Software Requirements Specification

Table of contents

1.Abstract:

The Online Shopping System Software Requirements Specification (SRS) document serves as a comprehensive guide for the development and implementation of an efficient and user-friendly online shopping platform. This abstract provides a succinct overview of the key components and objectives outlined within the SRS document. The primary objective of the Online Shopping System is to create a seamless and intuitive online shopping experience for users, facilitating the purchase of goods and services from the comfort of their homes. The SRS document delineates the functional and non-functional requirements necessary to achieve this objective. Functional requirements encompass the core functionalities of the system, including user registration and authentication, browsing and searching for products, adding items to a shopping cart, secure online payment processing, order management, and user feedback mechanisms. Additionally, the document outlines requirements for administrative functionalities such as inventory management, sales analytics, and content management. Non-functional requirements address performance, security, reliability, and usability aspects of the system. This includes specifications for system responsiveness, scalability to accommodate varying loads, data encryption to ensure secure transactions, backup and recovery mechanisms to safeguard against data loss, and an intuitive user interface design to enhance user satisfaction. The SRS document also defines system interfaces, including external interfaces with payment gateways, shipping providers, and third-party services, as well as internal interfaces between system components. Moreover, it delineates constraints and assumptions guiding the development process, such as technology stack preferences, budgetary constraints, and regulatory compliance requirements.

2.Functional Requirements:

2.1 User Registration and Authentication:

- Users should be able to create accounts with unique usernames and passwords.
- The system should verify user credentials during the login process.
- Users should have the option to reset passwords in case of forgetfulness or security concerns.

2.2 Product Browsing and Searching:

- Users should be able to browse products by categories, brands, or other relevant filters.
- The system should support a search functionality allowing users to find products based on keywords or attributes.
- Search results should be presented in a clear and organized manner, with relevant product information displayed.

2.3 Product Display:

- Each product should have a detailed description, including images, prices, sizes (if applicable), and other relevant attributes.
- Users should be able to view product reviews and ratings submitted by other customers.

2.4 Shopping Cart Management:

- Users should be able to add items to their shopping carts from product pages.
- The system should allow users to view and modify the contents of their shopping carts before proceeding to checkout.
- Users should have the option to remove items from their carts or update quantities as needed.

2.5 Checkout Process:

- The system should guide users through a seamless and intuitive checkout process.
- Users should be prompted to provide shipping and billing information.
- Multiple payment options (credit/debit cards, PayPal, etc.) should be supported.
- The system should calculate and display the total order amount, including taxes and shipping fees.

2.6 Order Management:

- Users should receive confirmation emails after successfully placing orders.
- Administrators should have access to an order management interface to view, process, and track orders.
- Users should be able to track the status of their orders through their accounts.

2.7 User Feedback Mechanism:

- Users should be able to submit reviews and ratings for products they have purchased.
- The system should display aggregated ratings and reviews on product pages.
- Administrators should have the ability to moderate and manage user-generated content.

2.8 Inventory Management:

- Administrators should be able to add, update, and remove products from the inventory.
- The system should track product availability and prevent users from purchasing out-of-stock items.
- Low stock notifications should be sent to administrators when inventory levels fall below predefined thresholds.

3.Non Function Requirements:

3.1.Performance:

- The system should be capable of handling concurrent user interactions efficiently without significant performance degradation.
- Page load times should be minimized to ensure a responsive user experience.
- Response times for critical operations such as search, adding items to the cart, and checkout should be within acceptable limits.

3.2.Scalability:

- The system should be designed to scale horizontally to accommodate increasing numbers of users and transactions.
- Load balancing mechanisms should be implemented to distribute traffic evenly across multiple servers or instances.
- Database scalability should be ensured to handle growing volumes of product data and user information.

3.3.Security:

- User authentication and session management should be implemented securely to prevent unauthorized access and data breaches.
- Personal and financial information should be encrypted during transmission and storage to protect user privacy.
- The system should employ mechanisms to detect and prevent common security threats such as SQL injection, cross-site scripting (XSS), and CSRF attacks.

3.4.Reliability:

- The system should be highly available, with minimal downtime for maintenance and updates.
- Failover mechanisms should be in place to ensure continuity of service in the event of hardware or software failures.
- Data integrity measures should be implemented to prevent data loss or corruption.

