ONLINE SHOPPING SYSTEM

A FINAL PROJECT REPORT SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR COURSE STAT295 – OBJECT ORIENTED PROGRAMMING

DEPARTMENT OF STATISTICS OF

MIDDLE EAST TECHNICAL UNIVERSITY

BY

FİTNAT KOÇ 2450542

MERVE AYDEMİR 2561058

NURAY TAGHIYEVA 2603264

SEZİN TARLIĞ 2561520

SİNEM SARICA 2552347

TABLE OF CONTENT

1. Abstract

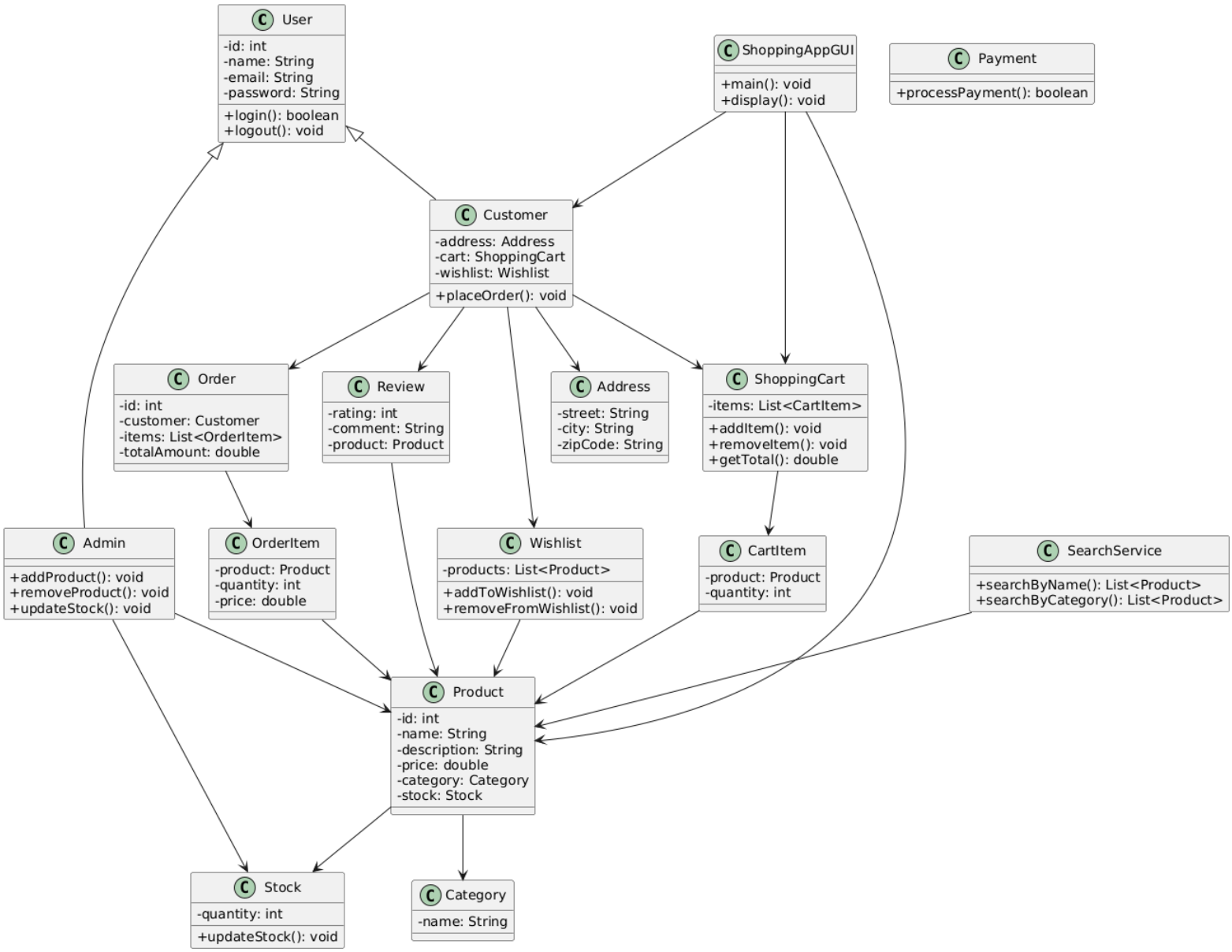
Online shopping has become an integral part of modern consumer behavior, offering convenience, speed, and accessibility that traditional retail often cannot match. As digital marketplaces continue to grow, understanding the structure and functionality of these systems has become increasingly relevant. This project focuses on the development of a simplified online shopping system that models the basic structure and functionality of modern e-commerce platforms. It aims to simulate common features found in such systems, including product browsing, shopping cart management, order placement, and administrative control over inventory. The project is intended to provide a clear understanding of how digital shopping environments operate and to explore the essential components involved in building and maintaining such systems. It serves as a foundational exercise in designing structured and user-oriented online applications.

