**Project Phase 2**

Deliverable 4

**Phase 2 Requirements**

Requirements for phase 2 are mentioned below:

**1. Shopping Cart Management:**

* Allow users to add, remove, and update the quantity of products in their shopping cart.
* The system shall calculate the total cost of whatever is in the cart, including taxes, shipping fees, etc. -dynamically.
* The system should also synchronize the cart with the backend, updating the prices and checking the availability of the product.

**2. Payment Processing:**

* The system shall be integrated with different payment gateways like Stripe and PayPal to facilitate secure transaction processing.
* Provide the system with mock payment methods.
* The system shall provide encryption over the data of payment and confirm a transaction, changing the status of an order based on successful payments.

**3. Tracking an Order and Management:**

* The system shall permit users to track their orders, review order history, and show real-time updates of order status and shipment tracking.
* The system shall clearly display the details of an order for user management of purchases.

**4. Addresses Management:**

* The system shall enable users to save, update, and delete shipping addresses.
* The system should securely store the address information and make it easily available at the time of checkout.
* The system shall allow the user to manage numerous addresses in a user-friendly manner.

**5. Testing and Quality Assurance:**

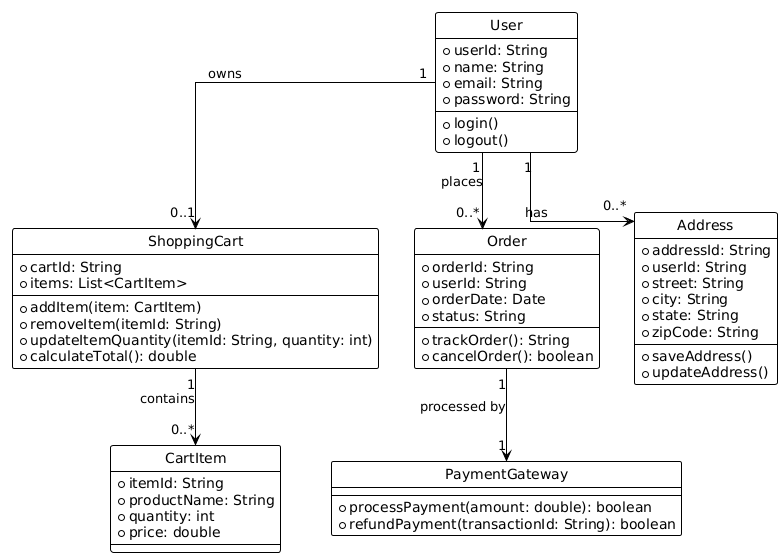
* The system will be subject to thorough unit testing for each new feature being developed in this phase.
* The system shall do regression testing in order to show that functionalities that are already implemented have not been compromised.
* The system shall be redeveloped for those components failing to meet the performance standards.

**6. Recommendation Engine Development:**

* The system is supposed to enhance its recommendation engine using data fed into it from user behavior.
* The system shall further develop algorithms in order to provide recommendations for personalized products.
* The system shall provide integration of the recommendation engine with existing product listing and filtering.

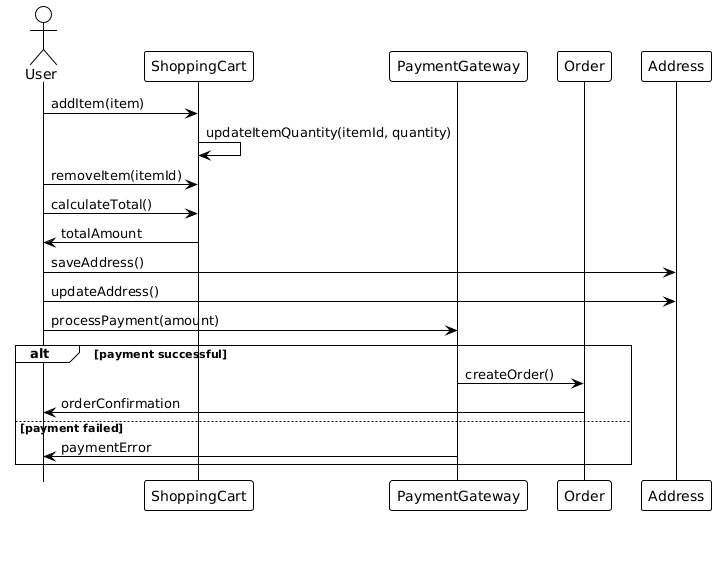
**2. UML Design for Phase 2**

**Class Diagram for Phase 2**



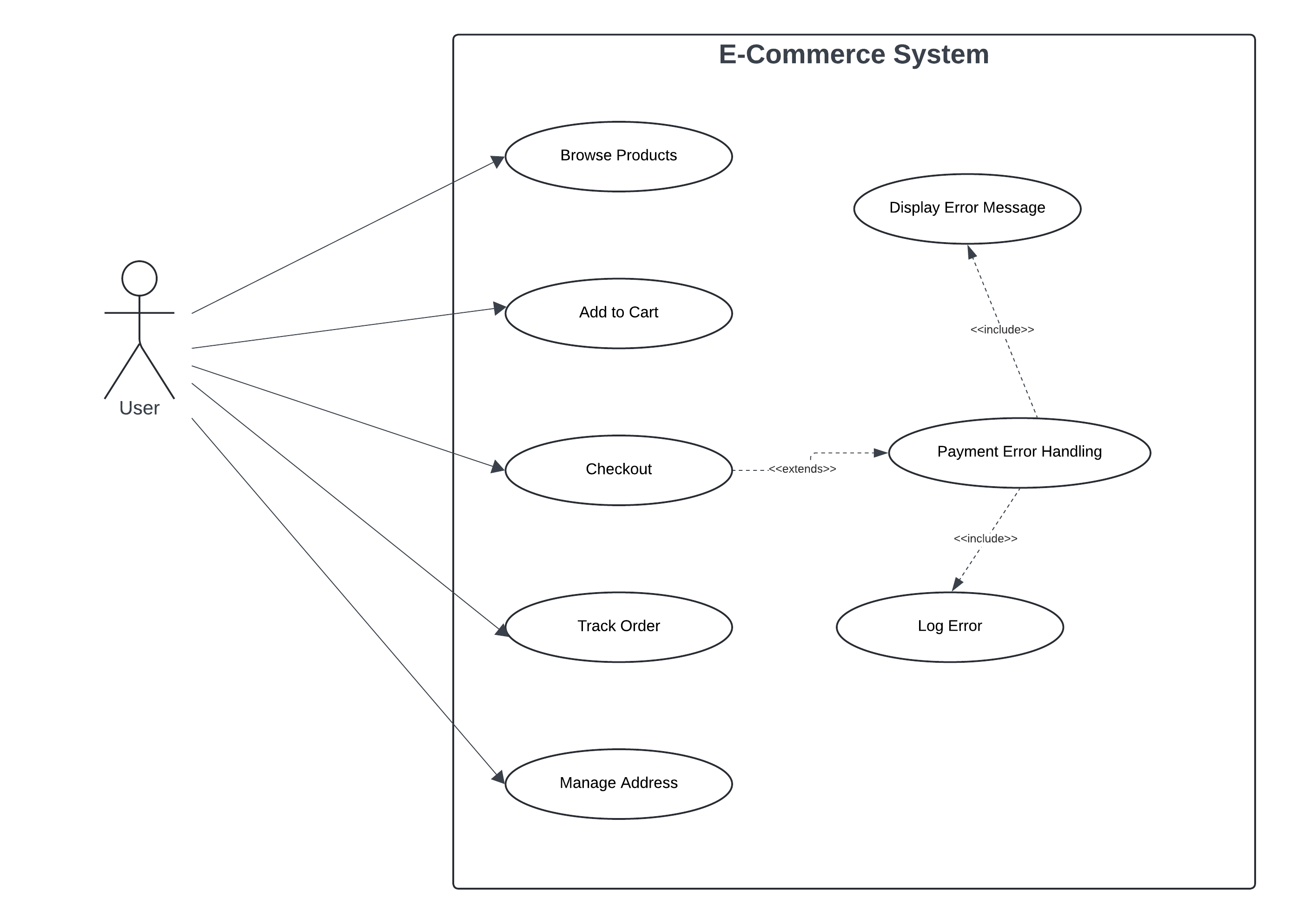
This UML class diagram represents the core components of an eCommerce system, focusing on user interactions and order management.

