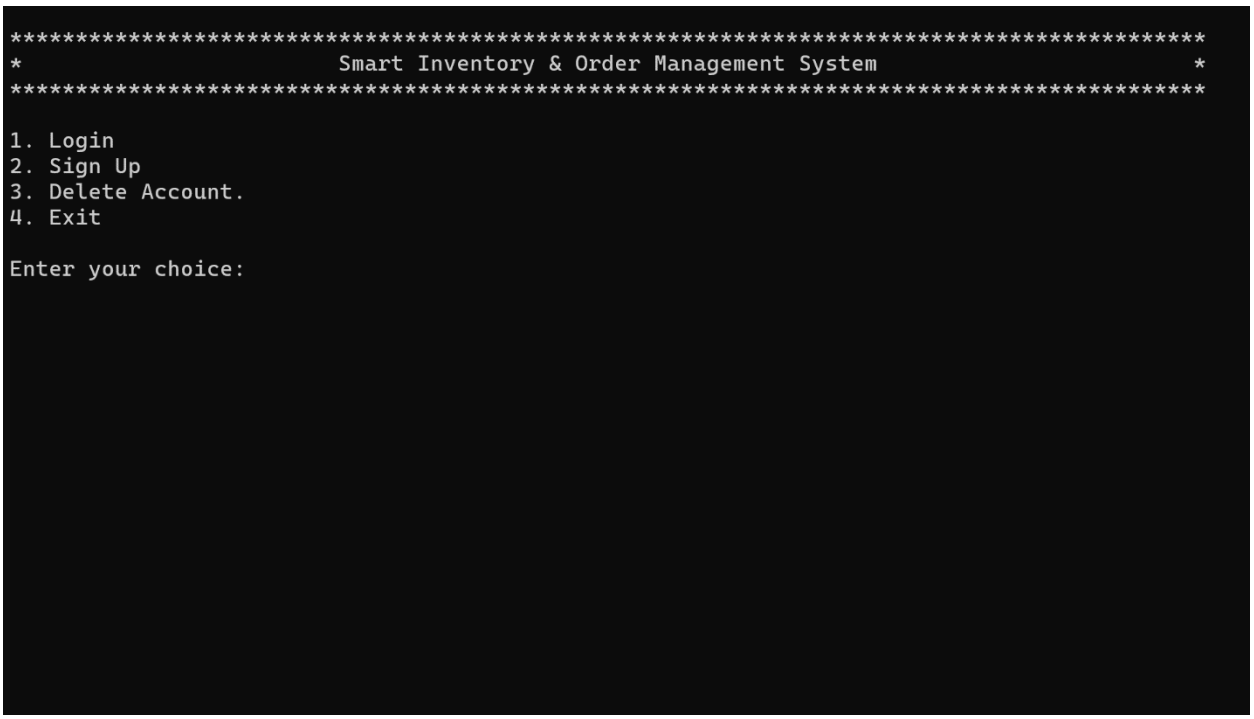
- **Admin/Manager**:
    - Add, update, or delete inventory items.
    - View a history of all the orders by various customers.
    - View and manage Stock.
- **Customers:**
    - View items and the details in the inventory.
    - Buy items
    - View a history of their own orders only.

☑ Functional Requirements

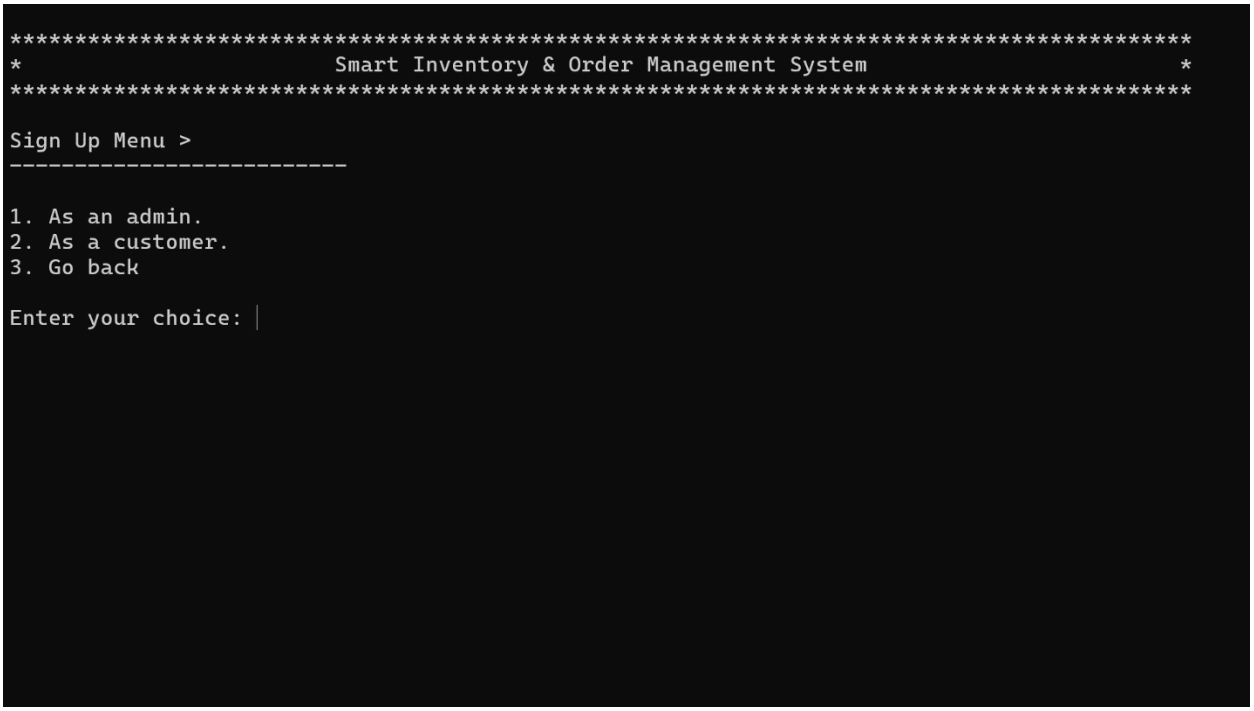
1. Add new inventory items.
2. Update or delete existing items.
3. View all inventory items.
4. Place new orders.
5. View history of all orders.
6. View history of orders by a specific person.
7. Check stock availability before placing orders.
8. Sorting items based on price.
9. Accounts based login/sign up.
10. Generate user friendly error messages.

Wireframes

- **Landing Page:**



- **Sign Up Menu:**



- **Admin Sign Up:**

```
*********************************************************************************************
*                         Smart Inventory & Order Management System                         *
*********************************************************************************************

Sign Up > Admin
------------------------

Enter the Admin PassWord to sign in as admin: admin123
Enter username: Talal
Enter password: 6890

ADMIN Added: Talal.
Your unique id is 0. Press any key to continue...
```

- **Customer Sign Up:**

```
*********************************************************************************************
*                         Smart Inventory & Order Management System                         *
*********************************************************************************************

Sign Up > Customer
------------------------

Enter username: Talal
Enter password: 1234

CUSTOMER Added: Talal.
Your unique id is 1. Press any key to continue...
```

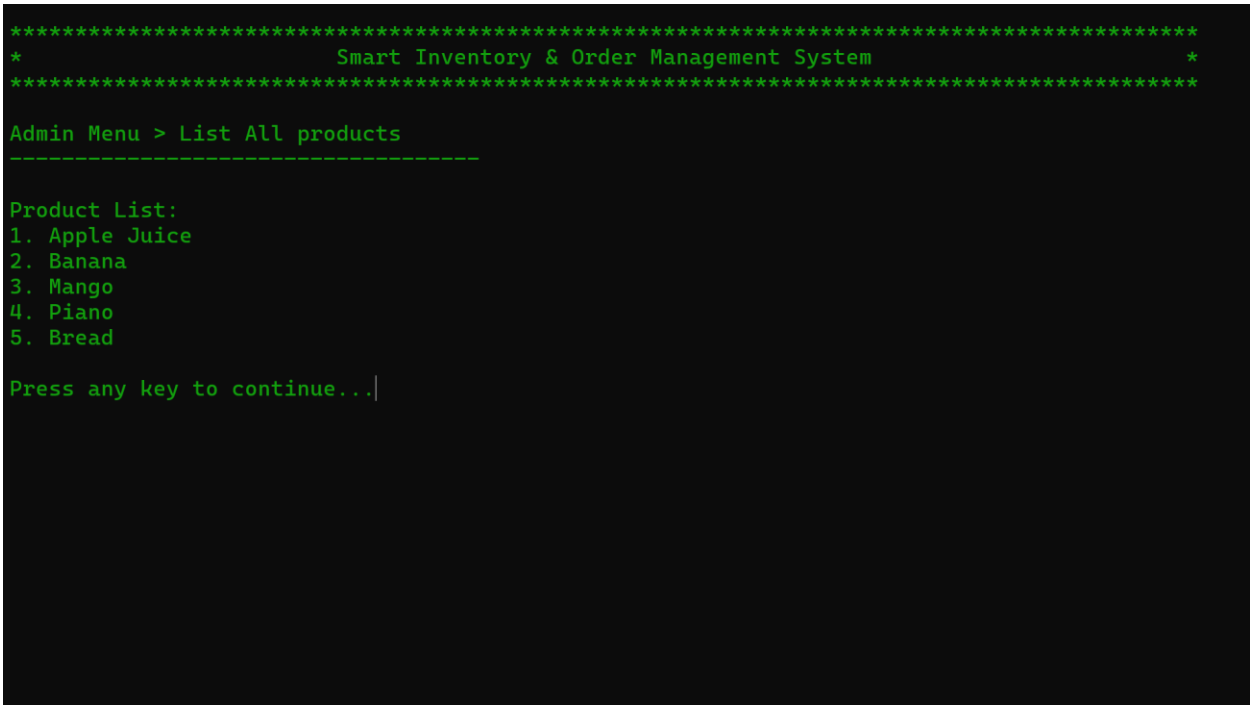
- **Admin Menu:**

```
*********************************************************************************************
*                        Smart Inventory & Order Management System                          *
*********************************************************************************************

Admin Menu >
-------------------------------------

1. Add products.
2. List All products.
3. Check product details.
4. Update product details.
5. Delete Product.
6. Orders History
7. Logout.

Enter your choice:
```

