SmartCart
Design Description

Version 1.3

Revision History

Date	Version	Description	Author
2011-10-20	0.1	Initial draft	SmartCart Team
2011-10-24	0.8	Revised draft	SmartCartTeam
2011-10-27	0.9	Revised draft no. 2 – ready for the final review before submission	SmartCartTeam
2011-10-27	1.0	First version of the document	Luka Božić
2011-11-27	1.1	Added Server API to references and chapter 2.3	Luka Božić
2012-01-12	1.2	Updates	Željko Brdarić, Filip Gvardijan, Ivo Štimac
2012-01-13	1.3	Final updates	Luka Božić

Table of Contents

1. Introduction

SmartCart is imagined as a mobile application which should ease up the daily process of buying groceries for end users. Smart Cart will enable users to enter groceries they have to buy to a list, and then find out in which store they can buy cheapest groceries and save money or find the closest store with requested product. Although there are some existing web sites and applications with purpose of finding cheapest items across multiple stores they do it on the basis of a single item. SmartCart on the other hand will do it for the whole list of groceries

1.1 Purpose of this document

The purpose of this document is to provide readers with design description of the SmartCart project. This document includes information about hardware, software and communication interfaces, software architecture and software design.

1.2 Intended Audience

Intended audiences of this document are all project stakeholders:

- Project customer,
- Project supervisor,
- Team members,
- All persons responsible for monitoring the project.

1.3 Scope

Scope of this document is to provide an insight into detailed design of the SmartCart project. Database design, server backend and mobile application architecture are explained.

1.4 Definitions and acronyms

1.4.1 Acronyms and abbreviations

Acronym or abbreviation	Definitions
ADT	Android Development Tools plugin for Eclipse
WIFI	Mechanism for wirelessly connecting electronic devices
3G	3rd generation of standards for mobile telecommunication services
GUI	Graphic User Interface
REST	REpresentational State Transfer
WCF	Windows Communication Foundation
JSON	JavaScript Object Notation
GPS	Global Positioning System

1.5 References

- Project homepage
  - http://www.fer.unizg.hr/rasip/dsd/projects/smartcart
- Project plan
  - http://www.fer.unizg.hr/_download/repository/Project_Plan_v1.0.pdf
- Requirements definition
  - http://www.fer.unizg.hr/_download/repository/Requirements_Definition%5B2%5D.pdf
- SmartCart server API
  - http://www.fer.unizg.hr/_download/repository/SmartCart_-_SmartCart_server_API.pdf

2. External interfaces

Different interfaces like hardware, software, communication and graphic user interface have to be defined in order to implement a system.

2.1 Hardware Interfaces

SmartCart operates similarly on various mobile devices with different physical characteristics that run on Android operating system. Since the Android client will be implemented as a thin client, meaning the most of the computing and data storing will be done on the server side, the mobile application will not consume a lot of CPU or phone memory.
Android application will use the Global Positioning System (GPS) to retrieve the current user location. The GPS will be needed for the purpose of locating the user and getting the nearest store or stores around him in a radius specified by the user. It provides the exact location all the time and will be integrated with Google Maps, so the users can see their location and the location of the stores on the map. The phone camera functionality will be used for barcode recognition and scanning.

2.2 Software Interfaces

The Application will be developed on 2.2 Android operating system platform, on which most of Android based smartphones nowadays operate[1] . SmartCart will be developed in Eclipse with the ADT plugin which extends the capabilities of Eclipse. ADT facilitates creating Android applications and designing user interface.

2.3 Communication Interfaces

The communication interface between the Android phone and web server is achieved through the web service (JSON / REST). REST permits many different data formats and one of them is JSON, which can be relatively easily consumed by an Android application. Another advantage is that REST has good performance and scalability.
The exact communication (message formats, parameters etc.) is described in the SmartCart Server API document.

2.4 Mobile application graphic user interface

The main graphic content will be implemented using embedded basic layouts and widgets which ADT provides. One of the design decisions is that the user's screen orientation will be locked in a portrait mode view. The user will start the application by selecting the SmartCart icon in the application menu. Error and help messages pop up when they have initially occurred.

A good GUI has to be:

- Intuitive
- User-friendly
- Fast

Mobile application GUI contains of several views which change regarding users actions. Figure 2.1 shows all the products and allows filtering them by product categories. Figure 2.2. shows currently selected products and their quantity on the shopping list. Figure 2.3. shows the screen with two ways of searching for products. The last figure 2.4. shows the map with user location and all the stores around the user within the specified radius.

