ONLINE SHOPPING MART

(SRS DOCUMENT)

- **Author** : Dharsan S

Table of Content pg.no

1.Abstract:

Online shopping marts have emerged as a significant component of e-commerce, offering consumers convenience, variety, and accessibility like never before. This abstract explores the evolution of online shopping marts, tracing their inception, growth, and impact on modern consumer behavior.

Initially, online shopping platforms began as simple websites where users could browse and purchase products. However, with advancements in technology and the proliferation of smartphones, online shopping marts have evolved into sophisticated ecosystems offering a plethora of features and services. From personalized recommendations and virtual try-on experiences to secure payment gateways and expedited delivery options, these platforms continuously innovate to enhance the shopping experience.

One of the key drivers behind the success of online shopping marts is their ability to transcend geographical boundaries, enabling consumers to access products from around the globe with just a few clicks.

However, the evolution of online shopping marts has also presented challenges, including concerns regarding data privacy, cybersecurity, and the displacement of traditional brick-and-mortar retailers.

2.Requirement Analysis:

1. Functional Requirement

2. Non-functional Requirement

2.1.Functional Requirement:

1.User Authentication and Management:

Users should be able to register an account with the online shopping mart.

Registered users should be able to log in securely with credentials.

Users should be able to update their profile information, including personal details and shipping addresses.

Password recovery mechanism should be provided for users who forget their credentials.

2.Product Catalog Management:

Admin should be able to add, update, and remove products from the catalog.

Each product should have details such as name, description, price, availability, and images.

Products should be categorizable for easy navigation.Admin should be able to manage product attributes like size, color, and variations.

3.Search and Filtering:

Users should be able to search for products using keywords.Advanced search options like filters by category, price range, brand, and rating should be available.

4.Shopping Cart and Checkout:

Users should be able to add products to their shopping cart.

Users should be able to view their cart contents, update quantities, and remove items.

Secure checkout process should be provided for users to complete their purchase.

Multiple payment options such as credit/debit card, PayPal, and others should be supported.

5.Order Management:

Admin should have access to an order management system to view, process, and fulfill orders.

Order status should be updated in real-time to reflect processing, shipping, and delivery stages.

Users should be able to track the status of their orders through their account.

6.User Reviews and Ratings:

Users should be able to leave reviews and ratings for products they have purchased.

Reviews and ratings should be visible to other users browsing the product.

7. Discounts:

Admin should be able to create promotional offers and discounts.

Users should be able to apply discount codes during checkout.

8.User Support:

A customer support system should be available for users to contact the support team.

2.2.Non-Functional Requirements:

1.Personalization:

The system should offer personalized product recommendations based on user browsing history, purchase patterns, and preferences.

Customization options should be available, such as saved preferences, wish lists, and personalized product feeds.

2.Real-time Inventory Management:

The system should update product availability in real-time to prevent overselling and ensure accurate stock levels.

3.Seamless Integration with Third-party Services:

Integration with external services, such as payment gateways, shipping providers, and customer relationship management (CRM) systems, should be seamless and reliable.APIs should be well-documented.

4.Dynamic Pricing:

The system should support dynamic pricing strategies, allowing for real-time adjustments based on factors such as demand, competitor pricing, and inventory levels.

5.Augmented Reality (AR) Integration:

The system should support AR technology for virtual product try-on experiences, allowing users to visualize how products will look or fit before making a purchase.

AR functionality should be compatible with supported devices and browsers, providing a seamless and immersive shopping experience.

6.Social Media Integration:

The system should enable seamless integration with social media platforms, allowing users to share products, reviews, and purchases with their social networks.

7.Localized Content and Currency Support:

The system should provide localized content and currency support for users in different regions, including language translations, region-specific pricing, and cultural nuances.

3.Design:

High level design

Low level design

3.1.High level design:

High-Level Design (HLD) refers to the architectural blueprint or overview of a software system. It provides a bird's-eye view of the system's structure, focusing on its major components, their interactions, and the overall flow of data and control.

1.System Architecture:

Describes the overall structure of the system, including the arrangement of its components and how they interact with each other.

