eCommerce Application Project Documentation

1. Skill Description

- **Technologies Used: Java, Spring Boot, Spring Cloud, Spring Data JPA, MySQL/MariaDB, RESTful APIs**

2. Problem Statement

Develop a distributed eCommerce Application using Spring Microservices where:

- **Admin can manage products, orders, customers, and inventory.**
- **Users can browse products, place orders, and manage their accounts.**

3. Microservices Architecture

3.1 Service Registry & Discovery

- **Eureka Server:**
  - **A centralized service registry where all microservices register themselves.**
  - **Enables dynamic discovery of services.**

3.2 API Gateway

- **Spring Cloud Gateway:**
  - **Acts as the entry point for all client requests.**
  - **Routes requests to the appropriate microservice.**
  - **Provides cross-cutting concerns like security, rate limiting, and logging.**

3.3 Authentication Service

- **User Authentication & Authorization:**
  - **Manages user registration, login, and JWT token generation.**
  - **Handles roles for Admin and User to secure access to different parts of the application.**

3.4 Product Management Microservice

- **Product Catalog:**
  - **Responsible for managing product listings.**

- Provides APIs for adding, editing, deleting, and fetching product details.
- Handles categories, product descriptions, pricing, and availability.

3.5 Order Management Microservice

- **Order Processing:**
  - Manages the lifecycle of customer orders from creation to fulfillment.
  - Provides APIs for creating, viewing, updating, and canceling orders.
  - Handles payment processing and order status updates.

3.6 Inventory Management Microservice

- **Stock Control:**
  - Manages inventory levels for all products.
  - Provides APIs for tracking stock, updating quantities, and managing reordering processes.
  - Ensures that orders cannot be placed for out-of-stock items.

3.7 Customer Management Microservice

- **Customer Profiles:**
  - Manages customer information, including shipping addresses and order history.
  - Provides APIs for customers to update their profiles and view their past orders.

4. Project Flow

4.1 Admin Module

- **Admin Dashboard:**
  - A centralized interface for the Admin to manage products, orders, customers, and inventory.
  - Provides analytics and reports on sales, inventory levels, and customer activity.
- **Product Management:**
  - Admin can perform CRUD operations on products through the Product Management Microservice.

    - **Organize products into categories and manage their visibility on the platform.**
- **Order Management:**
    - **Admin can view and manage customer orders through the Order Management Microservice.**
    - **Update order statuses and manage refunds or cancellations as needed.**
- **Inventory Management:**
    - **Admin can monitor stock levels and update inventory through the Inventory Management Microservice.**
    - **Set up alerts for low stock levels and automate reordering processes.**
- **Customer Management:**
    - **Admin can view and manage customer profiles and their order history.**
    - **Handle customer queries and issues related to orders or accounts.**

4.2 User Module

- **User Registration & Authentication:**
    - **Users can register and log in using the Authentication Service.**
    - **JWT tokens are used to secure API access.**
- **Product Browsing:**
    - **Users can browse the product catalog through the Product Management Microservice.**
    - **Filter and search products by categories, price, and other attributes.**
- **Order Placement:**
    - **Users can add products to their cart and place orders through the Order Management Microservice.**
    - **Manage shipping information and select payment methods during checkout.**
- **Order Tracking:**
    - **Users can track their order status and view order history through the Order Management Microservice.**

    - **Receive notifications for order confirmations, shipping updates, and delivery.**

- **Profile Management:**
    - **Users can update their personal information, manage shipping addresses, and view past orders through the Customer Management Microservice.**

5. Testing and Refinement

- **Unit Testing:**
    - **Each microservice is tested independently using JUnit and Mockito.**
- **Integration Testing:**
    - **End-to-end testing is performed to ensure seamless communication between microservices.**
- **Validation:**
    - **Implement validation for all user inputs at both client and server sides.**
- **Bug Fixing:**
    - **Continuously monitor and address bugs identified during testing.**
- **UI/UX Refinement:**
    - **Regularly update the user interface based on user feedback and testing results.**