**Sequence Diagram for Phase 2**



The sequence diagram illustrates the interactions in Phase 2 of the eCommerce system, focusing on the shopping cart, payment processing, and order management.

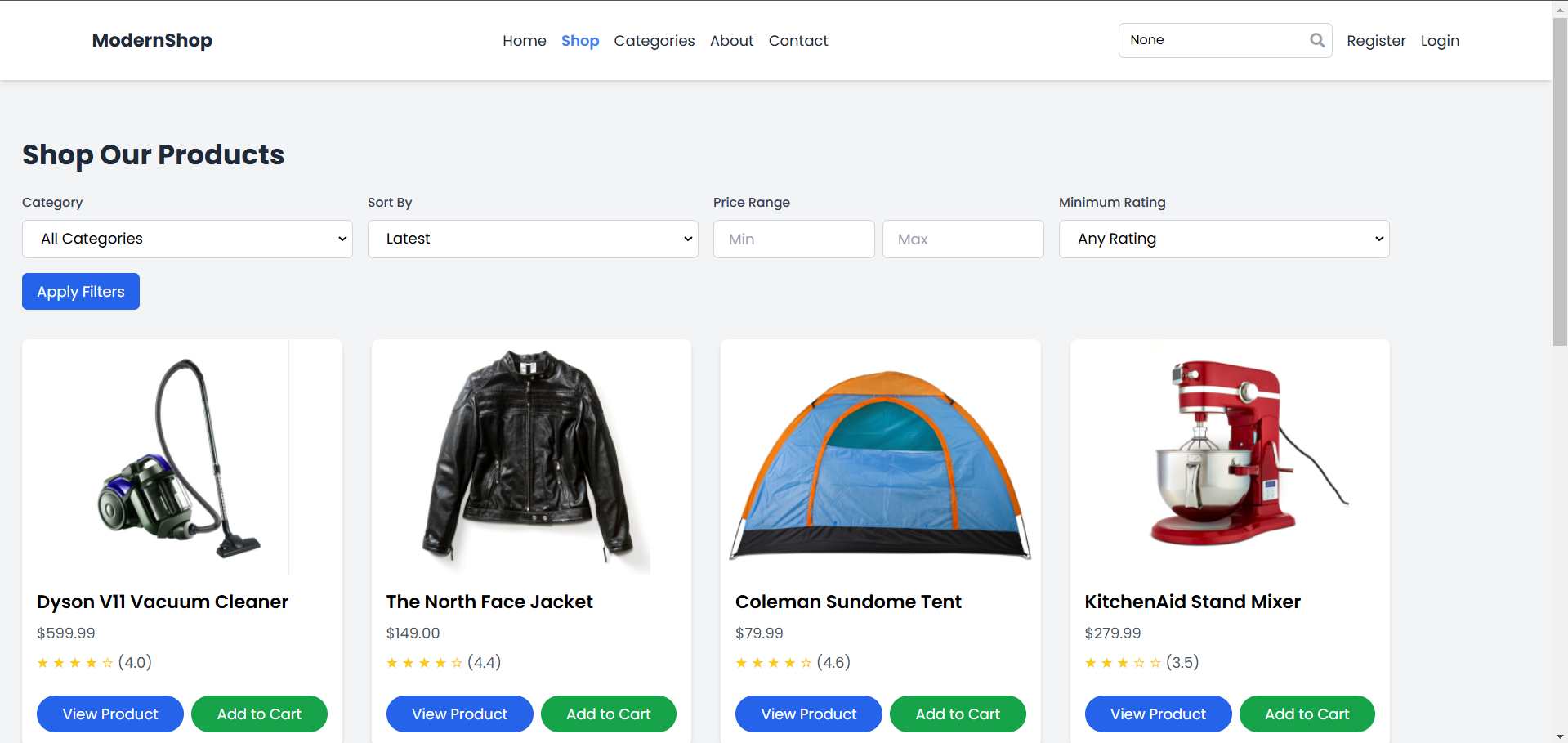
**Use Case Diagram for Phase 2**

****

The use case diagram provides a high-level overview of the functionalities available to a user in the eCommerce system, highlighting both normal and error scenarios.

