|  |  |
| --- | --- |
|  | PriceCart |
|  |  |
|  | Martynas M;Viktor D;Svetoslav B  Semester Project  12/10/13 |



**Aalborg University Copenhagen**

**ITCOM5 – PriceCart**

Supervisor: Per Lynggard

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Martynas Mockapetris

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Viktor Plamenov Denkov

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Svetoslav Borisov Bakalov

**Copies: 3**

**Pages: 65**

**Finished: 2013-12-14**

Table of Contents

[Introduction 4](#_Toc374983548)

[Problem Formulation 6](#_Toc374983549)

[Limitations 7](#_Toc374983550)

[Methodology 9](#_Toc374983551)

[Concept description 11](#_Toc374983552)

[Analysis 14](#_Toc374983553)

[Market Overview 14](#_Toc374983554)

[Survey 18](#_Toc374983555)

[SWOT 23](#_Toc374983556)

[Target Group 25](#_Toc374983557)

[Stakeholders 26](#_Toc374983558)

[GUI and Visual 28](#_Toc374983559)

[Risk Analysis 31](#_Toc374983560)

[Context Diagram 32](#_Toc374983561)

[Security 34](#_Toc374983562)

[Payment via mobile operator 36](#_Toc374983563)

[User Scenarios 37](#_Toc374983564)

[Use Cases 38](#_Toc374983565)

[Requirements Specification 42](#_Toc374983566)

[Design and Development 47](#_Toc374983567)

[Development 47](#_Toc374983568)

[Class Diagram 48](#_Toc374983569)

[Code reflection 49](#_Toc374983570)

[Testing 54](#_Toc374983571)

[Hierarchy Viewer 60](#_Toc374983572)

[Conclusion 62](#_Toc374983573)

[Future development 63](#_Toc374983574)

[Bibliography 65](#_Toc374983575)

# Introduction

Nowadays mobile phones are widely spread and mobile technologies constantly developed.  
These two factors drastically increase the demand for new services in this area. The smartphones have already become important part of the daily life of people. Using an application these days is more convenient and time saving. And time is becoming more and more valuable in this rapidly changing world. People spend a lot of time in shops, trying to figure what is worth buying or searching for some cheap offers and not everyone is fortunate to live five minutes away from a shop. Thinking about that made us wonder how we can reduce this time spent in walking around the shops, by developing mobile application. Mobile applications could be used from anywhere – while sitting at home or while having a walk outside, as far as you are in possessionof a smartphone and internet access. Having an application which could be used 24/7 in order to check and keep track of one or more product prices, closest to your location, is convenient and easy. Our team will present you with such mobile application among the pages of this report, application based on Android – **PriceCart.**



The main goal for our team is to develop a prototype of a mobile application, based on Android, which will be able to search the cheapest and closest grocery products in a given area. As developers, our final product has to be available on the mobile market (Google Store) and generate profits for our team. The prototype should be functional and give the possibility for users to search, navigate and/or buy desired products – by the end of this project we should have it ready and running on virtual and real environment. A secondary goal will be to attract Shops to provide us with their product databases - doing that will increase the amount of users and profit for both, our team and shops. Following our goals will help us keep the right track of the project and not lose time on unnecessary aspects and features not related to our mobile application.

For this project time, we are limiting ourselves to make a fully working log in window, main menu, with working search engine which would be provided with fake data and locations, possibly existing log out function. If the goals will be fulfilled before the deadlines the goals will be extended to have more options, and the future development will be included to create fully working idea.



## Problem Formulation

Among the pages of this report our team discussed, analyzed and developed an idea regarding:

**How** is it possible to provide the customer with information about the cheapest grocery products in closest proximity by using a smartphone?

* **How** to develop an Android application which users can use in order to search and find the cheapest and closest products?

By analyzing those and other manners, most important and main problem is based in the development of the android application and making it interesting, usable and profitable.



## Limitations

**Project limitations**

The purpose of this project part is to provide an outline of what this paperwork and final product (Android App) will address and what it won’t address. Project limitations may influence how we manage our project and may even determine whether or not we decide to proceed with the project. Project limitations typically fall into several categories. By recognizing these categories, you can focus our investigations and thereby increase the chances that we will discover all limitations affecting the project.

Determining limitations will help us to keep our focus on aspects and parts which are vital for our team.

Our team discussed and categorised main project limitations:

* **Time frame:**
* Due to lack of advanced experience, finalizing the main project idea in a best way requires more time than the provided.
* Finishing project parts on time by following pre-determined deadline – following Gantt Chart
* Delivering the final paperwork on time – December 2013
* **Resources:**
* Following guides on how to write our project, which parts to be included and which not
* Our team is consisted of 3 member, we have different ideas and views on how to implement different ideas, thus Meetings have to be done and we have to agree on one way and follow it.



**Mobile Application limitations**

Application limitations will present what we can’t do with our application even if we have discussed it and analyzed it among the pages of our report. The application at the end of this report is a functional prototype with core functions which presents our idea.

*Please refer to section Future development (see page: 56) for analysis of what we would like to implement on our android application.*

* **Java**

– Programming language which is used for Android development in which our team is not 100% trained

***- Our solution:*** *We followed online solutions and tested code examples.*

* **Using virtual Android device (emulator)**

– During our development, we were using emulator which has limited capabilities with GPS, memory, touch controls and performance.

***-******Our solution:*** *We have used different emulators (ADT; Genymotion) as well as real android device.*

* **Payment method within App**

– As a team we discussed build in payment system, we will not develop and integrate it in our app at this stage of development.

***- Our solution:*** *Our team made a research on how this could be solved (see page: 32)*

* **Updating our Database with Shops’ database**

– We will not provide automated database updates between our DB and external shops’ DB due to Law rules

***- Our solution:*** *If we want to eliminate this limitation we have to contact the major shops and have a discussion and ask for permissions to use the External shops’ DB.*

* **Local database** – at this particular development stage our database is stored locally on the mobile device (SQLi). Database is used for storing user data (emails; passwords; names) and validating among Login. Future development plan shows our idea on mobbing the local database to external (on a server) via http /https connection.

*-* ***Our solution:*** *Our team will revise the code of our android app and re-code it with. We will have to use a webhost server with PHP, API and MySQL compatibilities. (App submits to API, API stores on DB and gives response to App).*



## Methodology

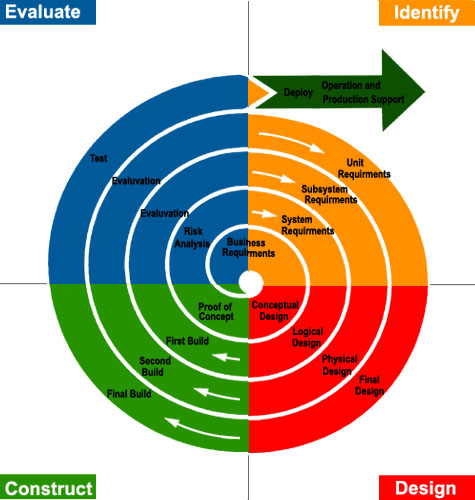
The project main idea is to create an application that will help to save extra money for our target group we are trying to offer it. Application must provide customer with a product results where is the cheaper product is and guide it using navigation possibility. Another feature that will be implemented in the future development is buying product from your home and receiving it as a delivery.

In the beginning of the project the group was lacking of the information how this product should be developed, if this kind of product already exist and if this kind of product actually could be helpful. Methods to answer these kind of questions that were used for the project was reviews from websites, it helped to solve programming issues that were faced during the project. It gave an easier solutions to the tasks we wanted to develop in the application. Researching over the internet also helped to answer if the product already exists and what kind of possible feedback it has received. Interviews that were made with the testers solved the question related to the application if it will be successful. Testers helped to improve the actual concept and to make it more user friendly. The final source that was used in the project were books, books helped to understand the actually application working concept, it also gave extra knowledge in the field we are working in.

Working environment with the group was based on the Gantt chart and all way agreement to work on the project parts. There were no real leader in the group thus this helped to create more freely atmosphere and better working environment in the group. Gantt chart helped to divide work and follow its working process and this helped to monitor the actual working process and speed in charts.

Considering Prototype and Spiral models in the beginning of the project we had to analyse each in order to find its advantages and disadvantages and finally decide upon one.

Prototyping model works better in bigger projects where a product had to be designed, built, tested and then reworked as necessary until a final prototype is achieved from which then a final product can be developed. We found this model as being too complex for the desired android mobile application project because a final product must be finalized and released online on a short deadline. Also in case that the application is a part of a bigger project (website with the same service as the app), its concept has to be finished on time in order for the other parts to go on. This specific model is similar to the trial-and-error which in this case does not apply having well established requirements to follow.

On the other hand **Spiral model[[1]](#footnote-1)** has a linear lifecycle which offered us the opportunity to finish the mobile application on time by working efficient and following the established point of the project plan. It is the most popular version of traditional models and it gave us the possibility to start two activities which are part of the same stage of the project. Once a phase is completed, the development proceeds to the next phase and there is no turning back, which helped us reach the deadline instead of developing prototypes all over again.

Looking at these two models we decided to choose to work with the spiral model because it suits our project flow better than the prototyping model. It has advantages as working path being nonlinear and the precise project plan points which lead to a project finished on time. The possibility of working at the different tasks in the same stage of the project helped us move forward with the project and finalize it faster than expected. Following the pre-set points form the Project plan helped us to keep track and not get confused what has to be done and when. The bellow diagram shows our way of developing the project application. We found it to be the most efficient model.

## Concept description

**Application implementation**

Price checker is an application for the people who are willing to save some money. People are allowed to purchase products via our application in various ways. Application will allow pay for your products by text message, removing money from your phone balance or banking. Price checker application is not a unique idea, however it is not used in Demark what can cause a revolution in Denmark shopping net.

**Application working example**

User is installing the application, in the very first window a loading screen is appearing. After this user will be guided to the next window where he will have option to sign up or log in. From there if the user will press sign up, user will be lead to the sign up window where the user will be requested to fill up form with his personal information. After the registration is completed user will receive a mail with the confirmation link. The link will lead back to the window where user had to sign up for the account, bar with note: registration is completed please log in will appear, user with newly created account have to log in. User will have option “remember me” meaning that it can add into memory his email name, but not the password.

When user is successfully logged in window with a menu will appear, menu will consists of couple tabs which are: Search engine, search history, settings, credits, log out possibility.

* Search engine: will lead customer/application user to the new tab where user will be allowed to search for one or multiple products. *“The search engine should be able to determine when user is searching one or more products and after each product put a “;” and place the next product in line*“. The application will have two different ways in searching products. When one product is entered search application will provide a new window with shops near the application user with the cheapest product. Then user will be able to choose where to go since the user will be provided with product price, shop name, working time, destination from user current location. If user is searching for more than one product a calculation of all products will be calculated and a total amount with lowest total amount will appear in top and the information provided for the user in a new window will be the same: product prices, shop location, work time, shop name.
* Search history: is considered to be important since some users might be constantly searching for the same product. In order to avoid time consuming, customer can simply check the history and rely on old price or select the product from search history and application through the internet will automatically rescan the prices and provide the new location of cheapest price if there were some changes in prices.
* Settings: window will have couple of options to be edited. The first possible settings are available to modify the search engine, user should be able to enter the distance of how far shops can be located from the application user current location. User should be also able to choose which shops should be included in the search engine since some people use only credit cards and some of the shops do not take credit cards. This would avoid unnecessary searching, possibly improving the time for search engine to find the cheapest product.

The next setting window should include the user information editing, however in order to edit the information user must be requested to re-log in to improve the security in order the mobile phone might be stolen and people could abuse the application if the user did not logged out of the application before. The information that could be modified in user settings is the yellow card information, address, payment type and banking information.

* Promotions: will simply provide the leaflets with discounts if there are any of them in each shot so customer could just simple look through the leaflets and possible buy the product if user finds something interesting.
* Credits: will be only displayed to introduce the application user with the purposes of this application, creator manes and intentions.
* Log out: user must log out from the application when he finished the searching and got the product. This should be done to improve the security, however a possible solution is to make a session time, but this could cause extra problems as: going to the market where the product is located and after session is over it would terminate the program and the data would be lost.

After the user finally picked up the product or products with the location, application should use the navigation either via google maps or already integrated mobile navigation to get to the product. The session time could be implemented if the data which user picked could be stored in the navigation system meaning that two systems cooperating with each other but working independently from each other. This could improve the application security since after the data would be stored in the navigation application our product search application could be terminated.

**Application advantages**

One of the application advantages is that user could find the cheapest product in his area and order it via internet either with banking system or operator phone bill. Both systems should provide the receipts after each purchase is done to have the proof for the seller that he is buying these products and the seller should have the possibility to scan that receipt in that way the system will understand that the transaction was successful. Seller approving the receipt would “inform” the application, bank/mobile operator and the same customer. The customer received the products, seller got the money, application made a successful purchase, and bank or mobile operator has to reserve a specific amount of money for those goods user purchased.

