*Software Requirements Specification*

*-*

**Author: Hubert Veyannie V**

**Date:** March 19, 2024

1. **Abstract…………………………………………………………………………………………………….3**
2. **Requirements…………………………………………………………………………………………….3**
   1. Functional Requirements…………………………………………………………………………………………3
   2. Non – Functional Requirements………………………………………………………………………………4
3. **Design………………………………………………………………………………………………………4**
   1. High level design………………………………………………………………………………………………….….4
      1. System Architecture……………………………………………………………………………………...4
      2. Home and search flow........................................................................5
      3. Purchase and checkout flow................................................................6
   2. Low level design........................................................................................7
4. **Flowchart............................................................................................................7**
5. **Use case diagram................................................................................................9**
6. **Class case diagram..............................................................................................9**
7. **Sequence diagram.............................................................................................11**
8. **Test cases..........................................................................................................12**
   1. Test Case – User Registration.....................................................................12
   2. Test Case – User Login..............................................................................12
   3. Test Case – Browse Items.........................................................................12
   4. Test Case – Search Items..........................................................................12
   5. Test Case – View Item Details....................................................................12
   6. Test case – Add Item to cart......................................................................12
   7. Test Case – View Shopping cart..................................................................12
   8. Test Case – Update Item Quantity in Cart....................................................13
   9. Test Case – Remove Item from cart............................................................13
   10. Test Case – Checkout.............................................................................13
9. **Abstract**

The online shopping system is a comprehensive web-based application designed to transform the shopping experience for customers and streamline the selling process for sellers. This innovative platform provides a virtual marketplace where customers can browse through a wide array of items from the comfort of their homes.

The system is equipped with a user-friendly interface that allows customers to view detailed descriptions of each item, including price, specifications, and customer reviews. This feature enables customers to make informed decisions about their purchases. Once a customer selects an item, they can easily add it to their shopping cart. The shopping cart is a virtual basket where customers can store their selected items before proceeding to checkout.

The checkout process is designed to be smooth and efficient. Customers can review their selected items, make modifications if necessary, and then proceed to make a purchase. The system supports various payment methods, providing flexibility and convenience to the customers.

In addition to facilitating a seamless shopping experience for customers, the online shopping system also serves as a robust management tool for sellers. Sellers can easily manage their inventory, track sales, and handle orders through the system’s interface. The inventory management feature allows sellers to add new items, update existing item details, and remove items from their inventory.

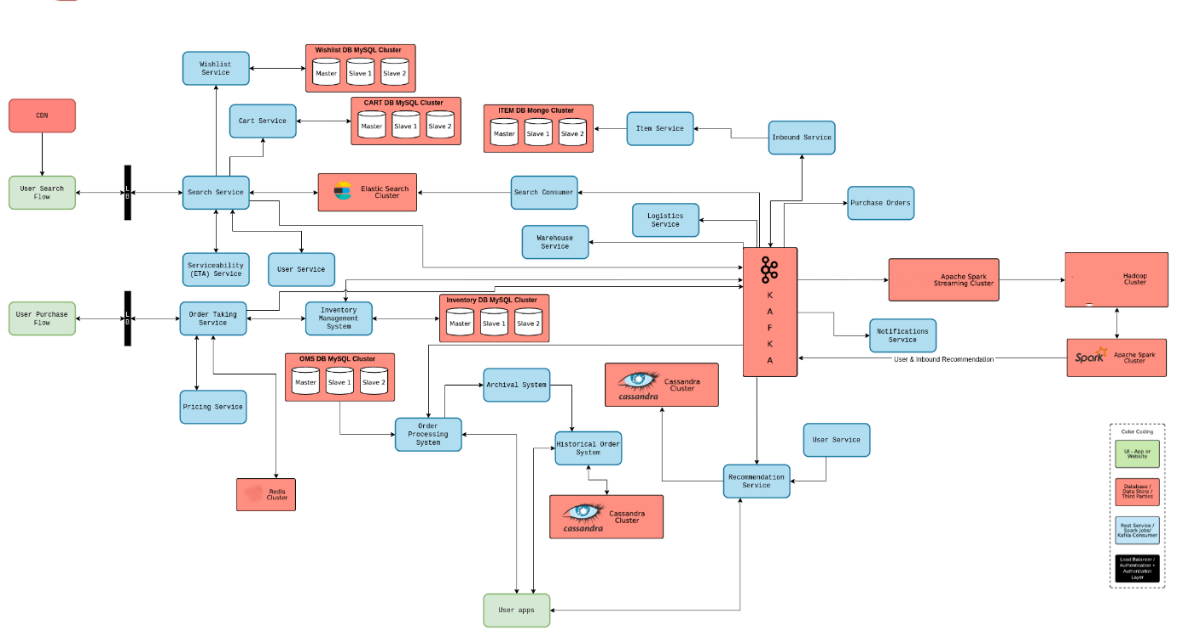
1. **Requirements**
   1. **Functional Requirements**