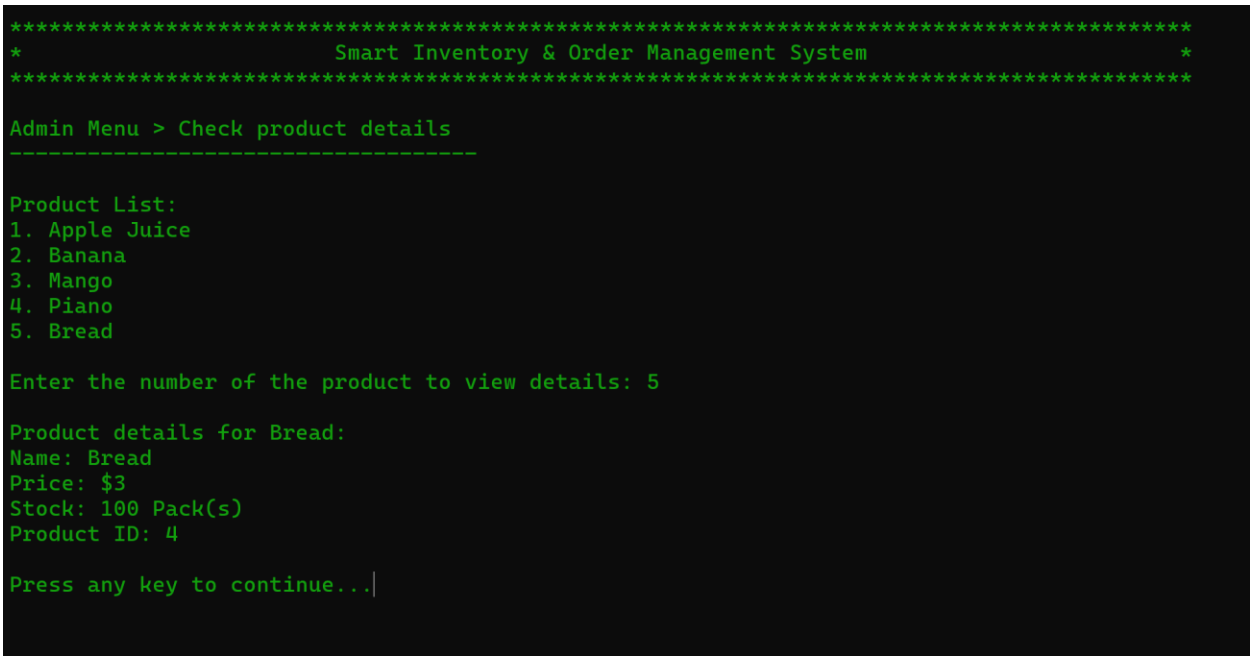
- **Admin Menu > Add Product:**

```
*********************************************************************************************
*                        Smart Inventory & Order Management System                          *
*********************************************************************************************

Admin Menu > Add products
-------------------------------------

You can enter 'q' or 'Q' to go back to previous menu any time.

Enter product name: Banana
Enter the price of 1 unit of Banana: $3
Enter the unit for Banana: Dozen
Enter number of Dozens of Banana in stock: 20

Product Added: Banana
The Unique Product ID is 0

Press any key to continue...
```

- **Admin Menu > List Products:**



- **Admin Menu > Product Details:**



- **Admin Menu > Update Product Details:**

```
*********************************************************************************************
*                          Smart Inventory & Order Management System                        *
*********************************************************************************************

Admin Menu > Update product details
-----------------------------------

Product List:
1. Banana
2. Mango
3. Apple Pie
4. Go back.

Enter the number of the product to view details: 1

You can enter 'q' or 'Q' any time to go back.

Enter new price: $4
Enter new stock number: 25
Enter new unit for Banana: Dosen
Product details updated for Banana

Press any key to continue...
```

- **Admin Menu > Delete Products:**

```
*********************************************************************************************
*                          Smart Inventory & Order Management System                        *
*********************************************************************************************

Delete Product >
------------------------

You can enter 'q' or 'Q' any time to go back.
1. Name: Banana, Product ID: 0
2. Name: Mango, Product ID: 1
3. Name: Apple Pie, Product ID: 2
Enter Product Name: Banana
Enter Product ID: 0

Product Deleted: Banana. Press any key to continue...
```

- **Admin Menu > Order History:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Admin Menu >
-----------------------------------

1. Add products.
2. List All products.
3. Check product details.
4. Update product details.
5. Delete Product.
6. Orders History
7. Logout.

Enter your choice: 6

All Order History:
1. Customer: Ali, User ID: 1: Total was $6
2. Customer: Ali, User ID: 1: Total was $6
3. Customer: Ahmed, User ID: 2: Total was $40
4. Customer: Ahmed, User ID: 2: Total was $36


Press any key to continue...
```

- **Customer Menu:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Customer Menu >
-----------------------------------
1. List All products along with the details.
2. Sort all products from cheapest to most expensive.
3. Sort all products from most expensive to cheapest.
4. Place an order.
5. Your Order History
6. Sign Out.

Enter your choice:
```

- **Customer Menu > List Products with Details:**

```
********************************************************************************************
*                        Smart Inventory & Order Management System                         *
********************************************************************************************

Customer Menu > List All products with details
----------------------------------------------

Available Products:
1.
Name: Mango
Price: $3
Stock: 30 Kgs
Product ID: 1

2.
Name: Apple Pie
Price: $2
Stock: 50 Plates
Product ID: 2


Press any key to continue...
```

- **Customer Menu > Sorting:**

```
********************************************************************************************
*                        Smart Inventory & Order Management System                         *
********************************************************************************************

Customer Menu > Sort products
-----------------------------------

Products sorted from cheapest to most expensive:
1. Apple Pie - $2
2. Mango - $3
3. Apple - $4

Press any key to continue...
```

- **Customer Menu > Place Order:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Customer Menu > Place an order
-----------------------------------


Available Products:
1. Mango - $3 (Stock: 28)
2. Apple Pie - $2 (Stock: 50)
3. Apple - $4 (Stock: 50)
4. Checkout

Enter the number of the product to order (or 4 to checkout): 1

You can enter 'q' or 'Q' to cancel this particular order. It won't checkout.
Enter the number of Kgs of Mango you want to buy: 2

2 units of Mango added to your order.

Available Products:
1. Mango - $3 (Stock: 26)
2. Apple Pie - $2 (Stock: 50)
3. Apple - $4 (Stock: 50)
4. Checkout

Enter the number of the product to order (or 4 to checkout): 4

Your total order price is: $6

Press any key to continue...
```

- **Customer Menu > Customer's Orders History:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Customer Menu >
-----------------------------------
1. List All products along with the details.
2. Sort all products from cheapest to most expensive.
3. Sort all products from most expensive to cheapest.
4. Place an order.
5. Your Order History
6. Sign Out.

Enter your choice: 5

Your Order History:

Order 1. Your total was $40
Order 2. Your total was $36
Press any key to continue...
```

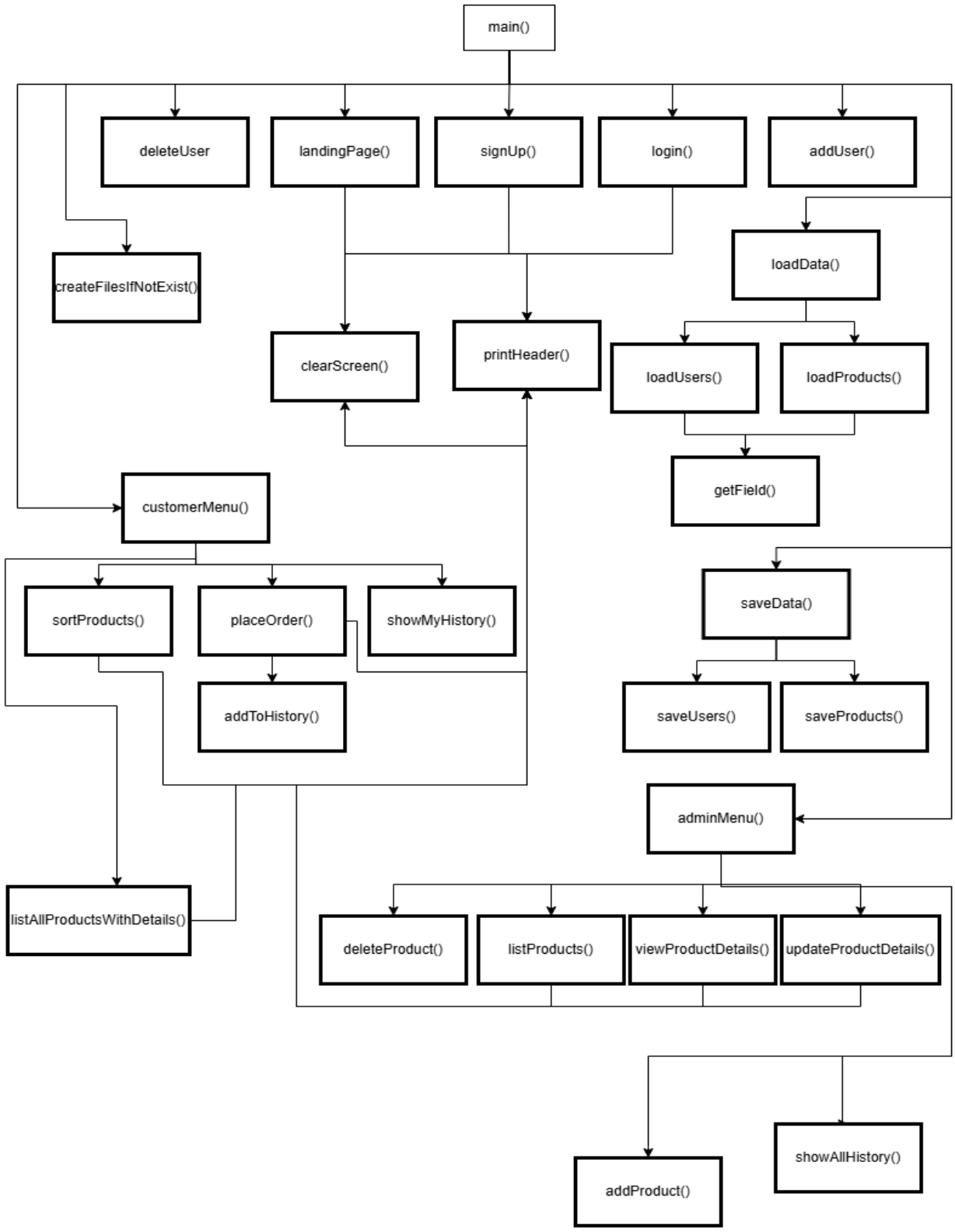
Data Structures

```
// Globol data structures START

const int TOTAL_USERS = 10;
string usernames[TOTAL_USERS];
string passwords[TOTAL_USERS];
int uniqueID[TOTAL_USERS];
string roles[TOTAL_USERS];
int userCount = 0;

const int TOTAL_PRODUCTS = 10;
string productNames[TOTAL_PRODUCTS];
float productPrices[TOTAL_PRODUCTS];
int productStocks[TOTAL_PRODUCTS];
string productUnits[TOTAL_PRODUCTS];
int productID[TOTAL_PRODUCTS];
int productCount = 0;

string loggedInAs;
int currentUserID;
string adminPassWord = "admin123";
string choice;

// Globol data structures END
```

