# **Introduction**

This document describes the design and implementation of product information system which publishes product information updates.

# **Title**

Scalable and Resilient Event Based Message Exchange for Real-Time Product Information Updates.

**Objective**

Implement a robust and scalable system for exchanging event messages that updates product information promptly. Ensure these updates reach downstream consuming systems promptly, considering the global distribution of various apps across a multi-hybrid cloud environment. A simple visualisation/dashboard about product information updates would be nice to have. Follow best practices in code maintenance for releasing features through testing in production and deploy the application in cloud-native infrastructure. System Design Architecture is important.

# **Assumptions**

Source Data Provider

* Product Information System: Assume it is running in On-premise Datacenters.

Downstream Systems

* Market-Based Product Promotion System for US: Assume it is running in Microsoft Azure. (list of products, category, features, target group, promotional channel, promotional message, imageUrls)
* Customer Portal for Mobile Application System for EU: Assume it is running in Google Cloud Platform (GCP).

Sample Events to Handle

* Product Price Update
* Country
* Specific Product Discount Update

**Design Considerations**

* Ensure eventual data consistency.
* Address geographical distribution and latency challenges.
* Strategy for safe feature release.
* Auto-scale based on compute capacity or processing backlog.
* Cost efficiency.

# **Defining requirements**

**Functional Requirements**

1. Provide product details to product information system
2. Receive product details in customer portal from product information system
3. Publish changes from product information system to channels like customer portal and promotion system like

* Product Price Update
* Country – product available/unavailable for country
* Specific Product Discount Update

**Non-Functional Requirements**

Low latency

High availability

Code quality

Portability – containerization and deployment

# **System Design**

A screenshot of a computer screen

Description automatically generated

**Data Models**

**A screenshot of a computer

Description automatically generated**

**A close-up of a list

Description automatically generated**

**Services**

1. **Product source service**

Provides the CRUD operation APIs to get the product details from various sources

1. **Data segregation Service**

Has scheduler/database listener.

Data is read, segregated and stored in different tables.

1. **Product Information Service – with dashboard**

Provides the APIs for modifying products, publishing changes etc.

With outbox pattern

1. **User portal service**

APIs to get products and its changes

1. **Marketing service**

APIs to get promotion data.

**API Documentation**

Doc references

# **Implementation**

Development of REST API with the help of Spring Boot.

Layered structure implementation with controller, service and utility classes.

1. Controller – Handling requests and responses
2. Service interface – classes can implement service interface to provide different implementations for tax calculations.
3. Service – Implement service methods.
4. Utility classes – Calculation, conversion and data reading.
5. Configuration –Configure object mapper to read dates in java.time format.
6. Exception handler and Validation – Global exception handler class contains handlers for exceptions and custom validation for checking white spaces in string.
7. Logging – Use Spring AOP to add logging.

# **Testing**

A test project with test cases for functional testing with the help of Junit testing framework for the following cases

Test cases