[1] This page provides data about the relative number of active devices running a given version of the Android platform.

http://developer.android.com/resources/dashboard/platform-versions.html

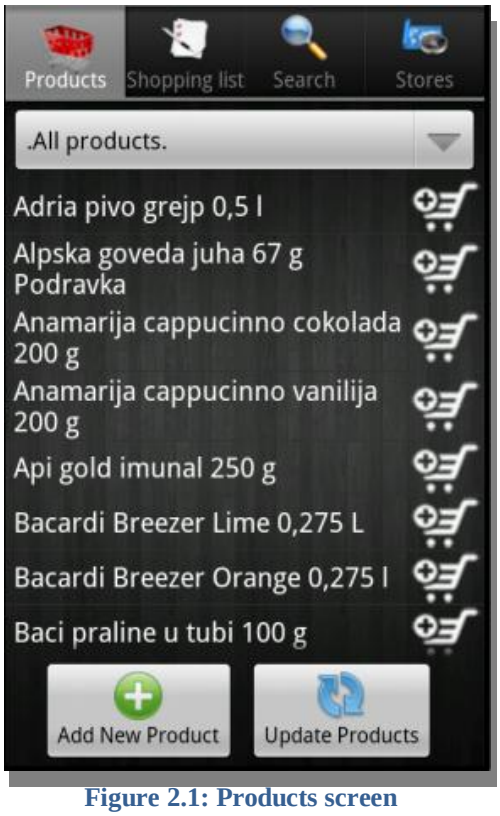

Figure 2.1: Products screen

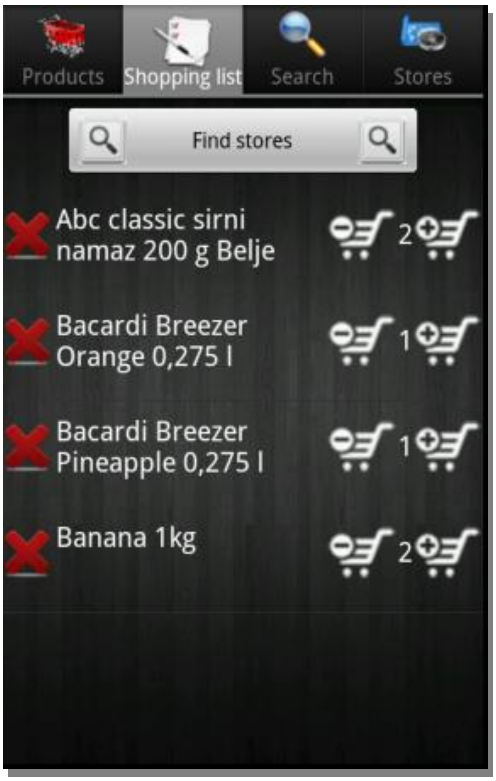

Figure 2.2: Shopping list screen

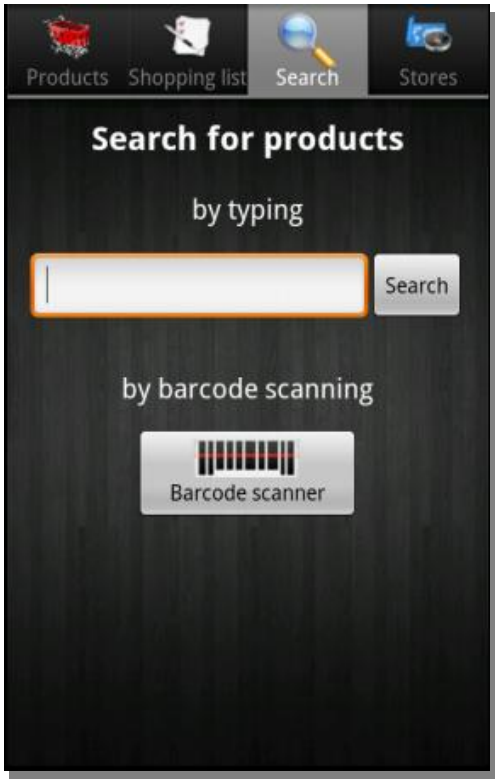

Figure 2.3: Search screen

Figure 2.4: Stores screen

3. Software architecture

Our system is implemented using Service oriented Architecture. The Figure 3.1 provides an abstract overview of the whole system. SmartCart server is a web based server, whose client is the Android application. The server processes and handles the client's data requests. The Android client and SmartCart server will communicate over the Internet, using a web service and the data format being exchanged between the client and server will be JSON based which has many advantages including reduced communication and parsing overhead. We can see that there are three building blocks of our software.

