# **Thought Process: Java Spring Application for Supermarket Product Management**

## **Overview**

This documentation aims to provide a comprehensive understanding of the functionality of the Java Spring application for managing supermarket products. The application allows retrieving products, adding items to the shopping cart, and calculating the total purchase value, including applicable discounts.

## **Architecture**

The application follows the MVC (Model-View-Controller) architecture, where:

* **Model**: Represented by Java classes defining domain entities, such as **‘Product’** and **‘Promotion’**.
* **View**: Not explicitly present in this application as it is a REST API, but clients can consume the endpoints to interact with the application.
* **Controller**: Represented by the **‘ProductController’** classes, which handle HTTP requests related to products, and **‘ProductService’**, which executes the business logic.

## **Key Features**

### **1. Retrieving Products**

* Endpoint: **‘/products’**
* HTTP Method: **‘GET’**
* Description: Returns all available products.
* Response: Array of **‘Product’** objects.

### **2. Retrieving a Specific Product**

* Endpoint: **‘/products/{id}’**
* HTTP Method: **‘GET’**
* Description: Returns a single product based on the provided ID.
* Response: **‘Product’** object.

### **3. Adding Items to the Shopping Cart**

* Endpoint: **‘/products/addToCart’**
* HTTP Method: **‘POST’**
* Description: Adds a list of items to the shopping cart and calculates the total purchase value, including applicable discounts.
* Parameters: List of **‘CartItemDTO’** objects containing product ID and quantity.
* Response: List of **‘ItemCheckoutDTO’** objects containing details of each item added to the cart.

## **Business Logic**

The business logic is primarily implemented in the **‘ProductService’** class. Here are the main operations performed:

1. **Retrieving Products**: Utilizes the **‘RestTemplate’** class to make GET requests to the mock service (WireMock) and fetch available products.
2. **Adding Items to the Shopping Cart**:

* For each item in the cart, retrieves the corresponding product through the mock service.
* Calculates the total item value, considering applicable discounts based on the promotions associated with the product.
* Applies different types of discounts, such as fixed percentage discount, "buy X, get Y free," and price override based on quantity.

## **Conclusion**

The Java Spring application offers a robust solution for managing supermarket products, facilitating product retrieval, addition of items to the shopping cart, and calculation of total values with applicable discounts. The modularity and clarity of the code make maintenance and scalability of the application easier.