Development of Online Shopping System using C+

Faiza Islam Nesa

ID: 23303048

Department of Computer Science and Engineering
College of Engineering and Technology

IUBAT–International University of Business Agriculture and Technology

Fall 2024

Student’s Declaration

I hereby declare that the project report entitled “**Online Shopping System**” submitted by me is an original work. I affirm that there is no plagiarism, data falsification, or any unauthorized use of materials in this report. All the content and information derived from various sources have been appropriately cited and referenced.

.

Faiza Islam Nesa
Student ID :23303048

Abstract

This report describes the development of an Online Shopping System designed to make online shopping easier for both customers and sellers. The project solves common e-commerce problems by including features like browsing products, managing a shopping cart, placing orders, and making secure payments. Users can easily create accounts, look through product categories, and track their orders.

The system was built using C++ with a focus on making it easy to use and reliable. It securely stores data to protect user privacy and ensures all transactions are safe.

Acknowledgments

I would like to express my sincere gratitude to Nahif Reza, my senior, for his invaluable assistance in fixing errors and providing helpful suggestions on how to build this program effectively. His guidance played a crucial role in the successful completion of this project. I also extend my thanks to online platforms such as YouTube and Cplusplus.com for offering insightful resources and tutorials that supported me during the development process. Their collective contributions have been instrumental in the accomplishment of this project.

Table of Contents

List of Figures

Chapter 1 Introduction

The Online Shopping System is a project designed to simplify the shopping experience for users and streamline product management for sellers. It allows users to browse products, add items to a cart, and securely complete transactions. The system includes features like product search, order tracking, and personalized recommendations, making it convenient and user-friendly.

For sellers, the system manages inventory, tracks sales, and ensures that customer needs are met efficiently. Built using C++ with a focus on reliability and simplicity, this project provides a practical solution for small to medium-sized online businesses.

1.1 Problem Statement

In today's fast-paced world, traditional shopping methods can be time-consuming and inconvenient. People often struggle to find the right products, compare prices, or manage their purchases efficiently. On the other hand, sellers face challenges in maintaining stock, managing orders, and providing personalized services.

The Online Shopping System addresses these issues by offering an easy-to-use platform where users can shop effortlessly, and sellers can manage their products effectively.

1.2 Objectives

The objectives of this project are:

- To create a user-friendly platform for online shopping.
- To provide essential features like product browsing, cart management, and secure payment processing.
- To allow users to track their orders and receive personalized product recommendations.
- To ensure that inventory and order management are efficient for sellers.
- To implement error-checking mechanisms to enhance reliability and prevent common issues like stock mismanagement or duplicate orders.

Chapter 2 Project Implementation

2.1 Creating User Class

The Product class represents an individual product in the shopping system. Each product is characterized by four main attributes: id, name, price, and stock. These attributes define the essential properties of a product, such as its unique identification, name, price, and availability in stock.

2.2 Creating ShoppingCart Class

The ShoppingCart class manages the user's shopping cart. It contains an array of Product objects and their quantities, allowing the user to add, remove, and view products in the cart. It also tracks the total cost of the cart and manages the stock of products.

2.3 Creating OnlineShoppingSystem Class

The OnlineShoppingSystem class serves as the core of the application, managing the interaction between users and products. It offers functionalities for browsing products, searching for items, adding/removing products from the cart, processing payments, and recommending products.

2.3.1 Declared Functions

Key functions implemented to support system operations include:

- void addProduct(Product& product, int quantity)
- void removeProduct(int productId, Product products[], int productCount)
- void viewCart()
- int getTotal()
- void clearCart(Product products[], int productCount)
- void saveToFile(ofstream &file)

- void loadFromFile(string line)
- void save()
- void load()
- void loadData()
- void showProducts()
- void searchProduct(const string& query)
- void processPayment()
- void trackOrder()
- void recommendProducts()
- void run()

2.3.2 Main Function:

Main function loads the shopping cart data from the file and runs the online shopping system, allowing the user to interact with it.

```
int main()
        {
           load();
           OnlineShoppingSystem system;
           system.run();
           return 0;
         }
```

2.3.2 void addProduct(Product& product, int quantity)

Adds a product to the cart. If the product is already in the cart, it updates the quantity. It also checks if enough stock is available.

```
{
    if (product.stock < quantity) {
        cout << "Not enough stock available for " << product.name << ".\n";
        return;
    }

    for (int i = 0; i < itemCount; ++i)
    {
        if (cart[i].id == product.id)
        {
            quantities[i] += quantity;
            product.stock -= quantity;
            cout << quantity << " x " << product.name << " added to cart.\n";
            itemCount += 1;
            return;
        }
    }

    cart[itemCount] = product;
    quantities[itemCount] = quantity;
    product.stock -= quantity;
```

```
            itemCount+=1;
            cout << quantity << " x " << product.name << " added to cart.\n";
    }
```

2.3.3 void removeProduct(int productId, Product products[], int productCount)

Removes a product from the cart based on the product's ID and updates the stock**.**

```
        {
            for (int i = 0; i < itemCount; ++i)
            {
                if (cart[i].id == productId)
                {
                    for (int j = 0; j < productCount; ++j)
                    {
                        if (products[j].id == productId)
                        {
                            products[j].stock += quantities[i];
                            break;
                        }
                    }

                    for (int j = i; j < itemCount - 1; ++j)
                    {
```

```
                cart[j] = cart[j + 1];
                quantities[j] = quantities[j + 1];
            }

            --itemCount;
            cout << "Product removed from cart and stock updated.\n";
            return;
        }
    }

    cout << "Product not found in cart.\n";
}
```

