SmartCart
Design Description

Version 1.0

Revision History

Date	Version	Description	Author
2011-10-20	0.1	Initial draft	SmartCart Team
2011-24-10	0.8	Revised draft	SmartCartTeam
2011-27-10	0.9	Revised draft no. 2 – ready for the final review before submission	SmartCartTeam
2011-27-10	1.0	First version of the document	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

2. External interfaces

Different interfaces like hardware, software, communication and graphic user interface have to be defined in order to implement a system that enables users to add groceries they need to a list, and find out which are the nearest or the cheapest stores that have the groceries from the list.

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. 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 use. 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.

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

The following figures illustrate just an example for a possible GUI for the SmartCart application.

The application GUI is made to be simple which is shown in Figure 2.1. On the main screen there are just 3 essential buttons which lead on other views: Products, Shopping list and Stores.

Figure 2.2 illustrates the 'Product's view will list all available products with possibility of manually adding new products to the list.

[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

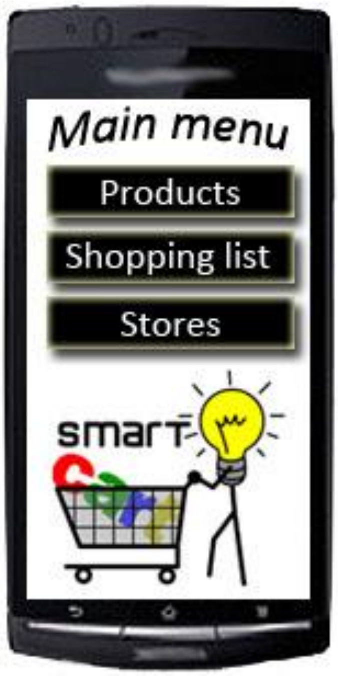


Figure 2.1: Main menu

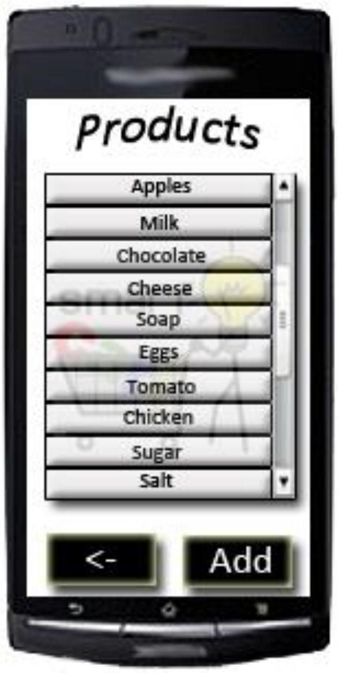


Figure 2.2: Products

Figure 2.3 illustrates that the “Shopping list” view will show currently picked groceries and enable editing of list with operations like add, remove and change quantity. There is also the “Get suggestions” button which task is to start search of the cheapest shopping list or/and nearest store.

Figure 2.4 illustrates the pop-up message which pops up when the user wants to add a particular product to the shopping list.

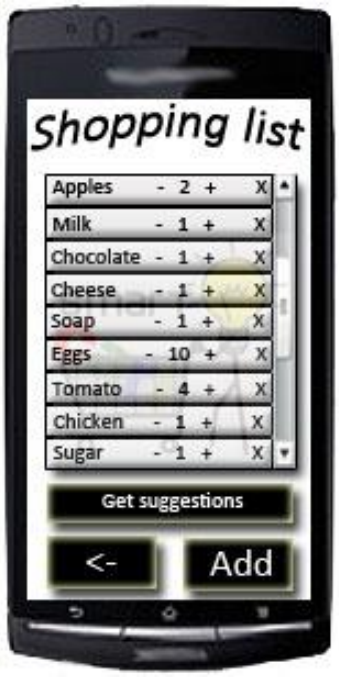


Figure 2.3: Shopping list

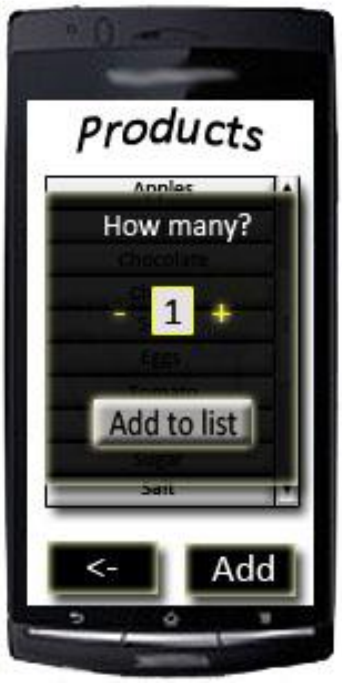
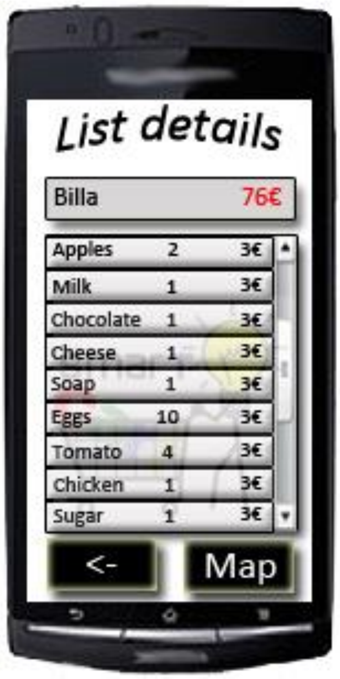


Figure 2.4: How many pop-up

Figure 2.5 illustrates "List details" which shows the name of the store and the shopping list with prices of the products (on the shopping list) in that store and the total price of products on the shopping list in that store.

