**ONLINE SHOPPING SYSTEM**

**ABSTRACT:**

The Software Requirements Specification (SRS) report for Online Shopping Systems describes the fundamental specifications and requirements required to create a user-friendly online shopping platform. This SRS provides the foundation for establishing an effective and user-friendly online shopping environment for administrators and users alike. Designing, creating, and implementing an online shopping system that meets the various needs of modern customers is the primary objective of this project.

This system will give customers effective order management, a large selection of products, user-friendly interfaces, and safe payment options. Administrators will also have access to a full suite of tools for managing sales and inventory. All things considered, this SRS provides the development team with a thorough set of guidelines to help them create a dependable, scalable, and user-focused online shopping platform that satisfies the requirements and expectations of administrators and customers alike.

**FUNCTIONAL REQUIREMENT:**

**User Management Module:**

* Registration: Users should be able to create accounts by providing necessary details.
* Login: Registered users should be able to log in securely.
* Profile Management: Users should be able to update their profiles, including personal information, addresses, and payment details.
* Password Management: Users should have the option to reset their passwords securely.

**Product Management Module:**

* Product Catalog: Display a categorized list of available products with details such as name, description, price, and images.
* Search and Filter: Allow users to search for products based on keywords and apply filters for categories, price range, brand, etc.
* Product Detail: Display detailed information about each product, including specifications, reviews, and ratings.
* Inventory Management: Ensure accurate tracking of product availability and manage stock levels.

**Shopping Cart Module:**

* Add to Cart: Allow users to add items to their shopping carts.
* Modify Cart: Enable users to update the quantity of items or remove items from the cart.
* Save Cart: Provide the option for users to save their carts for future purchases.

**Checkout and Payment Module:**

* Checkout Process: Guide users through a seamless checkout process with multiple steps, including shipping address, shipping method, and payment method.
* Payment Gateway Integration: Integrate with secure payment gateways to process transactions using credit/debit cards, net banking, or other payment methods.
* Order Confirmation: Provide users with a confirmation message and email receipt after completing the purchase.

**Order Management Module:**

* Order History: Allow users to view their order history, including order status, tracking information, and invoices.
* Order Tracking: Provide real-time tracking information for orders in transit.
* Order Processing: Enable administrators to manage orders, update order status, and handle order cancellations or returns.

**User Feedback and Ratings Module:**

* Product Reviews: Allow users to leave reviews and ratings for products they have purchased.
* Seller Ratings: Enable users to rate and provide feedback on sellers/vendors.
* Feedback Management: Provide administrators with tools to moderate and respond to user feedback.

**Admin Management Module:**

* User Administration: Allow administrators to manage user accounts, including registration approval, account suspension, or deletion.
* Product Administration: Enable administrators to add, edit, or remove products from the catalog.
* Order Administration: Provide tools for administrators to manage orders, handle disputes, and generate reports on sales and inventory.

**NON-FUNCTIONAL REQUIREMENTS:**

A system's overall performance, security, usability, and maintainability are all dependent on factors that are not directly related to its purpose. These factors are known as non-functional requirements for online shopping systems.

**Performance:**   
  
A large number of concurrent users should not cause the system to perform noticeably worse. Response times for necessary features like searching, browsing, and checking out should fall within reasonable bounds.   
  
**Scalability:**  
  
Without requiring a substantial redesign, the system should be scalable to support an increase in the number of users, goods, and transactions over time.   
Both horizontal (adding more servers or instances) and vertical (adding more resources to already-existing components) scaling options should be part of a scalable system.

