# Ecommerce Website

## Functionalities

### User Management

Registration: Allow new users to create an account using their email

Login: Users should be able to securely login using their credentials.

### Product Catalog

* 1. Browsing: Users should be able to browse products by different categories.

Product Details: Detailed product pages with relevant information.

* 1. Search: Users must be able to search for products using keywords.

### Cart& Checkout

* 1. AddtoCart: Users should be able to add products to their cart.
  2. CartReview: View selected items in the cart with price, quantity, and total details.

Checkout: Seamless process to finalize the purchase, including specifying delivery address and payment method.

### Order Management

Order Confirmation: After making a purchase, users should receive a confirmation with order details.

* 1. Order History: Users should be able to view their past orders.

### Payment

Multiple Payment Options: Support for credit/debit cards, online banking, and other popular payment methods.

* 1. Secure Transactions: Ensure secure transactions using 3rd party app like Razorpay.
  2. Payment Receipt: Provide users with a receipt after a successful payment.

### Authentication

Secure Authentication: Ensure that user data remains private and secure during login and throughout their session.

Session Management: Users should remain logged in for a specified duration or until they decide to log out.

# High-LevelDesign(HLD)forEcommerceWebsite

## ArchitectureComponents

* + - LoadBalancers(LB)
    - APIGateway
    - Microservices
    - Databases(RelationalandNoSQL)
    - MessageBroker(Kafka)
    - Caching(Redis)
    - SearchandAnalytics(Elasticsearch)

## LoadBalancers(LB)

**Function**: Distribute incoming user requests across multiple server instances to balance load and ensure high availability.

## APIGateway

**Function**: Entry point for clients. Routes requests to the right microservices.

## Microservices Architecture

### User Service

* + - Handles user registration, login, profile management and password reset.
    - Uses MySQLas the primary database for structured user data.
    - Uses Kafkato communicate relevant user activities to other services(e.g.,a new user registration event can trigger welcome emails or offers).

### Product Service

* + - Manages product listings, details, categorization.

#### Uses My SQL.

#### Uses Redis for fast in memory product detail access.

* + - **Scope -** Incorporating Elastic searchfor fast product searches, providing features like full-text search and typo correction.

### Cart Service

* + - Manage user's shopping cart.
    - Uses MongoDBfor flexibility in cart structures.
    - Uses Redisfor fast, in-memory data access(e.g., to quickly retrieve a user’s cart).

### Order Service

* + - Handles order processing, history etc.

#### Uses MySQL.

* + - Communicates with Payment Service and User Management Service **through Kafka**

For order status updates, payment verifications, etc.

### Payment Service

* + - Manages payment gateways and transaction logs.

#### Uses MySQL.

* + - Once the payment is confirmed, it produces a message on Kafkato notify the Order Management Service.

### Notification Service:

* + - Manages email and potentially other notifications(e.g.,SMS).
    - **Consumes Kafka messages** for events that require user notifications(like registration confirmations, order updates).

## Databases

MySQL: For structured data.

MongoDB : For flexible, unstructured data.

## Kafka

Central message broker allowing asynchronous communication between microservices, ensuring data consistency, and acting as an event store for critical actions.

## CachingwithRedis

Primarily by Cart Service for faster response times.

## Typical Flow of Client Requests

Part 1

* User logs in and searches for a product.
* Request reaches Service Discovery.
* Service Discovery routes the search request to Product Service.
* Product Service queries its database and return the response.

Part 2

* User adds a product to the cart.
* Cart Service produces a message to Kafka about this action.

Part 3

* User checks out, triggering the Order Service.
* After placing the order, a message is sent to Kafka.
* Payment Service consumes the Kafka message to process payment.