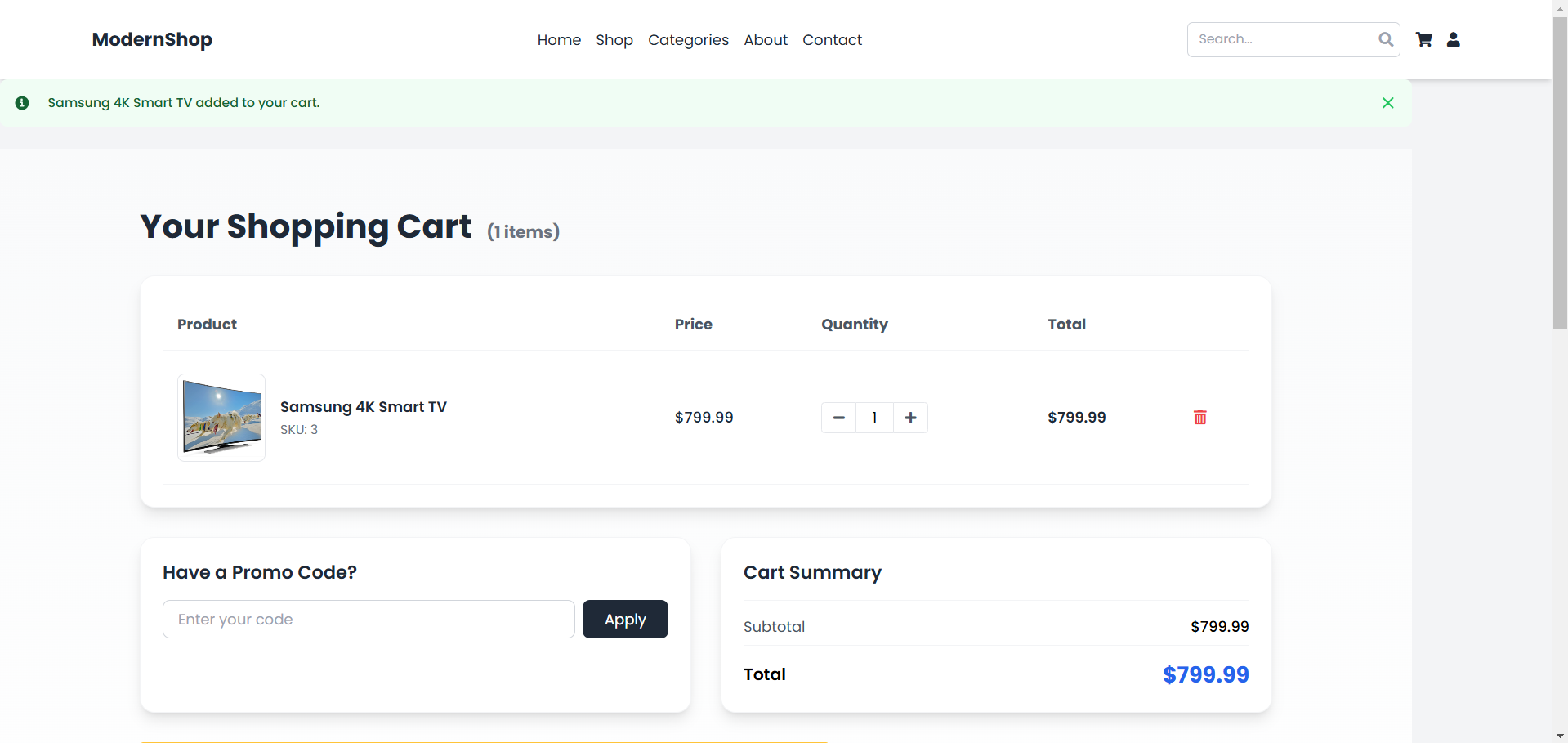
**Test Cases**

**Test Case: Browse Products**

****

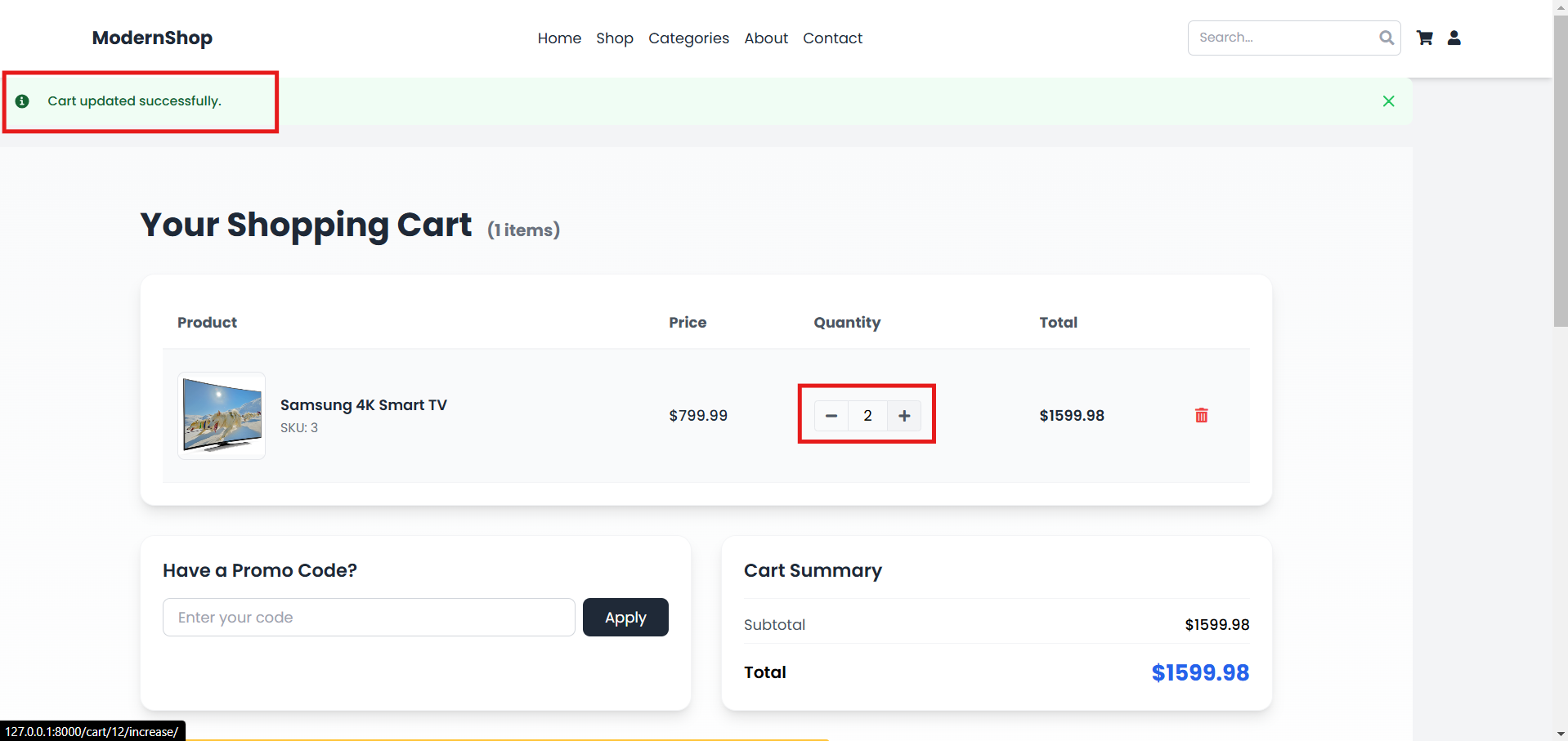
This test covers the mere functionality of being able to enter the product browsing page and successfully view a list of products with name, price, and availability details.