* User Registration: Users can create an account providing necessary details.
* User Login: Registered users can log in to the system.
* Browse Items: Users can browse items by category.
* Search Items: Users can search for items using keywords.
* View Item Details: Users can view detailed information about an item.
* Add to Cart: Users can add items to a shopping cart.
* Checkout: Users can purchase items in their shopping cart.
* Order History: Users can view their past orders.
* Inventory Management: Sellers can add, update, or remove items from their inventory.
* Order Management: Sellers can view and manage orders.
  1. **Non – Functional Requirements**
* Performance: The system should be able to handle a high number of users and transactions.
* Security: User data should be stored securely, and transactions should be encrypted.
* Usability: The system should be user-friendly with an intuitive interface.
* Scalability: The system should be able to scale to accommodate growth.
* Availability: The system should be available 24/7.

1. **Design**

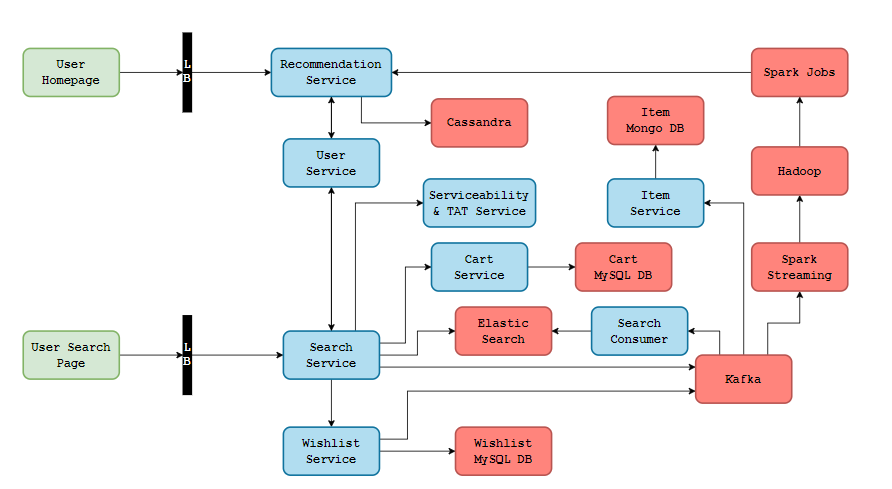
**3.1. High level design**

* + 1. **System Architecture**

****

Now for a system that handles such high traffic, especially during huge sales, meeting all three non-functional requirements might be difficult. But not everything needs to be always available, have low latency, and be extremely consistent. For example, payment and inventory systems should be highly consistent even at the cost of availability, Search needs to be highly available even if it is slightly inconsistent. Most user-facing components should have a low enough latency.

