Online Store Project Report

Aoun Abbas (24L-[redacted])
Instructor: Usman Anwar

May 13, 2025

Project Title: Online Store System in C++

Team Members:
Aoun Abbas (24L-[redacted])

Instructor:
Usman Anwar

Submission Date:
May 13, 2025

Abstract

This project implements an online store system using C++ and object-oriented programming (OOP) principles. The system allows users to browse product categories, add items to a cart, and proceed to checkout with delivery information, while administrators can manage products, view customer data, and handle feedback. Products and categories are persistently stored in a binary file, enhancing data retention across sessions. The purpose is to create an efficient, user-friendly shopping platform while demonstrating OOP concepts such as inheritance, polymorphism, composition, and aggregation. These principles enhance code modularity, reusability, and maintainability, making the system robust and scalable.

Contents

1 Introduction

1.1 Problem Statement

Traditional shopping requires physical visits to stores, which can be time-consuming and inconvenient. Existing online shopping systems may lack efficient product management, user-friendly interfaces, or persistent storage. This project addresses the need for a console-based online store that simplifies browsing, purchasing, and administration, with persistent data storage, while leveraging OOP for robust code design.

1.2 Objectives

- Develop a C++ console application for online shopping with user and admin functionalities.
- Implement a system to browse categories, manage carts, and save delivery information.
- Allow administrators to add, edit, or delete products and view customer data.
- Ensure persistent storage of products and categories using binary files.
- Apply OOP concepts to ensure modular, reusable, and maintainable code.

1.3 Motivations

The project is motivated by the growing demand for e-commerce solutions and the opportunity to apply OOP principles in a real-world context. It provides hands-on experience with C++ programming, file handling, system design, and data persistence, preparing for more complex software development tasks.