1. **Android Client**
   The client application exposes the system functionalities to the users. The client communicates with the server and consumes the web services offered by the server. The android application will be implemented as a thin client, which means that most of the computing and the data will be stored on the server side of the system. In the future that will enable faster, easier and consistent development of the SmartCart mobile application for other mobile platforms such as the Windows Phone 7.
2. **SmartCart Server**
   The server application encapsulates the major functionalities of the system. These functionalities include web services management, storage management, and user authentication.
3. **Web Services**
   The web services function as a middleware of our system. The client requests are sent to the server through the Internet and handled by the server. SmartCart will be using JSON based web services because of the reduced overhead related to parsing responses/requests as compared to the SOAP/XML.

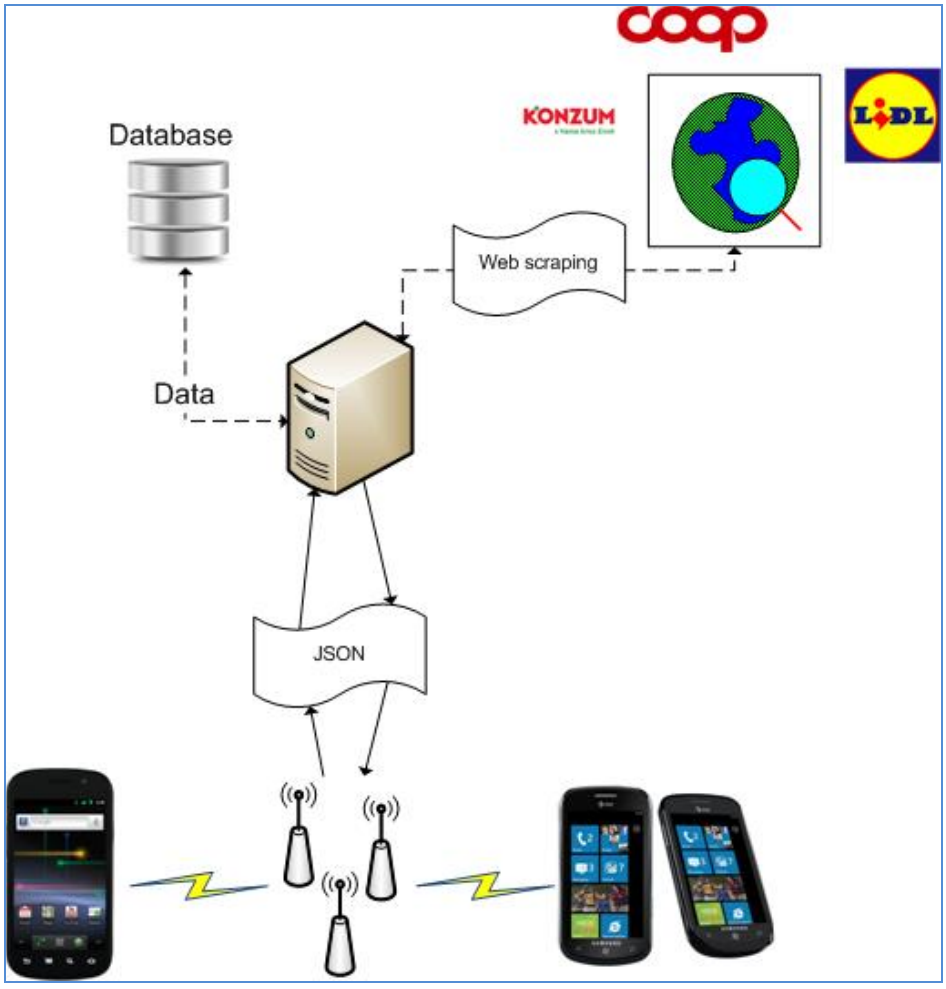

Figure 3.1: Service Oriented Architecture of SmartCart Application

3.1 Android Application Design

Android application is supposed to give the user insight into available products and stores that are nearby. For that sake it's designed to update important data: products list, product categories and stores; in user's vicinity on startup. For storing that data SQLite database is used and class DBAdapter is handling all communication with the database. Data is sent from and delivered to the application by calling defined services which accept / return data in JSON format. Class MyJsonParser is used to handle all incoming and outgoing communication with the web services. When sending request to the web service this class have objects as input and outputs JSON format to the web service. When receiving data from web service class handles JSON as input and outputs java objects. All of data objects used in the communication are defined as Java classes in package dsd.smartcart.model.

