Linnéuniversitetet
Kalmar Växjö

Report

Architecting and Design Assignment

SmartGrocery Application

Author: Yuyao Duan
Supervisor: Mauro Caporuscio
Semester: Spring 2022
Discipline: Software Design
*Course code: 2*DV608

Table of Contents

1 Introduction

This document aims to provide a feasible and reliable architecting and design for the upcoming SmartGrocery software which has an unique vision to provide convinent shopping information service for end user. In the following sections, the details of architecting and design of the application of will be presented.

2 Assumptions and dependencies

2.1 Critical factors of the application

1) Mainly users have smart phones, SmartGrocery should be compatible with Andriod and iOS systems.
2) SmartGrocery system should be able to withstand high-concurrency at the peak time, this due to end-users may want to shopping in the same time period.
3) With capability of providing up-to-date product information requires this application having good server cache management mechanism to balance between high-concurrency as well as information update.
4) Due to the demand of the capability of handling agile business models, SmartGrocery should be developed with light-weight frameworks in case of changing.
5) In order to enable the abilities of quick start, independent deployment, no dependencies on other services, and low coupling.

2.2 Dependencies on legacy software and system

1) The company's previous web based applications are developed with monolithic SSM (Spring + SpringMVC + MyBatis) the three main frameworks, therefore the potential dependencies with SSM may exist.
2) JDK 13 is being used by the develop team as stable version which will also be used for SmartGrocery. JDK 17.0.2 has been released, and will eventually supersede the current version however support for 17.0.2 in third party libraries is currently still low and stablility waiting to be tested, so the risk of using JDK 17 judged to be higher as of March 2022.

2.3 Resource constraints

1) The commercial operation team of SmartGrocery is a start-up, the application will be deployed on Amazon Web Services (AWS) instead of purchasing own servers due to the demand of light-asset operation.
2) The development team consists of 15 people, therefore maximum workload per person during development period is limited, the company should plan a reasonable development schedule and deadlines.
3) Due to the limitation of budget for SmartGrocery, the development team will stick with reliable free open-source frameworks.

2.4 Analysis

The current frameworks (Spring series frameworks) provide a very good high level structure for system development i.e. the development processes are well structured: three main parts including controller part, service and service Impl, DAO and DAO Impl; while, other system components such as web config adapter, common util, exeception handler, value object, system entity stc. can be developed depending on the actual business demands.

For SmartGrocery project, applying top-down design means all high-level decisions will be made before the detailed implementation. Therefore, the effectiveness of the system is significant since all low-level implementations are customized for higher-level demands. The inadequate of this design approach may lead to longer development cycle and ignoring of reusing available resources which can be considered as wasting. By comparison, bottom-up design will contribute better cost efficiency due to the fact of focusing on developing reusable low-level modules which will reduce development period for other projects. However, considering this case that switching from using traditional SSM approach to microservices, the purely bottom-up design may not be the ideal approach since many lower-level modules need to be changed in order to adapt new frameworks.

Therefore, in this design, a mix of top-down and bottom-up approach will be selected due to the fact that the current frameworks greatly simplifies the difficulty of doing this. The system designers can save a lot of energy for constructing upper-level structures, and put more effort on integrating appropriate reusable modules.

3 Architecturally significant requirements

3.1 Application compatible with third-party services

- **Requirement:** Add / update information about a product, including its price and the store where the product is located (FR 3) & Browse nearby stores on a map (FR 5).
- **Rationale:** the software architecture should developed with a reliable map interface to integrate with third-party maps, the selection of which map to use is also very criticial. This will also influence the calculation results for "requirement 4 – giving suggested stores" since the distance accuracy can be afftected by the map modules.

3.2 Implementing reliable frameworks for data processing

- **Requirement:** Edit the shopping list (add/remove items, modify quantity) (FR 2) & Display the list of stores ordered by distance (FR8) & Display the list of products ordered by price (FR 9).
- **Rationale:** The application's architecture must be able to provide stable service and performance for browsing products and editing the shopping list (database CRUD opertaions) during peak hours.

3.3 Protability

- **Requirement:** Communicate with client applications across different platforms (FR 11).
- **Rationale:** The application's architecture must support different working environments and system conditions.

