Project Requirements

Project Name: Shopping List

Team: Pascal Giehl, Stephan Matter, Neyah Rizzello, Linus Schwab, Orlando Signer
Customer: Mircea Lungu
Technical tutor: Andrei Chis

Revision History

Version	**Date**	**Revision Description**
1.0	30.09.13	First finished version of the document.

Date: September 30, 2013

Table of Contents

1. Introduction

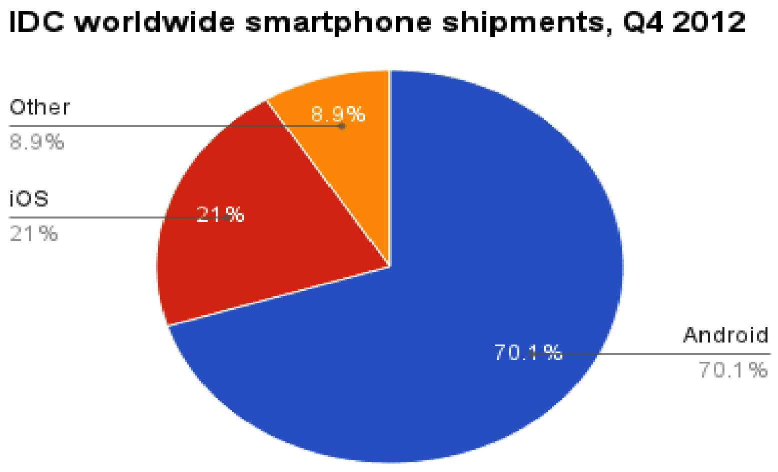
1.1 Purpose

We want to create a program to simplify the shopping experience for people. To reach that goal, we want to create a simple to use program for devices with the android system which allow to create and manage shared shopping lists. This will allow our users not only to have a easy to handle shopping list on their mobile devices, it will also synchronize them with their family and friends. This means that they have always the complete list of all stuff they and others need in their pockets when they enter next time a shop.

1.2 Stakeholders

It is a program with high potential and will be sold through the android play store. The potential market we want to serve are all people with an android device who are between 14 and 80 years old, which is probably almost all of the android users, which is the fastest growing branch of the smartphone device market (see chart).

Till May 2013, there were 900 million android phones activated (according to wikipedia), which would be about our potential market. If the program evolves to a cash cow, it would be able to expand into iOS and Windows Mobile to serve almost all smartphone users.



origin: http://techland.time.com/2013/04/16/ios-vs-android/

1.3 Definitions

1.3.1 Shopping list

A list which is used to write down all things a person needs to buy.

1.3.2 Android/iOS/Windows Mobile

Different operating Systems on mobile devices called smartphones. Android is created by Google, iOS by Apple and Windows Mobile by Microsoft. They are different constructed and so an app from one of those platforms won’t work on any other.

1.4 System overview

1.4.1 Basic functions

- Create different shopping lists (like Migros, Coop, Mall, urgent, long term…)
- Easily add, edit and remove items from shopping lists
- Autocomplete typing
- Informs other users when shopping
- Shared shopping lists (all the time up-to-date and see if someone else is looking at it, indicator if read, notification) and share them also through SMS/Email
- Add friends through phonebook or Email address
- Remember users by phone number, no need to set up account
- Set a date to a shopping list and get a notification when it’s due.
- Color: Classical Android Scene and Font changeable
- Name: very simple and easy to remember (for example: Shopping List, Items)

1.4.2 Planned extensions

- GPS-Location and route to next shop
- Remind user when he forgot something (or funny stuff)
- Make statistic what he bought through the year
- Make statistic about money spent (option to add additional items bought that were not on the shopping list to keep track of money spent)
- Option to add automaticly stuff every period of time (like milk every week)
- Share interesting offers with friends
- Option to add items to database and set places where they are sold

Effort/Usability table to see if planned extensions are advisable (from 1 to 10)

Function	**Programming effort**	**Usability**	**Conclusion**
GPS/route to next shop	7	4	not advisable
Reminders	4	7	advisable
Statistic of items	6	5	possible
Statistic of money	6	7	possible
Add stuff automaticly every period	4	6	advisable
Share offers	6	6	advisable
Database with all items and locations	10	7	not advisable

Further information to these functions follow in the use case scenarios. It's up to the customer to decide whether to integrate these planned extensions/functions or not.

1.5 References

1.5.1 Our references

Our team has worked already on a few android Projects. The biggest Project we realized till now is the "firstAndroidApp" Application which Google created. So we are a more or less experienced team which will have a lot of fun.

