

Vinguide

- Performing qualified product selection,
based on user generated ratings and using
image based search on a smartphone.



Master's Thesis

by

Group mea111030

Kasper Larsen Ekelund
Jeppe Veirum Larsen
Jacob Sloth Laursen

Long master project - Mediaology
September 2010 - May 2011

Abstract



**Department of Architecture,
Design & Media Technology**
Niels Jernes Vej 12-14
Phone +45 99 40 87 92
Fax +45 9940 9788
<http://www.sict.aau.dk/>

Title:

Vinguide - Performing qualified product selection, based on user generated ratings and using image based search on a smartphone.

Theme:

Master's Thesis

Project period:

September 2010 - May 2011

Projectgroup:

mea111030

Participants:

Kasper Larsen Ekelund (Technology)

Jeppe Veirum Larsen (Design)

Jacob Sloth Laursen (Technology)

Supervisor:

Thomas B. Moeslund

Publications: 5

Number of Pages: 246

Appendices: 4

Finished 31/05-2011

Synopsis:

This master's thesis investigates how an application for a smartphone can be designed and developed to guide its users towards a better insight of the quality of a wine based on user generated data. By using the smartphone's built-in camera, it should be easier for the user to choose a wine by snapping a picture of it. The focus lies on the creation of a complete product that has the potential of release to the Android Market. The end result, is an app that is able to search for wines in an online database, by utilizing the smartphone's camera to read the barcode. Additionally it is possible to mark wines as a favorite and attach a personal note. Advanced search is also possible, using normal text search and defining different criteria. The user generated data relies on users giving ratings to wines in the database. 4 studies have been conducted to support the development which has undergone two iterations. A questionnaire to support the need for the app, a heuristic usability test to support the initial design, a large user test after the last iteration, and lastly a field study in a wine department to conclude on the project as a whole. The overall result is very positive, and nearly all users showing highly interest in the app.

The content of this report is public, but may not be published without written approval from the authors.

Copyright ©2011, project group mea111030.

Contents

Preface	ix
I Introduction and Analysis	1
1 Introduction	3
1.1 Globalization	3
1.2 Vision	4
1.3 Initiating Problem	5
2 Analysis	7
2.1 Smartphones and Apps	7
2.2 State of the Art Technology	7
2.3 Image Based Search	11
2.4 Delimitation to Wine as the Product	13
2.5 Initial Idea	13
2.6 Delimitation to Smartphone Application	16
2.7 Market Analysis	17
2.8 User Generated Content	24
2.9 User Analysis	24
2.10 About Wine	35
2.11 Developing Platform	38
2.12 How to Recognize a Wine	40
2.13 Data management	43
2.14 Problem Statement	45
II First Iteration	47
3 Design	49
3.1 Fundamental Concept	49
3.2 Functionality Overview	52
3.3 User Interface Design Theory	53
3.4 Application Features	61
3.5 Designing for Android	64
3.6 Design Patterns	66
3.7 Low Fidelity Design	69
3.8 Data Source	83
3.9 Wine Features	84
4 Implementation	87
4.1 Implementation Overview	87

4.2	Barcode Structure and Theory	87
4.3	Database and Storing Data	92
4.4	Database Administration	98
4.5	Client - Server Communication	99
4.6	Server	101
4.7	Multi-threading	107
4.8	Developing for Android	108
4.9	Application Implementation	112
5	First Prototype Overview	121
6	Usability Test	127
6.1	Heuristic Evaluation	127
6.2	Test Setup	128
6.3	Test Conclusion	131
7	First Iteration Conclusion	133
III	Second Iteration	135
8	Revised Interface Design	137
9	Implementation	145
9.1	Implementing Notes and Favorites	145
9.2	Implementing The Search Screen Interface	146
10	Second Prototype Overview	151
11	Evaluation	157
11.1	Two Part Test	158
11.2	General Test Structure	158
11.3	Test Part 1	160
11.4	Test Part 2	166
11.5	Conclusion Whole Test	170
12	Second Iteration Conclusion	173
IV	Closing	175
13	Release on Android Market	177
13.1	Statistics	177
14	Future Improvements	179
14.1	User Rating in List View	179
14.2	Add to Favorites	179
14.3	Login and User Creation	179
14.4	Wine Buddy	179
14.5	Adding Wine and Suggesting Changes	180

15 Field Study	185
16 Discussion	189
16.1 Barcode Reader Error	189
16.2 Global Wine Rating	189
16.3 Prospective	189
17 Conclusion	191
Bibliography	193
Appendices	197
A List of Figures and Tables	199
List of Figures	199
List of Tables	205
B Initial Questionnaire	207
B.1 Questions	207
B.2 Answers	216
C Android Market Statistics 26. May 2011.	223
D DADIU Summery	225
D.1 Broken Dimensions - Kasper Larsen Ekelund	225
D.2 Blendimals - Jacob Sloth Laursen	228

Preface

This is a long master's thesis project report conducted in the period September 2010 to May 2011. The report is divided into 4 Parts:

- Part 1 - Introduction and Analysis
- Part 2 - First Iteration
- Part 3 - Second Iteration
- Part 4 - Closing

The final product of the project is an application for smartphones running Android. Everyone with an Android smartphone can download and try the application. To try the final product the application can be downloaded from the Android Market by searching for the name "Vinguide".

Kasper Larsen Ekelund (Technology Specialization) and Jacob Sloth Laursen (Technology Specialization) attended the second and last production at The National Academy of Digital, Interactive Entertainment (DADIU) during the semester from the 1st of March to the 1st of April 2011 as Game Programmers on the games Broken Dimensions and Blendimals, respectively. A short summary of their stay is placed in appendix D.

Also contained in the appendix is the list of figures and tables.

The attached DVD contains:

- Unedited video from Usability Test
- A digital copy of this report in PDF format
- A/V Production
- Results from the initial questionnaire
- The Blendimals game related to DADIU
- The Broken Dimensions game related to DADIU

Part I

Introduction and Analysis

1 Introduction

Looking back the last 20 years, one of the things that really has been at the frontier of new technology development and implementation is the mobile phone. It started as the name states, as a mobile device capable of making phone calls without a landline connection. That sole purpose started to change when the Short Message System (SMS) became increasingly popular. Especially popular among younger users, taking advantage of the much cheaper SMS messages compared to the high minute prices. Slowly the displays evolved from single or dual lined displays, to bigger resolution and color displays, capable of showing several lines of text, and giving the user the opportunity to play some simple games like snake. As the new millennial continued on, the Internet was on the rise and the mobile industry tried to incorporate a simplified version of it called WAP. It was slow, nearly impossible to navigate, costly, and in the end it didn't really catch on. Two more successful additions to the mobile phone were camera and Walkman-like music playing capabilities. Now it was possible to take a picture with your mobile phone and send it in an advanced form of SMS, called Multimedia Message Service (MMS) that made it possible to send a photo or a short animation combined with a short text message.

Despite the innovation in the mobile industry one thing did not change. Phones could differ in size, but they were all built using the same two components: a numeric keyboard and a screen. The interaction with a mobile phone was the same as it had always been. In 2007 a paradigm shift hit and shocked the mobile industry when Apple released their iPhone, soon to be followed by Google and their Android OS. A big multi touch screen replaced the normal screen and the numeric keyboard. In addition, computing processing power raised to a level never before seen in a mobile phone. Because of the many new features and abilities the phone offered, this was called a smartphone. With the smartphone a new phenomenon called App Store emerged. The App Store suddenly made it easy to make and distribute small apps at a low cost making it an attractive platform for mobile developers. Soon it was possible to remote control your TV, checking the weather for the next 14 days, use the built in GPS for turn by turn navigation, or destroy green pigs in wooden buildings with a bunch of angry birds...

Wherever you go, you have your smartphone, which gives you access to, phone, camera, e-mail, calendar, Internet browser, Facebook, right in your pocket, not missing out on a single important moment. In figure 1.1, an illustration of some of the used to be, standalone features, now integrated in the smartphone can be seen.

1.1 Globalization

Years ago, when buying products, it was an easy choice. Every category of products only had a limited selection. Today, choosing a simple product from the store can be a daunting task. There are literally a thousand different wines, cheeses, soaps, etc. to choose from. Add to

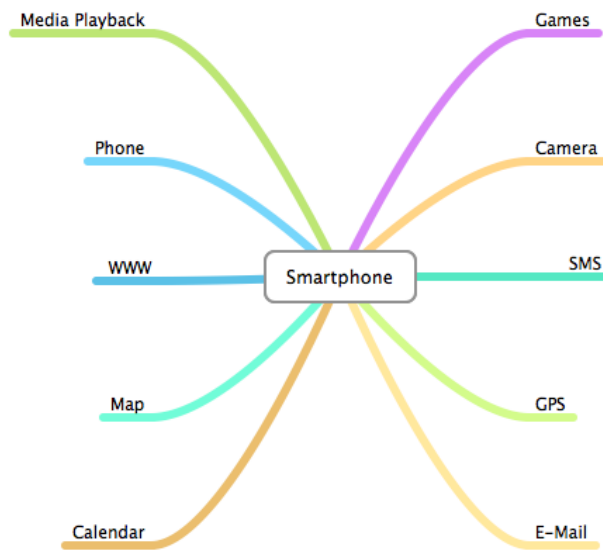


Figure 1.1: *Some of the features included in a modern Smartphone. Each feature has a standalone equivalent slowly being killed.*

that, the products that can be found on the Internet, and you practically get the whole world to choose from. You should think that more choices were only a good thing. Though in a hectic world, time it is not what people got the most of. Browsing the shelves, more choices do not equal a better product, because how do you find the best product of the bunch. How do you choose?

More and more people use the Internet to compare prices, read reviews of products, and read user comments on products they consider buying. There is a drawback though. Using the Internet to find reviews and comparing products, often take some time, if not considerable time. Imagine standing in the supermarket or the local electronics store about to make a buy. You are in a hurry and do not have the time to sit down hunting reviews or compare prices on the Internet. You could ask the staff for help. Unfortunately they have to make a living. They will do whatever they can to sell you something in their store. Beside the fact that the staff is salesmen and the time where people knew what they were talking about is long gone. People are working in big companies grossing too many different products that they by no means can know everything about everything. The outcome is that the quality of help is often not adequate.

1.2 Vision

Every smartphone have a camera of a decent quality and the ability to connect to the Internet to retrieve information. Instead of typing text to search the Internet like in the more traditional way, the smartphone's camera is used to snap a picture of a desired product. The smartphone then analyses that picture and return the relevant information about the given

object on the picture to the user. This system could be useful on practically any object of interest. For instance snapping a picture of a famous statue, thus giving historical facts, or snapping a picture of the television at the store to get the true specification, or the snapshot of a movie poster that would quickly provide newspaper reviews.

1.3 Initiating Problem

Imagine a system that, by snapping a picture of an object, provides the user with valuable information about it, fast and easy.

Using a smartphone and an app as focal point, how would such a system be designed and developed for easy to use?

2 Analysis

2.1 Smartphones and Apps

“Smartphone” and “App” has been the new buzzwords in the mobile industry lately. But what is a smartphone exactly and what is an app?

There is no actually standards a mobile phone need to fulfill to be called a smartphone. There are though some common features that are expected from a mobile phone to be called a smartphone. A multitouch display and hard or soft querty keyboard. PDA like features such as calendar, mail and contacts. Camera phone features such as taking pictures, shooting video, making phone calls, send SMS and MMS. Media playback features like video and music and the last and maybe most important feature: the ability to browse the full Internet. [1].

When you say smartphone you might as well say app. The word App is acronym derived from the word application. An Application is a software program that is designed to help the user complete a specific task, but the acronym App is however slowly losing its original meaning and is starting to refer only to smaller and simpler applications, designed to handle a single to a few tasks. [2].

2.2 State of the Art Technology

The vision of retrieving information about objects in a picture snapped by a mobile phone is not new. The subject has been discussed broadly, and already products, that uses such technology exists.

It is rational to look at those applications and technologies to see what can be accomplished, and what already exists. This section will look into those technologies.

2.2.1 Object Recognition for the Internet of Things

The research paper present a system which allows requesting of information on physical objects by taking a picture using the mobile phone[3]. Objects themselves do not have to be tagged with any kind of markers. The core of the system is an object recognition method, which identifies an object from a query image through multiple recognition stages, including local visual features, global geometry, and optionally also metadata such as GPS coordinates.

The system present two applications, namely a slide tagging application for presentation screens in smart meeting rooms, and a city guide on a mobile phone. The two applications act similar to each other, and following is a brief look at the city guide application.

The city guide application, hyperlinks physical buildings to digital content. The user snaps a picture of a tourist attraction to retrieve information. This gives nearly unlimited number of objects and therefore the functionality is restricted geographically by reading the GPS position and the current cell tower id. The basic idea of the city guide can be seen in figure 2.1



Figure 2.1: Client software for the city guide application: the user snaps a picture, waits a few seconds, and is redirected to the corresponding Wikipedia page [3]

The object recognition method is very similar in both applications: Putative matches between pairs of query and database images are found by nearest neighbor search for their SURF descriptors [4]. These putative matches can be validated with a geometry filter. Local descriptor algorithms and SURF will briefly be discussed later in the report, section 2.12.

To verify the robustness of the application an experiment was conducted. A database consisted of 147 photos of 9 touristic sights and their locations covering the objects from multiple sides. The database images were taken with a regular point-and shoot camera.

To test the system there were collected another 126 test query images, taken with different mobile phones, (Nokia N70 and Nokia 6280, both with 2 Megapixel camera) at different days and times of day, by different users and from random viewpoints. Of the 126 query images 91 contain objects in the database and 35 contain images of other buildings or background.

Table 2.1: Summary of recognition rates for city guide [3]

	Prec. with Geometry Filter		Prec. without Geometry Filter	
	Rec. rate	Avg. Matching Time	Rec. rate	Avg. Matching Time
Full database linear	88%	5.43s	67.4%	2.75s
GPS 300m Radius	89.6%	3.15s	76.1%	1.62s
Cell id	74.6%	2.78s	73%	1.34s

The result can be seen in table 2.1. As seen by the table the results have quite a high success rate. It indicates that it is not impossible to realize the addressed vision. The problem however is of course the massive database that has to be build, and the processing time it entail. Another key problem is how to present the information that is retrieved.

2.2.2 Scene Recognition with Camera Phones for Tourist Information Access

The article describes a system much similar to the above research[5]. It is a system using image input modality for information access in tourism applications. It is a working system that provides multi-modal description of text, audio, and visual, of a tourist attraction, based on its image, captured and sent by a camera phone, see figure 2.2.



Figure 2.2: *Image-based mobile tour guide [5]*

The prototype, called Snap2Tell, is developed and tested on a Nokia N80 client. The client-server protocol is developed using XML for communication over Wi-Fi and GPRS. Through the protocol, the client sends image queries and receives recognition results.

The prototype uses a pattern discovery approach to find local image patches that are recurrent within a scene class and discriminative across others. This selection strategy generates positive training patches for discriminative learning.

The work, report preliminary scene recognition results on 90 scenes, trained on 5 images per scene, with an accuracy of 92% and 88% on a test set of 110 images, with and without location priming.

Again a more or less successful product is presented, which again indicates the possibilities of realizing the vision described.

2.2.3 Present Mobile Scanner Applications

For the current smartphone market there already exists a couple of different apps that do image based searches. To name some of the majors:

- Google Goggles[6]
- eBay's Red Laser[7]
- ShopSavvy[8]

The above smartphone apps, are all applications which try to return relevant information about the subject that the camera is pointed at.

Though Google Goggles stands out in that it can use image recognition tasks to query the whole Internet, they all have in common that they can recognize standard barcodes, to find pre-indexed objects.

Google Goggles are by far the leading attempt to search the Internet by any means of visual information, as it encourages you to take pictures of text, landmarks, books, contact info, artwork, logos, etc. In addition it supports common barcode systems.

Red Laser features only bar code scanning of indexed products, and can return any relevant information back to the user about price, reviews etc. The drawback is of course the internationalization. It is not able to recognize Danish products and lacks information about prices in Denmark. This is primary because the target-audience is the US and UK consumers and not spanned to European countries as Denmark which makes the application less interesting for Danish consumers.

ShopSavvy have some of the same features as Red Laser. It enables the user to scan a barcode to get information about a product. The main functionality lies in price comparing and to be able to find nearest stores based on GPS position and cell tower id. But the same constraints as seen in Red Laser apply to ShopSavvy, and its target audience is the US consumers. ShopSavvy is partly user driven, and any user can submit data to the database.

Screenshots of the above mentioned applications can be seen in figure 2.3.

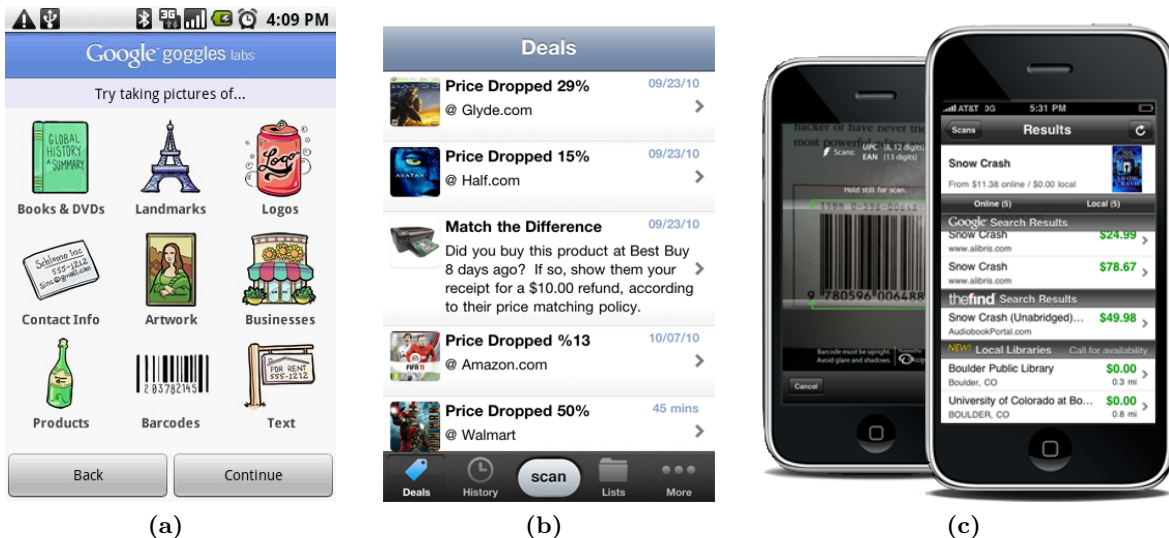


Figure 2.3: Screenshots from different mobile applications using the scanner principle. 2.3a: Google Goggles. 2.3b: ShopSavvy. 2.3c: Red Laser

2.2.4 Conclusion on State of the Art Technology

The applications mentioned above, is in a Danish perspective not so interesting because locale products are not to be found in the catalog and prices cannot be listed as actual prices in DKK.

A drawback on present smartphone applications like Google Goggles and ShopSavvy may be that they try to be too broad. Trying to encompass every product across every category is a big task.

When a user uses Google Goggles, Google does not know what the user is searching for, other than it has something to do with what is seen in the picture.

As example, imagine taking a picture of a Coca Cola bottle with Google Goggles. Does the search return an image of another Coca Cola Bottle, the Coca Cola logo, nearby stores, price comparing, reviews, the history of Coca Cola, or maybe the latest stock exchange information. It is hard to define the search result, and thus it is hard to use it as a searching device. On a standalone PC it is not as big of a problem, because the user have much better overview. It is easy to redefine searches, add keywords, and exploring different web pages. On the smartphone it can be very frustrating when the search is not displaying what you were looking for.

In the same way, when looking for a product to buy and want informations, you turn to the place where you know such information is present. If you want to find the price of a digital camera, you go to Price Runner or EDB Priser, and if you want to find reviews, you go to a consumer electronic magazine.

The conclusion and hypothesis is, that it is better to focus on only one category, only one product, or only one service, and specialize in returning one specific information. By that, the users know exactly when to use it and what to get, and the system knows exactly what to search for and what to present for the user.

2.3 Image Based Search

This section will briefly look at the phenomenon referred to as image based search, with focus on usage on a mobile phone.

Image based search, as defined in this report, is the process where a picture of any arbitrary size and content is used as a search query. This, opposed to a normal text based search, requires very advanced algorithms to be able to find any matches. This is partly because images contain far more data than a text string does, but also because image based information retrieval goes beyond only matching images, as information in other modalities also can be extracted from data collections.

On a standalone PC, image based search is not always very convenient over normal text based search. On a mobile device it is another story.

Making text searches on devices such as smartphones differ quite considerably from making searches on normal computers. First, the size of the display is small. Even though 4 inches

on some high-end smartphones is a larger display than a normal mobile phone, it is still far from the size of an average desktop computer. Second, text input is slower than with a full desktop keyboard and in many mobile situations, attention is divided which makes text input even harder. Third, as opposed to a desktop computer, the smartphone has a camera ready to shoot, anytime, at anything.

The problem about image based search is, as already mentioned, the difficulties in interpreting the data in the image. The image used for a search often present a vague query, as it is difficult to tell exactly what is wanted from the picture. At the same time it is suggested that image search could be used instead of, or as a complement to, a text-based search when users' search needs are vague, and/or text as input may be hard to define [9].

Another problem is the calculation needed to crunch through the data represented by an image. This can be an extensive task to accomplish on a mobile phone. An approach is to send the image into the cloud and get it processed by a faster external processor. This raises a new problem, which is bandwidth. Today through, it should not be as big of a problem because of the extensive use of mobile networks. On the other hand it can still be an abundant task, even for the fastest desktop processors to do on the other side of the phone, when search queries rise and the number of pairing matches increases.

Very important is it also to mention that from a user perspective it is a very subjective judgment whether an image is a good candidate. Applying this to an image matching system, a user might believe that an image is of bad quality, but it may be good enough for the system to find and match other images. The opposite can also be true, the user thinks the image is sufficient, but the system cannot handle it, or presents bad matches. Many factors can affect image quality that the user not necessarily is aware of.

User Experience Regarding Image Based Search

A study presented in the paper A Picture is Worth a Thousand Words [9], tries to explore users' experience of using image based web searches on a mobile device.

The study showed that the search performance was dependent on the search type. The findings are, that in some cases it is more effective and faster to use a text based search engine. Users may be looking for an image to illustrate a general theme, not exactly the object in front of them.

This indicates that image based search is most valuable when you know exactly what results to get from the image, just as was concluded in the State of the Art section, 2.2. With an app that specializes in a specific product, the user would know exactly what to get as a result from snapping a picture of that product.

In general, the respondents from the study seem to think that text based search is more accurate and of more use than image search, but that it is faster to snap an image than it is to enter a text query. Again, that is why a system using image based search should be very specific. If a user wants a review of a product, and he knows he will get it by snapping a picture of it, then a text search would be needless.

2.4 Delimitation to Wine as the Product

The conclusion from the above section 2.2 is that it is a good idea to focus on one category or one product. For that reason we delimit the vision of this project to focus on a single product. This is done to be able to conduct a problem analysis. The category chosen is wine which is a product sold worldwide. Wine is a complex topic and is therefore regarded as a fitting candidate for the further analysis.

2.5 Initial Idea

Choosing a wine at the supermarket can be a daunting task. The selection of wine is often abundant and making a selection from the front labels alone is difficult because they all look great and all screams "buy me". Turning the bottle to get more information from the backside label can be just as difficult. They often tell the same story of a proud family vineyard producing a fabulous wine, using some old traditions at some exotic place somewhere in the world. The next step is to ask for help which also can be a challenge. In smaller grocery stores, youth workers are often the only employees in the vicinity of the wine department and probably only eying next month's paycheck.



Figure 2.4: *A View of the wine department in Bilka Skalborg with its abundant selection and a forest of yellow price tags.*

These conditions, and an often packed schedule, are often a reason why asking for help is

abandoned. Most likely, inadequate guidance or simple waste of time hunting down an employee who can do some proper guidance, is what you get. For many people the outcome is a pick of wine that is just a gamble, or just a pick on the same old wine that always have ended up in the basket. And who knows, one wine could be just as good as another, maybe even having half the price tag.

2.5.1 The Wine Application

A system as described in the vision, section 1.2, could make a perfect candidate for a solution to the problematic scenario described above.

The idea is an application that installs on a smartphone making it available everywhere right from the pocket. With the wine application (theWine App) the user can easily search a wine of interest to get useful information, making him capable to do a much more qualified choice.

2.5.2 Conceptual Application Features

With the main feature the user uses the smartphone to snap a picture of a desirable wine, and the device quickly display all information of interest. The user can snap pictures of different wines comparing them against each other. See figure 2.5.

Information regards to everything that would help the user to a more qualified choice. Here is a small list of possible information the user might be interested in:

- User ratings
- Expert reviews from newspapers, magazine, etc.
- Food pairings
- Price comparing between wines and across stores
- Special offers on wines
- Regular information about the wine: Which grape is it, how is the taste, how old is it, and so on
- Recipes that goes with the wine
- Etc.

Some other features include, the possibility to access special offers or top rated wines in relation to the store the user is at, an encyclopedia to help self-education, or personal wine cellar, all together right on the phone.



Figure 2.5: *The main feature of the smartphone application is the ability to snap a picture of any given wine and quickly display relevant information*

2.5.3 The Big System

The smartphone application will be focal point and is the unique selling point for the system. It can be downloaded anywhere and accessed anywhere without involving anybody but the user.

To support the smartphone application a web portal will be available. On the web portal the user can access everything that is also available on the smartphone application and support potential extra features not possible on the phone. An illustration of the system can be seen in figure 2.6.

The smartphone application will be used on the go; in the store picking the right wine, on the restaurant to add the menu's wine to favorites or with the friend showing yesterday's good wine.

When at a standalone computer, the user has better overview and better control over all the information available. The user can manage its user profile and personal favorites much better, and organize the personal wine cellar. She can browser other users' favorites, print recipes, and share new discoveries on the forum.

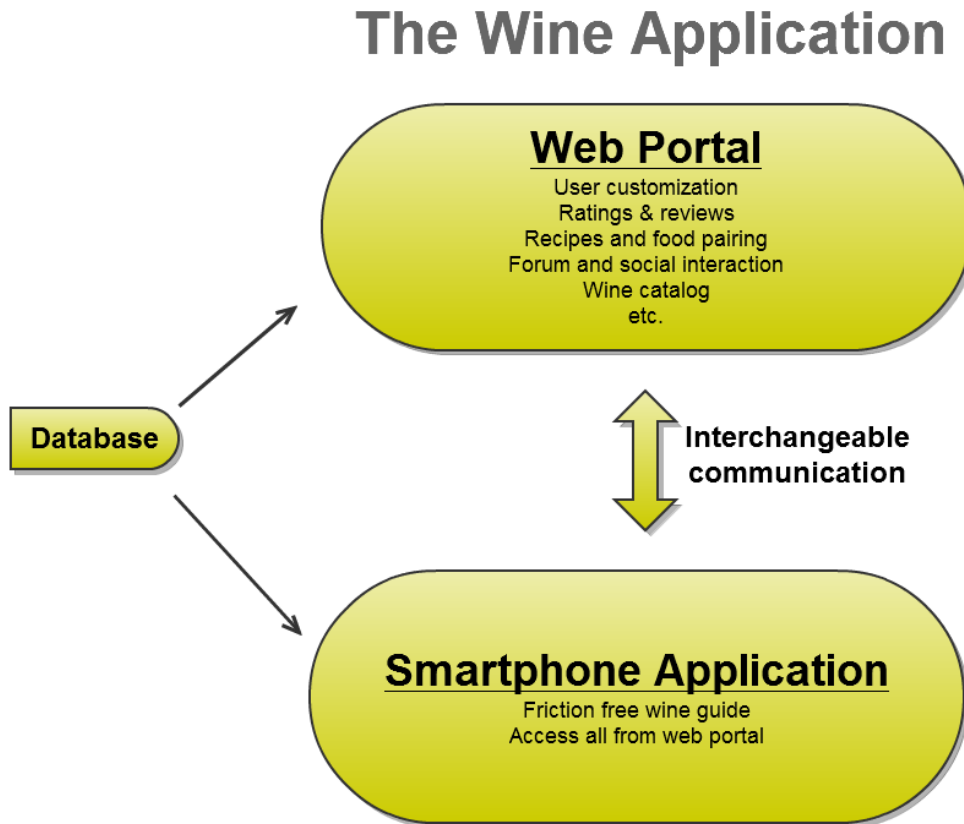


Figure 2.6: The illustration show the overall system of the wine application with a smartphone application used on the go and a web portal for better overview when time is to it

2.5.4 User Driven Application

Part of the initial idea is to make portions of the content in the application user generated, by having the user to participate in the content creation by own good will. For example the user might vote or rate for a wine, add missing descriptions or maybe add clues about special offers and so on. Advantage of that is of course less workload on the developers shoulder, but hopefully also a more immersive user that cares about the content and trust the content.

2.6 Delimitation to Smartphone Application

Both the web portal and the smartphone application described in the previous section are potential two very big projects.

The smartphone application is the systems unique selling point and the part which support the projects vision, whereas the web portal is more of a side product. To be able to make a finished and polished product it is chosen to focus solely on the development of the smartphone application as a wine guide. The web portal will not be discussed further.

2.7 Market Analysis

Before tossing into making a product that helps people when buying wine, it should be analyzed whether or not such a product would prove useful. To accommodate this, a market and a user analysis have to be made. This section will focus on the market analysis and later in the report the user analysis will be made.

2.7.1 The Wine Market

Since the seventies, wine consumption has had a steady increasing market share, and today wine is a very common consumer good which are consumed on many different occasions. Meininger's Wine Business International [10], does different analysis on the wine market and on the wine trade. According to an analysis done by Meininger, the Danish wine consumption in 1971 was around three liters per head. This increased to 37,88 liters per head in 2005 and Denmark was ranked as the seventh biggest wine drinking country by the International Wine and Spirits Record.

The statistics of wine as an increasing consumer good is supported by statistics done by FDB Analyse from February 2001 [11]. The analysis states that 19% of the Danes drinks wine on an average day and a Danish household spends about 2000 kr. a year on wine.

By examining the numbers from both studies it is clear that the wine market indeed is very big, and a product that potential will help users when buying wine could be useful.

2.7.2 Wine Assortment

Although the statement in above paragraph is true, it does not conclude that consumers actually have a problem when buying wine. Meininger also states that Coop, which is the biggest wine importer, retailer and distributor in Denmark and have a market share of 40%, have about 300-400 wines in their portfolio. That is a lot of wine to choose from and the number will just get higher if all of the other distributors of wines were added. Potential the user have difficulties in choosing the right wine with such a vast selection.

In addition, another survey mentioned by Meininger actually states that most consumers find it rather difficult to select a wine and that 75% will buy a wine they are already familiar with, rather than trying something new.

With wine being a very big market, with all the different wines to choose from, and consumers confused about which wine to buy, there should be a good chance that the product aimed for in this report could prove useful.

2.7.3 The Smartphone Market

For a wine app to be a success, of course there need to exist smartphones were the app actually can be installed and used on. According to Børsen [12], over half of all mobile phones sold in

Denmark are a smartphone.

In first quarter of 2010 only 29% of sold phones was smartphones. Second quarter the market share increased to 37% and in third quarter the number was 51%.

As seen, the sales of smartphones are exploding, and it would probably keep increasing a bit further.

2.7.4 Target Audience

The target audience is very much depicted by the aim of the product, where the aim is to create a smartphone application that will help a user when buying wine. This produces the following target audience:

- Users that use smartphones
- Users that drink wine

Smartphone Users

It must be assumed that users that have a smartphone actually use it as one, which means that they are aware of all of its functionalities and possibilities and also actually use them. Of course there will be a group that has acquired the smartphone without the intention, or knowledge, to use its potential. This could be one that has acquired it from a gift, from work, or just from the good salesman that has palmed off his deal.

Nevertheless, it is an all new way to interact with the world when using the phone to help with every given task. If it is assumed that smartphone users actually use it as a smartphone, it can also be assumed that these users do not have any issues regarding using apps for everyday tasks. After all, that is what a smartphone is capable off.

To narrow down the target audience it would be interesting to look at the gender distribution on smartphone usage. In figure 2.7 it is shown that on the total mobile usage there is a very even distribution on gender, while on smartphone usage there is a small skew towards the male gender.

Overall though, it must be said that the target audience regarding smartphone users is pretty even between male and female.

Also the age distribution for smartphone users is an interesting topic. One would expect that most of the smartphone users would tend to be at the younger part of the population because they usually adapt more easily to new technology. In figure 2.8 the age demographics for smartphone and total mobile users can be seen. It is shown that the total mobile audience overall remains comparatively flat across age groups. Although 55+ pops out of the chart, it must be remembered that this group covers a much larger age group.