3.4 Product information collection

- **Requirement:** Automatically collect products and prices from available store's web site (web scraper) (FR 10)
- **Rationale:** The application architecture should have a reliable and efficient module with advanced algorithm to collect product information that users are interested in.

4 Decisions, constraints, priorities and justifications

- The application will be designed with client-server model.
- It will be developed as thin-client i.e. this software will be designed as high dependence on network resources.
- The important functional modules such as map module will be designed as calling third-party module instead of making own module, the application will rely on third-party server's performance (Also a constraint).

- The program will be programmed in Java due to: a) Most people in the development team are most experienced with Java, and all previous projects are developed in Java which means most available resouces of this team are Java-based modules (Also a constraint). b) Java's powerful ecosystem and third-party library support are strongest among other options.
- In terms of business consideration, the development team will apply reliable official frameworks instead of self developing, the customization and special needs are inferior (Also a constraint). Figure 1. shows the design alternatives during the decision making process.

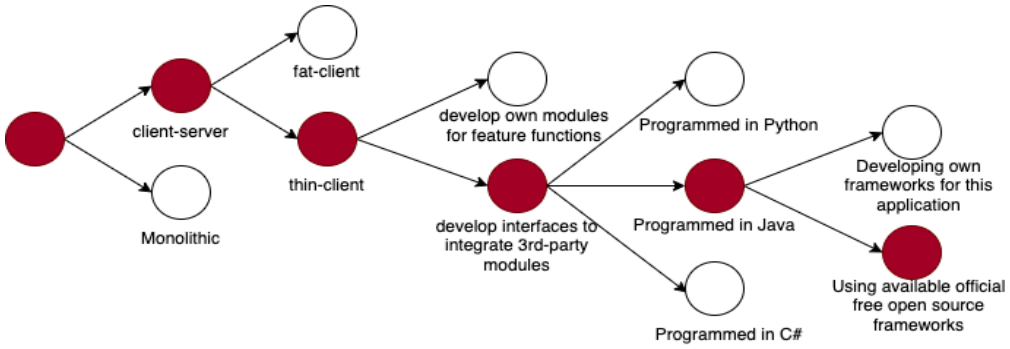

Figure 1. Design alternatives and decision making processes

5 Architectural Patterns

Combined with theoretical architectural patterns and current e-commerce development, most client-server based applications are developed with multi-layer architectural pattern i.e. applying "Spring+SpringMVC+MyBatis" to realize structured development for controller layer, business service layer, database persistence layer which largely simplify the development procedures.

A step forward of traditional SSM-based applications is implementing Spring Cloud microservices architecture. Microservices not only enable the convenience of project development and cross-platform (Principle 9), but also reduce the difficulty of operation and maintenance as well as improving the performance under the situation of unreliable network etc. Implementing microservice architectural pattern will enable the focal system to be consistent and uniform in the long-term since microservices are constructured with small and autonomous services (Principle 1). Updating, changing or removing separate service will not influence the system's performance as a whole. The microservices architecture enables the functional cohesion that each sub-service focusing on a typical service as well as reduce the interdependencies among other system modules due to the fact that each micro service can run independently (Principle 2 and 3). Implementing microservices architecture will benefit the development team in long run since the developed sub-modules/services can be easiliy ported to new applications (Principle 6). Microservice architecture allows inner

technology heterogeneity that different services are developed from dfferent techniques to achieve (Principle 5 and 7). Therefore, microservices will be considered as the most optimized strategy to develop software typically for long-term upgrades and maintenance. Based on above analysis, the microservices architecture is the ideal option for SmartGrocery development.