For the presentation of the data application uses standard android library with built in widgets. All presentation is handled in two layers. First data is processed in classes that belong to dsd.smartcart.main package, where every class extends android built in Activity class and declares which layout it uses. Layouts define placeholders in which data is placed after processing by the aforementioned classes. Also complete design of every screen is defined in the layouts files.

There are four main parts of android application:

- Products,
- Shopping list,
- Search and
- Stores.

Classes that implement the functionality for those parts are placed dsd.smartcart.main package.

The products feature enables user to search all available products. It is expected that more than 1000 products will be present in final version of the application, and in order to search for the products more easily filtering by categories is implemented in the products screen. Products are shown in the list where data is fetched from the database. User is able to add products to the shopping list, browse specific products for detailed prices in nearby stores and can change the price of the product in the store. User can also add new product by choosing product category, one of nearby stores in which products can be bought, product name, price and optionally its barcode.

The shopping list feature enables the user to add multiple products to shopping list in order to discover nearest and / or cheapest store that has all products. User can change radius and method of search in the settings. This data is sent to the web service, suggested stores are received and presented to the user.

The search feature enables user to search for the products already in the products list by entering products name or scanning it's barcode. Dropdown list is shown to the user when entering name. For barcode scanning, ZXing external library is used. Library is imported to the application, that way barcode reading is activated within main application installation process and doesn't require third-party software installation in order to work.

The stores feature shows location of the stores in users vicinity by using Google maps. Every store is represented by its logo on the map. Logos are placed on map as overlay and every logo can be clicked to obtain additional information. Class responsible for displaying the map and handling Google maps API calls is StoresScreen. Built in navigation software (like Google Maps) is used to navigate the user to the store by using user's current location and store's location. User can choose which application (if he has multiple available applications) he would like to use when choosing to get directions to the store. This is built in android functionality in which every application that can serve that kind of requests have defined activity that announces itself to the system. Process of invoking that behavior is implemented in StoresScreen class as well.

3.2 Conceptual design

3.2.1 Server Application Design

The Figure 3.2 shows the conceptual design of SmartCart Server. The SmartCart server is responsible for handling the client's requests. These requests include accessing or modifying data related to stores, products, products lists, shopping lists. The server has 4 components including the database. The core/application logic of our server is implemented inside the Controller module. The receiving and sending messages to the mobile

application is handled by the Web Services module, while secure and safe access to the database is provided by the Storage Controller. The component specifications of server application are provided in Table 3.1

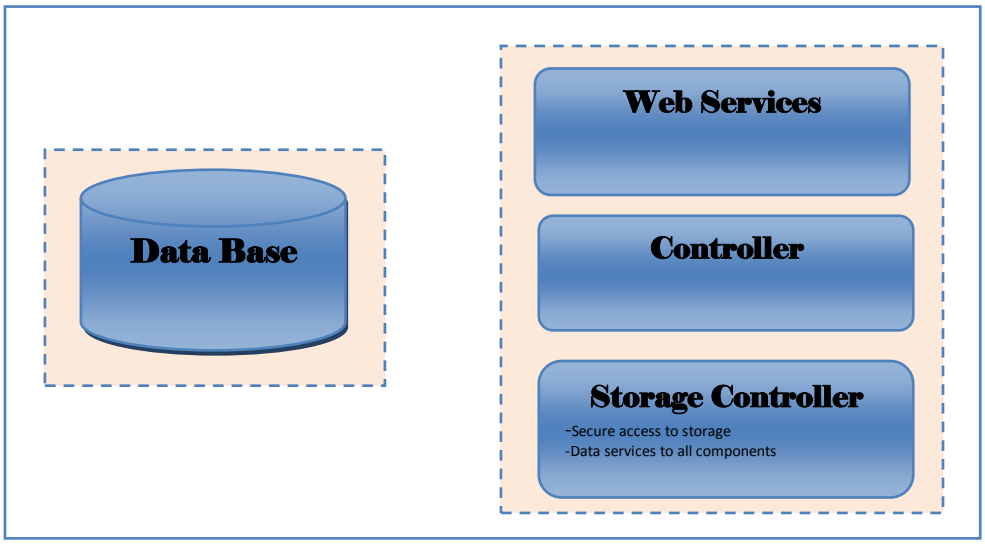

Figure 3.2: SmartCart Server Conceptual Design.

3.2.2 Android Client Application Design

The Figure 3.3 shows the abstract view of our Android client. The Android application conceptually has 6 major components. This division is based on the functionalities each component is providing to the users. The purpose of each component is explained in the next chapter, Table 3.2.