Identifies architectural patterns or styles used, such as client-server, microservices, or layered architecture.

2.Data Flow and Control Flow:

Illustrates how data flows through the system, from input to processing to output.

Defines the control flow, including the sequence of operations and decision points within the system.

3.External Interfaces:

Specifies the interfaces with external systems, services, or users.Describes how the system interacts with external entities, such as APIs, databases, or user interfaces.

4.Scalability and Performance Considerations:

Addresses scalability requirements, such as the ability to handle increasing loads or accommodate future growth.Considers performance optimizations, such as caching strategies, load balancing, and parallel processing.

5.Security and Reliability:

Outlines security measures to protect the system against unauthorized access, data breaches, and other security threats.Addresses reliability requirements, such as fault tolerance, redundancy, and disaster recovery mechanisms.

6.Technology Stack:

Specifies the technologies, frameworks, and tools to be used in the implementation of the system. Justifies the selection of each technology based on its suitability for the system's requirements and constraints.

7.Development Approach:

Defines the development methodology or approach to be followed, such as Agile, Waterfall, or DevOps.Outlines the development lifecycle, including phases, milestones, and deliverables.

8.Documentation and Communication:

Ensures that the high-level design is documented comprehensively and communicated effectively to stakeholders, architects, and development teams.

3.2.Low level design:

Low-Level Design (LLD) delves deeper into the implementation details of the system, focusing on the internal workings of individual components identified in the high-level design. It provides detailed specifications that developers can use as a blueprint for coding and implementation

1.Module/Component Details:

Breaks down each major component identified in the high-level design into smaller modules or units.Specifies the functionality of each module and its inputs, outputs, and processing logic.

2.Data Structures and Algorithms:

Specifies the data structures to be used for storing and manipulating data within the system.

Defines algorithms and algorithms for common operations such as searching, sorting, and data manipulation.Optimizes algorithms and data structures for efficiency and performance.

3.Database Design:

Designs the database schema, including tables, columns, relationships, and constraints.Defines SQL queries, stored procedures, and triggers for data manipulation and retrieval.Specifies data types, indexing strategies, and normalization techniques to ensure data integrity and performance.

4.Class Design:

Creates detailed class diagrams that represent the objects and classes within the system.Designs class hierarchies and interfaces to encapsulate behavior and promote reusability.

5.Sequence Diagrams:

Illustrates the flow of control and data between objects or components in various scenarios.Specifies message passing and parameter passing between objects, including synchronous and asynchronous communication.

6.Error Handling and Exception Handling:

Designs error handling mechanisms to detect and handle errors, exceptions, and unexpected conditions.Defines exception handling routines to gracefully handle exceptional situations and prevent system failures.

7.Interfaces and APIs:

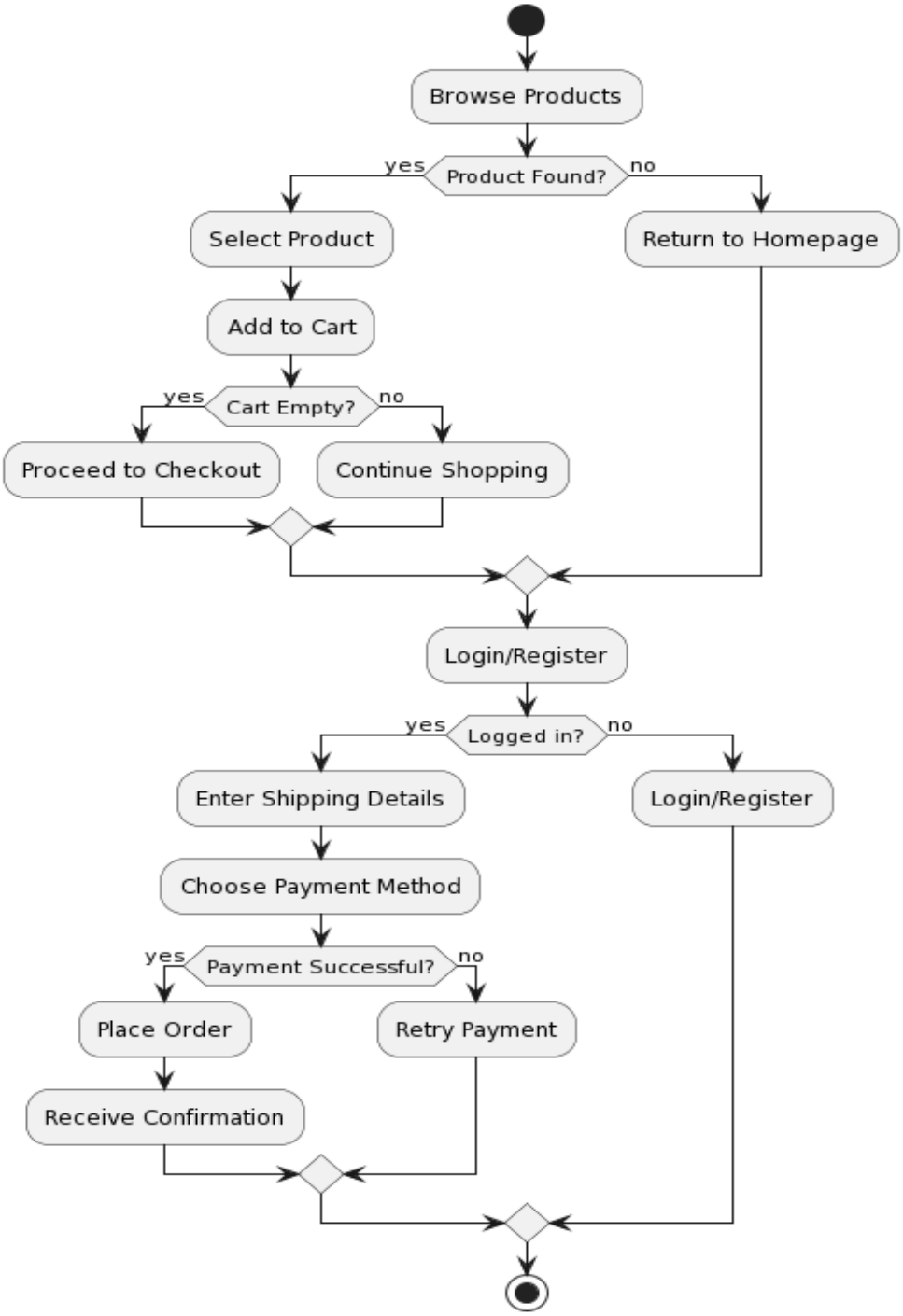
Designs interfaces and APIs for external interactions with the system.Specifies method signatures, parameters, return types, and error handling for each API endpoint.

8.Security Implementation:

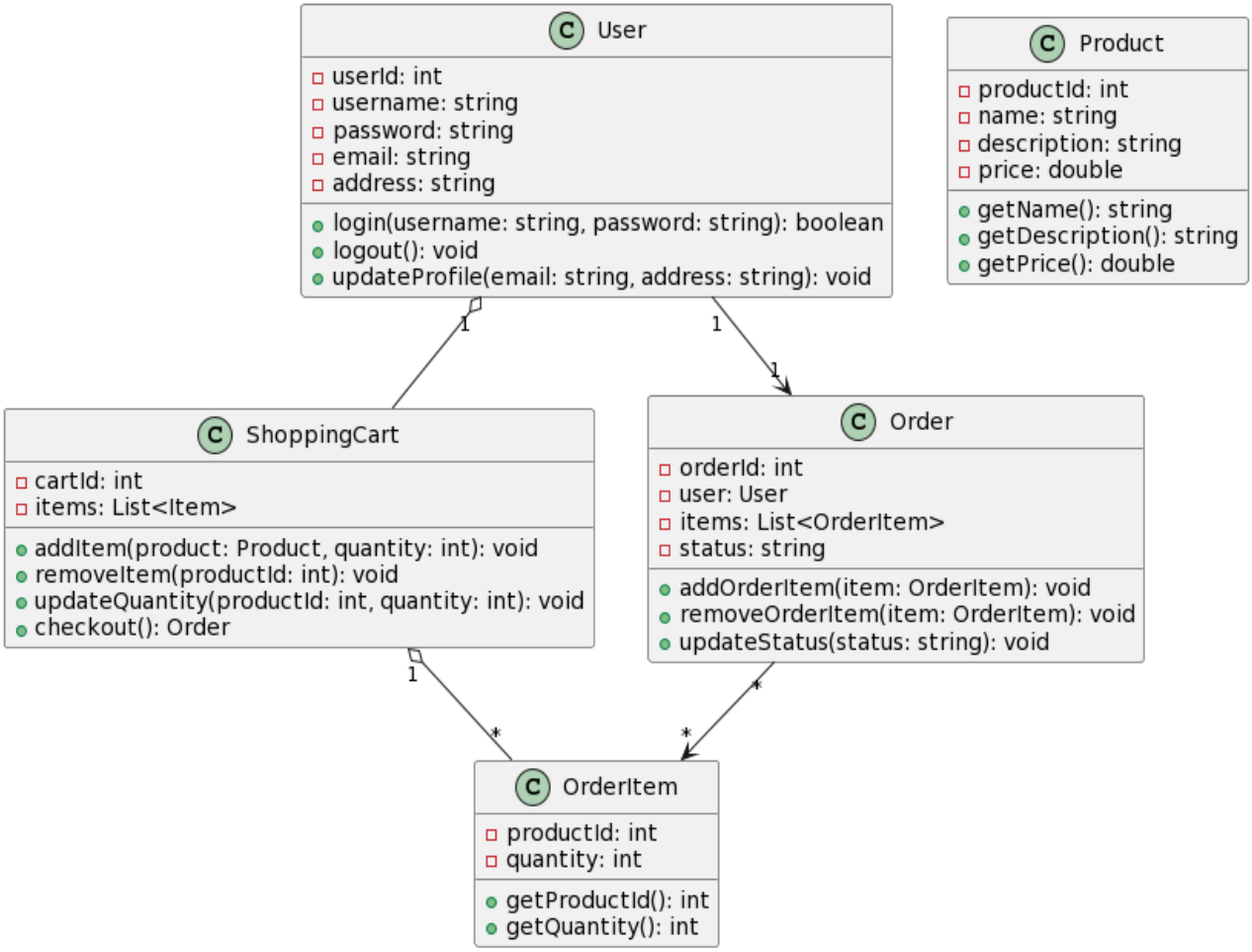
Implements security measures to protect the system against common security threats and vulnerabilities.Implements security best practices, such as input validation, output encoding, and parameterized queries, to prevent security breaches.