6 Architectural views

6.1 Component diagrams

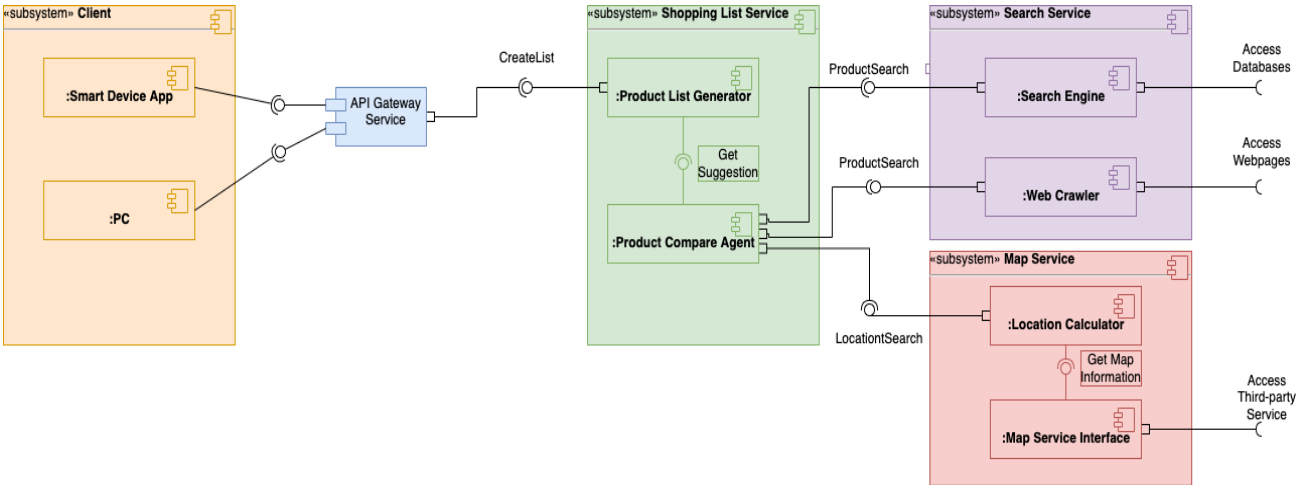

Figure 2. SmartGrocery Component Diagram

This component diagram provides an overview about how does SmartGrocery's system components related to each other. The system can provide service via both mobile app and PC web browser (though the mobile users are the main focus). All service requests will be forwarded via API Gateway Service module to call the microservices. There are many advantages, one of most significant part is that system administrator can know how people use the APIs through integrated analytics and monitoring tools which can contribute long-term business values. In the Shopping list service subsystem, the Product List Generator will call Product Compare Agent to get search result which Product Compare Agent is the core module in this application. Product Compare Agent will call other microservices including Search Engine, Web Crawler, and Location Calculator. All the data will be returned to Product Compare Agent which will be used to create the suggested list.