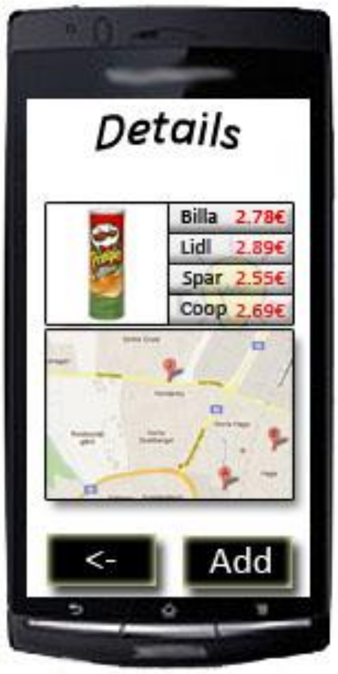
Figure 2.6 illustrates product's "Details" which shows the name, the picture, the price and the location of the product.



List details

Billa 76€

Apples	2	3€
Milk	1	3€
Chocolate	1	3€
Cheese	1	3€
Soap	1	3€
Eggs	10	3€
Tomato	4	3€
Chicken	1	3€
Sugar	1	3€



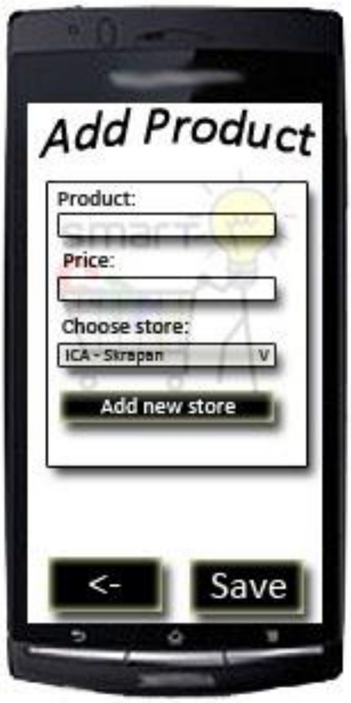
Details

Billa	2.78€
Lidl	2.89€
Spar	2.55€
Coop	2.69€

Figure 2.5: List details

Figure 2.6: Details

Figure 2.7 illustrates the procedure for adding new products to the system by typing product name, price and picking or adding the new store.
Figure 2.8 illustrates the procedure for editing prices which is similar to adding products.

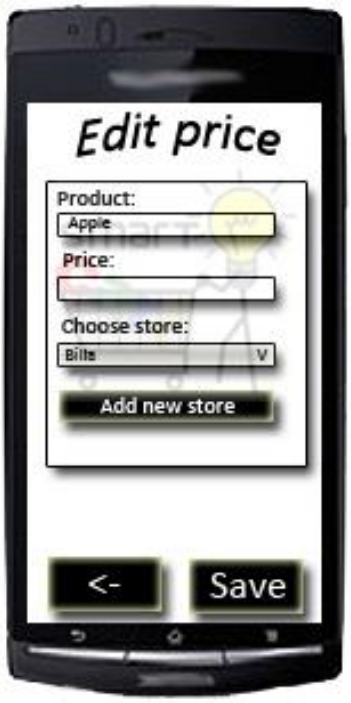


Add Product

Product:

Price:

Choose store: ICA - Skrapan



Edit price

Product: Apple

Price:

Choose store: Billa

Figure 2.7: Add product

Figure 2.8: Edit price

Figure 2.9 illustrates that the “Stores” view shows all available stores on a map.

Figure 2.10 illustrates the procedure of adding a new store by typing its name and address.