2.3.4 void viewCart()

Displays all products in the cart, their quantities, and total price**.**

```
{
    if (itemCount == 0)
    {
        cout << "Your cart is empty.\n";
        return;
    }
```

```
            cout << "Shopping Cart:\n";
            double total = 0;

            for (int i = 0; i < itemCount; ++i)
            {
                cout << cart[i].name << " x " << quantities[i]
                     << " - $" << cart[i].price * quantities[i] << "\n";
                total += cart[i].price * quantities[i];
            }

            cout << "Total: $" << total << "\n";
    }
```

2.3.5 int getTotal()

Returns the total price of all products in the cart.

```
  {
            int total = 0;

            for (int i = 0; i < itemCount; ++i)
            {
                total += cart[i].price * quantities[i];
            }
```

```
        return total;
    }
```

2.3.6 void clearCart(Product products[], int productCount)

Clears all items from the cart and updates the stock.

```
{
        for (int i = 0; i < itemCount; ++i)
        {
            for (int j = 0; j < productCount; ++j)
            {
                if (products[j].id == cart[i].id)
                {
                    products[j].stock += quantities[i];
                    break;
                }
            }
        }

        itemCount = 0;
    }
```

2.3.7 void saveToFile(ofstream &file)

Saves the cart's contents (item count, total amount, and products) to a file.

```
{
        file << itemCount << " " << max(totalAmount, getTotal()) << " ";
        for(int i = 0;i < itemCount;i++)
        {
            file << cart[i].name << " " << quantities[i] << " ";
        }
        file << '\n';
  }
```

2.3.8 void loadFromFile(string line)

Loads cart contents from a saved file and updates the cart.

```
    {
        string words[1002];
        string word;
        for(int i = 0;i < 1000;i++)
        {
            words[i] = "NULL_VALUE";
        }
        istringstream iss(line);
        int i = 0, j = 0;
        while(iss >> word)
        {
            words[i] = word;
            i++;
        }
        itemCount = stoi(words[0]);

        totalAmount = stoi(words[1]);

        cout << totalAmount << endl;

        i = 2, j =0;

        while(words[i] != "NULL_VALUE")

        {

            cart[j].name = words[i];

            quantities[j] = stoi(words[i+1]);

            j++;

            i += 2;

        }


    }
```

2.3.9 void save()

Saves all shopping cart data to a file ("records.txt").

```
{

    ofstream file("records.txt");

    for(int i = 0;i <= current_carts;i++)

    {

        allShoppingCarts[i].saveToFile(file);

    }

    file.close();

}
```

2.3.10 void load()

Loads shopping cart data from the file and populates the allShoppingCarts array.

```
{
    ifstream file("records.txt");
    if(file.is_open())
    {
        string line;
        while(getline(file, line))
        {
            allShoppingCarts[current_carts].loadFromFile(line);
            current_carts += 1;
        }
    }
 }
```

2.3.11 void loadData()

Displays all previous transactions (saved shopping cart data).

```
int tr = 1;

    for(int i = 0;i < current_carts;i++)
    {
        cout << "\n\nTransaction " << tr << ": \n" << endl;
        cout << "Total: " << allShoppingCarts[i].totalAmount << endl;
        cout << "Total Items: " << allShoppingCarts[i].itemCount << endl;

        for(int j = 0;j < allShoppingCarts[i].itemCount;j++)
        {
            cout << allShoppingCarts[i].cart[j].name << "         " << allShoppingCarts[i].quantities[j] << endl;
        }

        tr++;
    }
```

2.3.12 void showProducts()

Displays all available products.

```
    {
        cout << "\nAvailable Products:\n";

        for (int i = 0; i < 5; ++i)
        {
            products[i].display();
        }
}
```

2.3.13 void searchProduct(const string& query)

Searches for a product by its name and displays it if found.

```
{
    cout << "\nSearch Results for '" << query << "':\n";

    for (int i = 0; i < 5; ++i)
    {
        if (products[i].name == query)
        {
            products[i].display();
        }
    }
}
```

2.3.14.void processPayment()

Processes the payment for the cart and saves the transaction.

```
{
    double total = cart.getTotal();

    if (total == 0)
    {
        cout << "Your cart is empty. Payment cannot be processed.\n";
        return;
    }
```

```
            cout << "\nProcessing payment of $" << total << "...\n";
            cout << "Payment successful!\n";
            allShoppingCarts[current_carts] = cart;
            save();
            current_carts += 1;
            cart.clearCart(products, 5);
 }
```

2.3.15 void trackOrder()

Displays the order status (Processing -> Shipped -> Delivered).

```
    {
            cout << "\nOrder Status: Processing -> Shipped -> Delivered\n";
    }
```

2.3.16 void recommendProducts()

Displays product recommendations based on stock availability.

```
    {
            cout << "\nPersonalized Recommendations:\n";
            for (int i = 0; i < 5; ++i)
            {
               if (products[i].stock > 0)
               {
                  products[i].display();
               }
            }
        }
```