* + 1. **Home and Search flow**

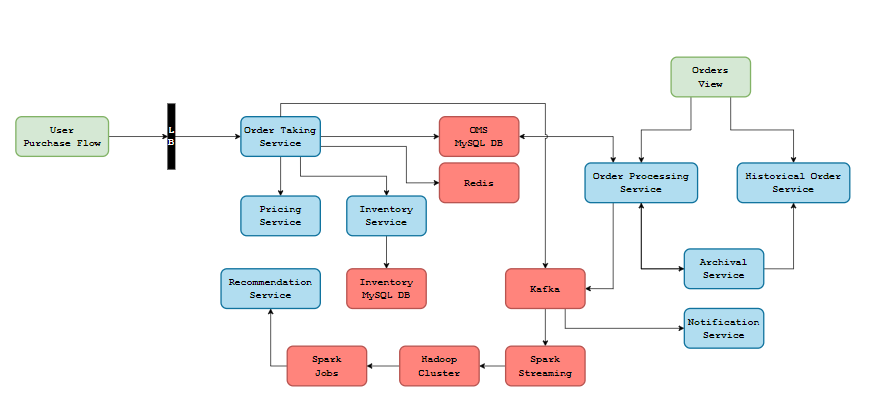
****

There will be two UIs that we will offer, a home screen which will have some recommendations, personalized or general depending on if it is a returning user or a new user, and a search page where users can see results based on some search text.

In an online shopping system akin to Amazon, managing multiple suppliers is streamlined through ‘Inbound Services’ that interface with supplier systems to retrieve updated data. This data, including new suppliers or inventory items, is funneled into the system via Kafka events, ensuring users have access to the latest information. The system includes an ‘Item Service’ that listens to Kafka for new item data, which is stored in MongoDB due to its unstructured nature, varying across different products. Additionally, a ‘Search Consumer’ processes and formats new item data for storage in an ElasticSearch database, chosen for its text search efficiency and fuzzy search capabilities, enhancing the user experience. This setup ensures that newly added items are promptly searchable, maintaining a seamless flow of information from suppliers to consumers. Search Service, powered by ElasticSearch, will offer APIs for product filtering, sorting, and searching, ensuring users only see deliverable items. This is achieved by interfacing with a Serviceability and TAT service that checks delivery feasibility from the warehouse to the user’s location, including route viability and estimated delivery dates. Concurrently, a Kafka service linked to a Spark streaming consumer will produce real-time analytics on consumer behavior and trends, feeding into a Hadoop cluster for advanced recommendation algorithms. These insights will be used by a Spark cluster to communicate with a recommendation service, providing personalized product suggestions based on user activity and preferences, ensuring relevant recommendations across different product categories.

The next component is the User Service. It is a repo of all users and also provides APIs to fetch, update, add, and delete users from our system. It sits on a MySQL database and maintains a Redis cache. So let’s say our search service wants to fetch a user’s pin code to communicate to the serviceability service, user service will first check in the Redis, if Redis doesn’t have the information it will look up in the MySQL database, get the user’s information, store it in Redis for further use and return it to the search service.

* + 1. **Purchase and checkout flow**

****

In an online shopping system, when a user places an order, the request is sent to an Order Taking Service, part of a larger Order Management System. This system operates on a MySQL database, which houses multiple tables such as customer, item, and order tables. To prevent multiple users from ordering the last item in stock due to a delay in database updates, the system uses the ACID properties of relational databases. When the Order Taking Service is invoked, a record is created in Redis with an order id, creation date and time, and an expiry time. The status of the order is initially set to “created”. The system then calls the Inventory Service to update the inventory.

Once the inventory is successfully updated, the Order Taking Service communicates with the Payments Service, which interacts with the payment gateway to process the payment. There are three possible outcomes from this payment flow - success, failure, or no response. If the payment is successful, the order status is updated to “placed” and an event is fired to Kafka indicating that an order has been placed. If the payment fails, the order is canceled, the inventory count is reverted, and the order status is updated to “canceled”. A Reconciliation System ensures that the order count and inventory count are consistent.

Users can view their past and ongoing orders through the Orders View, which communicates with an intermediate service that fetches order information from the Order Processing Service and Historical Service. Additionally, whenever an order is placed or its status changes to “in transit” or “delivered”, the seller or the customer is notified through the Notification Service. This comprehensive system ensures a smooth and efficient online shopping experience.