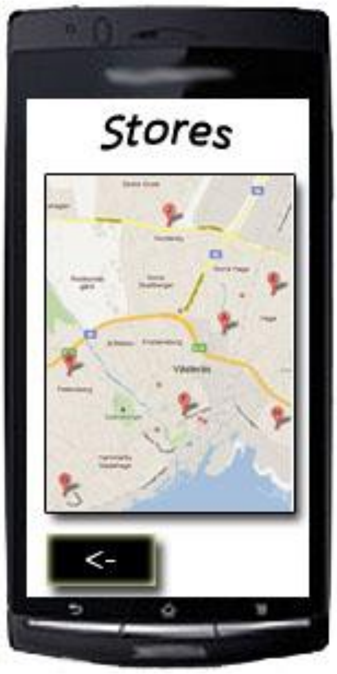


Figure 2.9: Stores

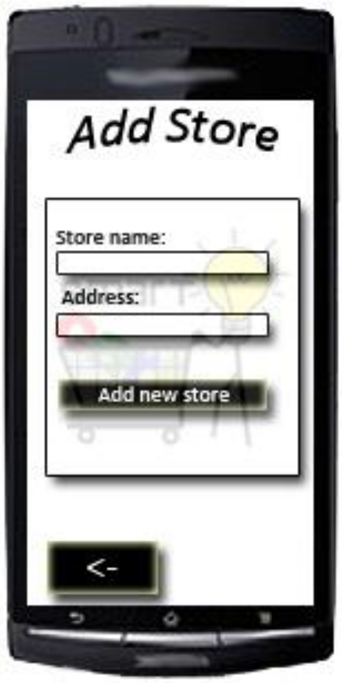


Figure 2.10: Add store

Figure 2.11 illustrates the pop-up that asks the user for search criteria.

Figure 2.12 illustrates search results for the stores with best choice depending on criteria.

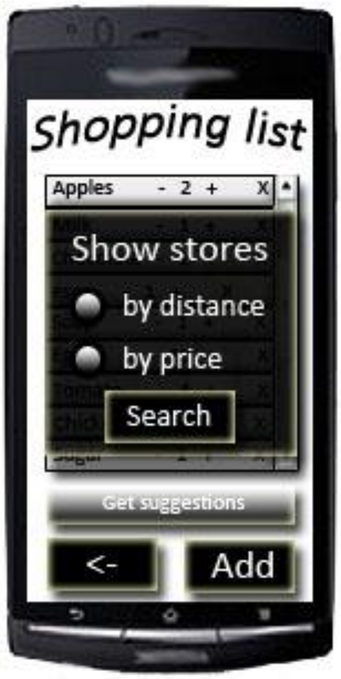
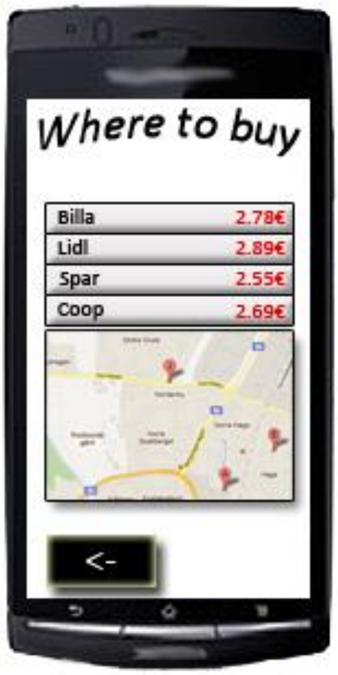


Figure 2.11: Search pop-up



Billa	2.78€
Lidl	2.89€
Spar	2.55€
Coop	2.69€

Figure 2.12: Prices and locations

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.