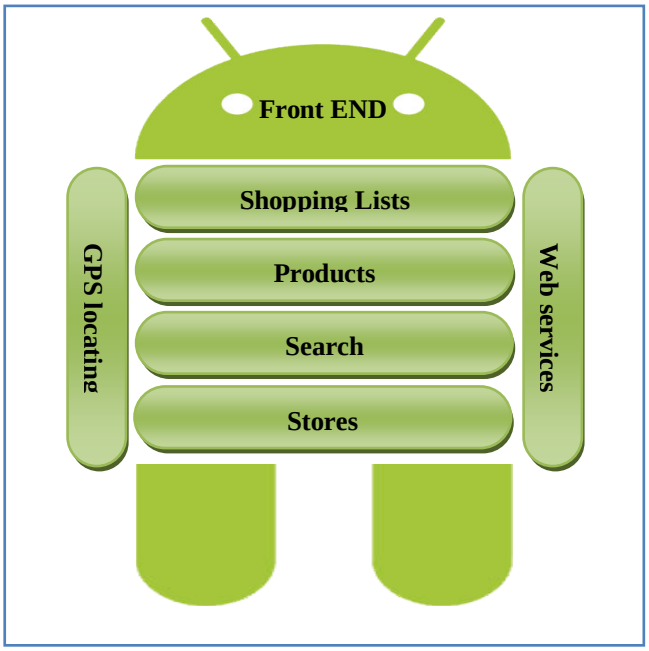

Figure 3.3: Android Clint

3.3 System specification

The system specification of SmartCart server and Android Client are described in the table 3.1 and 3.2 respectively.

Table 3.1: Showing the components of SmartCart Server

SmartCart Server		
Components	**Description**	**Technology**
Database	▪ The database will store the information of the overall SmartCart system.	MS SQL Server 2008 ▪ A powerful RDBMS which have very strong features for storing, managing and controlling databases
Web Services	▪ Deals with processing client requests for incoming and outgoing data.	WCF with JSON Enabled
Controller	▪ Represents the business logic of the server application	.NET Framework 4.0
Storage Controller	▪ Providing data services to all the components in the system ▪ Controls the access to the storage ▪	LINQ to SQL
Store loader	▪ Desktop application that loads a text file with stores and inserts those stores into the database	Windows Forms
Konzum Web Scraper	▪ Web scraper that collects the products and their prices from the Konzum web site and inserts them into the database	Windows Presentation Foundation

Table 3.2: SmartCart Android Client specification

SmartCart Android Client		
Components	**Description**	**Technology**
Front End	▪ The Frontend will acquire the data for every user interface and will display it to the user.	▪ Android SDK
Shopping Lists	▪ Processing services for creating, editing, and maintaining the shopping lists based on lowest prices, stores proximity.	▪ Android SDK

Products	▪ Maintaining the products list on a device and synchronizing it with the server	
Stores	▪ Provides options for controlling and maintaining stores. ▪ Provides GPS assistance for stores ▪ Shows stores on the Map	
Search	▪ Provides user the option to search for the products by product name or by scanning the barcode	
Storage	▪ The Storage part will store temporarily the SmartCart data. This data will work more like a cache for the client to speed up response time. ▪ Will be synced with server for updated information.	
Web Services	▪ This component is responsible for bridging the client with the server for information requests over the internet.	
GPS Locating	▪ This component obtains GPS data from the Android GPS Device and provides the data on requests from other components	
Database	▪ The database will store the information about products and categories depending on current user location.	▪ SQLite

3.4 Error handling

Error	Action
Client application cannot connect to the web server.	An error message is displayed explaining the internet connectivity problem
The user is trying to add a product to a store which does not exist.	User is prompted to add the new store first then allowed to add the product to it.
The user is tryning to set the product price in an invalid format.	The user reenters the price

Store name already exists.	1. An error notification appears in the page saying that the store name already exists and asking user to reenter it. 2. The user reenters the store name

4. Detailed software design

4.1 Sequence diagram

A sequence diagram is made to illustrate the dataflow and interaction between system entities. Figure 4.1 shows a sequence diagram of one possible usage scenario in which the user adds products to a shopping list and gets the suggested stores where he could buy the products from his shopping list. .