2. Introduction

The purpose of this project is to design and implement a comprehensive Online Shopping System that effectively demonstrates the practical application of Object-Oriented Programming (OOP) principles using the Java programming language. Developed within the scope of the STAT295 course, the system aims to provide a realistic and structured simulation of an online shopping platform, offering essential functionalities commonly found in real-world applications. Customers using the system are able to browse through a collection of products, add selected items to their shopping carts, manage wishlists, submit product reviews, and finalize their orders through a simplified payment mechanism. On the administrative side, the system allows authorized users to perform operations such as adding new products to the catalog, modifying or removing existing ones, and managing stock levels to ensure inventory accuracy. These operations are supported by a modular class design that reflects a clear separation of responsibilities, with each class focused on a specific role or function within the system. The project also places strong emphasis on the four foundational pillars of OOP—Encapsulation, Inheritance, Polymorphism, and Abstraction—which are integrated throughout the codebase in a way that enhances reusability, scalability, and maintainability. While the project does not include a full-fledged graphical user interface, the underlying architecture has been developed in a way that supports future expansion, making it

compatible with database integration and potential third-party service connections. Overall, this project not only fulfills the academic requirements of the course but also offers a valuable learning experience in software design and collaborative development.

3. UML Class Diagram



The UML class diagram presented above illustrates the structural design of the Online Shopping System, showing the main components, their internal structure, and how they interact with one another. The architecture is centered around several modules, including user management, product and inventory handling, shopping and order processing, and additional supporting features such as search, payment, and system interaction via a basic interface. At the foundation of the user hierarchy is the abstract User class, which defines shared attributes such as id, name, email, and password, along with common methods like login() and logout(). Two specialized user types extend from this base: Customer and Admin. The Customer class is designed to represent the end-user experience and maintains relationships with Address, ShoppingCart, and Wishlist objects. It includes the placeOrder() method, enabling customers to complete their purchases. The Admin class encapsulates management functionalities such as adding or removing products and updating stock levels, reflecting administrative access in the system. The system's product management is built around the Product class, which includes key attributes like name, description, price, and references to both a Category and a Stock classes. The Category class represents product classification, while the Stock class tracks the current inventory quantity and includes a method to update stock levels. These classes together support control over the product catalog. Customer interaction with products is managed through components such as the ShoppingCart, CartItem, Order, and OrderItem classes. Products added to a cart are stored as CartItem objects within a ShoppingCart, which provides methods to add, remove, and calculate the total value of items. When a customer completes a transaction, an Order is created containing multiple OrderItem instances, each recording the purchased product, quantity, and price at the time of checkout. The system includes a Wishlist class, which allows customers to save products for later consideration, and a Review class, where customers can rate and comment on products they have purchased. The Address class captures delivery-related information, including street, city, and zip code details, and is associated directly with each customer. For product discovery, the SearchService class provides methods to retrieve products based on their name or category, improving user navigation and catalog accessibility. The Payment class includes a basic method processPayment() to simulate the payment operation. Lastly, the ShoppingAppGUI class represents the main interaction point for

users and acts as the front-facing interface of the application. It includes methods such as main() and display() to initialize and manage basic user interactions via a console or simulated interface. Overall, the UML class diagram demonstrates a well-organized and modular system structure. It clearly defines relationships through inheritance, aggregation, and association.