One of the most obvious advantages for the customer side is that customer would be able to save some extra money and get the products right in front of the door. However this might cause problems to the market since it would require them to become more competitive. Smaller shops are able to offer cheaper products, but not many people are actually using them. This could also help to smaller shops to expand their business opportunities looking into a successful long run of this application.

# Analysis

## Market Overview

**Introduction**

Market analysis is a great tool to work with in order to understand what is happening in the market at that specific moment. Market analysis helps to identify competitors and to define marketing advantages and disadvantages when entering a competitive market. When analysing developing product with already existing ones it also helps to identify first threats and weaknesses or strengths and opportunities.

“PriceCart” is a unique android platform application for Denmark market. “PriceCart” main function is to provide application user with the lowest price product user is searching for. Many students or foreigners who move in to live in Denmark are often being surprised by high prices and person has to travel around every shop to save some money. However the application we are trying to develop should be able to offer a solution for the customer. Customer by our search engine will be pointed directly to the cheapest product in his area, moreover a delivery to home will be offered as well.

In order to find out if this kind of application will be successful a good comparisation with other platform applications must be done. A possible competitor was found in google play market under the name “Tilbudsavis”. Application was tested and compared with our mockups and development ideas.

**Tilbudsavis**

With google market score of 3.6/5.0 Tilbudsavis[[2]](#footnote-2) is offering access from MinReklame.dk what gives possibility to enter the largest selection of online catalogues. Tilbudsavis gives possibility easily search for products and read newspapers and furthermore find the nearest store to purchase the needed product. Tilbudsavis is offering all bigger brand shops starting from Grocery (Fakta, Fotex, Netto) shops ending with Mobile operator (Telenor) or House equipment items (Jysk, Silvan).

**Tilbudsugen**

Tilbudsugen[[3]](#footnote-3) in google play market has a score of a 3.0/5.0. The purpose of this application was to keep track on the things that are missing from the fridge. The application is automatically checking whether the goods that customer needs has a discount in the grocery shops such as: Netto, Aldi, Barley etc. Tilbudsuge gives a possibility to choose the chains to monitor and choose preferred distance from the customer. Customer can choose to view current offers as one or to view the stores which has the most entered products by customer in the top of the search. Application gives possibility to:

* Add items by scanning, typing or from advertising paper
* Track item when it goes on discount
* Share shopping list
* Short list by product or shops
* View current flyers
* See shop with the most products from your list

However testing this application some flaws were noticed, since most of the people who would be interested in this kind of application would be students from foreign countries it has only Danish language only. Another disadvantage is that application works slow and sometimes even crash.

|  |  |  |
| --- | --- | --- |
|  | TIlbudsugen | PriceCart |
| Mobile Platform | Android | Android |
| Language | Danish | English |
| Search options | Difficult to find search window | Simple navigation, well explained to reach it |
| Location Options | yes | yes |
| Shops | List of all possible shops | List of grocery markets |
| UI | Messy UI, no order | Easy to understand |
| Clicks until finding results | 4 / 5 | 2 / 3 |
| Directions to Target | Bike / car / walking | Bike / car / walking |
| Sharing grocery list | Yes | No |

Comparing PriceCart with Tilbudsugen it is clear to see that the only advantage comparing to our developing product is sharing grocery list. However the advantage for us is that the Tilbudsugen has a difficult menu and layout, during the testing it was hard to find the options that were mentioned in the application description. Comparing products, PriceCart will have less and better structured application layout meaning that the user will be able to receive results faster and easier than TIlbudsugen.

**Be better than existing ones**

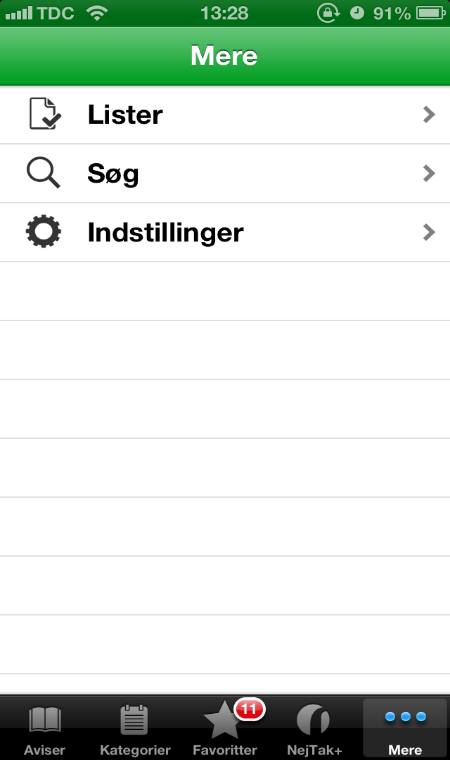
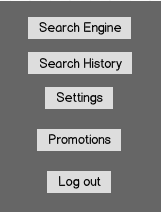
In order to understand better where we can be possibly better a table was created to compare our prototype with the most advanced competitor:

|  |  |  |
| --- | --- | --- |
|  | TIlbudsavis | PriceCart |
| Mobile Platform | iOS, Android | Android |
| Language | Danish | English |
| Search options | Hard to find, not native | Easy to find, native |
| Location Options | yes | yes |
| Shops | List of all possible shops | List of grocery markets |
| UI | Hard to follow | Easy to follow |
| Clicks until finding results | 7 | 2 / 3 |
| Directions to Target | Only by Car | Bike / car / walking |

From this table it is easy to see that compared to the offering platforms we are already huge step behind. However if the product we are trying to develop will be successful extension to Apple market will be done. Since the main target is students, the language we are providing is English, however a multiple language can be implemented under settings section to fit native user. However providing English is one of the advantage since the flow of foreigner students in Denmark is very large, not to mention that most students speak fluent English.

Since we are trying to make our application program as simple as possible, we compared our mockups with Tilbudsavis and the results were quite pleasing since compared to our application it was easy to find compared to competitors search option it was hard to find not to mention it was in Danish only. Both applications are offering location possibilities. Even though competitor is offering a wide range of different shops we are now focusing only on grocery shops, but for the future development this gave us a good perspective where we can expand our application.

Comparing the user interface with our ideas it was easier to follow than competitors. We compared the path to the search engine our application user must be able to reach search possibility within 3 taps while our competitor needs to make 7 taps since our competitor has its search option hidden as it is and extra feature.

 Tilbusavis PriceCart

The competitor possibility to travel is only by car, while our application is offering alternatives to travel by public transport bike or even walking since our directions will be provided via google maps. Furthermore a bigger advantage against competitor is to offer a delivery to home what would be a great attraction of customers’ eye when competing in market.

To sum up the analysis this is the results we have made:

**Tilbudsavis:**

|  |  |  |
| --- | --- | --- |
| The advantages | The disadvantages | Similarities |
| 1. Wide spoken language – English 2. Easy navigation 3. Easy to follow 4. Different traveling possibilities | 1. Small range of shops 2. Android platform only 3. No registration | 1. Location options 2. Same concept 3. Choose which shops to search 4. Choose the range |

**Tilbudsugen:**

|  |  |  |
| --- | --- | --- |
| The advantages | The disadvantages | Similarities |
| 1. Wide spoken language – English 2. Easy navigation 3. Easy to follow 4. Different traveling possibilities 5. Shopping online | 1. User can’t share grocery list 2. User can’t scan groceries to add to database 3. No registration needed 4. Can follow product when it comes to discount | 1. Location options 2. Similar concept 3. Choose between shops 4. Select range from current location |

**Customers and Income**

As we mentioned earlier we are mostly targeting students and to be more specific students who came from abroad. The customer that we chose will be explained in the target group and in survey analysis which will finalize the target customer. Since we are assuming that students are not so rich to define a right price for them should be done in questionnaire which will be mentioned later on. Questionnaire also helps to define if we are actually targeting the right group of people that we stated in stakeholders.

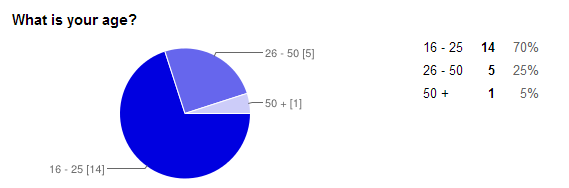
**Conclusion**

Results that were received during this market research helped to stay aware and prioritize requirements to give an edge when entering the market. Analysing market in this kind of structure also helps to define more functional and non-functional requirements that can be combined with MoSCoW[[4]](#footnote-4) method to improve the existing idea of the application. This method helps to show the importance of one or other aspect during the development process.Furthermore when comparing the concept of PriceCart and other competitors helped to identify points which has to be taken more careful than others in SWOT analysis *(see page: 21)*. Making Market analysis is a good starter defining strengths and opportunities, treats and weaknesses.

## Survey

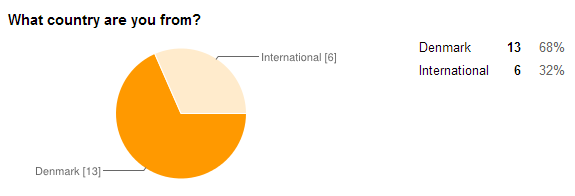
In order to understand better what mass of people we are targeting PriceCart developers agreed to make a survey to get some statistics about the customer and how the application could become more user friendly before releasing it. Survey was provided to 20 people this helped us to receive short feedback how successful our developed application could actually become, what has to be corrected in order to fit user needs. Short 10 question questionnaire was given to each interviewer. First part of questionnaire was meant to help to understand what age of people are more interested in our application. Second part of survey was provided while interviewers were looking through a prototype of our App, this was done to help improving the design and performance.

*Question 1:*



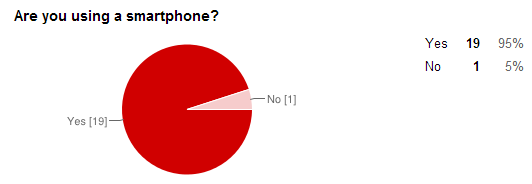
The feedback from this question gave us the answer that our majority of future users’ age is between 16 – 25 years, students and young people who prefer to save money on products. This is suits perfectly our survey since we wanted to find out the actual age of **Target group** that would be willing to use the application. The wide range of age also means that it will provide a good feedback for our stakeholders as well.

*Question 2:*



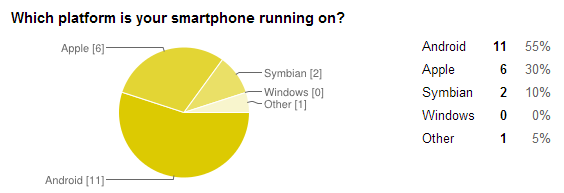
This results are helping us with analyses of who are actually our potential customers. Our focus is both on Danish and international groups of users.

*Question 3:*



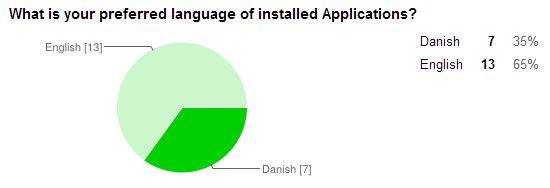
Combined with Question 1, we can conclude that our Target group is using smartphones and we can easily distribute our mobile application.

*Question 4:*



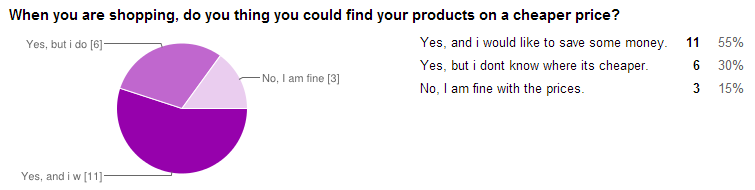
The feedback from this and with combination with previous questions is a plus of our team – our application supports only Android at this development stage. However if we want to expand our idea we will have to consider at least 2 more mobile platforms – Apple / Windows / Symbian.

*Question 5:*



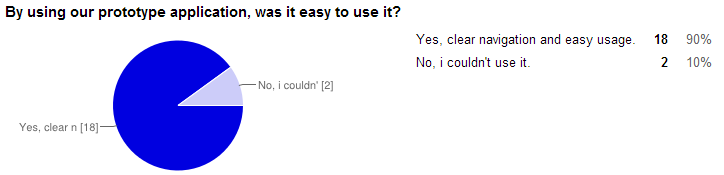
The feedback gathered from this question shows us that the majority of users would prefer the apps to be in English language. Despite that in Question 2 we can see more Danish users, they still prefer the Apps to be in English language. For us it’s a plus since our App would support only English language in the beginning and we can focus on both Danish and International groups of people.

*Question 6:*



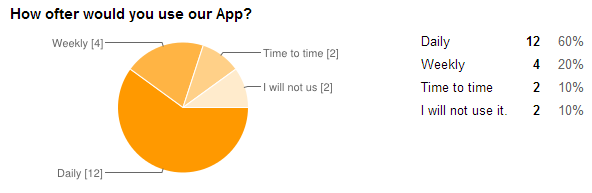
As we are checking if people will be interested in our idea (application) we can see that 55% are willing to save money also the 30% would consider saving money from shopping but they don’t know where it’s cheaper. According to the age of the interviewees (Question 1) it was expected that they are more willing to save money from shopping.