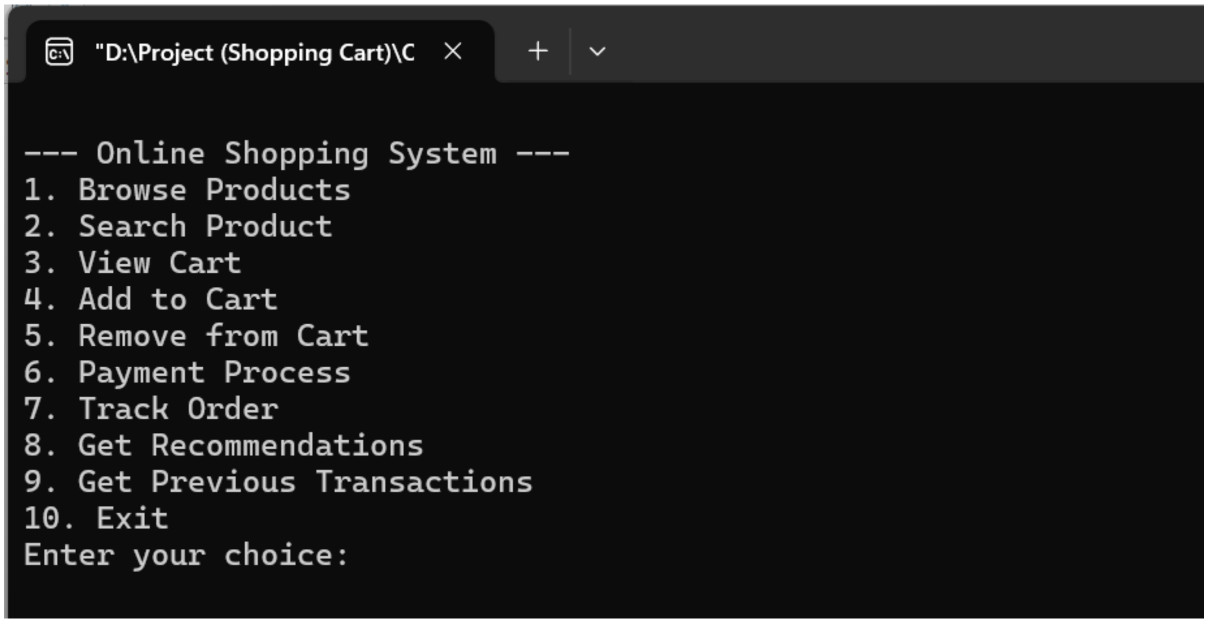
2.3.17 void run()

```
int choice;

    do {
        cout << "\n--- Online Shopping System ---\n";
        cout << "1. Browse Products\n";
        cout << "2. Search Product\n";
        cout << "3. View Cart\n";
        cout << "4. Add to Cart\n";
        cout << "5. Remove from Cart\n";
        cout << "6. Payment Process\n";
        cout << "7. Track Order\n";
        cout << "8. Get Recommendations\n";
        cout << "9. Get Previous Transactions\n";
        cout << "10. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice == 1) {
            showProducts();
        } else if (choice == 2) {
            string query;
            cout << "Enter product name to search: ";
            cin.ignore();
            getline(cin, query);
            searchProduct(query);
        } else if (choice == 3) {
            cart.viewCart();
        } else if (choice == 4) {
            int productId, quantity;
            showProducts();
            cout << "Enter product ID to add: ";
            cin >> productId;
            cout << "Enter quantity: ";
            cin >> quantity;
            cart.addProduct(products[productId - 1], quantity);
        } else if (choice == 5) {
            int productId;
            cout << "Enter product ID to remove: ";
            cin >> productId;
            cart.removeProduct(productId, products, 5);
        } else if (choice == 6) {
            processPayment();
        } else if (choice == 7) {
            trackOrder();
        } else if (choice == 8) {
            recommendProducts();
```

```
            } else if (choice == 9) {
                loadData();
            } else if(choice == 10) {
                cout << "Exiting the system. Thank you!\n";
                exit(0);
            } else {
                cout << "Invalid choice. Try again.\n";
            }
        } while (choice != 10);
}
```

Chapter 3 Testing the System

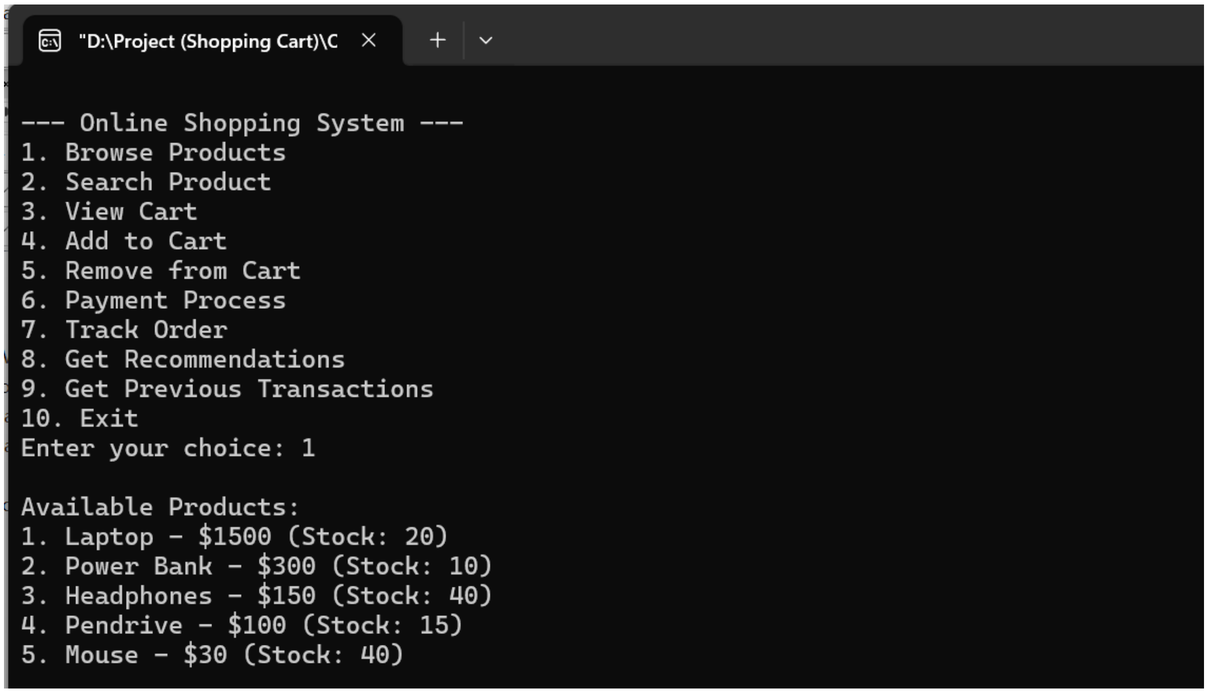
3.1 Login



```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice:
```