4. Methodology

4.1 Classes

The development of the Online Shopping System was based on designing and implementing a set of classes that represent the main features of a typical online shopping platform. The system starts with the User class, which defines basic information and login functionality for all users. This class is extended by two roles: Customer and Admin, each responsible for different tasks. The Customer class allows regular users to use features such as managing their address (Address class), keeping a shopping cart (ShoppingCart), and saving products to a wishlist (Wishlist). It also includes a method called placeOrder() that lets the customer complete a purchase. The Admin class is responsible for managing the platform, with methods to add, update, or remove products and manage stock levels. To handle products, the system uses the Product, Category, and Stock classes. Each product includes details like name, description, and price, and is linked to a category for classification and a stock object to track availability. This setup helps manage products clearly and update inventory when needed. When customers want to buy something, they add items to their cart using the ShoppingCart and CartItem classes. Once they place an order, the system creates an Order made up of OrderItems, which store details like the product, quantity, and price at that time. This makes it possible to keep an accurate record of past purchases. To improve the overall experience, the system includes a Review class so customers can give feedback and rate products. The SearchService class lets users search for products by name or category, making it easier to find what they need. A simple Payment class is included as a placeholder to simulate the payment step, with the option to expand it later. Finally, to test how everything works together, a basic interface was built using the ShoppingAppGUI class. Although it’s currently console-based, it shows how users would interact with the system

in real scenarios. In summary, the system was built by designing clear and focused classes, connecting them properly, and ensuring that each part supports the main functions of online shopping. The project is structured in a way that makes it easy to maintain, test, and improve in the future.

4.2 Methods

Following the establishment of a well-defined class structure, the implementation of methods focused on enabling the core functionality of the system in a way that directly reflects user and admin behavior within an online shopping environment. Each class was designed with clearly scoped responsibilities, and methods were developed accordingly to support those roles. For instance, the ShoppingCart class includes addItem(), removeItem(), and getTotal() methods, allowing customers to dynamically manage the contents of their cart and track their total expenses in real time. The Customer class implements the placeOrder() method to facilitate the transition from cart to order, automatically generating Order and OrderItem objects based on the current cart contents. The Admin class contains operational methods such as addProduct() and updateStock(), which grant system administrators the ability to maintain the accuracy and availability of the product catalog. Supporting classes like Wishlist and Review also contain relevant methods (addToWishlist(), submitReview()) that enhance user engagement and experience. The SearchService class introduces search-oriented methods (searchByName(), searchByCategory()), improving navigation and product discovery across the platform. Additionally, a placeholder method processPayment() in the Payment class simulates transaction handling, serving as a basis for future financial integrations. Throughout the development of these methods, the guiding objective was to ensure logical coherence, simplicity, and extensibility. Each method was tested independently and integrated gradually into the system, forming a cohesive application where user actions map directly to system behavior.