6.2 Deployment diagrams

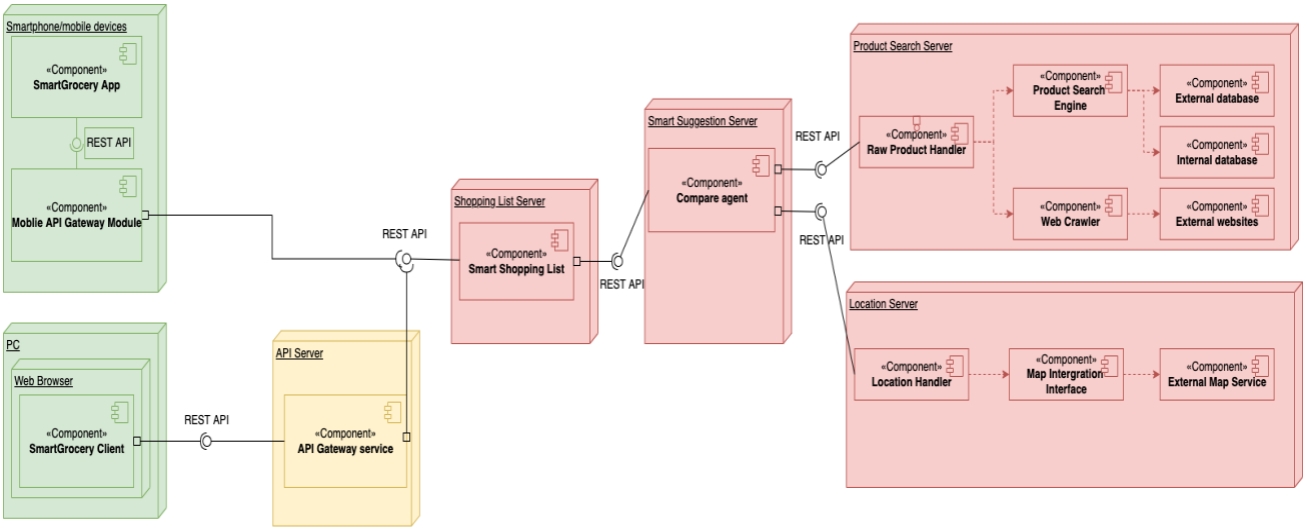

Figure 3. SmartGrocery Deployment Diagram

This deployment diagram gives information about how does SmartGrocery application system will be deployed. For the mobile device users, Mobile API Gateway Module is embedded with the application and for the traditional PC web browser users will access the API server. This design is due to the sake of efficiency for mobile applications. Under microservices architecture, different business functions are seperated as individual microservice. The pink parts present the main microservices in this software. Product Search Server provides products' price, brief introduction, the store information etc. Location Servier will call external map service (such as Google Maps) to get information such as distances. All information will be forwared to Compare Agent to process the raw data that will be used for generating Smart Shopping List.

6.3 Sequence diagrams

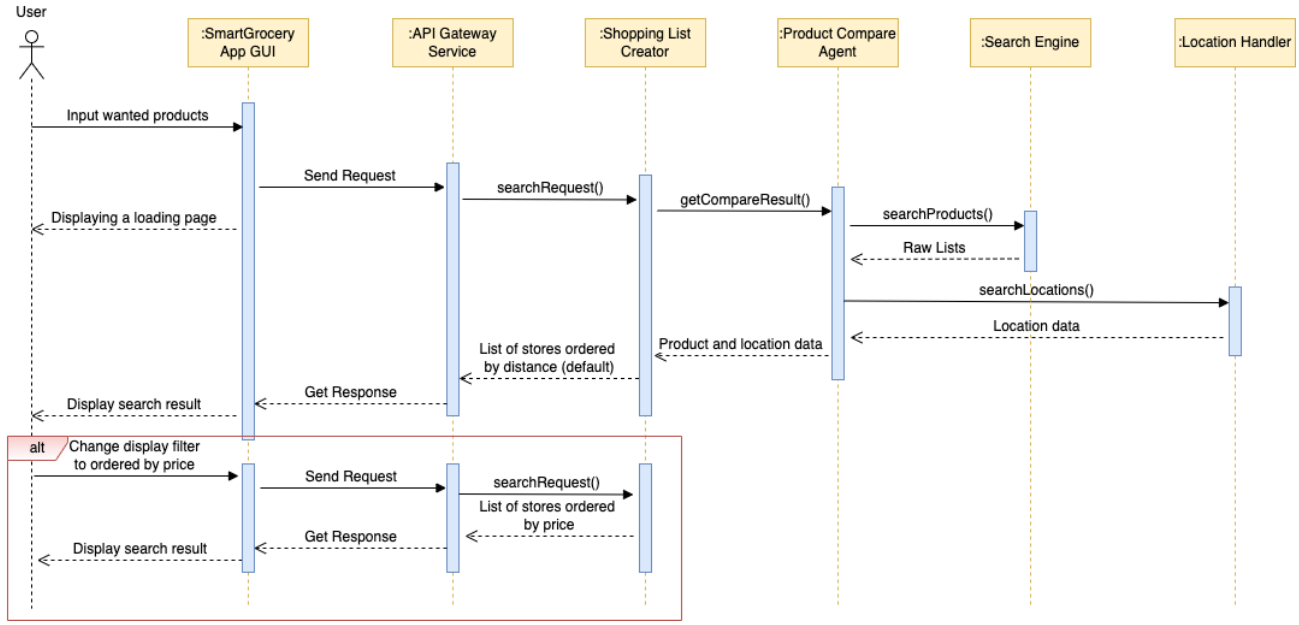

Figure 4. SmartGrocery Dequence Diagram

This sequence diagram provides an outline about how does SmartGrocery application system will behave during business operations. The system are expected to return the search result between 50 and 200 ms since longer than the time period will significant influence user experience. For the Shopping List Server (where Shopping List Creator located) will be designed with cache function that will acquire all the shopping list information, this will improve the system performance when users change their list query conditions since the list will be created from cache instead of accessing external databases or webpages second time. This design will typically benefit the application during peak hours under high concurrent situation.