Function Prototypes

```
// Function prototypes START
string getField(string record, int field);
void loadUsers();
void saveUsers();
void loadProducts();
void saveProducts();
void loadData();
void saveData();
void printHeader();
string deleteUser(string username, string password, int id);
string deleteProduct(string name, int id);
string landingPage();
void showAllHistory();
void adminMenu();
void clearScreen();
string login(string username, string password, int id);
void addUser(string username, string password, int id, string role);
string signUp();
void addProduct();
void listProducts();
void viewProductDetails();
void updateProductDetails();
void customerMenu();
void listAllProductsWithDetails();
void sortProducts(int sortType); // 1 for cheapest to most expensive, 2 for most expensive to cheapest
void placeOrder();
void addToHistory(string username, int userID, float total);
void showMyHistory(string username, int userID);
bool createFilesIfNotExist();

// Function prototypes END
```

Functions Working Flow



💻 Complete Code

- **Code:**

```
void printHeader() {
    cout << endl << "*****************************************************************************************" << endl <<
                    "*                          Smart Inventory & Order Management System                    *" << endl <<
                    "*****************************************************************************************" << endl << endl;
}

string landingPage() {
    clearScreen();
    printHeader();
    cout << "1. Login" << endl <<
            "2. Sign Up" << endl <<
            "3. Delete Account." << endl <<
            "4. Exit" << endl << endl <<
            "Enter your choice: ";
    cin >> choice;
    return choice;
}
```

- **Output:**

```
*****************************************************************************************
*                          Smart Inventory & Order Management System                    *
*****************************************************************************************

1. Login
2. Sign Up
3. Delete Account.
4. Exit

Enter your choice:
```

- **Code:**

```
string signUp() {
    clearScreen();
    printHeader();
    cout << "Sign Up Menu >" << endl <<
            "-------------------------" << endl << endl <<
            "1. As an admin." << endl <<
            "2. As a customer." << endl <<
            "3. Go back" << endl << endl<<
            "Enter your choice: ";
    cin >> choice;
    return choice;
}
```

- **Output:**

```
********************************************************************************************
*                          Smart Inventory & Order Management System                       *
********************************************************************************************

Sign Up Menu >
-------------------------

1. As an admin.
2. As a customer.
3. Go back

Enter your choice:
```

- **Code:**

```
choice = signUp();
clearScreen();
printHeader();
if (choice == "1") {
    cout << "Sign Up > Admin" << endl <<
            "----------------------" << endl << endl;
    cout << "Enter the Admin PassWord to sign in as admin: ";
    cin.ignore();
    string adminPass;
    getline(cin, adminPass);
    if (adminPass == adminPassWord) {
        string username, password;
        cout << "Enter username: ";
        getline(cin, username);
        cout << "Enter password: ";
        getline(cin, password);
        addUser(username, password, userCount, "ADMIN");
    }
    else {
        cout << endl << "Invalid Admin Password! Press any key to go back...";
        getch();
        continue;
    }
```

- **Output:**

```
********************************************************************************************
*                          Smart Inventory & Order Management System                       *
********************************************************************************************

Sign Up > Admin
-----------------------

Enter the Admin PassWord to sign in as admin: admin123
Enter username: Ahmed
Enter password: 123

ADMIN Added: Ahmed.
Your unique id is 3. Press any key to continue...
```

- **Code:**

```
else if (choice == "2") {
    cout << "Sign Up > Customer" << endl <<
            "-----------------------" << endl << endl;
    string username, password;
    cin.ignore();
    cout << "Enter username: ";
    getline(cin, username);
    cout << "Enter password: ";
    getline(cin, password);
    addUser(username, password, userCount, "CUSTOMER");
}
```

- **Output:**

```
**********************************************************************************************
*                        Smart Inventory & Order Management System                           *
**********************************************************************************************

Sign Up > Customer
-----------------------

Enter username: ALi
Enter password: 345

CUSTOMER Added: ALi.
Your unique id is 3. Press any key to continue...
```

- **Code:**

```
if (choice == "1") {
    if (userCount == 0) {
        cout << endl << "There are no users in the system yet. Press any key to continue...";
        getch();
        continue;
    }
    string username, password;
    int id;
    clearScreen();
    printHeader();
    cout << "Login >" << endl <<
            "-----------------------" << endl << endl;
    cout << "Enter username: ";
    cin.ignore();
    getline(cin, username);
    cout << "Enter password: ";
    getline(cin, password);
    cout << "Enter the unique id assigned to you at sign up: ";
    cin >> id;
    string w = login(username, password, id);
    while (true) {
        if (w == "ADMIN") {
            system("color 02");
            loggedInAs = username;
            currentUserID = id;
            adminMenu();
        }
        else if (w == "CUSTOMER") {
            system("color 03");
            loggedInAs = username;
            currentUserID = id;
            customerMenu();
        }
        else if (w == "WRONG") {
            cout << endl << "Wrong credentials. Press any key to try again...";
            getch();
        }
        break;
    }
    system("color 07");
}
```

- **Output:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Login >
-----------------------

Enter username: Talal
Enter password: 123
Enter the unique id assigned to you at sign up: 0
```

- **Code:**

```
void adminMenu() {
    while (true) {
        clearScreen();
        printHeader();
        cout << "Admin Menu >" << endl <<
                "-----------------------------------" << endl << endl <<
                "1. Add products." << endl <<
                "2. List All products." << endl <<
                "3. Check product details." << endl <<
                "4. Update product details." << endl <<
                "5. Delete Product." << endl <<
                "6. Orders History" << endl <<
                "7. Logout." << endl << endl <<
                "Enter your choice: ";
        cin >> choice;
        if (choice == "1") {
            addProduct();
        }
        else if (choice == "2") {
            listProducts();
        }
        else if (choice == "3") {
            viewProductDetails();
        }
        else if (choice == "4") {
            updateProductDetails();
        }
        else if (choice == "5") {
            if (productCount == 0) {
                cout << endl << "There are no users in the system yet. Press any key to continue...";
                getch();
                continue;
            }
            string name, id;
            clearScreen();
            printHeader();
            cout << "Delete Product >" << endl <<
                    "-----------------------" << endl << endl;
            cout << "You can enter 'q' or 'Q' any time to go back." << endl;
            for (int i = 0; i < productCount; i++) {
                cout << i + 1 << ". Name: " << productNames[i] << ", Product ID: " << productID[i] << endl;
            }
            cin.ignore();
            a:
            cout << "Enter Product Name: ";
            getline(cin, name);
            if (name == "") {
                cout << endl << "You didn't input anything." << endl;
                goto a;
            }
            if (name == "q" || name == "Q") return;

            b:
            cout << "Enter Product ID: ";
            getline(cin, id);
            if (id == "") {
                cout << endl << "You didn't input anything." << endl;
                goto b;
            }
            if (name == "q" || name == "Q") return;

            string w = deleteProduct(name, stoi(id));
            if (w == "DONE") {
                cout << endl << "Product Deleted: " << name << ". Press any key to continue...";
                getch();
            }
            else if (w == "FAILED") {
                cout << "Wrong Information. Press any key to continue...";
                getch();
            }
        }
        else if (choice == "6") {
            showAllHistory();
            cout << endl << endl << "Press any key to continue...";
            getch();
        }
        else if (choice == "7") {
            break;
        }
        else {
            cout << endl << "Invalid choice. Press any key to try again...";
            getch();
        }
    }
}
```

- **Output:**

```
**********************************************************************************************************
*                          Smart Inventory & Order Management System                                     *
**********************************************************************************************************

Admin Menu >
-----------------------------------

1. Add products.
2. List All products.
3. Check product details.
4. Update product details.
5. Delete Product.
6. Orders History
7. Logout.