Figure 3.1 Data Screening

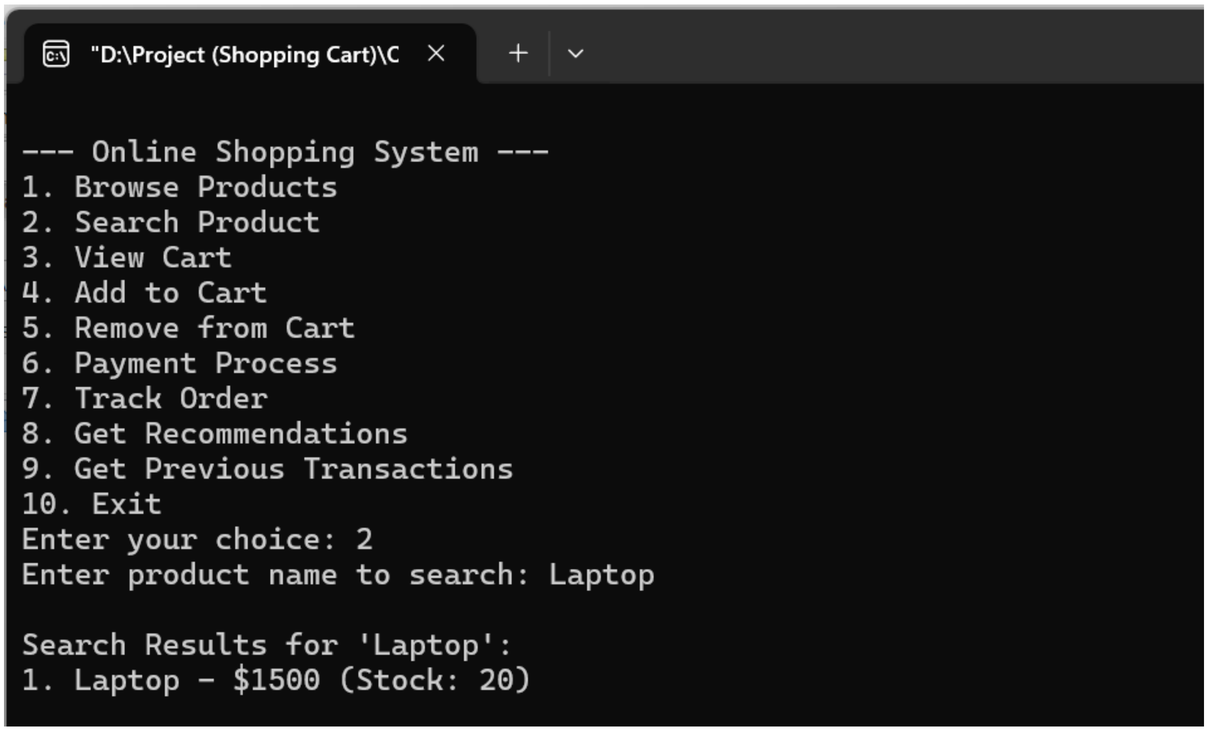
3.1.1 Browse Product



```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 1

Available Products:
1. Laptop - $1500 (Stock: 20)
2. Power Bank - $300 (Stock: 10)
3. Headphones - $150 (Stock: 40)
4. Pendrive - $100 (Stock: 15)
5. Mouse - $30 (Stock: 40)
```

