E-Commerce Platform

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

M.Sree Bindhu | AP23110010536

J.Lakshmi Manasa | AP23110010570

Ch.Vyshnavi | AP23110010562

B.Bhavigna | AP23110010544

Under the Guidance of

Mrs.Kavitha Rani Karnena

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

November,2024

Certificate

Date: 20/11/2024

This is to certify that the work present in this Project entitled

"**E-Commerce Platform**" has been carried out by **[Name of the Candidate]** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Co-supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Acknowledgements

The development of this e-commerce platform was carried out using C++ and object-oriented programming (OOP) principles. The project serves as a foundation for a fully functional online store, with features such as user registration, login, product browsing, and shopping cart management. The system includes a `Product` class to handle various product types and a `User` class for managing customer accounts and transactions. The platform allows users to browse products across multiple categories, add items to their cart, and make purchases. Each product is associated with details such as name, brand, price, and category, and the system uses vectors to store and manage product data efficiently. I would like to express my sincere gratitude to my mentor and instructor for their invaluable guidance and support throughout the development of this project. Their expertise in C++ and object-oriented design was essential in shaping the structure and functionality of the platform. I would also like to thank the C++ community. A special thanks to my peers and classmates for their collaborative spirit and constructive feedback, which helped refine the functionality of the system. Finally, I would like to acknowledge the support of my family and friends, whose encouragement and understanding motivated me to complete this project.

Table of Contents

Abstract

This project involves the design and implementation of an e-commerce platform using C++ and object-oriented programming (OOP) principles. The objective was to develop a simple yet scalable system that simulates the core functionalities of an online store. The platform allows users to register, log in, browse products across different categories (electronics, fashion, home appliances, books, and sports), manage a shopping cart, and make purchases. A key focus was on creating a flexible and efficient structure, with the use of classes like `Product` and `User` to encapsulate product details and user data. The `Product` class includes attributes such as name, brand, and price, while the `User` class manages personal information, authentication, and transaction history. By utilizing dynamic memory management and C++ standard libraries, the platform efficiently handles user interactions and product catalog management. This project not only demonstrates the application of object-oriented design but also provides a foundation for building more advanced e-commerce systems in C++. The system is designed to be user-friendly and modular, offering insight into the integration of essential e-commerce features in a simplified, programmatic context.

Statement of Contributions

Bindhu: Helped in creating the user system, including registration and login, and made sure names, emails, and phone numbers were entered correctly.

Vyshnavi: Designed the product sections like Electronics, Fashion, Home Appliances, Books, and Sports and made sure the product details are shown properly.

Manasa: Worked on features like adding items to the cart, viewing past purchases, and giving feedback. Also helped with the main menu options.

Bhavigna: Checked the program for mistakes and made sure it works smoothly. Also ensured the program handles errors properly and tested everything to make it reliable

Abbreviations

OOP: Object-Oriented Programming

C++: C Plus Plus

UI: User Interface

STD: Standard

Introduction

The rapid growth of online shopping has made e-commerce platforms a crucial part of the modern retail landscape. This project aims to develop a simplified e-commerce platform using C++ and object-oriented programming (OOP) principles. The platform provides a range of core features typical of online stores, including user registration, product browsing, shopping cart management, and purchase functionality.

The system is built around key objects like `Product` and `User`, each representing essential entities in an e-commerce environment. Products are categorized into various types such as electronics, fashion, home appliances, books, and sports. Users can browse these categories, add items to their shopping cart, and make purchases. The project demonstrates the application of object-oriented concepts like inheritance, polymorphism, and encapsulation to create a modular and expandable system.

While the primary goal of this project is to simulate the basic operations of an e-commerce platform, it also serves as an educational tool for understanding how OOP can be applied to real-world software development. The platform is designed to be flexible and can easily be extended to include features such as payment processing, order tracking, and integration with a database in the future.