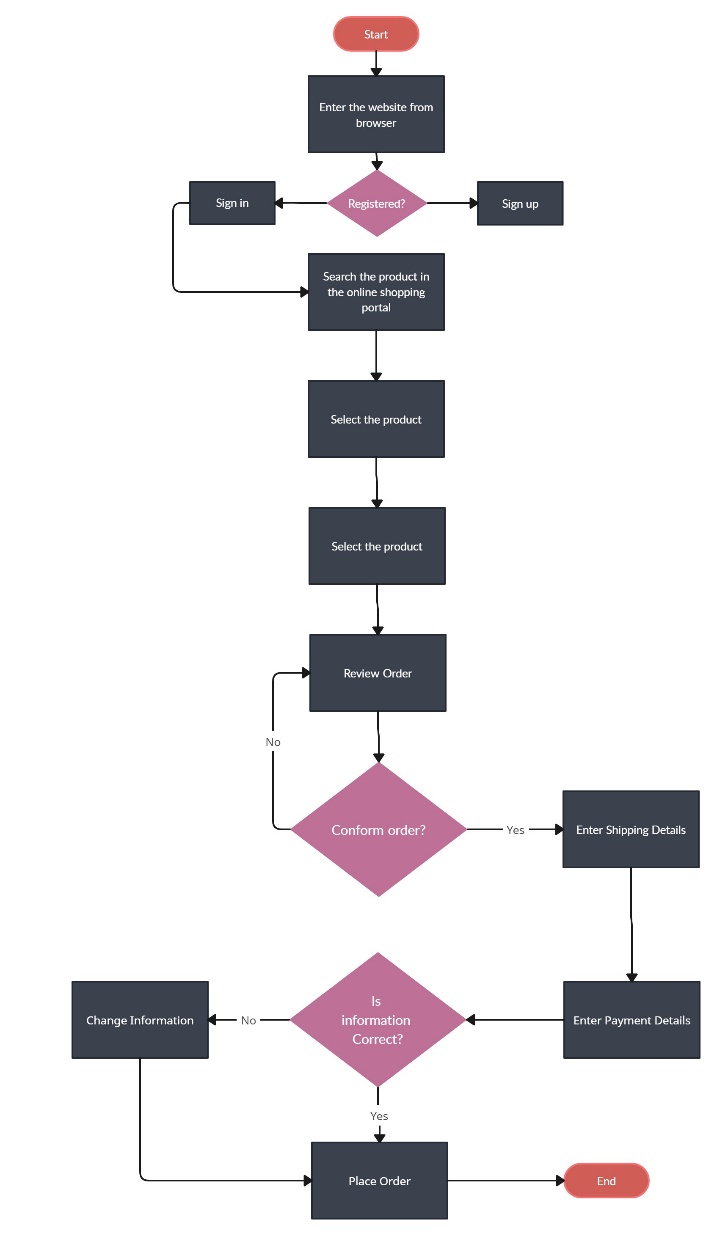
* 1. **Low level design**

The low level design includes the detailed descriptions of each module, their functions and the data flow between them. Some constraints are mentioned below.

* **User Class:** This class represents a user of the system. It has attributes such as username, password, email, and methods like login(), logout(), register(), updateProfile().
* **Product Class:** This class represents a product in the system. It has attributes like productId, productName, productDescription, productPrice, and methods like getProductDetails(), updateProductDetails().
* **ShoppingCart Class:** This class represents a shopping cart. It has attributes like cartId, userId, products (a list of Product objects), and methods like addProduct(), removeProduct(), updateQuantity(), getTotalPrice().
* **Order Class:** This class represents an order. It has attributes like orderId, userId, products (a list of Product objects), orderStatus, shippingAddress, and methods like placeOrder(), cancelOrder(), updateOrderStatus().
* **Payment Class:** This class handles the payment process. It has attributes like paymentId, orderId, paymentMethod, paymentStatus, and methods like makePayment(), refundPayment().
* **Shipping Class:** This class handles the shipping process. It has attributes like shippingId, orderId, shippingAddress, shippingStatus, and methods like updateShippingStatus(), getEstimatedDeliveryDate().