Figure 3.1.1 Enter 1 for Browse Product

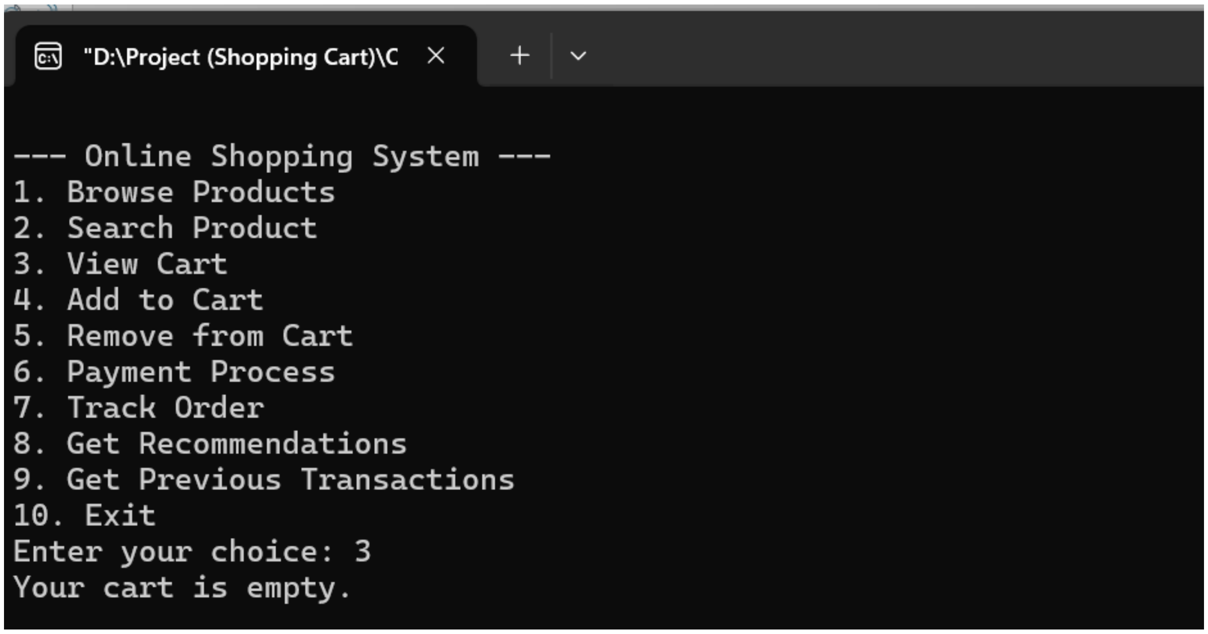
3.1.2 Search Product


"D:\Project (Shopping Cart)\C

```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 2
Enter product name to search: Laptop

Search Results for 'Laptop':
1. Laptop - $1500 (Stock: 20)
```

Figure 3.1.2 Enter 2 for Search product

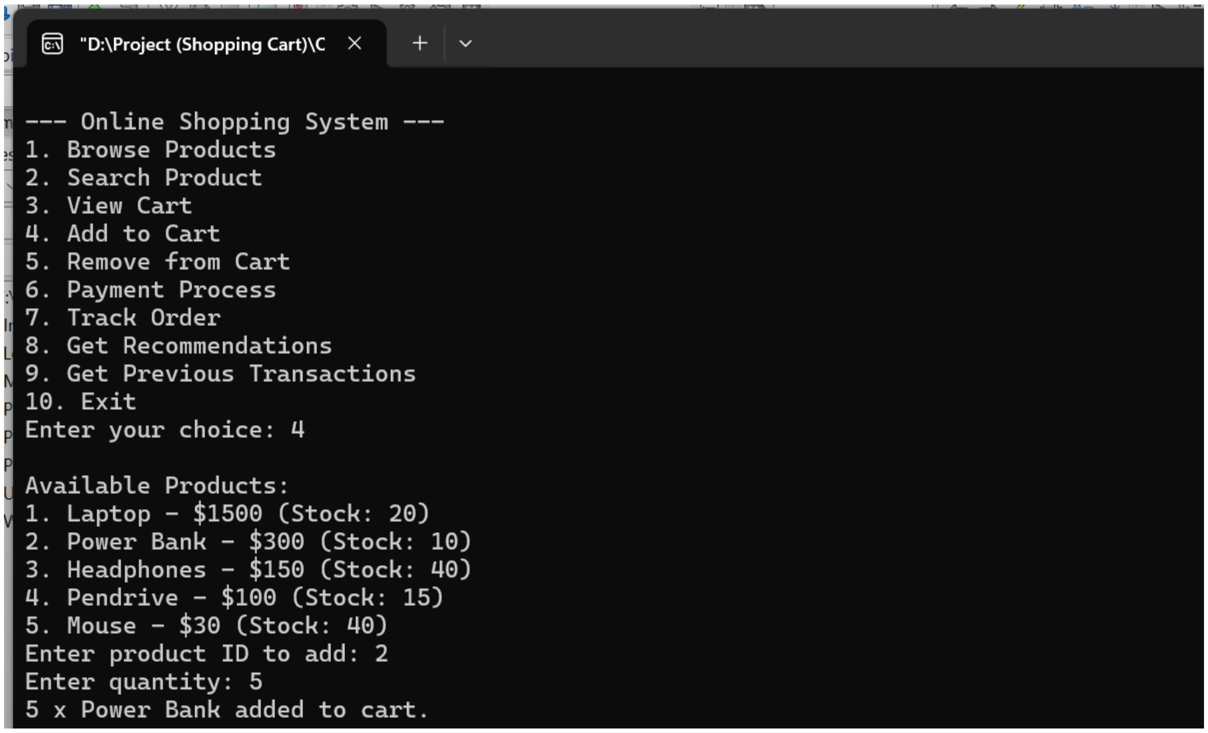
3.1.3 View Cart


"D:\Project (Shopping Cart)\C

```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 3
Your cart is empty.
```