Methodology

The e-commerce platform was developed using an **Object-Oriented Programming (OOP)** approach, focusing on modular design, user-centric functionality, and scalability.

1. **Requirement Analysis**: The core requirements included user authentication, product browsing, cart management, and order history. Non-functional requirements like security and scalability were also prioritized.
2. **System Design**: A class-based design was used, with the `Product` class serving as a base for different product categories (e.g., `Electronics`, `Fashion`). The `User` class managed user-specific data like login credentials and shopping history. Data was stored in **vectors** during runtime for simplicity.
3. **Object-Oriented Principles**: **Inheritance** allowed for category-specific product types, **polymorphism** enabled dynamic behavior for the `display()` method, and **encapsulation** ensured secure handling of user data.
4. **Core Features**: The platform supports user registration and login, browsing products by category, adding items to the cart, and viewing purchase history. Users can either add products to their cart or purchase them directly.
5. **Testing and Debugging**: The system underwent unit, integration, and user acceptance testing to ensure correct functionality and smooth user interaction.
6. **Optimization**: Memory management was handled through dynamic allocation, and unused resources were cleaned up after execution. Code refactoring ensured clarity and maintainability.

In summary, the methodology focused on creating a flexible, maintainable, and scalable e-commerce platform using OOP principles, with careful attention to both functionality and user experience.