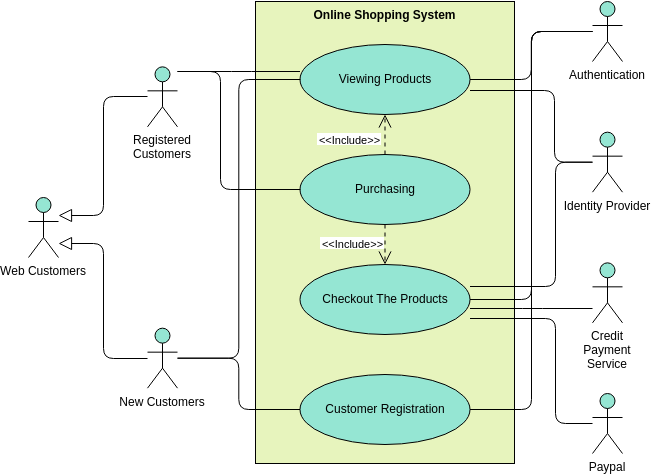
1. **Flowchart**

The following flowchart outlines the process of purchasing a book from the online shopping portal, starting from entering the website. This flowchart is a visual representation of the sequential steps involved in the online book purchase, including decision points leading to different paths based on the user’s choices. It is designed to provide a clear and logical progression from start to finish.

****

* Start: The process initiation point.
* Enter the Website: The user accesses the website from their browser.
* Registered: A decision point where if the user is already registered, they sign in; if not, they sign up.
* Search for Book: On the homepage, the user searches for the book of their choice.
* Select Book: The user selects their preferred book.
* Buy New or Old: A decision point asking if the user wants to buy a new book.
* If yes, the user selects a new book, adds it to the cart, and reviews the order.
* If no, the user selects an old book and enters shipping information.
* Information Correct: Another decision point to confirm the shipping information.
* If no, the user goes back to change the information.
* If yes, the user proceeds to select a payment method.
* Place Order: The user places the order.
* End: The conclusion of the process.