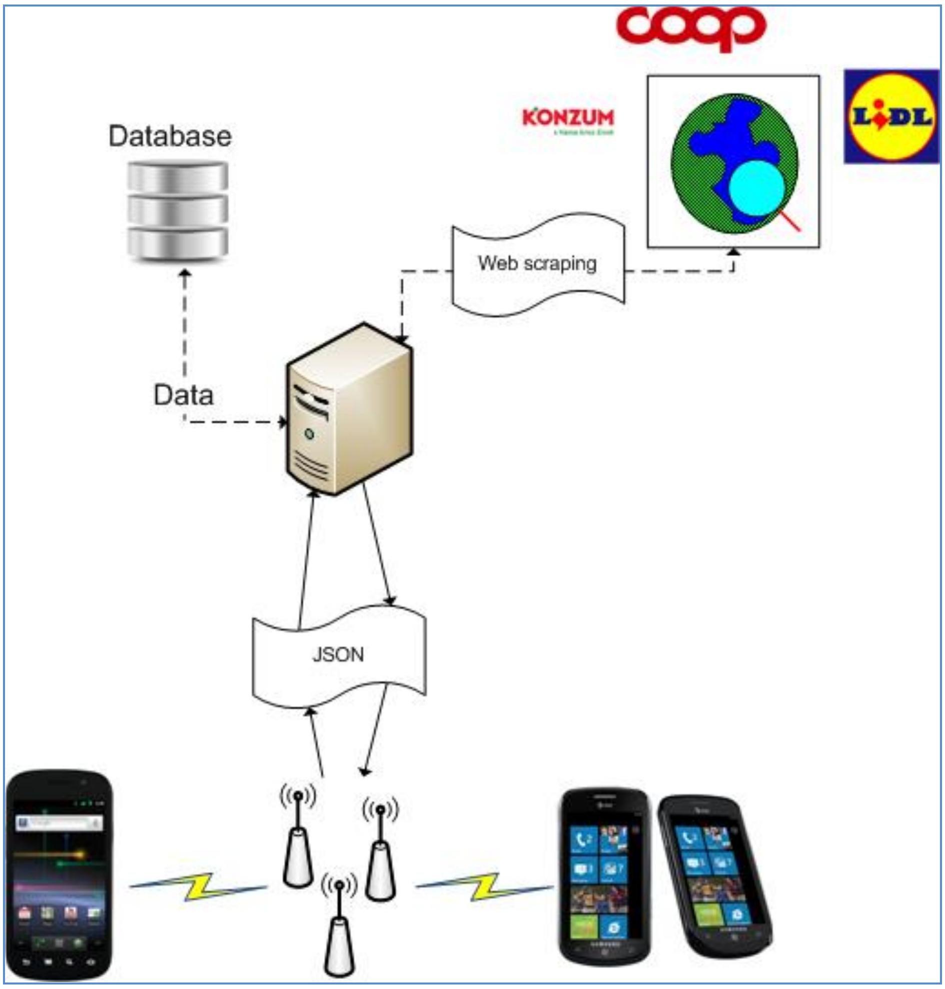


Figure 3.1: Service Oriented Architecture of SmartCart Application

3.1 Conceptual design

3.1.1 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

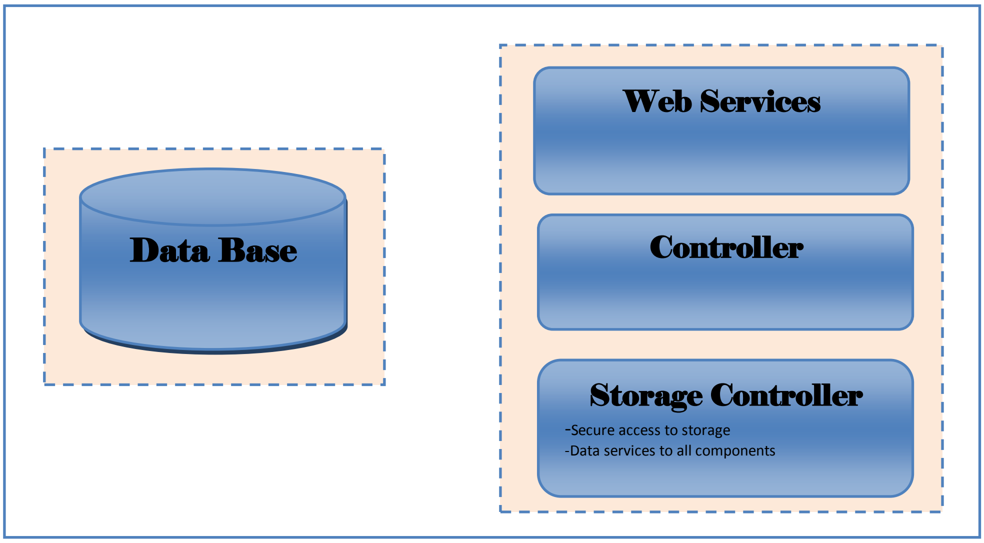


Figure 3.2: SmartCart Server Conceptual Design.

3.1.2 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.

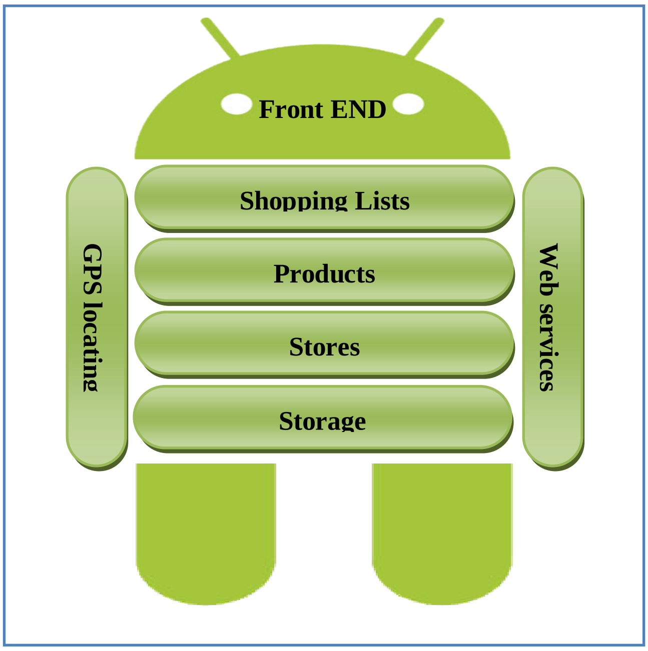


Figure 3.3: Android Clint

3.2 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

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	
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	

3.3 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.   .