**Test Case: Add to Cart**

****

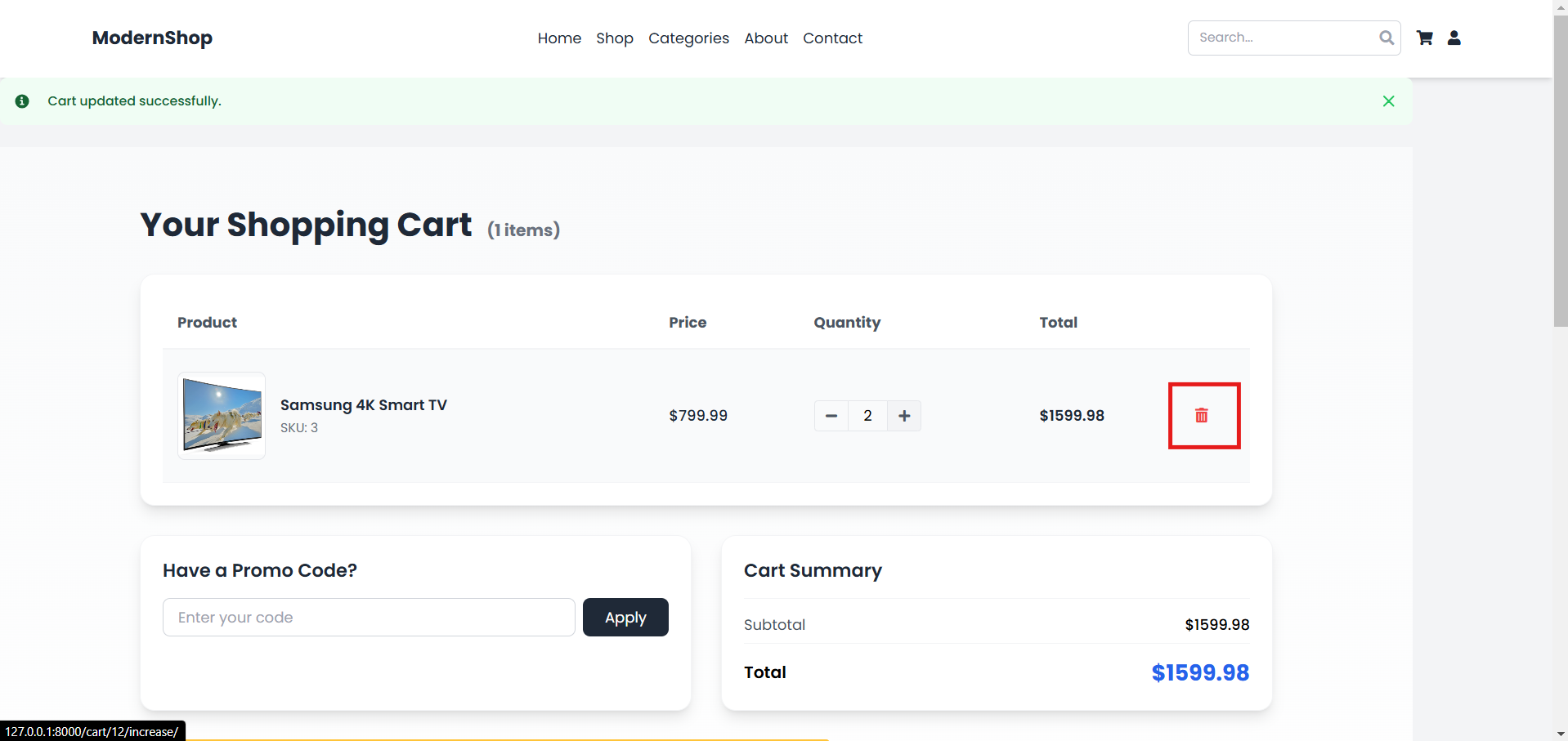
This will make sure that when a user selects a product and clicks "Add to Cart," the item is actually added to the Shopping Cart, and the count of the cart updates accordingly.

**Test Case: Update Cart Quantity**

****

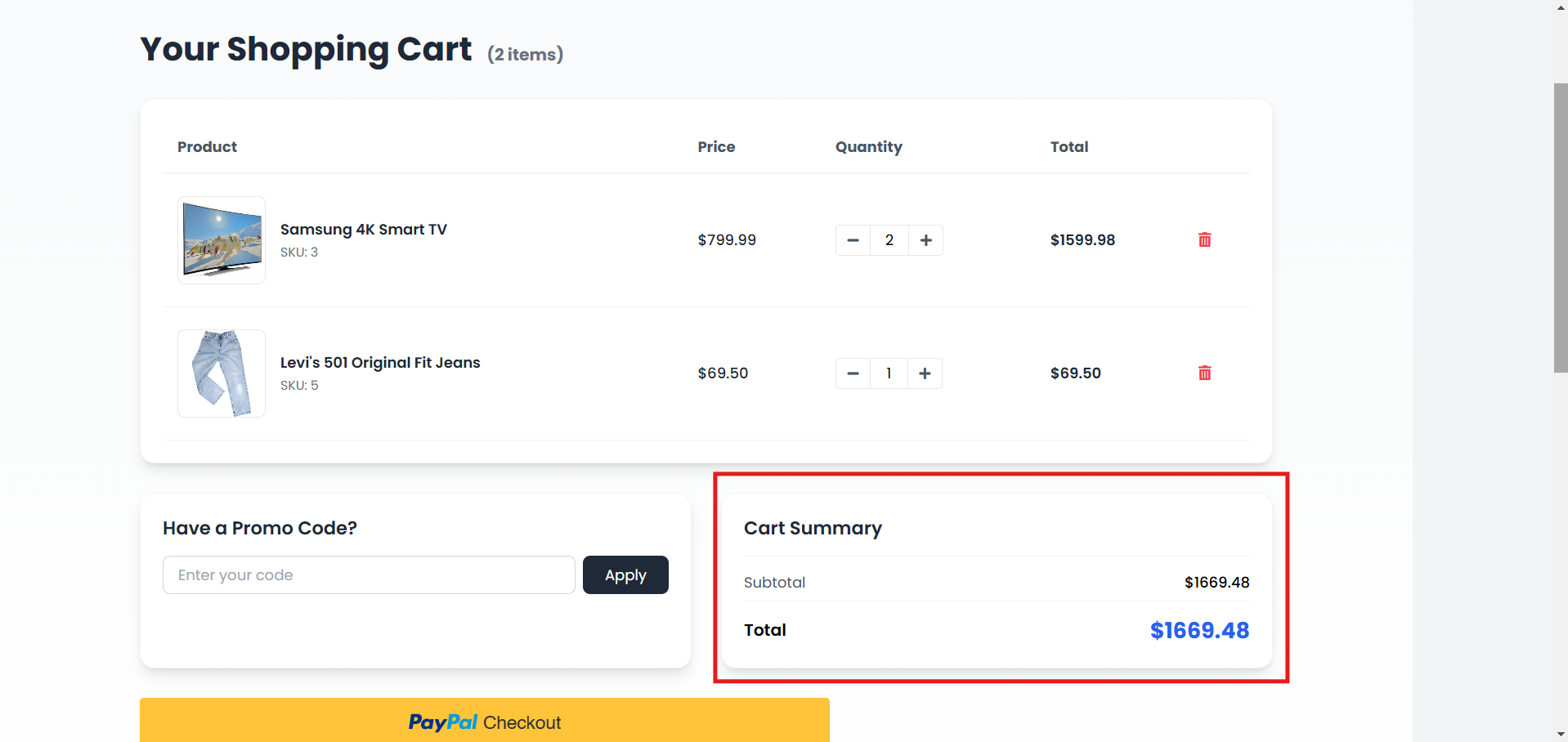
This test verifies that users can modify the number of items in the shopping cart in the Store and validates that the modified number of items appears within the shopping cart while the total price is updated.