4.3 OOP Principles

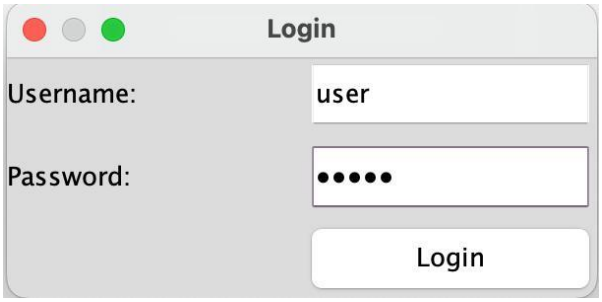
From the earliest stages of development, the project was built around the four foundational principles of object-oriented programming: Encapsulation, Inheritance, Polymorphism, and Abstraction. These principles were applied not just in code structure, but also in how functionality was distributed and managed across the system. Encapsulation was enforced through the use of private class attributes and public getter/setter methods, which ensured controlled access to internal data and preserved the integrity of object states. This can be seen in classes like Product, CartItem, and Order, where data such as price, quantity, or customer information is protected from direct manipulation. Inheritance was used to avoid duplicating shared features between different types of users. A general User class was created to store common information such as name, email, and password, and to handle basic actions like logging in and out. Then, two more specific classes were created —Customer and Admin—that extend User. This allowed us to reuse the common features from User while adding specific methods only where needed. For example, the Customer class includes features like placing orders and managing a shopping cart, while the Admin class focuses on tasks like updating product information. This approach made the system easier to manage and more flexible for future changes. Encapsulation was applied by keeping class attributes private and using getter and setter methods to access them. This helped protect data from unwanted changes. Polymorphism was used when different classes shared similar method names but handled them in their own way, depending on the role or situation. Finally, abstraction allowed us to simplify complex parts of the system by exposing only necessary methods, as seen in the Payment class, which provides a basic structure for handling transactions. Lastly, abstraction played a significant role in simplifying complexity. By exposing only the necessary methods and hiding implementation details—especially in classes like Payment and SearchService—the system remains easy to use, maintain, and extend. The deliberate use of these OOP principles ensured that the codebase is modular, scalable, and aligned with modern software development standards, which was a key learning objective of this project.

5. Results

The final version of the Online Shopping System was tested through its GUI, and all major functionalities were successfully executed. The following screenshots illustrate the core features and workflow of the application:

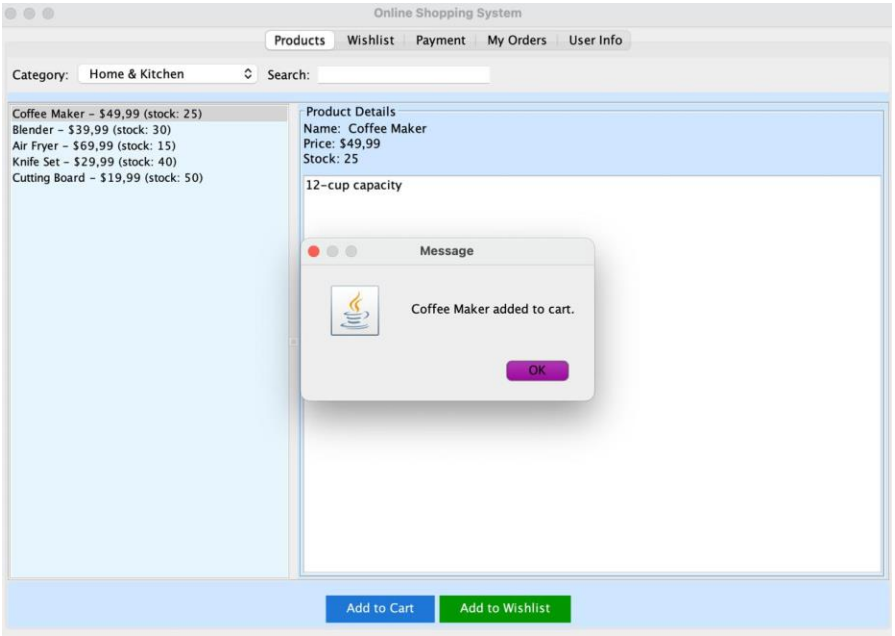
1) Login Screen

This screen allows users to log in by entering their username and password. Depending on the credentials, the system identifies whether the user is a customer or an admin and redirects them to the appropriate interface.