3.5.Usability:

- The user interface should be intuitive and easy to navigate, catering to users of varying levels of technical expertise.
- Accessibility standards should be followed to ensure that the system is usable by individuals with disabilities.
- Error messages and prompts should be clear and informative, guiding users in resolving issues encountered during their shopping experience.

3.6.Compatibility:

- The system should be compatible with a wide range of web browsers and devices, including desktops, laptops, tablets, and smartphones.
- Compatibility testing should be conducted to ensure consistent functionality and appearance across different platforms and screen sizes.

3.7.Maintainability:

- The system should be modular and well-documented to facilitate future updates, enhancements, and bug fixes.
- Code readability and consistency should be maintained to ease troubleshooting and code maintenance tasks.

- Version control and change management practices should be adopted to track and manage system changes effectively.

4.Design:

4.1.High-Level Design:

- System Architecture: The system will follow a client-server architecture, with a web-based frontend for users and an application server handling business logic and data processing.
- Components: Major components include the user interface layer, application server, database server, and external services/interfaces for payment processing and shipping.
- Communication Protocols: HTTP/HTTPS will be used for communication between the client and server components. RESTful APIs may be employed for interactions between different system modules.
- Scalability: The system will be designed to scale horizontally by adding more application and database servers as traffic increases.
- Security Measures: Security features such as encryption, firewalls, and intrusion detection systems will be implemented to protect data and prevent unauthorized access.

4.2.Low-Level Design:

4.2.1.User Interface Layer:

- Frontend technologies such as HTML, CSS, and JavaScript frameworks (e.g., React, Angular) will be used to develop the user interface.
- Responsive design principles will be applied to ensure compatibility with various devices and screen sizes.
- UI components will be modularized for reusability and maintainability.

4.2.2.Application Server:

- Business logic will be implemented using a server-side programming language such as Python, Java, or Node.js.
- The server will handle user authentication, product management, shopping cart operations, order processing, and other core functionalities.
- RESTful APIs will be defined to enable communication between the frontend and backend components.

4.2.3.Database Layer:

- A relational database management system (RDBMS) such as MySQL, PostgreSQL, or SQL Server will be used to store product data, user information, and order details.
- Database schema will be designed to ensure data integrity and optimize query performance.
- Indexing and caching mechanisms may be employed to improve database performance.

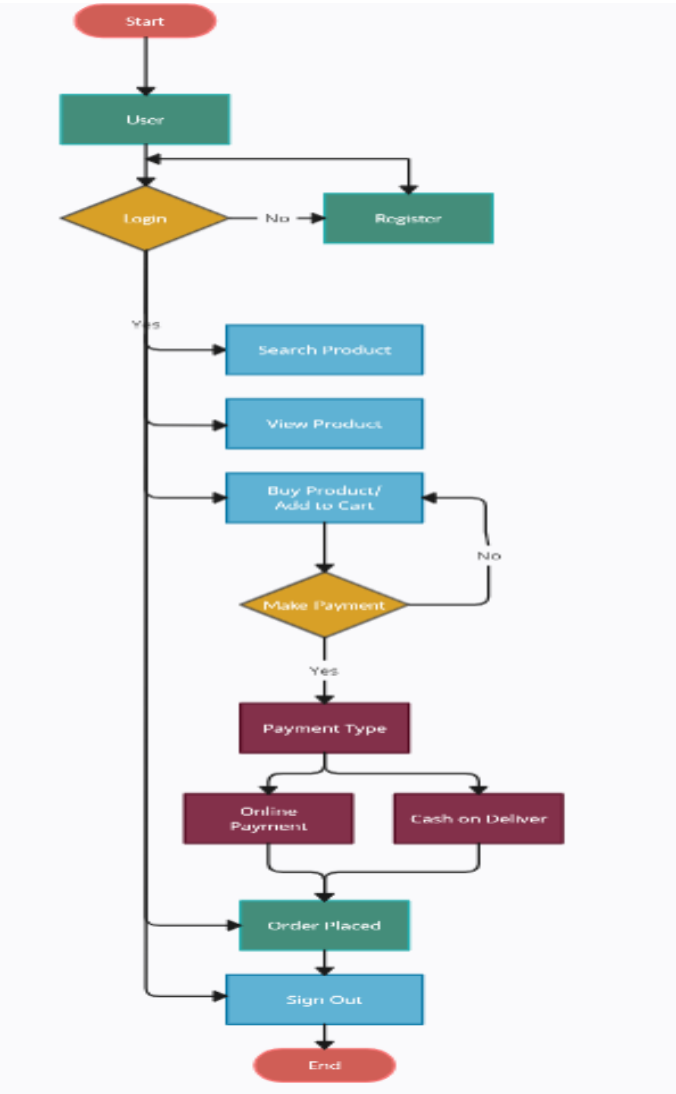
4.2.4.External Services/Interfaces:

- Integration with third-party services/interfaces will be implemented for payment processing, shipping, and other external functionalities.
- APIs provided by payment gateways (e.g., PayPal, Stripe) and shipping carriers (e.g., UPS, FedEx) will be utilized to handle payment transactions and order fulfillment.

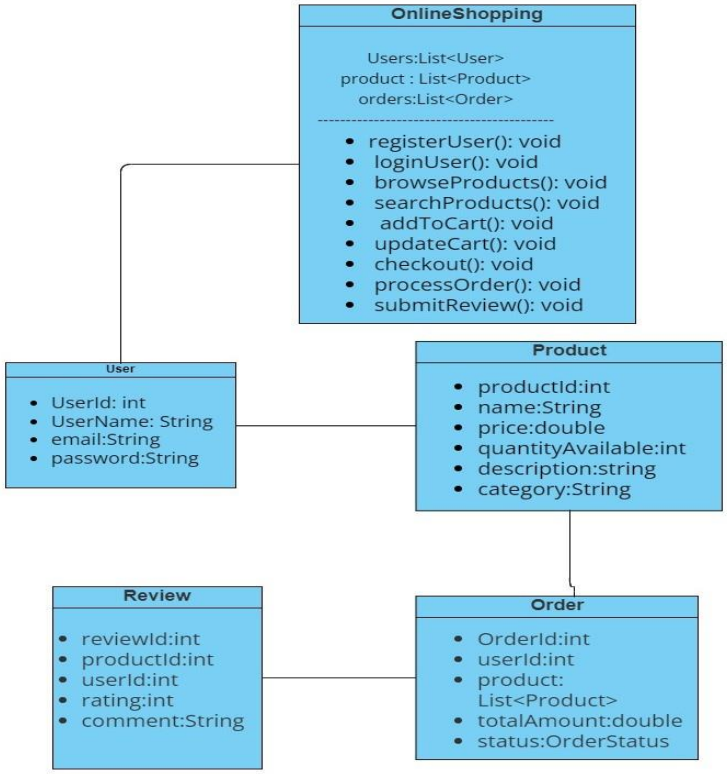
4.2.5.Security Implementation:

- HTTPS protocol will be enforced to encrypt data transmitted between the client and server.
- User authentication will be implemented using secure mechanisms such as JSON Web Tokens (JWT) or OAuth.
- Input validation and sanitization techniques will be employed to prevent injection attacks (e.g., SQL injection, XSS).
- Role-based access control (RBAC) will be implemented to restrict access to sensitive functionalities and data.

