Universität Bern

Introduction to Software Engineering

Shoppyapp

Software Requirements Specification Document

v0.1

02.10.2013

Customer  
Mircea Lungu

Technical Tutor  
Andrei Chis

Authors  
Sammer Puran / Lukas Galliker / Marc Schneiter / Sébastien Broggi

Introduction 1

Purpose 1

Stakeholders 1

System overview 2

References 2

Overall description 3

Use case diagram 3

Use cases 4

|  |  |  |
| --- | --- | --- |
| Version | Date | Revision Description |
| 1 | 02.10.2013 | First version of the document. |
| 1.1 | 08.10.2013 | Document reviewed, changed the use cases. |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

## Purpose

Shoppy is an application for the Android platform to manage shopping lists with the phone. The main purpose of Shoppy is managing your shopping lists and the ability to organize them together with your friends.

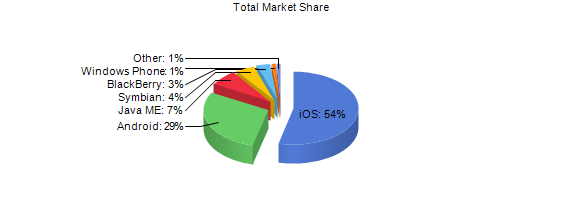
The goal of this application is to be a time-saver and cooperative tool to organize shopping with your friends and family.

## Stakeholders

The application is of use for anybody who likes to create a list before going shopping to not forget what he or she likes to buy. In that sense most people are potential customers of Shoppy. As we plan our initial release of the App only for the Android platform, just owners of an Android driven phones can use our application.

To make the potential amount of users a bit more visible, we include some statistics for mobile platform market shares.

Mobile/Tablet Operating System Market Share as of September, 2013



Source: http://www.netmarketshare.com/

These numbers are difficult to evaluate and differ significantly depending on what source is used, but they can give an impression of the mobile market.

## System overview

The suggested name of the application is Shoppy which is a short and easy to remember name and has a sympathetic and cute subtone.

The user can create multiple shopping lists for different purposes and share each of them with other people. All users of a shopping list can add, remove and edit items from shopping lists, mark them as bought and organize them in categories. All of these actions will be synchronized in case of a shared list. A shared list can be created by selecting people from the phone contacts.

If the user starts typing in a new item for a list, the applications suggests items that were already added to a list in the past

When you open the app it returns to the last opened list and presents you the items you have to buy.

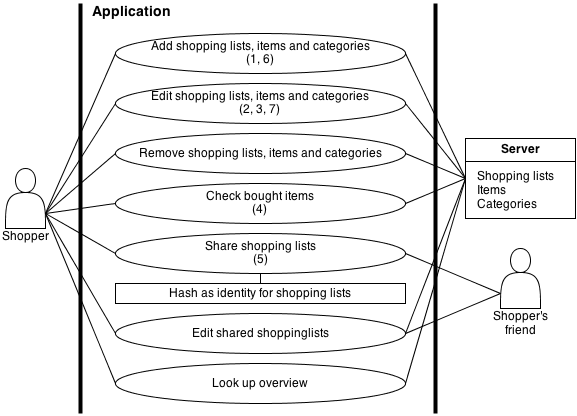
## References

There are a lot of already existing shopping list applications, but each of them has different downsides. We evaluated user comments for many apps and tried to build our feature set with these comments in mind. Some very popular applications are:

* [Out of Milk](https://play.google.com/store/apps/details?id=com.capigami.outofmilk)
* [Einkaufsliste](https://play.google.com/store/apps/details?id=com.DramaProductions.Einkaufen5)
* [myShopi](https://play.google.com/store/apps/details?id=com.agilys.myshopi)

# Overall Description

## Use case diagram



Add Numbers.