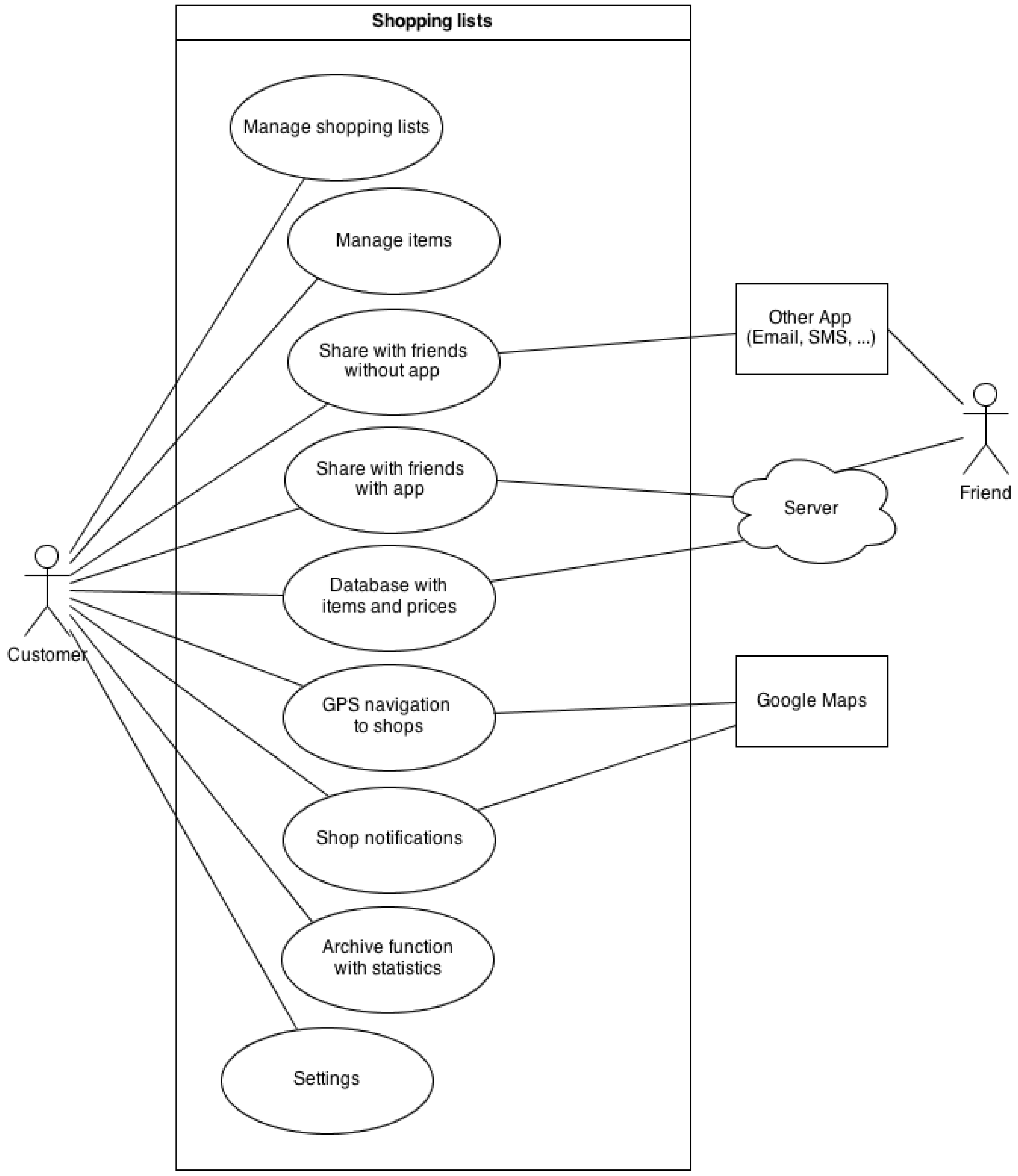
1.5.2 Program references

To get the main idea of the program, it is possible to download some shopping lists which already exist. Of course they won't support all the great function our product will, but for the core idea it's fine.

Suggestions:
1. Out of Milk
2. Rainbow Shopping List
3. Our Groceries Shipping List

2. Use cases

2.1 Diagram


Shopping lists
Manage shopping lists
Manage items
Share with friends without app
Share with friends with app
Database with items and prices
GPS navigation to shops
Shop notifications
Archive function with statistics
Settings
Customer
Other App (Email, SMS, ...)
Server
Friend
Google Maps

2.2 Use cases

2.2.1 Manage shopping lists

2.2.3.1 Actors
Customer

2.2.3.2 Description
As a customer I want to create, update and delete shopping lists. Each list should hava a unique name, for example: Migros, Coop, Mall, …
I want to set a due date for the list and receive a notification when a list is due.