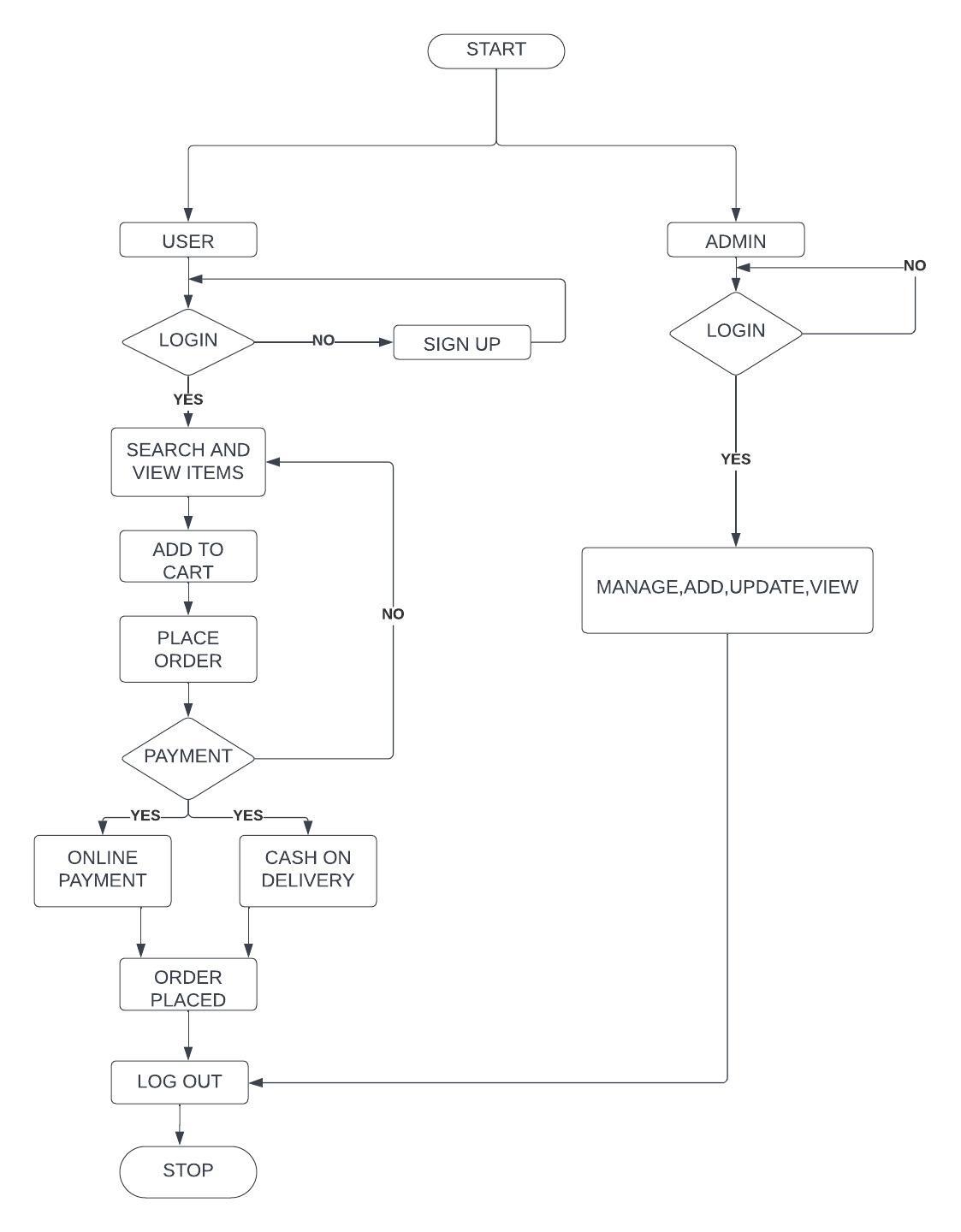
**Dependability:**   
  
There should be very little downtime for maintenance or unplanned malfunctions in the system. To guarantee data integrity and availability, procedures like redundancy, failover mechanisms, and frequent backups should be implemented.

**Security:**   
  
Strong security features should be implemented by the system to safeguard user data, including payment and personal information. To avoid data interception, all communications should use Secure Socket Layer (SSL) encryption.   
To guarantee that only authorised users have access to sensitive functions and data, authentication mechanisms should be put in place.

**Usability:**   
  
The user interface ought to be simple to use, intuitive, and consistent across various browsers and devices. It is important to adhere to accessibility guide lines (such as WCAG) to make sure that individuals with disabilities can utilise the system. Error messages ought to be understandable and instructive, offering users guidance on how to fix problems.

**Compatibility**:   
  
Customers should be able to use the system on a variety of devices (such as laptops, desktop computers, tablets, and smartphones) and browsers (such as Chrome, Firefox, Safari, and Edge).

**FLOW CHART:**



Login: The user is prompted to log in or sign up if they are new to the platform.

Search: Users have the option to search for specific products using keywords or filters.

View Product Details: Upon selecting a product, the user can view its details including description, price, and availability.

Add to Cart: Users can add desired products to their shopping cart.