**Test Case: Remove from Cart**

****

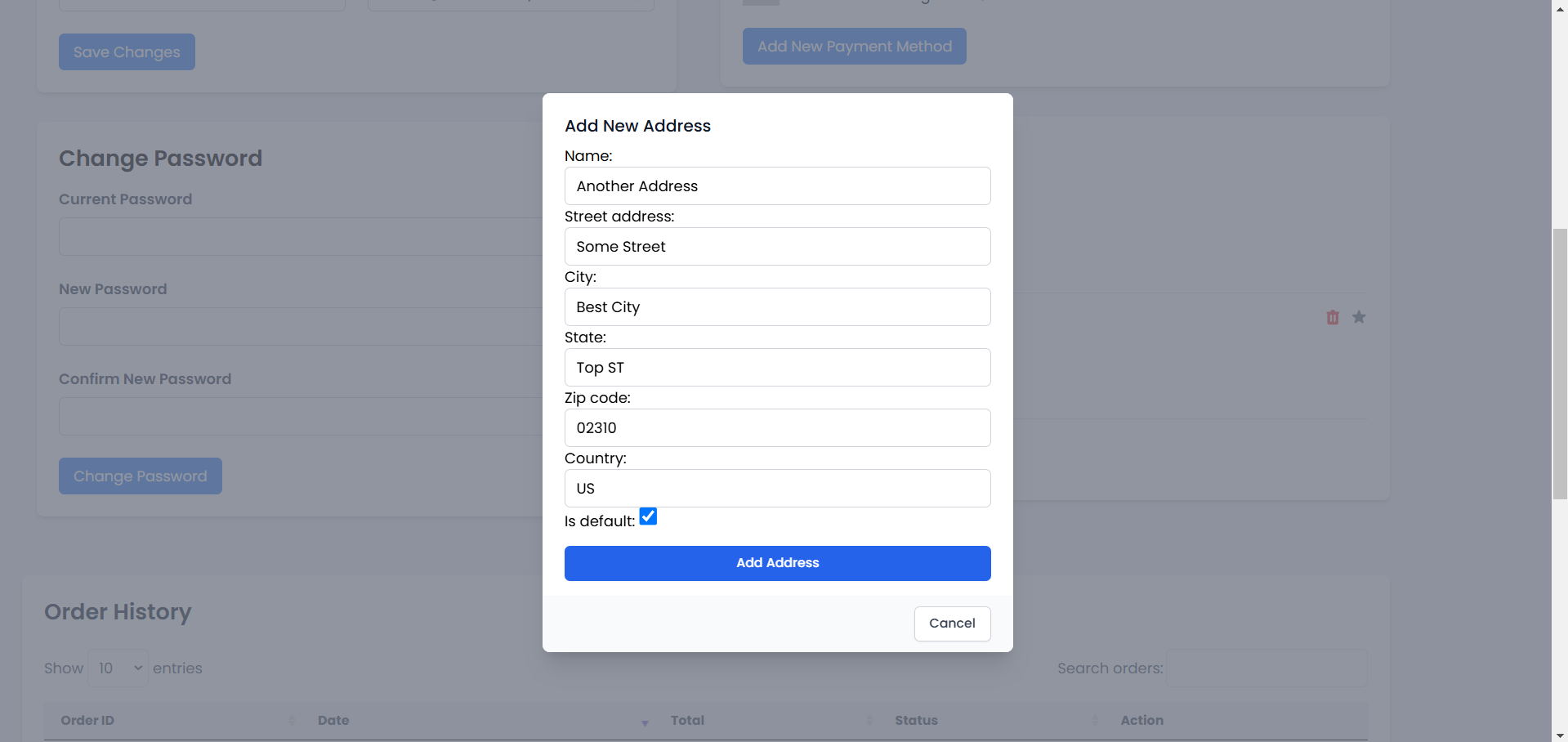
This test covers the scenario that users should be able to delete items added in a cart by selecting an item and clicking "Remove"; the count should decrease accordingly.