*Question 7:*



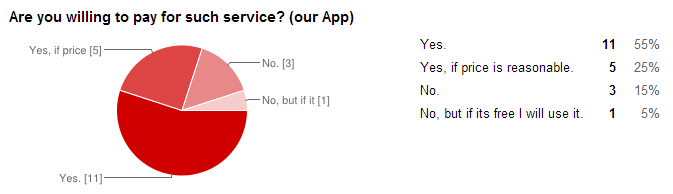
We gave our prototype Android application to a number of selected interviewees and waited to see their feedback – we are satisfied with the 90% positive answers but we have to consider the rest 10% negatives – we would like to achieve 100% positive answers and provide our users with the best application flow possible.

*Question 8:*



We are shopping on a daily, weekly or monthly bases. Our interviewees are willing to use the App for mostly daily and weekly shopping which will guarantee us constant flow of usage and future incomes.

*Question 9:*



Incomes depend on our users willing to pay for the service we are providing. However we have to re-consider our marketing strategy if we decide to charge users for the application. Currently our mobile application is free in order to gain a mass of people to use it.

*Question 10:*

At the end of our survey we, as interviewees tested our Android application (PriceCart), we gathered feedback on what should be improved. The majority of the interviewees (60%) would like the app to remain free in connection with Question 9 (80% would pay for the App). 30% would like also an Apple support of the application which is understandable. 10% would prefer not to register and login in our app but to use it as a free search – if our team decides to have such option we would have to consider in-app advertisements and different limitations.

**Survey Conclusion**

The survey helped to understand that the application is being developed to the right way. It is necessary to make such surveys when developing a product to a big mass in order to find out what users actually want and what possibilities they see in the application. It is also a good way to see first experience when people are looking through the prototype application. The first part of the questionnaire helped to answer if we are actually targeting the right group as we intended from the beginning. In the stakeholders we were targeting people around age 16 - 25. The survey answered that the younger people are more likely to experiment with the applications that might actually help to attract them to use this kind of application constantly. The next part helped to get a short opinion from interviewees if the application that is being created is actually logical and customer useful. Application Demo version helped to gain experience with the application in customers mind and compare it with already existing applications. This kind of testing showed that we are trying to develop a much simpler application that our competitors have. Comparing to the testing results, participants all the time managed to achieve goals much faster and easier than in competitor’s applications. Users also gave some hints that GUI could actually be improved however interviewers did not exactly specified which parts could be strengthen up. This leads developers to do more researches and try to manage the application to be more user friendly than it already is. The following issues will be covered in the application demo testing where user will be actually able to test the application and see what is wrong with it. Future development must provide more answers than questions.

**Link to the survey** <http://goo.gl/8DDJia>

## SWOT

SWOT analysis emphasize the internal environment[[5]](#footnote-5) of the project which helped us figure out the important parts of our application. This bellow analysis will present the soft and strong points of our idea.

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| * Unique for Denmark * User-friendly * Faster and easy access to Danish products * Mobility * Android market * Build-in payment * Build-in Map * Free of charge | * 3G / Wi-Fi Dependent * GPS dependent * OS Dependent – Android |
| **Opportunities** | **Threats** |
| * Advertisements * Profitable * Future development | * Competitors * Legal problems with DK shops * Shops’ Databases |

**Description:**

**Strengths** – Our application is unique in its service and experience for Danish users. Although there are similar applications for checking prices, none of them is actually providing complete list of shops and requested price and showing the distance to it. Our application can provide the cheapest product and save time to our customers for looking through different shops’ websites and apps to find which product is the cheapest. Our app is made for Android smartphones which are widely spread among nowadays generations. The app provides information for products in the area where the user is located. Another advantage is that payment for the desired product can be made via the phone – bank payment; internet; operator phone bill. The biggest advantage of our app is that its saves money for the customers (app users). The easy usage and clean design are yet another advantage - there are no too much fields and confusing options.

**Weaknesses –** Our application will require a 3G (mobile data) or Wi-Fi connectivity in order to be able to show results which could lead to a negative attitude from user point of view. We should consider that not everybody has mobile data plans on their mobile devices. Another negative point is that the device should have GPS module - almost every smartphone has but there models without GPS and location possibilities. Our application will be compatible for Android Devices only which automatically gives us a minus, majority of Danish users are using iOS  devices.

**Opportunities –** The scope of our application is wide in a sense where we can add features such as advertising – banners or splash screens leading to apps or we can advertise specific shops and their products. In this way we can benefit from our application and shops will get more customers. If we are going to compete with the majority of the market one of the opportunity is to enter Apple market what will possibly increase popularity in application and in same time reduce one of the major weakness.

**Threats –** As further researches of our survey will tell mobile market is hungry for new and fresh ideas thus we have to be aware of competitors which will provide similar applications. We have to develop our app and keep it interesting and unique for our current and future users.

## Target Group

Every successful mobile application has a target group - group of individuals chosen by age, gender and contribution. It’s very important the newly developed app to suit the target group. **PriceCart** targets individuals above 16 years up to 50+. As survey results will tell, youngsters are more interested to experiment with the applications and older people are more likely to test something that can actually help in their daily life. Furthermore our main target group youngsters are more likely to spend money more easily than older people and by this app we are hoping to cut their budged. In that case the focus is divided according to 2 main age groups:

* **From 16 to 25 years old**
* Students and young people: Our main focus is set to this group of users. Young customers will be more willing to use the app and searching for cheapest and closest product. This group of people prefers to buy cheaper products since monthly funds are limited – this makes our app the best choice for them. A plus is the growing usage of smartphones among this group age. *(see Survey page: 18)*
* **From 26 to 50+ years old**
* Mature group of people which prefers to buy products with better quality instead of the cheapest. However our application will be interesting because it can help this group of people to save money as well. People from this age group are more likely to choose a more expensive product then the cheapest.

People of any gender and working status, students or employees, are more likely to use our application, than young kids and elder individuals. Target group defines where the focus should be, promoting and advertising products suitable for the particular group of people. The app has to become familiar with customers habits, behaviours likes and dislikes. Choosing the correct market group is essential since our app can’t accommodate everyone’s preferences.

In order to determine real usage results our group made a survey and collected 20 answers which will be presented among the pages of this report – Reference: see page 17



## Stakeholders

Stakeholders analysis will help us find out who are the people that affect and interact with the mobile application more, what effect do they have over the app and its development and how can they help in order to improve the service, move innovative ideas forward and create more awareness about what the app does and will do. To cover most of the raised questions our team will make a survey which will determine whether our customers are satisfied or not and note their comments and ideas for improvement of the app. Survey will be presented among the pages of this report – Reference: see page 17

There are 5 main types of stakeholders which will have to be taken into consideration:

* 1. **Daily Users**

High power, interested people: these are the people that must be fully engage and make the greatest efforts to satisfy; this group of people has to be managed closely and kept satisfied.

* 1. **Weekly Users**

High power, less interested people: put enough work in with these people to keep them satisfied, but not so much that they become bored with your message; the group must be satisfied.

* 1. **Rare Users**

Low power, interested people: keep these people adequately informed, and talk to them to ensure that no major issues are arising. These people can often be very helpful with the detail of your project; this type should be monitored with minimum effort but informed.

* 1. **Visitors**

Low power, less interested people: again, monitor these people, but do not bore them with excessive communication; this group has to be kept informed.

* 1. **Shops**

Shop, providing their databases and products to us. This group has to be manages closely and with high support.

|  |  |  |
| --- | --- | --- |
| **Stakeholder Role** | **Communication Approach** | **Desired Support** |
| Daily Users | Manage Closely | High |
| Weekly Users | Keep Satisfied | High |
| Rare Users | Keep Informed | Medium |
| Visitors | Keep Informed | Medium |
| Shops | Manage Closely | High |



The more people have contact with the app and use it to search for products, the more interest grows for the online community. The stakeholders can influence good or bad the app and its reputation. Thinking about the bad influence, the first thing would be the competitors, not directly but through “fans” which, without knowing, can comment bad on a case that is presented on the Google Store and thus change the perspective of others.

The stakeholders who are keeping the App and our concept idea running by visiting/using our services (provided by our Mobile application) have to be managed closely and fully satisfied – our app has to generate sufficient amount of traffic and user flow to shops.

## GUI and Visual

Graphical interface and visual appearance is a vital part of our application. As we are developing a functional product we have to be able to present it to our potential users.

**Logo**

It’s the part of any application that catches the attention and attracts users while browsing on Google Play Store. The logo is the part that is used to visually present what we (our app) does and provides as well as to create a memorable and recognizable reference in people minds.

Our current logo is clear and presents our idea of saving money while shopping. White (#FFFFFF) color is a symbol of Peace, Purity and Simplicity while Black (#000000) is representing Classic and Serious manners[[6]](#footnote-6).



Our color choice will ensure a visibility at any background set by user. The shopping cart reminds of shopping – our main idea is to interact with the user and push him/her to use our application for shopping and checking prices.

**GUI**

Graphical user interface is yet another important part of any mobile application. Every button, every writing field and other touchable objects have to be placed carefully so they can fit user’ hands. Also the GUI has to be easy to follow and intuitive – following a logical flow when using the application. Building a really easy to use and fast user interface is absolutely critical to a successful mobile application. It will not only keep our users happy but will lead to sales - people will be coming from other apps that don't have a very good user experience. As Market analysis showed during the testing competitors have much more difficult menu than we intended to create. Further analysis in survey also revealed that customer during the testing of PriceCart and main competitors is more likely choosing PriceCart than the competitors’ application. The following section will describe the flow of PriceCart design development during its first steps, after market analysis and the final product.

**Stage 1 – Drawing**

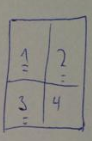
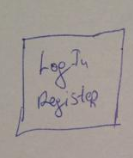
Our early development of GUI, drwen on paper at the start of the project.

Search

Login / Register

Results

Directions



**Stage 2 – Mokups**

After clearing the idea of what has to be included in the App, mokups have been made in order to see how our App will look like.



Directions

Results

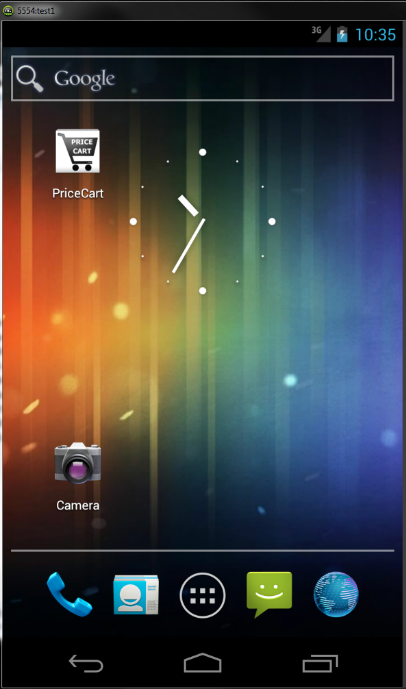
Search

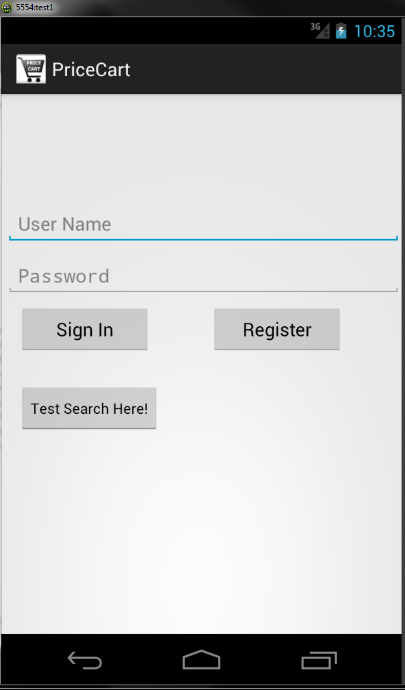
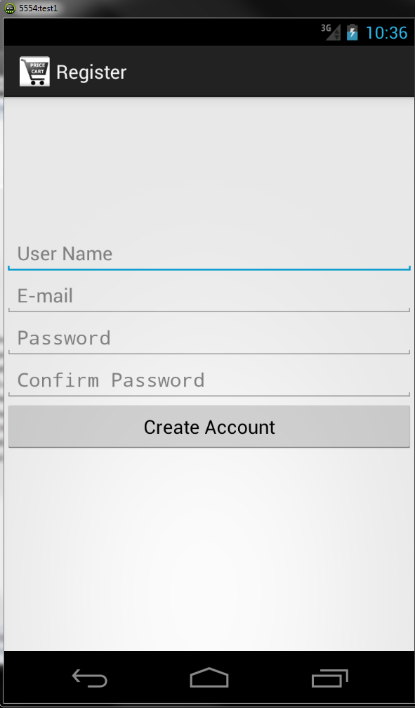
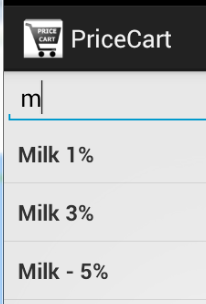
Options

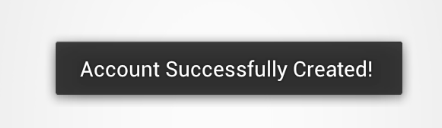
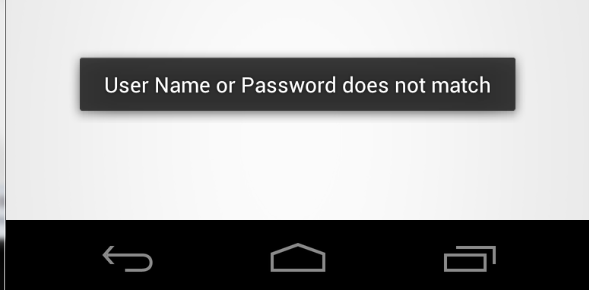
Login /Register



**Stage 3 – Final GUI**

Combining drawings and mokup we came with the following final GUI.