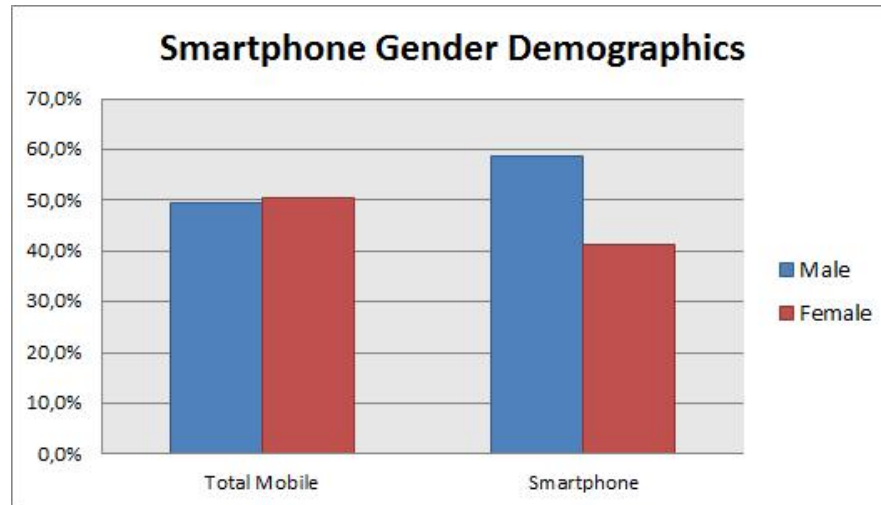


Figure 2.7: Gender distribution on mobile phones compared with smartphones. (Based on subscribers, Europe numbers, 2011) [13]

Contrary, when looking at the age distribution for smartphones it is clear that it skew towards persons aged 25-44 and elder distribution goes down a lot.

Though the usage of smartphones tend to be persons aged 25-44 there is still a significant use by persons outside this age group, and it is difficult to point out a specific age group regarding the target audience.

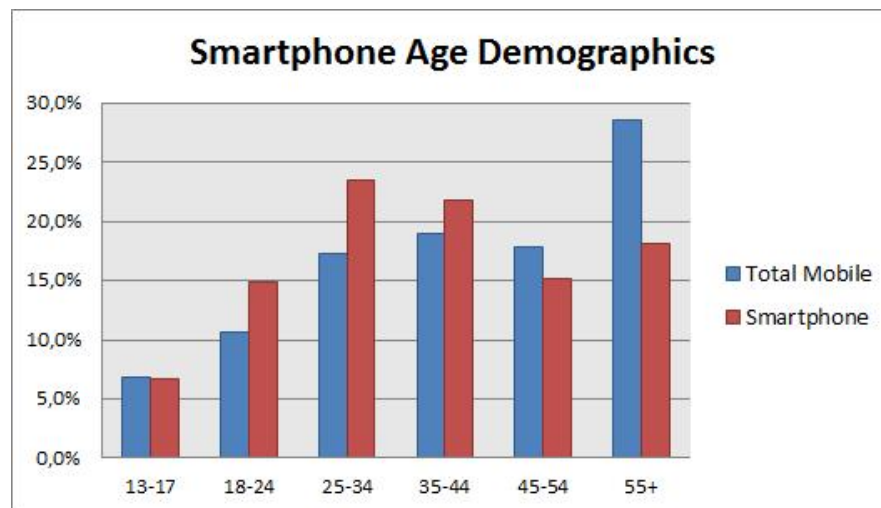


Figure 2.8: Age distribution on mobile phones compared with smartphones. (Based on subscribers, Europe numbers, 2011) [13]

Wine Users/Consumers

The general wine user has already been analyzed a little in this section. But it would be interesting, now that the age group for smartphones has been laid out, to examine the age

group of wine users. FDB Analyse has done such an analysis which can be seen in figure 2.9. There is a great tendency toward elders drinking the most of the wine. A very important notice here, is that the graph shows how much percentage of the age group drinks wine on an average and not how many of the different age groups that drinks wine in general.

So while the smartphone users tend to be younger, it is reversed for those who drink wine which tends to be older. This is a risk to take into consideration when developing a wine application.

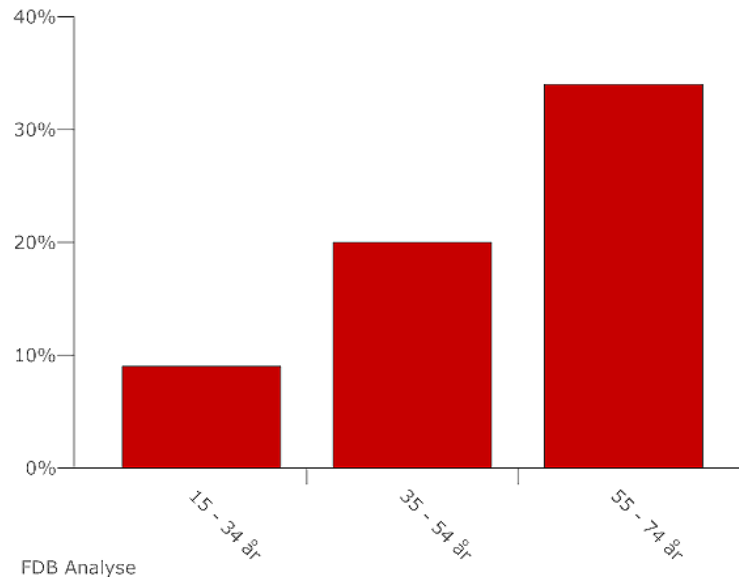


Figure 2.9: *While every third above the age of 55 drinks wine on an average day, the same only apply to every tenth below the age of 34. [11]*

2.7.5 Present Wine Applications

From the above analysis it seems rational to make a wine application for the smartphone. There already exist a lot of wine portals on the Internet where users can search for wines, rate wines, make comments, read reviews and compare prices, etc. Again this proves for the usefulness of a wine application. This section will try to cover the current market situation regarding other wine apps.

A quick search on Google [14], Android Market [15] and Apple App Store [16], quickly indicates that there currently already is a lot of wine applications present. In the following, a brief walkthrough of the top rated apps will be discussed. The following features are interesting in regards to the vision laid out in this project:

- What is the apps unique selling point / main features?
- How does the user search for a wine? Is it possible to use other than text search, e.g. use the camera to take a picture?

It is important to notice that the list of wine applications shown in this section is applications that were found in this project's start (September/October 2010).

Drync Wine

Drync Wine is an application for both Android and iPhone. The main feature of the program is to search for a wine using text, and preview your personal virtual wine cellar. That is basically it. When you have searched for a wine, you get basic information and you can add it to your cellar. You can also rate a wine, make notes, and make different tags like, "I want", "I drank" and so on, enabling to differentiate the wines in your cellar.

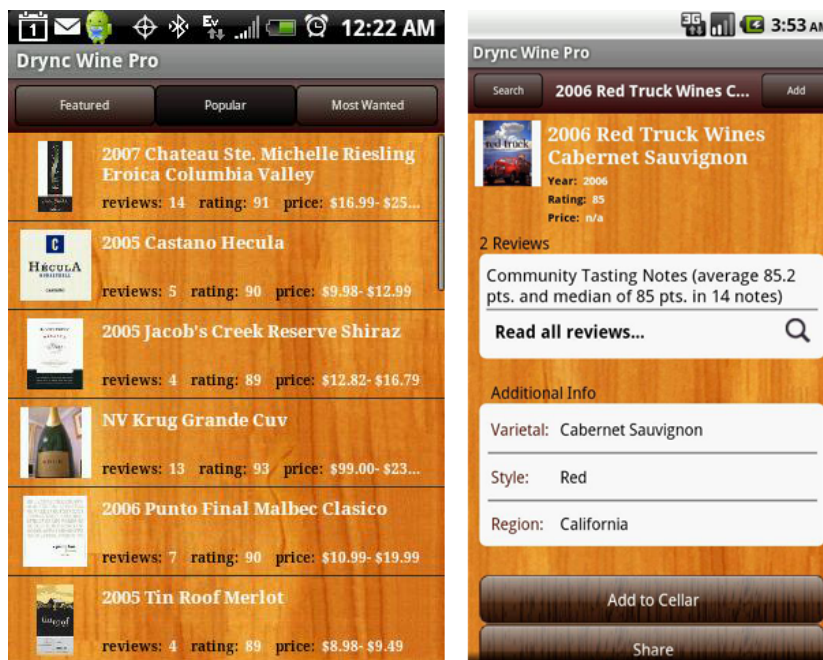


Figure 2.10: Screen shots from the Drync Wine app. [15]

Vivino

Vivino is an application for Android, iPhone and Windows Phone 7, which let you keep track of the wines you have tasted. With the app you can take a picture of a wine label and it will recognize it in about 33% of the cases, and the wine is then added to your list of personal wines. In the remaining cases it needs to be identified by the creators and manually added to your list. The user can tag the wine as a wine they like or dislike. The app is tightly connected to a website giving the user possibilities to get further information like ratings, dealer etc. [17]

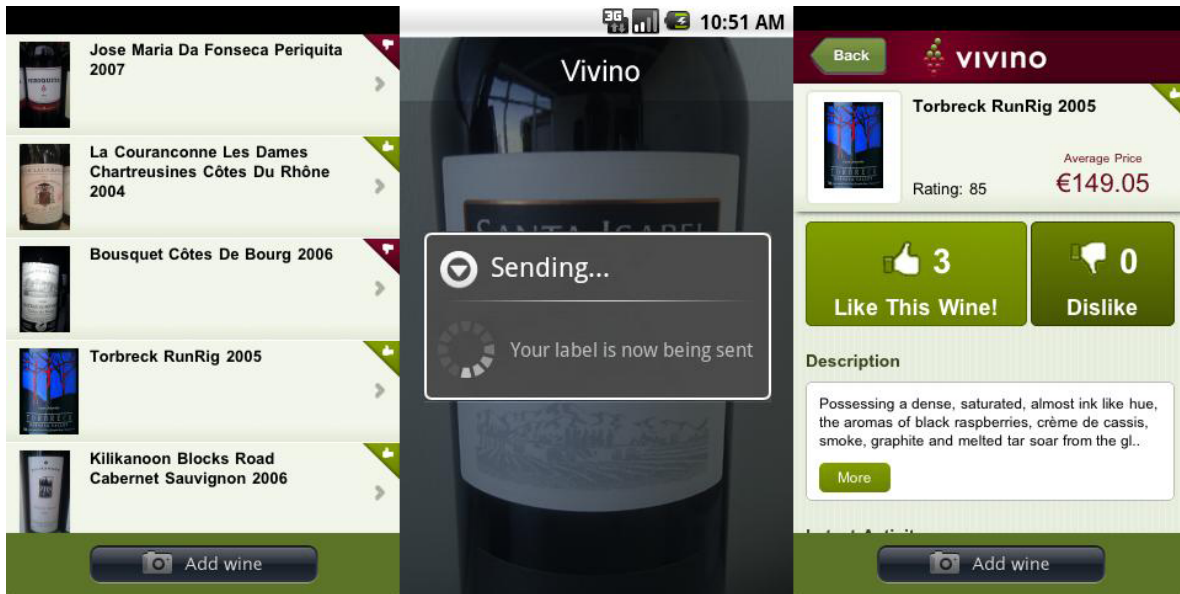


Figure 2.11: Screen shots from the Vivino app. [15]

Din Vintjener

Din Vintjener is a very elegant Danish wine application for iPhone. The basis of the program is to choose between two modes, either search by wine type or search by food pairings. You cannot do any form of specific search by for example typing a keyword. Instead you choose a category like red wine, and are then presented by a list of red wines. When searching by food pairings, for example you select main course and then a list of different recipes appear. Each recipe has some connected wines which pairs well to the food. It is not a very large database of wines or recipes. In addition the app only helps you to find French wines.

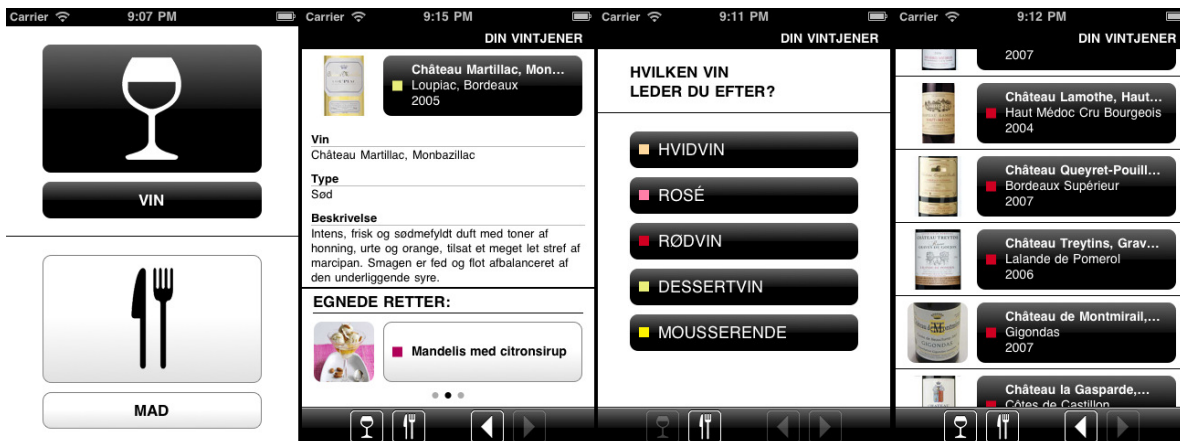


Figure 2.12: Screen shots from the Din Vintjener app. [18]

Hello Vino

Hello Vino is a very extensive application for both Android and iPhone. The main feature of the program is to try to find the perfect wine that you are looking for. You can search by food pairings, wines for specific occasions, by taste preferences, by country, region or by special events. If you already have a wine you can find food that pairs good with it, based on the grape varietal. It is not possible to search for a specific wine by using text or any other method.

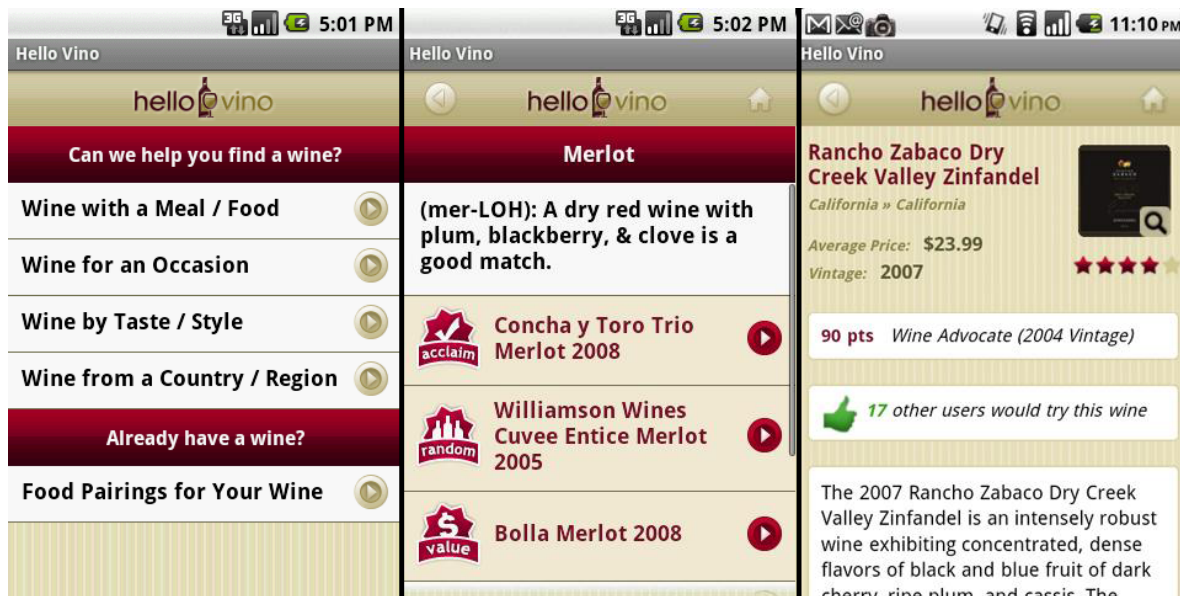


Figure 2.13: Screen shots from the Hello Vino app. [15]

Conclusion on Present Wine Applications

There exist a lot more different wine applications than described in this section, both for Android and iOS. Most of the apps that were looked at, tend to focus on the personal aspects. That is, add wines to a favorite list, add comments, etc. Many programs also did food pairings and the possibility to rate a wine and some did price comparing. Nearly all the applications that offered specific search only had normal text search as a possibility. The only program that differed was Vivino where you can take a picture of the wine label and use it as a search item, just the way the vision for this project describes it. Unfortunately it does not work well, and the developers also states that it only works 33% of the time. The final conclusion is that their definitely is a potential for a good wine application on the market. And if one could implement an innovative method using image based search, it would definitely get the market's attention.

2.8 User Generated Content

In the initial idea, section 2.5 on page 13, it is described that the wine app partly will rely on user generated content.

Today, user generated content is being adopted in many projects inside the digital world. The biggest players at the moment must be sites like YouTube and Facebook, which relies solely on the user's engagement to submit new content. These sites do not really create any content of their own, but only takes care of the storage for the content uploaded by their users. Other web sites, to name a few, could be Amazon, IMDB, Flickr, Twitter, etc.

Amazon for instance, has one of the world's largest online shops, selling everything from a crowbar to a television. The online shop at Amazon would work fine as a standalone, but one of the big advantages Amazon relies on, is their user submitted assessment of the products.

The strength of user generated content is that it is usually cheaper than professionally produced content, for one thing. And many users care more deeply about the content they have generated themselves than they do about the stuff that comes from the professionals. This leads to deeper levels of engagement from the user. In addition the amount of content that can be generated by the crowd is often vast compared to paid labor.

While all strengths sound intriguing, one has to be aware that they can also turn out as a drawback. Depending on the context within it is used, content created by users might not yield the quality required, and users may turn down services.

Another problem with user generated content, is the loss of control it involves. When creating content entirely on your own, you have full control of what is available and what is not.

Inappropriate content from users could emerge as any sort. For example missing or false information, bad behavior, and fight in comments and posts. From the user's perspective it could be on purpose, but could also be to damage the reputation of a product or person. Or it could just be to feed a strange desire, also called trolling. Another issue is that of what YouTube has had with repeated uploading of copyright infringing videos.

User generated content is about how to get the users to generate the kind of content that is wanted to be produced, instead of the kind they want to produce. Though it probably is inevitably to eliminate, one have to be creative to minimize inappropriate user generated content. Some solutions that are seen are registering of users, so that it is possible to ban inappropriate users. Another solution could be to award users for good content, for example by given points, making ranking systems or offer achievements.

2.9 User Analysis

As an extension to the market analysis made in section 2.7, this section, a user analysis will be conducted. The user analysis is made to further support the need of the wine app. The analysis is made on the basis of a small pre-study to investigate peoples consumption pattern when buying wine and using their phones. The study is not meant to be a major scientific

research, but rather a small study to hopefully favor the incentive to create the wine app. The section will describe what was sought from the study, how it was done, what was made from it, and what problems should be aware of.

2.9.1 Hypothesis

The desire for making a wine application as described in the vision in section 1.2 on page 4 of emerged from some hypotheses, and the study is sought to confirm these. The hypotheses have further been strengthened from the analysis that has been made until now. The hypotheses are as followed:

- Many people that buy wine are uncertain by their choice in different ways
- Many people just buy a wine by intuition, not having any specific idea if it is any good
- Many people would like more information about a wine but do not want to confront the staff
- Many people would like if they had an easy way to get information about a particular wine without having to confront staff
- Many people would find an app that helps them when buying wine useful

2.9.2 What is sought from the study

First it is of course sought to confirm the hypothesis described above. While examining users the chance of investigating other things should not go to waste. Therefore the study will also look into people's phone usage and whether or not they use smartphones, which is a prerequisite for using a wine app. Also it should be investigated whether the user normally retrieve information about products beforehand a buy or if they are generally more impulsive buyers.

2.9.3 Questionnaire

For the purpose of the study, quantitative data is preferable over qualitative data, as the questions that are needed answered are very specific. Also the data from a large population is needed to confirm any of the hypotheses and to find any tendencies or patterns. This is best done using a quantitative study. For that reason it is chosen to do a questionnaire. With a questionnaire the questions can be specific and a large group of people can be targeted and it is easy to do a statistical analysis.

The risk of using a questionnaire is that it can be hard not to make any ambiguous questions that different people interpret in different ways. So the questions need to be very specific to ensure that the answers the subjects submit actually are the answer that is sought for.

Following is a list of things that is wanted from the questionnaire. After each question it is explained why the question is needed and what is sought from it. The list of questions does not look exactly as they appear in the actual questionnaire that was sent out to the test persons. For a look on the actual questionnaire as a whole, refer to the appendix B. The questionnaire is in Danish.

Questions about wine:

Question:	What is sought from the question:
Does the test subject have a desire to tell others about a wine they have tasted?	This gives an insight in whether users would want to rate a wine after they have tasted it.
Does the test subject actually make the choice of the wine they buy or drink?	If the test subject does not choose his own wine, it is asked whether it is because he does not care or because he lack knowledge about wine. If the latter, then a wine app could be useful to them.
Does the test subject think that she has good knowledge about wine?	If users know a lot about wine, does they still want to have access to information through an app?
How often does a test subject go to the staff to get personal help on deciding on a wine?	This gives us insight in whether a test subject is used to confront the staff would find an app useful.
Which three factors about a wine does the test subject see as the most important?	This helps to define the content of the app. What content are most important and should be presented first in the app, and what information are less important. In addition it might be discovered what other information that users care about that were not thought of.
How certain do a test subject feel about the wine she buys?	The main feature of the app is to decrease uncertainty on the choice of a wine by providing useful information. But it would be insightful to know how uncertain a user actually is about buying wine.
If possible, does the test subject buy the same wine that she knows of or does she try out new ones?	If the user is afraid to try out new wines it might be because of uncertainty about good wines.
Would the test subject find a wine app as described useful?	The app is described for the test person on a very conceptual basis. This would give an indication whether the users would like to use such a program

Questions about phone usage:

Question:	What is sought from the question:
Have the test subject downloaded apps for her smartphone?	This is of course to see if they use apps.
How much does the test subject normally pay for an app?	This gives an insight on how much money people are willing to pay for apps.
Do commercials annoy the test subjects?	This is to see in how big scale we can use commercials

Miscellaneous questions:

Question:	What is sought from the question:
Does the test subject use the internet or other media to find information, reviews, etc. of any products beforehand a buy?	Spontaneous people might not find a wine app as useful as people that normally investigate products more before buying.
Does the test subject use social media like Facebook, Twitter, etc.?	Users that use social media may also want to use an app like the wine app.

Other mandatory questions:

Information about age, income, gender is also important, to establish any tendencies. It is also asked whether a test subject drinks wine or have a smartphone at all. If no, connected questions are excluded as they are needless.

2.9.4 Heuristic evaluation

Even for a small study like this, it can be extremely hard to get the information that is wanted. The questions have to be constructed very precise and in right order to avoid misunderstandings and answers that give misleading information.

To accommodate failures a heuristic evaluation has been done on the questionnaire. A heuristic evaluation is a method for quick, cheap, and easy evaluation and is normally used to find errors in user interfaces. But the method should be useful in detecting errors in a questionnaire as well.

For the test there were three evaluators taking the questionnaire. It was possible to take in another two evaluators if needed. But after the first three had completed the test it was not found necessary. According to Jacob Nielsen it is best to use 3-5 evaluators for a heuristic evaluation [19].

The evaluators is fellow students and is therefore experienced in the work and experienced in creating questionnaires which makes them good critics. Each evaluator completed the questionnaire separately and they were asked to think aloud, so that their thoughts could be analyzed on each aspect. The results and answers they produced were not used for the final questionnaire.

The results from the evaluation lead to minor correction, and the result is depicted in the following list:

- Some misspelling and questions that could be misinterpreted or that were not clear, was corrected.
- One of the evaluators said that there was good thread in the questions.
- Some multiple choice where corrected to not have a default answer selected.
- It was made more explicit that the section about general use of internet and social media did not only concern wine but concern all possible products.
- The length of the questionnaire was very good and did not have to be minimized.
- The questions were overall understandable and reasonable.

Overall the evaluators did not find many problems with the questionnaire and critics were pretty uniform across the test.

After a final iteration on creating the questionnaire it was sent out to friends and family through email and Facebook. The questionnaire is still active and a link can be found on the DVD.

2.9.5 Results from questionnaire

This section will cover the result of the questionnaire and what was learned from it. The section will not go into every bit of detail for every question as it would produce a small report of its own. Instead the section will focus on the most valuable discoveries that were made. In appendix B.2 it is possible to look through an auto generated summary of all answers.

The questionnaire had 80 responses, 40 males, 40 females. In figure 2.14 the age distribution of the respondents can be seen. It is clearly that the age distribution is not very even, and most of the respondents are between 22 and 26 years old, and clearly something that should be taken into account when drawing conclusions. On the other hand as seen in the market analysis on page 18 it somewhat correlate with the age distribution of smartphone users.

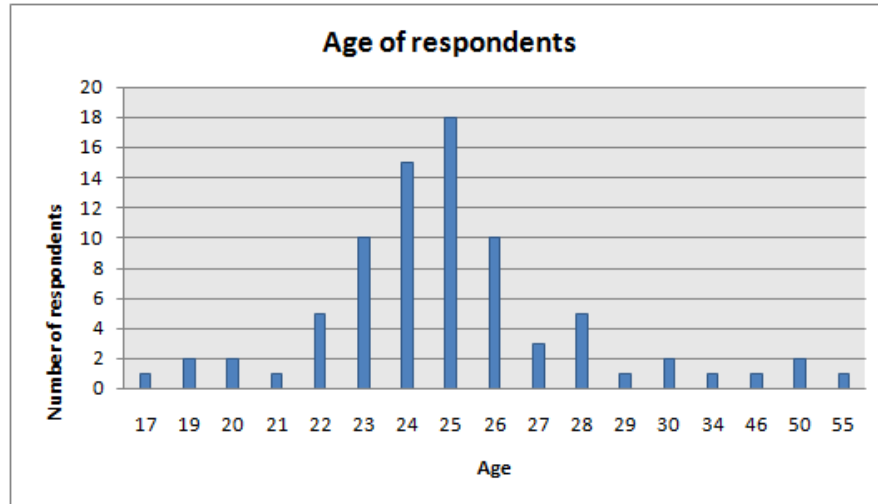


Figure 2.14: *Age distribution of respondents*

How many drink wine?

As a good introduction the respondents were asked if they drink wine at all. In figure 2.15 the results can be seen. It is only 13% of the respondents that never drinks wine and half of them drink wine often. This makes a good basis for a wine application.

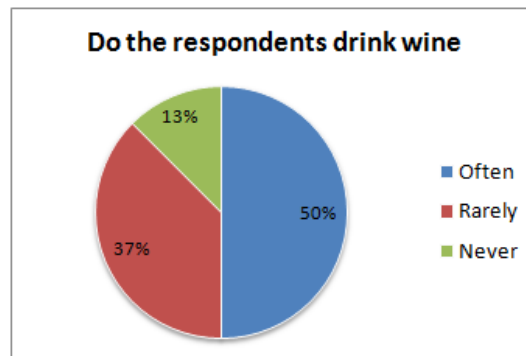


Figure 2.15: *How many of the respondents drinks wine*

How many make their own choice on wine

In figure 2.16 it can be seen if the respondents make their own choice on wine or they normally just drink what is being served to them. The graph is divided between those who drink often and those who drink rarely. It is interesting to see those who do not choose their own wine, whether it is because they lack knowledge or if they just do not care about wine. The answer to that question can be seen in figure 2.17. It can be seen that it is actually only 36% that say they do mind which wine they drink. Hopefully an app could help the rest to begin buying their wines, which leads to the next question where the same respondents are asked whether they would find such an app useful or not. The result can be seen in figure 2.18,

and demonstrates a great interest in the app from those that do not make their own choice of wine. 43% finds it useful to a great extent and 21% finds it useful to a medium extent.

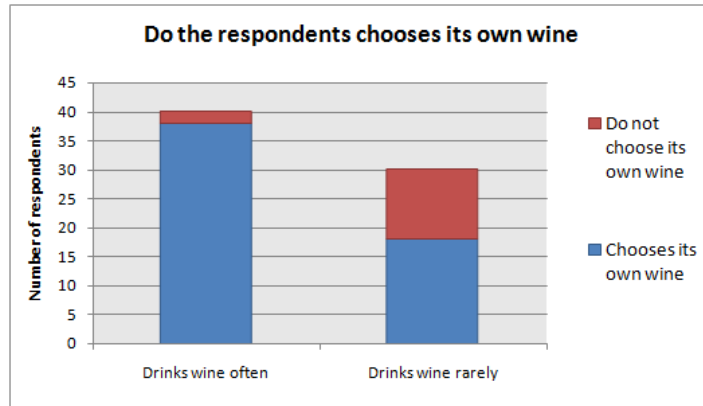


Figure 2.16: Shows how many of the respondents make their own choice of wine

How many like to share their experiences

In figure 2.19 it is seen that half of the respondents like to talk to their friends about good wine experiences. Although there is a big difference between talking to friends about good wines and commit thoughts to a public community, it could indicate that people would be willing to support the user generated data in an app.

How much knowledge about wine does the respondents think they have

On the chart in figure 2.20, respondents clearly show tendency towards a lacking knowledge about wine. Again this supports the need for an app that can help people when buying wine.

When examining how many respondents confront the staff to get help when buying wine it is seen that most respondents never or rarely do that. The numbers can be seen in figure 2.21. Whether or not the respondents do not want the extra information or just think it is

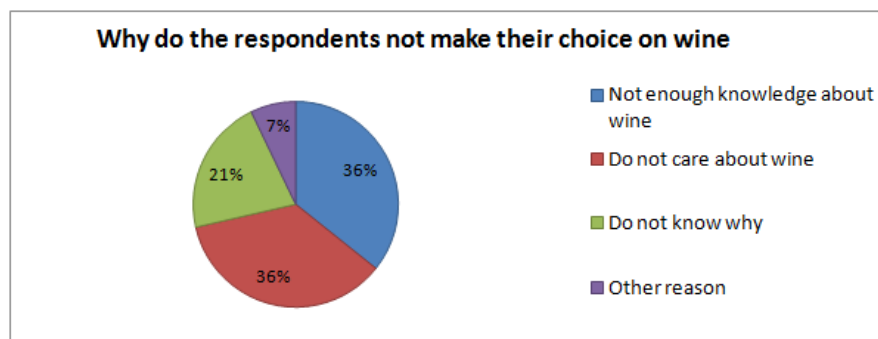


Figure 2.17: Why do some respondents not make their own choice of wine

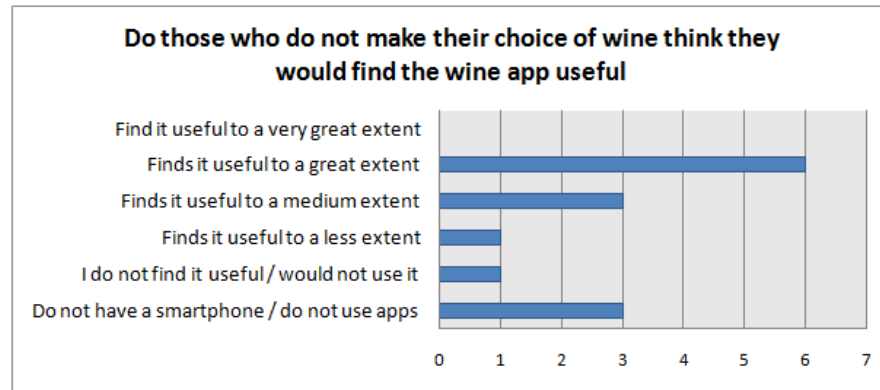


Figure 2.18: *This diagram shows whether the respondents that do not make their own choice of wine, if they would find a wine application useful or not*

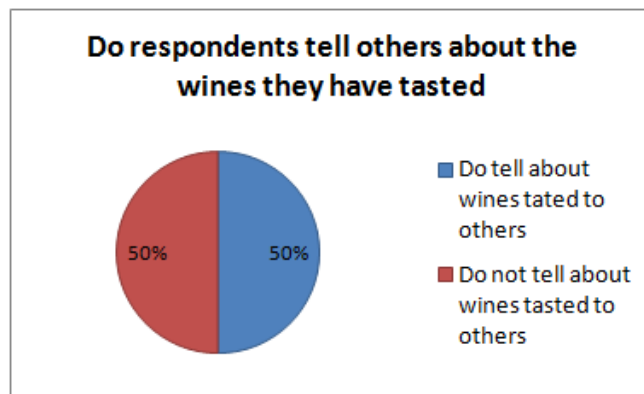


Figure 2.19: *Do the respondents that drink wine, tell others about their experiences*

too cumbersome or awkward was not asked. It is assumed that they would benefit from a wine app, which would give easy access to information.