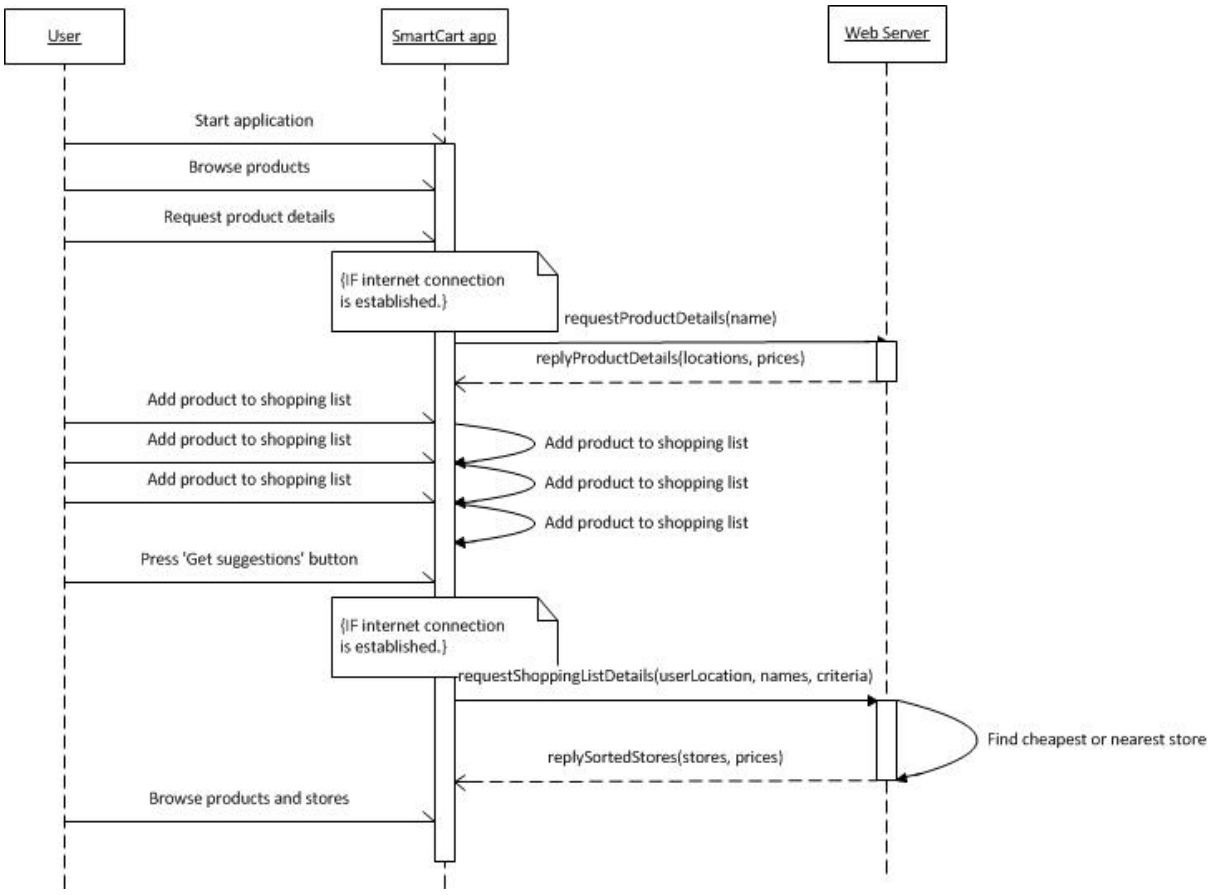

Figure 4.1: Sequence diagram

4.2 Class diagram

4.2.1 SmartCart Server class diagram

Figures 4.2, 4.3 and 4.4 show the class diagram of the server side of the system. Getters and setters are omitted in this class diagram for brevity.