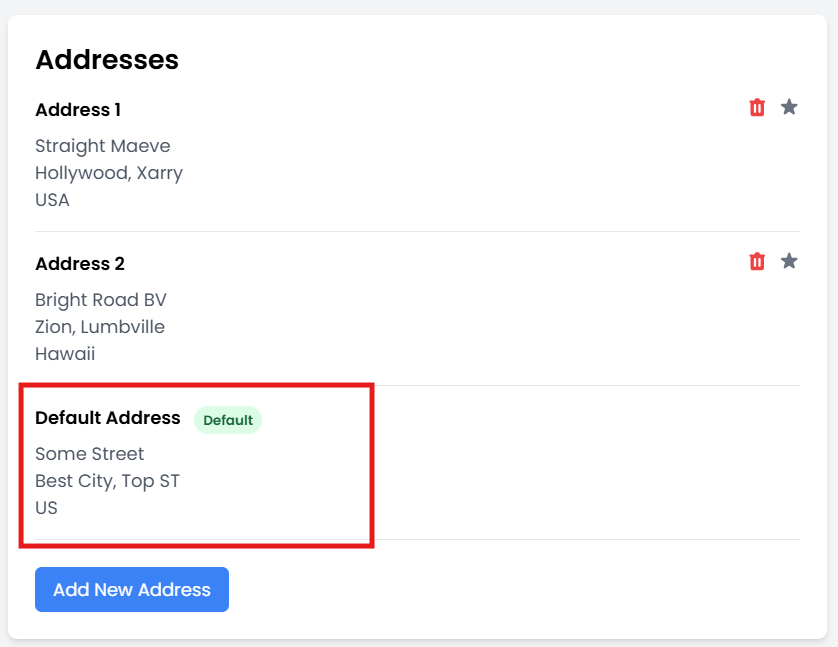
**Test Case: Compute Total**

****

This test ensures that the checkout's total cost contains base item prices, taxes, and shipping fees to give the correct overall value.

