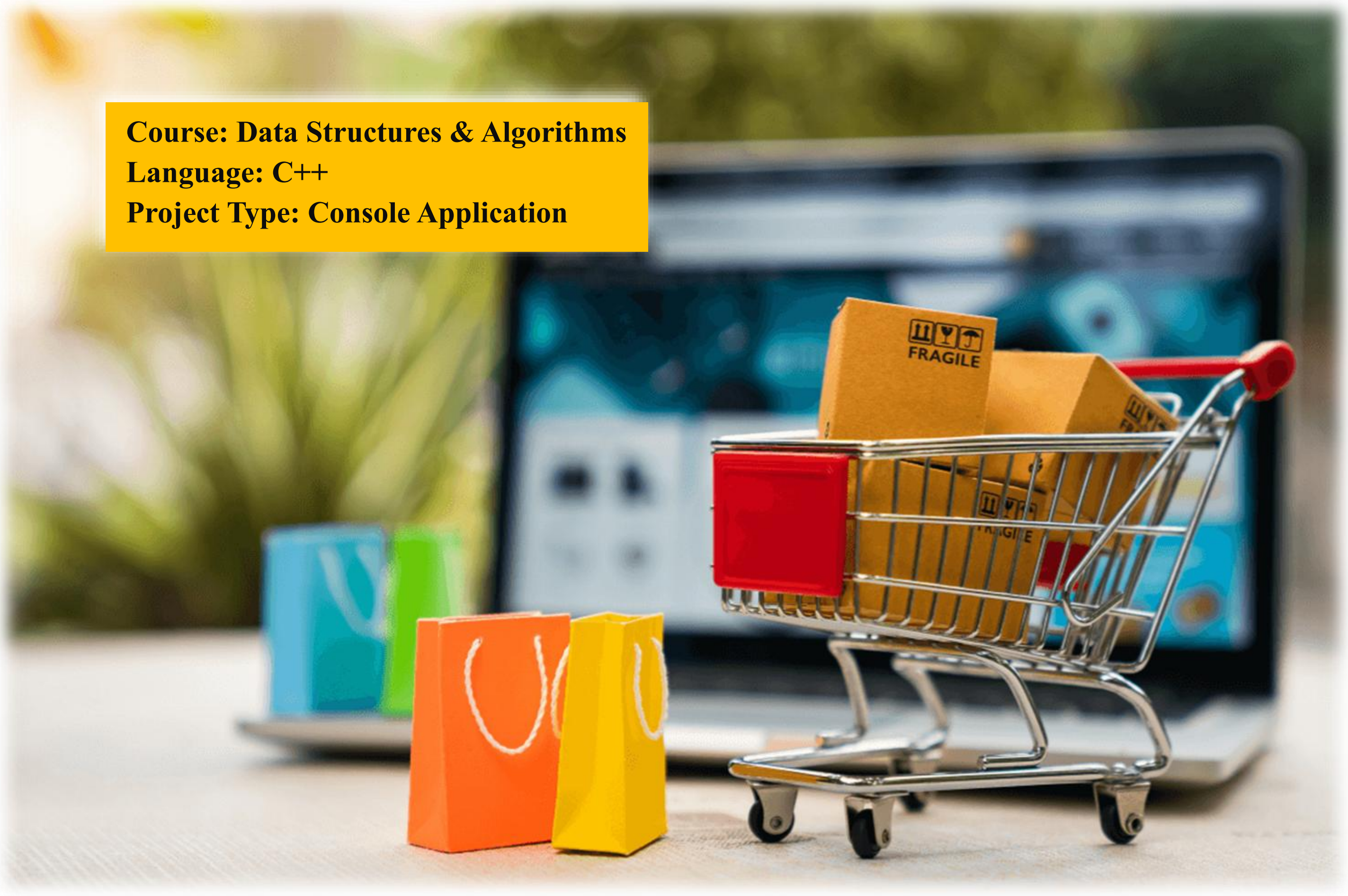
Online Marketplace Management System

Mohamed Khairy Eid Elzeblawy
Yusuf Ahmed Ibrahim Shoman
Mohamed Reda Ibrahim Ata



Course: Data Structures & Algorithms
Language: C++
Project Type: Console Application

Online Marketplace Management System

1. Overview

The Online Marketplace Management System is a simulation of an e-commerce platform. It facilitates the interaction between two types of users: Sellers and Customers. Sellers who add products, and Customers who browse, purchase, and rate products.

