<Smart Shopping>

Analysis and Design Document
Student: Andrei Rusu
Group: 30431

Revision History

Date	Version	Description	Author
<18.03.2020>	<1.0>	<initial specifications>	<Andrei Rusu>

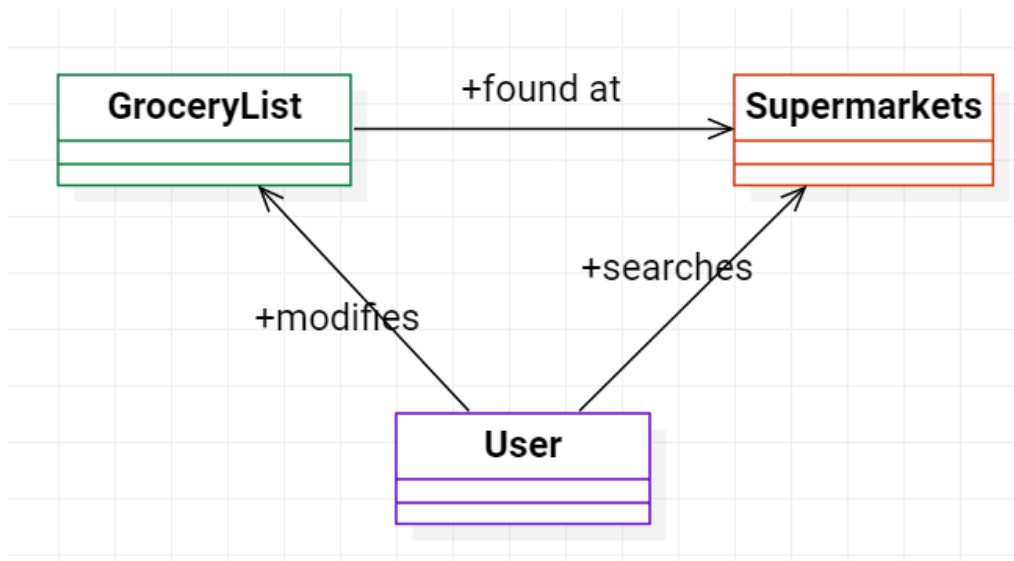| | Version: <1.0> |
| | Date: <18.03.2020> |
| <document identifier> | |

Table of Contents

I.   Project Specification

The project is a system designed to run on Android devices, together with a backend for loading and storing data, implemented in Firebase.

II.   Elaboration – Iteration 1.1

1.   Domain Model

The domain of the application covers a few distinct aspects. First, the user should be able to input a shopping list, save it to the cloud, and access it later for editing. Then, the user should search for a supermarket in their vecinity using the Google Maps API integrated into the application. After finding a suitable supermarket, the app should guide the user through the shop according to their grocery list, showing them in which part of the store their items are found.

	Version: <1.0>
	Date: <18.03.2020>
<document identifier>	

## 2.   Architectural Design

2.1   Conceptual Architecture

The system will be implemented using the Model-View-ViewModel (**MVVM**) architecture, using LiveData for real-time data update, Android data binding and various other Android Jetpack features.

The Model of the application will be made up of data classes, such as grocery list items, supermarkets and shop maps.
The View of the application is made up of what the user sees, aka the fragments and activities of the app, and the items in them, such as lists, maps and so on.
The ViewModel represents the domain and business logic of the applciation, it is what processes requests and connects the other components together. Also, the business logic of the application is decoupled from the UI.