1. **Use Case diagram**

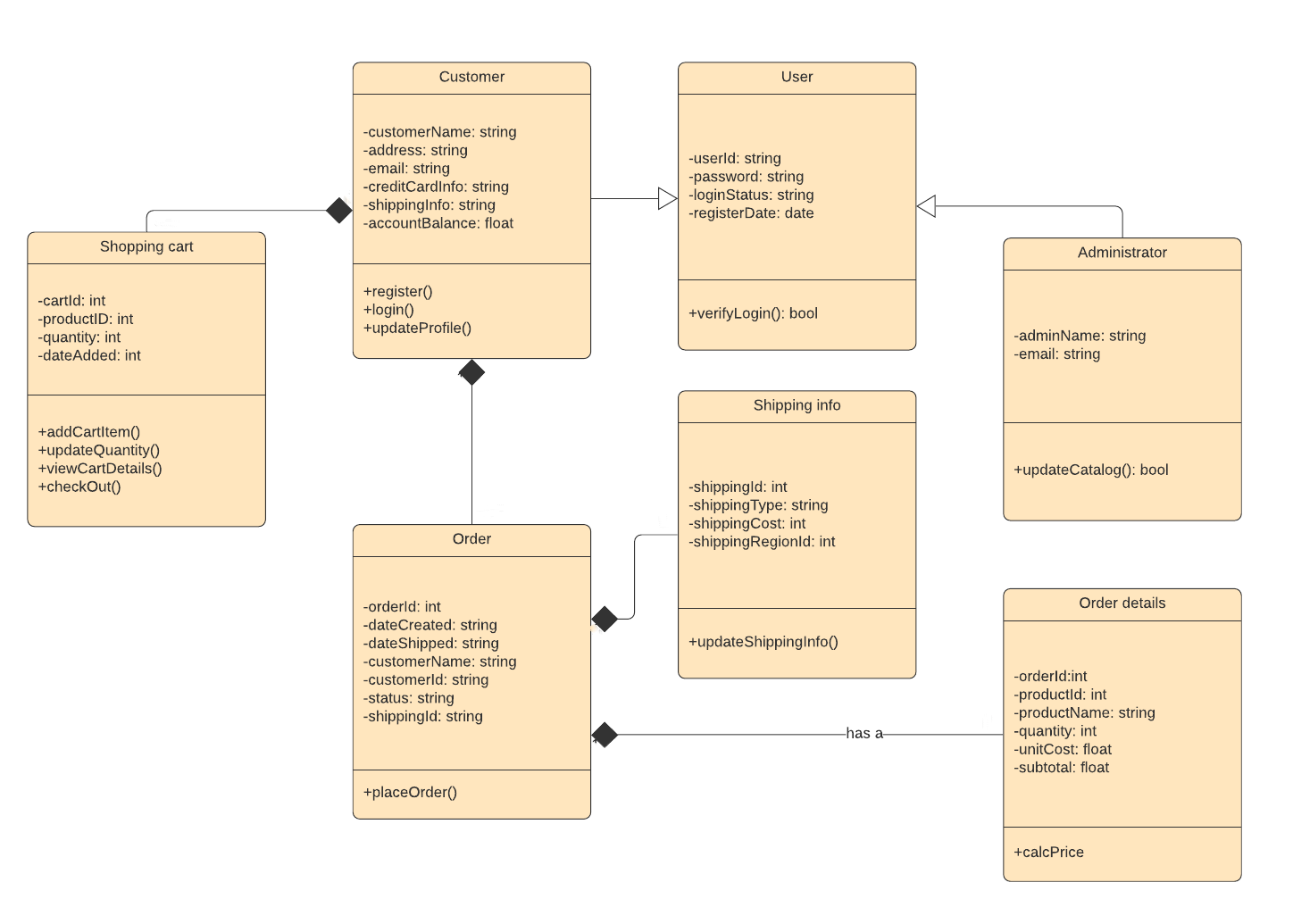
****

The use case diagram outlines the process flow from customer registration to product checkout, including various user types and authentication methods

* The central focus is labeled “Online Shopping System” and is outlined in a green rectangle.
* Inside this system, there are processes depicted as oval shapes: “Viewing Products,” “Purchasing,” “Checkout The Products,” and “Customer Registration.”
* Arrows labeled “<<includes>>” suggest a sequence of steps, connecting “Viewing Products” to “Purchasing” and “Purchasing” to “Checkout The Products.”
* Different customer types are represented on the left as stick figures: “Registered Customers,” “Web Customers,” and “New Customers.”
* On the right, stick figures represent different authentication methods: “Authentication,” “Identity Provider,” “Credit Payment Service,” and “Paypal.”
* Each customer type and authentication method is connected to the central online shopping system with arrows.

1. **Class diagram**

This Class diagram represents the structure of the online shopping system, detailing the class and its attributes, methods and the relationships between them.



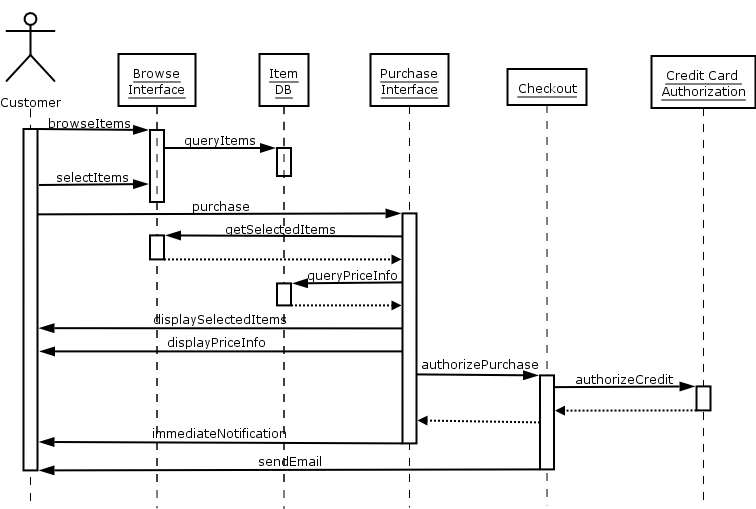
* **Customer Class:** Represents a customer with attributes like name, address, email, credit card information, shipping information, and account balance. It includes methods to register, login, and update the profile.
* **User Class:** A base class for users with attributes for user ID, password, login status, and registration date. It has a method to verify login credentials.
* **Administrator Class:** Represents an administrator with attributes for admin name and email. It includes a method to update the catalog.
* **Shopping Cart Class:** Contains attributes for cart ID, product ID, quantity, and date added. Methods include adding cart items, updating quantity, viewing cart details, and checking out.
* **Order Class:** Represents an order with a relationship to the Shopping Cart and Shipping Info classes. It would contain order-related attributes (not specified in the description).
* **Shipping Info Class:** Contains attributes for shipping ID, type, and cost. It is associated with the Order class.
* **Order Details Class:** Represents the details of an order and is likely to have a composition relationship with the Order class, indicating that Order Details cannot exist without an Order.