Implementation

```
#include <bits/stdc++.h>
using namespace std;
class Product {
public:
    string name;
    string brand;
    double price;
    Product(string name, string brand, double price)
        : name(name), brand(brand), price(price) {}
    virtual void display() const = 0;
};
class Electronics : public Product {
public:
    Electronics(string name, string brand, double price)
        : Product(name, brand, price) {}
    void display() const override {
        cout << "Electronics: " << name << " by " << brand << " for $" << price << endl;
    }
};
class Fashion : public Product {
public:
    Fashion(string name, string brand, double price)
        : Product(name, brand, price) {}
    void display() const override {
        cout << "Fashion: " << name << " by " << brand << " for $" << price << endl;
    }
};
class HomeAppliance : public Product {
```

```
public:
    HomeAppliance(string name, string brand, double price)
        : Product(name, brand, price) {}
    void display() const override {
        cout << "Home Appliance: " << name << " by " << brand << " for $" << price << endl;
    }
};
class Book : public Product {
public:
    Book(string name, string author, double price)
        : Product(name, author, price) {}
    void display() const override {
        cout << "Book: " << name << " by " << brand << " for $" << price << endl;
    }
};
class Sports : public Product {
public:
    Sports(string name, string brand, double price)
        : Product(name, brand, price) {}
    void display() const override {
        cout << "Sports: " << name << " by " << brand << " for $" << price << endl;
    }
};
class User {
private:
    string name;
    string email;
    string password;
    string phoneNumber;
```

```
    string address;
    vector<string> cart;
    vector<string> purchaseHistory;
public:
    User(string name, string email, string password, string phoneNumber, string address)
        : name(name), email(email), password(password), phoneNumber(phoneNumber), address(address) {}
    string getName() const { return name; }
    string getEmail() const { return email; }
    bool authenticate(const string &inputEmail, const string &inputPassword) const {
        return email == inputEmail && password == inputPassword;
    }
    void addToCart(const string &product) {
        cart.push_back(product);
        cout << product << " has been added to your cart." << endl;
    }
    void purchaseItem(const string &product) {
        purchaseHistory.push_back(product);
        cout << "Purchased: " << product << endl;
    }
    void displayCart() const {
        if (cart.empty()) {
            cout << "Your cart is empty." << endl;
        } else {
            cout << "Items in your cart:\n";
            for (const auto &item : cart) {
                cout << item << endl;
            }
        }
    }
```

```
    void displayPurchases() const {
        if (purchaseHistory.empty()) {
            cout << "You haven't made any purchases yet." << endl;
        } else {
            cout << "Your purchase history:\n";
            for (const auto &item : purchaseHistory) {
                cout << item << endl;
            }
        }
    }
};
User *loginUser(vector<User *> &users) {
    string email, password;
    cout << "Enter your email: ";
    cin >> email;
    cout << "Enter your password: ";
    cin >> password;
    for (auto user : users) {
        if (user->authenticate(email, password)) {
            cout << "Login successful! Welcome, " << user->getName() << "!" << endl;
            return user;
        }
    }
    cout << "Invalid email or password." << endl;
    return nullptr;
}
void registerNewUser(vector<User *> &users) {
    string name, email, password, phoneNumber, address;
```

```
    cout << "Enter your name: ";
    cin.ignore();
    getline(cin, name);
    cout << "Enter your email: ";
    cin >> email;
    cout << "Enter your password: ";
    cin >> password;
    cout << "Enter your phone number (10 digits): ";
    cin >> phoneNumber;
    cout << "Enter your address: ";
    cin.ignore();
    getline(cin, address);
    users.push_back(new User(name, email, password, phoneNumber, address));
    cout << "Registration successful!" << endl;
}
void displayCategories(User *loggedInUser, vector<Product *> &electronics, vector<Product *> &fashion,
                       vector<Product *> &homeAppliances, vector<Product *> &books, vector<Product *> &sports) {
    while (true) {
        cout << "\nSelect a category to browse:\n";
        cout << "1. Electronics\n2. Fashion\n3. Home Appliances\n4. Books\n5. Sports\n6. Exit\n";
        int choice;
        cin >> choice;
        vector<Product *> selectedCategory;
        switch (choice) {
        case 1: selectedCategory = electronics; break;
        case 2: selectedCategory = fashion; break;
        case 3: selectedCategory = homeAppliances; break;
        case 4: selectedCategory = books; break;
        case 5: selectedCategory = sports; break;
```

```
case 6: return;
default:
    cout << "Invalid choice, try again." << endl;
    continue;
}
cout << "\nProducts available:\n";
for (size_t i = 0; i < selectedCategory.size(); ++i) {
    cout << i + 1 << ". ";
    selectedCategory[i]->display();
}
int productChoice;
cout << "Select a product to view details: ";
cin >> productChoice;
if (productChoice < 1 || productChoice > selectedCategory.size()) {
    cout << "Invalid choice, returning to category selection." << endl;
    continue;
}
Product *selectedProduct = selectedCategory[productChoice - 1];
cout << "\nWhat would you like to do with this product?\n";
cout << "1. Add to Cart\n2. Buy Now\n3. Return to Categories\n";
int action;
cin >> action;
switch (action) {
case 1:
    loggedInUser->addToCart(selectedProduct->name);
    break;
case 2:
    loggedInUser->purchaseItem(selectedProduct->name);
    break;
```

```
        case 3:
            continue; // Return to categories
        default:
            cout << "Invalid choice, returning to categories." << endl;
        }
    }
}
int main() {
    vector<User *> users;
    vector<Product *> electronics = {
        new Electronics("Smartphone", "Brand A", 699.99),
        new Electronics("Laptop", "Brand B", 1299.99),
        new Electronics("Smart TV", "Brand C", 799.99),
        new Electronics("Tablet", "Brand D", 399.99),
        new Electronics("Smartwatch", "Brand E", 249.99)
    };
    vector<Product *> fashion = {
        new Fashion("T-Shirt", "Brand D", 19.99),
        new Fashion("Jeans", "Brand E", 39.99),
        new Fashion("Jacket", "Brand F", 89.99),
        new Fashion("Sneakers", "Brand G", 59.99),
        new Fashion("Dress", "Brand H", 49.99)
    };
    vector<Product *> homeAppliances = {
        new HomeAppliance("Refrigerator", "Brand F", 499.99),
        new HomeAppliance("Washing Machine", "Brand G", 349.99),
        new HomeAppliance("Microwave", "Brand H", 99.99),
        new HomeAppliance("Vacuum Cleaner", "Brand I", 79.99),
        new HomeAppliance("Air Conditioner", "Brand J", 499.99)
```

```
};
vector<Product *> books = {
    new Book("C++ Programming", "Author A", 29.99),
    new Book("Design Patterns", "Author B", 24.99),
    new Book("The Great Gatsby", "F. Scott Fitzgerald", 10.99),
    new Book("To Kill a Mockingbird", "Harper Lee", 12.99),
    new Book("1984", "George Orwell", 14.99)
};
vector<Product *> sports = {
    new Sports("Basketball", "Brand H", 24.99),
    new Sports("Tennis Racket", "Brand I", 49.99),
    new Sports("Football", "Brand J", 29.99),
    new Sports("Baseball Bat", "Brand K", 39.99),
    new Sports("Soccer Ball", "Brand L", 19.99)
};
cout << "\tWELCOME TO OUR ECOMMERCE PLATFORM\t" << endl;
while (true) {
    cout << "Are you an existing user? (1 for Yes, 2 for No, 3 to Exit): ";
    int choice;
    cin >> choice;
    User *loggedInUser = nullptr;
    if (choice == 1) {
        loggedInUser = loginUser(users);
        if (!loggedInUser) continue;
    } else if (choice == 2) {
        registerNewUser(users);
        loggedInUser = users.back();
    } else if (choice == 3) {
        break;
```

```
        }
        while (loggedInUser) {
            cout << "\nWhat would you like to do?\n";
            cout << "1. Browse Items\n2. View Cart\n3. View Purchases\n4. Logout\n";
            int action;
            cin >> action;
            if (action == 1) {
                displayCategories(loggedInUser, electronics, fashion, homeAppliances, books, sports);
            } else if (action == 2) {
                loggedInUser->displayCart();
            } else if (action == 3) {
                loggedInUser->displayPurchases();
            } else if (action == 4) {
                cout << "Logging out." << endl;
                loggedInUser = nullptr;
            } else {
                cout << "Invalid choice, try again." << endl;
            }
        }
    }
    for (auto user : users) delete user;
    for (auto product : electronics) delete product;
    for (auto product : fashion) delete product;
    for (auto product : homeAppliances) delete product;
    for (auto product : books) delete product;
    for (auto product : sports) delete product;
    return 0;
}
```

