Jose Eduardo Pons Fidalgo

Business Case Document for the Shopping Cart Rest API Project

Executive Summary

The Shopping Cart Rest API Project is designed to simulate the core functionalities of an online shopping platform that includes item management, cart operations, and batch report generation. This system is built using modern Java development tools such as **Spring Boot**, **Spring Data JPA**, **Spring Security**, **RabbitMQ**, **Mockito for testing**, and **Spring Batch**. It provides a seamless backend service that interacts with a simple frontend to allow administrators to manage items, generate sales reports, and handle messaging services.

Objectives

The main objectives of this project are:

- Provide REST APIs to manage items in the store.
- Allow administrators to add, edit, and delete items from the database.
- Integrate with **RabbitMQ** to send notifications when new items are added to the store.
- Use **Spring Batch** to generate sales reports in a scheduled or on-demand manner.
- Ensure the system's security by implementing **Spring Security**.
- Use **Mockito** to ensure robust unit testing of the components.
- Design a user-friendly frontend that enables interaction with the backend system.

Technical Stack

1. **Backend Framework**: Spring Boot (REST API)
2. **Database**: MySQL with Spring Data JPA for ORM
3. **Messaging Queue**: RabbitMQ for notifications
4. **Batch Processing**: Spring Batch for generating sales reports
5. **Security**: Spring Security for securing the endpoints
6. **Unit Testing**: Mockito for mocking dependencies and unit testing
7. **Frontend**: HTML/CSS/JavaScript

Scope

1. Item Management

The core functionality of the system is managing items in a store. The following operations can be performed on the item resource:

- **Add Item**: Administrators can add new items using the `/shop/add` endpoint. Each time an item is added, a notification is sent to RabbitMQ.
- **View Items**: All items can be viewed via the `/shop/items` endpoint.

- **Edit Item**: Administrators can modify existing items using the `/shop/edit/{id}` endpoint.
- **Delete Item**: Items can be deleted using the `/shop/delete/{id}` endpoint.

2. Cart Management

The cart management system is one of the primary functions of this API, allowing users to add, view, and remove items from their cart. It also provides the capability to calculate the total cost of all items in the cart.

- **Create Cart**: Users can create a new shopping cart using the `/cart/createCart` endpoint.
- **Add Item to Cart**: Users can add items to their cart through the `/cart/add/{cartId}/{itemId}` endpoint.
- **View Cart**: The current state of the cart can be viewed, displaying the items and their respective quantities.
- **Remove Item from Cart**: Users can remove or decrease the quantity of an item from their cart via the `/cart/delete/{cartId}/{itemId}` endpoint.
- **Total Cost Calculation**: The total cost of all items in the cart, including taxes, is calculated and stored using the `/cart/totalCost/{cartId}` endpoint.
-

3. Sales Report Generation (Spring Batch)

The project uses **Spring Batch** to automate the process of generating a sales report:

- The batch process reads cart data from the database, processes the data (calculating the total sales), and writes the result into a CSV file.
- This process is configured in the **BatchConfig** class, which includes an `ItemReader`, `ItemProcessor`, and `ItemWriter`. The sales report can be triggered either manually or at regular intervals.
- The report is generated when the administrator triggers the report generation from the UI or programmatically through the `/batch/runSalesReport` endpoint.

4. Notifications (RabbitMQ Integration)

- RabbitMQ is integrated into the system to handle notifications whenever a new item is added to the inventory.
- The `CartNotificationService` is used to send notifications to a queue (`cartNotificationQueue`) each time a new item is added.
- This ensures real-time notifications and system extensibility.
- **User**: guest **Password**: guest

5. Security (Spring Security)

- The project uses **Spring Security** to protect sensitive endpoints. Only administrators are allowed to add, update, or delete items, whereas general users can view items.
- Basic authentication or JWT-based authentication can be added to further enhance security.
- Security configurations define roles (`admin` and `client`), with granular control over what actions each role can perform.
- **User**: admin **Password**: admin
- **User**: client **Password**: client

6. Testing (Mockito Integration)

- **Mockito** is used to write unit tests, especially for mocking dependencies such as `ItemRepository` and `CartNotificationService` during testing.
- Tests are written to ensure that business logic such as adding items and sending notifications works correctly.
- Example test case includes verifying that when an item is added, the `save()` method on the `ItemRepository` is called and a notification is sent to RabbitMQ.

7. Frontend Implementation

The frontend is implemented using a basic HTML/CSS/JavaScript structure, which connects to the backend REST API for operations:

- **Admin Dashboard**: The frontend provides an admin panel to allow users to add, view, edit, and delete items.
- **Sales Report Download**: A button on the frontend allows administrators to download the sales report.
- **Menus**:
    - **Home**: A simple home page that connects the admin to the dashboard.
    - **Add Item**: A form that lets the admin add new items to the store.
    - **Edit Item**: Allows administrators to edit existing items.
    - **Delete Item**: Lets administrators remove items from the inventory.
    - **Sales Report**: Button to generate and download the sales report.
    - **Security**: Login functionality to access secure pages.

Conclusion

The **Shopping Cart Rest API Project** successfully integrates modern Java tools and frameworks to deliver a secure, functional, and scalable solution for item management, notifications, and reporting. With the use of **Spring Batch**, **RabbitMQ**, **Mockito**, and **Spring Security**, this project not only fulfills the basic requirements of an e-commerce back-end system but also includes powerful capabilities like real-time messaging and automated batch processing. Furthermore, the addition of unit testing ensures robustness and reliability of the system.