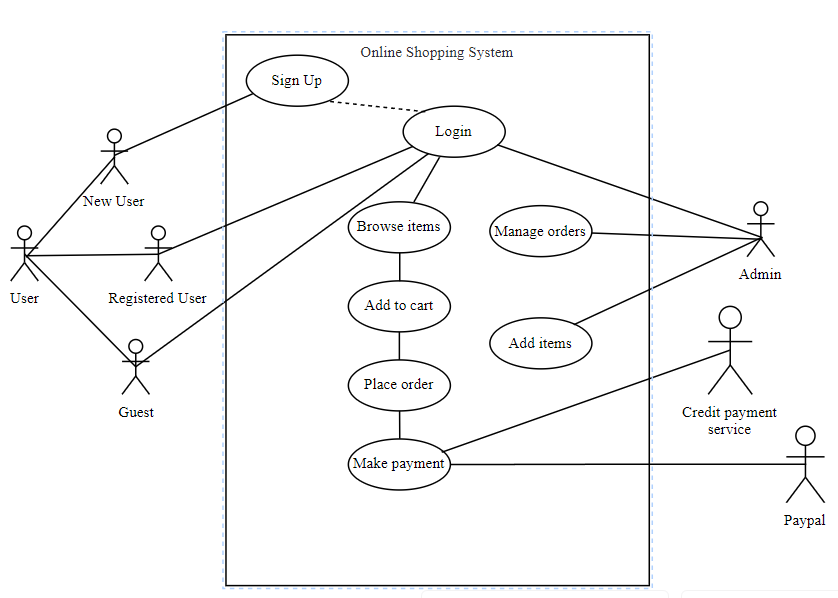
Proceed to Checkout: Once the user has finalized their selections, they proceed to the checkout process.

Select Payment Method: Users choose their preferred payment method (credit card, PayPal, etc.).

Place Order: Users confirm their order and complete the payment process.

**Use case Diagram:**

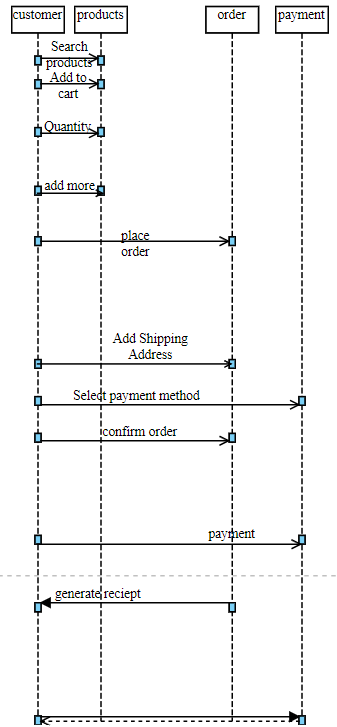
An essential tool for system design is the use case diagram, which shows users' interactions with a system visually. It acts as a road map for knowing a system's functional requirements from the viewpoint of its users, facilitating stakeholder communication and directing the development process.



1)Sign Up if New Customer: The user enters their email address to register for the system.   
2)Login if a Registered User: To access the system, the customer must provide their login credentials.   
3)Browse Items: The online store's inventory is visible to customers.   
4)View Item Details: A customer can view comprehensive details regarding a certain item.   
5)Add Item to Wishlist: A customer may list things they would want to have in the future.   
6)Place Order: After verifying the items of their shopping cart, the customer places an order.   
7)Make Payment: In order to finalise the purchase, the customer gives payment details.   
8)Manage items (Admin): The online store's administrator can add, edit, or remove things.