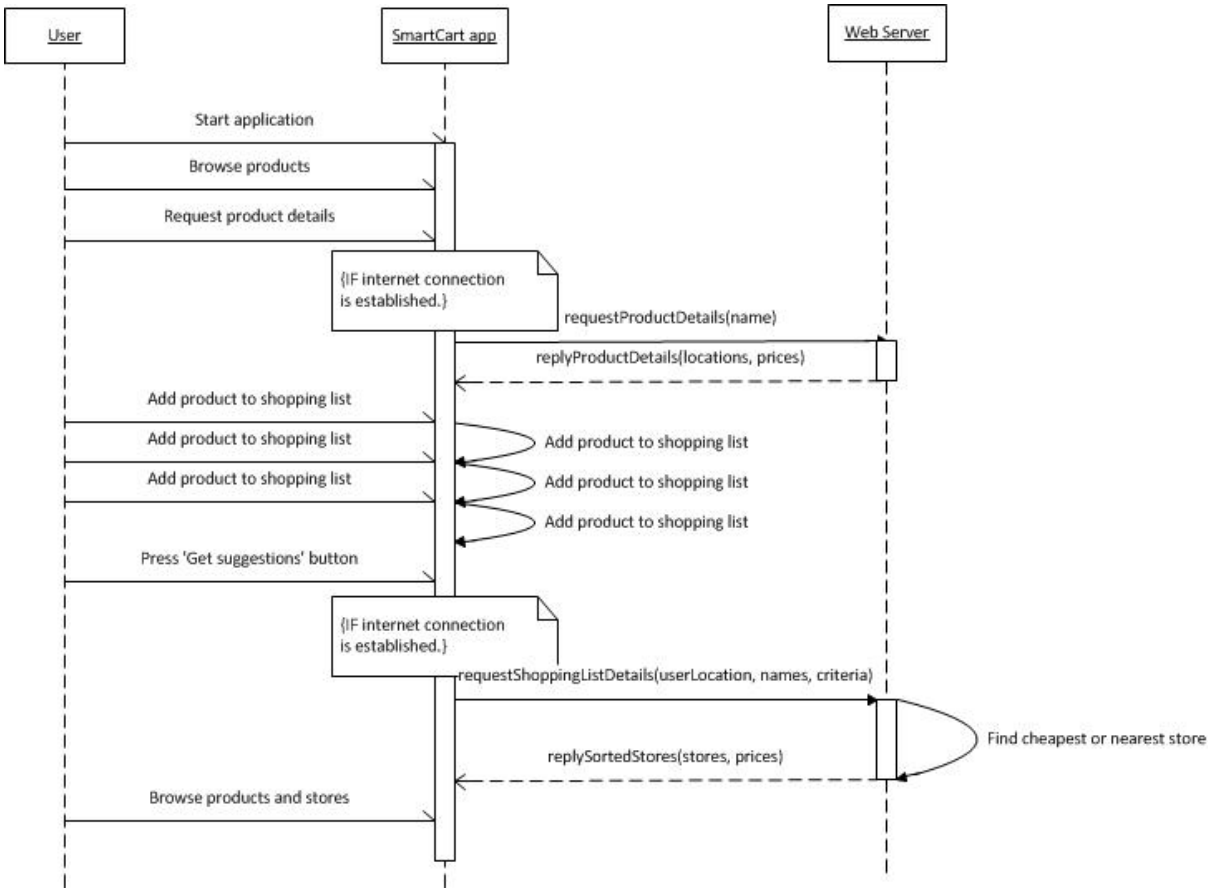
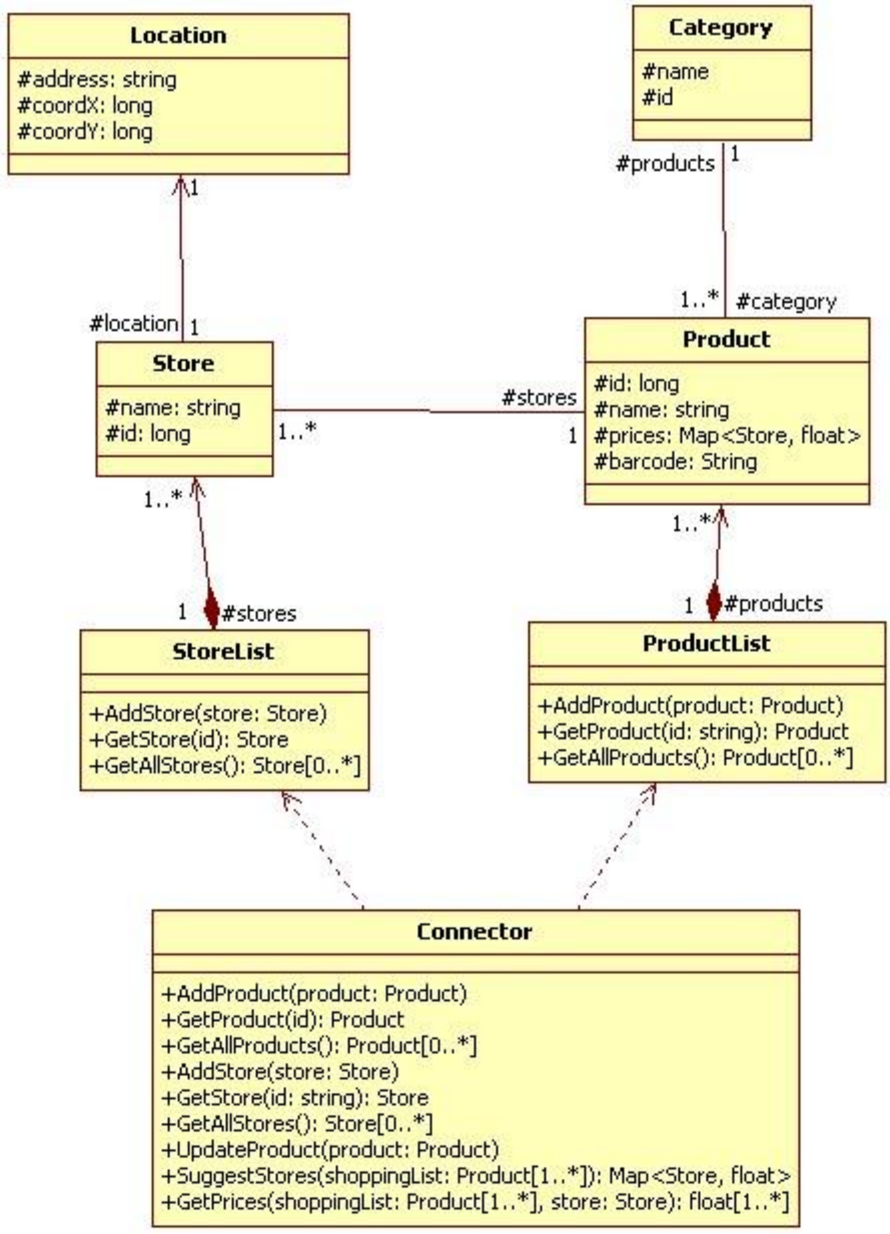