What factors influence respondents when picking out a specific wine

Another very interesting observation is what guides the respondents to buy a specific wine. The answers can be seen in figure 2.22. It is very interesting to see that recommendation from friends and family is in top three. This support the basis for an app with user generated ratings and reviews. The list is also very interesting when designing which content should be available in the app and what should be presented for the user first.

Is the respondents confident on their choice when they buy a wine

In figure 2.23 it can be seen how confident the respondents are on choosing a good wine when buying one. As the initial hypothesis predicted, there is a large amount of respondents that actually just buy a wine and let luck decide if it is any good or not. Many consequently buy



Figure 2.20: *This diagram shows the respondents own judgment of knowledge on wine*

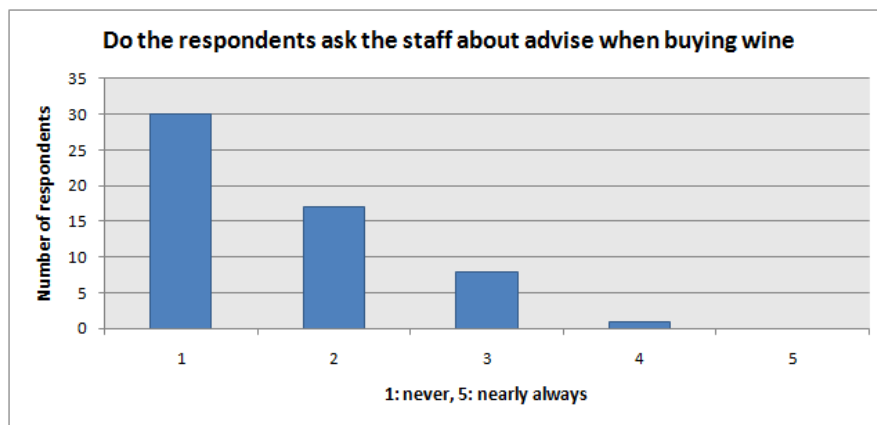


Figure 2.21: *The diagram shows how often the respondents confront the staff to get help to buy the right wine*

wines just by spontaneous intuition. Nevertheless there is also a large amount of respondents that actually feel very confident when buying wine.

Does the respondents find a wine app useful

It is of course very interesting to see if respondents will find a wine application useful. In figure 2.24 it can be seen what the respondents answered to the question. The question addresses those who actually make their own choice on wine, thus it is different from the chart shown in figure 2.18 on page 31. Most of the respondents that use apps only find an app useful to an average degree.

Quick thoughts on second part of the questionnaire

The second part of the questionnaire looks at the consumption pattern regarding smartphone usage and common use of Internet.

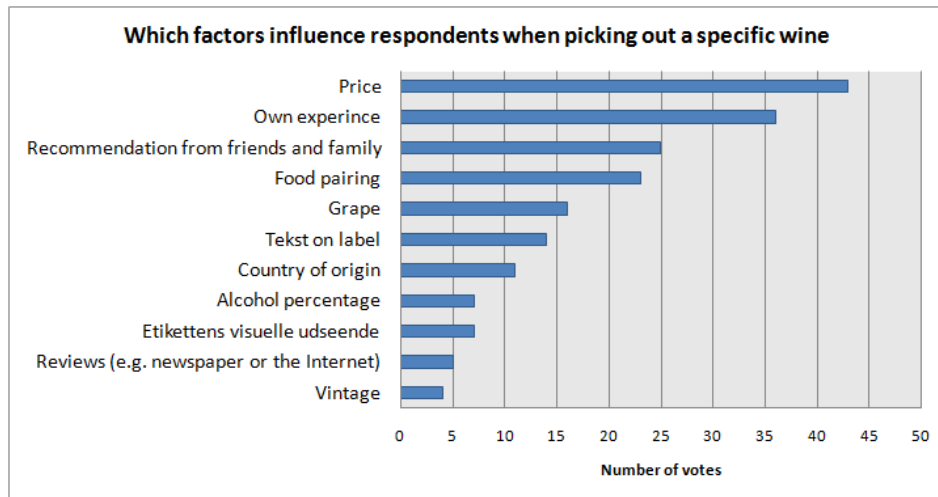


Figure 2.22: *The respondents were asked to pick the three most important criteria when choosing a wine*



Figure 2.23: *This diagram show how confident the respondent feels after having bought a wine. Is their pick on wine normally just a wild guess, or do they know what they choose*

Questions about smartphone usage did not show any significant results that were not expected. It is already known that the smartphone market is huge and that app's are very popular. And as the smartphone market and their users already have been analyzed in section 2.7.3 and section 2.7.4, on page 2.7.3, this part of the results will not be analyzed further.

Likewise the result on internet and social media usage, did not give much useful information. As most of the participants have either gained the link to the questionnaire via mail and Facebook, it came as no surprise that there was a lot that uses social media.

However, in figure 2.25 it is seen how much the respondents think they use the internet to search for information about a product before they buy. As seen, most respondents use the Internet a lot for such purpose. This boost the wine apps potential as it would help in that regard.

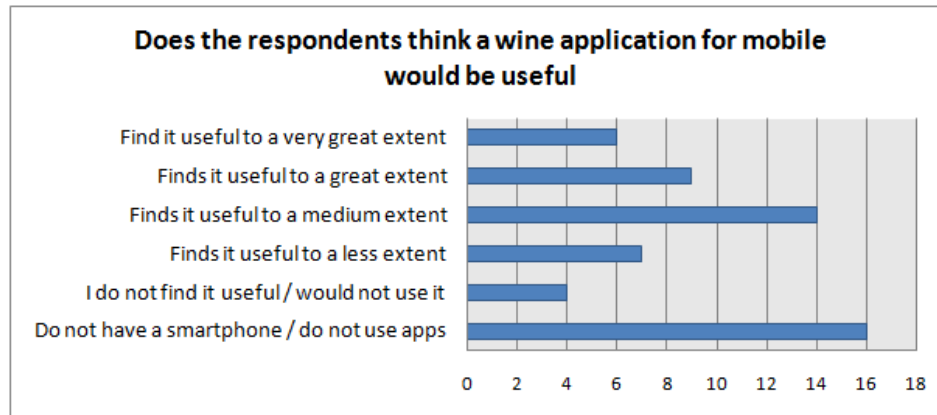


Figure 2.24: The diagram shows whether the respondents would find a wine app useful or not. This only shows the result from those who actually make their own choice of wine, which is different from figure 2.18.

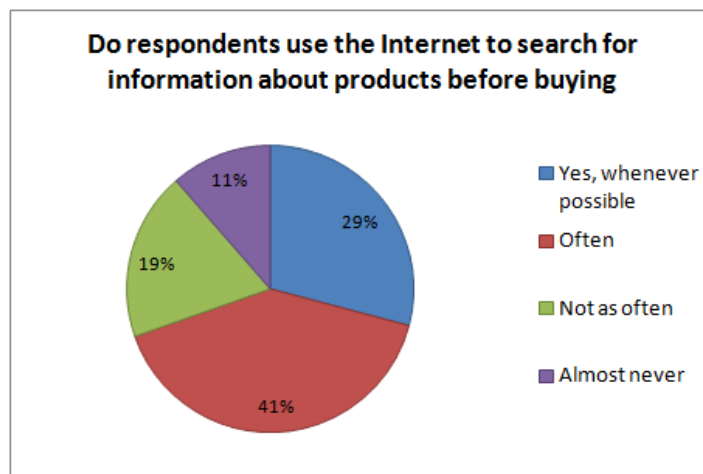


Figure 2.25: The chart shows whether the respondents use the Internet to search for information about products in general beforehand a buy .

2.9.6 Conclusion on User Analysis

All of the hypotheses are completely or partly confirmed.

With the majority reporting that they have nothing to little knowledge about wine it is clearly that many could use an app to help them. That explains why half of the respondents report that they usually just buy a wine on intuition not knowing if it is any good or not.

For the main feature of the app, the respondents also picked recommendation from other people in their top three criteria when choosing wine. This makes a good foundation for a user driven rating system in the app. It was also shown that price and food pairing was very important when choosing wine.

Despite the low knowledge about wine, the respondents also rarely confront the staff for help. This could be explained by it being too troublesome and time consuming in this rushing world.

An app could give the user information at once without beginning interaction with another human with all that implies.

Lastly it is seen that the respondents uses the Internet a lot to find information about products they buy. Also they were decent or slightly above average that the respondents would find the app useful.

Thoughts on User Analysis

It is hard to make a bulletproof questionnaire. This section will briefly look at problems and misleading that could emerge from the constructed questionnaire.

There is the obvious problem that most of the respondent are aged 22-26 year, and that many of them got the questionnaire through Facebook.

Some questions may not have been specific enough. For example when asking whether an app would be useful or not it can be argued that it would always be useful, but that does not mean that one would actually use it. Another example is the question where the respondents are asked whether they tell about the wines they have tasted to others. Instead it could just have been asked if they would contribute to a rating system by actually give ratings.

Lastly it could also have been good to know if the respondents contributed to other user driven communities.

2.10 About Wine

For creating a wine application it is very appropriate to know something about wine. This section will take a peek at what wine is, and what affects the outcome and taste of a wine. The section is based on a wine encyclopedia book called Wine[20], and the Wikipedia article also called Wine[21].

2.10.1 What is wine?

Wine is made from fermented fruit most often grapes, but wine made from apples and cherry are also common. The grape is crushed, and yeast is added to start the fermentation. The yeast consumes the natural sugar of the crushed grapes and converts it into alcohol. When the fermentation has stopped the wine is bottled and corked.

2.10.2 Grapes

The most important part of wine is of course the grapes. There are two distinct types of grapes, namely green and blue. The green grapes is used for white wine where only the juice is used and the blue is mostly used for red wines where skin and seeds is put together with

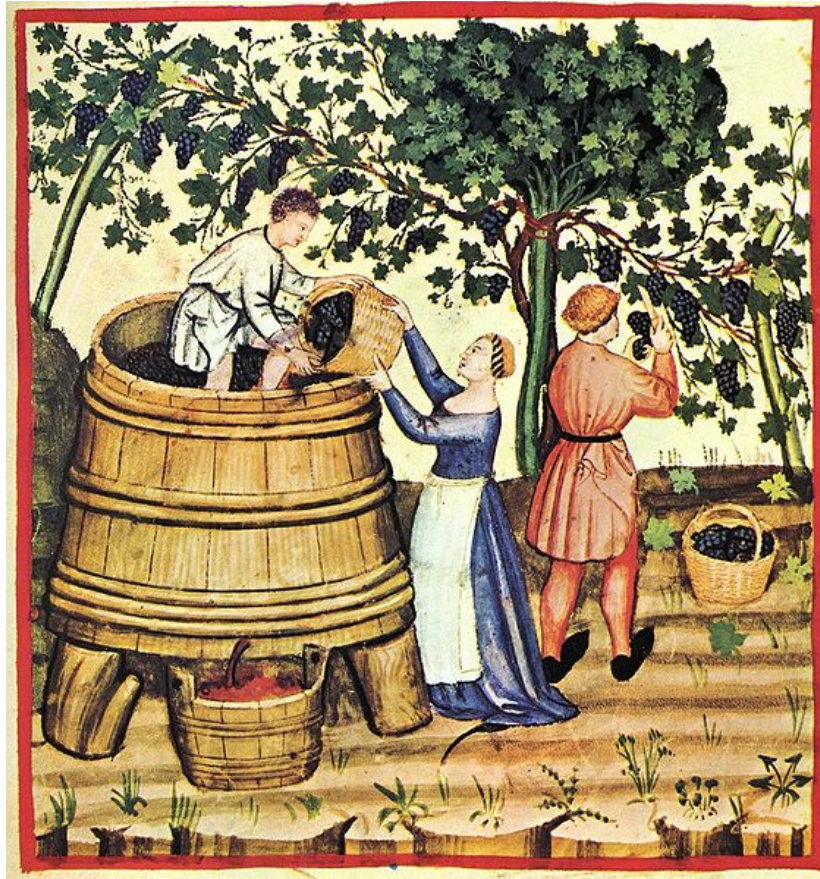


Figure 2.26: *The ripe grape is crushed and the juice is collected. When making red wine the seeds and skin is mixed in with the juice during the fermentation process.[21]*

the juice. The blue grapes can be used for white wine or sparkling wine if the skin is removed before fermentation. Each type of grape consists of different sorts like Cabernet Sauvignon, Merlot, Sauvignon Blanc or Chardonnay. Each sort has its own special characteristics, like leaf shape, color and size, shape of the grapes, vermin resistance, and most importantly the taste. Different sorts of grapes is often mixed after fermentation to create a new taste experience. A popular mix is for example Cabernet Sauvignon and Merlot.

2.10.3 Region

Even though the same sort of grape is planted in different parts of the world the outcome is different. The climate and local soil conditions hugely contribute to the different taste flavors within the same sort of wine.

2.10.4 Winery

The wine farmer or winery is as important as the grapes. Without his skill and care for the vines there would be no wine. The wine farmers can have great impact on the flavor.



Figure 2.27: (a) An example of a vineyard with its long straight lines of vines.[21]. (b) A vine with a series of ripe blue grapes.[21].

For example if the wine farmer carefully grafts the same sort through generation, he can manipulate the outcome in the direction he wishes. Thereby he can differentiate himself from the competition by creating better tasting and more interesting wines. How the grape is stored can also influence the overall taste of the wine. Some let their wines mature in oak caskets before they are bottled. The wine farmers' knowledge combined with local weather conditions hugely effects how the taste will be in the final product.



Figure 2.28: Some wineries are casking their wine in a wine cellar before bottling. The casks usually made of oak affects to affect the taste that typically oak can give.[21]

2.11 Developing Platform

When developing smartphone applications there are several operating systems that can be developed for. Android, iOS, HP webOS, Windows Phone 7, Symbian and Blackberry OS, just to name the most prominent names in the market. Looking at the graph of smartphone operating system market shares in figure 2.29, it clearly states that there are three big players in the world market. The big three is iOS, Android and Blackberry. It is on purpose that Symbian is not regarded as one of the big players despite their actual market share. Nokia has more or less abandoned Symbian in the start of 2011 to focus on Windows Phone 7. To narrow it down further we do not look at Blackberry OS because the phone, and thereby the OS, is not widely used in Denmark.

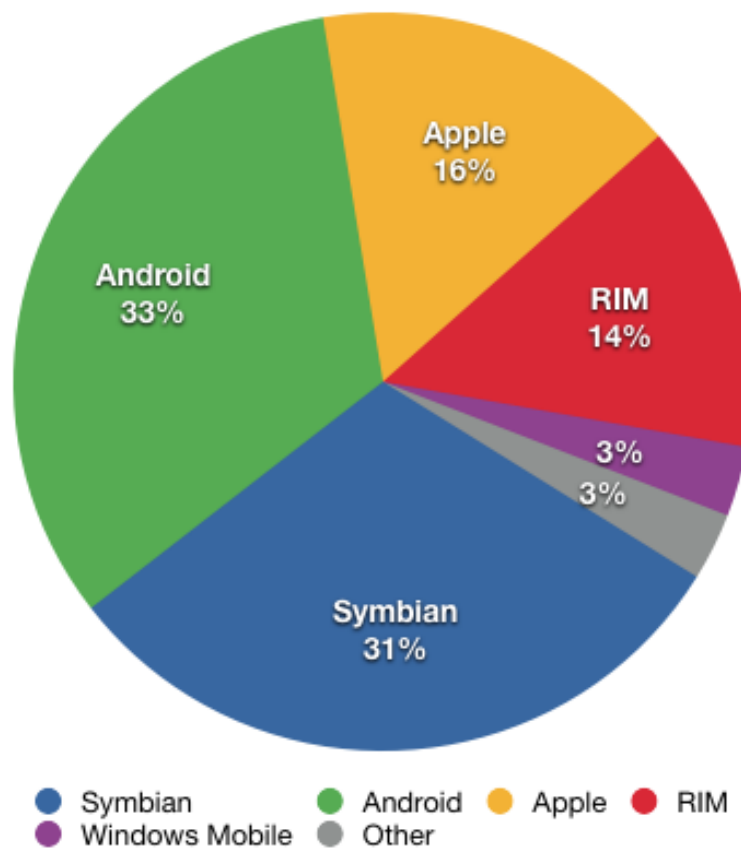


Figure 2.29: *Smartphone sales worldwide from fourth quarter 2010 based on operating system.*[1]

2.11.1 Android

Google's addition to the smartphone market is their operating system called Android. Android is an open source smartphone OS and are currently developed by the Open Handset Alliance, which is a business alliance of 80 firms led by Google [22]. Android has gotten a big market

share since its launch in late 2007, and just continues to grow. In third quarter 2009 Android had a market share on 3,5% which grew to 25,5% in third quarter 2010. Android supports all the usual features expected from a modern smartphone like Internet connection via 3G, MMS, video and music playback, advanced games, GPS, Camera, Apps and much more.

Android is quickly becoming a more and more popular platform for application developers. One is the increasing market share and because Androids application store called Android Market has matured greatly. Android applications are written in the Java programming language with an API on top developed by Google. The documentation on the Android developer site is well written and plentiful. [23][24]



Figure 2.30: *Android logo.*[24]

2.11.2 iOS

With Apples introduction of the iPhone in 2007 they also introduced the iPhone OS. iPhone OS SDK was released to developers in 2008 and was an instant success. The App Store has grown to an enormous 350.000 apps [16]. The iPhone OS was renamed to iOS in 2009. Applications for iOS are written in Objective C or C with the Cocoa API. Like Apple's other products it is a closed system, so making applications for iOS devices requires a Mac computer and the Xcode IDE, that are free to download. Before an application can enter the App Store, a developer license needs to be obtained and it has to go through Apple's App approval process. Lately some of the restrictions in Apple's terms and conditions has been lifted and has made it possible to compile games and programs made in other programs like in Unity 3D or Adobe Flash. iOS is installed on 17% of all smartphones [25][26].

2.11.3 Conclusion

Looking at the different operating systems is it clear that they all are capable of handling rich media applications. What really set them apart are the small details, the hardware carrying the operating system and of course their market share. Both iOS and Android OS have great developer sites and they both have App stores to distribute applications to their users. In our case Android is most appealing primarily because it is free and it is easier to obtain a developer license and release Apps to the Market and we have easier access to Android Hardware.



Figure 2.31: *Apple's iPhone 4*[27]

2.12 How to Recognize a Wine

Recognizing or differentiating between two brands of wine, is for the human eye and brain not a big problem. For a computer vision system it is another matter. This section will look at different computer vision techniques that can be used to identify a wine.

When differentiating between wines the first thing recognized is the shape of the wine bottle itself. It can vary in shape, and a certain shape can indicate what type of wine it is. E.g. a Bordeaux wine bottle has broad shoulders according to tradition. It is a crude method and not all wine bottles follow the traditions, hence the method is rejected.

A more suitable approach is to look at the label. A wine bottle is usually fitted with two labels, one on the front and one on the back. The front label usually consists of some sort of graphical design, combined with text and numbers. The back label has less focus on the graphical elements, but a more explaining text. Besides the graphic and text there is another interesting part on the back label, which is a barcode. Almost every product today is fitted with a unique barcode used for identification. On figure 2.32 a wine with its front and back label can be seen. There are different solutions to identifying a wine using the label.

2.12.1 OCR - Optical Character Recognition

Optical Character Recognition (OCR) is, as the name states, a technique for recognizing characters. In this case it could be useful because there is always text on the label of a wine, e.g. the name of the wine, year and region, which would be enough information to identify a wine. Commercial OCR software has not yet reached a 100% recognition rate, especially not when handling non-standard types. It is fair to say that the letters themselves on a wine label is a part of the graphical expression and therefore take many different forms far from standard types. This diversity in shapes and sizes combined with the possible low contrast image input, OCR seems not able to deliver a promising result in a non-controlled environment. It may result in either faulty look ups, or no matches when looking up a wine in a database.[28][29]



Figure 2.32: An example of a front label (a) and a back label (b).

2.12.2 Template matching

In the industry of machine vision for quality control, registration and inspection in production lines, object recognition plays a major role. The technique behind these machine vision systems is often based on the template matching filtering method where correlation between an image template and a candidate image is used to determine the identity of the candidate image. The template is a sub image that looks exactly like the object which is searched for. This technique is often sufficient in controlled environments where illumination and pose are limited to a minimum, as in the industry. When illumination, rotation, scale and 3d transformations are allowed to vary, this technique gets very computationally inefficient because of the pixel-to-pixel comparison.

2.12.3 Local Descriptor Algorithms

Another approach, instead of the template matching algorithm is to use a local descriptor algorithm to compute features in the candidate image which are partially invariant to illu-

mination and pose. These features are then matched to a set of features in a database to find similarities. Local descriptor algorithms search for significant regions in an image characterized by e.g. edges or corners and tries to describe these regions by feature vectors, called descriptors. These descriptors are high dimensional vectors with dimensionality of e.g. 64 or 128 depending of algorithm used. A detector's repeatability is the most valuable property as it determines its reliability in finding the same significant regions which are distinct under different viewing settings. A local descriptor algorithm can then be said to have two steps: The first one is to detect points of interest to compute local descriptors and the second is to extract these descriptors [30]. A local feature vector in an image is a simple distinctive piece of information describing that region where the feature is located. The feature should be invariant to e.g. illumination, pose, and scale when used for object recognition. The actual matching between descriptors for a candidate image and a database of potential objects are based on a distance between the descriptor vectors e.g. Mahalanobis or Euclidean distance. For object recognition based on a local descriptor algorithm, three steps are involved:

1. Detect interest-points.
2. Describe the found interest-points with descriptors (feature vectors)
3. Match descriptors with database to find a match.

Two popular local descriptor algorithms includes Scale Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF), proposed by [31] and [4] respectively. SIFT and SURF uses fairly different methods for detecting and describing, but SURF proves to win on speed in many circumstances. In a research where SIFT is compared with surf [32], it is shown that SURF is faster than SIFT with same performance, but lack stability to rotation and illumination changes. SIFT is slower and not as good at illumination changes, but invariant to rotation, scale and affine transformations.

The step of matching image descriptors to a database of descriptors is of rather big concern when dealing with high dimensional feature vectors, a problem which has been known as "the curse of dimensionality". A SIFT descriptor for example, is a feature vector of 128 dimensions. A linear search would involve each feature vector to be compared naively to each feature vector in the database. As e.g. Euclidian as the distance measure any correspondences whose distance is lower than a specified threshold are kept. This type of exhaustive search is very slow as it is also seen in [30] where a database of 10 images, each having average 600 descriptors per image, takes 30 seconds to compute an exact match.

2.12.4 Barcodes for recognition

Another, but very simple approach is to facilitate barcodes to identify known objects. This of cause requires barcodes to be printed on every object that is to be identified. Today nearly every product in our supermarkets is labeled with a barcode to identify the product.

The Global Trade Item Number (GTIN) maintained and developed by the standards association GS1 is an entire family of GS1 data structures used to identify products and services worldwide. Probably the most known data structure and globally used in this family, is the

European Article Number (EAN). EAN-13 which is a superset of the original Universal Product Code (UPC) is used worldwide for identifying products. UPC is widely used in the United States and Canada. EAN-13 use 13 characters instead of UPC which uses 12, however the two standards are interchangeable.

The first three characters of an EAN-13 number indicate the country of origin. But as United States and Canada uses UPC codes which have one less character, their country code is the digit 0. When an UPC encoded barcode is read by an EAN-13 standard, the missing digit is read as a 0 and are therefore recognized as United States or Canada origin. EAN has been renamed to International Article Number, but the abbreviation remains the same. [33]

2.12.5 Conclusion

When looking at the pros and cons of the above mentioned technologies the best way to identify a wine would probably be some kind of combination. The scope of the report is to have a finished and polished product and not to research the perfect solution for recognizing wine. Therefore the approach of combining technologies is not attempted. Looking at each of the available technologies the pros and cons will be discussed to establish which candidate will fulfill our need of wine identification.

The use of the wine app will be in an uncontrolled environment with changing light conditions and the images taken with the phone might be noisy or skewed. At the same time a wide variety of fonts used on the wine labels rules out OCR and Template Matching which simply are not robust enough in this case. Local Descriptor Algorithms is very interesting because of its robustness to light changes, scale and rotation, but the slow descriptor matching in a database with several hundred entries does that some kind of database search optimization is needed before it could be just remotely useful. Implementation of such a system would consume a lot of time which might not even be enough. Vulnerability is change in label design. Even a small change in label design would make it impossible to make a clear identification and a new entry is needed in the database.

The last contender is the barcode. Barcodes is printed on most products and each barcode is unique so it is possible to use it to identify wines. The choice of identification technology will be barcodes because of its availability, simplicity and robustness.

2.13 Data management

This section will explore how data can be stored so it can be accessible on a smartphone. It is already known that a vast selection of wine is present, so an optimal data management is necessary.

2.13.1 External Stored Data

Data can be stored on a server on the Internet, and be retrieved with an active Internet connection. A server-client model can be used to utilize this. A server holds information in a database and feed it to the clients that connect to the server, see figure 4.10. Having data on a centralized server makes it easy to maintain and have the same data shared on all the clients that requests the data. When the clients are mobile devices such as smartphones, data bandwidth has to be considered because network connectivity can vary a lot and have a big impact seamless dataflow and wait time. Internet servers can be leased or hosted privately.

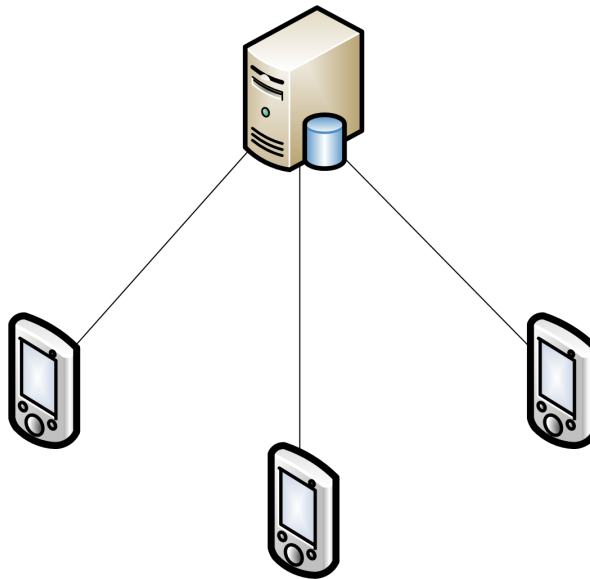


Figure 2.33: *Server-client model. A server with a database serving data for its clients.*

2.13.2 Locally Stored Data

Storing data locally i.e. on the smartphone's storage memory is an option. The biggest strength of that is of course that it always has the data at hand, and that requests to other sources are not necessary. This would greatly reduce, if not eliminate waiting time of retrieving data and showing them. But by having all the data in the smartphone's storage memory, induces use of significant space that is most valuable to a smartphone user. Also this method is decentralized which mean that the data is static unless the database is updated to newer versions. This update could of course be done by a server that issues updates regularly to its clients.

As allude to, the most appropriate method for storing data would be on an external location.

2.14 Problem Statement

A smartphone application that uses the camera to identify a wine with its barcode, enabling the user to do a more qualified choice by displaying relevant information, how would such an application be designed and developed?

2.14.1 Success Criteria

- Be able to quickly search and find a wine using the barcode.
- Enable the user to make a more qualified choice when choosing wine
- The application should feel finished and polished. It should NOT feel like a prototype.
- The finished and polished product should be published to the Android Market.
- More than one external and unknown user should download the app from the market.
- Make use of user generated content
- The app should be easy, fast and pleasant to use
- The app should be approved by test participant as an app they possible would use in their everyday life

Part II

First Iteration

3 Design

The design of the application is of very importance in many aspects. This chapter will describe the different parts of the design that has been established for the development of the application. The user interface design should insure a pleasant and smooth experience for the final user.

This section is the first design section out of two, that will be encountered throughout this report. The section will cover all choices regarding the design process that have been made during the first iteration. The design will be followed with an implementation, a test, and finally a conclusion. This will clear the way for the second iteration which will be described in part III of the report.

The topics that will be covered in this design section is:

- Fundamental concept of the app
- Choices about content and features
- Interface and interaction design
- Design choices about data management

3.1 Fundamental Concept

The first question many people ask themselves when buying a wine is: Is this wine any good? But why not ask someone who has actually tasted the wine you wonder is any good? It is not a plausible option to go around the store and ask people if they have tasted this wine and what they think of it. The smartest place to ask is online and that is where the smartphone comes in handy with its online capabilities.

Using the Image Search

Imagine you are standing in the supermarket's wine department. You want to bring home a good bottle of wine and you are on a budget and cannot afford to spend too much. You see a selection of wine on sale, three bottles for 100 DKR, but cannot decide which one to pick because they all look their best. You are in a hurry and does not have the time for hunting down the local store wine-expert, so what do you do?

You pick up your smartphone, fire up an application, and snap a picture of the wine to initiate a search. The application recognizes the wine and shows you all the relevant information about the wine.



Figure 3.1: *A user despairing looking for the right wine to buy, not having a clue how any of them tastes.*

You do the procedure on the other two wines as well and you quickly realize that two of the wines have a really low user rating, but the last one scanned has a reasonable user rating and a decent expert review. You quickly put three bottles of the best rated in your cart and head towards the register. The application made you able to get the information you needed to make a qualified decision about your wine purchase.

Rate the Wine

After coming home with the wine recommended by the application that got a reasonable rating you get pleasantly surprised by a very good wine. You feel that the rating does not fit the quality of the wine. You grab your phone and quickly browse to your recent searches and find the wine you bought. Now it is possible to give it the rating you think it deserves and thereby contribute to the community of the app.

3.1.1 User Generated Content

As already stated, a great part of the data in the application will be user generated. By using regular people and their responses and comments, it is ensured that the application reflects the opinion and diversities from as many as possible and give the broadest image possible. Furthermore, the salesman is cut out, that maybe sometimes does not know that much about the suggested wine anyway. As a salesman in his store, he has an obligation to promote it as



Figure 3.2: *A user that have found a wine of interest, snaps a picture with his phone, and quickly find out the rating of the wine.*

good as possible, even though he does not like it himself.

Rating

One of the main features is to see the average user rating of a wine. This entails that all users have to be able to submit their personal rating.

Comments

Users should be able to leave private or public comments for a specific wine. It could be possible, anything they want to share about it.

Add to favorites

Users should be able to add specific wines they care about to a personal list called favorites.

Add wine to database

If a wine is not in the database, the user should be able to add it, and thereby contribute to the growing community.

Edit information

Any information about a wine that are missing or incorrect should be editable by a user.

The User Profile

For the user to be able to submit content, a user profile is needed. A user profile is of course not a strictly necessity, just to submit content, but it enables us, as developer, more control over the program. It will hopefully help some of the problem which can arise with user generated content, which were discussed in section 2.8 on page 24.

First of all, when a user needs to spend time on user creation, she is more inclined to be serious in what is submitted. Second, when the user has her name written on every submitting, again she will be more serious on what is submitted. Third, it is possible to track a user that is abusing the system or acting improperly. The user can be banned, and even the phone id can be banned, and all history can be erased.

The problem by having a user profile, is that the user have to spend time on the processing, writing in email, username and password. This might, or will, stop some users in submitting content.

3.2 Functionality Overview

The previous section outlined the fundamental concept of the wine app:

1. You look up a wine.
2. You choose a wine compared to your criteria, like ratings, food pairings, etc.
3. When you have tasted a wine, you rate it.

Though that is the basic of the app, the app should support more functionalities than just that. In this section all of the possible functionalities will be discovered. The section will act like a limited brainstorm, meaning that all possible plausible functions will be addressed. Later the list of functions will be stripped down to fit the need of the app, and fit the time schedule. The functionality overview will make basis for the actual interface design later.

3.2.1 User Scenarios

The functionality of the app is pretty much depicted by the different scenarios the user can encounter. From the market and user analysis, section 2.7 on page 17 and 2.9 on page 24, a lot of user patterns were discovered.

It was stated that many people who drink wine have problems in finding the right wine, and that many people think they lack knowledge about wines in general. It has also been stated that the user often just grab a wine completely on lucky intuition, because there is no way to know any better. In addition, the major factors that influence people on the choice of wine is, price, own experience, recommendation and food pairing.

From these findings, and from own personal experience, and from the users perspective, a list of scenarios can be boiled down and used to visualize the app's functionalities:

- I have a wine, I want to know more about it.
- I have a wine, I wonder if it is any good?
- I have tasted a wine, and it is absolutely wonderful, I would like to recommend it to others.
- I have a wine, I wonder if I can do with something cheaper?
- I had such a great wine last week, what was it called again?
- I wonder what other people might say about this wine?
- I have a wine, which food does match it?
- I have this wonderful recipe, but which wine will match it?
- I have a wine which states it is a "Merlot", what does that mean?
- I have found the perfect wine, I wish i knew if there were any special offers on it in any nearby store?

3.2.2 List of Functionality

From the above list of scenarios, and from the decisions and the discoveries made in the problem analysis, it is possible to make a brainstorming of functionalities that could be possible in the wine app. The list of functionalities is tried illustrated in figure 3.3. With a large list of functionalities for the app, the actual design process can begin.