**Sequence Diagram:**

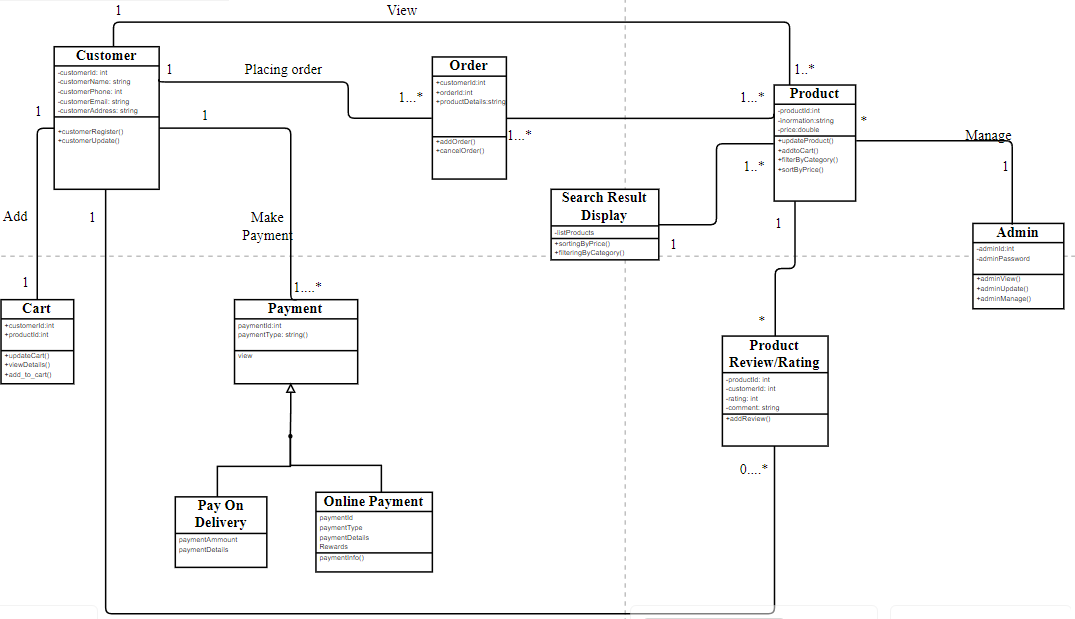
UML Sequence Diagrams are interaction diagrams that show the steps involved in performing an operation. They show how items interact with one another when working together. Sequence diagrams are time-focused and represent the sequence of the interaction by displaying the time, what messages are sent, and when on the vertical axis .



**Class Diagram:**

The primary building block of object-oriented modelling frequently shows the various objects in a system, together with their various attributes, functions, and interactions with one another. These building blocks are known as Class Diagram.

Aggregation and Multiplicity are two important points that need to take into consideration while designing a Class Diagram.



1.Customer Class:

* Attributes: customerId, customerName, customerEmail, customerAddress
* Functions: customerRegister(), customerUpdate()

2.Cart Class:

* Attributes: customerId, productId
* Functions: updateCart(), viewDetails(), add\_to\_cart()

5.Order Class:

* Attributes: productDetails, customerID, productID
* Functions: addOrder(), cancelOrder()

6.Payment Class:

* Attributes: paymentID, paymentType
* Functions: makePayment(), verifyPayment(), generateReceipt()

7.Product Class:

* Attributes: productId, information, price
* Functions: filterByCategory(), sortByPrice(), updateProduct()

8.Admin Class:

* Attributes: adminId, adminPassword
* Functions: adminView(), adminManage(), adminUpdate()

**Testing:**

Software testing is the process of executing with the goal of identifying software bugs. Software testing serves as a last assessment of earlier software phases, such as specification, design, code creation, etc., and ensures the software's quality.

Unit testing:

Unit testing focuses the verification effort on the software design's smallest unit, which is a module or component. Unit testing is a dynamic verification approach that involves the compilation and execution of the programme. Parallel to the coding process is the unit testing phase. Software modules or pieces are tested individually using unit testing. I've tested every view and module of the programme separately.

Integrating testing:

Testing for issues resulting from component interaction is done through integration testing, which examines a system made up of several modules. It is recommended that integration testing be derived from the system specification. It is necessary to integrate and test a minimum configuration first. In my project, I began construction and testing with atomic modules and conducted integration testing in a bottom-up manner.

Performance testing:

To make sure the system can manage several users at once, carry out load testing. Examine the important functions' reaction times at both normal and high loads. Verify that the system is capable of managing a sizable number of users and products in the database.

Security testing:

Look for issues like SQL injection, and other security risks. Make sure that private user information is safely stored. Check the system's ability to handle illegal access.

Functional Testing:

Check that all of the predefined functionalities—such as product browsing, account creation, cart management, order placing, etc.—function as designed by conducting functional testing. Check the effectiveness and correctness of the search functionality. Verify the procedures for user identification and registration. Make sure a safe login and exit.

Not-functional Testing:

Verify the system's responsiveness in a range of network scenarios. Verify that the system works with various browsers and devices. Verify that the system complies with security specifications (safe payment gateways, SSL encryption).