2.2.3.3 Trigger
Customer starts the app and clicks either the ‘New shopping list’ button or selects an already existing shopping list.

2.2.3.4 Pre-conditions
1. To update or delete a list, there must be at least one shopping list

2.2.3.5 Post-conditions
1. The customer created a shopping list
2. The customer changed the name of a list or set a due date.
3. The customer deleted a list.

2.2.3.6 Main Scenario
a) Create a list
1. The customer clicks the ‘New shopping list’ button
2. A new screen appears. The customer can enter informations about the list (name, due date, gps location of the shop) and clicks the save button.
3. System validates the user input (Nonempty and non-existing name, due date is empty or in the future, gps coordinates are either empty or valid)
4. If the input is valid, the system creates a new list and the screen to add items to this list appears (see use case 2.2.2 Manage items)

b) Update a list
1. The customer selects an existing list and clicks the edit button.
2. A new screen appears. The customer can change the informations about the list (name, due date, gps location of the shop) and clicks the save button.
3. System validates the user input (Nonempty and non-existing name, due date is empty or in the future, gps coordinates are either empty or valid)
4. If the input is valid, the system updates the list and closes the screen.

c) Delete a list
1. The customer selects an existing list and clicks the delete button.
2. System shows a warning: ‘Do you really want to delete this list?’ with two buttons: ‘OK’ and ‘Abort’
3.a) Customer clicks abort, the warning disappears.
3.b) Customer clicks ok, the system deletes the selected list.

2.2.3.7 Alternative Scenario

a) and b) 3. Is the input invalid, the system shows an error message and the customer needs to fix the error.

2.2.3.8 Special Requirements

No more than two seconds to create, update or delete a list.

2.2.3.9 Notes

None

2.2.2 Manage items

2.2.3.1 Actors
Customer

2.2.3.2 Description
As a customer I want to create, update and delete items to a shopping list. I want to mark some items as urgent, so they also appear on a special ‘urgent’ list.
When I’m shopping, I want to mark items, that I already put in my shopping cart, as bought, so they don’t appear on the list anymore.

2.2.3.3 Trigger
Customer selects a shopping list and clicks the ‘Add Item’ button.

2.2.3.4 Pre-conditions
1. Customer created at least one shopping list.

2.2.3.5 Post-conditions
1. The customer added an item to a shopping list.
2. The customer updated an item in the list
3. The customer marked an item as bought

2.2.3.6 Main Scenario
a) Add an item to a list
1. The customer selects an existing shopping list.
2. A new screen appears. The customer can enter informations about the item (name, price, urgency) and clicks the save button.
3. System validates the user input (Nonempty name, price is empty or greater than 0)
4. If the input is valid, the system adds the new item to the shopping list

b) Update an item
1. The customer selects an existing item and clicks the edit button.
2. A new screen appears. The customer can change the informations about the item (name, price, urgency) and clicks the save button.
3. System validates the user input Nonempty name, price is empty or greater than 0)
4. If the input is valid, the system updates the list and closes the screen.

c) Mark an item as bought
1. The customer selects an existing list marks an item as bought.
2. System removes the item from the list.

d) Urgency list
1. The customer selects an item as urgent.
2. System adds this item also to a ‘urgent’ list, where all urgent items are stored.

2.2.3.7 Alternative Scenario

a) and b) 3. Is the input invalid, the system shows an error message and the customer needs to fix the error.

2.2.3.8 Special Requirements

No more than two seconds to create or update an item.

2.2.3.9 Notes

None

2.2.3 Share with friends with app

2.2.3.1 Actors
Customer
Friend

2.2.3.2 Description
As a customer I want to share some of my shopping lists with different friends. The lists are automatically synchronized between users who have the app installed and the customer gets notifications on changes.