The objective of this project is to demonstrate the practical application of data structures--Vectors, Stacks, Queues, and Priority Queues--to solve real-world problems such as product management, shopping cart operations, and order processing.

2. System Architecture

The system is built using Object-Oriented Programming (OOP) principles. The core architecture consists of the following classes:

Class Overview

- Product: Represents an item for sale. Attributes: ID, name, price, category, stock quantity, sellerId, and rating statistics.
- Seller: Represents a vendor user. Attributes: ID, name, email.
- Customer: Represents a buyer. Attributes: ID, name, address, phone, email and unique stack-CartItem- to manage their shopping session.
- Marketplace: The controller class. It manages the main data containers (Vectors), handles File I/O (Persistence), and controls the application flow (Menus).

3. Data Structures & Algorithmic Analysis

This project utilizes specific data structures to optimize performance and functionality. Below is the justification and complexity analysis for each choice.

3.1. Vector (std::vector)

- Usage: Storage for the main databases (Sellers, Customers, Products).
- Justification: Vectors provide dynamic arrays that can resize automatically. They offer $O(1)$ random access, allowing the system to quickly retrieve data by index.
- Time Complexity:
    - Access by Index: $O(1)$
    - Insertion at End: $O(1)$
    - Search (Linear): $O(N)$

3.2. Stack (std::stack)

- Usage: Customer Shopping Cart.
- Justification: A shopping cart often requires an "Undo" feature. A Stack follows the LIFO (Last In First Out) principle, making it the perfect structure to implement "Remove Last Item" without needing to traverse a list.
- Time Complexity:
    - Push (Add Item): $O(1)$
    - Pop (Undo Item): $O(1)$

3.3. Queue (std::queue)

- Usage: Checkout Process.
- Justification: When processing an order, items should be handled in the order they were added to the cart, this simulates a cashier line. A Queue follows the FIFO (First In First Out) principle.
- Time Complexity:
  - Enqueue (Push): $O(1)$
  - Dequeue (Pop): $O(1)$

3.4. Priority Queue (std::priority_queue)

- Usage: "View products by rating" Feature.
- Justification: We need to efficiently display products with the highest ratings. A Priority Queue automatically orders elements so that the maximum element is always at the top.
- Time Complexity:
  - Insertion: $O(\log N)$
  - Access Top (Max Rating): $O(1)$

4. Key Algorithms & Logic Flows

4.1. The "Undo" Feature (Stack Logic)

When a customer adds an item, it is pushed onto their cartStack.

If the customer selects "Undo":

1. Check if cartStack is empty.
2. Perform cartStack.pop().
3. The item is instantly removed from the top.

4.2. The Checkout Process (Stack to Queue Transformation)

To process the order and generate a receipt:

1. Transfer: Elements are popped from the Cart (Stack) and pushed into a temporary Checkout Queue.
2. Process: The system loops while the Queue is not empty.
3. Validate: For each item, the system checks the main Product Vector to ensure Stock >= Quantity.
4. Update: Stock decreased, and Total Price is calculated.
5. Rate: The user is prompted to rate the product (1-5), updating the Product's average rating.

4.3. Data Persistence (File I/O)

To ensure data is not lost upon program exit, the system uses flat file storage:

- Loading: On startup, the Marketplace constructor reads sellers.txt, customers.txt, products.txt, and carts.txt.
- Saving: On exit and also after critical actions, the system writes the current state of all Vectors and Stacks back to these text files.

5. File Structure

The project relies on four text files for database management:

1. sellers.txt: Stores ID | Name | Email
2. customers.txt: Stores ID | Name | Address | Phone | Email
3. products.txt: Stores ID | Name | Price | Category | Stock | SellerID | RatingSum | RatingCount
4. carts.txt: Stores CustomerID | ProductID | Quantity

6. User Guide

For Sellers:

1. Register/Login: Create an account or log in with your email.
2. Add Product: Enter product details (Name, Category, Price, Stock).
3. Dashboard: View your session status.

For Customers:

1. Browse: Select "Browse All Products" to see items sorted by Highest Rating.
2. Filter: Select "Filter by Category" or "Filter by name" to view specific types of items.
3. Add to Cart: Enter the Product ID and Quantity.
4. Undo: Select "Undo Last Item" to remove the most recently added product.
5. Checkout: Finalize the purchase, view the receipt, and rate items.