Figure 3.1.3 Enter 3 for View Cart

3.1.4 Add to Cart

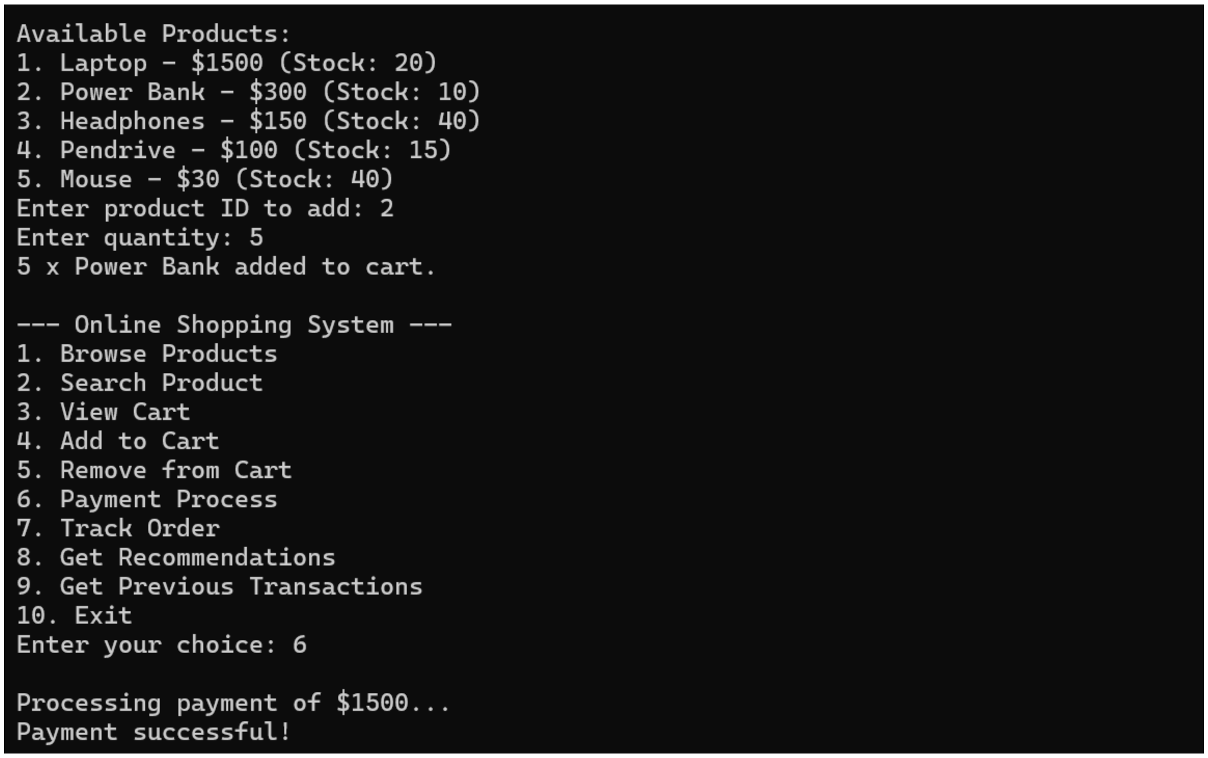


"D:\Project (Shopping Cart)\C

```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 4

Available Products:
1. Laptop - $1500 (Stock: 20)
2. Power Bank - $300 (Stock: 10)
3. Headphones - $150 (Stock: 40)
4. Pendrive - $100 (Stock: 15)
5. Mouse - $30 (Stock: 40)
Enter product ID to add: 2
Enter quantity: 5
5 x Power Bank added to cart.
```

Figure 3.1.4 Enter 4 for Add to Cart

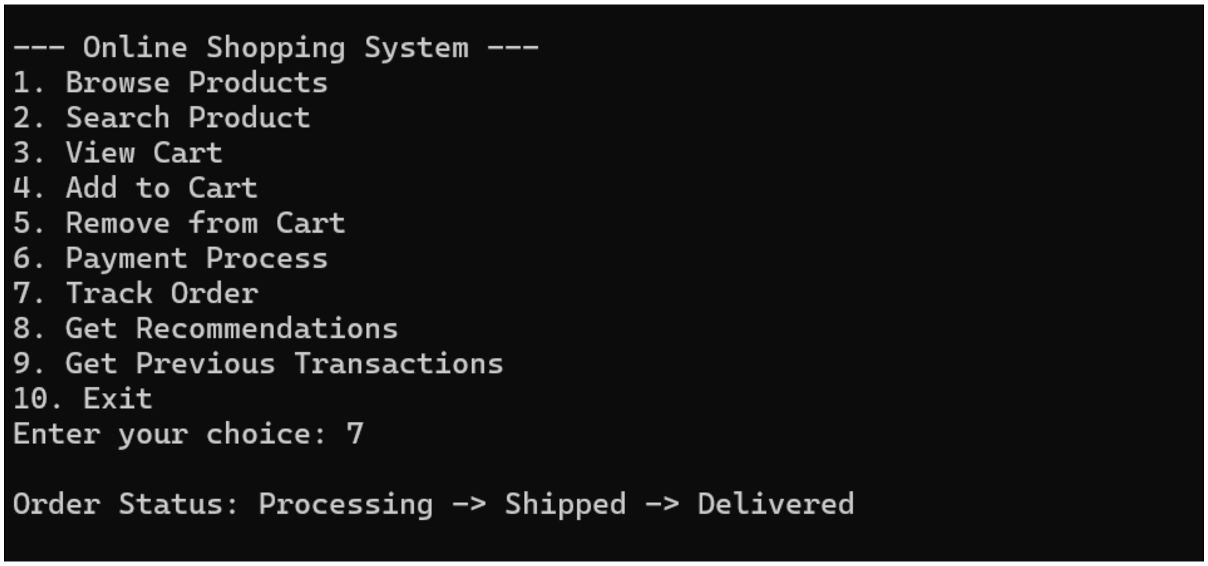
3.1.5 Payment Process



```
Available Products:
1. Laptop - $1500 (Stock: 20)
2. Power Bank - $300 (Stock: 10)
3. Headphones - $150 (Stock: 40)
4. Pendrive - $100 (Stock: 15)
5. Mouse - $30 (Stock: 40)
Enter product ID to add: 2
Enter quantity: 5
5 x Power Bank added to cart.

--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 6

Processing payment of $1500...
Payment successful!
```