Enter your choice:
```

- **Code:**

```
void addProduct() {
    clearScreen();
    printHeader();
    string name, unit, price, stock;
    cout << "Admin Menu > Add products" << endl;
    cout << "-----------------------------------" << endl << endl;
    cout << "You can enter 'q' or 'Q' to go back to previous menu any time." << endl << endl;
    if (productCount < TOTAL_PRODUCTS) {
        cin.ignore();
        // Product Name
        Name:
        cout << "Enter product name: ";
        getline(cin, name);
        if (name == "") {
            cout << endl << "You didn't input anything." << endl;
            goto Name;
        }
        if (name == "q" || name == "Q") return;

        // Product Price
        Price:
        cout << "Enter the price of 1 unit of " << name << ": $";
        getline(cin, price);
        if (price == "q" || price == "Q") return;
        if (price == "") {
            cout << endl << "You didn't input anything." << endl;
            goto Price;
        }
        for (int i = 0; price[i] != '\0'; i++) {
            if (!isdigit(price[i])) {
                cout << endl << "Invalid Input. Try again." << endl;
                goto Price;
            }
        }

        // Product Unit
        Unit:
        cout << "Enter the unit for " << name << ": ";
        getline(cin, unit);
        if (unit == "q" || unit == "Q") return;
        if (unit == "") {
            cout << endl << "You didn't input anything." << endl;
            goto Unit;
        }
```

```
        // Product Unit
        Unit:
        cout << "Enter the unit for " << name << ": ";
        getline(cin, unit);
        if (unit == "q" || unit == "Q") return;
        if (unit == "") {
            cout << endl << "You didn't input anything." << endl;
            goto Unit;
        }

        // Product Stock
        Stock:
        cout << "Enter number of " << unit << "s of " << name << " in stock: ";
        getline(cin, stock);
        if (stock == "q" || stock == "Q") return;
        if (stock == "") {
            cout << endl << "You didn't input anything." << endl;
            goto Stock;
        }
        for (int i = 0; stock[i] != '\0'; i++) {
            if (!isdigit(stock[i])) {
                cout << endl << "Invalid Input. Try again." << endl;
                goto Stock;
            }
        }

        productNames[productCount] = name;
        productPrices[productCount] = stof(price);
        productUnits[productCount] = unit;
        productStocks[productCount] = stoi(stock);
        productID[productCount] = productCount;
        cout << endl << "Product Added: " << productNames[productCount] << endl;

        cout << "The Unique Product ID is " << productID[productCount] << endl;
        productCount++;
        cout << endl << "Press any key to continue...";
        getch();
    }
    else {
        cout << endl << "Product limit reached (" << TOTAL_PRODUCTS << "). Press any key to continue..." << endl;
        getch();
    }
```

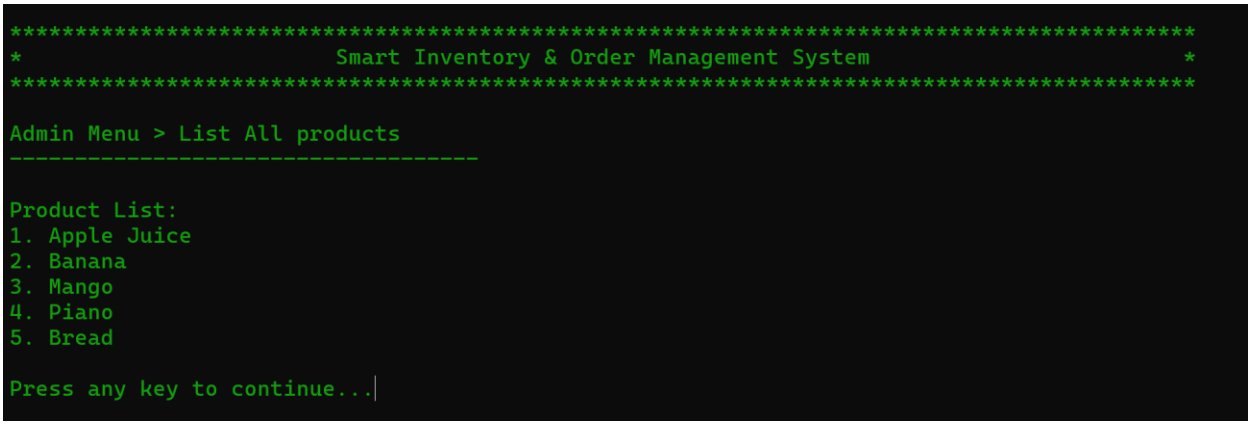
- **Output:**

```
********************************************************************************************
*                          Smart Inventory & Order Management System                       *
********************************************************************************************

Admin Menu > Add products
----------------------------------

You can enter 'q' or 'Q' to go back to previous menu any time.

Enter product name: Banana
Enter the price of 1 unit of Banana: $3
Enter the unit for Banana: Dozen
Enter number of Dozens of Banana in stock: 20

Product Added: Banana
The Unique Product ID is 0

Press any key to continue...
```

- **Code:**

```
void listProducts() {
    clearScreen();
    printHeader();
    cout << "Admin Menu > List All products" << endl;
    cout << "----------------------------------" << endl << endl;

    if (productCount == 0) {
        cout << "No products added yet. " << endl;
    }
    else {
        cout << "Product List: " << endl;
        for (int i = 0; i < productCount; i++) {
            cout << i + 1 << ". " << productNames[i] << endl;
        }
    }
    cout << endl << "Press any key to continue...";
    getch();
}
```

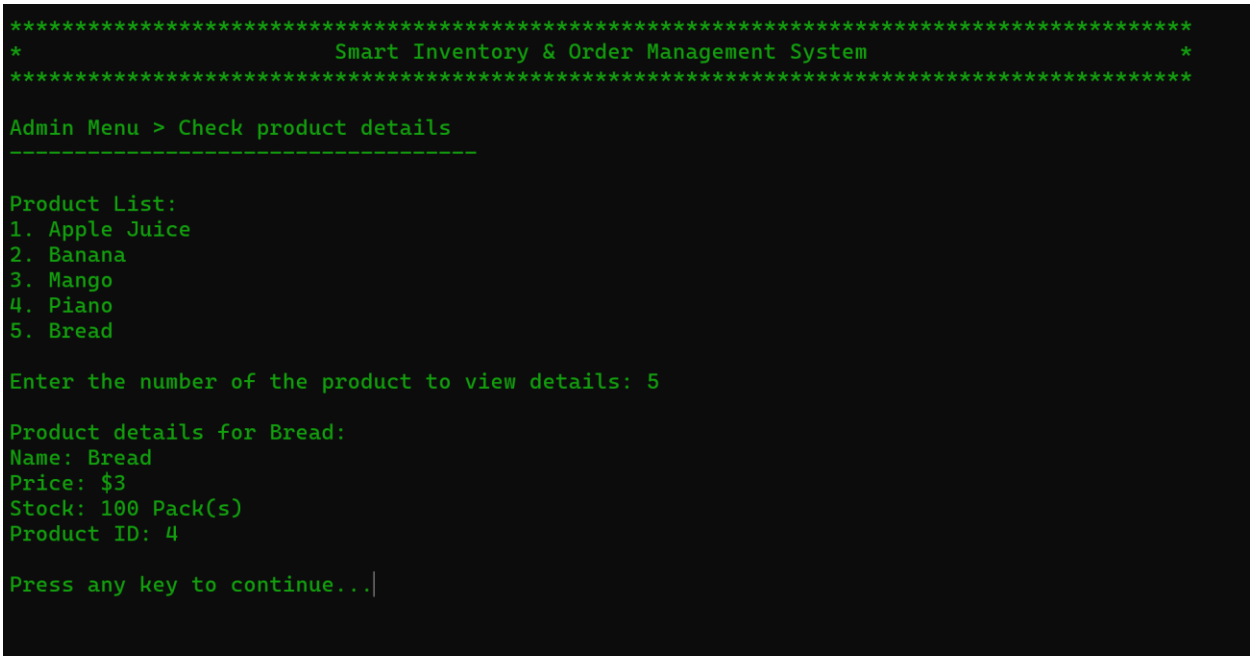
- **Output:**



- **Code:**

```
void viewProductDetails() {
    clearScreen();
    printHeader();
    cout << "Admin Menu > Check product details" << endl;
    cout << "----------------------------------" << endl << endl;

    if (productCount == 0) {
        cout << "No products added yet. " << endl;
        return;
    }
    cout << "Product List: " << endl;
    int last;
    for (int i = 0; i < productCount; i++) {
        cout << i + 1 << ". " << productNames[i] << endl;
        last = i+2;
    }
    cout << last << ". Go back" << endl;
    cout << endl << "Enter the number of the product to view details: ";
    string productNumber;
    cin.ignore();
    getline(cin, productNumber);
    int productNumberInt = stoi(productNumber);
    if (productNumber == to_string(last)) return;
    if (productNumberInt >= 1) {
        if (productNumberInt <= productCount) {
            int index = productNumberInt - 1;
            cout << endl << "Product details for " << productNames[index] << ":" << endl;
            cout << "Name: " << productNames[index] << endl;
            cout << "Price: $" << productPrices[index] << endl;
            cout << "Stock: " << productStocks[index] << " " << productUnits[index] << "(s)" << endl;
            cout << "Product ID: " << productID[index] << endl;
        }
    }
    else {
        cout << endl << "Invalid product number. " << endl;
    }
    cout << endl << "Press any key to continue...";
    getch();
}
```

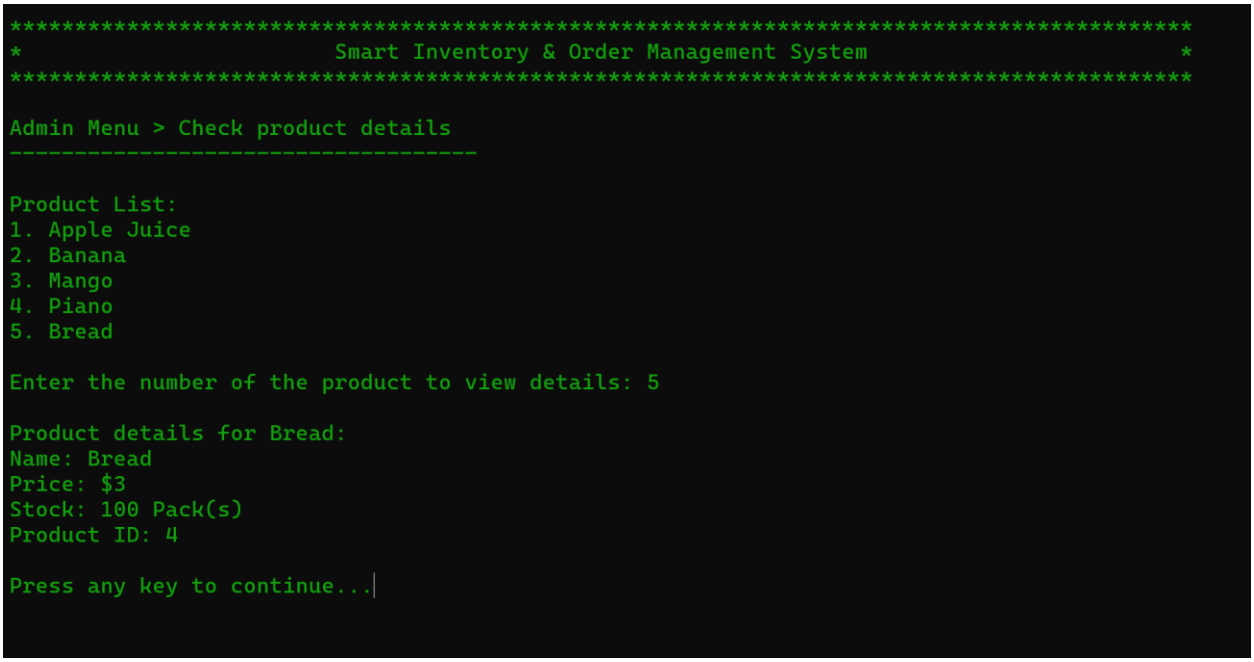
- **Output:**



- **Code:**