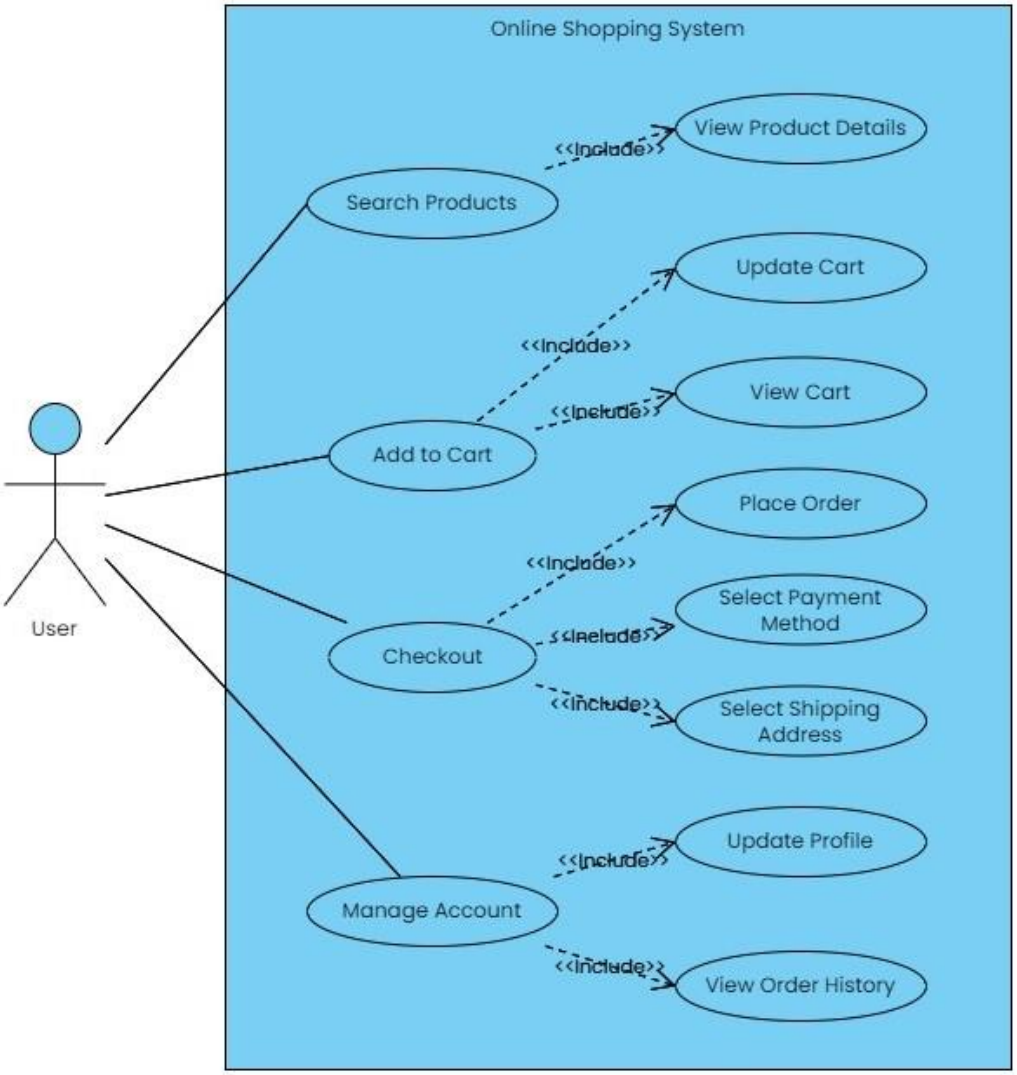
5.Flowchart:


Start
User
Login
No
Register
Yes
Search Product
View Product
Buy Product/
Add to Cart
No
Make Payment
Yes
Payment Type
Online
Payment
Cash on Deliver
Order Placed
Sign Out
End