3.3 User Interface Design Theory

From the above section, an outline of all the possible functions of the app was made. These will be stripped down and defined precisely, to exactly define the interface design. To define the interface, first knowledge about interface design must be made.

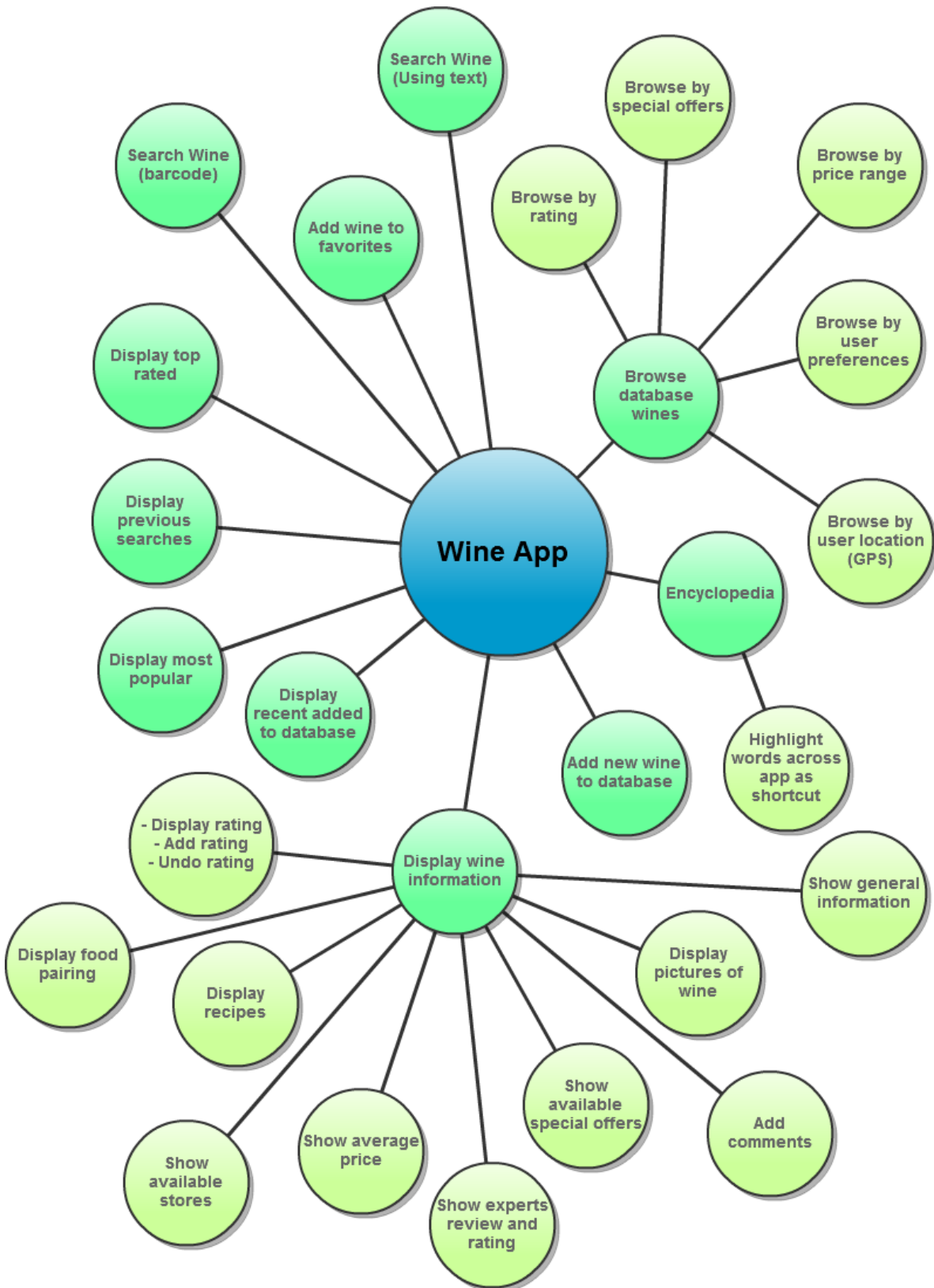


Figure 3.3: The diagram outlines all the possible functionalities thought of for the wine app.

There have been written endless texts about how to do a good user interface design appropriate for different applications and uses. When designing for smartphones there are many similarities with that of designing computer programs. On the other hand there are also many other important things that differ from normal software interface design.

The design of the wine application will be based on theories and rules showed by Google and Apple. It is obvious to make use of these companies experience and knowledge about interface design and developing for smartphones. Apple is one of the trend setters in modern design and usability, and both Apple and Google have a huge role in the development of applications to smartphones.

For that reason, most of the section will be based on Apples human interface guidelines for iOS [34], Google's Android development guide [35], and two conferences from Google I/O 2010; one conference about how to create positive user experiences [36] and one conference about user interface design patterns for Android [37] The rules and theories in the section may not always fit into the whole range of smartphones, as it depends on what is defined as a smartphone and which phone is looked at.

3.3.1 Smartphone Characteristics

It is necessary to have an insight into what differ from normal interface design development. Knowledge about this ensures an optimal interface that is carefully carved for the purpose of use.

First the screen size on smartphones is very compact compared to that of a normal desktop or laptop computer, and has an impact on different things. For example, content showed on screen need to be very specific, as there is no room for everything. On screen user help is normally minimal, so an interface must depict a clear language for the user to understand.

Also, there are no windows as seen in for example Microsoft Windows. Apps have a single window, and people interact with one app at a time. It can be a little tricky to switch back and forth between apps. It may be useful to let an app enter a suspended state when they transition to the background. When people restart a suspended app, it can on most mobile operating systems, instantly resume running from the point where it quit, without having to reload its user interface. If the operating system supports multitasking, it is possible to let the app run in the background while users run another app in the foreground.

As mobile devices refer to devices that are small, so are their hardware capabilities. Memory is limited and it is very important to have both running and storage memory usage at a minimum. The computation power, along with graphic and 3d capabilities, on mobile devices is of course also limited compared to a desktop device. Although it is something that should be considered while developing apps, the programming API built in limitations, helps a lot in that course.

3.3.2 Principle of User Interface Design

“A great user interface follows human interface design principles that are based on the way people - users - think and work, not on the capabilities of the device. A user interface that is unattractive, convoluted, or illogical can make even a great application seem like a chore to use. But a beautiful, intuitive, compelling user interface enhances an application’s functionality and inspires a positive emotional attachment in users.” [34]

The phrase from Apple is an excellent description of what a good interface design is all about, but it can be hard to accomplish and understand in practice. The most important categories of user interface design principles, depicted by Apple and Google, and relevant for this design document, are as followed:

- Clarity
- Aesthetic Integrity
- Consistency
- Direct Manipulation
- Feedback
- Metaphors
- User Control

Clarity

A common view is that good user interface should be simple. While this might be true in some circumstances, simple has some negative implications. For example, lack of power; a good user interface should have an appropriate set of features, options and information that corresponds to the given task, and should not cut down on this just to make it simpler. A better expression over simple is to use the word “clear”, which means that the interface should be clear to use. It should not confuse its user and should not lead to frustration. In general not place the user where she does not know how to use a product.

At the same time, while keeping the user interface clear and have appropriate number of features, overloading on features or having too many complex interactions are bad and should be avoided.

By focusing on clarity over simplicity products can be developed that are both straightforward to use and powerful at the same time.

Aesthetic Integrity

“Aesthetic integrity is not a measure of how beautiful an application is. It’s a measure of how well the appearance of the app integrates with its function.” [34].

What it suggests, is that the amount of decorative elements should be depicted by the programs function and task.

A serious and productive application, as for example a mail program or a news program, should tone down on artistic elements and keep them in the background to make room for the actual purpose of the app, which is being productive.

Contrary, an immersive application, such as a game, should have much more focus on having a beautiful appearance.

This engages the user and makes the application more fun and pleasant to use.

Consistency

By making applications consistent with other applications and device standards, users can transfer their knowledge and skills from previously used programs to new programs. In this way the user uses less time on learning new programs and can get started right away.

By spending fewer resources on how to use a program, more attention is available for the actual purpose of the program.

It is however important to notice that, applications should not be a simple copy of each other. Instead it is the interactions that should be consistent taking advantages of standards.

There are many choices available today, and products should focus on engaging their users by keeping the interactions consistent but varying the treatment.

Direct Manipulation

Direct manipulation is all about making the users engaged in their product and the task they are performing by having a more direct contact with the product.

For example, when controlling an object, instead of having buttons and separate control objects, let the user control the object directly by touching it. This is of course only possible on smartphones with touch screens.

By having direct contact with the objects the user is manipulating, they begin to be more engaged and they will easier understand the results of their actions.

There have already been shown various scenarios where this principle has be brought to practice. For example when zooming on an Apple iPhone, the user just uses the pinch gesture

to directly expand or contract an area of content, instead of having zoom controls at the side of the screen for example.

Feedback

“Feedback acknowledges people’s actions and assures them that processing is occurring. People expect immediate feedback when they operate a control, and they appreciate status updates during lengthy operations.” [34].

When creating feedback it can take the shape of many effects. Displaying visual feedback when controlling text or buttons by for example highlighting, is one good feedback that tells the user that the product recognizes input.

Vibrations in one way or another also make for a very strong feedback, as it triggers more than just the visual senses.

In the same way can sound in some cases be used successfully as feedback to support other than the visual senses. But sound should not be used as primary feedback because users might have their phone on silence or be in places where they cannot hear them. Also sounds can be annoying if used too much.

In addition all interactive user interface elements should have at least 4 states, as seen in figure 3.4.



Figure 3.4: As an example of good feedback, all interactive interface design elements should have at least 4 states. [34]

Metaphors

When users want to do a specific action, they want to be able to figure out how to do it fast. In addition when presented for different actions or controls the user should be able to very quickly grasp what they do. If this fails, the user gets frustrated and irritated, and this makes a bad user experience.

By using real world actions or objects as metaphors for similar actions or controls in a product the user will be able to identify its function much easier. Metaphors can both be used directly, such as an on/off switch to turn something on and off, or indirectly, such as using a house for communicating an action to go to the home or main screen.

User Control

The app should not take control over functions and actions. The user should always feel that they are controlling the app. For instance an app should not be the one to make decisions about actions, but could of course warn about dangerous consequences of an action.

The user should always be able to cancel an operation before it begins and also be able to cancel while that operation is underway. There should also be a chance of confirmation when doing a potentially destructive action.

3.3.3 Positive User Experience

Creating a positive user experience makes the user pleased and comfortable and makes them more engaged in a product. What is meant with a good experience is that the user experience positive emotions while interacting with a product.

Among other things a positive user experience originates from a good interface design. There have already been discussed some of the common principles in the previously section that involves usability and interaction design.

These principles make the core of the experience that people have, using a product. But there are many other things that affect the user experience that do not fall into this category of design. In this section more ways to create positive user experience will be described.

Fast Application

This is both a technical issue and a design issue, but is nevertheless very important to retain positive user experience.

An application that is not fast, lets the user wait or complicates the process through a task, frustrates the user.

The technical aspect of fast is latency. An app has to load and respond fast compared to its respective task, meaning that it should not take longer to respond than the user expect it should. A simple click on an item should respond instant while for example a music indexing is allowed to use a longer time.

It is important to notice that speed matters more to mainstream users because they do not know how hard it is for the application to be fast. They just want it to work. Developers are more sympathetic to how difficult it is to make a fast application.

The app should just feel snappy in general. If there are a lot of animation and graphic or other heavy elements that can be fragmented, the most important stuff can be loaded first.

The design aspect of fast is to provide shortcuts in the application. It may sound trivial, but there are more shortcuts that can be provided than one might expect. It is just a matter of being creative. Some examples are:

- If it is known that the user is going to use a text field it should be automatically focused and the keypad should appear.
- Remembering recent input. This can be relevant in many situations like recent searches, recent selections etc.
- Make as much as possible of an element clickable. This is very important when using touch screens. It could be when using radio buttons, instead of only making the radio clickable, make the whole text associated with it clickable.

Offer a good way out

Always offer the user a way to cancel a specific operation.

When terminating an operation for whatever reason, provide an explanation why it happened, what implication are involved, and what the user can do to make it good again.

When provide a cancel action, do not just label it "cancel", but inform the user of what he is going to cancel. For example "Cancel and don't save?"

When possible offer a way to set an operation on hold to be resumed later.

Focus on the primary task

By focusing on the primary task of the app, features can be cut down to what is needed and nothing more. This increases clarity of the app and makes it both satisfying and enjoyable to use.

It can be beneficial to create an application definition that clearly states what the app should be able to do and what features are needed. Afterwards every screen should be analyzed to see if any information or feature is non-critical. If non-critical, it should be considered if it is necessary to stay there.

When deciding what to have on each screen, use as much of screen as possible what people really care about - content. And specifically their content like music and status updates etc.

Use terminology that users understand

When using text in the app it should be written in a language that the user understands. As a developer many technical terms might be used, but the user may not be able to understand this.

Minimize user input

On a smartphone it can be irritating and time consuming when having to feed the app with input. As such, user input should be kept at a minimum.

It is also possible to make it easier making input by being creative. For example by offering a list of selectable options instead of having the user to type in text, can be very pleasant for a user.

3.4 Application Features

As stated in the interface design theories in section 3.3, it is very important to create a specific application feature list.

By creating a complete feature list it enables the developer to know exactly what the app should be capable of, and what not. This is obvious useful when designing the interface.

Second, it hopefully will increase clarity in the app by minimizing functions that are not needed. This should create an app that is straightforward to use and still powerful.

As learned in the design theory, it is important to focus on the main task. This means that only features that are important should be contained on the feature list.

As a result the development of the list has undergone several iterations, starting with a large brainstorm of any possible feature, each iteration stripping down on features until a final list has been created.

The final list of features can be seen here. The list has been developed on basis of the fundamental concept and functionalities in section 3.2 on page 52. The final features is illustrated in figure 3.5.

In the upcoming section each feature will be described.

3.4.1 Core App Features

Here the most important features will be described. Together they will constitute most of the main screen of the app. There are two absolute major features in the app that should be easily accessible all the time throughout the app. These features make the core functionality and establishes livelihood of the app. The two major features are:

- Search for a wine
- Rate a wine

A third major feature, which emerge as direct effect from the other major features is:

- Wine information



Figure 3.5: *The diagram outlines what specific features that are going to be implemented in the first design.*

Search for a Wine

The search feature is why the user would want to use the app in the first place. There are two methods the user can search for a wine. Firstly, by using the built-in camera to take a picture of a wine and let the app recognize it. Secondly, if the camera is not a solution or the user do not want to use the camera, the user should have the opportunity to do a text string input and by that search in an ordinary manner.

The search feature is called scan in the app because the word "scan" is a more expressive word for the action, the user does when taking a picture of a wine to be recognized. It resembles the task of scanning a barcode.

Rate a Wine

The rate functionality is essential for the app to grow and develop and maintain user awareness. If the wines in the database do not get rated by users, much of the original functionality is gone. Therefore it is highly crucial that the user have easy access to do a rating of a wine.

Wine Information

The wine information is displayed when a user have found a specific wine through the search. From the functionality overview, section 3.2, a lot of different information about a wine should be available. For the first design however, there will be a focus on the main purpose of the app. Hence the wine information will only consist of the following:

- Picture of the wine
- Basic information
- Rating
- Add/remove from favorites

3.4.2 Other App Main Features

The last of the main features, not as crucial as the first three are:

1. Show recent searches
2. Show user favorites
3. Show most popular wines
4. Database search
5. Edit profile
6. Encyclopedia

Show Recent Searches

By allowing the user to explore recent searches and lookups, the app accommodate fastness of the app and bring shortcuts to the user. By providing a recent tab, the user does not have to find the appropriate bottle of wine to look it up or to rate it, if she already made the same search before. The recent tab can also be used if a user remembers a good wine and want to find it again, but have not added it to her favorites.

Show User Favorites

This is again about creating shortcuts and making the user experience fast. The user should be able to add a specific wine to her favorites so she can easily access it at a later point.

Show Most Popular Wines

Users are always curious what other users think. In addition, by allowing the user to see popular wines such as, top rated, most viewed and recently added, the user get the feeling that she knows the market, even without any real research.

Database Search

This feature is provided to give the user an opportunity to just search for a wine without having a specific wine that she is looking for. This can be used for just browsing randomly through the database.

The main purpose of this feature however, is to help the user find an appropriate wine when in doubt of what to buy. The user should be able to use different filters to cut down the search. This could be a price range, rating range, and so on.

Edit Profile

A users profile is the users' identity in the specific content. Therefore it should be easy to edit. Maybe the user has gotten a new email address, wants to edit what news she subscribes to, or the user simply want to delete her account. Anyway this should be fast and easy to reach.

Encyclopedia

When having an application that tells about different wines, it is obvious to provide a reference to different terms and uses. The encyclopedia should also be fast and reachable, and therefore words that are within the encyclopedia, across the app should be highlighted to act as a hyperlink, to an entry in the encyclopedia.

3.5 Designing for Android

Before designing and realizing the described concept, it should be stated what is available when developing for android. In figure 3.6, a HTC Desire is shown. The Desire runs android and is the phone that will be used for testing while developing the wine application.



Figure 3.6: A *HTC Desire* smartphone[38].

3.5.1 Native Navigation

All Android smartphones are built with five standard buttons for navigation. In figure 3.7 the five buttons can be seen placed at the bottom of the phone as physical buttons. The buttons described from left to right, allow the following navigation:

- **Home button:** Takes the user to the home desktop screen. All running application is minimized and enters a suspended state.
- **Menu button:** If an application supports it, a context menu is displayed. The menu can be customized as needed when developing the app.
- **Track pad and enter button:** With the track pad the user can select all focusable views in the app by sliding his finger. If pressed it corresponds to make a touch click.
- **Back button:** All navigation in the Android OS is recorded and placed in a hierarchy. When pressing the back button, the user returns to the previous screen, closing the current screen.
- **Search button:** This can be predefined to do any possible action, but the icon of course allude to, that it is used to do some sort of search.

As regards design, there are functionalities that are less important compared to regular interface design. For example the need of an explicit exit button is negated. The user, uses the home button or the back button to exit the application.



Figure 3.7: A closeup of the HTC Desire. At bottom is placed five buttons used for navigation in Android. This five buttons is standard on Android mobile phones[39].

Likewise, an explicit back button is not needed because of the physical back button, always present on Android phones.

3.6 Design Patterns

User interface design patterns do not differ much in principle from software design patterns. Design patterns are used as a general solution to a recurring problem. It is not necessary the perfect solution, but it is a good, solid, reliable and broadly applicable solution to the problem. The solution does not have to be executed down to the last detail, but can be modified to satisfy the specific need. Design patterns are acknowledged by experts for common use, and are therefore safe to use.

This section will look at different design patterns useful for the app development. The design patterns are solely patterns described by Google at the Google I/O 2010 in the conference on Android UI Design Patterns [37], and are intended to work on the Google Android system.

The Dashboard Design Pattern

The Dashboard can be used as a main menu for an app. It should give a good overview and quick introduction of the app and try to lay out the key functionalities. It addresses what the user can with the app and if there is any news to the app. The Dashboard can be seen in figure 3.8.

The Dashboard is full screen and should give the user a kind of clear waypoints into the particular subtasks that the app can perform for them. It can be broken up in some different ways. For example by features, different entry points to posting or searching, or more content focused by category for example, or it could display different accounts.

The Dashboard can be split into two different areas; the top part is fixed and reliable where

the user knows where things are. The bottom can then be used for status updates of what is new to the user. The news could be content specific for the user and the app, or it could be an update on new feature or version.

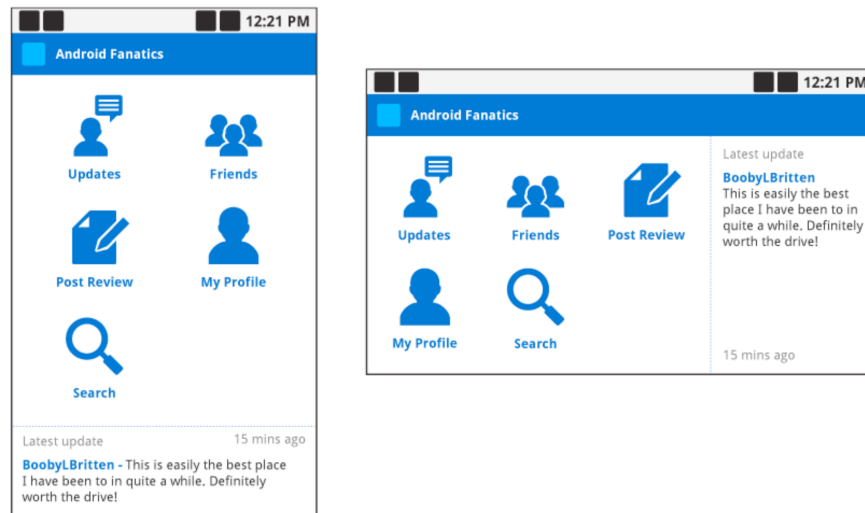


Figure 3.8: Example of a Dashboard from the Twitter app on Google Android. Top screen shows the main features of the program, while the bottom gives the user updates on content and application updates. [37]

The ActionBar Design Pattern

The ActionBar is a design element that should be accessible throughout the most of the application. It is used for actions that are common across the app as for example a search, refresh or compose. It should be placed in the top of the screen and should replace the title bar not to take up any extra space of the app. The ActionBar can be seen in figure 3.9.

The left of the ActionBar still contains the application name, or the current location in the app, and an icon. The right side is a set of actions. It acts like a toolbar, but should not contain every feature of the app and is not a replacement of the normal menu, i.e. the menu that is displayed when the user presses the dedicated menu button on the device.

The number of actions should be of three or less. The ActionBar should be consistent throughout the entire app. It should not be used for contextual actions, which is what the next design pattern should be used for, the Quick Action.

The Quick Action design pattern

The Quick Actions are used to make fast interaction with different objects in an app. Examples could be, interacting with a contact or interacting with a post or email.

The Quick Action functions as a kind of pop up from an element with the most useful interactions and should easily show what a user can do with a specific thing. By using a pop

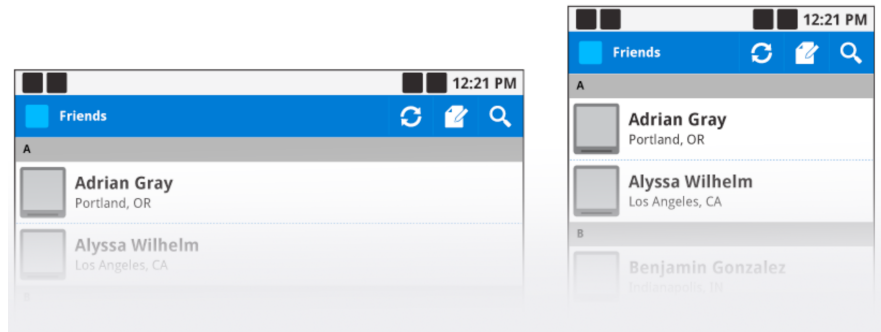


Figure 3.9: An example of the ActionBar from the Twitter app for Google Android. The ActionBar is the bar covered in blue at the top. To the left the application name, or the application status is shown alongside the application icon. To the right the actual actions are located. The actions here are refresh, compose, and search. [37]

up instead of a new screen to show interaction option, you prevent the user from been drawn away from the actual content.

The user can easily get an overview of his options and continue with the original task if he pleases. The Quick Actions can be seen in figure 3.10.

There should be a distinct target on the screen that the actions are emerging from, and it should be a visual target like a more button or an icon to show the user that here is something more to do.

The Quick Actions should be minimally disruptive to the screen context, and should kind of pop up above or below the item they are acting on. The actions themselves should be pretty straightforward, and the actual function could be communicated with an icon or just shown by a word or two of labels.

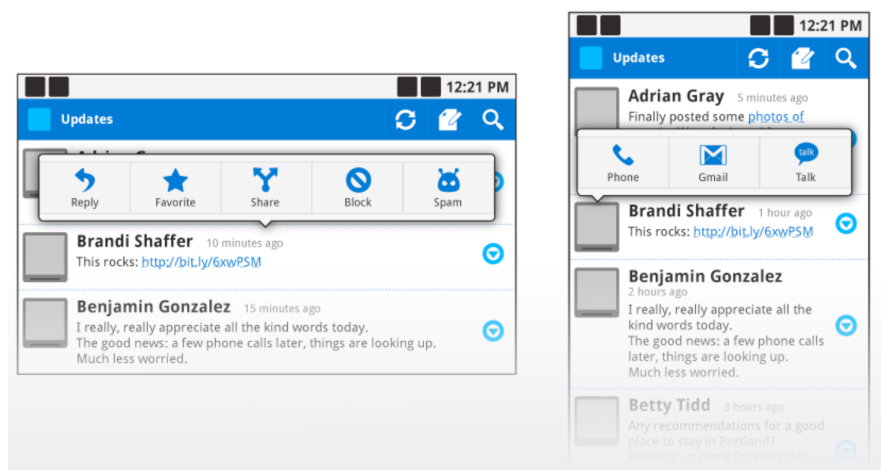


Figure 3.10: An example of the Quick Action design pattern used in the twitter app for Google Android. It is a popup that makes for quick and intuitive interaction with a specific element. [37]

3.7 Low Fidelity Design

The low fidelity design is the design of how it is thought the app is going to look like before actually implementing it. It is not yet known if the design will work probably, and testing will have to be done before settling with the final design. In addition the whole functionality is not yet completely defined, and the low fidelity design therefore may lack details compared to a final design. To develop the low fidelity design, the theory of interface design, section 3.3 will be used, and design patterns explored in previous section, will be used. This will prove consistency through the app, and it is ensured that many of the design choices will work and function in practice. To further increase consistency with the Android platform we will also look at other successful apps like the Facebook app [40], IMDB app [41], and the Gmail app [42].

3.7.1 Using the Dashboard Design Pattern for the Main Screen

For the main screen of the wine app the Dashboard design pattern will be used. For a productivity application like this, the Dashboard works great, as it ensures seamless access to the programs main features. By providing an overview, the Dashboard makes it very easy and straightforward to get going with the app.

The design pattern suggests a six-pack pattern along with an update section. What is meant is that the programs 6 utmost main features are shown at the top, and a section for showing content update or other updates, are placed at the bottom, see section 3.6 for a description of the dashboard.

The wine app will not have much use of an update section at the bottom. Program updates are supported by the Android operating system that informs the user of any software updates. Content updates are minimal at current time. But in the future there might be more content updates.

As it does not seem necessary for an update section it is chosen to have eight main features shown instead of six.

Below in figure 3.11 a sketch can be seen of the wine apps main screen using the Dashboard design pattern. The name "Winelicious" has been used as a dummy name for the application name. The button named "Catalog" refers to the database search functionality, described in the application feature section 3.4 on page 61, while the button named "Scan" refers to the search functionality. The other buttons and names should be straightforward to understand. Also the sketch has been modified to fit the need of the report.

3.7.2 Using the Actionbar Design Pattern for the Wine App

As described the Actionbar replaces the application bar with shortcuts to the most common features of the app. As described there are two very important main features that should be placed in the Actionbar, which are the search and the rate functionality. Below in figure 3.12 the Actionbar can be seen.

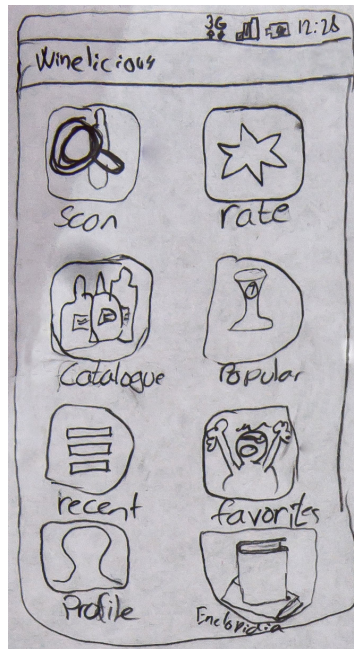


Figure 3.11: Sketch of the wine app main screen using the Dashboard design pattern. The Dashboard consists of the eight most important features. The name "WineLicious" is used as a dummy name for the application name. "Catalog" refers to the database search functionality. "Scan" refers to the search functionality.

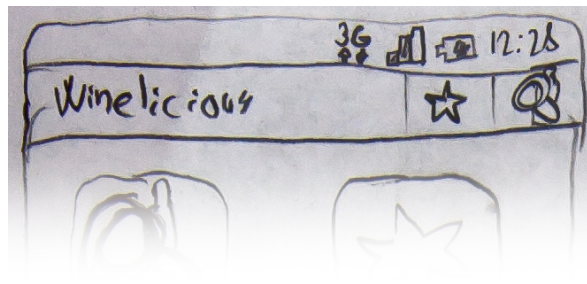


Figure 3.12: The ActionBar replaces the application title bar and provide shortcut to the most common features of the program which is the rate and the search functionality.

3.7.3 Common Screen Layout

Every item on the main screen opens a new screen to support the appropriate feature. To make the app consistent throughout the whole app, the basic design of every screen will follow the same convention.

To increase clarity and make the user interface as obvious as possible it is chosen to follow a very common tab-based design which is commonly used in Android applications and in the Android operating system itself.

In figure 3.13, the common screen layout can be seen. The layout operates with a number of different areas which will be described below. Refer to the image to see where each area is

located.

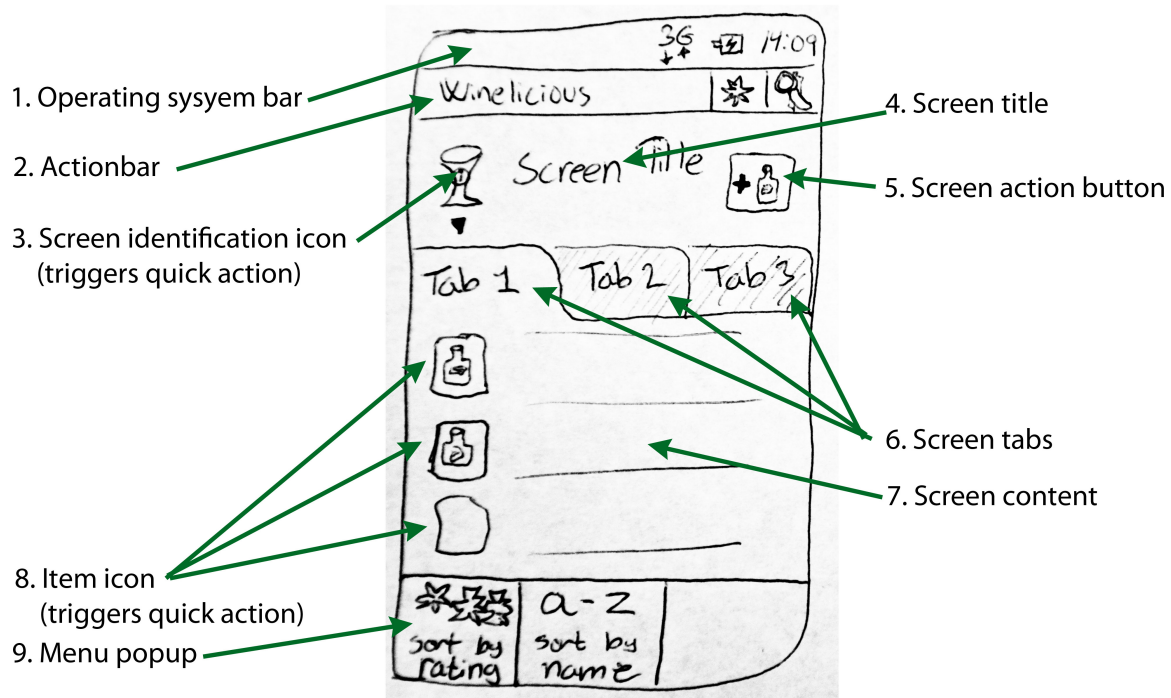


Figure 3.13: All sub screens from the main screen follow a common layout as shown in this figure.

1. **Operating system bar.** This bar is standard in the Android operating system and is not part of the program.
2. **Actionbar.** Contains the application name and the two most common app features, which are rating and searching.
3. **Screen identification icon.** This icon will be the same icon as used for the same screen on the main screen. It will enable the user to quickly know where she is located in the app. A small triangle beneath the icon indicates that it can be interacted with. The triangle is more or less commonly used in other Android application. When the screen identification icon is tabbed the Quick Action design pattern is used to make shortcuts to four or five of the programs other main features. This is demonstrated in figure 3.14 and described in detail below, in section 3.7.3.
4. **Screen title.** In text, shows which part of the app the user is located in.
5. **Screen action button.** If any common action is associated to the screen there is room for it here.
6. **Screen tabs.** Different tabs are used to present different aspects of the screens functionality.
7. **Screen content.** Here the appropriate content is located. It could be a list, a search field, or something else.