```
void updateProductDetails() {
    clearScreen();
    printHeader();
    cout << "Admin Menu > Update product details" << endl <<
            "-----------------------------------" << endl << endl;
    if (productCount == 0) {
        cout << "No products added yet." << endl;
        return;
    }
    int last;
    cout << "Product List: " << endl;
    for (int i = 0; i < productCount; i++) {
        cout << i + 1 << ". " << productNames[i] << endl;
        last = i + 2;
    }
    cout << last << ". Go back." << endl;
    cout << endl << "Enter the number of the product to view details: ";
    string productNumber;
    cin.ignore();
    getline(cin, productNumber);
    int productNumberInt = stoi(productNumber);
    if (productNumber == to_string(last)) return;
    if (productNumberInt >= 1) {
        if (productNumberInt <= productCount) {
            int index = productNumberInt - 1;
            string price, stock, unit;
            cout << endl << "You can enter 'q' or 'Q' any time to go back." << endl << endl;
            c:
            cout << "Enter new price: $";
            cin.ignore();
            getline(cin, price);
            if (price == "q" || price == "Q") return;
            if (price == "") {
                cout << endl << "You didn't input anything." << endl;
                goto c;
            }
            for (int i = 0; price[i] != '\0'; i++) {
                if (!isdigit(price[i])) {
                    cout << endl << "Invalid Input. Try again." << endl;
                    goto c;
                }
            }
            d:
            cout << "Enter new stock number: ";
            getline(cin, stock);
            if (stock == "q" || stock == "Q") return;
            if (stock == "") {
                cout << endl << "You didn't input anything." << endl;
                goto d;
            }
            for (int i = 0; stock[i] != '\0'; i++) {
                if (!isdigit(stock[i])) {
                    cout << endl << "Invalid Input. Try again." << endl;
                    goto d;
                }
            }
            e:
            cout << "Enter new unit for " << productNames[index] << ": ";
            getline(cin, unit);
            if (unit == "q" || unit == "Q") return;
            if (unit == "") {
                cout << endl << "You didn't input anything." << endl;
                goto e;
            }

            productPrices[index] = stof(price);
            productStocks[index] = stoi(stock);
            productUnits[index] = unit;
            cout << "Product details updated for " << productNames[index] << endl;
        }
    }
    else {
        cout << endl << "Invalid product number. " << endl;
    }
    cout << endl << "Press any key to continue...";
    getch();
}
```

- **Output:**



- **Code:**

```
string deleteProduct(string name, int id) {
    int index = -1;
    for (int i = 0; i < productCount; i++) {
        if (productNames[i] == name && productID[i] == id) {
            index = i;
        }
    }
    if (index == -1) return "FAILED";
    for (int j = index; j < productCount-1; j++) {
        productNames[j] = productNames[j + 1];
        productPrices[j] = productPrices[j + 1];
        productStocks[j] = productStocks[j + 1];
        productID[j] = productID[j + 1];
        productUnits[j] = productUnits[j + 1];
    }
    productCount--;
    return "DONE";
}
```

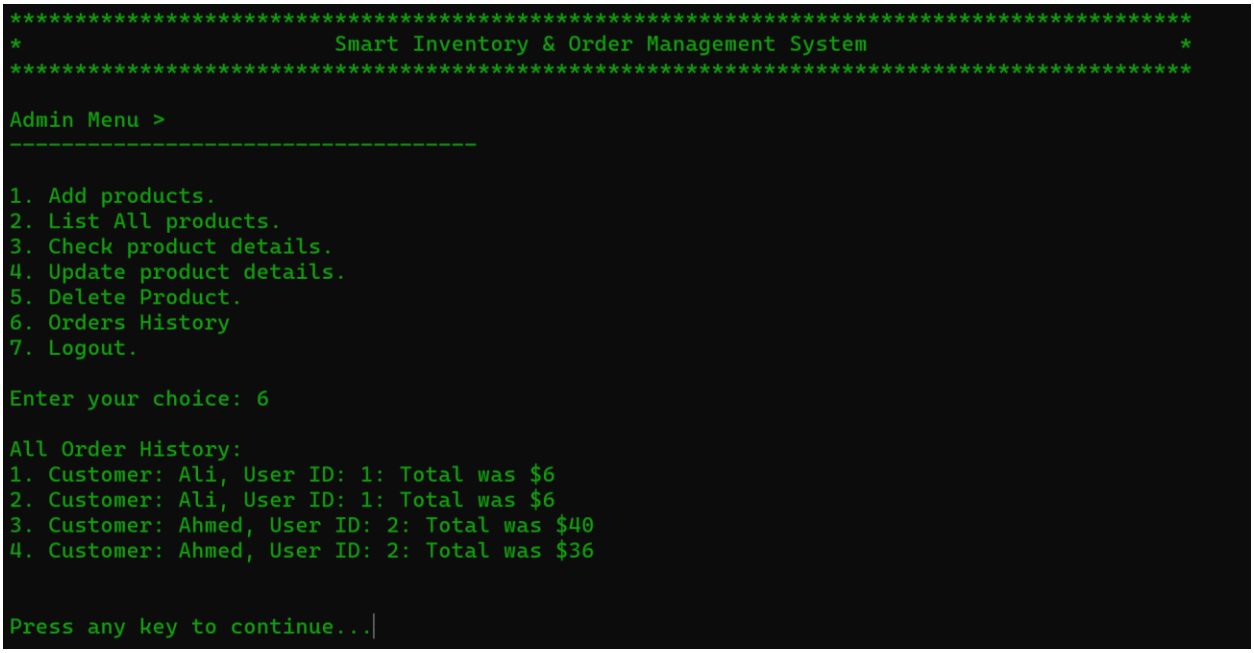
- **Output:**

```
**********************************************************************************************
*                          Smart Inventory & Order Management System                         *
**********************************************************************************************

Delete Product >
-----------------------

You can enter 'q' or 'Q' any time to go back.
1. Name: Mango, Product ID: 1
2. Name: Apple Pie, Product ID: 2
Enter Product Name: Mango
Enter Product ID: 1

Product Deleted: Mango. Press any key to continue...
```

- **Code:**

```
void showAllHistory() {
    int n = 0;
    fstream file;
    file.open("history.txt", ios::in);
    string name, id, total;
    int i = 1;
    cout << endl << "All Order History: " << endl;
    string empty;
    getline(file, empty);
    string record;
    while (!file.eof()) {
        getline(file, record);
        name = getField(record, 1);
        id = getField(record, 2);
        total = getField(record, 3);
        cout << i << ". Customer: " << name << ", User ID: " << id << ": Total was $" << total << endl;
        i++;
    }
    if (i == 1) {
        cout << endl << "No orders yet." << endl;
    }
}
```

- **Output:**



- **Code:**

```
void customerMenu() {
    while (true) {
        clearScreen();
        printHeader();
        cout << "Customer Menu >" << endl <<
                "-----------------------------------" << endl <<
                "1. List All products along with the details." << endl <<
                "2. Sort all products from cheapest to most expensive." << endl <<
                "3. Sort all products from most expensive to cheapest." << endl <<
                "4. Place an order." << endl <<
                "5. Your Order History" << endl <<
                "6. Sign Out." << endl << endl <<
                "Enter your choice: ";
        cin >> choice;
        if (choice == "1") {
            listAllProductsWithDetails();
        }
        else if (choice == "2") {
            sortProducts(1);
        }
        else if (choice == "3") {
            sortProducts(2);
        }
        else if (choice == "4") {
            placeOrder();
        }
        else if (choice == "5") {
            showMyHistory(loggedInAs, currentUserID);
        }
        else if (choice == "6") {
            break;
        }
        else {
            cout << endl << "Invalid choice. Press any key to try again...";
            getch();
            continue;
        }
        cout << endl << "Press any key to continue...";
        getch();
    }
}
```

- **Output:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Customer Menu >
--------------------------------------
1. List All products along with the details.
2. Sort all products from cheapest to most expensive.
3. Sort all products from most expensive to cheapest.
4. Place an order.
5. Your Order History
6. Sign Out.

Enter your choice:
```

- **Code:**

```
void listAllProductsWithDetails() {
    clearScreen();
    printHeader();
    cout << "Customer Menu > List All products with details" << endl;
    cout << "----------------------------------------------" << endl << endl;
    if (productCount == 0) {
        cout << "No products available." << endl;
    }
    else {
        cout << "Available Products:" << endl;
        for (int i = 0; i < productCount; i++) {
            cout << i + 1 << ".\nName: " << productNames[i] <<
                             "\nPrice: $" << productPrices[i] <<
                             "\nStock: " << productStocks[i] << " " << productUnits[i] << "s" <<
                             "\nProduct ID: " << productID[i] << endl << endl;
        }
    }
}
```