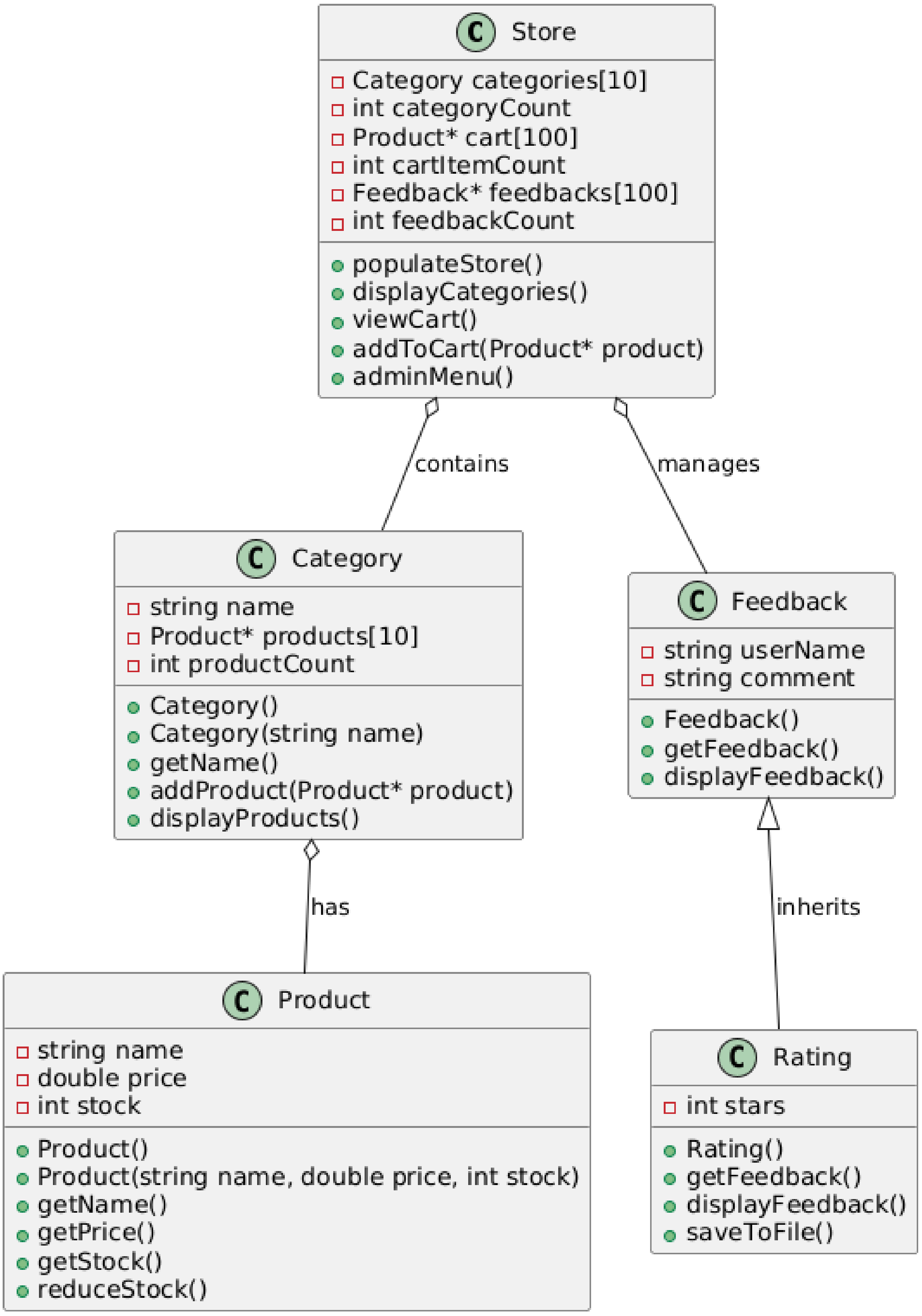
2 OOP Concepts Used

The online store system leverages several OOP principles to enhance code quality:

- **Inheritance**: The `Rating` class inherits from the `Feedback` class, extending its functionality to include star-based ratings. This promotes code reuse and allows specialized feedback handling.
- **Polymorphism**: Virtual functions (`getFeedback` and `displayFeedback`) in the `Feedback` class are overridden in `Rating`, enabling dynamic behavior based on the object type.
- **Composition**: The `Store` class contains arrays of `Category` objects, and each `Category` contains `Product` objects. This strong ownership ensures that categories and products are managed within the store's lifecycle.
- **Aggregation**: The `Category` class holds `Product` objects, but products can exist independently (e.g., in a cart). This loose coupling allows flexible product management across categories.

These concepts improve modularity, reduce code duplication, and make the system easier to extend or maintain.

3 UML Diagram



4 Test Cases

The following test cases verify the system's key functionalities, updated to reflect the new code features (e.g., persistent storage, sound effects):

1. **Browse Categories and Add to Cart**
   - *Input*: Choose option 1 (Browse Categories), select category 1 (Electronics), select product 1 (Smartphone), press 'Y' to add another, select product 2 (Laptop), press 'N' to return.
   - *Output*: Displays Electronics category, adds Smartphone (Rs. 20000) and Laptop (Rs. 50000) to cart with beep sounds on input, returns to category list.
   - *Expected Behavior*: Products are added to cart, cart item count increases, sound feedback provided.
2. **View Cart and Remove Item**
   - *Input*: Choose option 4 (View Cart), select 'Y' to remove item, enter item number 1.
   - *Output*: Displays cart with Smartphone and Laptop, total Rs. 70000. After removal, displays cart with Laptop only, total Rs. 50000.
   - *Expected Behavior*: Specified item is removed, total is updated.
3. **Checkout and Delivery Info**
   - *Input*: From cart, select 'Y' to checkout, enter mobile "03001234567", address "123 Main St", name "Aoun Abbas".
   - *Output*: Saves data to record.csv, displays order confirmation with total Rs. 50000 and delivery within 30 minutes.
   - *Expected Behavior*: Order is recorded, cart is cleared.
4. **Admin Login and Add Product with Persistence**
   - *Input*: Choose option 2 (Admin Login), enter password "oop", select option 1 (Add Product), enter category index 1, product name "Mouse", price 1000.
   - *Output*: Confirms product added successfully to Electronics category, saves to products.dat with beep sound on input.
   - *Expected Behavior*: New product is added to specified category and persisted to file.
5. **Submit Feedback with Rating**
   - *Input*: Choose option 5 (Rate Experience), enter name "Aoun", comment "Great store!", rating 4.
   - *Output*: Displays feedback: "Feedback by Aoun: Great store! Rating: **** (4 stars)", saves to ratings.txt with enhanced UI prompts.
   - *Expected Behavior*: Feedback is saved and displayed with improved user interaction.

6. **Load Products from File on Startup**
    - *Input*: Start the program after adding a product (e.g., Mouse in Electronics).
    - *Output*: Loads categories and products from products.dat, including the newly added Mouse.
    - *Expected Behavior*: Persistent storage ensures products are retained across sessions.

These test cases cover user shopping, cart management, checkout, admin tasks, feedback, and persistence, ensuring all features function as intended.

5 Future Directions

To enhance the online store system, the following features could be implemented:

- **Graphical User Interface (GUI)**: Integrate a GUI using Qt or SDL to replace the console interface, improving user experience.
- **Database Integration**: Use SQLite or MySQL to store products, categories, and customer data, replacing binary files for better scalability.
- **Online Payment Processing**: Add payment gateways (e.g., Stripe) to support secure transactions.
- **Product Search Enhancements**: Implement fuzzy search or filters (e.g., by price or rating) to improve product discovery.
- **User Accounts**: Allow users to create accounts, save cart history, and track orders.
- **Inventory Management**: Add stock tracking to prevent overselling and notify admins of low stock, aligning with potential UML diagram features.

These enhancements would make the system more robust, user-friendly, and suitable for real-world e-commerce applications.