## Use cases

1. Create Shopping List:
   1. Actors
      1. Customer
   2. Description
      1. As a customer I want to add a shopping list, name it and want to see it.
   3. Trigger
      1. Select the “new list” option.
   4. Pre-conditions
      1. The customer has to have bought the app.
   5. Post-conditions
      1. A new shopping list has been created.
   6. Main Scenario
      1. Customer opens the app.
      2. Customer selects the “add” option.
      3. Customer names the list.
      4. Customer optionally specifies who to share the list with. (no-person option)
      5. Customer saves the list.
         1. Friends receive invitations to the list.
         2. User receives notification of friends joining his list.
      6. Customer starts editing new list. (“add-item” view)
   7. Alternative Scenarios
      1. Add by importing a list from a friend (sharing):
         1. Customer opens the app.
         2. Customer receives invitation in the notification center.
         3. Customer accepts or declines the invitation.
         4. List is retrieved from the server.
         5. Customer starts editing newly downloaded list.
   8. Special Requirements
      1. None
   9. Notes
      1. The Customer cannot name a list the same as an existing list. It gives an error and the user is prompted to give it another name.

1. Edit Shopping List:
   1. Actors
      1. Customer
   2. Description
      1. As a customer I want to edit an already existing list, i.e. change it’s name and properties.
   3. Trigger
      1. Select the “edit list” option.
   4. Pre-conditions
      1. There already exists a shopping list to edit.
   5. Post-conditions
      1. The shopping list is successful edited (name, category, people to share).
   6. Main Scenario
      1. Customer opens the app.
      2. Customer selects a list from the list screen.
      3. Customer selects the “edit” option for the selected list.
      4. Customer enters a new screen with the option he has to modify the selected list, i.e. rename the list, add categories, add/remove persons with whom he wishes (or not) to have the list shared with.
         1. Rename the list
            1. Customer types another or over the existing name to change the name
            2. The Customer cannot name a list the same as an existing list. It gives an error and the user is prompted to give it another name.
         2. Add Categories
            1. Customer adds one or more categories from an existing categories-library to the list.
         3. Share the list
            1. UC 5, Page 8
      5. Customer saves the changes by selecting the “save list” option and exits the edit-view.
   7. Alternative Scenarios
      1. None
   8. Special Requirements
      1. None
   9. Notes
      1. The Customer cannot name a list the same as an existing list. It gives an error and the user is prompted to give it another name.

1. Add/Remove Item to/from Shopping List:
   1. Actors
      1. Customer
   2. Description
      1. As a customer I want to add items to an already existing Shopping List.
   3. Trigger
      1. Select the “add items” option inside any list. (or “remove-item” on a selected item)
   4. Pre-conditions
      1. There already exists a shopping list to add items to.
   5. Post-conditions
      1. The item/-s is/are added/removed to the Shopping List.
   6. Main Scenario
      1. Adding items:
         1. Customer opens the app.
         2. Customer adds a list (UC 1, Page 4) and is now in the “add items” view.
         3. Customer selects the “add item” button.
         4. Customer specifies the item (name, category, quantity).
         5. Customer saves the changes.
      2. Removing items:
         1. Customer opens the app
         2. Customer selects a list.
         3. Customer selects an item.
         4. Customer selects the “remove” option on that item.
   7. Alternative Scenarios
      1. Add by manually add items to an already existing list:
         1. Customer opens the app.
         2. Customer selects an already existing list.
         3. Customer selects the “add items” option (on that list).
         4. Customer is now in the “add items” view, same as Main Scenario.
   8. Special Requirements
      1. None
   9. Notes
      1. The Customer cannot name an item the same as an existing item. If he does (and the same named items are in the same category), the quantities of both the items are added together.
2. Use Shopping List
   1. Actors
      1. Customer
   2. Description
      1. As a Customer I want an option to check/uncheck items in a shopping list to mark them as bought/not bought.
   3. Trigger
      1. Customer selects the “check” option on an item.
   4. Pre-conditions
      1. There already exists a shopping list the Customer can use.
   5. Post-conditions
      1. All the checked items are greyed out and are no longer to buy.
   6. Main Scenario
      1. Customer starts the app.
      2. Customer chooses the shopping list he wants to buy items from.
      3. Customer selects the “check” option on the item/-s he just bought.
   7. Alternative Scenarios
      1. None
   8. Special Requirements
      1. None
   9. Notes
      1. Main feature - Definitely implemented
3. Share Shopping List
   1. Actors
      1. Customer
      2. Friend of the Customer (who also owns the app)
   2. Description
      1. As a Customer I sometimes want to share one of my shopping lists with my friends, so they can see what I have to buy and buy it for me. The shared list is always up-to-date to simplify things. (If I no longer want my friends to see the list I can edit the list as in UC 2)
   3. Trigger
      1. Customer edits a list (UC 2, Page 5) and invites friends to see (and edit) his list.
   4. Pre-conditions
      1. There has to be an existing list the Customer can share (or the Customer specifies it in the creating of a new list).
      2. The Friend the Customer wants to share the list with has to own the app too.
   5. Post-conditions
      1. The Friend receives an invitation from Customer, which he can accept or decline.
      2. The Customer receives feedback from his friends if they accepted or declined it.
   6. Main Scenario
      1. Customer starts the app.
      2. Customer selects the list he wants to share.
      3. Customer edits the list (UC 2, Page 5).
      4. Customer adds friends he wants to share the list with to the sharing option of the list.
      5. Customer saves the newly edited properties.
      6. Friends receive invitations to the list, which they can accept or decline.
      7. They enjoy the sharing option.
   7. Alternative Scenarios
      1. Customer receives invitation from Friend
         1. Customer starts the app.
         2. Customer receives notification for an invitation to a list.
         3. Customer declines it.
            1. Friend gets notified. (Declination)
            2. Customer’s lists stay the same.
         4. Customer accepts it.
            1. Friend gets notified. (Acceptation)
            2. Friend’s list is imported and added to Customers lists.
            3. Customer has a new list.
   8. Special Requirements
      1. None
   9. Notes
      1. How should sharing exactly be implemented?