2.2.3.3 Trigger
Customer presses the share button.

2.2.3.4 Pre-conditions
1. Customer created at least one shopping list
2. Internet connection is available

2.2.3.5 Post-conditions
1. Customer sees who he is sharing the list with
2. Friend(s) receive notification
3. Friend(s) have access to the shared list

2.2.3.6 Main Scenario
1. Customer opens the shopping list he wants to share
2. Customer presses the share button
3. Customer chooses to share with friends with app
4. Customer chooses friend(s) from a list
5. System shows list of invited friends to customer
6. System sends request to friend(s)
7. Friend(s) accepts sharing request
8. System synchronises shopping list with friend
9. System updates invite status list
10. Systems sends notification to customer

2.2.3.7 Alternative Scenario
1a. Customer opens context menu by long-pressing on a shopping list in list overview

    1. Use Case resumes on step 2 of normal flow

7a. Friend refuses sharing request

    1. System notifies customer

2.2.3.8 Special Requirements
Non-functional requirement: internet connection

2.2.3.9 Notes

1. How many friends can the customer share the list with?
2. The sharing option should be implemented in the standard android share menu/list

2.2.4 Share with friends without app

2.2.4.1 Actors

Customer
Friend

2.2.4.2 Description

As a customer I want to share some of my shopping lists with different friends. The customer can share lists by Email, SMS and other apps with friends who don't have the app.

2.2.4.3 Trigger

Customer presses the share button.

2.2.4.4 Pre-conditions

1. Customer created at least one shopping list
2. Internet connection is available

2.2.4.5 Post-conditions

1. Shopping list is sent to friend(s)

2.2.4.6 Main Scenario

1. Customer opens the shopping list he wants to share
2. Customer presses the share button
3. Customer chooses to share by Email, SMS or other apps supported
4. System sends shopping list information (title, items, …) to chosen app
5. Customer chooses friends he wants to send the list to
6. Customer presses send
7. Chosen app sends list to friend(s)
8. System takes the customer back to the shopping list

2.2.4.7 Alternative Scenario

1a. Customer opens context menu by long-pressing on a shopping list in list overview
    1. Use Case resumes on step 2 of normal flow

2.2.4.8 Special Requirements

Shopping list has to be in a format compatible with other apps.

2.2.4.9 Notes

None

2.2.5 GPS navigation to shops

2.2.5.1 Actors
Customer

2.2.5.2 Description
As a customer I want to find the way to a specific shop.

2.2.5.3 Trigger
Customer presses the button “access Maps”

2.2.5.4 Pre-conditions
1. Customer has internet access
2. Customer has the application “Maps” on his device
3. Customer allows the application to locate him.

2.2.5.5 Post-conditions
1. Customer gets the nearest location of the shop he’s looking for.

2.2.5.6 Main Scenario
1. Customer press the “access Maps” button
2. Customer selects the option “my position”
3. Customer selects the option “search”
4. Customer enters the shop he’s looking for
5. Customer gets the nearest location of the shop he’s looking for

2.2.5.7 Alternative Scenario
None.

2.2.5.8 Special Requirements
Functional requirements:

- Must have a button to access “Maps”

2.2.5.9 Notes
None

2.2.6 Shop notifications

2.2.6.1 Actors
Customer

2.2.6.2 Description
As a customer I want to be notified when I come across a shop which is on one of my list/item

2.2.6.3 Trigger
Customer adds a shop to a list/item

2.2.6.4 Pre-conditions
1. Customer has internet access
2. Customer has the application “Maps” on his device
3. Customer allows the application to locate him.

2.2.6.5 Post-conditions
1. Customer is notified when he passes across a shop which is on one of his list/item

2.2.6.6 Main Scenario
1. Customer specifies a particular shop for some lists/items
2. Customer is notified if he’s close to this shop

2.2.6.7 Alternative Scenario
None.

2.2.6.8 Special Requirements
Functional requirements:

- Be able to add a specific shop to a list/item
- Get notified if the shop is located by “Maps” in a restrict area of “my position”

2.2.6.9 Notes
None

2.2.7 Archive function with statistics

2.2.7.1 Actors
Customer

2.2.7.2 Description
As a customer I want to browse through my old shopping lists and get statistics (amount of specific product bought, money spent, etc.) over all my current and archived shopping lists.

2.2.7.3 Trigger
Customer enters the archive menu

2.2.7.4 Pre-conditions
1. Customer has archived shopping lists

2.2.7.5 Post-conditions
1. Customer gets an overview over all shopping lists and items he bought.
2. Customer can watch several statistics over his archived lists / items like how much milk he bought through the last year.

2.2.7.6 Main scenario
1. Customer enters the archive menu.
2. He gets an overview over all his archived shopping lists.
3. Through a submenu he can choose which statistics he wants to show up.

2.2.7.7 Alternative scenario
1. Customer enters the archive menu.
2. He restores or copies an old shopping list he used before to his current shopping lists.

2.2.7.8 Special requirements
Functional requirements:

- All lists / items the customer bought need to be moved to the archive.
- Archived lists / items need to be restorable from the archive.
- Archived lists can be deleted definitively.

Non-functional requirements:

- The archive should not occupy too much storage space.

