**Microservice Guide**

[Use Case 2](#_Toc174974409)

[Solution 2](#_Toc174974410)

[Identified Microservices 2](#_Toc174974411)

[1. User Service 2](#_Toc174974412)

[Responsibilities 2](#_Toc174974413)

[API Endpoints 2](#_Toc174974414)

[2. Product Service 3](#_Toc174974415)

[Responsibilities 3](#_Toc174974416)

[API Endpoints 3](#_Toc174974417)

[3. Price Service 3](#_Toc174974418)

[Responsibilities 3](#_Toc174974419)

[API Endpoints 3](#_Toc174974420)

[4. Cart Service 3](#_Toc174974421)

[Responsibilities 4](#_Toc174974422)

[API Endpoints 4](#_Toc174974423)

[5. Order Service 4](#_Toc174974424)

[Responsibilities 4](#_Toc174974425)

[API Endpoints 4](#_Toc174974426)

[6. Notification Service 4](#_Toc174974427)

[Responsibilities 4](#_Toc174974428)

[API Endpoints 4](#_Toc174974429)

[Architecture Components 5](#_Toc174974430)

[1. API Gateway 5](#_Toc174974431)

[Responsibilities 5](#_Toc174974432)

[2. Service Discovery 5](#_Toc174974433)

[Responsibilities 5](#_Toc174974434)

[3. Distributed Tracing 5](#_Toc174974435)

[Responsibilities 5](#_Toc174974436)

[Flow for Adding a Product and Checkout Process 5](#_Toc174974437)

[1. Adding a Product 5](#_Toc174974438)

[2. User Interaction 6](#_Toc174974439)

[3. Add to Cart and Checkout 6](#_Toc174974440)

[4. Handling Notifications 6](#_Toc174974441)

## Use Case

Develop a backend for a web application of an ecommerce company using Microservices Architecture. The application should be able to perform following operations in first release (design must be extendable for further operations)

* Admin can add/remove new products to inventory
* Admin can add/remove product details like size/price/design
* Product service can fetch detail about the product like size/price/design from another service (like Price service or Product detail service)
* User can view all the products list at any page
* Add to cart and checkout functionality need to provide.
* Notification can be sent to user (log notification in console as of now) based on event received from services.

## Solution

The microservice architecture provides a scalable and maintainable backend for an e-commerce application. Each microservice is responsible for a specific aspect of the application, allowing for independent development, scaling, and deployment. The API Gateway and Service Discovery components help manage inter-service communication and maintain a flexible system that can be extended with additional features and services in the future

## Identified Microservices

### User Service

#### Responsibilities

* Handles user registration, authentication and profile management
* Manages user information such as contact details and addresses
* Responsible for user authorization such as provide role at the time of user creation to authorize user to perform specific task
* Provide jwt token for user authentication at the time of user login

#### API Endpoints

* **BaseUR**L**:** <http://localhost:9006/>
* **Register User:** **POST** /api/v1/users/register
* **Login User (Generate token):** POST /api/v1/auth/token
* **Get All Users:** **GET**  /api/v1/users
* **Get User By ID:** **GET** /api/v1/users/{userId}
* **Update User:**  **PUT** /api/v1/users/{userId}
* **Delete User:**  **DELETE** /api/v1/users/{userId}

### Product Service

#### Responsibilities

* Manages product creation, updates and deletion
* Manages product information such as product name, description and inventory
* Aggregate the product data such as price from another microservice (Price Service)
* Provide the aggregate result of all product related microservices

#### API Endpoints

* **BaseUR**L**:** <http://localhost:9001/>
* **Create Product:** **POST** /api/v1/products
* **Get All Products:** **GET**  /api/v1/products
* **Get Product By ID:** **GET** /api/v1/products/{productId}
* **Update Product:**  **PUT** /api/v1/products/{productId}
* **Delete Product:**  **DELETE** /api/v1/products/{productid}

### Price Service

#### Responsibilities

* Manages and provides product pricing details
* Handles product price creation, updates and deletion

#### API Endpoints

* **BaseUR**L**:** <http://localhost:9002/>
* **Create Price:** **POST** /api/v1/prices
* **Get All Prices:** **GET**  /api/v1/prices
* **Get Price By ProductID:** **GET** /api/v1/prices/{productId}
* **Update Price:**  **PUT** /api/v1/prices/{productId}

### Cart Service

#### Responsibilities

* Manages shopping cart for users
* Handles adding, updating and removing items from cart
* Provide cart contents to the user

#### API Endpoints

* **BaseUR**L**:** <http://localhost:9003/>
* **Create Cart:** **POST** /api/v1/carts
* **Get All Carts:** **GET**  /api/v1/carts
* **Get Cart By ID:** **GET** /api/v1/carts/{cartId}
* **Delete Cart:**  **DELETE** /api/v1/carts/{cartId}
* **Add Cart Items:** **POST** /api/v1/carts/{cartId}/items
* **Update Cart Item:** **PUT** /api/v1/carts/{cartId}/items/{itemId}
* **Delete Cart Item:** **DELETE** /api/v1/carts/{cartId}/items/{itemId}

### Order Service

#### Responsibilities

* Handles order creation, updates and deletion
* Manages checkout process and order fulfilment

#### API Endpoints

* **BaseUR**L**:** <http://localhost:9004/>
* **Create Order:** **POST** /api/v1/orders
* **Get All Orders:** **GET**  /api/v1/orders
* **Get Order By ID:** **GET** /api/v1/orders/{orderId}
* **Get All Orders By UserId:**  **PUT** /api/v1/orders?userId={userId}
* **Delete Order:**  **DELETE** /api/v1/orders/{orderId}

### Notification Service

#### Responsibilities

* Handles the sending of notifications to users based on events
* Logs notification to the console

#### API Endpoints

* **BaseUR**L**:** <http://localhost:9005/>
* **Post Notifications:** **POST** /api/v1/notifications

## Architecture Components

### API Gateway

#### Responsibilities

* Routes incoming requests to the appropriate microservices
* Handles cross-cutting concerns such as authentication and authorization (microservices security)
* Distributes incoming requests across service instances (i.e. load balancing)

### Service Discovery

#### Responsibilities

* Keep track of all available service instances and their locations
* Allow API Gateway and other services to discover and communicate with each other dynamically

### Distributed Tracing

#### Responsibilities

* Tracks and visualizes requests as they flow through various microservices, providing insight into the complete lifecycle of a request
* Captures errors and exceptions along with trace information, making it easier to identify and resolve issues
* Measures and reports on latency at different stages of the request lifecycle, helping identify bottlenecks and performance issues

## Flow for Adding a Product and Checkout Process

### Adding a Product

* Admin user uses the User Service to register in the application by providing basic information such as email, password, name and role as “ADMIN”
* Admin user generates the token to access other services using User Service
* By using Product Service admin add product to database by providing product and price information which include calling the Price Service to create price information

### User Interaction

* User uses the User Service to register in the application by providing basic information such as email, password, name and role as “USER”
* User generates the token to access other services using User Service
* Users view products through the Product Service, which aggregates information from the Price Service and Product Service

### Add to Cart and Checkout

* Users create a cart via Cart Service
* Users add the items by providing productid and quantity via Cart Service
* During checkout, the Order Service processes the order, communicates with the Cart Service to retrieve cart details, and check and updates product inventory via calling Product Service
* Notifications are sent by the Notification Service to inform users of order status

### Handling Notifications

* Notification Service listens to events and logs or sends notifications to users as needed