There will be a local and remote data source. The local data source will be implemented with Room Database, and the remote one will use Firebase Realtime DB as data storage. This type of layer separation is great for building modular and scalable apps that can easily be modified and upgraded later.
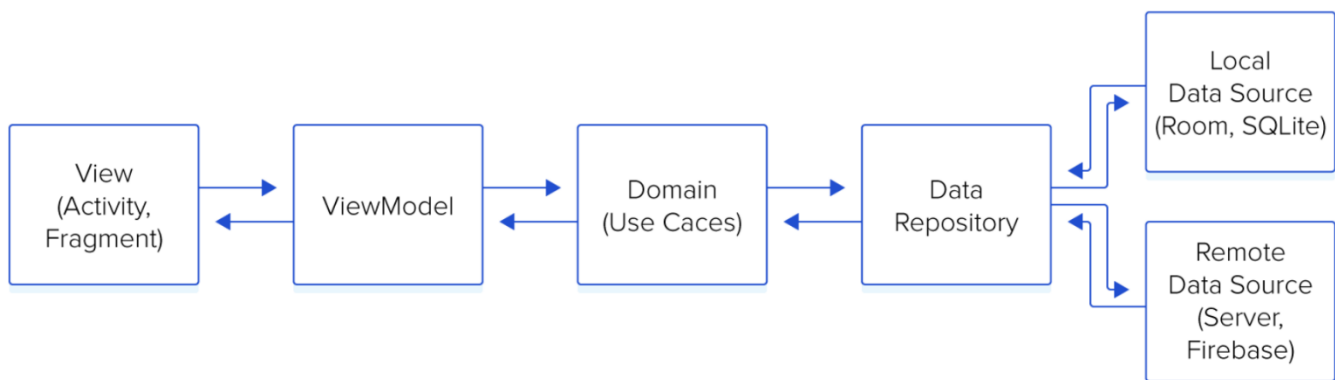
2.2   Package Design
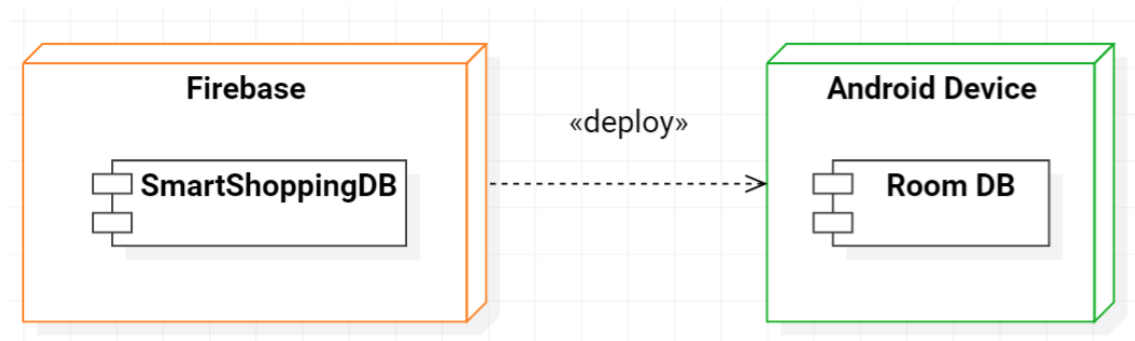
The preliminary package diagram is shown below, depicting the main packages of the MVVM pattern:
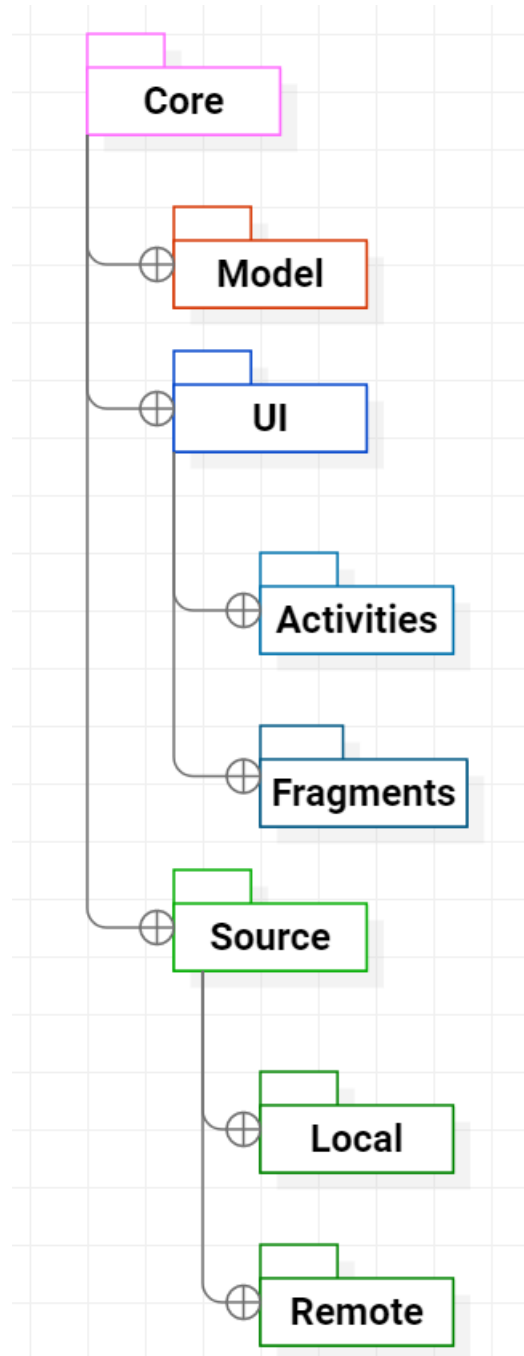
2.3 Component and Deployment Diagrams