- **Output:**

```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Customer Menu > List All products with details
----------------------------------------------

Available Products:
1.
Name: Mango
Price: $3
Stock: 30 Kgs
Product ID: 1

2.
Name: Apple Pie
Price: $2
Stock: 50 Plates
Product ID: 2


Press any key to continue...
```

- **Code:**

```
void sortProducts(int sortType) {
    clearScreen();
    printHeader();
    cout << "Customer Menu > Sort products" << endl;
    cout << "-----------------------------------" << endl << endl;

    if (productCount == 0) {
        cout << "No products available to sort." << endl;
        return;
    }
    // Using copy of the original array for sorting purpose.
    string tempNames[TOTAL_PRODUCTS];
    float tempPrices[TOTAL_PRODUCTS];
    int tempStocks[TOTAL_PRODUCTS];
    for (int i = 0; i < productCount; i++) {
        tempNames[i] = productNames[i];
        tempPrices[i] = productPrices[i];
        tempStocks[i] = productStocks[i];
    }
    for (int i = 0; i < productCount - 1; i++) {
        for (int j = 0; j < productCount - i - 1; j++) {
            if ((sortType == 1 && tempPrices[j] > tempPrices[j + 1]) || (sortType == 2 && tempPrices[j] < tempPrices[j + 1])) {
                // Swaping values
                string tempName = tempNames[j];
                tempNames[j] = tempNames[j + 1];
                tempNames[j + 1] = tempName;
                float tempPrice = tempPrices[j];
                tempPrices[j] = tempPrices[j + 1];
                tempPrices[j + 1] = tempPrice;
                int tempStock = tempStocks[j];
                tempStocks[j] = tempStocks[j + 1];
                tempStocks[j + 1] = tempStock;
            }
        }
    }
    if (sortType == 1) cout << "Products sorted from cheapest to most expensive:" << endl;
    else cout  << "Products sorted from most expensive to cheapest:" << endl;
    for (int i = 0; i < productCount; i++) {
        cout << i + 1 << ". " << tempNames[i] << " - $" << tempPrices[i] << endl;
    }
}
```

- **Output:**

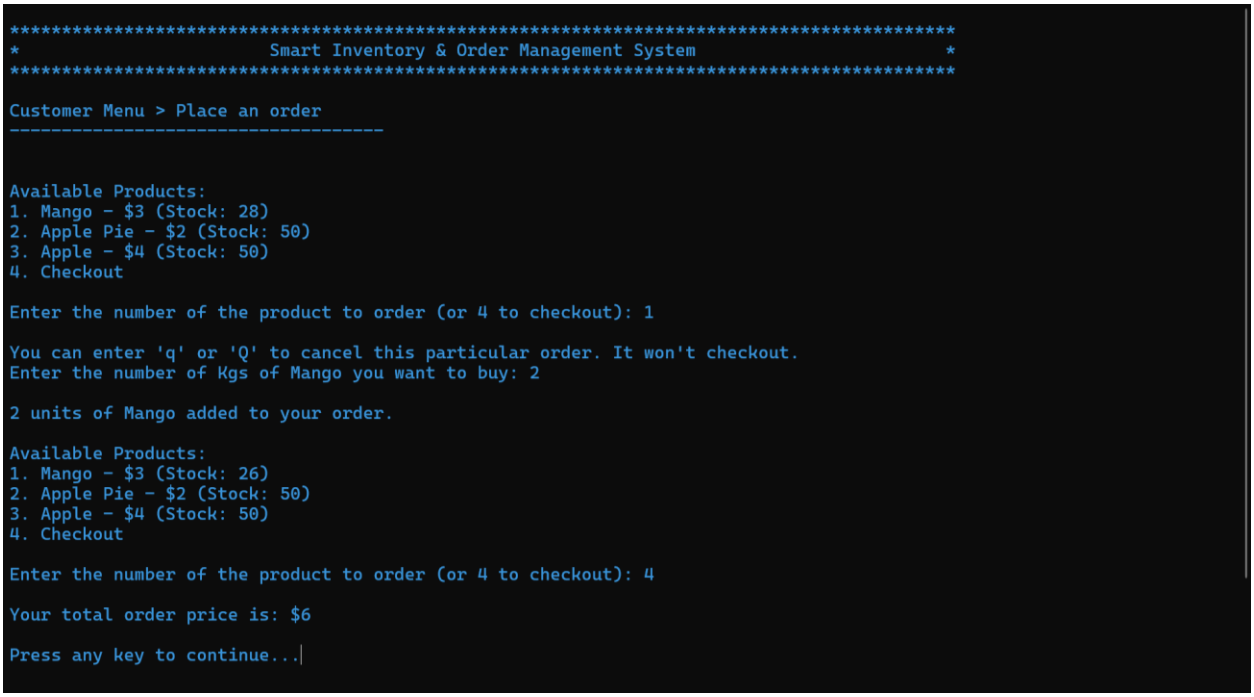
```
*****************************************************************************************
*                        Smart Inventory & Order Management System                      *
*****************************************************************************************

Customer Menu > Sort products
-----------------------------------

Products sorted from cheapest to most expensive:
1. Apple Pie - $2
2. Mango - $3
3. Apple - $4

Press any key to continue...
```

- **Code:**

```
void placeOrder() {
    clearScreen();
    printHeader();
    cout << "Customer Menu > Place an order" << endl;
    cout << "-----------------------------------" << endl << endl;
    if (productCount == 0) {
        cout << "No products available to order." << endl;
        return;
    }
    float total_price = 0.0;
    while (true) {
        cout << endl << "Available Products:" << endl;
        for (int i = 0; i < productCount; i++) {
            cout << i + 1 << ". " << productNames[i] << " - $" << productPrices[i] << " (Stock: " << productStocks[i] << ")" << endl;
        }
        cout << productCount + 1 << ". Checkout" << endl << endl;
        cout << "Enter the number of the product to order (or " << productCount + 1 << " to checkout): ";

        cin >> choice;
        int orderChoiceInt = stoi(choice);

        if (orderChoiceInt >= 1) {
            cin.ignore();
            int index = orderChoiceInt - 1;
            if (orderChoiceInt <= productCount) {
                cout << endl << "You can enter 'q' or 'Q' to cancel this particular order. It won't checkout." << endl;
                f:
                cout << "Enter the number of " << productUnits[index] << "s of " << productNames[index] << " you want to buy: ";
                string quantityStr;

                getline(cin, quantityStr);
                if (quantityStr == "") {
                    cout << endl << "You didn't input anything" << endl;
                    goto f;
                }
                if (quantityStr == "q" || quantityStr == "Q") continue;
                for (int i = 0; quantityStr[i] != '\0'; i++) {
                    if (!isdigit(quantityStr[i])) {
                        cout << endl << "Invalid Input. Try again." << endl;
                        goto f;
                    }
```

```
                int quantity = stoi(quantityStr);
                cout << endl;
                if (quantity > 0) {
                    if (quantity <= productStocks[index]) {
                        total_price += quantity * productPrices[index];
                        productStocks[index] -= quantity;
                        cout << quantity << productUnits[index] <<  "s of " << productNames[index] << " added to your order." << endl;
                    }
                    else {
                        cout << "Insufficient stock for " << productNames[index] << "." << endl;
                    }
                }
                else {
                    cout << endl << "Invalid quantity." << endl;
                }
            }
            else if (orderChoiceInt == productCount + 1) {
                cout << endl << "Your total order price is: $" << total_price << endl;
                addToHistory(loggedInAs, currentUserID, total_price);
                break;
            }
            else {
                cout << "Invalid choice. Press any key to continue..." << endl;
                getch();
            }
        }
    }
}
```

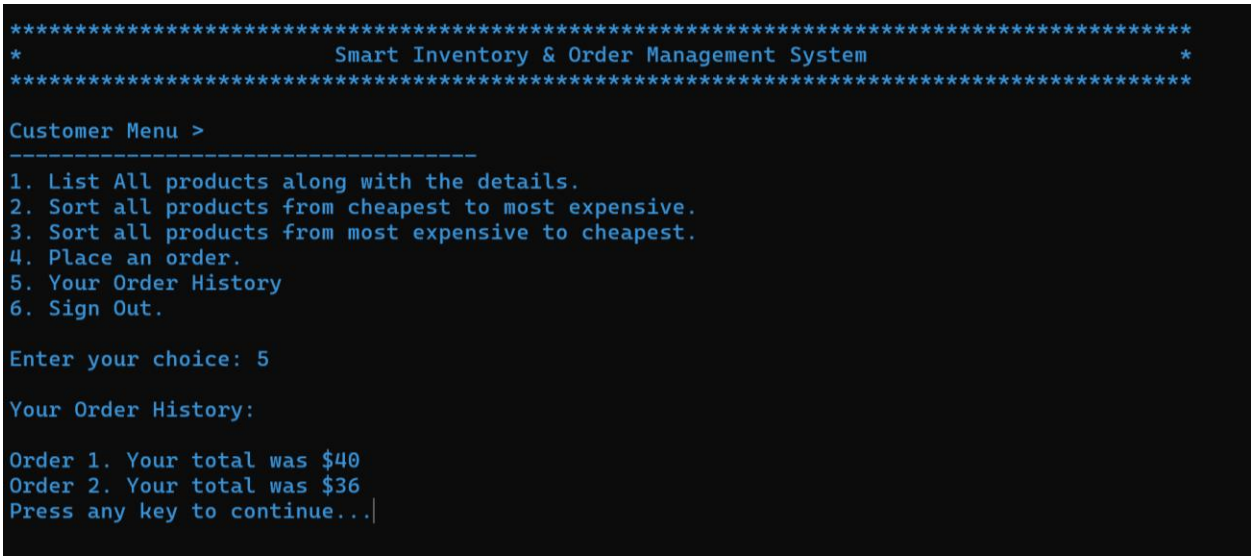
- **Output:**



- **Code:**

```
void showMyHistory(string username, int userID) {
    fstream file;
    file.open("history.txt", ios::in);
    string name;
    int id, i = 1;
    float total;
    cout << endl << "Your Order History: " << endl;
    string empty;
    getline(file, empty);
    string record;
    while(!file.eof()) {
        getline(file, record);
        name = getField(record, 1);
        id = stoi(getField(record, 2));
        total = stof(getField(record, 3));
        if (name == username && id == userID) {
            cout << endl << "Order " << i << ". Your total was $" << total;
            i++;
        }
    }
    file.close();
}
```