Output: WELCOME TO OUR ECOMMERCE PLATFORM

Are you an existing user? (1 for Yes, 2 for No, 3 to Exit): 2

Enter your name: Bhavigna

Enter your email: Bhavi@gmail.com

Enter your password: Bhavi

Enter your phone number (10 digits): 1665489023

Enter your address: 1-3h/4

Registration successful!

What would you like to do?

1. Browse Items

2. View Cart

3. View Purchases

4. Logout

1

Select a category to browse:

1. Electronics

2. Fashion

3. Home Appliances

4. Books

5. Sports

6. Exit

2

Products available:

1. Fashion: T-Shirt by Brand D for $19.99

2. Fashion: Jeans by Brand E for $39.99

3. Fashion: Jacket by Brand F for $89.99

4. Fashion: Sneakers by Brand G for $59.99

5. Fashion: Dress by Brand H for $49.99

Select a product to view details: 4

What would you like to do with this product?

1. Add to Cart

2. Buy Now

3. Return to Categories

1

Sneakers has been added to your cart.

Select a category to browse:

1. Electronics

2. Fashion

3. Home Appliances

4. Books

5. Sports

6. Exit

5

Products available:

1. Sports: Basketball by Brand H for $24.99

2. Sports: Tennis Racket by Brand I for $49.99

3. Sports: Football by Brand J for $29.99

4. Sports: Baseball Bat by Brand K for $39.99

5. Sports: Soccer Ball by Brand L for $19.99

Select a product to view details: 4

What would you like to do with this product?

