# ****E-Commerce Application with Microservices****

## ****Project Overview****

The project involves developing a **Backend for Frontend (BFF)** for an e-commerce platform using **Java**, **Spring Boot**, and **Microservices Architecture**. The platform will provide a modular, scalable, and secure system that supports essential e-commerce functionalities such as product catalog management, customer management, order and cart processing, dynamic pricing, and payment handling.

The architecture will emphasize industry-standard microservices practices, including centralized configuration, service discovery, load balancing, API gateway integration, and secure inter-service communication. Each service will be self-contained, independently deployable, and communicate via REST APIs.

## ****Key Features****

### ****1. Product Management****

* Create and manage product types with dynamic attributes.
* Add, update, delete, and retrieve products and product variants.
* Provide search and filtering functionality for customers to browse products.

### ****2. Customer Management****

* Allow customer registration, login, and profile management.
* Provide role-based access control (Admin and Customer).
* Secure customer data and sessions using JWT-based authentication.

### ****3. Order and Cart Management****

* Enable customers to manage carts (add, update, remove items).
* Process orders from carts and maintain order history.
* Track order statuses (e.g., Processing, Shipped, Delivered).

### ****4. Dynamic Pricing Management****

* Configure unique price combinations for each product based on:
  + Customer group
  + Country
  + Currency
* Retrieve prices dynamically based on a given context using the following priority:
  + **Customer Group + Country + Currency**
  + **Customer Group + Currency**
  + **Country + Currency**
  + **Currency**
* Provide fallback mechanisms to ensure a price is always returned if possible.

### ****5. Payment Processing****

* Simulate integration with a third-party payment gateway.
* Handle payment statuses (Success, Failure) and retries.
* Notify the order service upon payment completion.

## ****Microservices and Their Responsibilities****

### ****1. Product Service****

* Manage product catalog, including types, attributes, and variants.
* Provide APIs for product creation, retrieval, updates, and deletion.
* Support search and filter operations for customer browsing.

### ****2. Customer Service****

* Handle customer registration, login, and profile management.
* Provide APIs for authentication and secure role-based access.
* Manage JWT-based session tokens for secure communication.

### ****3. Order Service****

* Manage customer carts and order processing.
* Maintain order history and status tracking.
* Integrate with the Pricing Service to fetch dynamic prices.

### ****4. Pricing Service****

* Configure and manage unique price combinations for products.
* Retrieve prices dynamically based on context with a fallback mechanism.
* Provide APIs to fetch, add, update, and delete prices.

### ****5. Payment Service****

* Simulate payment processing and gateway integration.
* Manage payment statuses and retries.
* Notify the order service upon successful or failed payments.

## ****System Architecture Overview****

### ****Key Architectural Components****

1. **Centralized Configuration Server**:
   * Store and manage configuration for all services centrally.
   * Enable dynamic updates without redeploying services.
2. **Service Registry**:
   * Maintain a registry of all services for discoverability.
   * Enable load balancing and health-check monitoring.
3. **API Gateway**:
   * Act as a single entry point for clients.
   * Handle routing, load balancing, and request aggregation.
   * Implement security features like token validation and rate limiting.
4. **Security**:
   * Use Spring Security for role-based access control.
   * Secure all APIs with JWT for inter-service communication.
   * Encrypt sensitive data in transit using HTTPS.

## ****High-Level Requirements****

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

#### **Core Functionalities**:

* **Product Service**:
  + Manage dynamic product types and attributes.
  + Provide APIs for product CRUD operations and search/filter capabilities.
* **Customer Service**:
  + Manage customer registration, authentication, and profiles.
  + Ensure secure role-based access to APIs.
* **Order Service**:
  + Provide cart management and order processing functionalities.
  + Maintain order history and enable status tracking.
* **Pricing Service**:
  + Configure and fetch dynamic prices based on customer group, country, and currency.
  + Support fallback mechanisms to handle missing price combinations.
* **Payment Service**:
  + Simulate payment gateway integration and manage payment retries.

#### **Integration Requirements**:

* Each service must communicate securely through REST APIs.
* API Gateway will aggregate requests and manage routing to the appropriate services.
* Services will be independently deployable and scalable.

### ****2. Non-Functional Requirements****

* **Performance**:
  + APIs must support efficient handling of large data sets with pagination.
  + Optimize pricing lookups with appropriate indexing and query design.
* **Scalability**:
  + Each microservice must scale independently to handle varying loads.
* **Reliability**:
  + Ensure fault tolerance using retries and circuit breakers for inter-service communication.
* **Security**:
  + Implement robust authentication and authorization mechanisms.
  + Protect data in transit with HTTPS and secure tokens.
* **Deployability**:
  + Containerize services using Docker and deploy them using Kubernetes.

## ****Brief Summary of Services****

|  |  |
| --- | --- |
| **Service Name** | **Description** |
| **Product Service** | Manage product catalog, types, attributes, and variants. |
| **Customer Service** | Handle customer registration, login, and profile management. |
| **Order Service** | Manage customer carts, process orders, and track order statuses. |
| **Pricing Service** | Configure and retrieve dynamic prices based on customer group, country, and currency. |
| **Payment Service** | Simulate payment gateway integration, handle payment statuses, and notify order service. |