Notification messages according to users inputs and interaction with the App.

Notification when User creates an account successfully.

Notification when User tries to login with wrong credentials.

## Risk Analysis

Being aware what could harm the final product and how different aspects could interfere or change the goals is important. Analysing different risk aspects could and will improve both, project work and final application.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk Source** | **Additional information** | **How to Prevent** | **High** | **Medium** | **Low** |
| Physical threats | Issues posed to the IT infrastructure | Host our services and servers at a reliable company |  | x |  |
| Data threats | Issues posed to files or database | Have a backup of the DB | x |  |  |
| Backup of Database | External place to backup our Database | Backup on weekly or daily base to external source. Being able to restore at any time. |  | x |  |
| Errors by people | Users misbehaving | Clean and easy to use UI | x |  |  |
| Technical failure | Bugs and OS crash | Proper code and future updates | x |  |  |
| Infrastructure failure | Database or server crash | Backup | x |  |  |
| Payments | Credit card and payment fraud | Encrypted user data | x |  |  |
| Hacker threats | Attempt to ruin or overtake a website or server | Strong passwords; backup | x |  |  |
| Competitive Ideas | Companies with same or similar idea to ours | Simplicity of using the app, clear UI |  | x |  |
| DDOS attacks | Denial-of-service, leading to slow performance and accessibility problems for users | If we have our own server, we will have Load balancer |  |  | x |
| Lack of users | App becomes useless and not interested, less profitable | Advertisements at Metro, Bus stops, S-tog stations; Social Media Advertising (Facebook, Google +) |  | x |  |
| Keeping the set deadlines | Following the Project Plan | Following the Guntt Chart | x |  |  |
| Database mismatch | Mismatch price information from DB and actual price of the product in the shop | Updates based on daily or weekly base |  | x |  |
| No Interest | Users might not be interested to use our app | Unique features – searching, ordering, directions | x |  |  |
| OS platform | Danish customers are more into Apple mobile devices – our app is developed for Android OS | Develop the same app for Apple | x |  |  |
| Legal Issues | Legal issues might rise with different Danish shops | Following the Danish Lows | x |  |  |
| Personal data leakage | Protecting sensitive user data – address, CPR, names, etc. | Encryption of the data | x |  |  |
| Java programming | Our team could have difficulties using this language for development | Following tutorials and examples from various sources |  | x |  |

Having risk analysis written in a visible way helps to determine the aspects which can harm the work of the project and the final product – also being aware what and where could fail is a great advantage. Preventing failures and issues on an early development stage saves time in the future work on the project and product. The growing usage of mobile applications in business requires controlling the associated risks. Fortunately, online based risks are similar to those encountered in other business environments - managing information systems risks. As an online business we have to protect customer’s data and details from outside access.

## Context Diagram

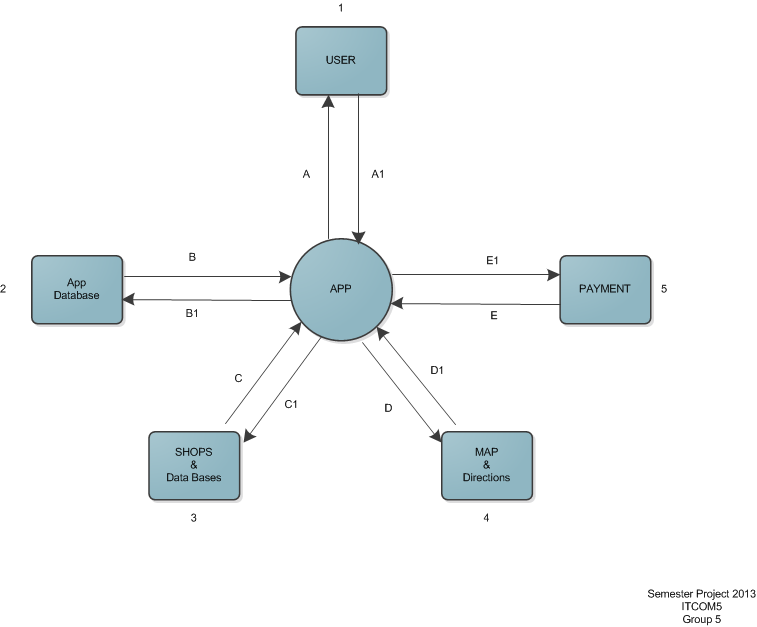
Diagram 1 – According to the App[[7]](#footnote-7)

Diagram 1 – Descrtiption of what the terminator does

|  |  |
| --- | --- |
| **Terminator ID (T)** | **Descrtiption** |
| 1 | User. He/she Interacts with App: enters search inputs and receives answer as a map and table of products. |
| 2 | App database. It offers the following functionallities: User login/ Registration, Search History, Products price, Shops locations. |
| 3 | External Shops and their Databases. Offered functionallities: Data exchange, Product prices updates. |
| 4 | Map. It offeres the following functionallities based on user (T1) input: Digital Map, directions to results. |
| 5 | Payment. Offered functionallities: Payment options via GSM provider, Bank. |

Diagram 1 – Flow of the exchanged data/content

|  |  |
| --- | --- |
| **Flow ID** | **Descrtiption** |
| A, A1 | Data entered by user (T1) sent for processing to the App and returned as ready result. |
| B, B1 | Data exchange between App and its remote Data base - User login/ Registration, Search History, Products price, Shops locations. |
| C, C1 | Data exchange between App’s database and external Shops’ databases - update producs’ priceses. |
| D, D1 | Data containing coordinates and directions presented via T4. |
| E, E1 | Payment process. Data generated from App, sent to payment gateway: producs price, address, credit card info, username. Transaction enitiated by user (T1). |

Diagram 2 – According to the App Database

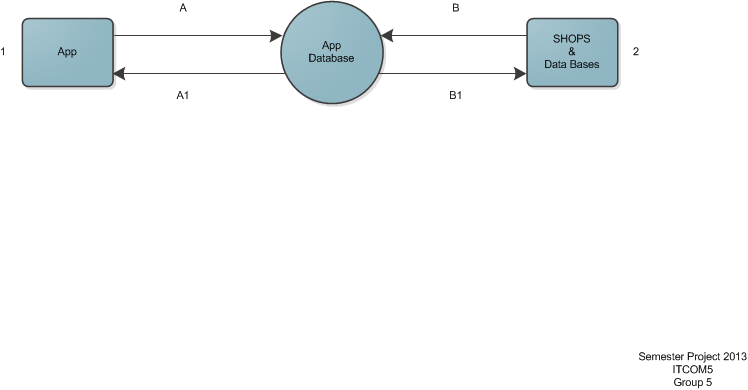


Diagram 2 – Descrtiption

|  |  |
| --- | --- |
| **Terminator ID** | **Description** |
| 1 | Application – operates with data, gathers input from user. |
| 2 | External shops’ databases – holds prices and products availability. |

Diagram 2 – Flow

|  |  |
| --- | --- |
| **Flow ID** | **Descrtiption** |
| A, A1 | Data sent and received to/from our App, data is stored on Database – names, emails, producs pricing, shops location, payments |
| B, B1 | External Shops and their Databases. Offered functionallities: Data exchange, Product prices updates. |

## Security

Being secure is important nowadays – our app will be dealing with sending and receiving data to and from Database. The connection between them has to be secured via https in order to avoid intruders to gain valuable information for our users – login, names, emails, addresses and etc. We have to ensure that only the important data is protected in order not to slow down entire application and database. We should encrypt only users’ passwords and CPR numbers (the prototype application is not gathering CPR data).

**HTTPS**

* **Https[[8]](#footnote-8)** between app and Database – **Why** do we need it?
* **Encrypt traffic** – in case where the network traffic is disturbed by third person, he/she will not be able to read what our users have sent to App database.
* **Confidentiality** – unauthorized access to user data will be set to minimum, preventing sensitive information from reaching the wrong people, while making sure that the right people can in fact get it.
* **Integrity** - protects the data from being modified while transmitted to and from App Database



Encryption

As we have our communication protected, we also have to protect the data which is stored on our database. Although the server which holds the database might be protected, we can’t leave the information on the database in plain text – it is vulnerable in case of hacking or stealing the entire database.

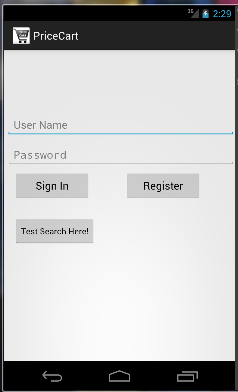
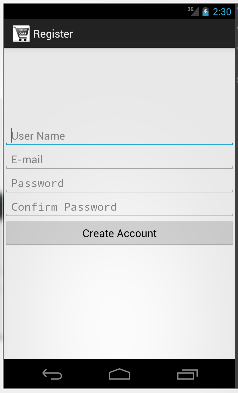
* Encrypting user data inside Database - Encryption is a technique of encoding data, so that only authorized users can understand it. Encryption alone, however, is not sufficient to secure our data. Protecting data in the database includes access control, data integrity, encryption, and auditing.

A possible solution for our needs could be to encrypt all user data with their own password. This will prevent us and other non-authorized units to decrypt the data and extract sensitive information like emails, names, bank details.

* Encrypt user data with password hash - we will hash passwords to prevent an attacker with read-only access from escalating to higher power levels. Password hashing will not make our database impervious to attacks. A drawback of password hashing is that since we will not store the passwords themselves (but only a piece of data which is sufficient to verify a password without being able to recover it), our team cannot send back their passwords to users who have forgotten them. Instead, we or our system must select a new random password for them, or let users choose a new password.

**Login function**

Our final functional prototype application has a basic security feature – Login and Authentication. We are using the Login to ensure that only registered users can access the application’s core feature – Searching for products. The data from new registered users is stored on a local database. During the login process the user data is checked and validated. The code and core functions are described at Code reflection *(see page 44)*



## Payment via mobile operator

In order to provide such feature there is a set of rules that each company or service has to fit in[[9]](#footnote-9). Danish rules and restrictions states that the service must comply with any rules set by the Telecommunication Industries Association in Denmark. Furthermore service must be in compliance with Fortumo terms of service, Danish legal acts and good morals. In order to provide such payment possibility user must be provided with:

* The final price of the product
* Notification with the receipt that money was successfully added to the phone bill
* Service provider legal name and Danish customer support contact information
* Broad information about the limitations of money spend under 18 years old
* Refund possibility of 2 weeks and a link to the refund page from Fortumo
* Adult, lotteries, gambling or donating related services are strictly forbidden

Service provider also has to ensure that the product or material being sold is appropriate to children. This also includes that the product can’t be pornographic nor contain sexual services. Service cannot contain references or descriptions of sexual aspects nor establishing sexual relationship or asking for the information that could provide possibility of doing that. Service cannot be related anyhow to the act of violence.

The amount charged for the product or service cannot be split into more than one charge and it has to be the same as it was stated in the advertisement amount, no hidden payments must be done. The receipt that must be received in the end of a successful payment must be in Danish.

## User Scenarios

Having analysed our market environment, and the application contextual design, the next step will be to look on how the app will be used by the users. The main goal for that will be to identify requirements, both functional and non-functional. The scenarios are created to present users from the different target groups, and 2 different cases in which the app could be used.

In our first scenario the actor will be a student, who will use our App to find cheap drinks available nearby.

1. Adam is 22 years old, he is a student of Copenhagen University. He was surfing in google play market for applications and found “PriceCart” application. Since the application had a high rate, he decided to download it.