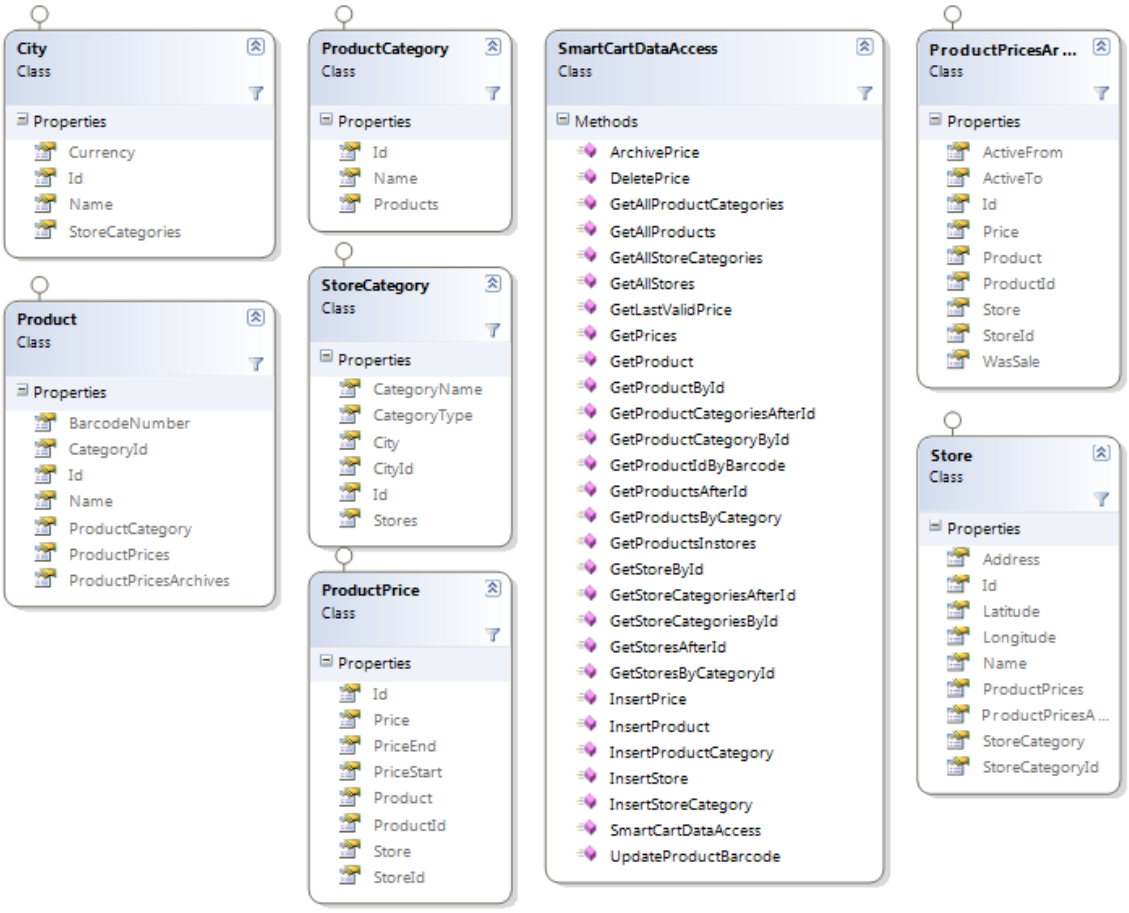

Figure 4.2: Data layer class diagram

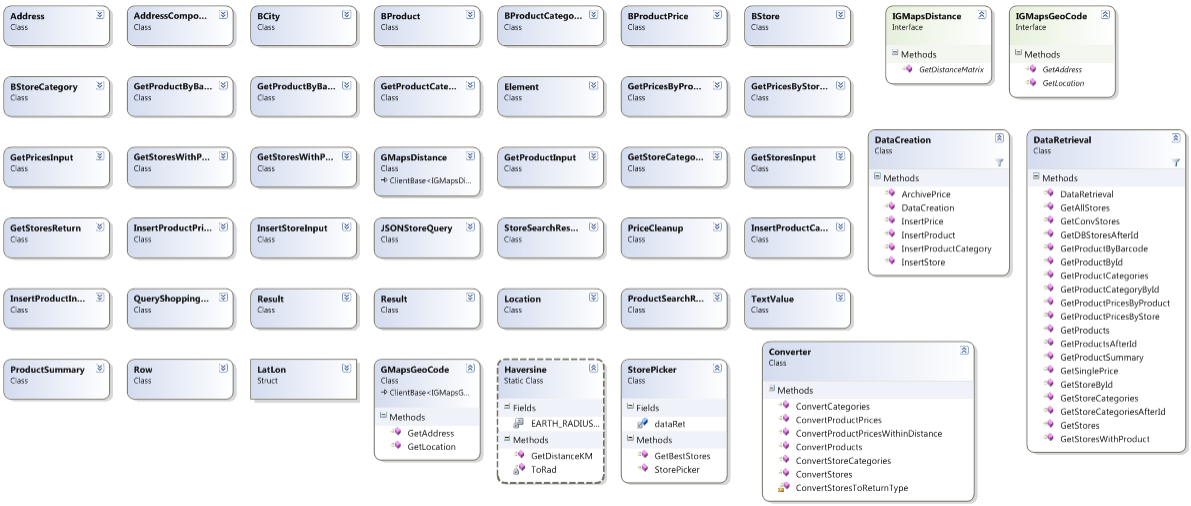

Figure 4.3: Business layer class diagram

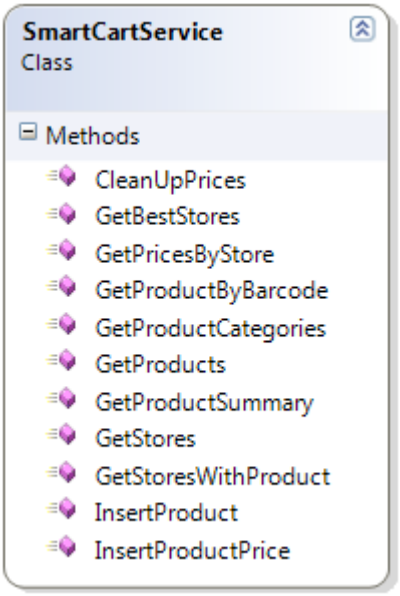

Figure 4.4: Service layer class diagram

4.3 Database diagram

SQL Server 2008 will be used as the database