1. Add to Cart

2. Buy Now

3. Return to Categories

2

Purchased: Baseball Bat

Select a category to browse:

1. Electronics

2. Fashion

3. Home Appliances

4. Books

5. Sports

6. Exit

6

What would you like to do?

1. Browse Items

2. View Cart

3. View Purchases

4. Logout

2

Items in your cart:

Sneakers

What would you like to do?

1. Browse Items

2. View Cart

3. View Purchases

4. Logout

3

Your purchase history:

Baseball Bat

What would you like to do?

1. Browse Items

2. View Cart

3. View Purchases

4. Logout

4

Logging out.

Are you an existing user? (1 for Yes, 2 for No, 3 to Exit):

Discussion

This e-commerce platform project showcases the implementation of a simple online shopping system using **Object-Oriented Programming (OOP)** principles. The system is designed around core components like `Product`, `User`, and their interactions, offering essential features such as product browsing, cart management, and purchase history tracking.

The **main() function** in this e-commerce platform is responsible for managing the user experience and the flow of interactions within the system. It handles user login, product browsing, adding items to the cart, making purchases, and logging out. Here's an in-depth look at how the main function works:

Program Setup:

At the beginning of the main function:

- **Product and User Data**: The program initializes predefined data for various product categories (electronics, fashion, home appliances, books, sports). These products are instantiated as objects of the `Product` class and its derived classes (`Electronics`, `Fashion`, etc.).
- **User Collection**: A vector `users` is used to store the list of registered users. The platform can handle multiple users at once, even though the main focus is on one user interacting with the system at a time.
- **Welcome Message**: A welcome message is displayed to greet the user upon starting the program:
  `"\tWELCOME TO OUR ECOMMERCE PLATFORM\t"`

User Interaction Flow:

1. **Main Loop:**

   The program enters an infinite `while (true)` loop, providing continuous interaction until the user explicitly decides to exit the platform (by choosing option 3).
2. User Choice - Login, Register, or Exit: The program prompts the user with the following options:
   - Option 1: Existing User
     If the user selects option 1, they are prompted to enter their email and password. The program then calls the `loginUser()` function, which attempts to authenticate the user by matching the entered credentials with the stored user data. If authentication is successful, the user is logged in.
   - Option 2: New User
     If the user is new, option 2 allows them to register. The `registerNewUser()` function collects the necessary personal details (name, email, password, phone number, and address) and creates a new `User` object, which is added to the `users` vector. After registration, the newly registered user is automatically logged in.
   - Option 3: Exit
     Selecting this option breaks the loop and ends the program.
3. User Menu After Successful Login: After a user is logged in (either by selecting option 1 or 2), the program presents a menu with several actions:
   - Option 1: Browse Items
     The user is presented with categories of products (electronics, fashion, home appliances, books, sports). They can select a category, and the program displays the products available in that category. The user can then:
     - View Product Details: The program shows more details about the selected product, such as its name, brand, and price.
     - Add to Cart: If the user wishes to purchase the item later, they can add it to their cart by choosing this option.

        - Buy Now: If the user wants to purchase the product immediately, the system marks the product as purchased and adds it to their purchase history.
    - Option 2: View Cart
      The user can check the contents of their shopping cart, where all selected items are listed. If the cart is empty, the user is informed. Otherwise, they can review the products added.
    - Option 3: View Purchases
      The user can view a list of their past purchases, which are stored in the `purchaseHistory` vector for each user. This option gives users a sense of their past shopping activity.
    - Option 4: Logout
      The user can log out, and the program returns to the main menu where they can choose to log in again, register, or exit. This option resets the current session.
4. Interaction with Categories and Products:
    - When browsing items (Option 1), the program calls the `displayCategories()` function, which shows the available products in the selected category. Each product's details (name, brand, price) are displayed.
    - The user is prompted to select a product by number, after which they can choose to either Add to Cart or Buy Now.
    - The user is provided with feedback after adding an item to the cart or completing a purchase.
5. Invalid Input Handling:
    - The program ensures that only valid choices are accepted from the user. If the user provides an invalid input (for example, selecting an option outside the available range), the system prints an error message and prompts the user to try again.
    - This process ensures a smooth user experience and prevents the program from crashing or behaving unexpectedly.