When the application was successfully installed, he launched it. In the beginning he saw a short loading screen which after lead to a log in screen. Since Adam did not had an account in our database he had to create an account, by pressing button “Sign up” a new window with a registration form appeared. He had to fill up all required fields such as: “Name”, ”Last Name”, ”Address”, ”E-Mail”, ”Phone Number” and etc. after the form was filled up Adam pressed the “Create Account” button. By this action record in our database was created and a confirmation letter to specified email was sent. Adam received the letter and pressed on the link he received via mail. Link directed him back to the log in window where now he was able to log in with the information he filled up during the registration. After the successful log in Adam sees a menu with Options “Search”, “Promotions”, “Settings”, “Search History”, “Credits”, “Log out”, Adam pressed the search button what lead to a new window to the search engine. He tries to enter “Faxi Condi” and by starting to type it automatic fill in suggestion starts to show available possibilities and Adam press on the suggested product he was searching for. Search then begins to scan all solutions and within couple of seconds provides all the shops within 10km radius from Adams location (Distance is set to a default since Adam did not modified the settings manually). The information Adam sees is that the cheapest drink is 8km from him for a price of 12kr in Fotex which is working 8-20 today, however Adam noticed that alternative to it is 2km from his location for a price of 13kr in Netto which is working from 8-22 today. Adam chooses the closer shop and presses on the tab, what opens a new window with location where the shop is located and possibility to get the directions how to reach the shop. Adam presses “Get directions” button, this terminates the application and opens google maps. In the google maps Adam already sees the entered destination points and the directions how to get there.

1. Jonas is a 42 years old single person. He found “PriceCart” application under the section “Recommended for you”.

He launches the application and a loading screen with the logo of our brand appears, after loading is finished Jonas sees the Log in window, he successfully logs in to his previously created account. He sees that today is a very bad weather and he is not into the mood to travel a lot today, so he goes to the “Settings” and changes the distance from 15km to 5km and goes back to the main menu. Before beginning search he decides to check the promotions and so he opens “Promotions” there Jonas can see all best discounts within 5km range and what he could possibly order. However Jonas does not see anything he would be interested buying. He goes back to the main menu and presses “Search” he then starts to enter the products he wants to order. He begins to type Pizza and a possible solutions of available products starts to appear he chooses the pizza he prefers most and after he is finished with that he is placing “;” to be able to add more products he then begins to type Heineken Beer and he also chooses the suggested solutions, since he entered all the products he want to order Jonas presses the search button and a request is being executed. After a few seconds Jonas sees that new window opened with the product prices. He sees that the cheapest product combination is in Aldi for a price of 52kr but the distance is 5km away. Jonas is very hungry so he is choosing the closer alternative for 54kr from Fakta which is 1km distance from his current location. Jonas presses on the chosen product combination and he presses the possibility “Order” a pop up window with the payment possibilities appears, he decides to pay via mobile operator, the payment is being executed and a receipt for his products being received. Within 15minutes a delivery arrives where the QR code from Jonas receipt is being scanned to prove that exactly these products were ordered. After that Jonas is able to cook his pizza and enjoy his beer.

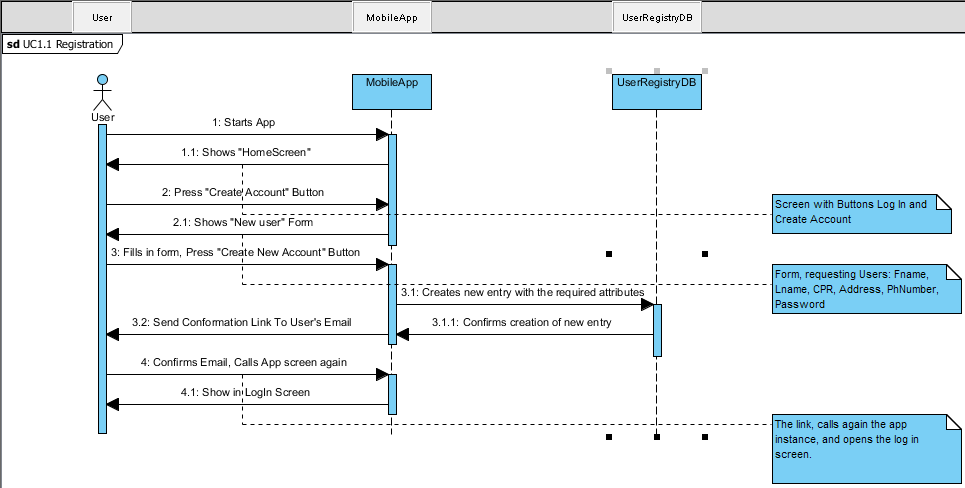
Having these two scenarios, we managed to create a basic overview of the functionalities that our app should have, in order to work. In the figure bellow the different use cases can be seen.

## Use Cases

From the scenarios we managed to derived different use cases of our app. In this section, the main focus will be on the use cases followed by flow diagrams. The main goal for that is to analyse the flow of the data in the different methods, also identify the classes, and the methods that have to implemented, to manage the user input and interactions.

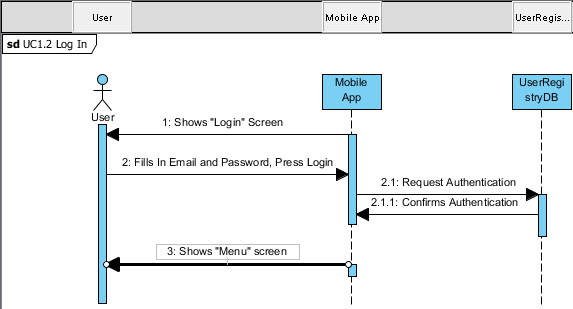
**Use case - Registration**

|  |
| --- |
| **Scope:** Google play market application  **Level:** Costumer goal  **Primary Actor:** Costumer  **Stakeholders and Interests:**  Costumer wants to create an account in order to get access to downloaded application.  **Preconditions:**  -Smartphone  -Internet connection on the smartphone  -Application connection to database  **Success Guarantee or Post condition:**  -The costumer will be able via the application to create an account and application must store account information into database.  **Main success scenario:**   1. The costumer goes to the app store on his smartphone and downloads the application. 2. The costumer needs to fill in the registration form in order for the application to work. 3. The costumer press register button and sends request to create an account. 4. The customer receives confirmation letter to his mail.   NOTE: By using the registration form the application makes sure that the customer is authenticated.  After creating the account user is allowed to use the application.  **Extensions:**   1. Database sends a confirmation hyperlink to activate the account to customer mail.   The costumer will also get a notification if there is any problems with the connection while finalizing the registration, he can therefore always know if there is any problem.  **Special Requirements:**  -User needs to buy the product. |

****

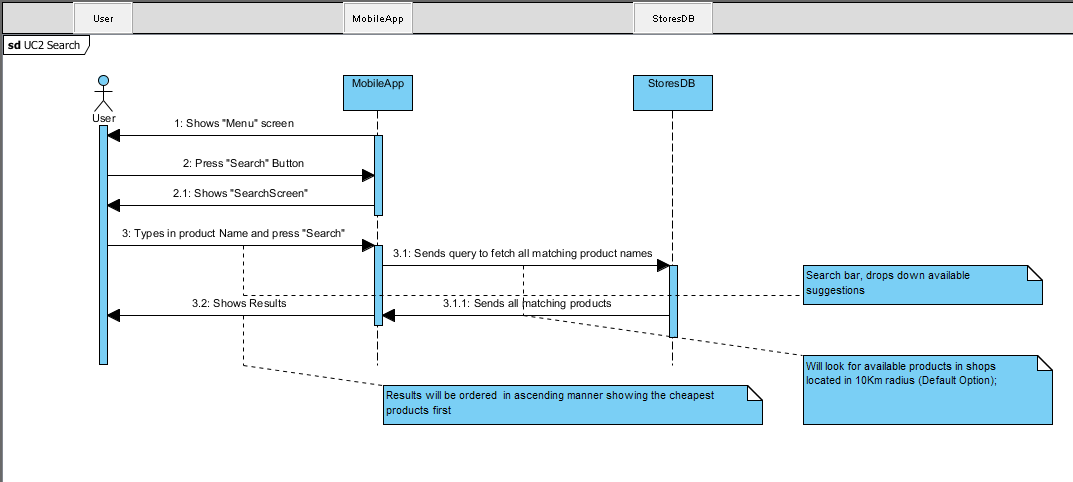
**Use case – Log in**

|  |
| --- |
| **Scope:** Google play market application  **Level:** Costumer goal  **Primary Actor:** Costumer  **Stakeholders and Interests:**  Costumer wants to log in to his/hers account in order to get access to use application.  **Preconditions:**  -Smartphone  -Internet connection on the smartphone  -Application connection to database  **Success Guarantee or Post condition:**  -The costumer will be able via the application to log in to account and application must request account information from database.  **Main success scenario:**   1. The costumer goes to the app on his smartphone and opens the application. 2. The costumer needs to fill in the log in information in order to use the application. 3. The costumer press log in button and sends request to access account. 4. Database confirms the authentication.   Application displays menu screen.  **Extensions:**  Requested authentication failed, application displays user that the information was entered incorrectly and it has to be reentered again.  **Special Requirements:**  -User must to have valid account. |



**Use case - Search**

|  |
| --- |
| **Scope:** Google play market application  **Level:** Costumer goal  **Primary Actor:** Costumer  **Stakeholders and Interests:**  Costumer wants to search for a specific product.  **Preconditions:**  -Smartphone  -Internet connection on the smartphone  -Application connection to database  **Success Guarantee or Post condition:**  -The costumer will be able via the application to search in Database system for specified product.  **Main success scenario:**   1. The costumer logs in to the app. 2. The costumer presses search button. 3. The application returns search engine window. 4. The customer provides product name in search engine. 5. Application accepts request and sends query to shop database. 6. Database returns all matching results to application, application displays to the user.   **Extensions:**   1. Internet connection was terminated search engine fails to send query pop up window appears asking to turn on internet connection. Specified product did not matched any results, application receives 0 results, user gets message no results found. 2. **Special Requirements:**   -User must have a valid account.  -Search request must match database information. |

****

## Requirements Specification

After coming up with user scenarios, extracting different use cases and analysing them, the next logical thing is to identify the requirements[[10]](#footnote-10) which need to be fulfilled in order the app to execute the different users’ actions. Note that the requirements are not only identified, but also prioritized using the MoSCoW model.

**Function requirements**

The functional requirements must define actions/outputs which the system must perform in coherence with the input given by the actor related to the application. Using the use cases in the following section, the extracted requirements are presented.

Personal Information:

* The user should be able to Log in to his profile, or register a new one if he does not have an account.
* The user should be able to customize his profile.

Requirements: The system should be able to store, retrieve and modify the personal data in/from remote database in a secured manner. The App should be able to prompt the user when log in information is wrong, and provide him with password recovery function in case the user loses his key.

Using the app:

|  |  |  |
| --- | --- | --- |
| FR1. | User should be able to search for shops/products. | Must |
| FR2. | The user should be able to make multi-product search | Could |
| FR3. | The user should be able to customize the search.(proximity range) | Could |
| FR4. | The user should be able to customize the search results.(items per page) | Should |
| FR5. | The user should be able to see his last 10 searches. | Could |
| FR6. | The user should be able to see the shops on Google maps. | Should |
| FR7. | The user should be able to receive directions to the desired shop | Should |
| FR8. | The user should be able to choose whether he wants his goods delivered or picked up. | Wouldn’t |
| FR9. | The user should be able to choose payment type | Should |

Requirements:

|  |  |  |
| --- | --- | --- |
| FR1. | The application should be able to search for shops in Denmark, using connection to: - Database where Shop information (e.g. Name, Location(Googlemaps.api),address, working time,) is kept in a secured manner.  - Database with information about products (Name, Price in shops, Availability in shops) | Must |
| FR2. | The system (The App) should be able to provide the user with auto-fill suggestions (using cookies). | Should |
| FR3. | The App should have a settings section, where the user can set his search preferences | Should |
| FR4. | The system must be able to execute multi-object search, it must be able to recognize when different strings are separated with “;”. | Could |
| FR5. | The App should have a customizable results layout, in order to allow the user to freely arrange his result by required characteristic like price, shop distance etc. | Could |
| FR6. | The system must be able to store, access and modify, a history of the 10 last searches of the user, on local storage. | Should |
| FR7. | The system should be able to communicate with Google maps and Google navigation. | Should |
| FR8. | The system should provide the user with several options to execute the transaction: via mobile payment, Credit card, or with cash on delivery/pick up. | Should |
| FR9. | The app should be able to provide the user with information about the app, it’s purpose of usage and information about it in “Credits” section | Must |

**Non-Functional requirements**

People who will be willing to use this product are willing to have more than just a running application. They would rely on our search engine which is supposed to provide the cheapest products in there that are possible by that time. People as well are interested in a good application performance, reliability and many other factors. A number of characteristics are defined as non-functional requirements for the system.