2.2.7.9 Notes

1. Archived shopping lists will occupy a lot of storage space.
2. The following statistics will be provided:
    - amount of specific items bought over a specific time period (e.g. 3l milk / 1 week)
    - spent money on specific items (if price is available)
    - spent money over all or a subset of archived lists (if price is available)
    - popular items / which items the customer buys most

2.2.8 Database with items and prices from different stores

2.2.8.1 Actors

All customers which use the app

2.2.9.2 Description

As a customer I don’t want to enter every new item by myself. I want to search through a database which is available over the internet and provides different items from different stores and their prices

2.2.9.3 Trigger

Customer adds an item to one of his shopping lists. He automatically gets recommendations for items from the database which provide the price, an image of the product and possible locations to buy it.

2.2.9.4 Pre-conditions

1. A database with as many as possible items needs to be set up.
2. The database needs to be accessible over the internet.
3. Other customers need to provide initial information for every product in the database (name, price, image)
4. The customer needs to enter the same (or at least a similar) name as the one which is stored in the database

2.2.9.5 Post-conditions

1. Customer gets recommendations for items from different stores with name, price and image.
2. Customer doesn’t need to enter the whole data for each item he adds to a list.

2.2.9.6 Main scenario

1. Customer wants to add an item to one of hist lists.
2. He starts typing “Mi..”
3. He automatically gets recommended “Milk from Migros, Price: CHF 2.-” and “Milk from Coop, Price: CHF 3.-”
4. He selects the product he prefers and it gets added to his shopping list.

2.2.9.7 Alternative scenarios

1.1 Customer wants to browse through different products with the name “Milk”
1.2 He gets an overview over all “Milk” products and can see the exact name, the price and an image of it.
2.1 Customer wants to add a completely new item (which isn’t in the database) to his shopping list.
2.2 He enters all data he knows for it and it gets automatically synchronised with the database for further customers.
3.1 Customer wants to adjust a price of an item in a shop.
3.2 He edits the item in his shopping list and enters the new price

2.2.9.8 Special requirements

Functional requirements:

- database with all items that have been entered by the customers
- each item needs to be synchronised with the database
- offline database, when the customer is not connected to the internet

Non-functional requirements:

- the database should be very responsive
    - short latency between entering a name and providing recommendations
- the database should be reachable 24/7
- the recommendations should be correct (right price, image, name)

2.2.9.9 Notes

This database is not realizable in this project (too time-expensive).
It may be worth a try for later updates.

2.2.9 Settings

2.2.9.1 Actors
Customer

2.2.9.2 Description
As a customer I want to change my font and the colors.

2.2.9.3 Trigger
Customer enters the settings menu.

2.2.9.4 Pre-conditions
1. Customer has a valid version of our program on his device

2.2.9.5 Post-conditions
1. Customer has the font combined with the colors he likes

2.2.9.6 Main Scenario
1. Customer enters the setting menu
2. Customer presses on button to change font
3. He chooses the font he likes.
4. He chooses the colors he likes for this font.
5. He finishes the process and returns to the main program.

2.2.9.7 Alternative Scenarios
None

2.2.9.8 Special Requirements
- Can take as long as desired
- Fonts are delivered with program

2.2.9.9 Notes
1. How many fonts do we support?
2. How many colors do we support?

3. Specific requirements

3.1 Functional requirements

- Add and delete shopping lists
- Add and delete items to/from shopping lists
- Send notification to user when a shopping list or an item is due.
- Be able to add a specific shop to a list/item

- Sharing function integrated into the Android sharing menu
- Provide shopping lists in a format usable by other apps
- Synchronize shared shopping lists (and users with access) with server database
- Synchronize list of friends who have the app installed with server database
- Send notifications to other users of a shopping list
- Automatically create user account by phone number for new users
- The app has to work offline (without internet connection)

- Must have a button to access “Maps”
- Get notified if the shop is located by “Maps” in a restrict area of “my position”

- Finished shopping lists / items need to be moved to the archive
- Restoring shopping lists from archive
- Definitively deleting shopping lists from archive/current shopping lists
- Calculate statistics over archived shopping lists or items

- Database with all entered items by all customers on a server

- The look of the app can be changed through a settings menu (font, color)

3.2 Non-functional requirements

- Compatibility with Android 2.2 or later devices (95% of the market)
- Enough performance for no visible delays (max 2 seconds)
- Simple design (standard Android) that allows users to use the app without training
- Compatibility with all resolutions higher than 320 x 240 pixels
- High reliability, bugs should not occur in more than 0.1% cases, they should never lead to user data loss
- Internet connection to share lists with friends