Figure 3.1.5 Enter 6 for Payment Process

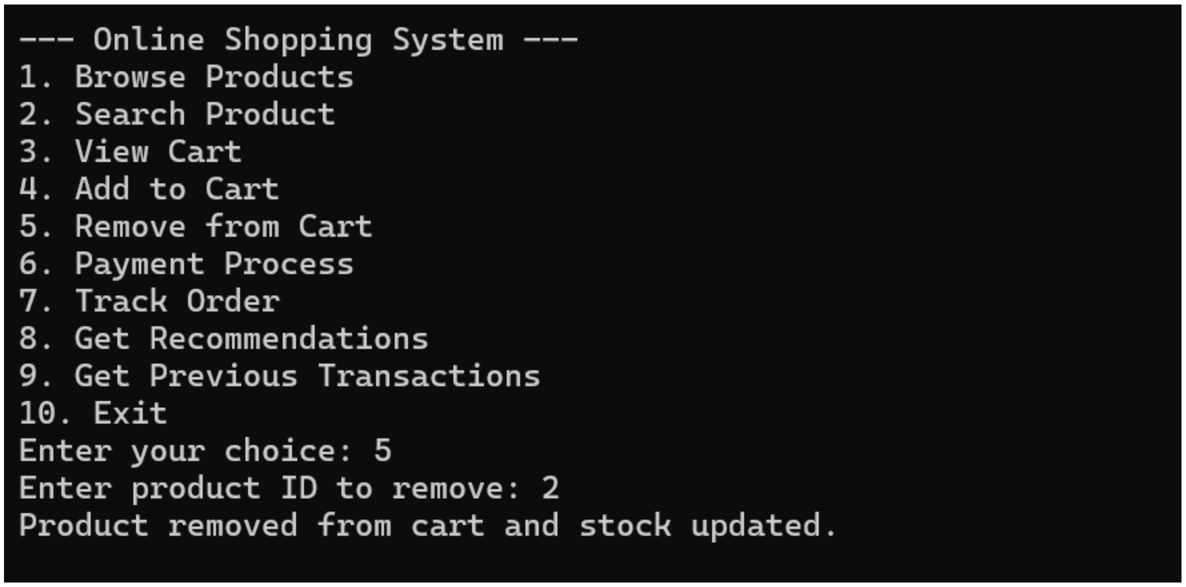
3.1.6 Track Order



```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 7

Order Status: Processing -> Shipped -> Delivered
```

Figure 3.1.6 Enter 7 for Tracking Order

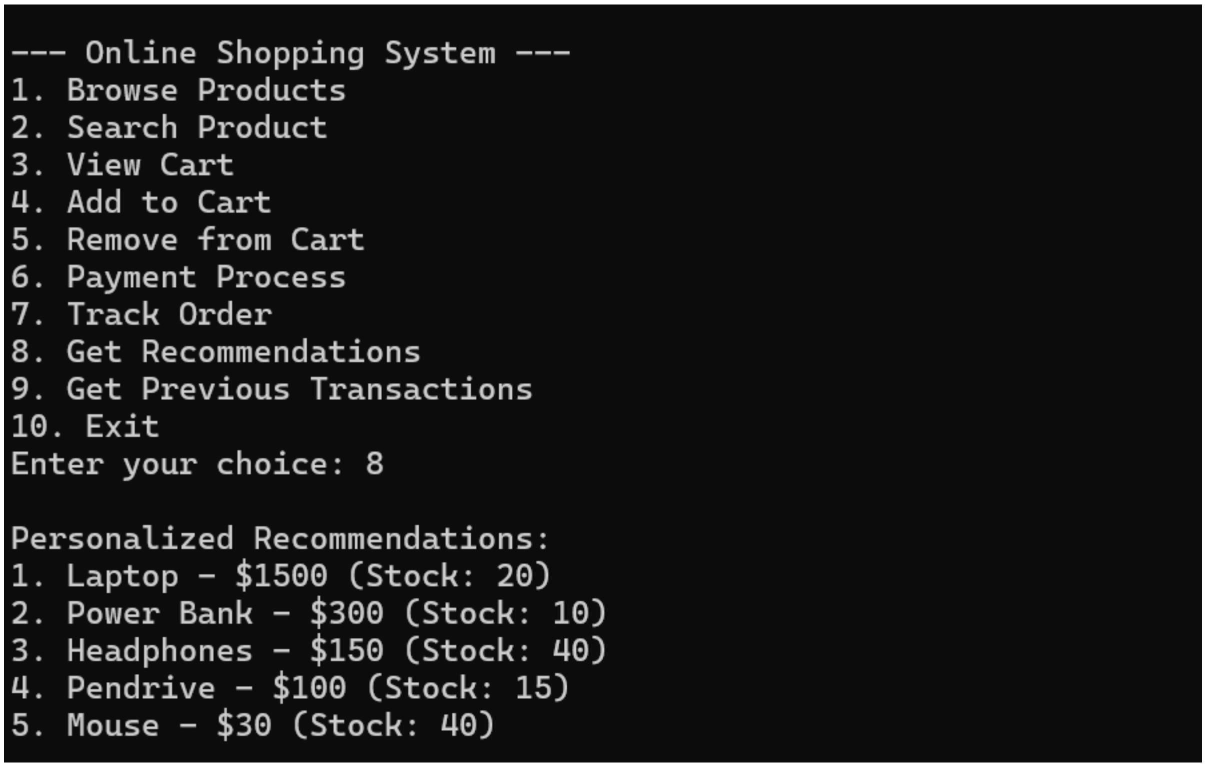
3.1.7 Remove from Cart



```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 5
Enter product ID to remove: 2
Product removed from cart and stock updated.
```