**Usability –** is the ease of use and learnability of a human-made resource. Resource and be a software application, website, book, tool or anything that human interacts with. Usability study may be conducted as a secondary job function by designer, technicians. Usability is including methods of weighting the usability of the resource. Analysing primary nation of usability helps to make resource more efficient to use, what helps to reduce time in accomplishing tasks. Easier to learn, purpose can be learnt by experimenting with the resource. More satisfying to use.

|  |  |  |
| --- | --- | --- |
| UNFR1. | Language must be provided in English | Must |
| UNFR2. | Multilanguage could be a possible advantage | Should |
| UNFR3. | Application providers must provide fast problem solving | Must |
| UNFR4. | Support must handle issues within 10-2h frame | Must |
| UNFR5. | Video tutorial of how properly use application could be provided | Could |

**Price** – In ordinary life price is the quantity of payment given by one party to another for the resources or services. Price sometimes is referred to a payment that is requested by a seller for its resources or services. Often this price is called asking price or selling price, when the actual payment can be named as transaction price or traded price.

|  |  |  |
| --- | --- | --- |
| PNFR1. | Application must enter google market | Must |
| PNFR2. | Optional price 15kr for a wide range of people using smartphones | Could |
| PNFR3. | To avoid customer loss and attract bigger crow application must be provided for free till 10000 views | Must |
| PNFR4. | Future statistics must provide the answer if product will be payable | Should |

**Availability** – Is a ratio of a total time the consumer can use resource within the interval of a given time. System availability is measured by reliability factor, meaning that if system reliability increases so does the availability. System availability may also increase by focusing strategy on increasing testability and maintainability, but not on reliability. Availability plan must provide a clear strategy for availability control.

|  |  |  |
| --- | --- | --- |
| ANFR1. | Product must be reachable all the time whenever user has access to the internet | Must |
| ANFR2. | System must be installed and phone must be on | Must |

**Performance** – characterized by the amount of successful work accomplished compared by the time and resources used. Good performance can be referred to:

* Short response time for a given task
* High throughput – rate of successful message delivery
* High availability of computing system or application
* Highly compact data compression and decompression
* Low data transmission time

The performance of any computerized system can be evaluated in non-technical terms. Using researches and compared relative to other systems or even the same system before and after the maintenance. Using technical performance metrics there are many software to test the possibilities of the hardware or other software.

|  |  |  |
| --- | --- | --- |
| PNFR1. | Product must achieve a short response time for faster performance using search engine and better reviews against competitors | Must |
| PNFR2. | High throughput – important, since application is taken into consideration to provide e-shop | Must |
| PNFR3. | Application performance must be simple to fit most people using smartphones | Should |

**Dependency on other parties** **–** Factor that is important when trying to combine 3rd parties in order to achieve best system performance. Resource can be dependent on aspects such as competitors, banks, government, banks, mobile operators and etc. In this case the system that is being developed is highly dependent on mobile operators, banks and shops.

|  |  |  |
| --- | --- | --- |
| DNFR1. | Application must have access to most grocery shop chain databases in Denmark | Should |
| DNFR2. | To achieve e-shopping payment via mobile operator or bank must be provided | Could |
| DNFR3. | All parties must agree to cooperate within each other to have 100% developed idea working | Could |

**Reliability**[[11]](#footnote-11) **–** Constrains on the run-time behaviour of the system. Reliability is the ability of a system to perform its required functions under the stated condition for a specified period of time. Reliability can be referred to availability that the system is available for service when requested by end-users. Failure rate can be assigned to reliability since it describes how often the system fails to deliver the service as expected by the end-user. To add more reliability is one of the most difficult function parts to be analysed, it is very hard to tell how the final product will appear from the beginning. There are no guarantees for end customer nor developers that the primary idea will be fully developed or unchanged during development process.

|  |  |  |
| --- | --- | --- |
| RNFR1. | Improperly working application must not reach customer. | Must |
| RNFR2. | All bugs and errors must be sorted during development process or user involvement testing. | Must |
| RNFR3. | Application must detect the newest system updates and update them automatically via google play whenever internet access is available | Should |
| RNFR4. | After update application must continue from the last session | Could |
| RNFR5. | Backup before each update must be made | Wouldn’t |

**Compatibility** – Is a requirement that helps to analyse the system and provide it to as many customer as possible. Compatibility requirements gives a view of what treats or issues the developing application might face when it is released. It helps to understand with which platforms, OS versions can be used as minimum and maximum requirement.

|  |  |  |
| --- | --- | --- |
| CNFR1. | Minimum OS version requirement is 2.0.3 | Could |
| CNFR2. | Application must be available to newest OS updates | Must |

**Maintainability[[12]](#footnote-12) –** In engineering maintainability is when product can be maintained in order to:

* Isolate defects
* Correct defects
* Secure system from unexpected breakdowns
* Optimize safety, reliability, efficiency
* Maintain to meet new agreements

In some cases maintainability involves constant improvements meaning that it is being learn from the past in order to improve the ability to maintain system, or improve reliability. Maintenance can be either in software or in hardware.

|  |  |  |
| --- | --- | --- |
| MNFR1. | If bug was found during application release it must be isolated, download possibility stopped till issue is fixed and update made for users using application (refer to UNFR3.) | Must |
| MNFR2. | Database can be edited any time if it is needed. | Should |
| MNFR3. | Application updates must be provided via google play store | Could |

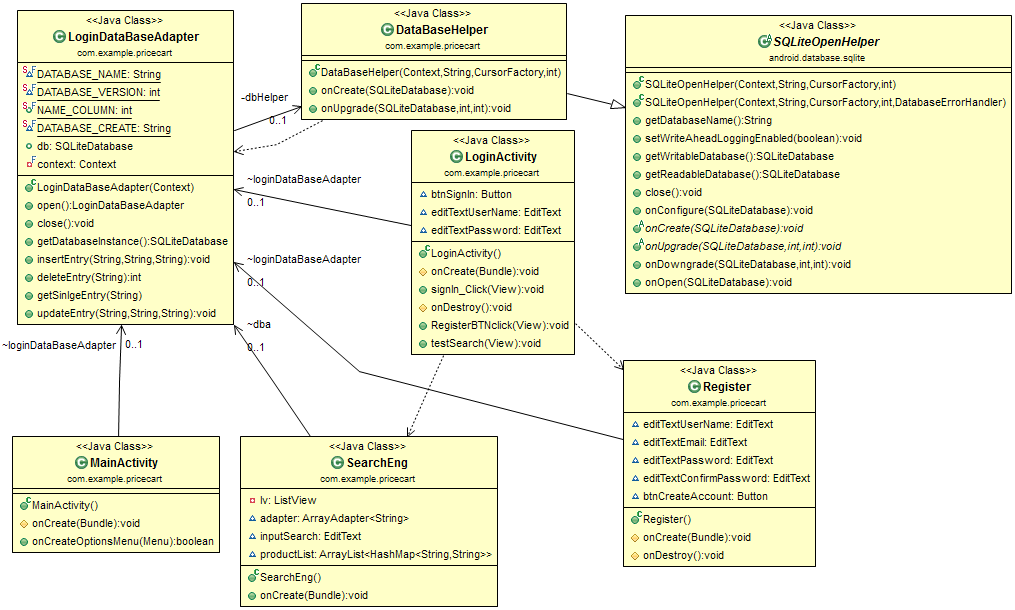
# Design and Development

## Development

After the requirement specification, user scenarios, use cases and graphical user interface were defined, it was easier to develop the application. Requirement specification helped to understand how the skeleton should be structured in order to make it as simple as possible. Use cases helped to understand how they should work for the customer and graphical interface was the final part to sum things up. In this documentation part we will briefly describe the application development process including class diagrams, concept coding, testing parts and improvements by survey help. Class diagrams will provide a clear vision how the application is working with itself. It is very crucial part in development process since it helps to understand even better how the application should work. In coding part we will try to explain how the application was began and some of the features how they are actually working. In the further sequence you will find out how the concept was shaped and what was done to get such product. Furthermore user testing will be explained to point out the questions why the application was created in such way. In the final part you will be able to see what implementations was done by a cooperation with testers and what was pushed to the future development.



## Class Diagram

Based on the analysis and information gathered from the context diagram *(see page 29)* and the use cases *(see page 34)*, our team managed to design the following glass diagram which will be used to develop first the prototype, and later the application.**Classes***MainActivity*Every android app, starts with main activity. Our main class is associated with the LoginDataBaseAdapter.class

*LoginDataBaseAdapter*  
One of the core requirements for our app is to store, get and modify data in a database *(see page 38)*, a class to deal with these operations had to be implemented.  
This class is mainly responsible for operations with the database. It implements methods to get readable or writable database or in the case where there isn’t such, it will create new one. In order to create and update new tables it implements the DataBaseHelper Class.

*DataBaseHelper*  
This is a class, that is associated with the build in SQLiteOpenHelper  
  
*LoginActivity*  
Since in the requirements sections *(see page 38)* is stated the application must allow the user to Log in in his profile. This class simply calls the Interface (login\_activitu.xml) and implements methods that will allow the user authentication. It creates GUI object: EditText field required for user’s ID and key. The core method it implements, is the sign-in function, which will try to find match in the user registry DB, and return a message weather the authentication was successful or not.

*Register*  
This class is also designed to cover the main requirement. It is a main form for registering a new user, and adding it to the user Registry.Calls GUI. Creates EditText fields and a button, required to register a new user (Add new registry in the database).

*Search*  
Searching for items stored in Database, is also one of the core functionalities(FR1,FR2). This class’s main purpose is to deal with that functionality.

In this section we have designed the classes needed to cover the core functionalities of our app. These functionalities were discovered and discussed in the previous sections. In the next section a reflection of the code itself, and explanation of how the methods work to fulfil the requirements, can be found.

## Code reflection

In this section we will introduce parts of our code from the time of the development of the Android application. We will present screenshots of core parts of our code.

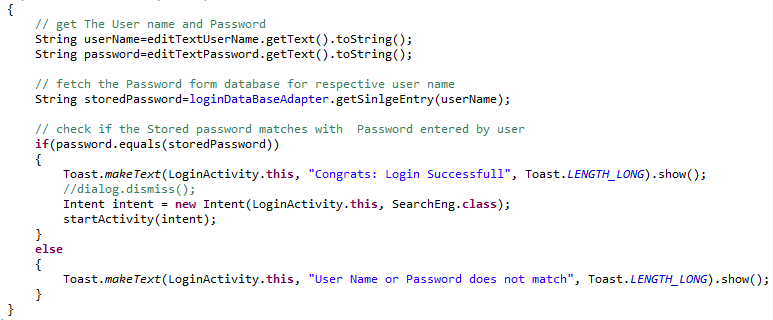
**Registration**

Our application provides user registration in order to ensure that only authenticated uses will have the access to the Searching functions. All the entered data is stored in our local database for later usage and validation at Login.

When button is pressed, OnClick event is triggered and the data from text fields (editTextUserName, editTextEmail, editTextPassword and editTextConfitmPassword) is taken and processed – the code checks if all fields are filled and if the password matches, if the checks are passed, the data is being processed by the DatabaseAdapter and stored on the local database. User is notified of his action – with 3 different notifications, if he missed to fill in a filed, if password fields does not match and when the creation of the account was successful.

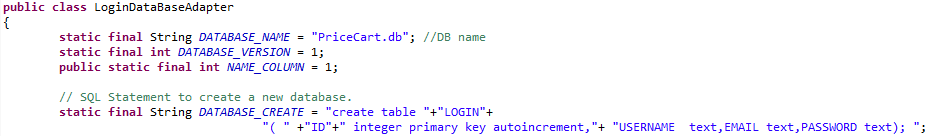
Closing the current view and sending user to the main activity – Login. Finish method is used in order to close the Register activity and “remove” it from device’s memory – the activity can’t be called again unless Register Button is not clicked again, this prevents double information to be processed and entered into database.

**Login**

As soon as the Registration is completed the user is sent to the Login screen. The login process checks if the user and password exist in the database and compares them. If the credentials are valid, user is authenticated and automatically sent to Search screen.

The code is reading the input from both text fields, username and password, and is fetching the data from the database, if the data is valid, user is granted access and notified with successful message and led to the Search screen (SearchEng.class). In case the login data does not match with any of the data at database, the user receives a notification that the “Username or password does not match”.

**Login Database**

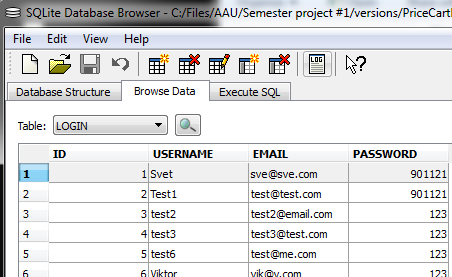
Database is a vital part of our application, it stores all users’ data and it’s used to validate and authenticate (Login / Register) our users upon the process of logging in. The current version of the code is soring the database locally on the device.

The above piece of code is reflecting the creating of our login database, it sets the name of the database, “PriceCart.db” and creates a table called “Login” and all the needed rows – ID, Username, Email and Password.



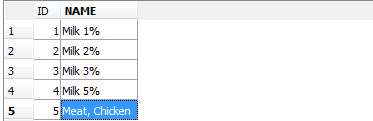
After the database is created on the device, it’s stored among rest of the Android application. Database is located at package com.example.pricecart:

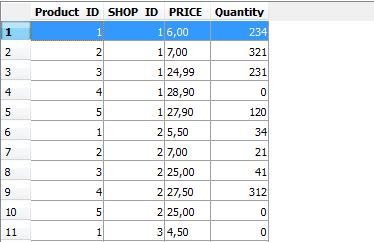
Data inside the database, SQLite Browser is needed in order to open it. Here we can see all the registered users.