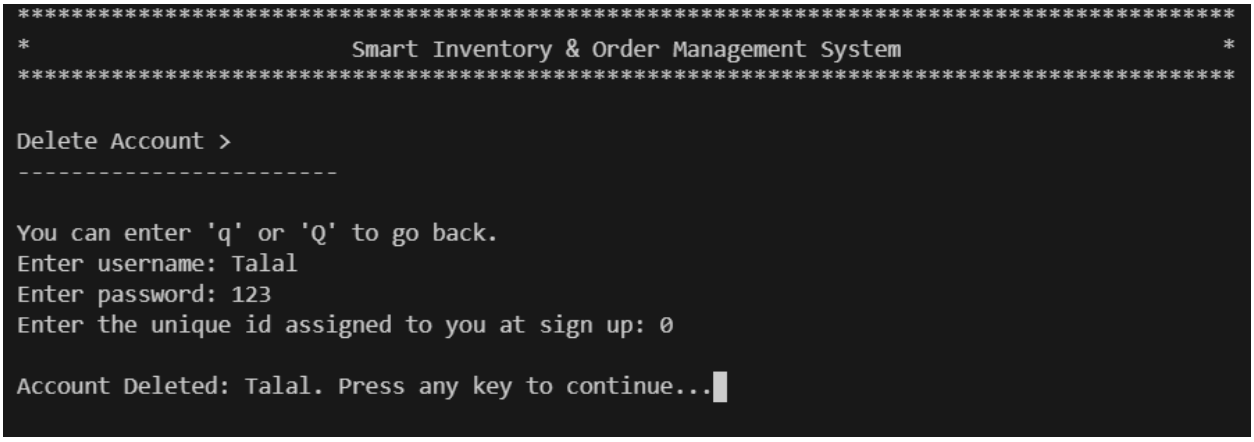
- **Output:**



- **Code:**

```
else if (choice == "3") {
    if (userCount == 0) {
        cout << endl << "There are no users in the system yet. Press any key to continue...";
        getch();
        continue;
    }
    string username, password, id;
    clearScreen();
    printHeader();
    cout << "Delete Account >" << endl <<
            "-----------------------" << endl << endl;
    cout << "You can enter 'q' or 'Q' to go back." << endl;
    cout << "Enter username: ";
    cin.ignore();
    getline(cin, username);
    if (username == "q" || username == "Q") continue;
    cout << "Enter password: ";
    getline(cin, password);
    if (password == "q" || password == "Q") continue;
    cout << "Enter the unique id assigned to you at sign up: ";
    getline(cin, id);
    if (id == "q" || id == "Q") continue;
    string w = deleteUser(username, password, stoi(id));
    if (w == "DONE") {
        cout << endl << "Account Deleted: " << username << ". Press any key to continue...";
        getch();
    }
    else if (w == "FAILED") {
        cout << "Wrong Credentials. Press any key to continue...";
        getch();
    }
}
```

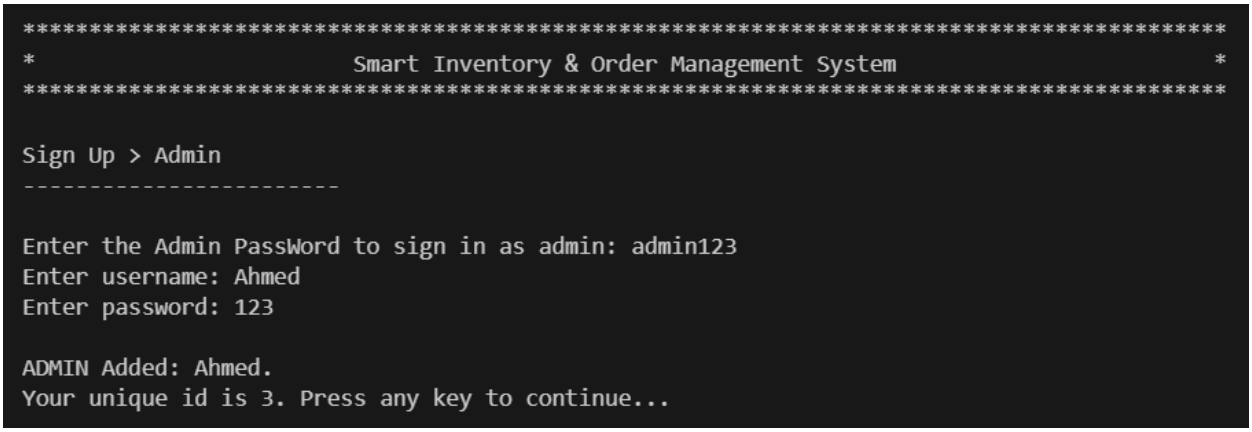
- **Output:**


****
* Smart Inventory & Order Management System *
****
Delete Account >

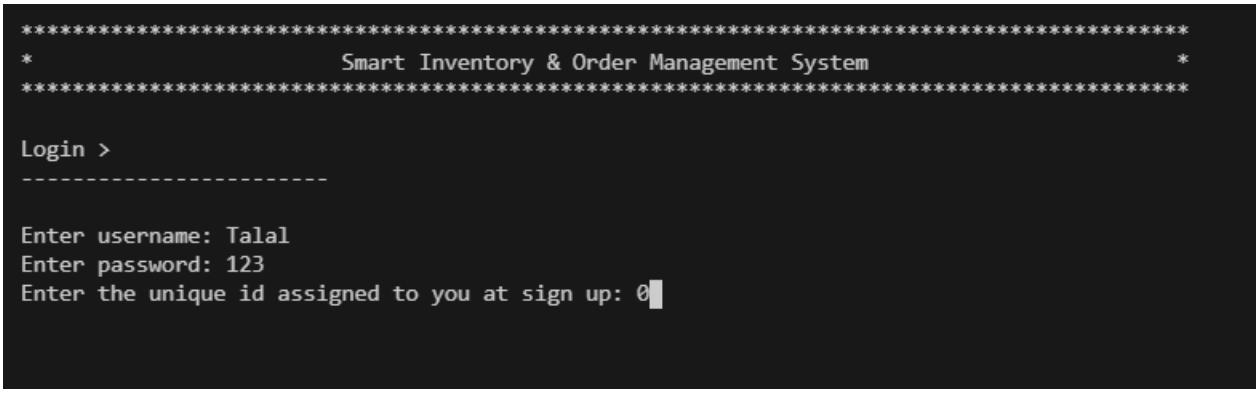
You can enter 'q' or 'Q' to go back.
Enter username: Talal
Enter password: 123
Enter the unique id assigned to you at sign up: 0
Account Deleted: Talal. Press any key to continue...