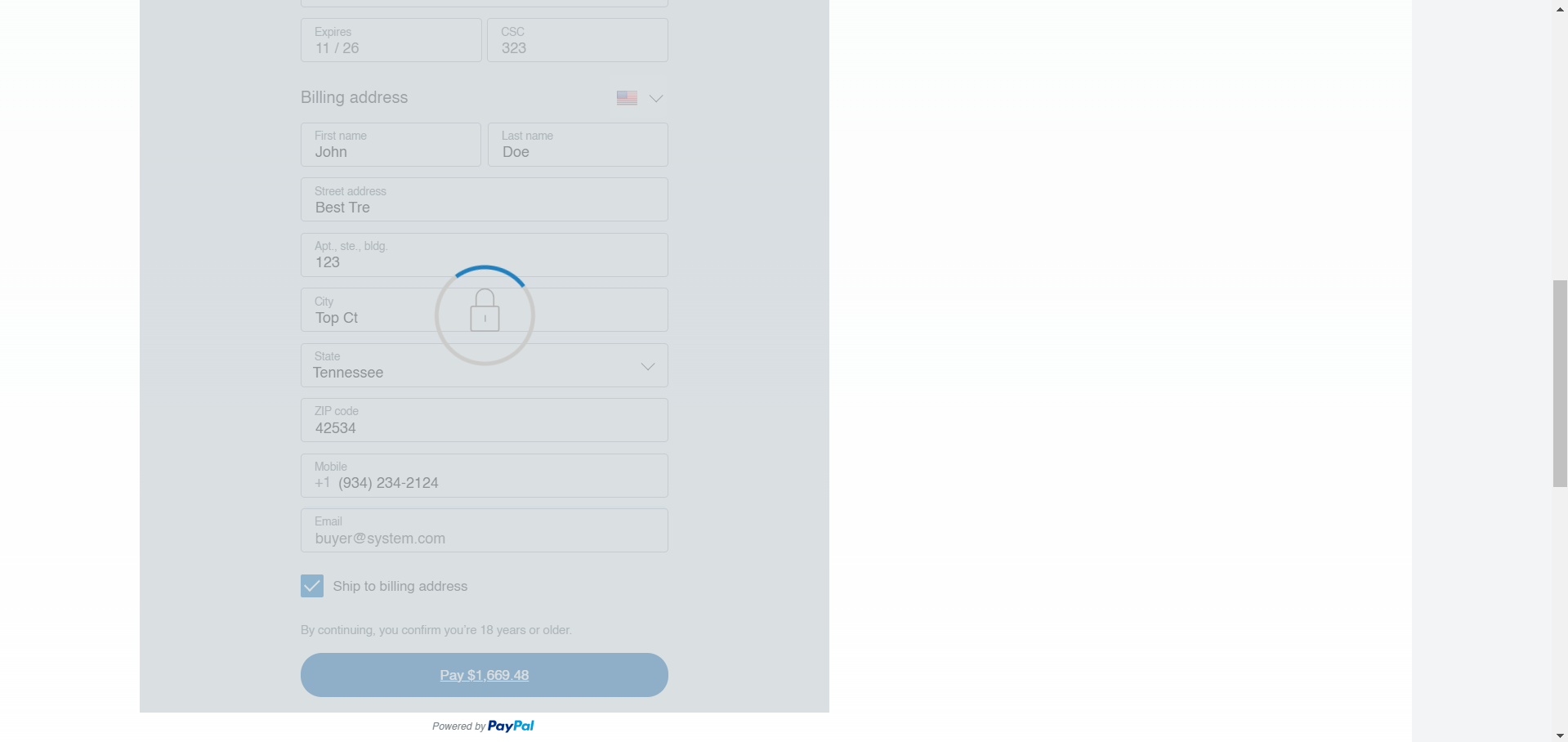
**Test Case: Save Address**

****

****

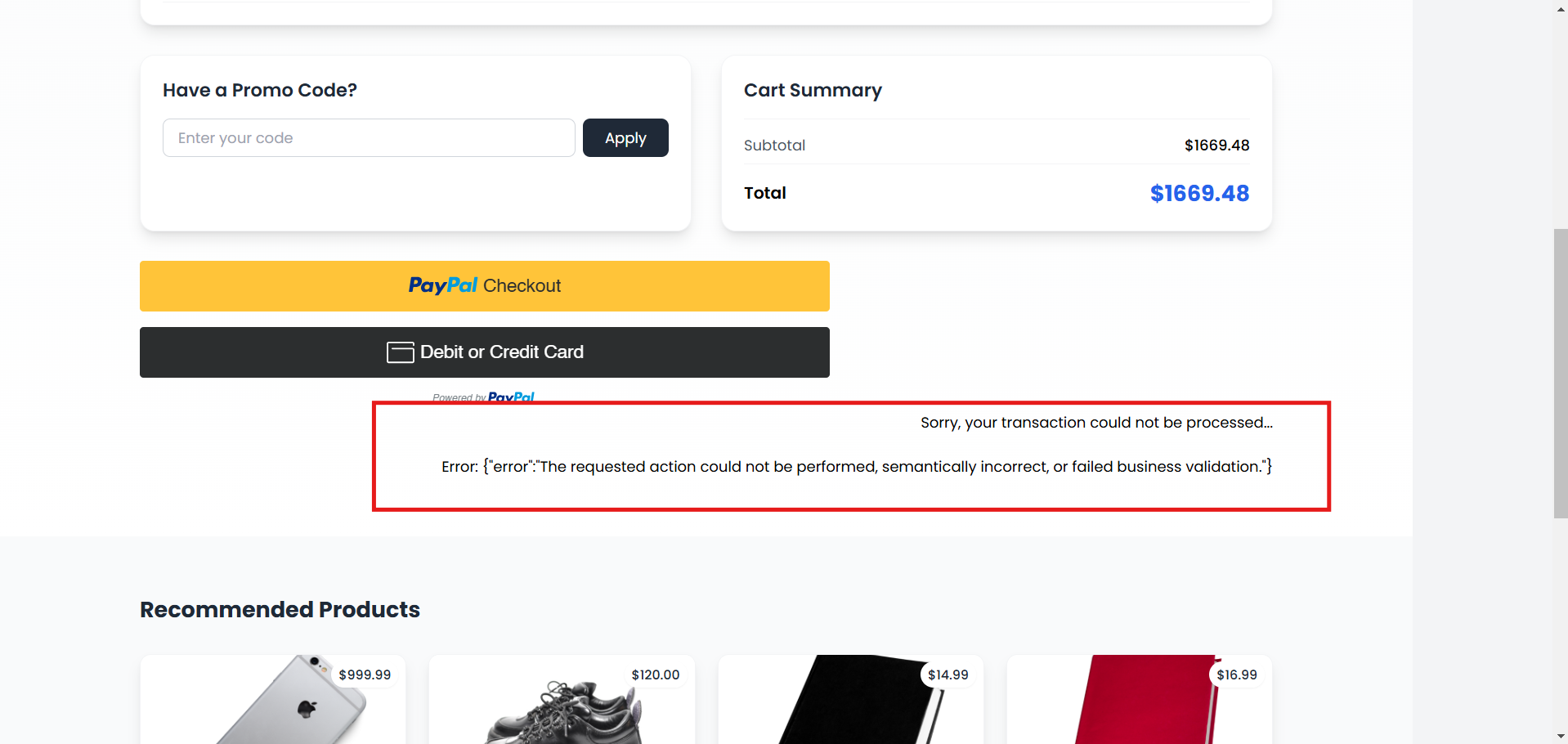
This test covers when details are filled in and the "Save" button is clicked; it confirms a new shipping address was saved and the address shows up within the user's address book.

**Test Case: Payment Processed Successfully**

****

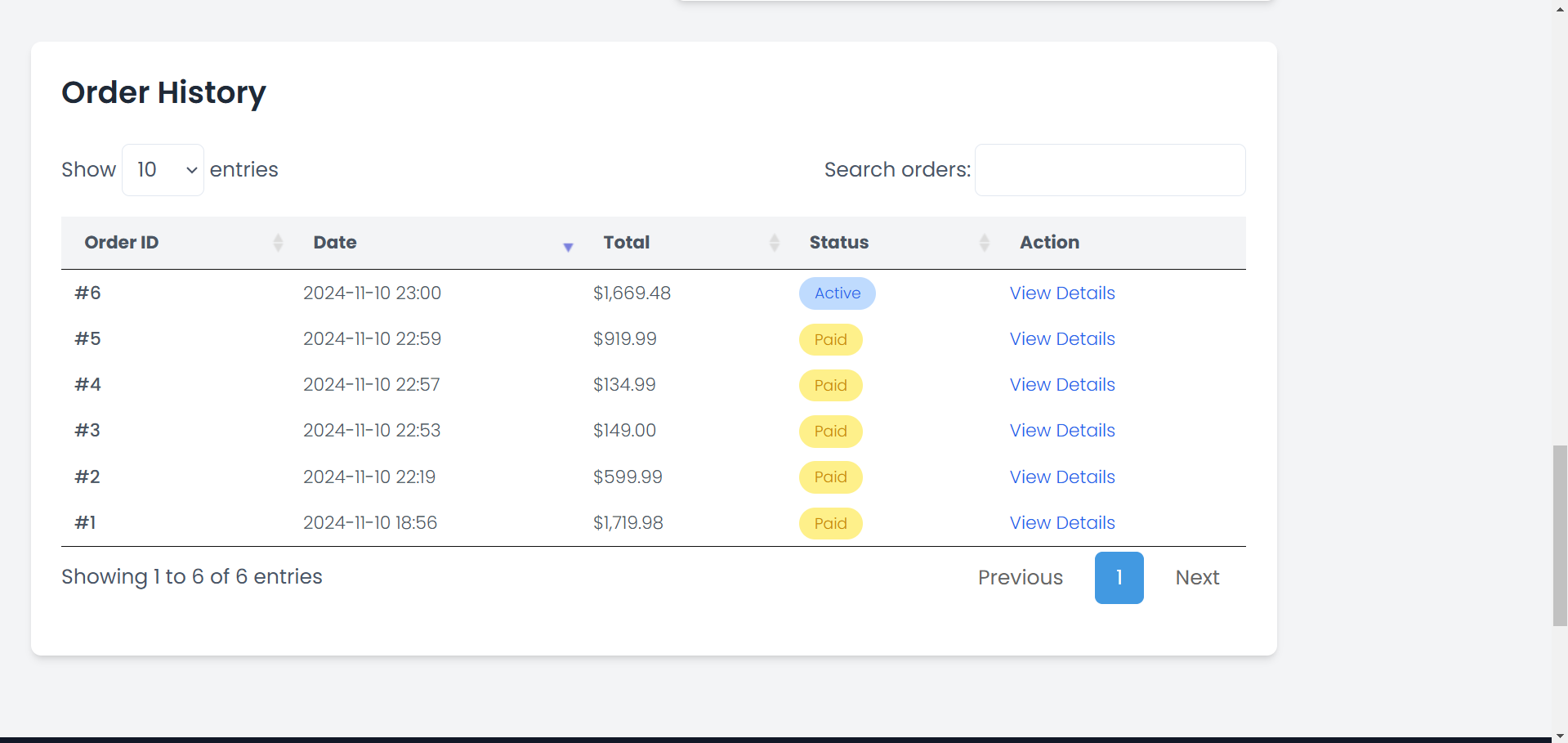
The test below ensures that, on successful payment with valid credit card details by the user, the payment is successfully taken in and sends the order confirmation to the user.

**Test Case: Failure in Payment Processing**

****

By reason, this test would verify that when an attempt is made to make a payment with invalid credit card details, the payment request is declined and a relevant error message is displayed for the user.