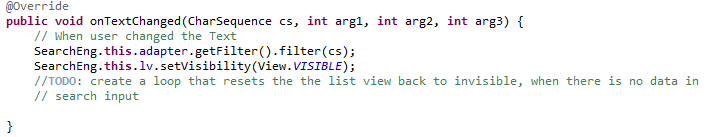


**Search function**

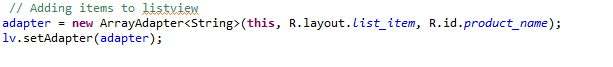
As discussed before, the Search function is also one of the vital parts of this application. Based on the extracted requirements *(see page 38).* There it is stated that our application should be able to search for products in a remote database (shops database), but since we weren’t able to use their database records *(see limitations page 7),* a locally stored database was designed.



One of the featured requirements was the application should provide auto-fill in suggestions. To fulfil that, the EditText field implements onTextChanged method, which filters each character and displays the products that match the string.



The adapter used is a ArrayAdapter. It simply uses getProducts() method form DataBaseLoginAdapter.class, which sends query to retrieve all records from the Product Table, and arrange then in an Array.



Then the items from the array are set up in a listView, which is made visible when the user starts typing.

**Results function**

The items on the list view implement OnItemClickListener, which triggers another query to the locally stored database, getting all the matching available products and arranging them by price (ascending by default).

These results are put in an ArrayAdapter which is used to put them as items in list view.

## Testing

Testing our application is a vital part of the development process, it helped us to find errors and fix them during the development process, before releasing our prototype to public. We have tested our Requirements and application performance. The results will give us hints what has to be fixed in order to have a functional application.

**Test Requirements**

According to the pre-set functional requirements the application should be able to fulfill them. Test cases are extracted from Requirements *(see page 38).*

Testing rounds – testing rounds were taken in different stages during the development of PriceCart.

* Round 1 – testing during early development
* Round 2 – testing during middle stage development
* Round 3 – testing the Prototype with core functionality

http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png - Success

http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png - Fail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test** | **Test round 1** | **Test round 2** | **Test round 3** |
| 1 | User is able to run application | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 2 | User is able to Register | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 3 | User is able to Login | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 4 | User is able to perform ‘Search’ | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 5 | Store user data on local database | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 6 | Determine location | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 7 | Support of English language | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 8 | App Published on Google Store | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png |
| 9 | App is running on 4.0.3 Android | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png | http://www.clker.com/cliparts/I/b/r/1/6/n/simple-green-check-button-hi.png |
| 10 | Automatic Update of the App | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png |
| 11 | User should choose payment | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png |
| 12 | Directions to the desired shop | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png | http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/White_X_in_red_background.svg/450px-White_X_in_red_background.svg.png |

**JUnit Test**

With the JUnit test, the development team, aimed to test three aspects. These three test are based on the functional requirement we managed to extract form the use case. Since we are planning to test the interface and the user input, it was decided to build a test case on Junit.

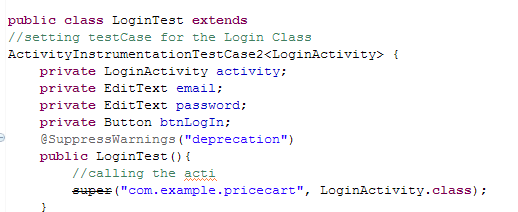
The 1st test will have the aim to test if the activity is loaded successfully. To do that it will call the LoginActivity.class, and initialize the activity.

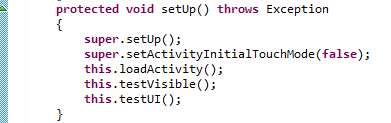
The 2nd test will focus on testing whether the GUI is visible, since user must see it and be able to interact with it

For the 3rd test, we chose to test the User input, since our application is required to recognize it. To be able to test it, first the test has to meet some preconditions. In our case these preconditions are that the EditText fields must not be empty.

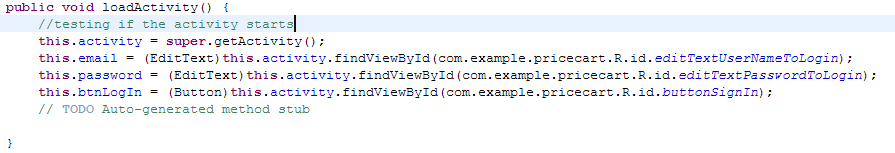
Having finished with the design of the test project, the next logical step was to writhe the automation process to test the unit we wanted.

To test our unit (LoginActivity.class), we have created a new JUnit Android project.

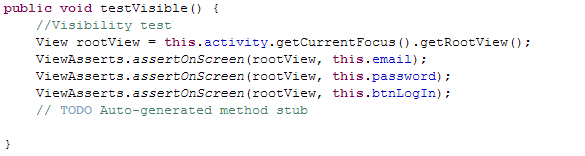
  
In the following lines of code, the declaration of our test case can be seen. It extends the ActivityInstrumentationTestCase2. The reason for extending this class is that it provides convenient environment for tasting units, and its functions. In the next lines the objects required for the testing if the unit are declared.



In this lines the SetUp of the test can be seen. The super.setActivityInitialTouchesMode(False) method, disables the initial touch of the LoginActivity. After that it will call our first test which will determine whether the activity is loaded.

Test1: Loading the activity

In order to test that, we simply assign values to the variables of our test project. The values assigned are taken from the unit we wanted to test (LoginActivity.class). As seen they are referred to our projects package. If all the elements are loaded successfully, the test passes which means that the activity is loaded

Test2: Visibility

The aim of this test is to determine if the GUI objects are visible. To do so, it creates a rootView instance that will be used to check if every element is present on the screen, using assert statements. If the test passes it means that all of the objects are present in the screen.

Test3 User Input ()

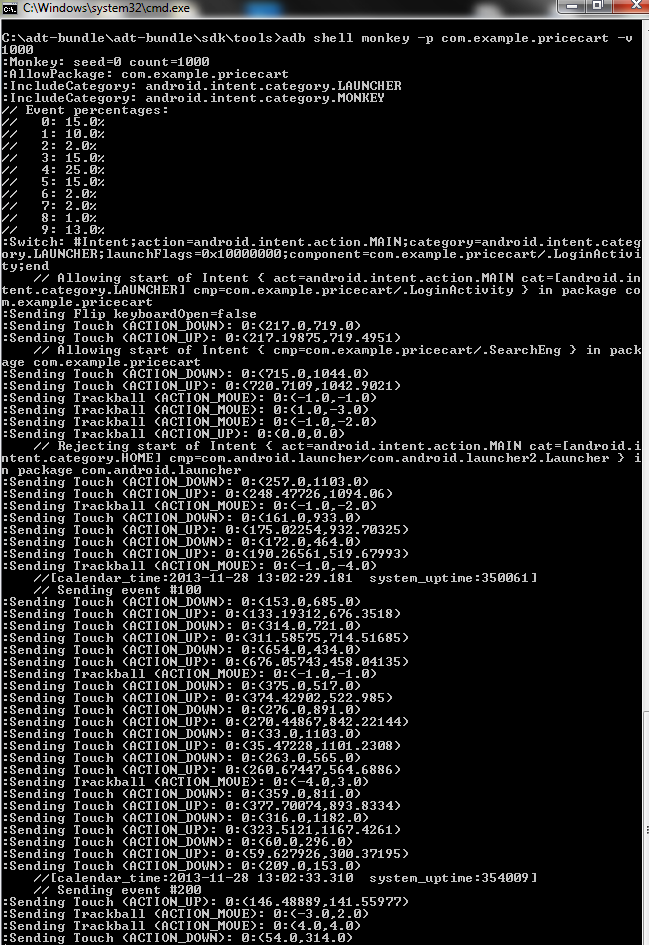
To test the user input we first create a method that simulates keystrokes using the sendKeys() build-in method. The sendKeys will sends character stream to the EditText field using two strings, 1 for user name and 1 for password

After that it will use assert statements to validate the user input:

If the text in the fields matches, the test will pass.

**Monkey Test**

The Monkey is a command-line tool that we can run on any emulator instance or on a device. It sends a pseudo-random stream of user events into the system, which acts as a stress test on the application software we are developing.



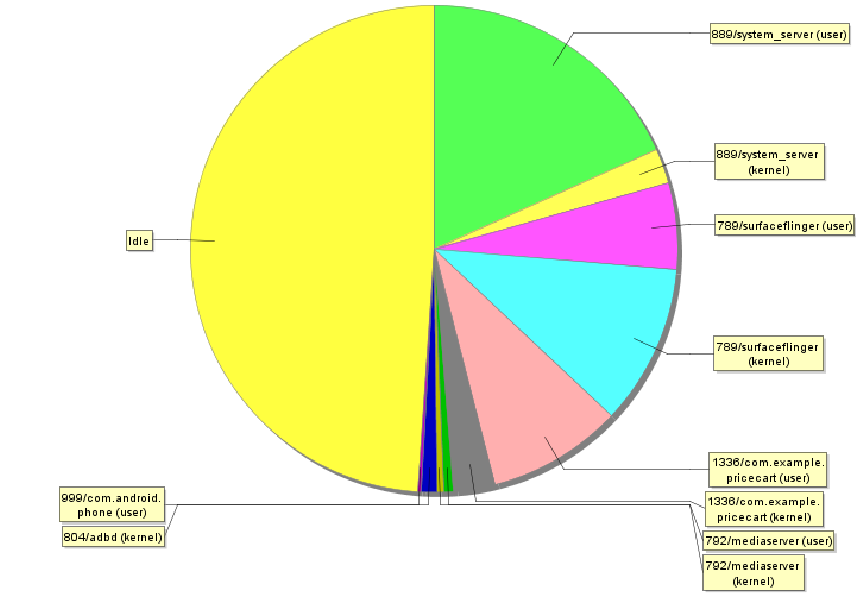
“adb shell monkey -p com.example.pricecart -v 1000” executes 1000 events on PriceCart application.

The application was able to handle the Eclipse Monkey testing events without crashing or showing unstable behavior.

**Usage**

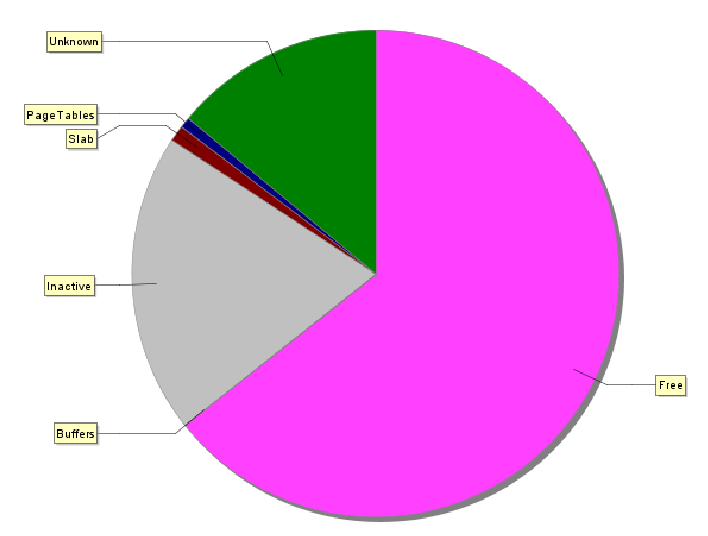
In order to determine what the usage is and how the mobile device is performing while running our application we used DDMS plugin from Eclipse. Every mobile device has limited CPU and RAM which could be used by apps.

**CPU**



CPU power need by our app, while the app is running its not spending 100% CPU. Its using ~ 20% of the mobile proccesing power.

🡨 PriceCart Usage

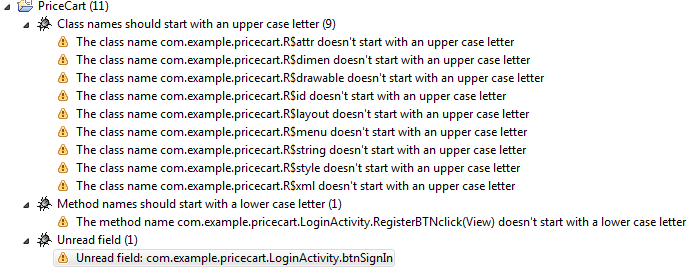


**RAM**

Ram testing shows that the device is running normally. Pink color presents the free memory.