Test Cases

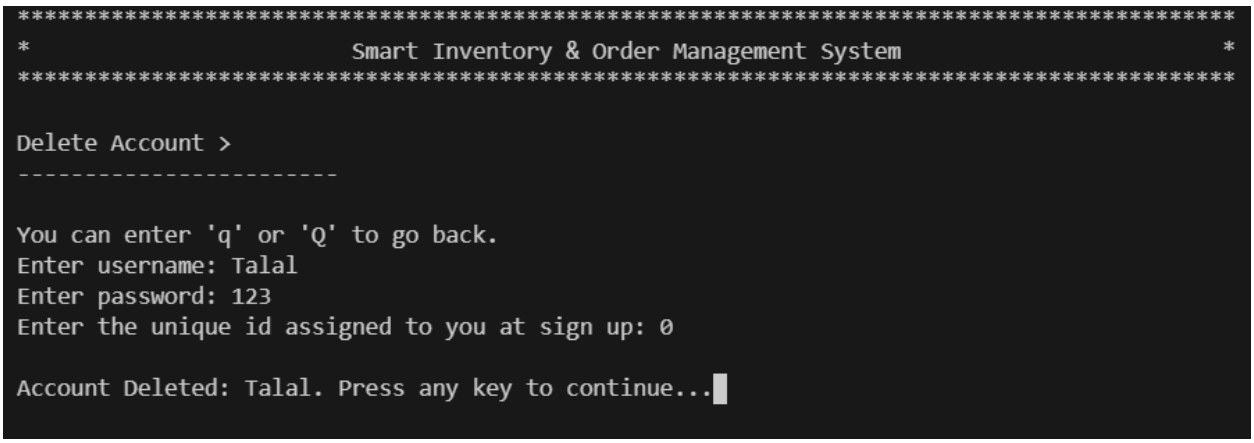
- **User Sign Up:**



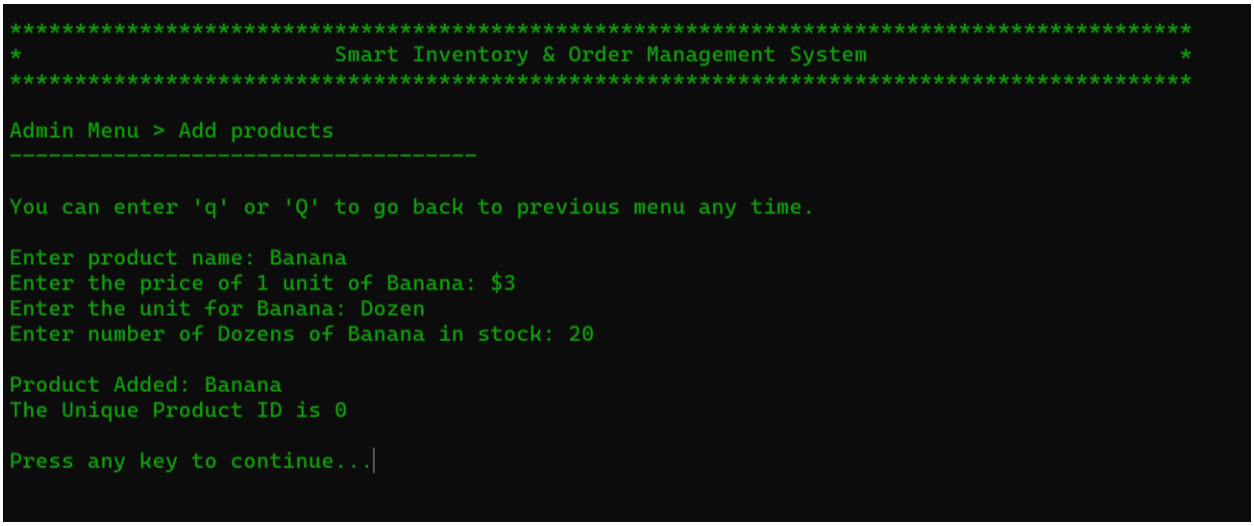
- **User Login:**



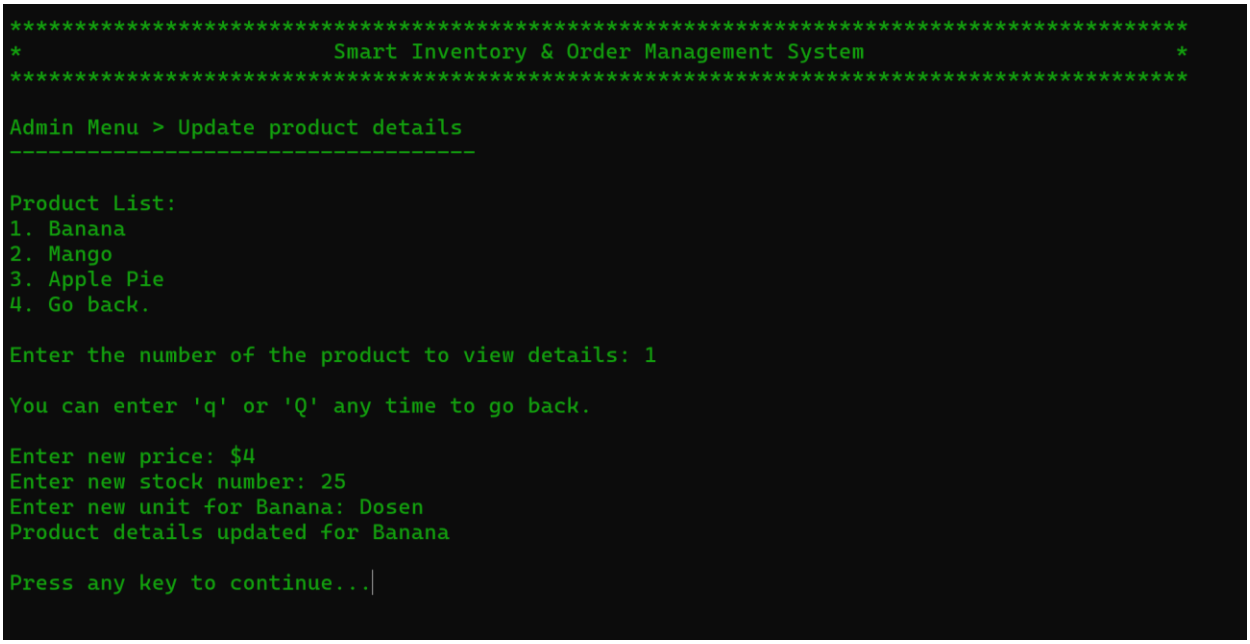
- **Deleting Account:**



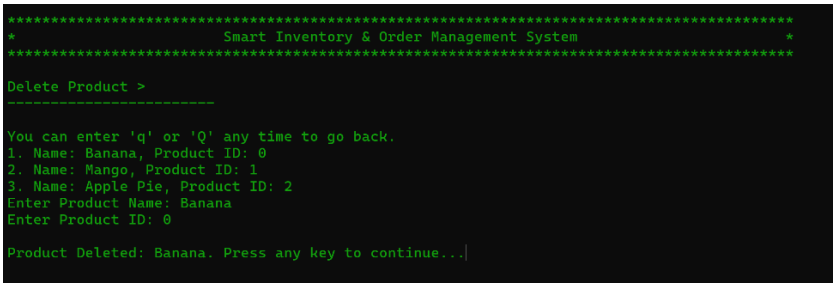
- **Add Product:**



- **Check/Update Product details:**

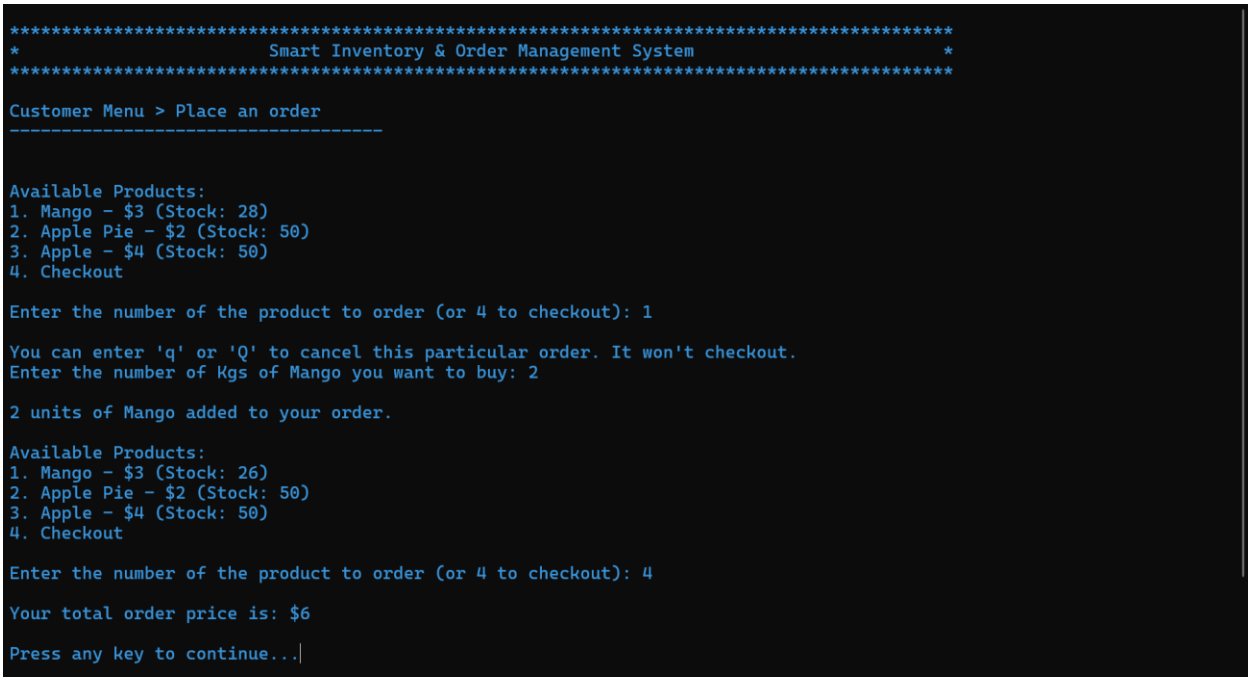


- **Delete Product:**



- **Sorting:**

- **Placing Order:**



📌 Limitations

1. **Limited User Roles**
   The system currently supports only basic users (e.g., Admin or Operator) without fine-grained permission levels.

2. **No GUI Interface**
   The application is console-based. This limits ease of use and may not be ideal for non-technical users.

3. **No Concurrent Access**
   The system does not support multiple users accessing or modifying data at the same time.

🚀 Future Work

1. **Role-Based Access Control**
   Implement multiple user roles with different permissions (e.g., Admin, Manager, Clerk).
2. **Graphical User Interface (GUI)**
   Develop a user-friendly interface using a GUI framework like Qt, WinForms, or a web-based UI.
3. **Inventory Analytics**
   Include sales and inventory reports, graphs, and low-stock alerts.
4. **Barcode Scanning Support**
   Add support for scanning product barcodes to automate item lookup and entry.
5. **Multilingual Support**
   Provide localization options for different languages.
6. **API Integration**
   Create a RESTful API for accessing inventory and order data from other apps or services.

Student Reg. No.: 2025(S)-CS-67 **Student Name:** Mirza Talal Ahmed

	A-Extensive Evidence	**B-Convincing Evidence**	**C-Limited Evidence**	**D-No Evidence**
Documentation Formatting **Grade:**				
Documentation Formatting Criteria: In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented				
Documentation Contents **Grade:**				
Documentation Contents Criteria: **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined - Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning.				
Project Complexity **Grade:**				
Code Style **Grade:**				
Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added.				
Code Documentation Mapping **Grade:**				
Data Structure (Arrays) **Grade:**				
Sorting Features **Grade:**				
Modularity **Grade:**				
Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)-Demo Data Functionality Added-At least Two Unit Tests are defined.				
Validations **Grade:**				
Recommendation Feature				
Presentation and Demo **Grade:**				
Student Understanding with the Code. **Grade:**				

Checked by: _____

Student Reg. No.: 2025(S)-CS-67 **Student Name:** Mirza Talal Ahmed

	A-Extensive Evidence	**B-Convincing Evidence**	**C-Limited Evidence**	**D-No Evidence**
Documentation Formatting **Grade:**				
Documentation Formatting Criteria: In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented				
Documentation Contents **Grade:**				
Documentation Contents Criteria: **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined - Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning.				
Project Complexity **Grade:**				
Code Style **Grade:**				
Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added.				
Code Documentation Mapping **Grade:**				
Data Structure (Arrays) **Grade:**				
Sorting Features **Grade:**				
Modularity **Grade:**				
Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- Demo Data Functionality Added-At least Two Unit Tests are defined.				
Validations **Grade:**				
Recommendation Feature				
Presentation and Demo **Grade:**				
Student Understanding with the Code. **Grade:**				

Checked by: _____