server. Object-relational mapping will be handled via LINQ to SQL. Database operations will be written mainly in LINQ, except the situations where an appreciable increase in speed can be achieved by using T-SQL. Figure 4.5 shows the database scheme!

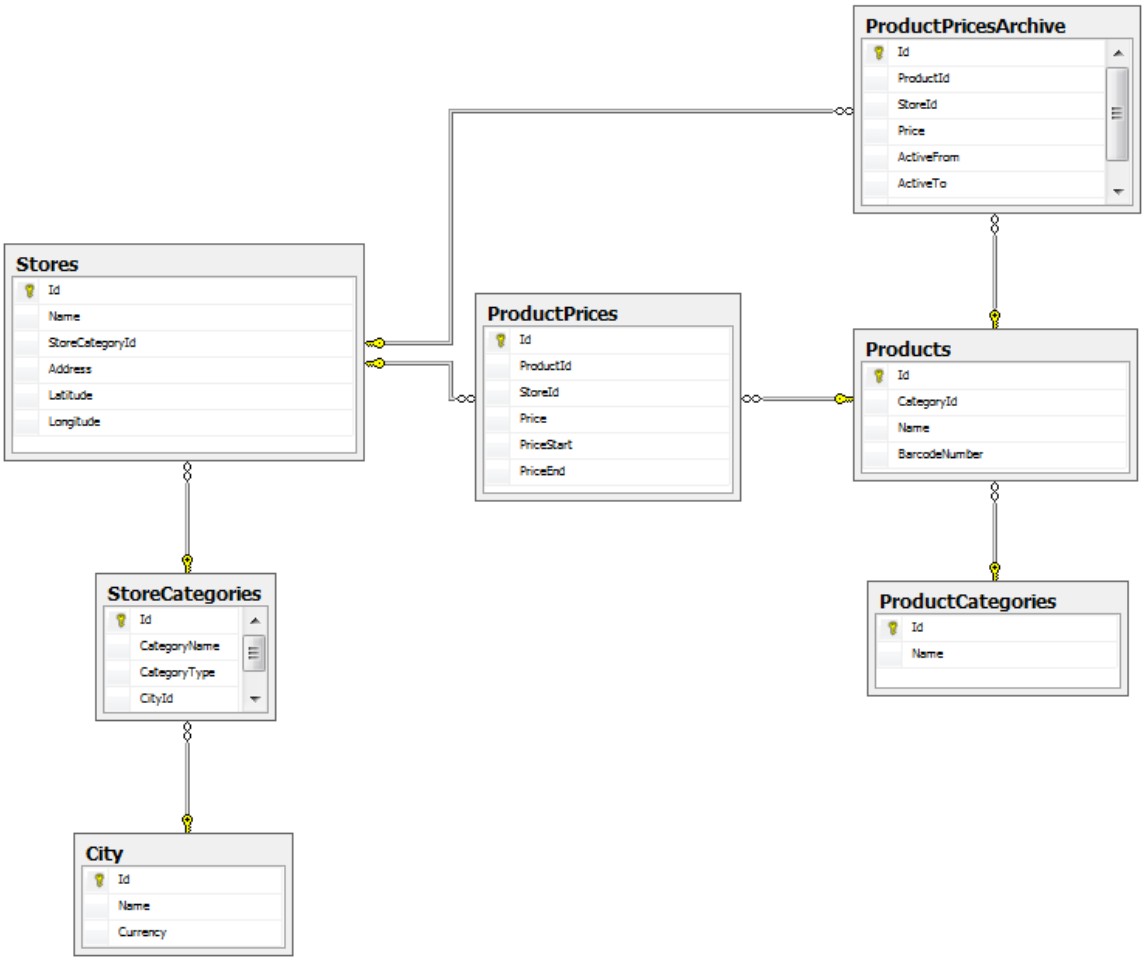

Figure 4.5: Database scheme