Figure 3.1.7 Enter 5 for Remove from Cart

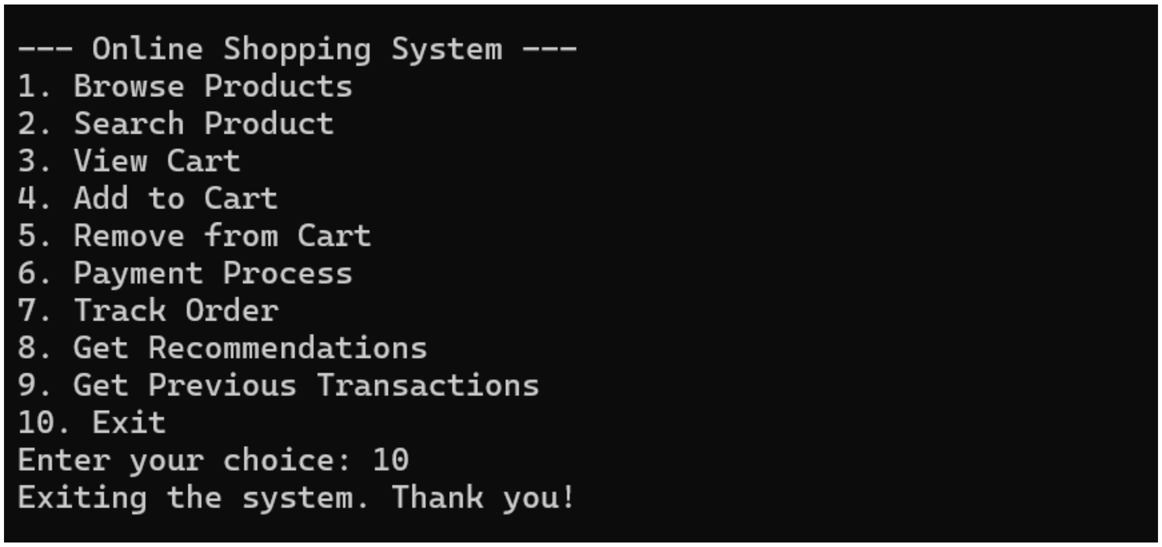
3.1.8 Get Recommendations



```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 8

Personalized Recommendations:
1. Laptop - $1500 (Stock: 20)
2. Power Bank - $300 (Stock: 10)
3. Headphones - $150 (Stock: 40)
4. Pendrive - $100 (Stock: 15)
5. Mouse - $30 (Stock: 40)
```

Figure 3.1.8 Enter 8 for Get Recommendations

3.1.9 Exit



```
--- Online Shopping System ---
1. Browse Products
2. Search Product
3. View Cart
4. Add to Cart
5. Remove from Cart
6. Payment Process
7. Track Order
8. Get Recommendations
9. Get Previous Transactions
10. Exit
Enter your choice: 10
Exiting the system. Thank you!
```

Figure 3.1.9 Enter 10 for Exit

Chapter 4 Conclusion

4.1 Challenges Faced

During the development of the Online Shopping System, I faced a few challenges but found solutions for each. At first, the program couldn't save or load cart data, which caused data loss when closed. I fixed this by learning file handling in C++ and using ofstream and ifstream to properly save and load data. Another issue was with managing stock when products were added or removed from the cart. I updated the code to make sure stock was always accurate. I had trouble managing multiple user carts, and the data would get mixed up. To fix this, I added a system to track multiple carts separately. With these fixes, the system became fully functional.

4.2 Future Recommendations

- User Accounts: Add login and signup options so users can save their order history and personal details.
- Product Categories: Group products into categories like "Electronics" or "Accessories" to make it easier to find things.
- Discounts: Add discounts or coupon codes to make shopping more exciting.
- Mobile App or Website: Make a mobile app or website so users can shop on their phones or computers.
- Reviews: Allow users to rate and review products so others can know what's good.

4.3 Conclusion

The Online Shopping System in C++ is a simple program that works like a basic online store. Users can browse and search for products, add or remove items from a shopping cart, and make payments using a simulated payment system. It also lets users track their orders and see product suggestions based on what is available. The system saves transaction history, which can be viewed later. This project uses important programming concepts like classes and file handling, making it a good starting point. With more features like user accounts and better recommendations, it could become more advanced.

References

• Balagurusamy, E. (2021). Object-Oriented Programming with C++.

• CodeWithHarry. (n.d.). File Handling in C++ [Video]. Retrieved from YouTube: https://www.youtube.com/

• Cplusplus.com.