8. **Item icon.** If there are any icons associated with the content they will be tab-able. When an icon is tabbed the Quick Action design pattern is again used to provide shortcuts to different interaction possibilities. This is demonstrated in figure 3.14 and described in detail below, in section 3.7.3.
9. **Menu popup.** When the user presses the dedicated menu button on the phone, the menu popup pops up. It usually contains a home button or options to do sorting of a list in different ways.

Using the Quick Action in the App

Quick actions become accessible when tabbing an icon or other designated parts of the screen in the app. The screen identification icon, described in figure 3.13, is available through the entire app but the main screen, and shows the five most or four most important screen locations of the app. The number is dependent on screen space and the exact number will have to be tested.

Which quick actions that is available when tabbing an icon depends on the specific item. Most possible it is a wine in a list that is tabbed, and quick actions could open up the actual wine description, make a quick rating, remove a rating, add to favorites etc.

Both uses of quick actions can be seen in figure 3.14.

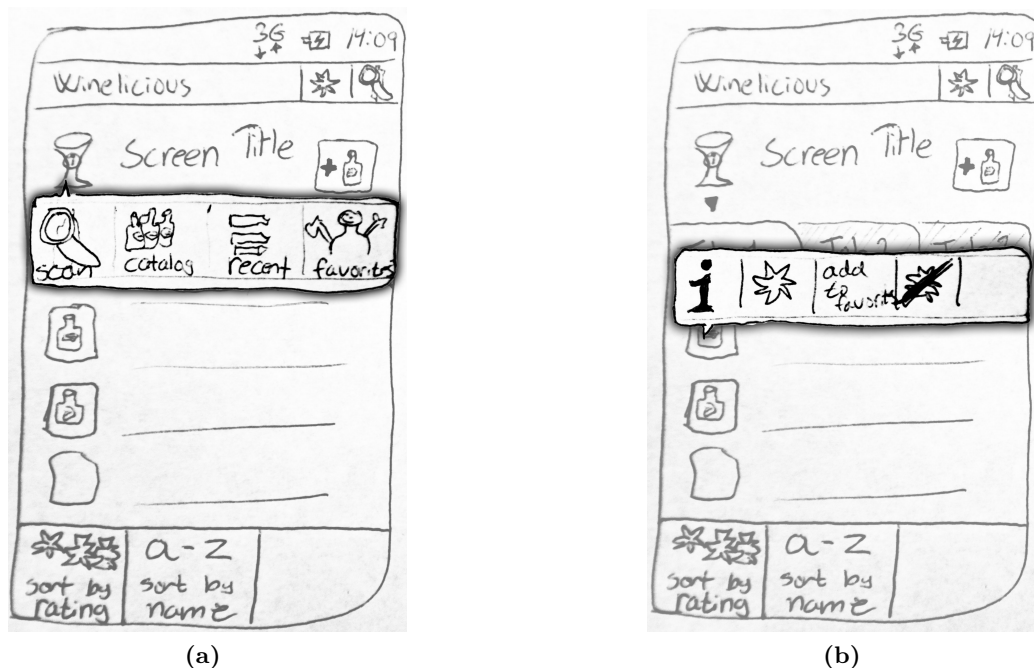


Figure 3.14: Both left and right show how the Quick Action design pattern used in the app. The quick actions are triggered either when tabbing the screen identification icon or a content item icon. The quick actions enables the user to quickly change screen location or get access to most common item interaction.

3.7.4 Overview and Application Flow

Before tossing into presenting all the different screens that will be available in the app, the final result will be presented by two different flowcharts. Hopefully it will be easier to understand the different screens and the program flow, when it is being described in the next couple of sections. It must be stated that some of the content of the flowcharts might be confusing to the reader, because proper description follow in the following sections of the low fidelity design.

The first flowchart in figure 3.15 on the next page, display the screen overview of the app. The main screen, the dashboard, is placed in the middle and all its sub screens emerge from it.

The second flowchart in figure 3.16 on page 75, show the actual application flow and what actions the user can do, and how the app responds to the users actions.

3.7.5 The Different Screens in the App

It is time to look at the different screens and how they will be implemented.

For better convenience while reading this part, have a look at the application overview and flow described in previous section, 3.7.4.

The Search Screen

As described the search screen needs both an image search and a text search. This calls for two tabs to support this. The search screen can be seen in figure 3.17 on page 76.

In the first tab the camera is activated so the user is able to see what is being taking picture of. In the second tab the user is represented with a text field as seen in ordinary searches. When the user has made a search she is taken to the search result screen also called the wine information screen.

The Rating Screen

As described the rating of wine is of utmost importance for the existence of the program. Rating of wine can be done in more than one way throughout the app. It does not necessarily have to be done through the rating screen that is accessible from the main screen. Whenever a wine is displayed throughout the app, in one way or another, it is possible to make a rating for the specific wine either through quick actions or just directly through the wine information screen.

But the rating screen is made to make it even easier to do a rating. It is important to have this screen to prevent the user from having to go through a lengthy path to find a way to rate

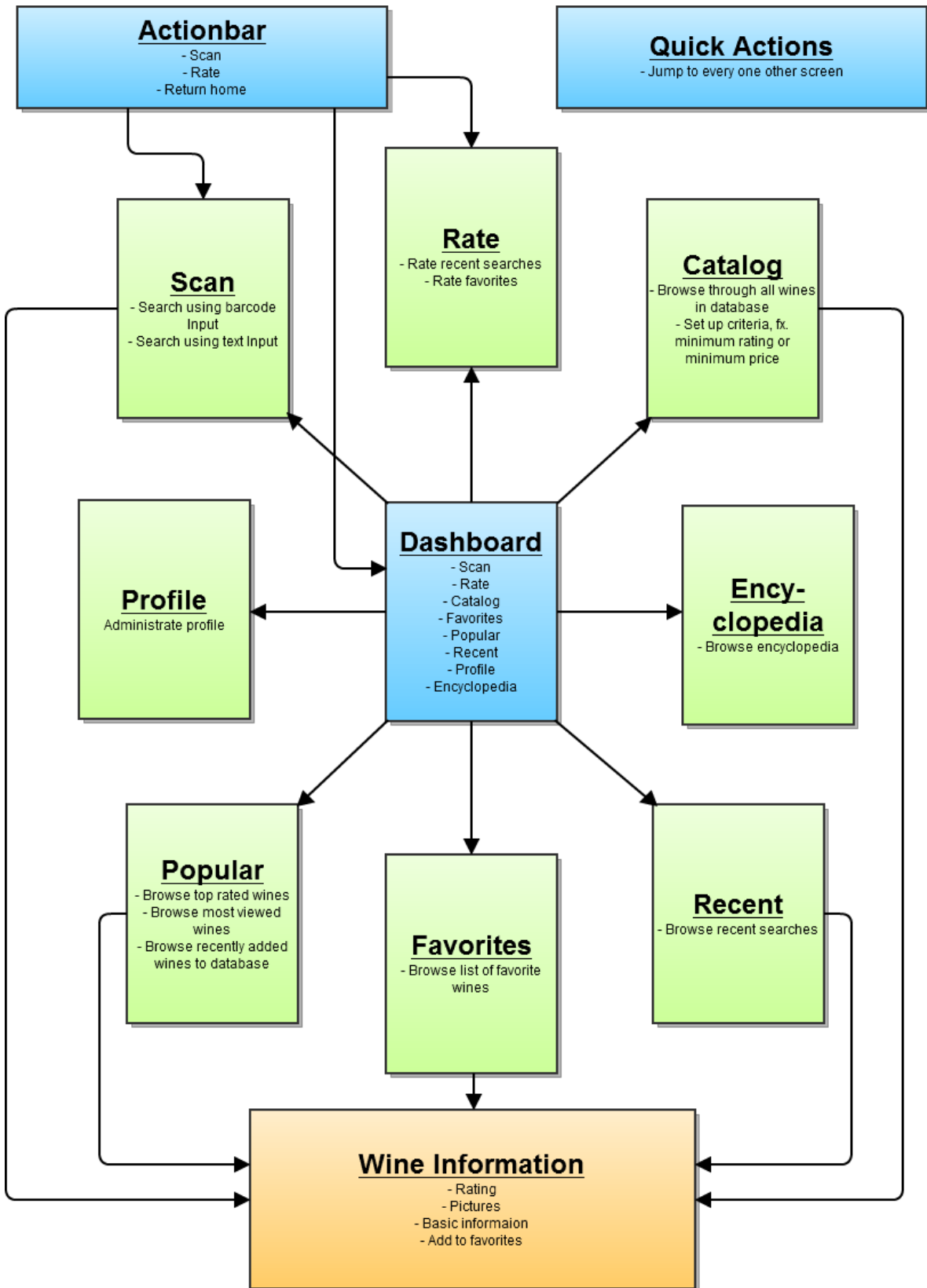


Figure 3.15: This flowchart display screen overview of the app. The main screen, the dashboard, is placed in the middle and all its sub screens emerge from it.

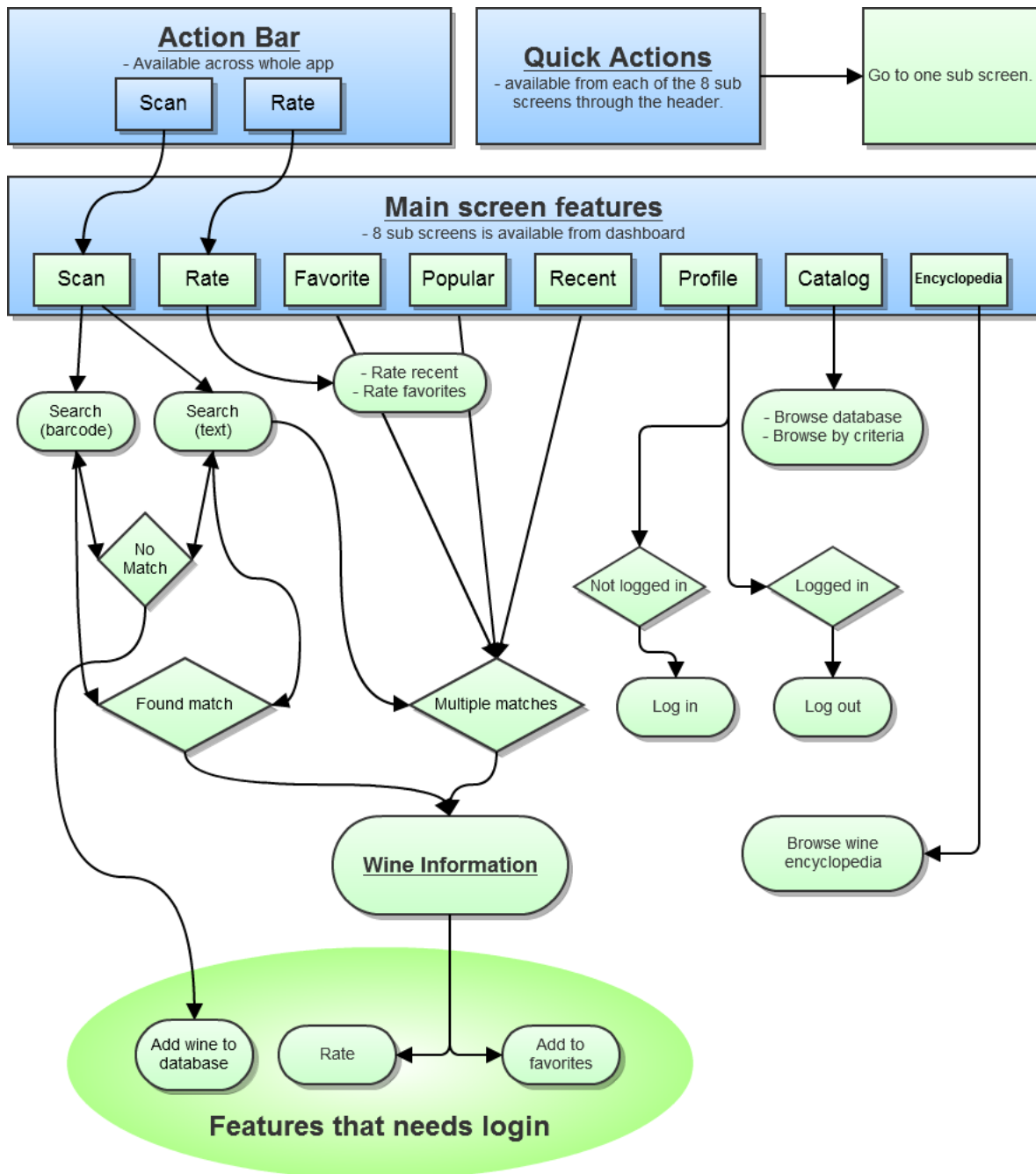


Figure 3.16: This flowchart display the actual application flow and what actions the user can do, and how the app responds to the users actions.

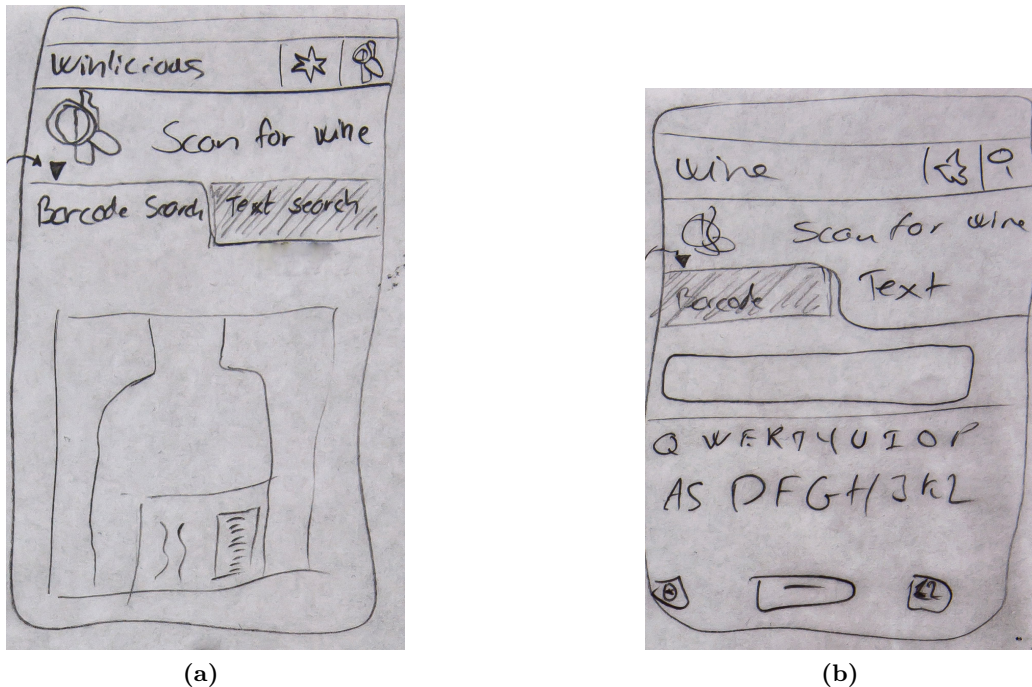


Figure 3.17: Sketch of the search screen, called scan in the app, which consists of two tabs. First tab enables the user to search by using an image, while the second enables the user to do a normal text search.

a specific wine. When the user fires up the app and thinks "I want to do a rating", she can quickly reach it by pressing the rating icon from the main screen or from the Actionbar.

Without the ratings screen the user should first search for the wine or look through her recent searches or favorites. With the rating screen the user can reach the rating in both ways which will create a good user experience. Hopefully this will lead to more ratings of wine.

On figure 3.18 the layout of the rating screen can be seen. It is only possible to rate wines that the user has either done a recent search on or has added to her favorites. These are the only wines that the app knows that the user is familiar with.

Two tabs are present, one for recent vines and one for favorite wines, and both shows a list of the wines. From the list the user can easy make ratings to the wines. When tabbing the icon of an item in the wine list, quick actions will be available as described in the common layout section 3.7.3.

The user can sort each lists in different ways by bringing front the menu pop up by pressing the dedicated menu button on the device, also described in the common layout section 3.7.3.

If the user has a wine that has not been done search on before, the user will have to go through the search screen instead. On the other hand that should be rather intuitive; you have to have done a search on a wine beforehand to do a rating of it.

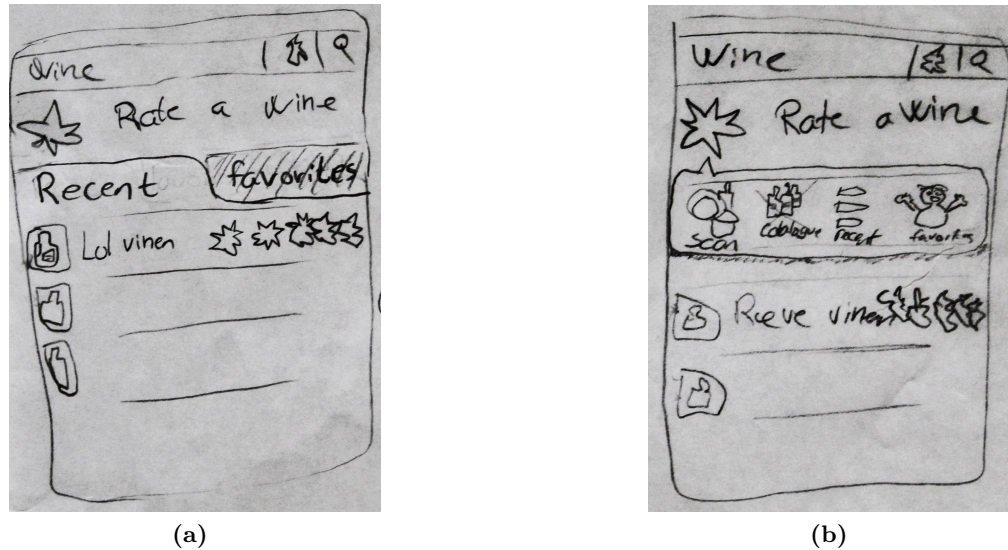


Figure 3.18: Left is shown a sketch of the rating screen. The user can browse through a list in the recent tab and the favorites tab to do rating. As described Quick actions are available when tabbing either the screen identification icon as shown in right, or tabbing an item icon in the list.

The Recent Screen, the Favorites Screen and the Popular Screen

These three screens looks a lot alike and will therefore be described together as one here.

The recent screen enables the user to browse through a number of wines previously looked at. It is like a search history. This makes it easy for a user to find a specific wine again after closing the app or switching to another location of the app.

In the favorite screen the user can browse through the wines that she has added to her favorites. It enables quick access to wines that the user liked, or wines that the user might want to look at again on a later occasion.

The popular screen is where the user can browse through wines that other users finds interesting. It enables the user to look through wines that she may not be aware of existed, and can inspire for other wines to try out.

The recent screen and the favorite screen only contain one tab each which lists the wines in each appropriated category. In figure 3.19 the recent screen can be seen in the left picture and the favorite screen can be seen in the right picture.

For the popular screen it has three tabs of lists of wines which are top rated wines, at the moment most popular wines, and recently added wines. A sketch of the popular screen can be seen in the left picture of figure 3.20.

As described in the common layout section 3.7.3, quick actions are provided when tabbing appropriate icons such as the screen identification icon or icons in the list views. Also the menu pop up is displayed when the user presses the dedicated menu button on her device which allows for different sorting methods. The menu pop up can be seen in the right picture

of figure 3.19. The quick actions are showed in the right picture of figure 3.20.

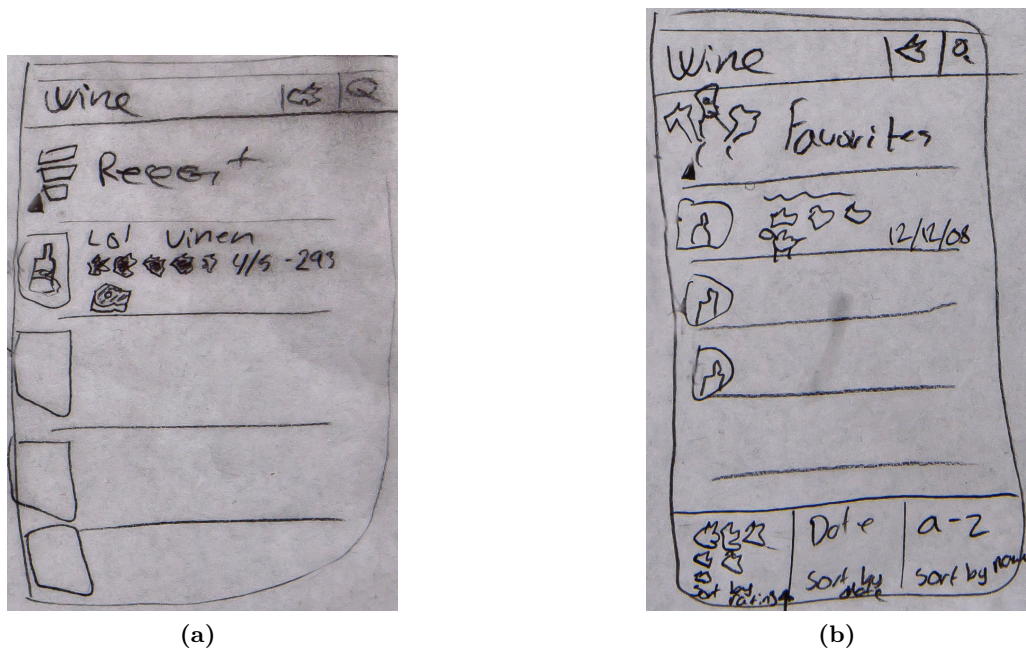


Figure 3.19: Left a sketch of the recent screen is shown. It consists of a list the user can browse through. To the right a sketch of the favorite screen is seen. It consists of a list containing wines that are added to the favorites. At the bottom of the favorite screen it can be seen how the menu pop up is displayed when the dedicated menu button on the device is pressed.

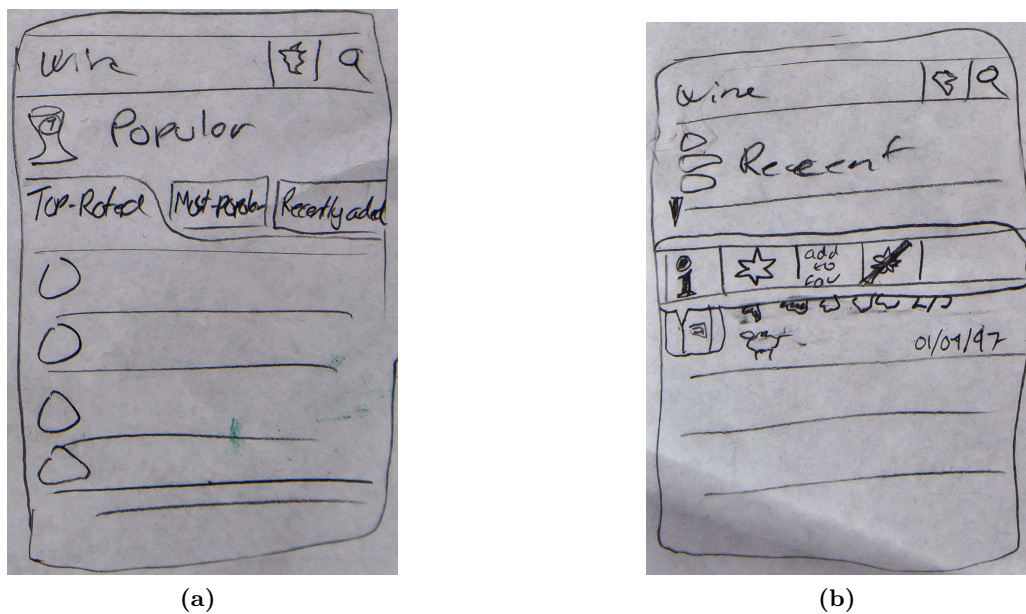


Figure 3.20: Left is a sketch of the popular screen which consists of three tabs for listing the top rated wines, the most popular wines and recently added wines. Right is shown how quick actions are available when tabbing appropriate icons in the different screen such as icons from the list.

The Profile Screen

The functionality of having a user profile is not yet completely defined. But it is a screen where the user will be able to change all his user information. It will contain much of the functionality that is seen on any other account page on the web or other apps. But the most important is a log in and a log out option. In addition the user should be able to create an account of course. A preview of the of the profile screen can be seen on figure 3.21. The ability to incorporate the very popular Facebook as a login feature is possible and by doing that it will get very easy for Facebook users to login, without creating a new user. Facebook has 2.7 million registered users in Denmark [43], and they are potentially users of the wine app.

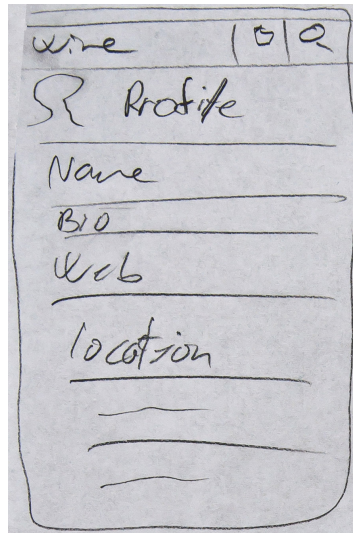


Figure 3.21: A sketch of the profile screen. The content of the profile screen is not completely defined, but will be like an account settings where the user can change information about her.

The Encyclopedia Screen

From this screen the user can browse through different terms used with wine and other relevant descriptions of phrases and expressions.

To promote positive user experience various ways of finding a specific term is provided, both directly and indirectly. Directly, the user can either do a typed search, lookup recent searches done, or browse through an alphabetic list.

Indirectly the user can look at most frequent lookups done by all users to see what other users finds interesting. Another feature is that whenever a word is used in other parts of the app that is contained in the encyclopedia, it will be highlighted or indicated that it is present within. The user can tab the word and go directly to the description in the encyclopedia. This makes it easy to make a lookup on a word unknown to the user, and support the interface design theory, by being fast and grant shortcuts to the user.

On figure 3.22 the encyclopedia screen can be seen with the search tab open.

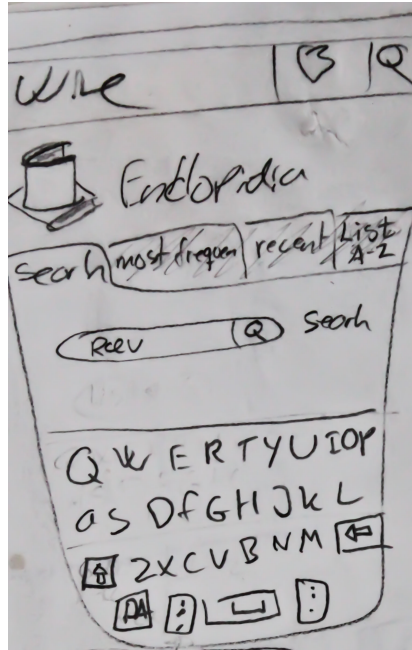


Figure 3.22: A sketch of the encyclopedia screen. Here the search tab is open and the user can type in a search.

When listing items it has the look as shown in figure 3.23. A small triangle like the one used below the screen identification icon indicates that it is expandable when tabbed.

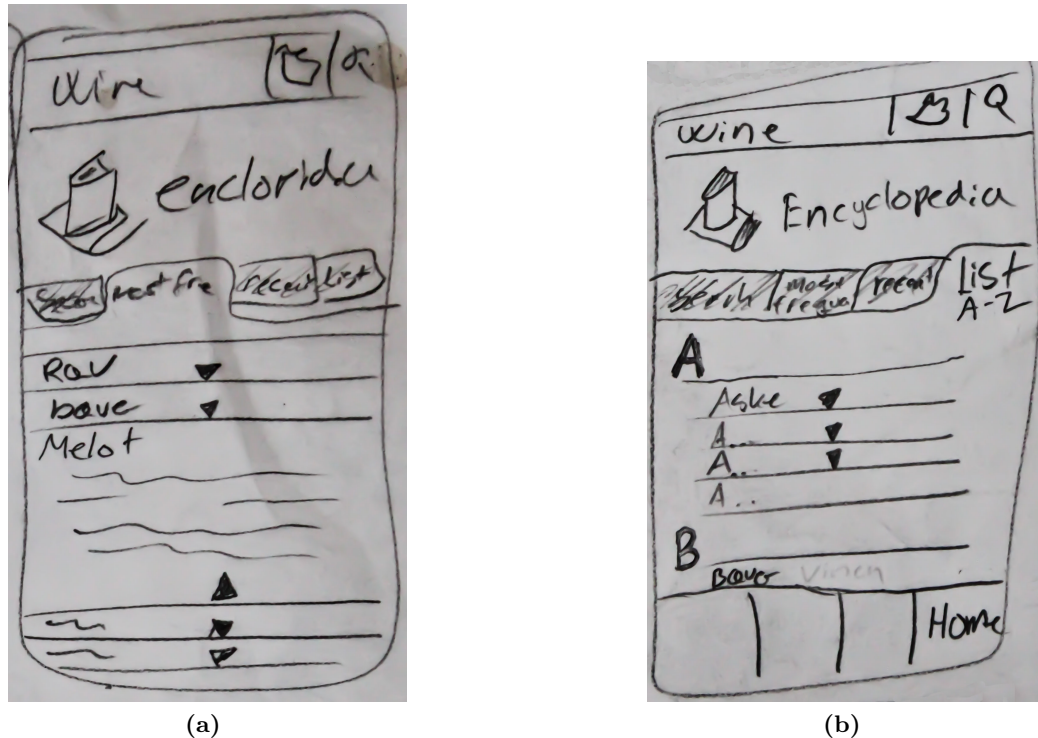


Figure 3.23: The encyclopedia with different tabs opened to show how listing of items is done. The small triangle on the item indicates to the user that it can be expandable by tabbing it. On the left it is shown how the alphabetical view should look like.

The Wine Information Screen

This is the screen the user will be presented for when tabbing a wine or selecting a wine after a search. The wine information screen can be seen in figure 3.24.

For the first iteration the wine information screen will only contain, image of wine, basic information, rating.

The picture of the wine will allow the user to see that the wine selected is the actual wine she is looking for.

For the rating it should be possible to easy rate the wine or remove a rating from here, to make the app fast and easy to use. This will be done by tapping the average rating.

3.7.6 Low Fidelity Test

The overall interface design has been demonstrated in the low fidelity design. Normally a low fidelity test would be made to try out the different interactions and see how they work before implementing it. In this case it would be done by making the different screens with pen and paper and have the user try out the functionality by interacting with the paper.

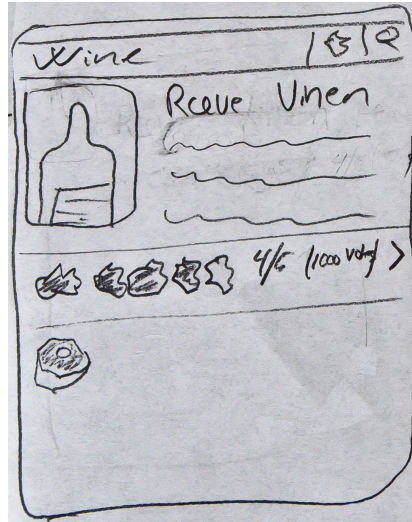


Figure 3.24: A sketch of the wine information screen. This screen contains all the information of a specific wine. The icon that looks like a nut is actually an icon of a hunk of meat to indicate that the wine goes good with that.

In this way, many pit falls can be avoided and any slow, confusing, or stupid interaction or functionality can be spotted and removed or changed to the better. This prevents wasting time on the change of a product that does not function appropriately.

Though many things, calls for doing a low fidelity test, it is chosen that the resources spend on developing, finding test subjects, and analyzing data, is not worth the effort in this specific case. Three reasons argue against doing a low fidelity test.

First, at the point where this design document was established, we as developers had minor experience with developing for Android. Instead of spending time on a low fidelity test we could spend the time better, by learning Android development and implementing some, or all of the low fidelity design. This way we use the time being experts in Android, which we need to spend time on anyway, and then it is possible to do a much better and more reliable test on the high fidelity prototype.

Second, with the minor knowledge about Android development we believe that it is fairly easy to move around with the different screens, buttons and interactions if something seems to be confusing. And as the content and functionality is more or less defined, it is just about placing it right in the application.

Third, the design patterns from Google have been thoroughly tested and are used in various successful applications, and indicate that most of the major interactions should work pretty well and be straightforward to use.

We estimate that the risk of not spending resources on a low fidelity version is low. Instead a high fidelity test will be made when the core interface has been implemented. If something is confusing or unnecessary it should be rather easy to change it in code.

3.7.7 Thoughts on Low Fidelity Design

Though most of the interface design has been settled, still there are many things that are not thoroughly defined. For example it is not defined how it should look like when the user make a search for a wine. Very possible there should be some indication that the application is searching and having the opportunity to abort the search, like the theory prescribe.

Also the database search function has not been defined because the functionality has not been discussed properly and it depends on other things like the construction of the database.