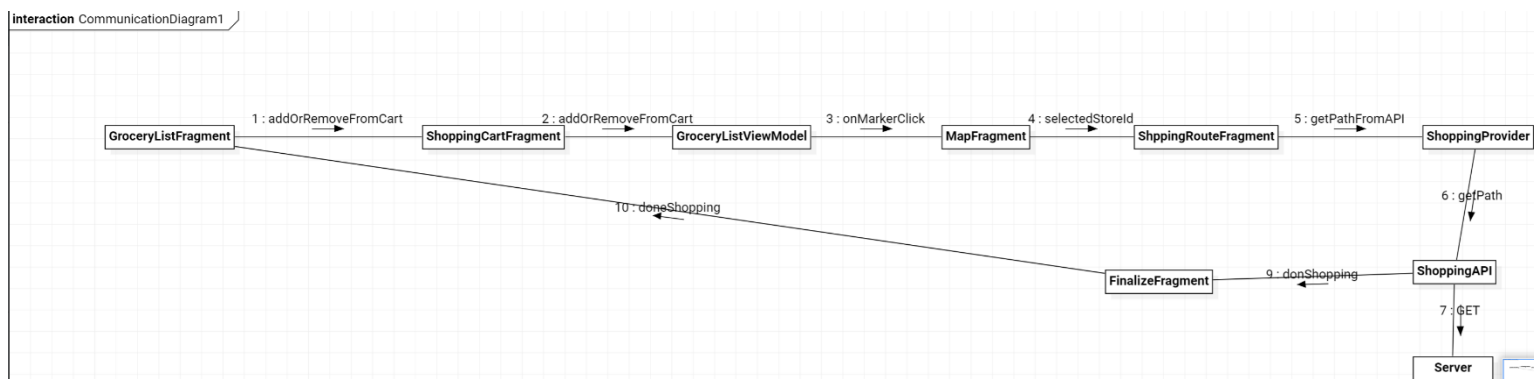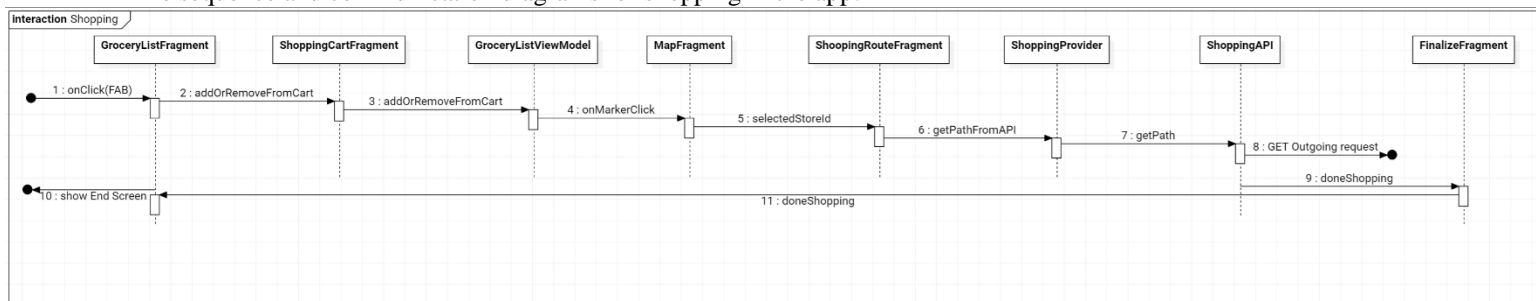# III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

The sequence and commuincation diagrams for shopping in the app:

1.2   Class Design

Inside the application, the Factory Method pattern is used for Fragment instantiation, the Builder pattern for dialogs, the Observer pattern for live data and also other patterns used in smaller areas, part of the recommended programming practices of Android and the MVVM pattern. This is similar to the A2 assignment, except more expansive and complete.
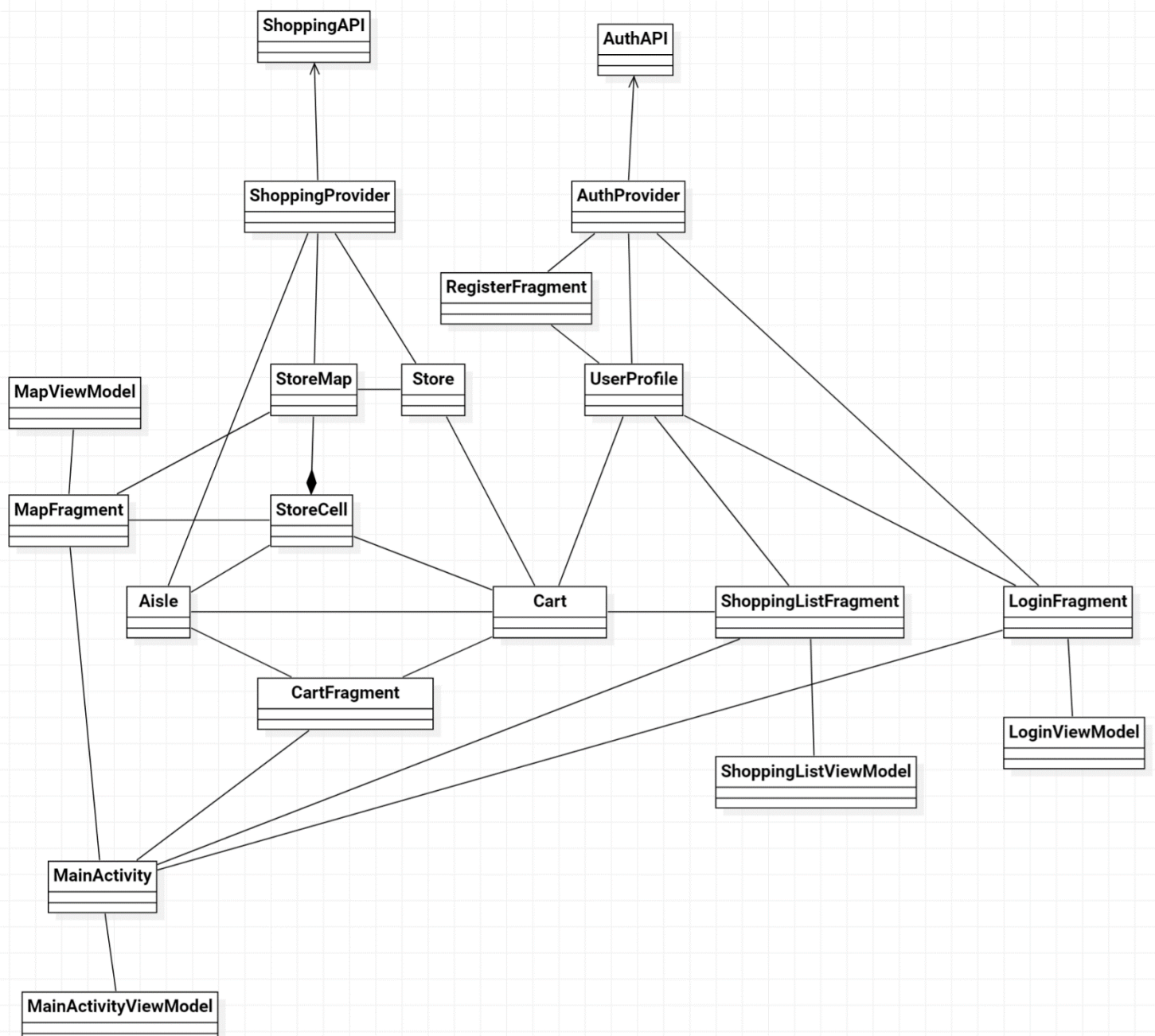
2.   Data Model

The data of the system is organized into the following models:

The User:
- username: String
- name: String
- password: String
- token: String

The Store:
- ID: Int
- name: String
- latitue: Double
- longitude: Double
- mapId: Int

The StoreCell:
- ID: Int
- row: Int
- col: Int
- type: Int
- store_id: Int

The Aisle:
- ID: Int
- name: String

The User is, as the name implies, the end user of the application. They add their desired groceries to the application and search for nearby stores. The Store represents a physical supermarket and has latitude and longitude information describing its location. Inside, the store is divided into cells, each cell having a row and column coordinate and a type (types like dairy products, bakery, etc). The Aisles represent singular-type areas of the store.

3.   Unit Testing

[Present the used testing methods and the associated test case scenarios.]

IV.   Elaboration – Iteration 2

1.   Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2.   Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V.   Construction and Transition

1.   System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2.   Future improvements

[Present future improvements for the system]

VI.   Bibliography