The diagram uses UML notation, with rectangles representing classes, and lines indicating relationships such as inheritance (User to Customer), associations (Customer to Shopping Cart), and aggregations (Order to Order Details with a diamond symbol).

This class diagram is a blueprint for how the online shopping cart system is structured in terms of object-oriented design. It provides a clear overview of the system’s classes and their interconnections, which is essential for understanding the system’s design and for further development

1. **Sequence diagram**

The sequence diagram illustrates the interactions between the objects in the online shopping process. This sequence diagram provides a clear visualization of the interactions between different parts of the online shopping system. It shows how each part contributes to the overall process of purchasing an item online. It also outlines the understanding the flow of operations and the roles of different parts of the system



* **Customer:** This is the user of the system who wants to purchase items online. The customer initiates the process.
* **Browse Interface:** This is the part of the system that allows the customer to browse items. It interacts with the Item Database to fetch item details.
* **Item DB (Database):** This is where all the item details are stored. When queried by the Browse Interface, it returns the requested item details.
* **Purchase Interface:** Once the customer selects items to purchase, this part of the system retrieves the selected item details and price information. It’s responsible for managing the purchasing process.
* **Checkout:** This is the process where the customer finalizes their purchase. It involves authorizing the purchase, which includes credit card authorization.
* **Credit Card Authorization:** This is an external system or service that verifies the credit card details provided by the customer and authorizes the transaction.

The sequence of interactions is as follows:

* The customer browses items using the Browse Interface.
* The Browse Interface queries the Item DB for item details.
* The customer selects items to purchase.
* The Purchase Interface retrieves the selected item details and price information.
* The customer initiates the checkout process.
* The Checkout process interacts with the Credit Card Authorization system to authorize the purchase.
* Upon successful authorization, an immediate notification is sent to the customer, followed by an email confirmation

1. **Test Cases**
   1. **Test Case – User Registration**

**Test Steps**: Navigate to the registration page, fill out the registration form with valid details, and submit the form.

**Expected Result:** The user account is created, and the user is logged in.

* 1. **Test Case – User Login**

**Test Steps:** Navigate to the login page, enter valid credentials, and submit the form.

**Expected Result:** The user is logged in and redirected to the homepage.

* 1. **Test Case – Browse Items**

**Test Steps:** Navigate to the homepage and browse through the listed items**.**

**Expected Result:** All available items are displayed**.**

* 1. **Test Case – Search Items**

**Test Steps:** Enter a search term in the search bar and submit the search.

**Expected Result:** Items matching the search term are displayed.

* 1. **Test Case – View Item Details**

**Test Steps:** Click on an item to view its details.

**Expected Result:** The item’s details are displayed, including price, description, and reviews.

* 1. **Test Case – Add Items to cart**

**Test Steps:** From the item details page, select a quantity and click the “Add to Cart” button.

**Expected Result**: The item is added to the shopping cart, and the cart’s item count is updated.

* 1. **Test Case – View shopping cart**

**Test Steps:** Navigate to the shopping cart page.

**Expected Result:** All items in the shopping cart are displayed with their details and total price.

* 1. **Test Case – Update item quantity in cart**

**Test Steps:** In the shopping cart, change the quantity of an item and update the cart.

**Expected Result:** The item’s quantity is updated, and the cart’s total price is adjusted.

* 1. **Test Case – Remove item from cart**

**Test Steps:** In the shopping cart, click the “Remove” button for an item.

**Expected Result:** The item is removed from the cart, and the cart’s item count and total price are updated.

* 1. **Test Case – Checkout**

**Test Steps:** Click the “Checkout” button in the shopping cart, enter valid shipping and payment information, and submit the order.

**Expected Result:** The order is submitted, the shopping cart is emptied, and a confirmation message is displayed.