4.Implement

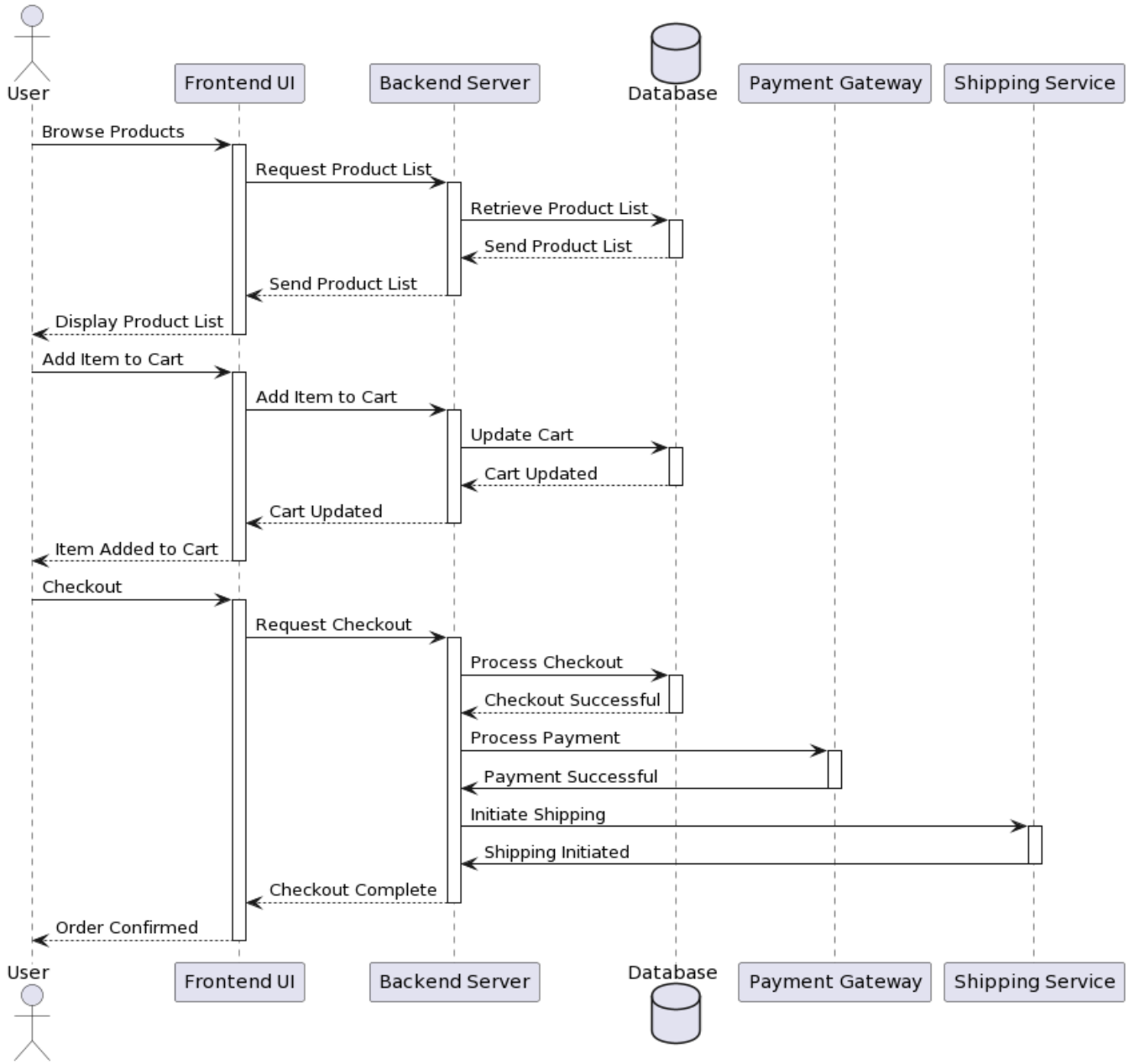
4.1 Flow Diagram:



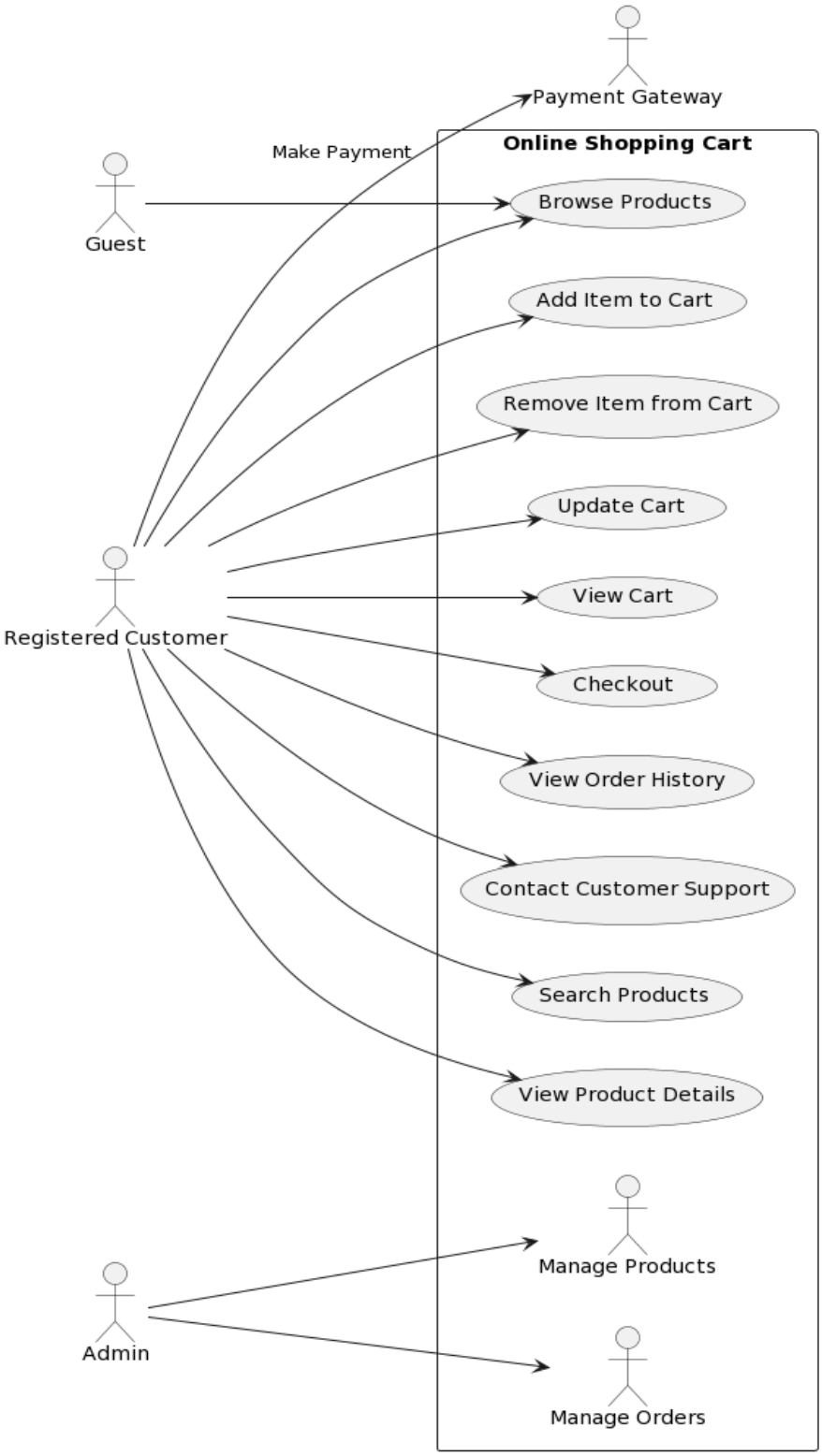
4.2.Class Diagram:



4.3.Sequence Diagram:



4.4.Usecase Diagram:



5.TestCase:

1.User Registration:

Test case 1: Verify successful registration with valid credentials.

Test case 2: Verify registration failure with an invalid email format.

Test case 3: Verify registration failure with a weak password.

Test case 4: Verify registration failure with an existing username.

2.User Login:

Test case 5: Verify successful login with correct credentials.

Test case 6: Verify login failure with incorrect password.

Test case 7: Verify login failure with a non-existing username.

Test case 9: Verify login failure for a deactivated account.

3.Product Browsing:

Test case 10: Verify users can view the list of available products.

Test case 11: Verify users can filter products by category.

Test case 12: Verify users can sort products by price or popularity.

Test case 13: Verify users can view product details by clicking on a product.

Test case 14: Verify users can view product reviews and ratings.

4.Shopping Cart:

Test case 15: Verify users can add products to the shopping cart.

Test case 16: Verify users can remove products from the shopping cart.

Test case 17: Verify users can update the quantity of products in the shopping cart.

Test case 18: Verify the shopping cart retains items after user logout/login.

5.Checkout Process:

Test case 19: Verify users can proceed to checkout with items in the shopping cart.

Test case 20: Verify users can enter shipping details.

Test case 21: Verify users can select a payment method.

Test case 22: Verify users receive an order confirmation after successful checkout.

6.Order Management:

Test case 23: Verify users can view their order history.

Test case 24: Verify users can track the status of their orders.

Test case 25: Verify users can cancel an order before shipment.

Test case 26: Verify users receive email notifications for order updates.

Test case 27: Verify users can contact customer support for order-related inquiries.