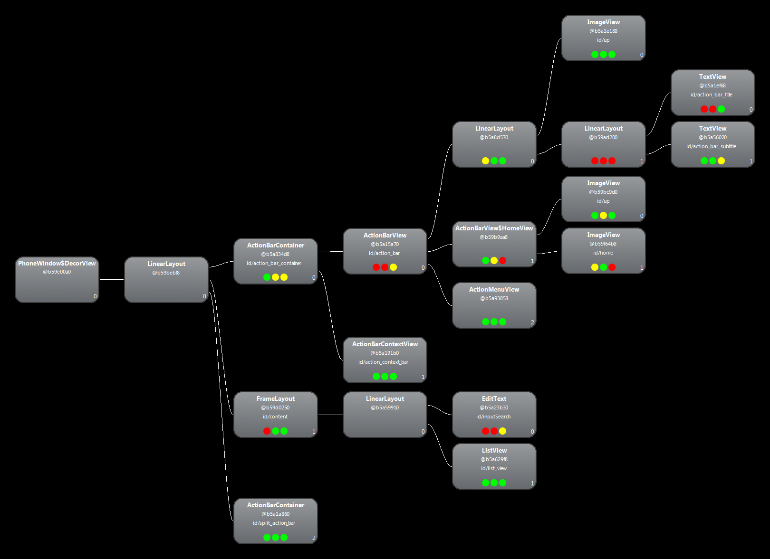
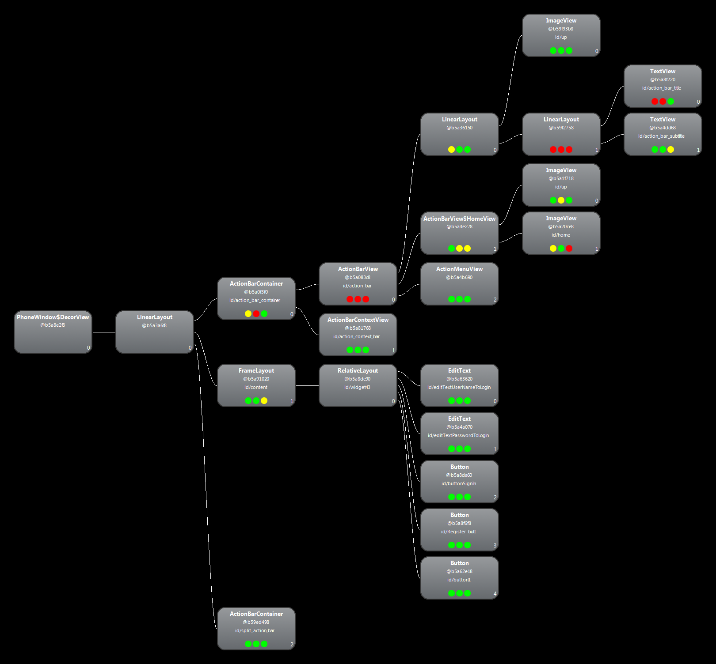
Having a low resourceful application is a benefit for us as developers – our application does not leak in memory and it’s not overloading the mobile device. Usage test plus Bugs Testing are showing that our application is properly developed. For the end user, having an application which is not draining the battery because of extreme RAM and CPU usage, is benefit as well – end user will not give negative feedback and delete the app.

**Bugs**

In order to find any bugs, memory leaking and other problems. We have used the plugin FindBugs. The errors it found are not major bugs and easy to fix, those errors will not bring the application to non-responsive mode and make it crash.

## Hierarchy Viewer

Testing the UI for overloading with too many controls. The Hierarchy Viewer is a powerful tool for visualizing and inspecting the components that make up my Android app user interfaces. It can be used to help streamline layout design, debug layout problems, and determine where my UI performance bottlenecks are.

Activity Login Activity Search

The overall performance of the Login screen is good, the problem (red dots) comes from ActionBar Widget and it’s causing a slight delay – not noticeable for users.

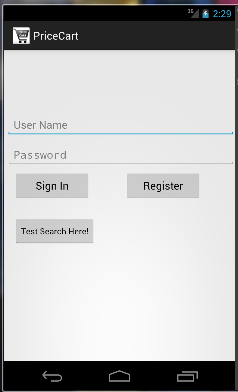
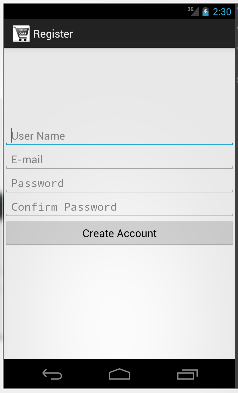
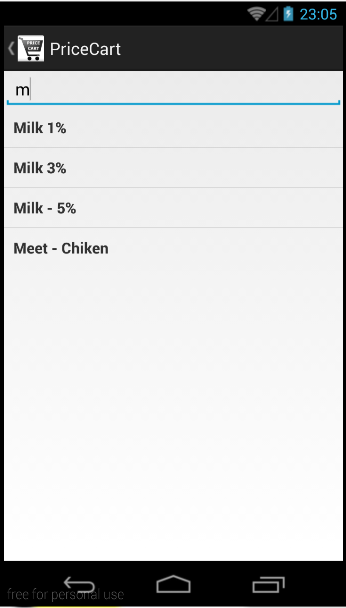
The overall performance of the Search screen is good, the problem (red dots) comes from ActionBar Widget and it’s causing a slight delay – not noticeable for users.

A solution for minimizing the problems could be editing the problematic views or replacing them entirely.

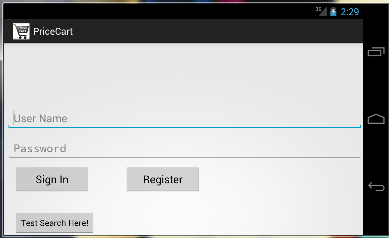
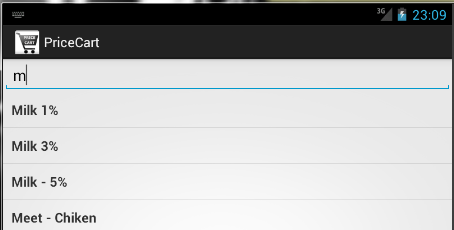
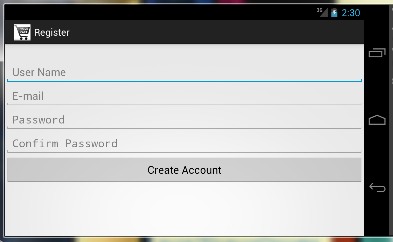
**Visual Testing**

Testing the visual appearance and performance is important, after the Monkey test, we have to test the Portrait and Landscape views if they present the screens as planned.

**Portrait View**

*No issues were detected.*

**Landscape View**

*No issues were detected.*

The visual testing in both, landscape and portrait mode has gave us a proof that our application is able to handle users’ actions and covers a regular surfing through the main core functionalities.

The overall tests consisted of testing our requirements, Monkey, Usage, Bugs, Hierarchy and Visual test have concluded that our application can resists on users’ actions and is able to work without issues from the starting point until the end point. As we continue developing PriceCart we will test it after the implementation of every new feature in order to keep the performance and stability.

# Conclusion

The project which is consisted of documentation and technical part was fully fulfilled according to the given time and pre-selected ideas and goals. The report presents how an idea gets developed, defined, analysed and developed until a final product could be shown. During the working process we, as a team, have learned how to work with Android development framework, obtained knowledge about working methods and models, and successfully combined our skills and knowledge for this project. Trying to solve the defined problem question helped us to gain new knowledge regarding Android development framework. By fulfilling the requirements we set at the start of the project we developed a functional prototype application. After we analysed the Danish Market we believe that our application is a strong competitor and could grain significant popularity and high usage. After implementing our ideas from Future development section, PriceCart might reveal its full opportunities and become a stable and profitable application.

Researches and future development will be the key role of the developers’ motivation in order to finish the project completely as it was defined in the beginning. Even though a lot of positive knowledge was gained the real target is the profit from the application which will also motivate team to finish the application as fast as possible.

## Future development

Future development helps to determine the product goals which has no limitations. With future goals team is determined to make an ideal and fully working product looking from our angle. Even though the real concept of the application is done it is still lacking many features. All these features and ideas how to enter the market will be mentioned in this section.

**Marketing**

In order to be successful in marketing it is necessary to analyse all possible ways to somehow be better than already existing company or product. In our case, after all the researches were done, we are trying to enter a quite strong competition. As a result to be catchy and in the same way competitive PriceCart developers came to a solution to provide two different versions of the application.

**Free**

Free version will include all the crucial parts of the application. Application will have log in function which is necessary all the time to have some kind of security level. Further it will have a core function which is search and a result display. Customer will be limited only to view the products that are cheaper in one or other place. To make at least a minimal profit free version will also be provided with ads which will give some sort of profits from the free user.

**Payable**

This version will be fully functional. Functionality includes a log in, main menu, search function, result display, navigation to the store via google maps and possible payment types in case customer will be interested to order product home. Main menu will have possibility to edit settings that will allow to filter credentials. Furthermore customer will be able to view most recent search history and do a multiple product search over the search engine.

The next important step according to the survey analysis is to provide in the future product to the apple store. It is a very important aspect since a huge number of people are using apple products and this would lead to a bigger notice.

By dividing product into two parts it is a real gamble between success and a loss. It is very crucial to provide correct amount of information in order to hook the customer to upgrade to full version. If the customer will be satisfied with the product it will not hesitate to pay for the product what will be covered later on using the application. However making such step might also distract people from actually buying the full version since customer might be satisfied with the free version possibilities. Only the implementation of such ideas will show if the idea is actually profitable and legit, this will be found out in the future.

**Development**

Due to short period of time many useful features were moved to the future development. To make application as it was drawn in the beginning of the project it must import map section, menu panel and payment possibility. However the most important thing for the application in the nearest future is to move its data from local to external database.

**Integration of the map**

To fulfil the future marketing goals and the complete sketch of the application is to implement the navigation to the shop feature. Map integration into the PriceCart application was a challenging task that was taken serious during the project time and postponed due to the lack of knowledge. Even though the development of the activity is stopped the researches did not stopped and map implementation will be easily implemented by done researches. Solving the problem it was important to connect the Android API level 18 with Google Maps API that was an issue in the first place.

**Menu integration**

Making application more structured an implementation of the menu is needed. Menu will help to put many things in one place and keep the application simple to reach the goal as simple as possible. The menu will consist of search engine possibility, preferences, credits, search history and log out function.

**From local to external**

In order to fit time frame and have a working prototype until the end of the semester we have used local SQLight database. Our general idea was to have an external online server. Our team will revise the code of our android app and re-code it with. We will have to use a webhost server with PHP, API and MySQL compatibilities. (App submits to API, API stores on DB and gives response to App).

**Conclusion**

As mentioned in the beginning, finishing all the feature development that was mentioned in this section will lead to a real realization of the application bringing some sort of profits. The future development will start right after the hand in deadline. The real motivation for this project will be shown in the future if it will be finished or given away.

# Bibliography

* 1. Blanchard S. B., Maintainability: A Key to Effective Serviceability and Maintenance Management, John Wiley & Sons Inc., NewYork 1995

1. http://visual.ly/color-psychology-logo-design
2. http://www.iai.uni-bonn.de/III/lehre/vorlesungen/SWT/RE05/slides/09\_Non-functional%20Requirements.pdf
3. http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/
4. https://itunes.apple.com/dk/app/tilbudsavis/id412905815
5. https://play.google.com/store/apps/details?id=tilbudsugen.app
6. http://www.projectsmart.co.uk/moscow-method.html
7. http://www.mindtools.com/pages/article/newTMC\_05.htm
8. http://sict.moodle.aau.dk/file.php/914/Course\_Material\_subject\_01/context\_sch.pdf
9. http://answers.oreilly.com/topic/2576-youre-using-https-but-what-does-it-do/
10. http://fortumo.com/country/denmark
11. http://www.cs.toronto.edu/~sme/CSC340F/slides/17-specifications.pdf

1. http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/ [↑](#footnote-ref-1)
2. https://itunes.apple.com/dk/app/tilbudsavis/id412905815 [↑](#footnote-ref-2)
3. https://play.google.com/store/apps/details?id=tilbudsugen.app [↑](#footnote-ref-3)
4. http://www.projectsmart.co.uk/moscow-method.html [↑](#footnote-ref-4)
5. http://www.mindtools.com/pages/article/newTMC\_05.htm [↑](#footnote-ref-5)
6. http://visual.ly/color-psychology-logo-design [↑](#footnote-ref-6)
7. http://sict.moodle.aau.dk/file.php/914/Course\_Material\_subject\_01/context\_sch.pdf [↑](#footnote-ref-7)
8. http://answers.oreilly.com/topic/2576-youre-using-https-but-what-does-it-do/ [↑](#footnote-ref-8)
9. http://fortumo.com/country/denmark [↑](#footnote-ref-9)
10. http://www.cs.toronto.edu/~sme/CSC340F/slides/17-specifications.pdf [↑](#footnote-ref-10)
11. http://www.iai.uni-bonn.de/III/lehre/vorlesungen/SWT/RE05/slides/09\_Non-functional%20Requirements.pdf [↑](#footnote-ref-11)
12. Blanchard S. B., Maintainability: A Key to Effective Serviceability and Maintenance Management, John Wiley & Sons Inc., NewYork 1995 [↑](#footnote-ref-12)