Product Requirements Document (PRD) for Ecommerce Website

Functional Requirements

1. User Management

1.1. Registration: Allow new users to create an account using their email or social media profiles.
1.2. Login: Users should be able to securely log in using their credentials.
1.3. Profile Management: Users should have the ability to view and modify their profile details.
1.4. Password Reset: Users must have the option to reset their password through a secure link.

2. Product Catalog

2.1. Browsing: Users should be able to browse products by different categories.
2.2. Product Details: Detailed product pages with product images, descriptions, specifications, and other relevant information.
2.3. Search: Users must be able to search for products using keywords.

3. Cart & Checkout

3.1. Add to Cart: Users should be able to add products to their cart.
3.2. Cart Review: View selected items in the cart with price, quantity, and total details.
3.3. Checkout: Seamless process to finalize the purchase, including specifying delivery address and payment method.

4. Order Management

4.1. Order Confirmation: After making a purchase, users should receive a confirmation with order details.
4.2. Order History: Users should be able to view their past orders.
4.3. Order Tracking: Provide users with a way to track their order's delivery status.

5. Payment

5.1. Multiple Payment Options: Support for credit/debit cards, online banking, and other popular payment methods.
5.2. Secure Transactions: Ensure user trust by facilitating secure payment transactions.

5.3. Payment Receipt: Provide users with a receipt after a successful payment.

6. Authentication

6.1. Secure Authentication: Ensure that user data remains private and secure during login and throughout their session.
6.2. Session Management: Users should remain logged in for a specified duration or until they decide to log out.

High-Level Design (HLD) for Ecommerce Website

Architecture Components

- Load Balancers (LB)
- API Gateway
- Microservices
- Databases (Relational and NoSQL)
- Message Broker (Kafka)
- Caching (Redis)
- Search and Analytics (Elasticsearch)

1. Load Balancers (LB)

Function: Distribute incoming user requests across multiple server instances to balance load and ensure high availability.
Tool: Amazon Elastic Load Balancing (ELB).

2. API Gateway

Function: Entry point for clients. Routes requests to the right microservices, handles rate limiting, and manages authentication.
Tool: Kong.

3. Microservices Architecture

3.1 User Management Service

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

3.2 Product Catalog Service

- Manages product listings, details, categorization.
- **Uses MySQL**.
- **Incorporates Elasticsearch** for fast product searches, providing features like full-text search and typo correction.

3.3 Cart Service

- Manages user's shopping cart.
- **Uses MongoDB** for flexibility in cart structures.

- **Uses Redis** for fast, in-memory data access (e.g., to quickly retrieve a user's cart).

3.4 Order Management Service

- Handles order processing, history, and tracking.
- **Uses MySQL.**
- Communicates with Payment Service and User Management Service **through Kafka** for order status updates, payment verifications, etc.

3.5 Payment Service

- Manages payment gateways and transaction logs.
- **Uses MySQL.**
- Once the payment is confirmed, it produces a message **on Kafka** to notify the Order Management Service.

3.6 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).
- Integrates with third-party platforms like Amazon SES for actual email delivery.

4. Databases

MySQL: For structured data.
MongoDB: For flexible, unstructured data.

5. Kafka

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

6. Caching with Redis

Primarily by Cart Service for faster response times.

7. Elasticsearch

- Used by Product Catalog for fast and relevant product searches.

Typical Flow with Kafka & Elasticsearch Integration

Part 1

- User logs in and searches for a product.
- Request reaches LB, then passed to API Gateway.
- API Gateway routes the search request to Product Catalog Service.

- Product Catalog Service queries Elasticsearch for a fast product search.

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 Management Service.
- After placing the order, a message is sent to Kafka.
- Payment Service consumes the Kafka message to process payment.