Introduction

This document describes the detailed planning for the development of the Grocery Store system, where products can be added to the cart, promotions can be automatically applied, and customers can view their savings. The system will integrate with an external product API (via Wiremock) and will be flexible enough for future expansions.

Technologies

Java 21
Spring Boot
Feign for external API communication
JUnit 5 and Mockito for automated testing
Lombok to reduce boilerplate code

System Objectives

Implement a shopping cart that allows products to be added in any order, with automatic application of promotions.
Display to the customer the savings from the applied promotions.
Ensure the system is extensible, allowing new promotions or future integrations.

System Structure

External API Communication Layer - ProductClient

The ProductClient will be the component responsible for consuming the external API for products and promotion details, using Feign.
This client should have methods to fetch all available products and details of a specific product.

REST Controllers - ProductController and CartController

ProductController:
Should be responsible for providing REST endpoints to access all products available in the system and the details of specific products.
Should use the ProductService to retrieve product data.

CartController:
Should be responsible for managing the shopping cart.
Should allow the customer to add products to the cart, clear the cart, and view the current state (such as added products, total price, and savings).
Should delegate the cart manipulation logic to the CartService.

Services - ProductService and CartService

ProductService:
Should be responsible for interacting with the ProductClient to retrieve product data.
Should provide methods to fetch all products and details of a specific product.

CartService:
Should manage the state of the shopping cart.
Should allow adding products to the cart, updating the total price, final price with discounts, and calculating the total discount applied by promotions.
Should apply promotions when applicable, based on the type of promotion associated with the product (QTY_BASED_PRICE_OVERRIDE, BUY_X_GET_Y_FREE, or FLAT_PERCENT).
Should allow the cart to be cleared.

Data Models - Product, ProductDetail, Promotion, Cart

Product:
Represents a basic product, containing information such as id, name, and price.

ProductDetail:
Extends the Product class to include additional information such as applicable promotions for the product.

Promotion:
Represents a promotion, containing details such as the type of promotion, required quantity for the discount, free quantity (if applicable), and percent value of the discount, depending on the promotion type.

Cart:
Represents the state of the shopping cart, storing the list of added products, the total price, the final price (after promotions are applied), and the savings.

Promotions - PromotionType

PromotionType:
QTY_BASED_PRICE_OVERRIDE: Applies a fixed price when a minimum quantity of products is purchased.
BUY_X_GET_Y_FREE: Offers a free product when a specific quantity is purchased.
FLAT_PERCENT: Applies a percentage discount directly to the product price.

Automated Testing

The system should include automated tests to ensure that all functionalities work correctly.
The tests will be implemented using JUnit 5 and Mockito to cover:
Unit tests : Validate the behavior of each service in isolation.
Integration tests: Ensure that communication with the external API, via Feign, works as expected.
End-to-end tests: Use MockMVC to verify the behavior of the exposed endpoints from the controllers.