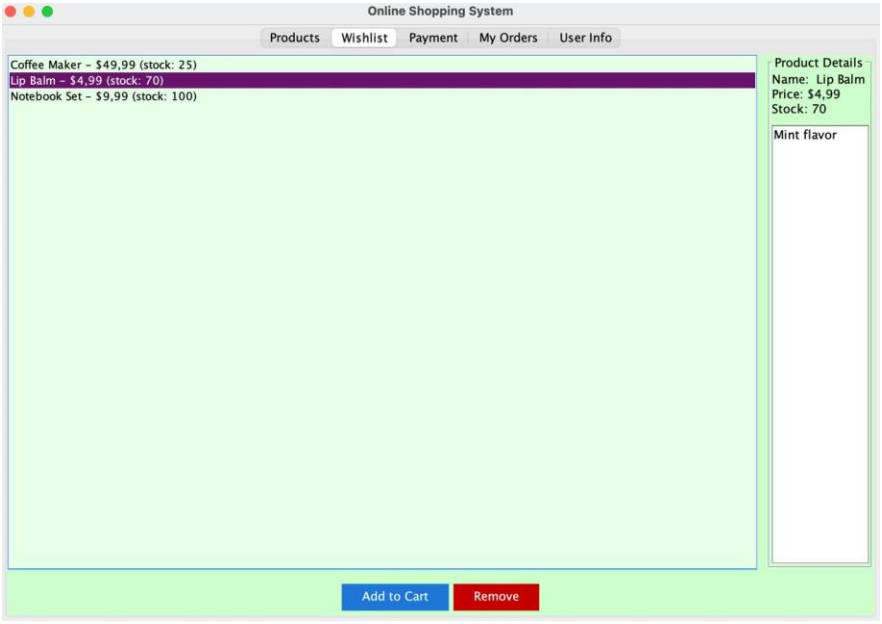
2) Product Browsing and Cart Functionality

Users can browse available products by category and view detailed information such as name, price, and stock status. In this example, the “Coffee Maker” product is selected, and a pop-up confirms that it has been successfully added to the cart.



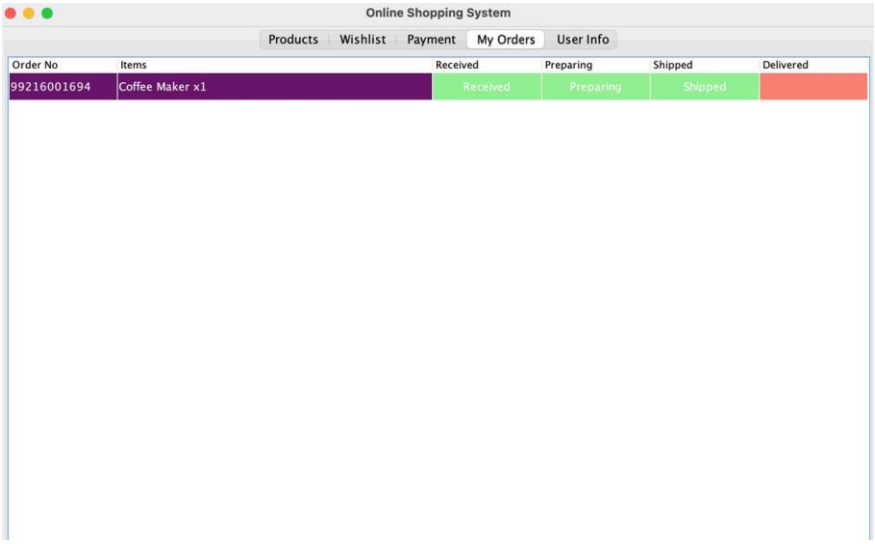
3) Wishlist Interface

This screen shows products that the user has added to their wishlist for future consideration. Users can view product details and either add them to the cart or remove them from the list.