User
SmartCart app
Web Server
Start application
Browse products
Request product details
{IF internet connection is established.}
requestProductDetails(name)
replyProductDetails(locations, prices)
Add product to shopping list
Add product to shopping list
Add product to shopping list
Add product to shopping list
Add product to shopping list
Add product to shopping list
Press 'Get suggestions' button
{IF internet connection is established.}
requestShoppingListDetails(userLocation, names, criteria)
Find cheapest or nearest store
replySortedStores(stores, prices)
Browse products and stores

Figure 4.1: Sequence diagram

4.2 Class diagram

4.2.1 SmartCart Server class diagram

The Figure 4.2 shows the class diagram of the server side of the system. Getters and setters are omitted in this class diagram for brevity.



Location
#address: string
#coordX: long
#coordY: long
1
#location
1
Category
#name
#id
#products
1
1..*
#category
Store
#name: string
#id: long
1..*
#stores
1
Product
#id: long
#name: string
#prices: Map<Store, float>
#barcode: String
1..*
1
#stores
1..*
1
#products
StoreList
+AddStore(store: Store)
+GetStore(id): Store
+GetAllStores(): Store[0..*]
ProductList
+AddProduct(product: Product)
+GetProduct(id: string): Product
+GetAllProducts(): Product[0..*]
Connector
+AddProduct(product: Product)
+GetProduct(id): Product
+GetAllProducts(): Product[0..*]
+AddStore(store: Store)
+GetStore(id: string): Store
+GetAllStores(): Store[0..*]
+UpdateProduct(product: Product)
+SuggestStores(shoppingList: Product[1..*]): Map<Store, float>
+GetPrices(shoppingList: Product[1..*], store: Store): float[1..*]