Additional Details:

- **Authentication and User Data:**

    - The user data (name, email, password, etc.) is stored in memory during the session. The `authenticate()` function checks whether the entered credentials match those of any registered user.
    - After successful login or registration, the `User` object stores important information like the user's name, email, cart items, and purchase history.
- Product Management:
    - Products are stored in vectors corresponding to their categories, and each product is an object derived from the `Product` class (e.g., `Electronics`, `Fashion`, etc.). The `display()` function, overridden in each subclass, ensures that the correct product details are shown to the user based on the category they select.
    - The `addToCart()` method in the `User` class handles adding selected products to the user's shopping cart, and the `purchaseItem()` method records a purchased product in the user's purchase history.
- Memory Management:
    - The `main()` function ensures proper memory management by deleting dynamically allocated memory at the end of the program using the `delete` operator. This prevents memory leaks by ensuring that objects like `User` and `Product` are properly deallocated.

Exit Process:

Once the user selects the "Exit" option from the main menu (Option 3), the program breaks out of the main loop and proceeds to clean up dynamically allocated memory. This includes deleting the `User` objects and `Product` objects created during the session to free up resources. The program then terminates gracefully

Concluding Remarks

This e-commerce project successfully simulates a basic online shopping platform, incorporating essential features such as user registration, login authentication, product browsing, cart management, and purchase history tracking. The project is built using object-oriented programming (OOP) principles, ensuring the system is modular, extensible, and maintainable. The use of abstract and derived classes for product categories like Electronics, Fashion, and Home Appliances enhances the flexibility of the system, allowing for easy extension of product types without changing the core architecture.

Scope for Improvement

While the project meets its core objectives, there are several areas for enhancement:

- **Payment System Integration**: Integrating a real payment gateway would allow users to complete transactions, adding authenticity to the platform.
- **Advanced Product Features**: Implementing search, filtering, and sorting would enhance usability and help users find products more efficiently.
- **Security Features**: Introducing password hashing, data encryption, and session management would improve security and protect user information.
- **Mobile or Web UI**: Developing a responsive UI or mobile app would make the platform more accessible and provide a better user experience across devices.

Future Work

The project lays a solid foundation for a basic e-commerce platform, but several areas offer opportunities for further development and enhancement. In the **future work**, the following improvements could be explored:

- **Payment Gateway Integration**: Implementing real payment gateways such as PayPal, Stripe, or credit card processing would allow users to complete secure transactions, transforming the platform into a fully operational online store.
- **User Personalization**: Adding features like personalized recommendations based on browsing history or purchase behavior would improve user engagement and satisfaction.
- **Product Review and Rating System**: Introducing user-generated reviews and ratings for products would enhance trust and provide valuable feedback for potential buyers.
- **Multi-Language and Currency Support**: Expanding the platform to support multiple languages and currencies would allow it to cater to a global audience, broadening its market reach.
- **Mobile Application**: Developing a mobile app for iOS and Android would provide a more accessible and convenient user experience, especially for users on the go.
- **Admin Panel for Product Management**: Implementing an admin dashboard for managing products, orders, and user accounts would streamline platform administration and ease management tasks.

These enhancements would not only expand the platform's functionality but also provide a more user-centric, secure, and scalable e-commerce solution suitable for real-world use.

References

1. **Bjarne Stroustrup**, *The C++ Programming Language*, 4th Edition, Addison-Wesley, 2013.
    - This book was the primary reference for the C++ programming concepts and object-oriented design principles applied throughout the project.
2. **Google** (https://www.google.com/)
    - Used extensively for researching C++ functions, debugging issues, and exploring various programming-related resources to implement features like user authentication, cart management, and product display in the e-commerce platform