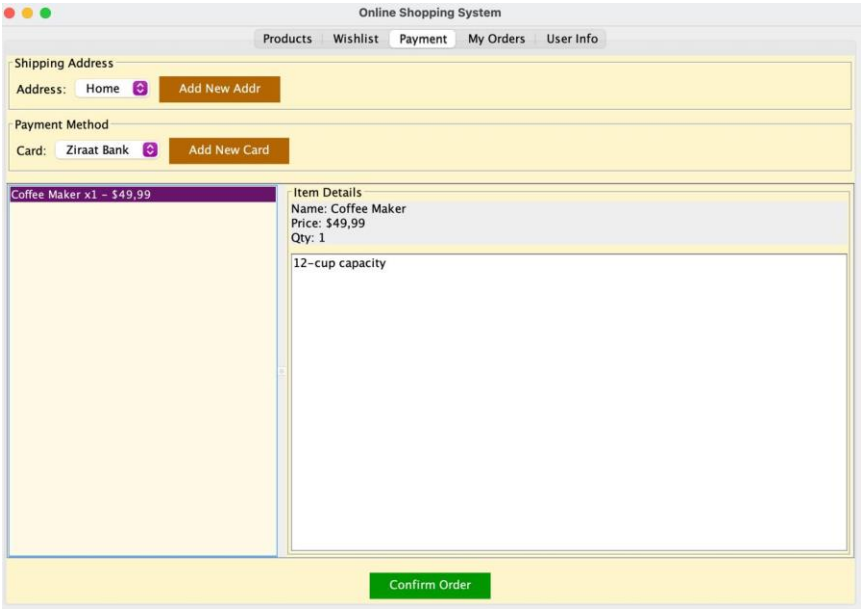
4) My Orders Page

The “My Orders” section provides users with a summary of their completed purchases. Each order displays its number, item(s), and current delivery status—ranging from “Received” to “Delivered.”



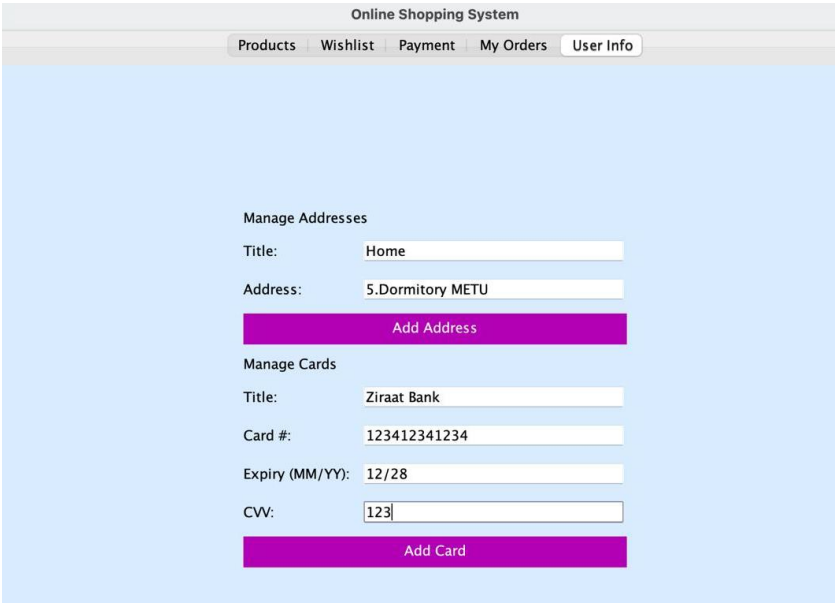
5) Payment and Checkout

This screen shows the checkout process, where users select a delivery address and payment method before confirming their order. The item details and final amount are also displayed.



6) User Info – Address and Card Management

In the "User Info" section, customers can manage their saved addresses and credit/debit card details. New entries can be added, which are then available during the checkout process.



6. Conclusion

This project resulted in the successful development of a complete Online Shopping System that simulates the core functionalities of a real-world e-commerce platform, including user registration and login, product listing and categorization, cart and wishlist management, order placement, payment simulation, and administrative operations. The implementation process followed a structured methodology based on object-oriented programming principles, and each class was designed to serve a specific role in the overall architecture. By applying concepts such as encapsulation, inheritance, polymorphism, and abstraction, we ensured that the system remains modular, readable, and extendable for future improvements. Throughout the project, we aimed to maintain clean and reusable code while also focusing on the user experience through a functional graphical interface that connects all parts of the system. Extensive manual testing was carried out using various user scenarios to verify the accuracy and reliability of all features, and the application performed as expected in each case. The project not only fulfilled the academic requirements of the STAT295 course but also provided valuable experience in translating design ideas into working software, collaborating as a team, solving practical coding problems, and understanding the importance of scalable system architecture. As a result, the system stands as a functional prototype that can easily be extended to include additional features such as persistent data storage, advanced authentication, or third-party service integration in future versions.