Figure 4.2: Class diagram

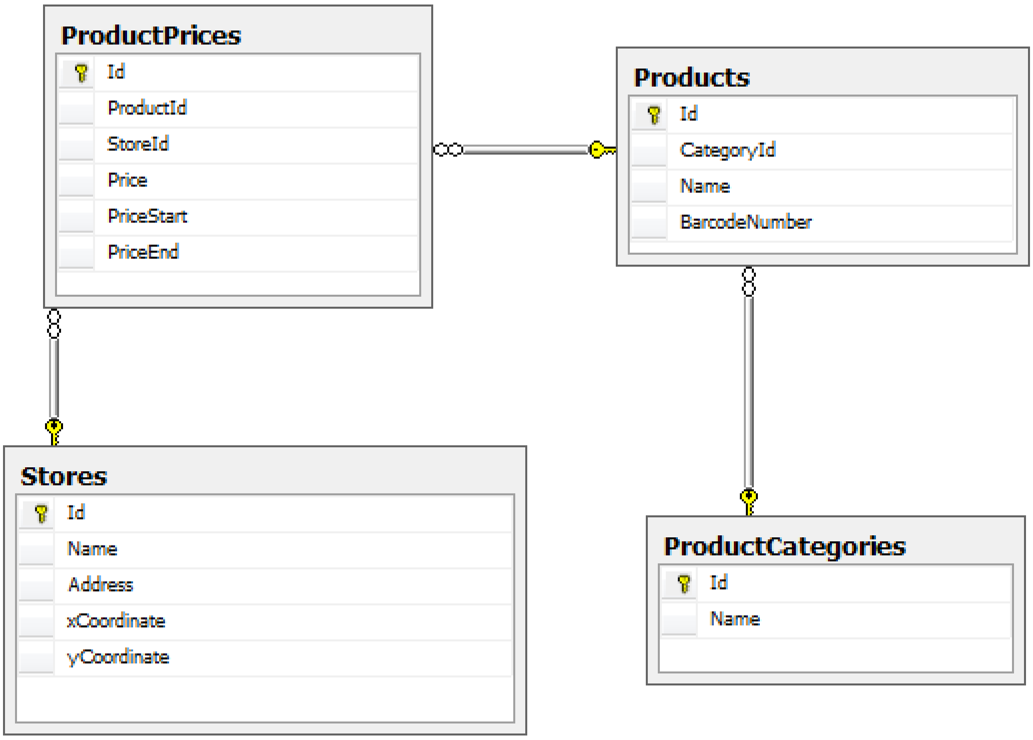
4.3 Database diagram

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.

Below is a description of the database as it will be utilized at the end of the development of each of the three requirement groups.

4.3.1 Level 1 Requirements Database

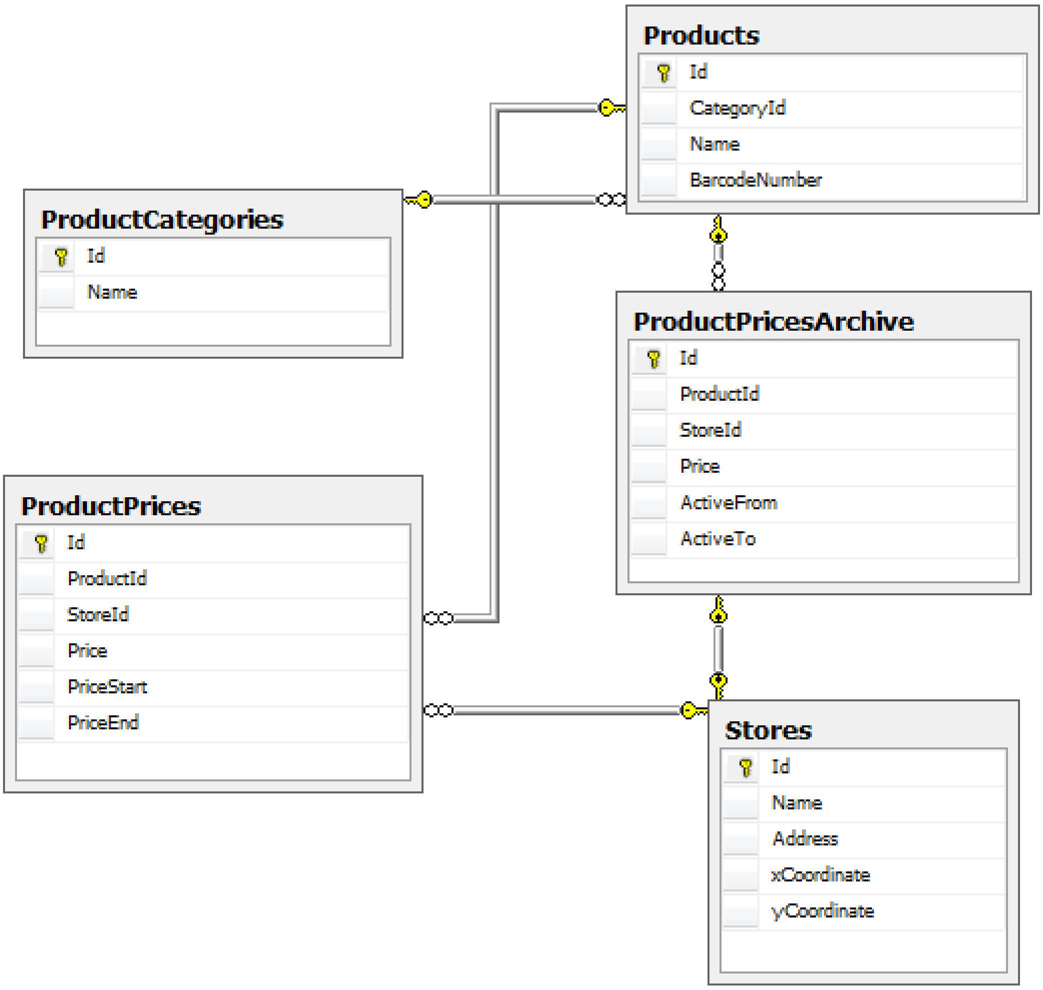
Initially, most storage will be taking place locally, so the server database won’t store any user data, just general data about products and prices.