**Test Case: Track Order**

****

This test is performed to check that, after a user has finally made a purchase, it is possible for him or her to track down his or her order by being channeled through the order tracking page, where the status of this is shown.

**User Manual**

**Creating an Account**

1. **Visit the Homepage:**
   * Go to our website and click on the "Sign Up" button located at the top right corner.
2. **Fill in Your Details:**
   * Enter your name, email address, and create a password.
   * Click "Register" to create your account.
3. **Email Confirmation:**
   * Check your email for a confirmation link and click on it to verify your account.

**Logging In**

1. **Access the Login Page:**
   * Click on the "Login" button on the homepage.
2. **Enter Credentials:**
   * Input your registered email and password, then click "Login."

**Browsing Products**

* Navigate to the "Products" section from the main menu to explore our catalog.
* Use filters and search options to find specific items.

**Managing Your Cart**

**Adding Items to Cart**

* Click on a product to view details, then select "Add to Cart" to include it in your shopping cart.

**Viewing and Editing Cart**

* Access your cart by clicking the cart icon at the top of the page.
* Adjust quantities or remove items as needed.

**Checkout Process**

1. **Review Cart:**
   * Ensure all desired items are in your cart and proceed by clicking "Checkout."
2. **Enter Shipping Information:**
   * Provide your shipping address or select a saved address.
3. **Select Payment Method:**
   * Choose your preferred payment option and enter necessary details.
4. **Confirm Order:**
   * Review your order summary and click "Place Order" to complete the purchase.

**Tracking Your Order**

* Go to the "My Orders" section from your account dashboard to view order status and tracking information.

**Managing Addresses**

* Access the "Address Book" from your account settings to add, update, or delete addresses.

**Handling Payment Issues**

* If a payment fails, an error message will appear with instructions on how to resolve the issue.

**Setup**

1. **Clone the Repository:**
   * Use a version control system like Git to clone the project repository to your local machine.
   * Navigate into the project directory using the terminal.
2. **Create a Virtual Environment (Optional but Recommended):**
   * Create a virtual environment to manage dependencies separately from your system Python installation.
   * Activate the virtual environment.
3. **Install Dependencies:**
   * Install the required Python packages using the command pip install -r requirements.txt. This file should list all necessary packages for your Django project.

**Run Program**

1. **Apply Database Migrations:**
   * Run python manage.py migrate to apply any pending database migrations. This step sets up your database schema according to the models defined in your Django app.
2. **Create a Superuser (Optional):**
   * If needed, create an admin superuser for the Django admin interface using python manage.py createsuperuser.
3. **Start the Development Server:**
   * Use python manage.py runserver to start the Django development server. By default, it runs on http://localhost:8000.
4. **Access the Application:**
   * Open a web browser and go to http://localhost:8000 to view your application. Use http://localhost:8000/admin to access the admin interface if you created a superuser.

These instructions provide a comprehensive guide to setting up, running, and testing your Django project. Adjust any specific paths or commands as necessary to fit your project's configuration.

**Peer Review Feedback**

Improvements were pointed out during the code inspection session, and that set up quite a few valuable insights into enhancements in this project. The overall structure was basically commended for being organized, but there was a feeling that further modularization of some of the larger views could be even more readable or maintainable. This was brought back in the form of feedback that has led to a refactoring effort: breaking these views down into smaller, more manageable functions and classes, in general cleaning up this code base.

Other important feedback regarded documentation. The code was indeed working, but it was mentioned that the code lacked sufficient inline comments and docstrings, especially in complex functions. Accordingly, all major functions and classes were elaborated with extensive docstrings and inline comments, explaining clearly what each was meant for and what it does. This enhancement makes it easy not only for present developers but also for future maintenance and the addition of new team members.

Handling errors was also pointed out as the one that needed much attention. Some parts of the code did not handle possible exceptions, which may lead to an unexpected crash. In order to be able to resolve this, try-except blocks were placed in every vital area in order for the application to handle the error well and stay stable during an unexpected event.

Testing grounds showed that some modules, especially user authentication and the processing of online payments, were not well covered with tests. In this regard, more test cases were created focusing on previously untested modules and their edge cases. This greatly improved the robustness of the test framework so as to ensure that the application behaves as expected under different circumstances.

Finally, performance optimization was done, and a few database queries were suggested to be potentially very inefficient. Better use of Django's ORM was called for. Thus, revisiting the database queries and optimizing them by using various techniques such as select\_related and prefetch\_related reduced the number of database hits and hence generally optimized the performance.

These combined actions raised the quality, stability, and performance of the project to a much closer standard conformance, improving the user experience. The improvements were informed by feedback provided during the code inspection session, supporting the benefits of frequent code reviews and collaborative input.

**Reflection on Project Accomplishments**

**Accomplishments**

These included the implementation of major functionalities, setting up a robust testing framework for the code, and performance optimization of some critical parts. The application features a refactored codebase that follows the best practices for maintainability and scalability. Furthermore, comprehensive documentation and improvements in error handling not only boost the overall reliability but also improve the experience of working with it.

**What went well**

Several things went really well: firstly, the development in collaboration with everyone in the groups created even more opportunities to fix any arising problems more quickly, to realize new ideas, and to inspire others. All the feedback given during the code inspection session was useful and allowed rewriting big parts of code in an even better way. Refactoring the code structure in many classes and optimizing database queries gave good results to make noticeable improvements in performance. In addition, the improved test coverage has enhanced robustness in ensuring the application meets any prerequisites for reliability.

**Areas to Improve**

These notwithstanding, there is quite a lot of room for improvement. For example, the initial phases could have been better with detailed planning and analysis of requirements to avoid scope creep, hence capturing all the needs of the stakeholders. While the documentation was done in enhancements after the inspection, a more thorough documentation process right from the very beginning could have saved time and reduced ambiguities. Finally, it was hoped that an investigation into the use of automated testing tools and continuous integration practices could further reduce development cycles and improve code quality.

**Member Contribution Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Member Name** | **Contribution Description** | **Overall Contribution (%)** | **Note (if applicable)** |
| Harini | Frontend shop, sequence diagram, meeting minutes | 12.5% |  |
| Osama | Backend user | 12.5% |  |
| Nirupama | Frontend users | 12.5% |  |
| Niharika | Frontend shop, usecase diagram, meeting minutes | 12.5% |  |
| Jaswanth | Backend users | 12.5% |  |
| Rajsekhar | Backend shop | 12.5% |  |
| Triveni | Frontend users | 12.5% |  |
| Sharanya | Backend shop, class diagram | 12.5% |  |