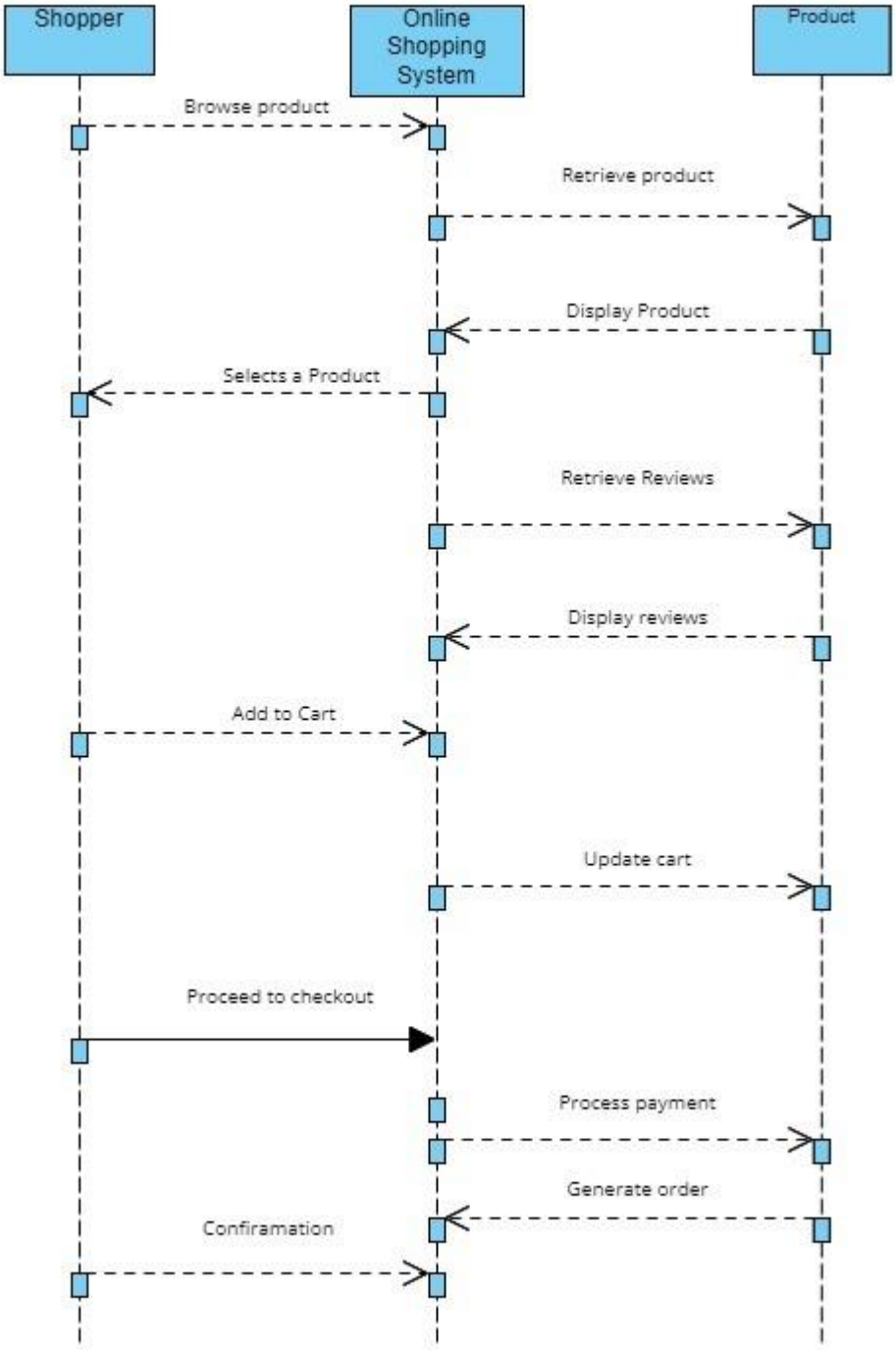
6.Class Diagram:



7.Use case diagram:



8.Sequence Diagram:



9.Test cases:

- Test Case 1: Verify that a user can successfully register with valid credentials.
- Test Case 2: Attempt registration with invalid or duplicate email addresses to ensure proper error handling.
- Test Case 3: Check that all mandatory fields (e.g., username, email, password) are required for registration.
- Test Case 4: Verify that registered users can log in with valid credentials.
- Test Case 5: Attempt login with incorrect username/password combinations to validate error handling.
- Test Case 6: Test login functionality across different devices and browsers to ensure compatibility.
- Test Case 7: Verify that products are displayed correctly with accurate details (e.g., name, price, image).
- Test Case 8: Check that users can browse products by categories and filters.
- Test Case 9: Ensure pagination works correctly for large product catalogs.
- Test Case 10: Verify that the search function returns relevant results based on keywords.
- Test Case 11: Test searching with misspelled or partial keywords to validate search accuracy.
- Test Case 12: Check that search results are displayed in a clear and organized manner.
- Test Case 13: Verify that users can add products to their shopping cart from product pages.
- Test Case 14: Attempt to add out-of-stock items to the cart to ensure proper inventory management.
- Test Case 15: Check that the shopping cart updates dynamically to reflect added items.
- Test Case 16: Verify that users can view and modify the contents of their shopping carts.
- Test Case 17: Test updating quantities and removing items from the cart to ensure proper functionality.
- Test Case 18: Check that the total order amount is accurately calculated based on cart contents.
- Test Case 19: Verify that users can proceed through the checkout process smoothly.
- Test Case 20: Test checkout with different payment methods (e.g., credit card, PayPal) to ensure compatibility.
- Test Case 21: Check that users receive confirmation emails after successful order placement.
- Test Case 22: Verify that administrators can view and manage orders from the admin interface.
- Test Case 23: Test order processing and status updates to ensure accurate order tracking.
- Test Case 24: Check that users can view the status of their orders through their accounts.
- Test Case 25: Verify that users can submit reviews and ratings for products they have purchased.