4.3.2 Level 2 Requirements Database

For the second stage of development, GPS data about stores will be collected and utilized, and stored on the server. Also, archives of the outdated rows in the ProductPrices table will be kept in order to facilitate reverting to a previous price in case the current price is merely a time-limited offer.

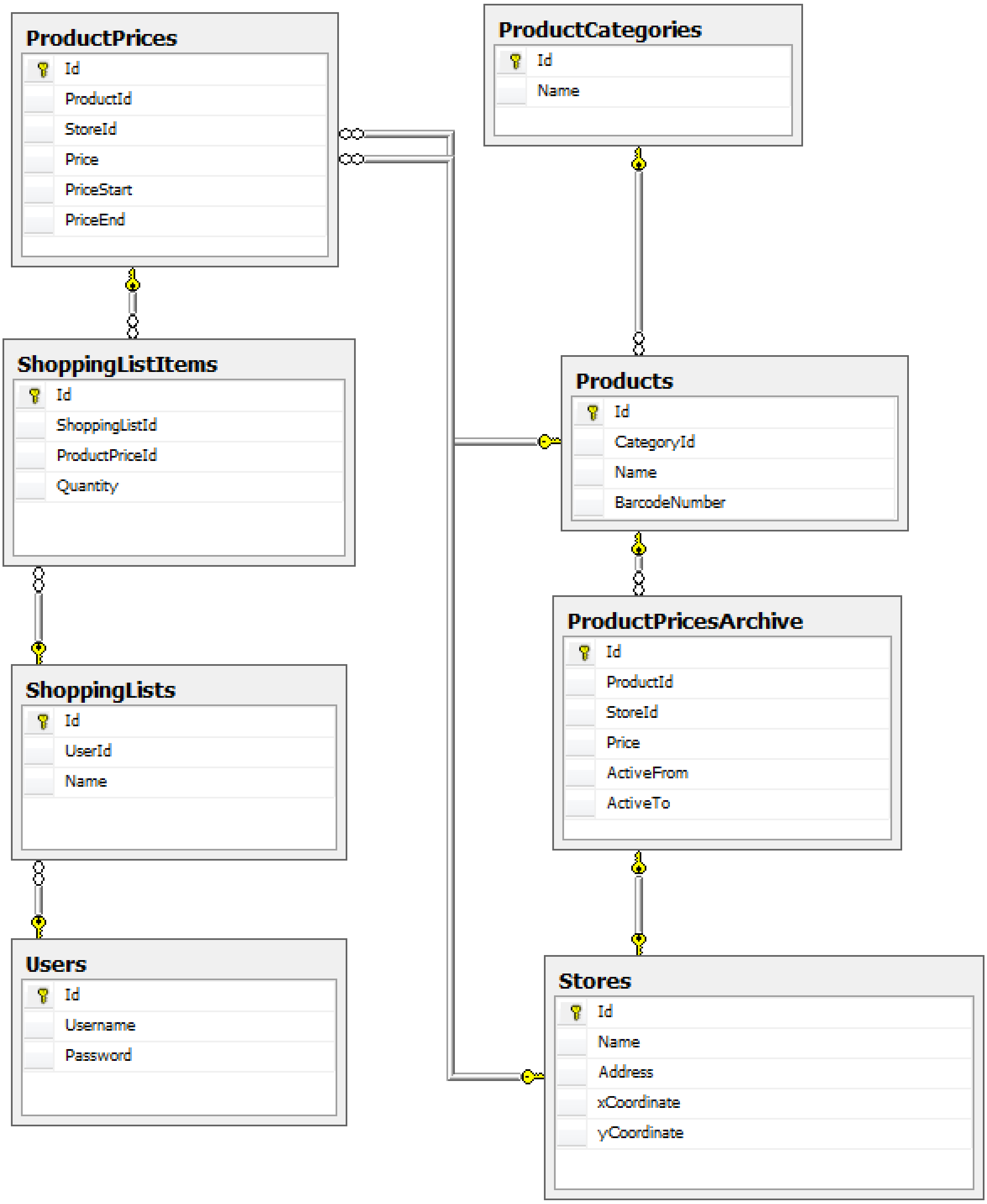
Additionally, barcode information on products is stored so customers can quickly access or input product data via a barcode scanner.



4.3.3 Level 3 Requirements Database

For the final stage of development, personal data pertaining to shopping lists and the like will be moved to the server side enable users more freedom in choosing where and from which device they will use the application.

In this version, user registration is added, and user accounts are linked to shopping lists and items.


ProductPrices
Id
ProductId
StoreId
Price
PriceStart
PriceEnd
ProductCategories
Id
Name
ShoppingListItems
Id
ShoppingListId
ProductPriceId
Quantity
Products
Id
CategoryId
Name
BarcodeNumber
ProductPricesArchive
Id
ProductId
StoreId
Price
ActiveFrom
ActiveTo
ShoppingLists
Id
UserId
Name
Users
Id
Username
Password
Stores
Id
Name
Address
xCoordinate
yCoordinate