As described on the Dashboard design pattern 3.6 on page 66, the bottom part of the screen should be dedicated for content update, version updates, etc. In the version described here, it did not seem necessary for such part of the screen. In a future process it might seem more convenient for a place to do updates. To make room for such a part for example the recent screen and favorite screen could be merged into one screen. And the profile screen, which is not a very important part of the app yet, could be moved to the menu pop up, i.e. the menu that is displayed when the dedicated menu button is pressed.

3.8 Data Source

To be able to show information about wines a lot of data is needed. There is a vast selection of different wines, and all the data about the different wines have to come from somewhere. In the fundamental concept in section 3.1.1 on page 52 it was mentioned that the user should be able to add wines as part of the user generated content feature. However that feature was removed from the design for the first prototype, and to avoid a gaping database with no wine information contained, it must be decided how to supply the database. In the following that will be examined.

3.8.1 Wine Data Conditions

In section 2.10 on page 35 some features of wine is described and this thesis regard those as a minimum of information that should be in a candidate database with each wine entry. Furthermore the barcode, the wine title, and a photo of the wine, front and back labels, should also be contained:

- Wine photo
- Wine Image
- Grape varietal
- Region
- Country
- Winery
- Barcode number

3.8.2 Existing Wine Data Sources

Databases exist as well as Internet sites that present the wine data of a particular wine as well as reviews and user ratings. Wine.com is one example of such an Internet site guiding its users about wine and even with an API for developers allowing them to search their database. They have a lot of information about the wines, however they do not have barcode numbers in their wine data. The same seems true for other online sites in this category. Dansk Supermarked was contacted to investigate what they contain in their database of wine products. They contain only the barcode, a limited text-string and the price of the wine. That is not satisfying either.

3.8.3 Crawling and Pairing

One possible solution to the missing barcode information in Wine.com's database could be to get access to Dansk Supermarked's database and then pair the wines that are both in Wine.com's and Dansk Supermarked. The complexity of this solution was considered to be out of scope for the context that this thesis operates in and was not examined further in detail. However, it might be an interesting expansion to the future product of the wine app.

3.8.4 Making our own Data

As a consequence of the above, it was decided that making our own data was the best choice, even if it meant a lot of tedious work, but the end result would be a starter database with actual wine data for the first iterations of the wine application versus an empty database that is not very welcoming for users of the application.

The collection of the wine data was carried out in Bilka Skalborg in Aalborg which is a major shopping mall. Photos were taken of each wine in their wine division and later being inserted to the database. This was of course done with the acceptance from the supervisors at Bilka Skalborg.

3.9 Wine Features

Wine is a diverse subject and many things and features describe a wine. The wine application should be able to show information about a wine and therefore an investigation of what describe a wine needs to be done.

In section 2.10 on page 35 a brief description of wine was carried out and some important features was mentioned such as the grape varietal, region, country and winery. A more detailed research is needed to define details of wine and how wine can be classified and categorized so the user can make more refined and advanced searches. The further research revealed many features that could be interesting for users and how wine can be categorized.

Firstly, wine can be classified as a type such as red, white, sparkling, rosé, dessert and sake

[44]. The type of a wine is almost always defined by its grape varietal, so for example the grape varietal Merlot is of type red and a Riesling is of white.

The country that a wine is produced in and the region plays a major role. Not all countries in the world has wineries that produce wine and there is some sort of distinction between the classic countries such as France, Italy, Spain, Portugal, Germany and Austria and countries outside Europe. The countries inside Europe are often referred to as the "old world" and those outside as the "new world" [45].

The year that a wine is produced is also important because it can state how the weather was that season and can also play a role together with region, because of local weather.

The alcohol percent is also an important feature. Other things like what cap the wine bottle has, if the wine is a bag, the volume of liters it contains, photos of it and the description is all features that may or may not concern the user and the importance can be discussed.

Either way, its seems important for some users and by that it should be contained and showed to those that are interested.

One feature of a wine that has not been mentioned is the title of a wine. There seem not to be a clear distinction of what a title is. Some wineries gives their wines a distinct title, others only their winery brand name, others only the grape varietal as a title and others a mix of those three.

Many of the features above can be enumerated in lists and by doing that the more advanced searches the user can do. For example the user will specify from a list of types that he is interested in a red wine. After that he would maybe choose from the grape varietal list that he wants it to be a Merlot and the same for the year and so on. All the features that was found and if they are enumerated can be seen in table 3.1.

When the research was carried out it was experienced that most often the barcode of a wine does not change accordingly when a new vintage of a wine is released to the market. What this means is that there will exist more than one vintage on the same barcode. This should be taken care of when results of a scan is presented to the user.

Feature	Enumerated
Wine title	
Type	X
Grape varietal	X
Year	X
Region	X
Country	X
Winery	X
Cap	X
Container	X
Alcohol	
Price	
Liter volume	
Label description	
Front wine photo	
Back wine photo	
Full wine photo	

Table 3.1: *All features that was found and if they can be enumerated in lists.*

4 Implementation

This section is the first implementation section out of two, that will be encountered throughout this report. The section will cover how the established design from iteration one have been implemented into a functionally smartphone application.

This first prototype on the wine app, will clear the way for a afterwards test, that will prove whether the choices made on design, interaction and aesthetic feel, is a success.

4.1 Implementation Overview

The complete system as a whole can be described as a client-server relationship where the Android application is the client that communicates with the server. The application should be able to make requests to the server asking the server to do specific operations and return data if any.

The server needs to maintain a sensible database, and there need to be implemented a logic communication between the client and the server.

The client application should feature the user interface described in the user interface design, making the user capable of interacting with the server without any technically knowledge.

To enable the recognition of a wine, a barcode scanner needs to be implemented that is easy to use for the user.

4.2 Barcode Structure and Theory

For implementing a barcode reader in the wine application, knowledge about how a barcode is structured and interpreted is needed. This section will describe the structure of a barcode in detail and how a barcode is decoded and encoded. The section is based on Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras. [46] and Gs1 Australia user manual - bar code [47].

4.2.1 Information

The number beneath the barcode is not only a unique number, it actually contains several different pieces of information, see figure 4.1. The first two or three digits, depending on country, are the country code. Denmark has country code 57 and Azerbaijan has country code 476. The next 4 - 5 digits following the country code are the manufacturer code. Following

the manufacturer code is the unique 5 digit product code, and the last digit is the Check Digit, see figure 4.1.



Figure 4.1: The digits marked with blue is the country code, green the manufacturer code, pink the product code and yellow the check digit.

4.2.2 Structure of an EAN-13 Barcode

When looking at a barcode as the one shown in figure 4.2, it consists of two parts. First the actual barcode which is a sequence of vertical lines called bars, either black or white, and its decoded number shown underneath the barcode. When a scanner reads a barcode the number is actually not used, and the number is only printed for human interpretation only. To be able to decode the barcode to the printed number sequence, the barcode has a certain construction. As a security measure the barcode is encapsulated by three guards. Each guard is represented by the same block of bars. There is a start guard, a center guard and a stop guard, as seen in figure 4.2. Between the guards, blocks of so-called symbol characters are placed, and it is the symbol characters that decodes to the actual barcode number.

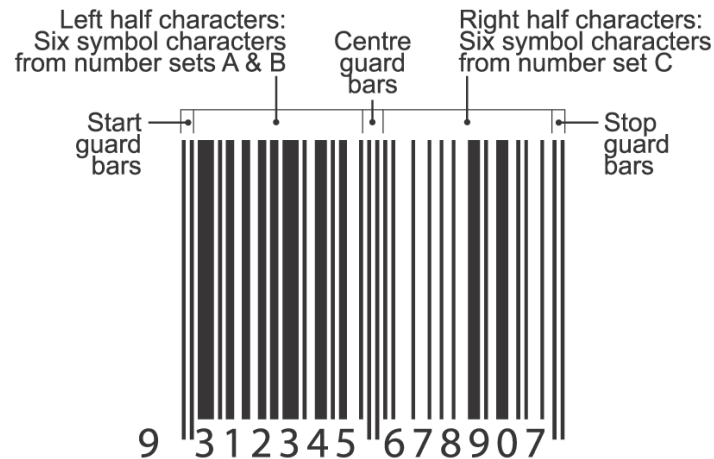


Figure 4.2: The EAN-13 Barcode's 5 parts. [47]

4.2.3 Encoding

Now that the general structure has been settled, the actual encoding should be analyzed. Each number in the number sequence beneath the barcode, except the first which is placed outside the start guard, see figure 4.2, are each represented by seven so called modules. These seven modules can be either white or black, and seven modules together are called a symbol

character. Each symbol character corresponds to a digit written below the barcode. Each symbol character has exactly two black and two white bars which vary in width from one to four of the total 7 modules, see figure 4.3.

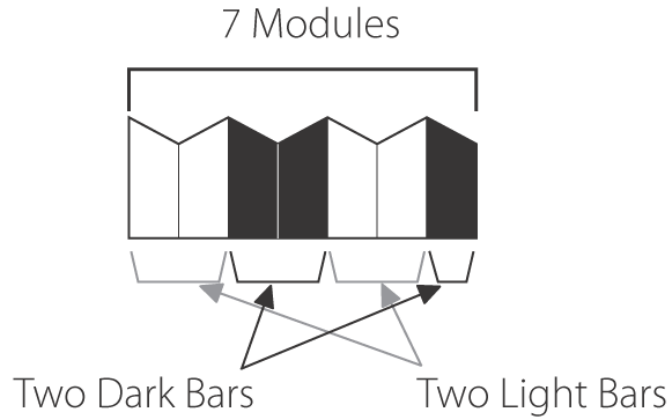


Figure 4.3: It takes seven modules to create a symbol character. A symbol character has two white and two black bar differing in thickness where each bar has a maximum thickness of four modules. [47]

The barcode from start guard to stop guard spans a total of 95 modules. In addition, 18 modules of so-called quiet space is placed as a region around the barcode to prevent other graphic from interfering when reading. The 95 modules are 12 symbol characters, 6 for the left half and 6 for the right half, and three special symbol characters namely the left, middle and right guard, see figure 4.4.

Symbol Type	Left Quiet Zone	Left guard bar pattern	Left half character	Centreguard bar pattern	Right half character	Right guard bar pattern	Right Quiet Zone	Total
EAN-13	7 modules	3 modules	6 x 7 = 42 modules	5 modules	6 x 7 = 42 modules	3 modules	11 modules	113 modules

Figure 4.4: The EAN-13 Barcode's total number of modules. [47]

Different number sets are used to encode each symbol character. There are three different number sets an A,B and C set. The A set have an odd number of black modules. B and C have an even number of black modules, see figure 4.5. The left half of the barcode are using the A and B set and the right half of the barcode uses only the C set.

How the number set A and B is used, is determined by the first digit in the country code which is the left most digit in the barcode, e.g. the number outside the guards. Take the barcode, 5701012822110, as an example. The first digit in the country code is 5. Then the left half which is the numbers 701012 are encoded using the following sequence of number sets: ABBAAB. The right half, which are the numbers 822110, are encoded using only the number set C, see figure 4.6. The guard symbols are unique both in construction and length compared to the symbols and is constructed according to figure4.7

Value of digit	Number set A	Number set B	Number set C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Note: 0 represents a light module and 1 represents a dark module.

Figure 4.5: Shows the encoding of digit 0 - 9 in Number set A, B and C. 0 Represents white modules and 1 represents black modules.[47]

Number sets used for coding left half of bar code						
Value of 13th digit	12th digit	11th digit	10th digit	9th digit	8th digit	7th digit
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Figure 4.6: The sequence of number sets used to create the left half of the barcode is determined by the leftmost number in the barcode. If the country code is 57 the sequence is: ABBAAB.[47]

The Check Digit

The last digit in an EAN-13 Barcode is the check digit. The check digit is made to ensure that the barcode is constructed and read correctly.

Example:

We will calculate if the check digit is correct from the following barcode, 7804315304944, where the check digit is the last digit, namely 4. Using equation (4.2) you sum all odd positions in this case 703509 and multiply by 1 which in this case equals 24. Sum all even positions in this case 841344 and multiply by 3 which equals 72. Add the two sums together and you get the total weighted sum, in this case 96. To get the check digit you subtract 96 Modulus 10 from 10 which gives 4 and corresponds with the check digit from the barcode, see equation 4.4.

Auxiliary characters	Number of modules	Module set
Left and right guard bar pattern	3	1 0 1
Centre guard bar pattern	5	0 1 0 1 0
Special guard bar pattern (for UPC-E Symbols)	6	0 1 0 1 0 1

Note: 0 represents a light module and 1 represents a dark module

Figure 4.7: The figure shows the number of modules and the encoding of the barcode guards. [47]

$$WeightedSum = \left(\sum_{i=1}^{12} x_i(i \bmod 2) \right) \cdot 1 + \left(\sum_{i=1}^{12} x_i((i+1) \bmod 2) \right) \cdot 3 \quad (4.1)$$

$$96 = \left(\sum_{i=1}^{12} x_i(i \bmod 2) \right) \cdot 1 + \left(\sum_{i=1}^{12} x_i((i+1) \bmod 2) \right) \cdot 3 \quad (4.2)$$

$$CheckDigit = 10 - (WeightedSum \bmod 10) \quad (4.3)$$

$$4 = 10 - (96 \bmod 10) \quad (4.4)$$

4.2.4 Decoding

To read a barcode you use a barcode reader. The most common barcode reader is a laser scanner like the one found in a supermarket or at the post office. The laser scanner uses laser light to illuminate the barcode and a photo diode reads the laser light reflected from the barcode. The laser and diode is tuned to each other using the exact same frequency filtering out any unwanted illumination [48].

When the photo diode has read the reflected light you get a signal looking similar to figure 4.8. In a perfect world it would be a perfect square wave. After some signal processing you get a result like figure 4.9.

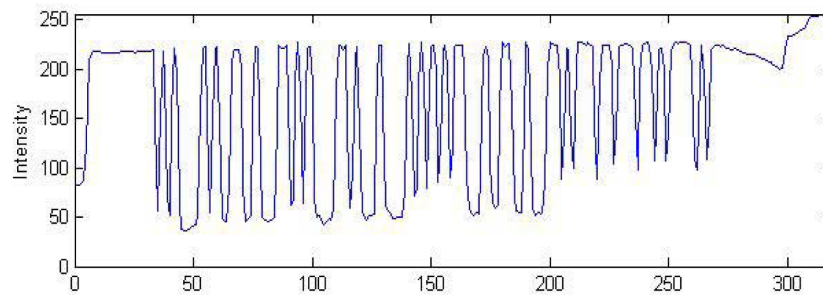


Figure 4.8: A possible photo diode response of reflected laser light.[46]

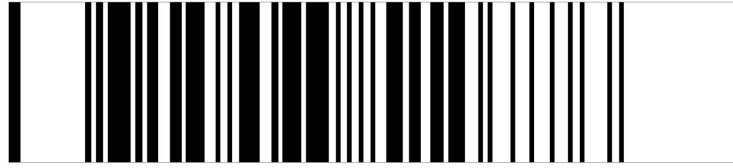


Figure 4.9: *The thresholded result from figure 4.8.[46]*

When the signal is processed and ready to be decoded, the decoding algorithm first look for a guard to see where the barcode begins. When the start of a barcode is found you start decoding the modules to get the individual symbols, which can be translated to a number, see figure 4.5. If you start reading a barcode from left to right you will soon find out that it only consist of twelve digits. The left most digit is not represented by 7 modules like the rest. When you have read the left half of the barcode and reached the center guard you can construct the first number by looking at which number set the given number is written with. When done for all 6 numbers in the left half you can reconstruct the missing number by looking at figure 4.6. When you have read all 13 digits and reach the end guard you compare the first 12 digits to the last which is the check digit to see if you have read all the numbers correctly using equation 4.2 and 4.4.

4.3 Database and Storing Data

How to store data greatly depends on what kind of data it is. Binary data such as music are most often just stored as the archives (files) that they typically are organized as. Plain text can be stored just like music in archives on a hard disc, and organized (named) into what subjects they refer to. But keeping data in this manner is not very flexible. Searching a hard disc for a certain subject in a collection of plain text files, is not very effective because it takes a lot of time opening each file from the disc. And keeping track of how data relate to each other is not very easy.

Of course data could be separated into meaningful categories in one file, but maintenance of such a database is hard and not optimized. Dedicated database software exists where the data is structured and optimized for quick retrieval.

The probably most used database model is the relational database model which organizes data into *relations*, also typically referred to as tables. Each relation has *attributes* and *tuples* which describes the relation. Attributes and tuples are also commonly referred to as columns and rows like in an ordinary table [49]. Managing a relational database is most often done by using Structured Query Language (SQL).

The database software used for the wine app will be the MySQL database developed by MySQL AB, a subsidiary of Oracle. The following will describe the fundamentals of relational databases and how to interact with them. Furthermore we will look at how the database containing wine entries are designed and implemented.

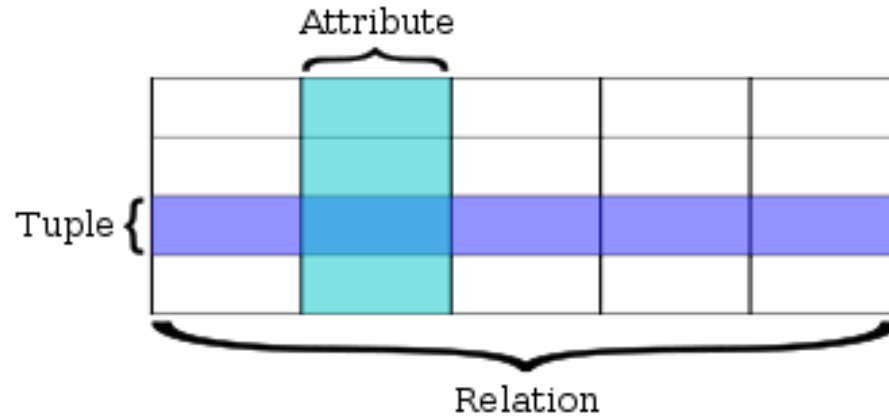


Figure 4.10: *The terminology describing a relation in a relational database. A relation is typically referred to as table and attributes and tuples are commonly referred to as columns and rows like in an ordinary table. [49]*

4.3.1 Structure of Relational Databases

So a relational database consists of relations with attributes and tuples. We will here refer to a relation as a table, an attribute as a column and a tuple as a row. The basic idea of a relational database model is that it consists of a series of tables each representing data that relate to each other. A row in a table can be seen as a relationship between the data across the row. Each column in the table represents a permitted set of values also called the domain or type of the column. A column's domain can for example be of only integers in an interval or a text-string.

An example of a table in a database is seen in table 4.1 where some fictional data about date and temperature relate to each other. Besides a “date” and a “temperature” attribute, it also has a “key” attribute which purpose are to make the data unique and is a common design principle to keep unique references to data. Of course the actual raw data behind the database and its tables are stored in a much more efficient way than plain tables so retrieval and manipulation of the database is optimized as much as possible. It is all up to the authors of the relational database software how that is done and is out of the scope of this report.

key	date	temperature
1	01-07-2011	12.5
2	02-07-2011	18.4
3	03-07-2011	20.0
4	04-07-2011	19.3

Table 4.1: *A table representing a relation with three columns (attributes) and four rows (tuples). Each attribute has a permitted set of values and each row describe a relation between the data across it.*

4.3.2 Managing Relational Databases

A relational database is often maintained with Structural Query Language (SQL) which is a database computer language. SQL is a declarative language which means that you only express *what* is going to happen and not *how* it will happen. SQL uses specific statements which determine what is going to happen with some specified data in a database. Listing 4.1 shows a SQL query where we want to get the date when the temperature was over or equal to 19. When getting data from a table you always get a new table containing the data. The SQL query in listing 4.1 would return a table with one column (date), if any data matches the SQL expression.

```
1 SELECT date FROM temperature_table WHERE temperature >= '19';
```

Listing 4.1: Shows a query where the date attribute is selected from a table named “temperature_table” where “temperature” is over or equal to “19”.

4.3.3 Designing Relational Databases

To be able to design the structure of a database, containing some sort of items, you need to identify all the data or features that are required to understand that sort of item in the context that you are working in. For example, if you are making a database of all the customers that are placing orders on a website you maybe need some information about those customers like their address and what they have ordered. There are some principles when designing relational databases and we will cover a few of them here.

Tables

Each table in a relational database should only contain one type of information or items. This can be customers of a web store or the orders that they are placing [50].

Uniqueness

Every row in a table should be unique so that a row can be uniquely referenced from a programming context. This can be done by having a primary key (PK) column that has a unique value for each row. This is typically done with a column with an integer domain that is auto incremented each time a new row is added. The column can now be used as an identifier for a certain row in that table. It can also be an actual attribute of the data that you know are always unique. [50].

Relationships and Foreign Keys

Designing databases consist usually of creating several tables that in some way relate to each other. A primary key comes very much in handy when you want to create relationships between the tables. The primary key of one table can be referenced from another table creating a relationship between those two. Foreign keys (FK) are columns that are used to reference primary keys in another table [50].

4.3.4 Designing the Wine Database

From section 3.9 on page 84, features that can characterize a wine are extracted in addition to extra information that might have an interest for the user. All of those should be contained in the wine database in addition to the barcode numbers: ean8 and ean13 numbers that is paired with each wine.

A complete list of all the 18 attributes that a wine entry should be able to contain can be seen in table 4.2 along with an example of the input-data of each. By looking at those features we can get an overview of the data and start analyzing each feature to determine which domain it is.

#	Feature	Example
1	Wine Title	Banrock Station
2	Type	Red wine
3	Grape varietal	Merlot
4	Year	2009
5	Region	South eastern Australia
6	Country	Australia
7	Alcohol	13%
8	Winery	Banrock Station
9	Cap	Cork
10	Container	Bottle
11	Price	57.99 DKK
12	Liter volume	750mL
13	Label description	Great red wine from south eastern Australia with a long tradition for craftsmanship...
14	Front wine photo	\photos\front_photo.jpg
15	Back wine photo	\photos\back_photo.jpg
16	Full wine photo	\photos\full_photo.jpg
17	EAN8	52374654
18	EAN13	9311043058585

Table 4.2: All features and information that should be contained in a wine entry with examples of each.

wines (main_table)	
PK	<u>wine_id</u>
	ean8 ean13 title year alc volume_litres cap price label_description country_iso container winery front_photo back_photo full_photo region grape varietal type

Figure 4.11: A database model diagram showing the main table of the wine database. An almost replicate of table 4.2 and it needs further optimizations. The final database model diagram can be seen in figure 4.12

We will start with one table that will be referred to as the main table with 18 columns corresponding to the wine features as in table 4.2 on the previous page. The first thing to do is to make each row unique. There are no feature of a wine entry that are unique so an auto incrementing PK named “id” are added to the table, making it 19 columns in total. Figure 4.11 show a database model diagram of the main table with the columns.

Recall from 3.9 that some of the wine features are enumerated: type, grape varietal, region, country, winery and container which mean that only a pre-defined set should be able to be contained in that column. For example, the grape varietal set is a list of grapes such as “Merlot”, “Shiraz”, “Cabernet Sauvignon” and the country is a list such as “France”, “Spain”, “Denmark” and so on. This leads to defining new tables for each of these features and then draw relationships with the main table with foreign keys.

Furthermore some additional relevant data needs to be kept for the winery and by making a table only for wineries we can add those extra columns where it is relevant. The same is done for the photos that are paired with the wine entry, namely a table containing front, back and full paths columns. The complete database model diagram can be seen in figure 4.12 on the facing page.

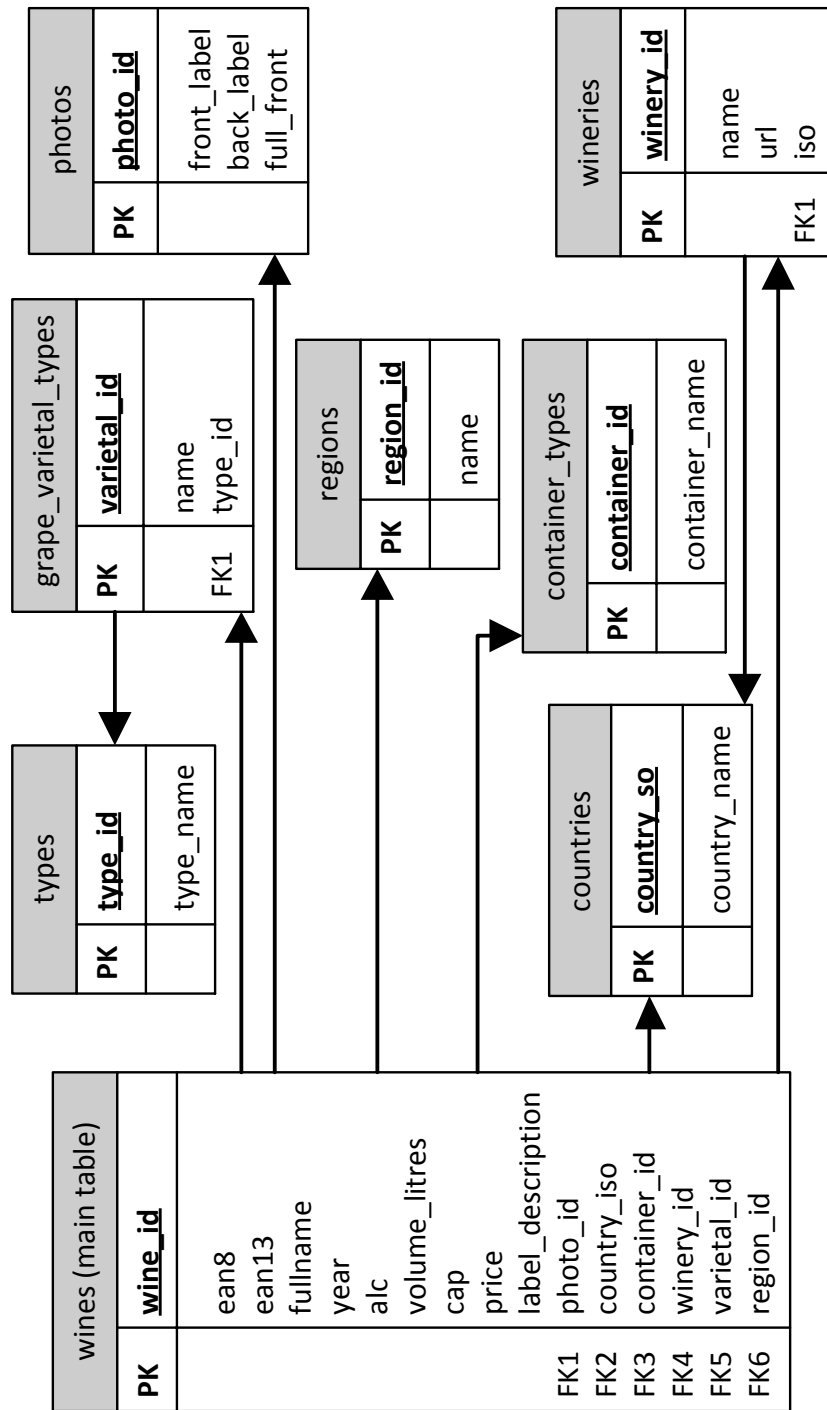


Figure 4.12: Database model diagram showing the tables in the wine database and their relationships. The main table (*wines*) contains foreign keys to other tables which define the relationship with those.

4.4 Database Administration

To be able to administrate the database such as adding new wine entries, a web application was developed and will be referred to as the back-end. The back-end lies on the same server as the MySQL database and was written in PHP. It can only be accessed by authorized users and are not a web application that ordinary users would use, but only administrators and therefore there is not defined any rules for the user interface.

As described in 4.3.3 the database consists of several tables that relate to the main table in some way. This means that when a wine entry are added it is not just one table that has to be manipulated, but all related tables. Without some interface to add a wine entry, it can quickly become very confusing and time-consuming to add just a single wine, because you will have to update a lot of tables.

Also error-checking and formatting of the data can be done before inserting it to the database. This basically ensures that the database does not become corrupt. Figure 4.14 on the next page shows a flow-chart depicting the back-end when an user (administrator) wants to add a wine. Figure 4.13 shows an error helping from the web application, to prevent the user not doing duplicate wine entries.

Backend vin.lommeguiden.dk

Wines

Users

Du er logget ind som Kasper L. E. [Log af](#)

Tilføj vin

Barkoder

EAN8:

EAN13:

EAN13 nummeret for denne vin eksisterer allerede i databasen!

Årgange:

- 2009

Figure 4.13: Shows an example of an error / warning to the administrator, so bad or duplicate data are not inserted to the database.

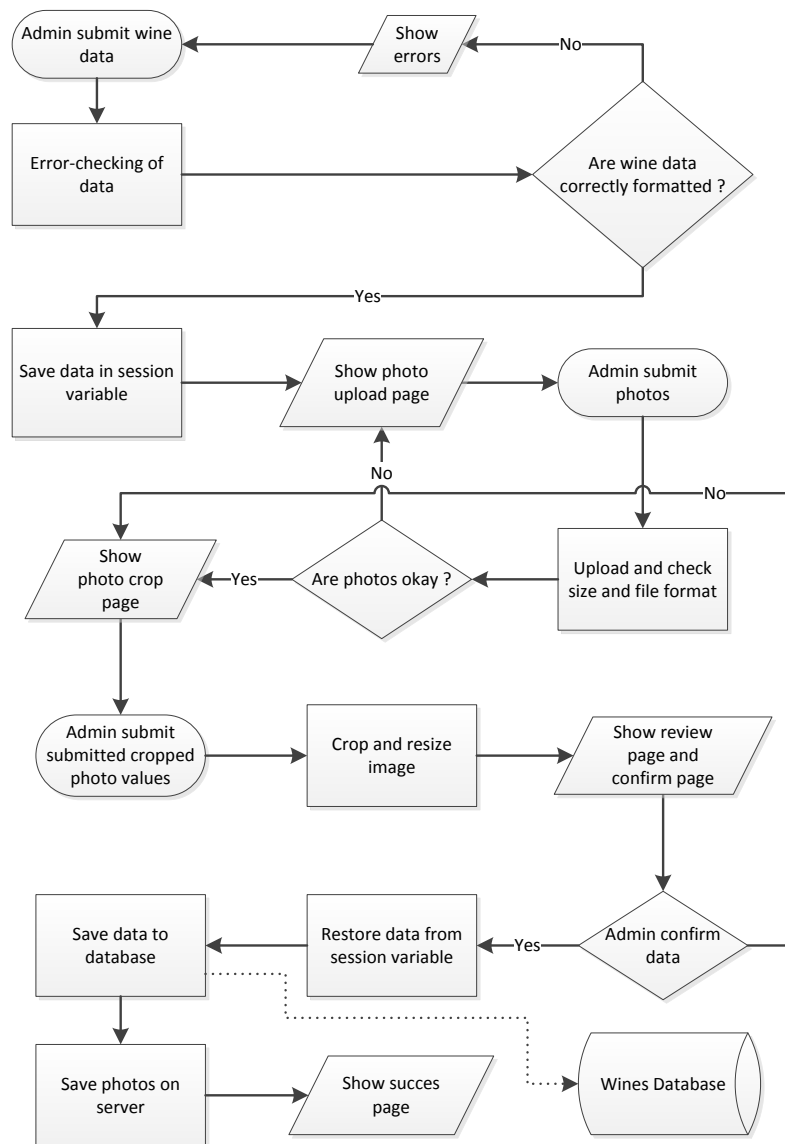


Figure 4.14: A flow-chart depicting the back-end when an administrator wants to add a wine.

4.5 Client - Server Communication

The Android application will communicate with a server over the Internet and will be the client in a client-server relationship. The application should be able to make requests to the server and ask for specific information such as searching for a wine in the database and rate wine etc. This kind of communication is often referred to as pull technology because the client requests some data and the server replies with a response.

The server will be implemented with a software style referred to as RESTful Web Service [51] derived from the Representational State Transfer (REST) style. It is a style that is mainly implemented to communicate over the Hypertext Transfer Protocol (HTTP) with the standard HTTP operations POST, GET, PUT and DELETE that often are referred to as REST verbs.

Characteristics of REST are that it is stateless and thereby every request from a client holds everything needed to complete the request with a server. A session is therefore not needed before the client can ask the server for something. REST operates on resources and returns representations.

To understand the concept, consider listing 4.2 which show a Uniform Resource Identifier (URI). An URI specifies where a resource is located on a network and has a structure that consists of four elements: scheme, host, path and query. The scheme represents the protocol that is used to reach the resource. The host identifies the physical server where the resource is located, path determine the precise resource and the query string tells the server if additional operations should be applied to the resource or how the representation should be returned.

A resource is typically a file on the server and the representation can for example be an image or a sound file. Often the representation when communicating with RESTful Web Services is XML or JSON and when a client receives the response from the server the representation can be determined by the “content-type header” in the HTTP response.