1. Create Categories
   1. Actors
      1. Customer
   2. Description
      1. As a Customer I want to create categories to add to a shopping list or an item. (I also want the option to create subcategories?)
   3. Trigger
      1. By selecting the “new category” option.
   4. Pre-conditions
      1. The Customer has to have bought the app.
   5. Post-conditions
      1. The category has been created and is ready to be added to a list, item or category.
   6. Main Scenario
      1. Customer starts the app.
      2. Customer selects the “add category” option.
      3. Customer specifies the category (name).
      4. Customer saves the newly created category.
   7. Alternative Scenarios
      1. None
   8. Special Requirements
      1. None
   9. Notes
      1. Couldn’t we just have an add item button and the customer himself chooses how to arrange his items and if he wants to arrange his items in categories? Can be discussed.
2. Edit Item
   1. Actors
      1. Customer
   2. Description
      1. As a user I want to edit an already existing item, i.e. changing the name or category of this specific item.
   3. Trigger
      1. By selecting the “edit item” option.
   4. Pre-conditions
      1. There exists a shopping list
      2. There exists an item in it.
   5. Post-conditions
      1. The item is edited in the way the Customer wants it to be.
   6. Main Scenario
      1. Customer starts the app.
      2. Customer selects a shopping list.
      3. Customer selects the item, which he wants to edit.
      4. Customer selects the “edit item” option on this item.
      5. Customer edits the item by changing its name or category.
         1. Change the name
            1. Customer rewrites the name in the way he wants it.
         2. Change the category
            1. Customer adds/removes the item from specific categories
      6. Customer saves the changes.
   7. Alternative scenarios
      1. None
   8. Special requirements
      1. None
   9. Notes
      1. The customer cannot rename the item after an already existing item. If he does he will be asked to choose another name.
      2. The Customer cannot name an item the same as an existing item. If he does (and the same named items are in the same category), the quantities of both the items are added together.

Functional requirements

Three different types of elements are implemented in the application

'Items' are the objects you want to buy in the store.

'Categories' contain different items, which are added by the user.

'Shopping lists' are collections of items and categories.

Add, edit and remove items, categories and shopping lists.

Items can be renamed and added to shopping lists and categories.

Categories can be renamed and items can get added or removed.

Shopping lists can be renamed, items can get added or removed and they can be shared with friends.

Items can get checked as ‘bought’, a shopping list gets labeled as 'done' once all containing items are bought.

All items which have not been bought yet can be displayed in a overview.

Shared shopping lists can be edited by all users.

Non-functional requirements (external, performance, etc.)

No login is required to share a shopping list with friends.

Updates on shared shopping lists should be pushed from the server to the mobile devices. This guarantees for a fast performance.

All categories and shopping lists are saved on a server accessible by the Application.

Shopping lists can be accessed by a specific hash in the application (and with a browser ?).