1	<code>http://</code>	<code>//scheme</code>
2	<code>www.server.dk</code>	<code>//host</code>
3	<code>/path/to/resource</code>	<code>//path</code>
4	<code>?limit=5&offset=0</code>	<code>//query string</code>

Listing 4.2: *Show a Uniform Resource Identifier and its structure.*

REST encourages to use the standard HTTP operations and that the use corresponds to the way the HTTP operations was meant to be used, also referred to as “CRUD” mapping [52]:

- When creating a new resource, POST should be used. (Create)
- When retrieving a resource, GET should be used. (Retrieve)
- When updating a resource, PUT should be used (Update)
- When removing an existing resource, DELETE should be used (Delete)

There is not a standard way of implementing a RESTful Web Service, because REST is a style and not a protocol. RESTful Web Services can be prototyped quite easy and fast, but the practical use when doing that can be that the REST style is not completely fulfilled, because the developer might not use the correct HTTP operations for the action that they perform. For example a developer can make a Web Service that only uses GET for all interactions: create, retrieve, update and delete. The implementation of the server in scope of this report will not fulfill the REST style completely, because it is considered more important to get the server working in general and communicate with the Android application.

4.6 Server

The actual server software will be a script running on the same server as the MySQL database and programmed in PHP. In REST terminology the script would be the resource. This script will just be referred to as the server. The REST style does not rely on any distinct environment and the standard that it is derived from (the HTTP standard) is designed to work on all environments between operating systems. The Android application will use the HTTP protocol to make requests to the server and the server will return a response with a representation in a format that the Android application can decode and understand.

4.6.1 JavaScript Object Notation

The representation that will be used will be the JavaScript Object Notation (JSON), because it is greatly supported in the Android framework as well in the PHP API. JSON is a lightweight data format to exchange common types that is easy for both humans and computers to read and write, and is based on a subset of the JavaScript Programming Language. The power of JSON is that it is completely independent of programming languages, but still use conventions that are very much similar to the C-family of languages. Listing 4.3 shows a JSON encoded object with an array name “colors”. The elements of the “colors” array are itself objects of two strings in each.

```
1 {  
2   "colors": [  
3     {  
4       "name": "red",  
5       "hex": "#FF0000"  
6     },  
7     {  
8       "name": "green",  
9       "hex": "#00FF00"  
10    },  
11    {  
12      "name": "blue",  
13      "hex": "#0000FF"  
14    }  
15  ]  
16 }
```

Listing 4.3: Shows a JSON string that defines an object with an array called “colors” where each element is an object with string elements within each.

4.6.2 Filtering clients

The server is reachable from the Internet and by that everyone that are connected to the Internet can make requests to the server. To gain a filtering method so only authorized clients can make requests a security layer will be implemented. This will be done by having the server require a key from the client supplied in the request. The key is known in the

Android application (not to the user) and should be transferred in every request. This is of course not a hardcore bulletproof security layer, and the key can be intercepted by monitoring the HTTP stream between client and server. This could be avoided by using an encrypted stream such as the Hypertext Transfer Protocol Secure (HTTPS) protocol that uses the SSL/TLS protocol to provide secure communication between a client and a server. The scope of this server and resources available did not allow for such an implementation. Furthermore it should only respond to requests that it knows and if not the request should be rejected.

4.6.3 Determining what to do

As stated in 4.5 the implementation will not fulfill the style of REST completely in that it won't use the HTTP operations as defined in the REST style when creating, retrieving, updating and deleting. All requests will be tunneled through GET and the server will reject all others. This is simply because it has been faster and easier to debug and implement a quick server solution this way.

The server will decide what the request contains and what the client asks, based on the query string. The query string is a collection of fields and values separated with "&". A field and a value are formatted as "field=value" so a query string could look like: "field1=value1&field2=value2". In the script context the fields and their values can be accessed as variables. By setting distinct variables in the query string you can from the script check if those variables are set and do specific functions. The variables will as well function as parameters.

4.6.4 Processing the Request

When it is determined what the server should do from the request query string, the actual processing of the request will be done. This will often include making queries to the database and do sub-routines and error checking to complete the request and finally send a response to the client that initiated the request.

4.6.5 Client Users and Authorization

Users should be able to create a user-account and login with it so user information such as adding wines to his or her favorites can be accomplished. Furthermore it is a requirement that a user is logged in if the user wants to rate a wine.

The implementation of this will be based on giving a unique token that clients receive on successful login requests to the server. In the database holding user information, each user will have a unique id as well as the currently valid token for that user. The token and user id should then be transmitted in every request that the client initiates for at least those functions that require a logged in user. The server will then only allow tokens that are valid for the user that the client initiates its requests as. This also allows the client to save the user id and token to keep the user logged until the user explicitly logs off.

In figure 4.15 the login request is depicted.

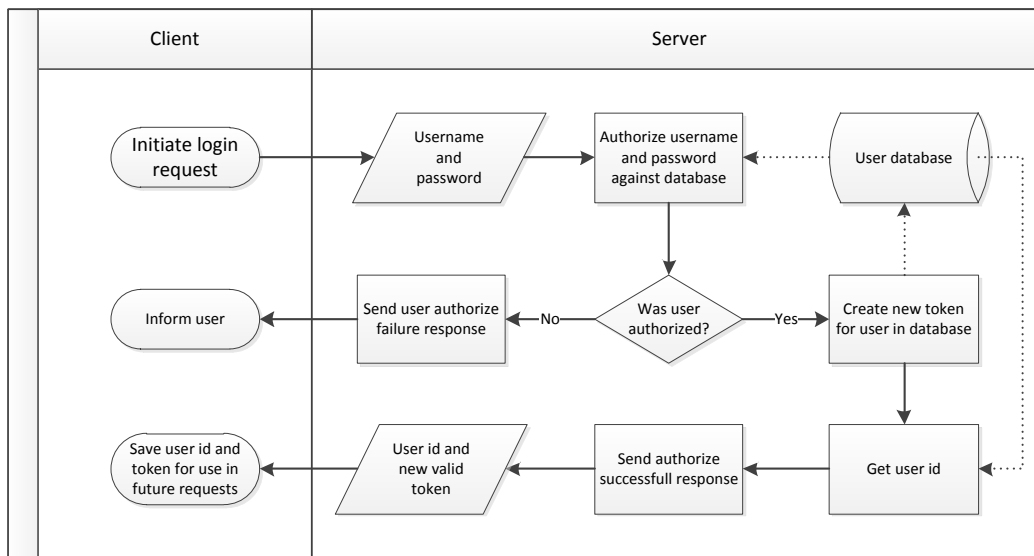


Figure 4.15: Users will login on the client with their username and password and the client initiates a request and ask the server if the user is authorized. If so, user id and a token will be send back to the client that will use it to make requests as a specific user.

4.6.6 Putting it Together

The overall functionality can be summarized in figure 4.16 depicting the flow between client and server.

4.6.7 Supported requests

In the following we focus on two kinds of requests that a client can make to the server: Searching the wine database and rate a wine.

Search the Wine Database

In section 3.4 it is defined that the user should be able to do a search with barcode search (scan) or a search with text with advanced options such as defining filters to refine the search. The two search options differ in the numbers of features that may be sent with the request. If a barcode search is performed only that barcode number should be enough to do a search, but if an advanced search is done the number of features can vary between one and the complete suite of features that define a wine in the database. The easiest solution is to make one search request type that has a parameter that can contain from minimum one feature to the total number of features.

The field in the request query that will be used as the parameter will be identified as “filter”

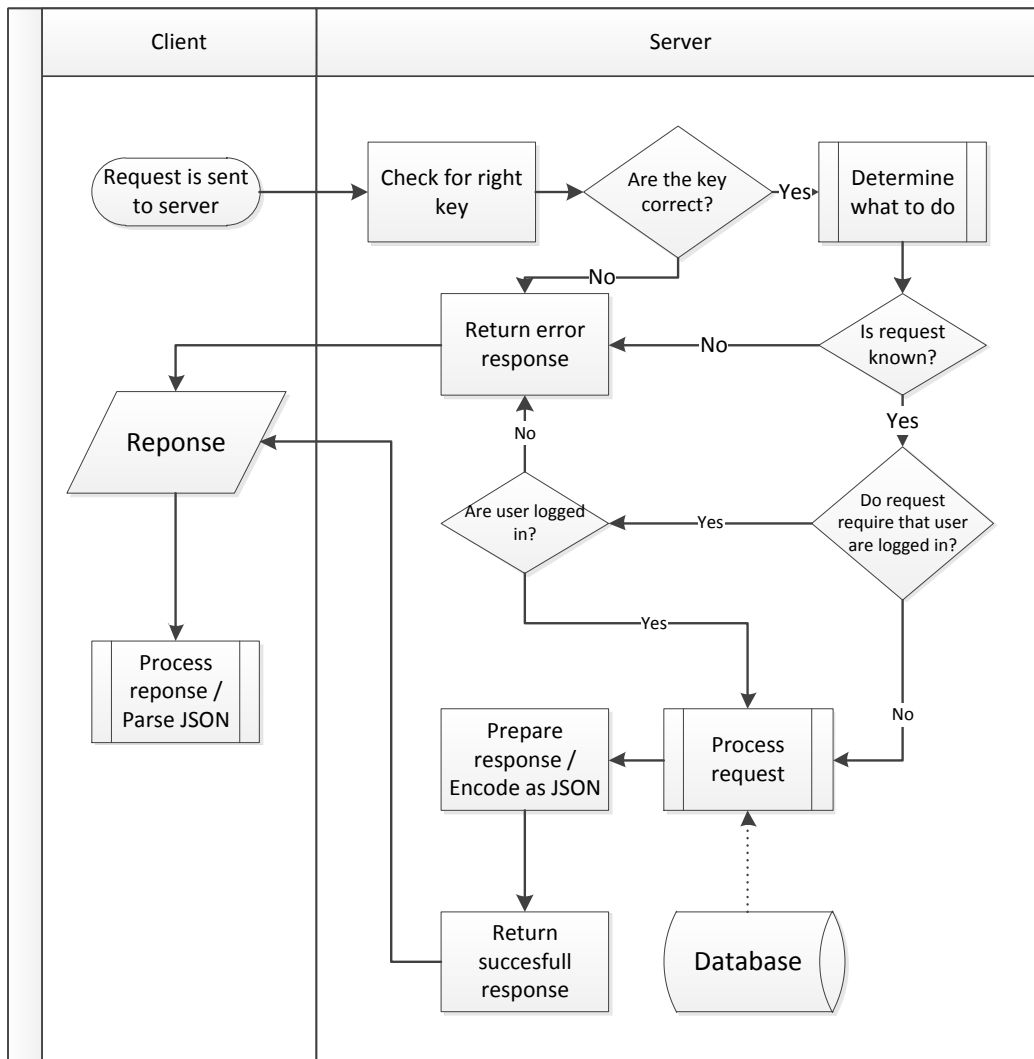


Figure 4.16: Shows the general flow of a request sent from the client to the server.

and the value will contain feature names formatted as “feature(constraint)” separated with “;”. Examples of such query strings in a search request can be seen in listing 4.4.

```

1 filter=title(Amselkeller);alc(>12.5);type(red)
2 filter=ean13(4049366000152)

```

Listing 4.4: Show examples of two search queries. Line 1: an advanced text search. Line 2: a barcode search using ean13 number.

The processing of the request involves splitting the filter parameter with “;” as delimiter and then performs text-pattern recognition to identify the features and their constraints. Next a

SQL query is constructed from the constraints and sent to the database and are depicted in figure 4.17.

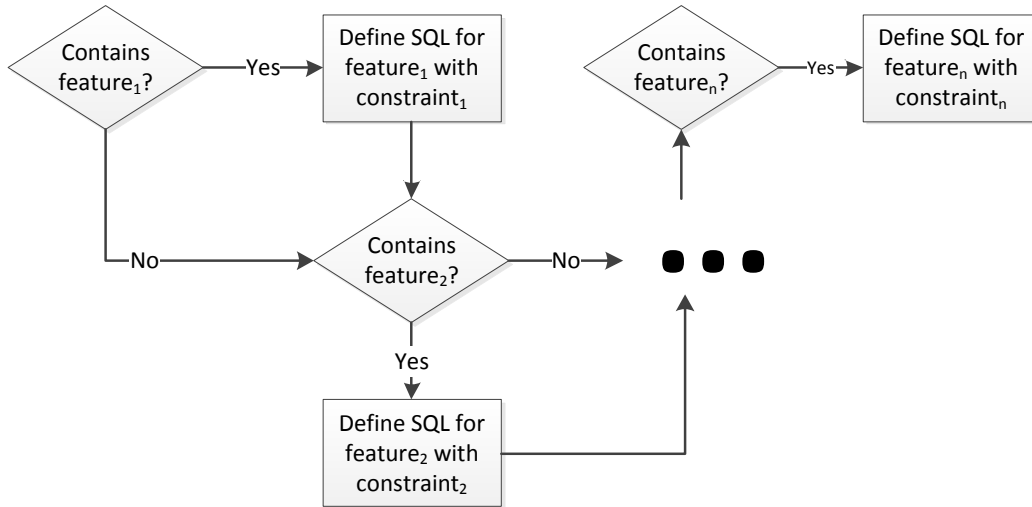


Figure 4.17: The SQL that will query the database in a search are generated from features and its constraints if they was contained in the filter parameter in the query string sent as part of the request.

The result of the search can either be an exact match with one wine entry or a match with several wine entries. The response will always be an object with a size element and a list element. The size specifies how many wine entries is carried with the response and zero if there were no match at all and the list element is the actual array of wine entries where each entry is an object with all its features attached. The search request is depicted in 4.18.

Rate Wine Request

To rate a wine it is required that the user is logged in. The parameters that must be sent with a rate wine request are the user id, wine id and the actual rating that the user gave the wine. When the user is logged in, the user id should always be transmitted with the request. The user id is required to perform a rating because we want to keep track of which wines a user has rated. To keep track of this a new table are added to the database that will hold all ratings including the actual rating, wine id and user id. Each time a rating request are successful it should be recorded in that table. Besides that, the wine table (main table) will be altered with two extra columns. An average rating and a total number of ratings column. When a wine is rated the average rating of that wine will be recalculated with the new total number and the new total rating. If the user already has rated the wine beforehand the rating should be changed. This will involve updating the ratings table with new rating and removing the old rating from the total sum of ratings and adding the new one before calculating the new average rating. The whole process of the rating request can be seen in figure 4.19.

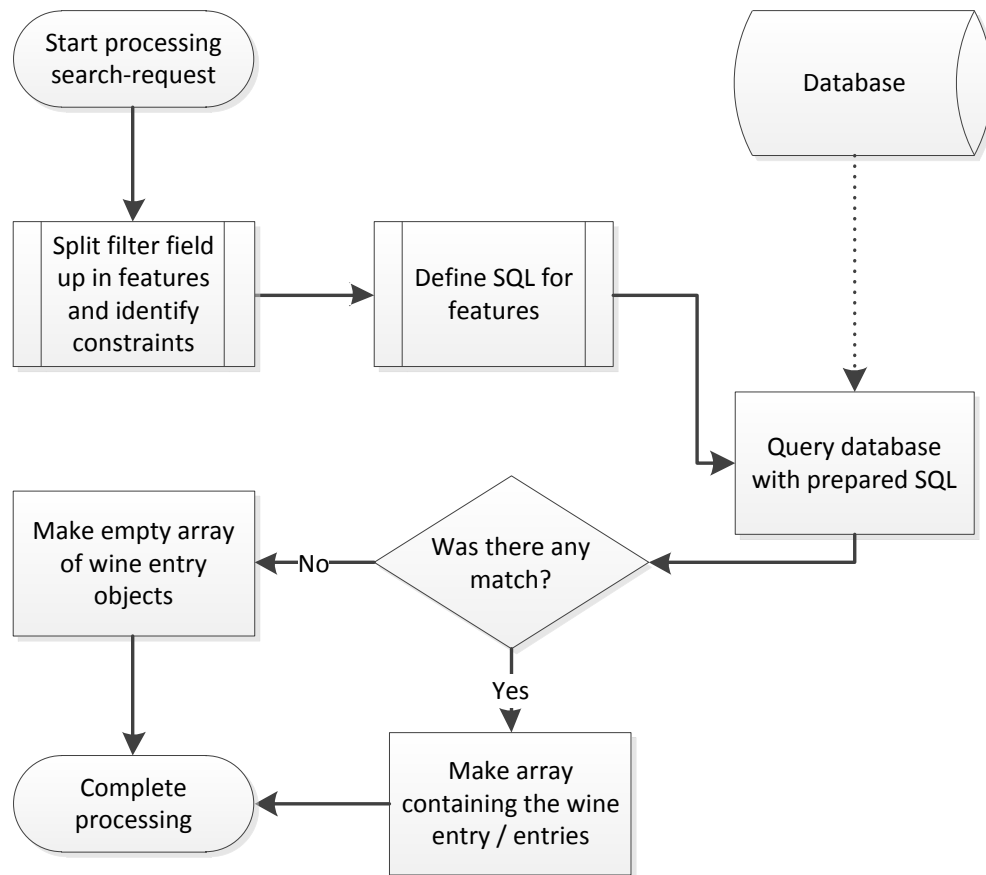


Figure 4.18: The processing of the search request.

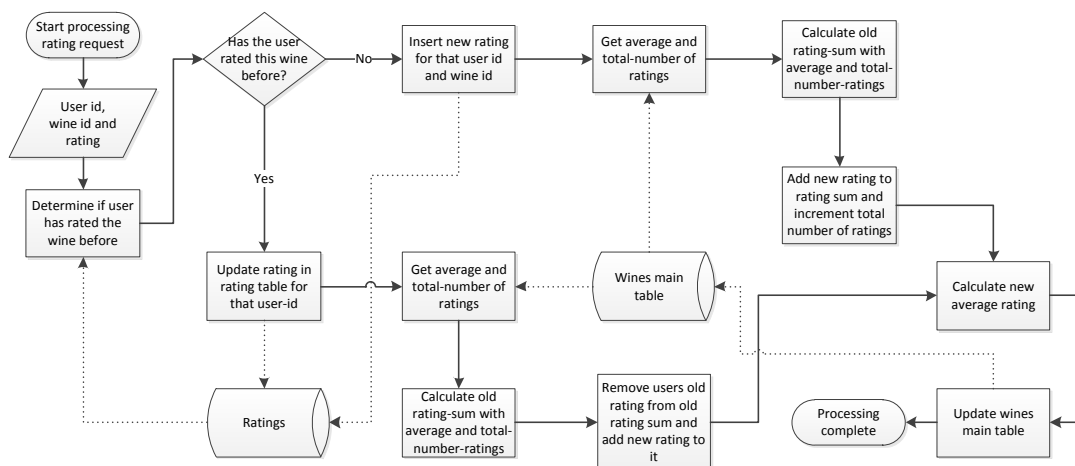


Figure 4.19: The processing of the rating request.

4.7 Multi-threading

The following will introduce the concept of threads as they play a big role in making fluently user interfaces where work are performed in the background.

Multi-threading refers to concurrently executing code in two or more threads. In most operating systems, threads lives inside a process and share the same resources such as memory, in contrast to processes that are completely separated in their address space and therefor do not share any memory. Typically when a process is started, a main thread is also spawned that will keep running until the process is terminated and usually if more threads are spawned, it is the main thread that initiates the additional threads which in turn can create new threads and so on [53].

Of course on a device with only one processing unit (CPU) like many mobile devices, threads are not actual running simultaneously, but the execution of the threads occur by switching between different threads. This switching happens so frequently that it seems that the threads are running in parallel. On multi-processor systems, threads will actually run in parallel, as each processor will execute one thread at a time.

Threads share the same memory space as the process that they where initiated in. This mean that threads in a process can access the same resources and thereby communicate and exchange data. This intraprocess communication is less expensive than interprocess communication between processes, because processes saves more information about their state that will have to be restored on switching process.

But intraprocess communication between threads should also be dealt with carefully because it can cause some serious errors. These threading-errors can be hard to track and debug, because the error nous code will in many cases compile and run, but only fail some times. This comes from the unpredictability of threads. You cannot predict when the CPU will switch from one thread to another. If two threads access the same resource at the same time, an error will occur because the memory address of the resource can only be read or written by one thread at a time. Threading errors of this type can be prevented blocking other threads from accessing the resources that the blocking thread are using. The place in code where a thread is blocking other threads are typically called the critical section.

When blocking a critical section in code some issues can happen if the threaded problem is not thought well enough, namely *deadlocks*, *livelocks* and *starvation*.

Deadlock

A deadlock refers to when two threads are in a state where they both are waiting for each other to complete or waiting for the other to release a resource. They will be in this state forever because neither of them completes or releases the resource because they both block.

Livelock

A livelock is similar to a deadlock, but instead the threads constantly change state for the sake of the other thread to make progress, but still blocks the other. They are both busy-waiting for each other, but never makes any progress because the thread's action is a response to the action of the other thread. An analogy of this is when two people meet in a narrow corridor, both of them tries to be polite and makes way for the other, but step to the same side and

keep blocking each others way.

Starvation

Thread starvation occurs when a thread cannot execute its code because other greedy threads are blocking resources for longer periods of time, and not making the required resources available regularly, so the thread can make progress.

4.8 Developing for Android

The following sections will introduce the basics of developing applications for Android.

4.8.1 Preparation

Applications for Android are written in the Java programming language and XML. To start developing you need the Android SDK which supplies you with the necessary tools and APIs. An Android application is an "android package" which is compiled with the SDK tools, from all the code, data and resources that makes up the whole application. The package is a file with the suffix ".apk" and can be distributed as a self-contained Android application on, for example, the Android Market.

4.8.2 Architecture of Android

Android relies on the Linux kernel and runs Java programmed applications on top. This does not mean that Android run its applications in the standard Java Virtual Machine (JVM), rather applications run in Google's own specialized virtual machine called "Dalvik" executing byte code compiled in Dalvik format and not the standard Java byte code. This makes a big difference when comparing to normal mobile development with the Java ME package in that most of the Java SE framework is supported on Android in addition to the Android framework.

The different layers and major components of Android can be seen in figure 4.20. The top layer is where all applications run. These include core applications shipped with Android and those written by third party developers. The "Application Framework" layer interfaces with the lower layers supplying the framework that the applications depend on. The lowest layer is the Linux kernel which Android relies on.

It can be noted that the Android OS does not differentiate from core applications and third party applications as they run on the same layer and has all the same privileges per default. This also means that core applications of Android can be replaced as wanted.

4.8.3 Features of Android

According to the Android Developer Guide [54], Android has the following features:

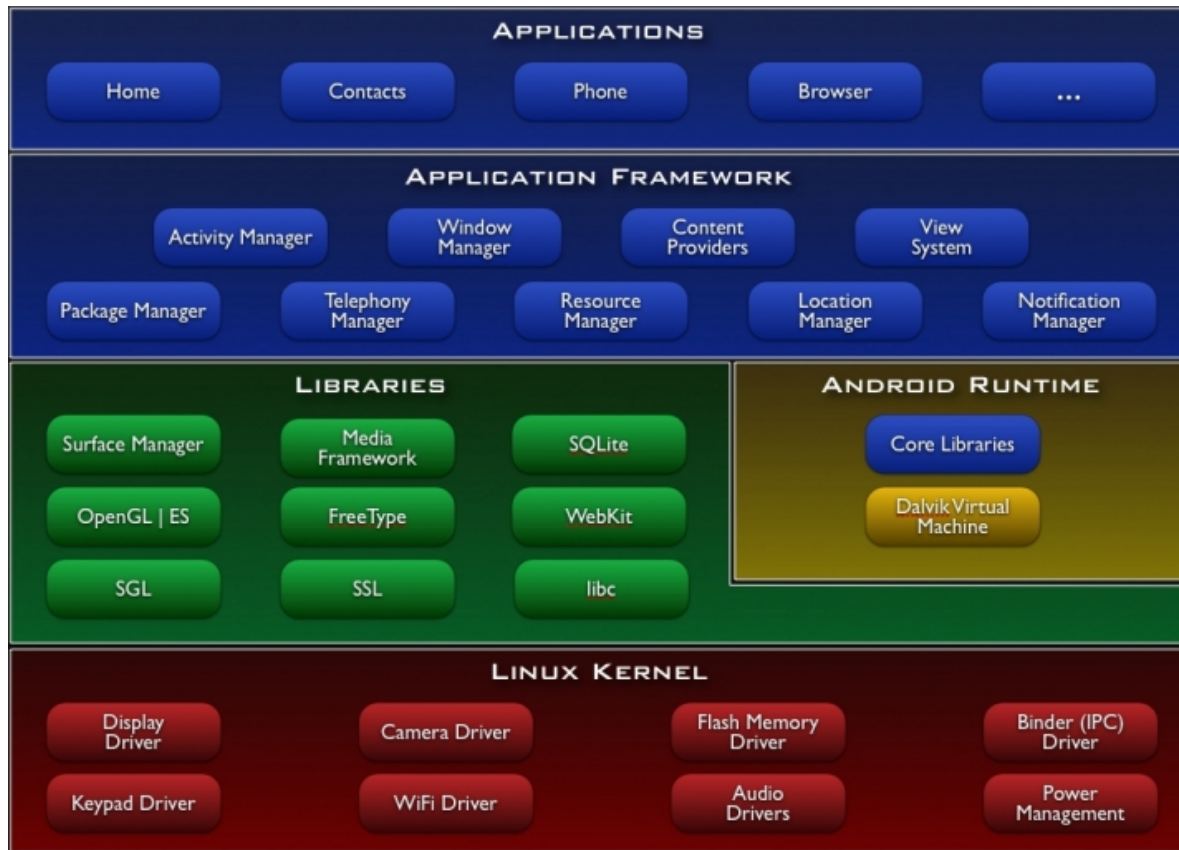


Figure 4.20: Shows the Android system architecture and its layers. [54]

- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source WebKit engine **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

4.8.4 Building Blocks for Android Applications

Android has four types of main components when developing [54]:

1. Activities
2. Services
3. Content Providers
4. Broadcast Receivers

Each of the components defines a different entry point that the system can enter to the application.

Activities

The activity component is one of the main building blocks when building applications. They are as their name suggests: something that *does* something. An activity has a user interface attached for the specific action that the activity stands for. An activity could for example be the creation of a new calendar appointment or reading a list of news feeds. An activity is created by directly sub classing the “Activity” class and overriding some methods. A true power of Android activities is that they can be shared among other applications and form a framework that other applications can use to share data with each other.

An activity is activated by a message called Intent, and carry information about what the requesting activity wants from the called activity. This is also why an Android application has not a single entry point like a main() function. An activity should be able to be started from everywhere in any context. Multiple activities usually define a complete application where each activity performs different user actions [55]. In listing 4.5 an activity is created by sub classing the Activity class and the onCreate method is overridden on line 4. The onCreate method is called when the Activity is created and is used as the entry point to the activity.

```
1 public class AnActivity extends Activity {  
2     /** Called when the activity is first created. */  
3     @Override  
4     public void onCreate(Bundle savedInstanceState) {  
5         super.onCreate(savedInstanceState);  
6         setContentView(R.layout.main_layout);  
7     }  
8 }
```

Listing 4.5: *An activity is created by sub classing the Activity class.*

Services

In contrast to an activity, a Service is an activity with no user interface attached. More specifically a service performs long running operations in the background. Operations could be playing music, perform network activity requesting data from the Internet without blocking user interactivity. A service is created by sub classing the Service class.

Content Providers

On Android there is one way to share data across applications. There is no shared storage mechanism for data that is not stored on the SD card. Content Providers makes specific application data available to other applications. Data can for example be SMS data, phone book contacts, images etc. It's mostly meant to provide a way for other applications to utilize your data. If you don't want your data public you simply don't provide any Content Providers.

Broadcast Receivers

Broadcast Receivers is used to respond to system-wide events that need a quick response. Broadcast receivers typically don't have a user interface as they can be regarded as gateways to other components, that reacts to the event and display something on the screen. Many events come from the Android system, such as a SMS received event or an incoming phone call.

Resources and User Interfaces

All resources such as images, strings, user interfaces, audio, animations and everything related to the visual presentation of the application are separated from the code and declared in XML files in the projects resources folder. Every resource that is declared by the Android SDK tools is automatically assigned a unique id that can be referenced from the code to manipulate it.

On Android user interfaces are built with “views” that are objects based on the View class. A view can for example be a button or a text view. User interfaces can be defined in code, but the advised way is to define them in XML files so code and layout are kept separated.

Manifest file

Another important property of an Android application is its manifest file. The manifest file contains information about the application that the Android OS needs to know and are declared in XML. Permissions, activities, intents, package name and other attributes need to be declared in the manifest file. An important thing to know is that you need to set up permissions to use special functionalities such as network capability for reaching the Internet, accessing the camera and other that can have an impact of the device the application is installed on. All those have to be declared in the permissions of the manifest file.

4.8.5 Threads on Android

When an application is launched on Android the system creates a thread for that application called “main”. On this thread all code that is dispatched from this application is run including all UI updates for that application why it is often called the “UI” thread. Performing tasks that does very intensive or long running work should not be done in the main thread because it will block. When the main thread blocks no interaction or UI updates will occur and it will seem to the user that the application is not responding or hangs. Furthermore, after 5 seconds that the main thread are blocked the Android system will pop a dialog saying that the application is not responding and is not very desirable.

Also the main thread is not thread safe when accessing the Android UI toolkit framework. It is the framework that has to do with all of Androids layout classes such as Views etc. Therefore it should never be accessed from others threads than the main, as it can cause threading-errors.

Creating new threads from the main thread in an Android application is the same as creating threads on the Java SE platform with the Thread class and Runnable interface. However threads can be complex if you want to access the UI toolkit in the main thread. The AsyncTask class in the Android framework should be used cause it automatically performs the blocking operations when accessing the main thread so the developer are less likely to make threading-errors when accessing the UI toolkit outside the main thread.

4.8.6 Android Versions and API levels

As a result of the progression that Android undergoes, new versions of the system are released and new features and changes are added. This results in different requirements to the applications because the framework that they rely on is committed. Therefore for each version of Android shipped a corresponding API level is issued. This API level can be leveraged so developers can specify for which platform their application is targeted at. Table 4.3 shows versions of Android shipped and their API levels. The API level is an integer that started at level 1 for the first version of Android shipped: 1.0. As of now the current newest release is 3.1 and the corresponding API level is 12. The API declaration is specified in the manifest file.

4.9 Application Implementation

The fundamental of Android development have been established, and the back-end have been detailed. In this section the development of the actual client, the wine app, will be described. The section will not go into detail on every bit that has been created, only the fundamental of building and organizing activities, interfaces and structures will be addressed. This section will be based on the Android Development Guide [54].

Platform Version	API Level
Android 3.1	12
Android 3.0	11
Android 2.3.4	10
Android 2.3.3	10
Android 2.3	9
Android 2.2	8
Android 2.1	7
Android 2.0.1	6
Android 2.0	5
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

Table 4.3: *The different versions of Android and their corresponding API level as of May 2011 [54]*

4.9.1 Overview of Activities

From the definitions of the different screens in the application in section 3.7.5, ten screens are identified to be contained in the application, namely: Dashboard, Scan, Rate, Catalog, Encyclopedia, Recent, Favorites, Popular, Profile and Wine Information. For screen overview refer to the figure 3.15 on page 74. In addition to these, three screens more are considered. A login screen that users will authorize themselves, a new user screen where they can create a new user to login in with and a screen which show a list of wines. The Android activities in the application are naively made from these screens and are seen in figure 4.21.

4.9.2 Displaying an Interface

Every activity have an interface attached to itself, describing the structure and purpose of the interface. The most common way to define an interface is with an XML file. An interface can also be defined using native code, but is mostly used when something more dynamical want to be achieved. Best practice is to use XML files.

4.9.3 Defining an Interface

When defining an interface, for example through a XML file, it is build using View and ViewGroup objects, which are objects available in the Android SDK.

The View class serves as the base for subclasses called widgets, which offer fully implemented UI objects, and widgets serves as an interface for interaction with the user. Hence, View widgets is the elements that appear to the user. Widgets could be any of standard objects such as a text field or an image view, but could also be more complex interfaces like a date picker or zoom controls. The Android SDK provides a wide range of different widgets that

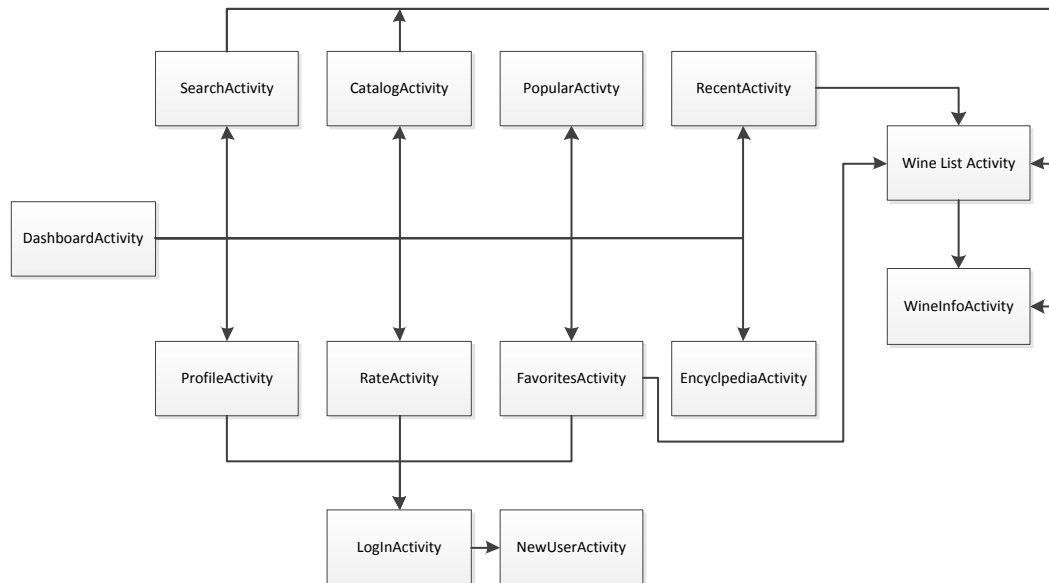


Figure 4.21: *An overview of all the activities identified for the application.*

support most of the basic needs. It is also possible to custom define a view, creating actionable elements, or extending or combining existing widgets.

The ViewGroup class serves as the base for subclasses called layouts, which offer different kinds of layout architecture. The layout structure holds all the View widgets and is used to define the positions and arrangement of them. The Android SDK offers many different layouts to use, and some of the most common is the LinearLayout, TableLayout and RelativeLayout.

A View Widget

As described the View is used to create subclasses called widgets. A most common widget could be an editable text view, called EditText, which in the app, would be displayed as a text field, where the user can edit its content of text.

When defining the widget, the developer has a wide range of attributes to adjust, that will define the widgets behavior. The View class has some standard attributes which every widgets inherit, which for example could be size, padding, etc. In addition the EditText will have its own attributes to be defined such as, which characters it accepts and how many lines it supports.

A ViewGroup Layout

As described the ViewGroup is used to create subclasses called layouts. A most common layout could be the LinearLayout, which is a layout that arrange its child Views in a linear order one after another.

As with widgets, when defining the layout, the developer again has a wide range of attributes that defines the behavior of the layout. For a LinearLayout for example, it is most appropriate to define whether its content should be arrange horizontal or vertical, and for example the gravity of the content.

4.9.4 Structure of an Interface

To build an interface with ViewGroups and Views, a hierarchy of nested nodes is created, which can be as complex or as simple as it is needed for the application. A ViewGroup is the parent and the View is its child. A hierarchy of ViewGroups and Views can be seen in figure 4.22.

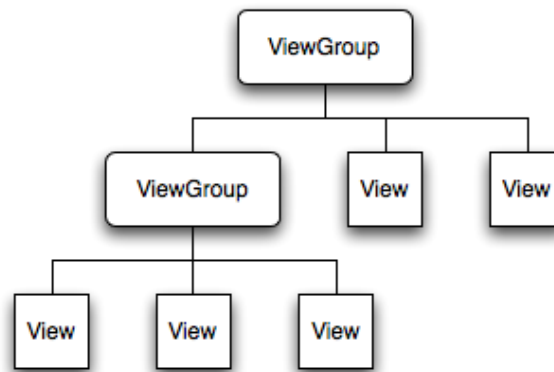


Figure 4.22: The view hierarchy in Android used to define interfaces. The ViewGroup act as parent, defining the positions of its children. A View is child of a ViewGroup and is a actionable element called a widget, that the user can interact with.

The Dashboard as Example Screen

For the purpose of demonstrating the creation of an interface, the structure of the dashboard will be presented. Recall from the interface design, section 3.7.1 on page 69, how the main screen, constructed as a dashboard, would consists of a grid, of two by eight button icons.

To build the dashboard, a structure of nested linear layouts and image button widgets is needed. The structure can be seen in figure 4.23.

A interface always have one, and only one, root ViewGroup, which in this example is a LinearLayout. The orientation is set to vertical, having all of its children arranged one after each other, in a vertical direction starting from the top.

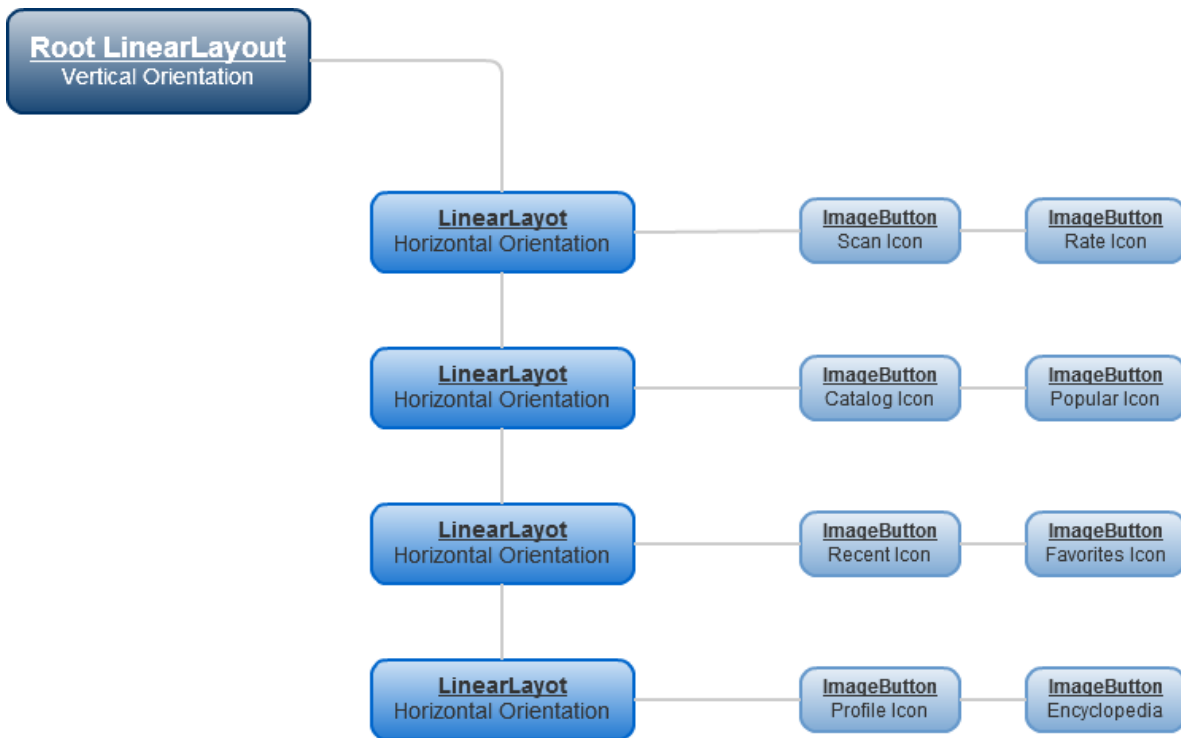


Figure 4.23: The diagram shows how to create the Dashboard interface used for the main screen of the wine app. It is a hierarchy of ViewGroups, the LinearLayots, and View Widgets, the ImageButtons

Inside the root layout is again four nested LinearLayouts which have their orientation set to horizontal. This results in the buttons to be arrange two by two in four rows.

The final result of the interface can be seen in figure 4.24.

Defining Remaining Interfaces

All activities in the app with an attached interface, is defined using the same principle as described above, using the Android SDK's build in widgets and layouts.

4.9.5 Barcode Scanning

The implementation of barcode reading will be based on a third-party implementation called “ZXing Barcode Reader” (ZBR), which utilizes the “open-source, multi-format 1D/2D barcode image processing library implemented in Java.” [56] by the same developers. ZXR can be used to read barcode’s in EAN8 and EAN13 format in the application because it can be easily integrated with an Intent to the ZXR’s scanning activity. It will return the scanned barcode number which then can be sent to the server.



Figure 4.24: *The final result of the defined interface from figure 4.23*

4.9.6 Communicating with the server

The server described in section 4.6 receives HTTP requests from its clients and makes a response to the client in JSON format. The application should be able to send these requests and parse the response to communicate with the server. The functionality of this will be encapsulated in one class and will be used throughout the application when requests are needed.

The time it takes sending requests and retrieving the response is a process that relies heavily on the speed of the currently active network connection of the device and the server response latency. Therefore all communication between the application and the server should be put in a thread for itself to avoid blocking the main thread, hence not hang the user interface.

4.9.7 Searching for a wine

The search for a wine can take two forms, either with a barcode or just a text search. As defined in section 4.6.7 a search request should contain a field called “filter” in the query string with all the features in the value as seen in listing 4.4. The response can either be no match or a list with the matches. If the user chooses to search with a barcode an Intent will be initiated that will start the ZXR activity and return the barcode the user scans.

The search request should contain a field called “filter” in the query string with all the features in the value as seen in listing 4.4. To make it easier to format that string a filter class was defined and will contain methods that add constraints to the wine to search for. For example the class contains methods like `addEan13(string value)` or `addVarietal(int value)` which take arguments according to the feature type. The class will have a method that returns a string representation of the filter field and its values and will be used when sending a search request.

The request is sent to the server and a response in JSON will be received specifying if there

were a match or not. If a match was found a list of all the matching wines should be shown where the user can choose from and if only one the information screen of that wine will be shown. Figure 4.25 depicts the search functionality.

4.9.8 The Wine List

A list in Android can be implemented with the `ListView`, see figure 4.26. The `ListView` is a view that are backed up by an “adapter” which job is to provide the data and views to be filled in the list, usually an array of items. Each item in a `ListView` are essential a view generated from the data that the adapter provides. To make a custom `ListView` one have to sub class the adapter class and override the method that returns the views that are visible on the screen. Putting a big dataset into a `ListView` can have an impact on the smoothness of the `ListView` when scrolled and therefore when implementing an adapter for a `ListView` you’ll have to deal with reusing resources such as the views. Android does this by supplying a view in the methods arguments that can be null if empty and not null if it can be reused. Reused vies are just views that has been instantiated before and still are in memory, so it is just the content of the views that gets replaced.

In the actual implementation of the wine list, views are getting reused and in addition, all photos for each wine item are downloaded on the fly in a separate thread so scrolling of the list are affected as little as possible.

4.9.9 User Login

In order to authorize users against the server a request must be sent with the supplied username and password. If the server authorizes successful, an authentication token are returned in the response and this token should be saved and used in all future requests that requires login. In addition the user id will also get returned which should be saved as well because it identifies the currently logged in user. The token and user id will be saved in the Android’s Shared Preferences that is only available to the saving application. In this way users can stay logged in even if the application is terminated because token and user id can be fetched on relaunch. The data will only be deleted when the user explicitly logs out and he will have to log in again. If the token and user id are empty then the user are not logged in. The other way around means that he is, so that can be used to determine the state of the user.

The Facebook login integration relies on the Facebook SDK for Android [57]. It function in the same way as the native login, by using an authentication token that are issued on a successful authorization with the Facebook servers. When a successful token is issued from Facebook, our server are contacted and if it is a first time user, a new user will be created in our user table, so the data for that user can be tracked and a new token is issued from our server to be used on the client.

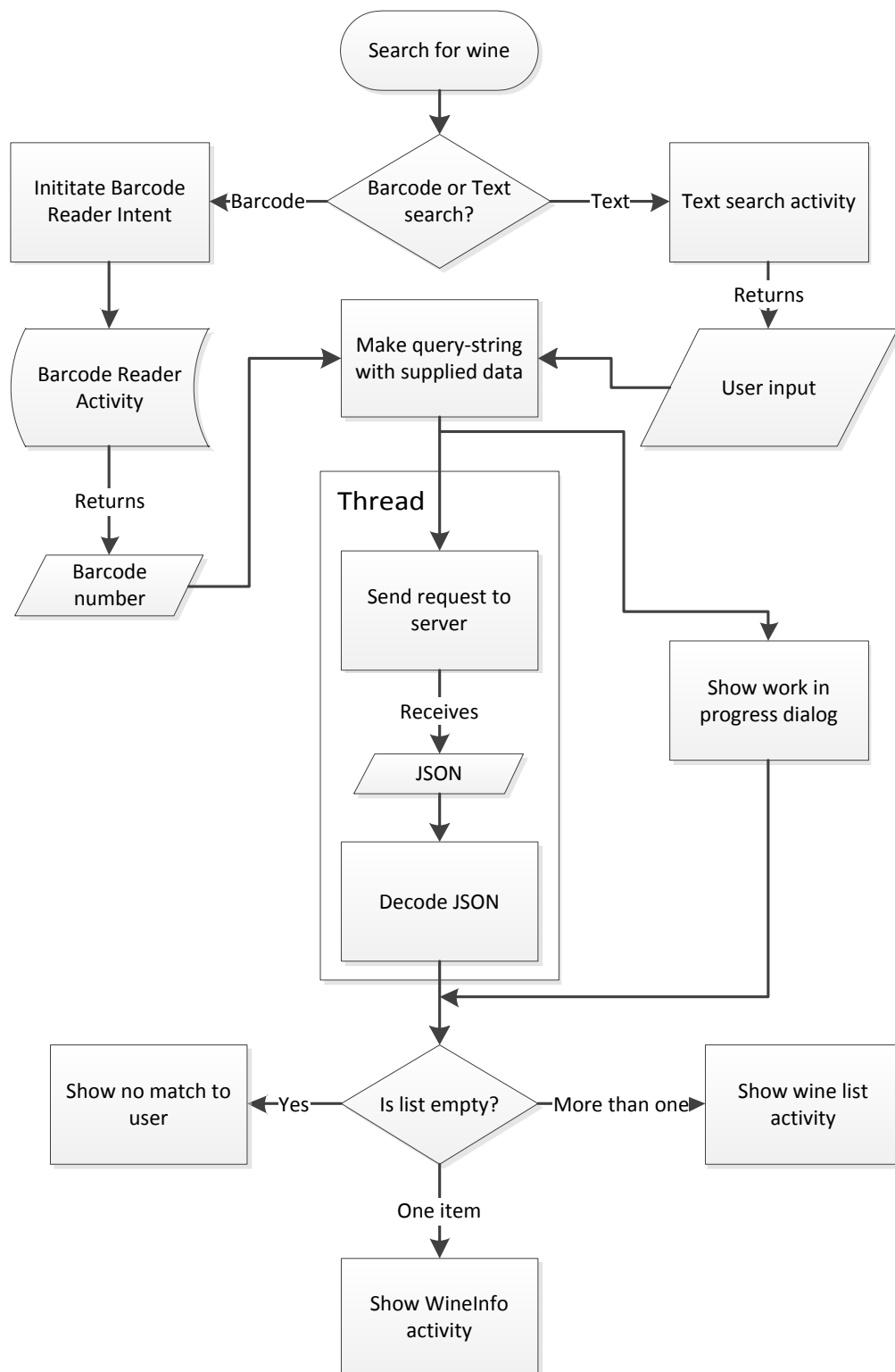


Figure 4.25: Shows a flowchart depicting the search in the application.

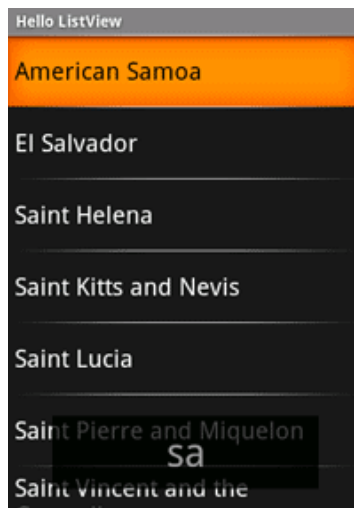


Figure 4.26: *An Android `ListView` with some items.*

5 First Prototype Overview

To get a better understanding of the implementation process, this small chapter will try to create an overview of the current state of the application to sum up on the implemented features done under the first iteration of implementation.

In figure 5.5 on page 126, the current state of the application, as it is present after the first implementation iteration, is illustrated. The diagram is based on the same diagram that was presented in the low fidelity design in figure 3.15 on page 74. The new diagram illustrates two important things:

First it illustrates which screens that have been implemented, and which screens that has not yet been implemented. Screens colored in green are implemented, which means that the navigation works, the layout is implemented, and tabs have been set up. Screens that are grayed out in the diagram, are screens that are not yet implemented or visible in the application.

Second, each feature on its respective green colored screen, is marked whether it has been implemented or not. Those features marked red and with a minus sign (-), are features that are missing or not fully functionally. Features marked in black and with a plus sign (+), are features that are more or less functionally.

Screenshots

On the following pages, screenshot taken from the wine application will be shown. It should give a clear view on the implementation process.

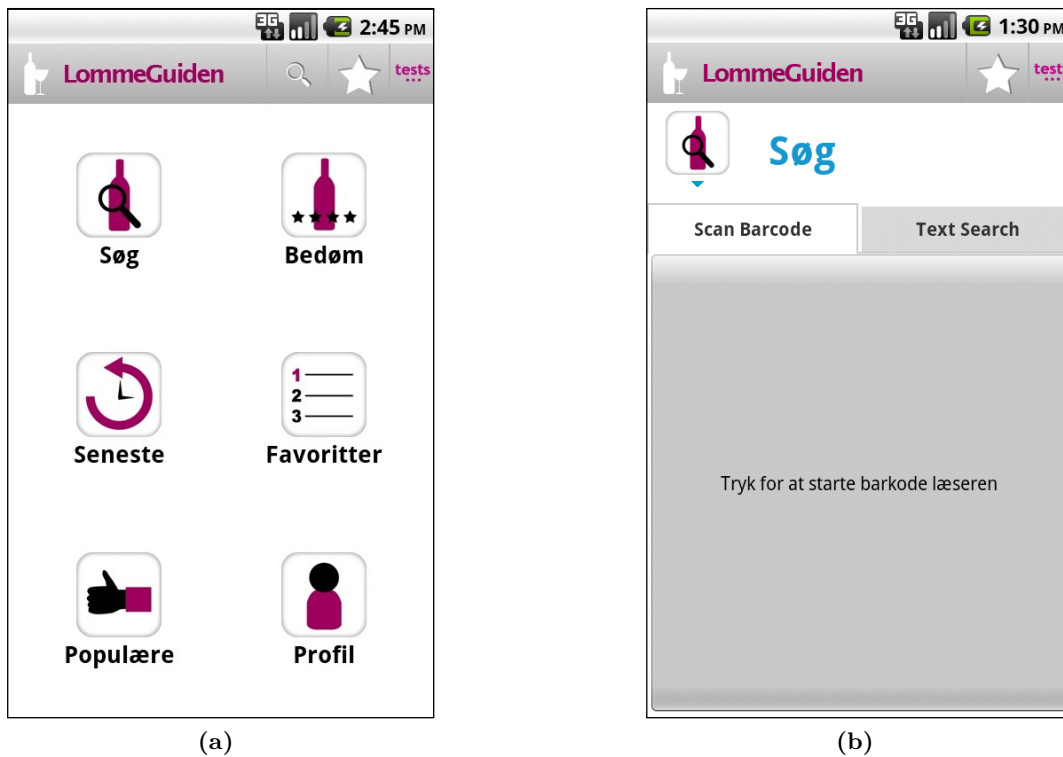


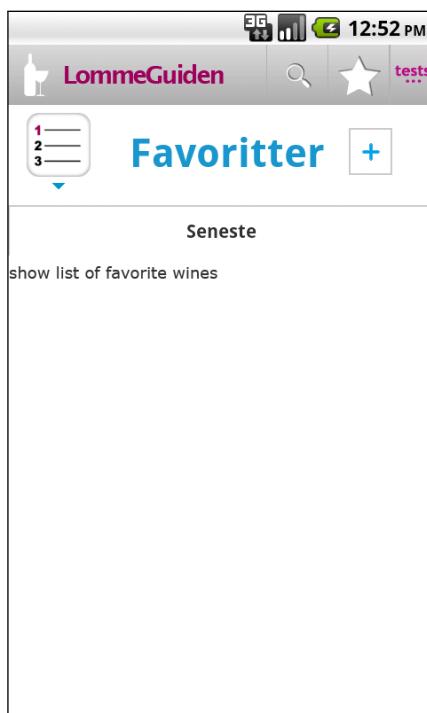
Figure 5.1: (a) The dashboard with the six icons and the actionbar at the top of the screen. The top rightmost icon named test located into the actionbar, is for test purpose and not a part of the design.
(b) The search view with the big 'Push to start barcode reader' button.



(a)



(b)

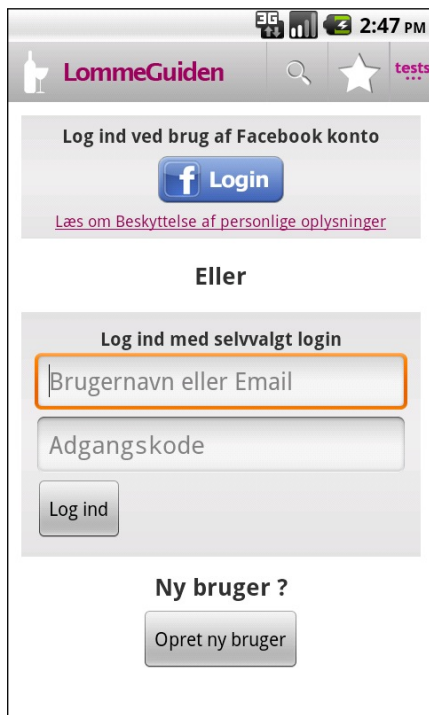


(c)

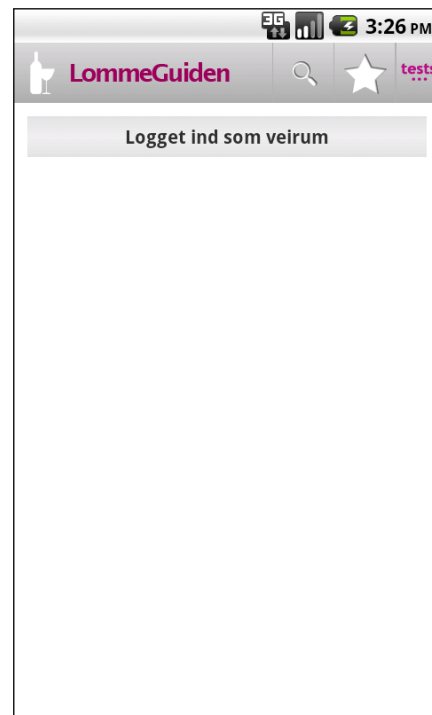


(d)

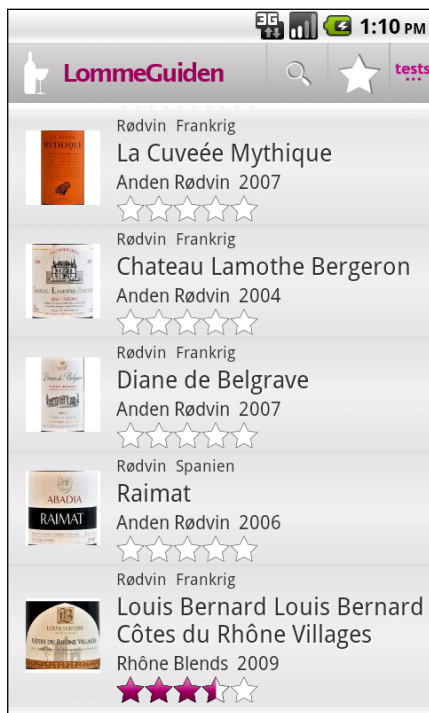
Figure 5.2: (a) The placeholder screen for Rate. (b) The placeholder screen for Recent. (c) The placeholder screen for Favorites. (d) The placeholder screen for Popular.



(a)



(b)



(c)



(d)

Figure 5.3: (a) The Login screen with the ability to use and existing Facebook account for login or create a new user and login. (b) The placeholder screen for Profile (c) List (d) Wine Info



Figure 5.4: You rate a wine by dragging your finger across the stars to fill them. When you have filled the number of stars you think the wine deserves. To submit your rating you press *ok* or press *cancel* if you do not want to rate the wine.

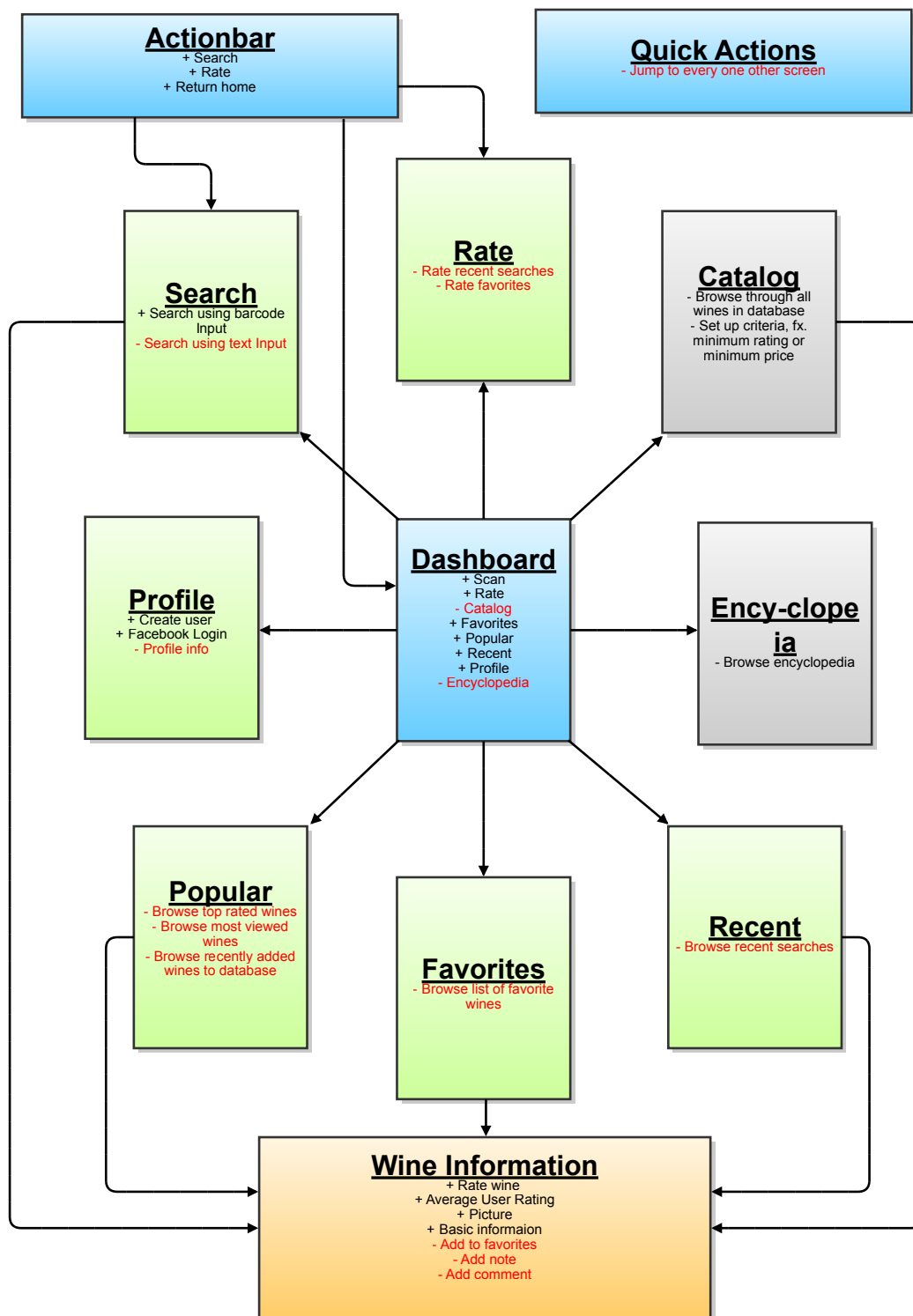


Figure 5.5: An overview of the implemented features following the first implementation iteration. Features marked red with a minus sign (-), indicates planned features. Features marked black with a plus sign (+), indicates implemented features. Gray boxes indicates features and screens not implemented at all.

6 Usability Test

The application will be tested in two iterative phases. The first test is a small usability test designed to catch the more severe faults in the applications, let it be functionality or user interface. The use of a lo-fi test was omitted because we wanted to have a running hi-fi model to avoid the often cumbersome wait that comes when a paper need to be replaced every time e.g. a button is pressed and a new screen is "loading".

6.1 Heuristic Evaluation

To evaluate the overall user interface design and the first implemented features described in the interface design, section 3 on page 49, a Heuristic Evaluation [19] will be conducted to verify the design.

Three to five evaluators should be able to find between 60 - 75% of the usability errors according to figure 6.1 which in this case should be more than sufficient. The most critical errors is believed to be found by nearly all of the evaluators and therefor the remaining 25% of the usability errors is considered a minor factor in the overall user experience and application functionality. The reported usability errors and bugs will be corrected in the next implementation iteration.

The evaluators chosen for the heuristic evaluation will be Medialogy Master Students from 10. semester as they should be considered experts in user interface design and usability testing.

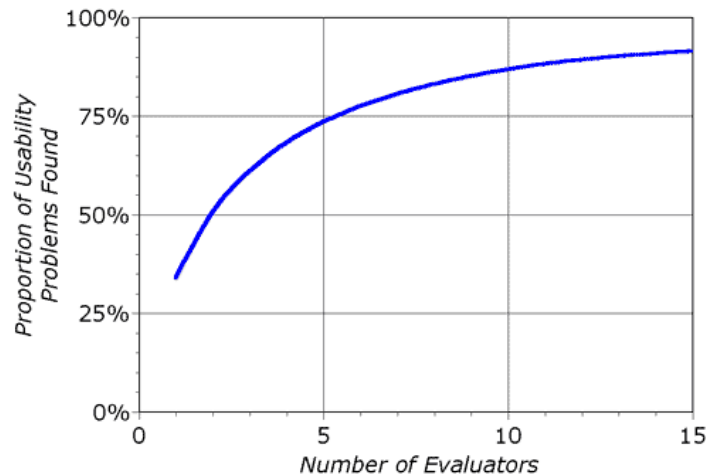


Figure 6.1: A curve showing the percentage of usability problems per evaluator found in an interface using heuristic evaluation[19].

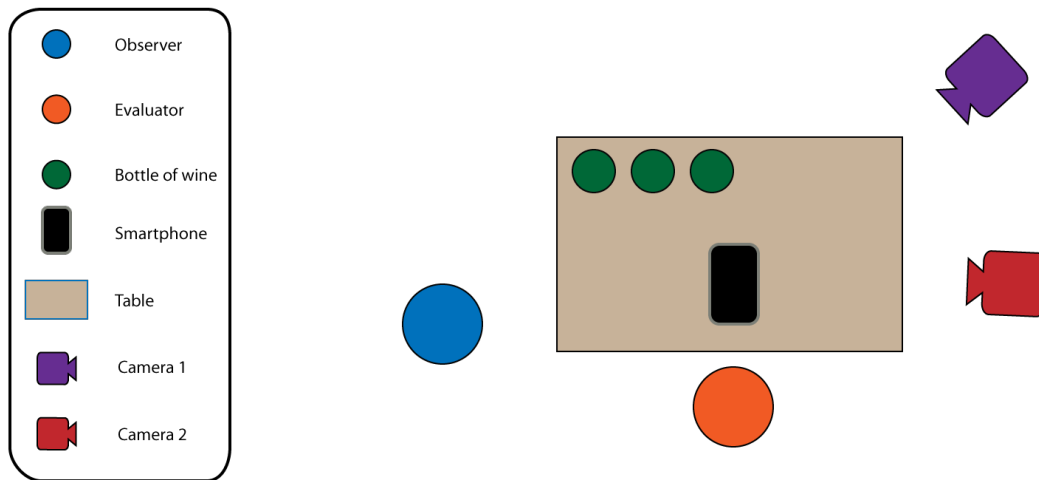


Figure 6.2: A drawing of the test setup. Camera 1 is filming the whole scene and Camera 2 is filming the screen of the smartphone.

6.2 Test Setup

The actual test will be conducted in a controlled environment. The setup consists of a table, three different wines and two cameras. An observer will conduct the test and guide the evaluators if any problems occur. He will note his observations on paper and ask questions to clarify actions performed by the evaluator if not obvious. The first camera will be used in conjunction with the Think-aloud method[58] so further examination of the video material can be used to extract relevant or hidden information there might have gone unnoticed. The second camera is focused on the smartphone's display. This is to see how the evaluator uses the user interface and to see if any difficulties arise when used.

The smartphone used for the test is a HTC Desire running Android 2.2.

6.2.1 Pre Evaluation and Application Status

The Evaluators will be asked to complete some pre-defined tasks designed to help evaluate the implemented features at the current state.

The first task

Sort three wines from best to worst. To complete this task the evaluators need to scan the barcodes of the wines using the application.

The second task

Login with an existing Facebook account or create a new user and login with that.

The third task

Rate one of the three wines. To rate a wine the evaluators need a user account to store their

personal data which was obtained in the second task.

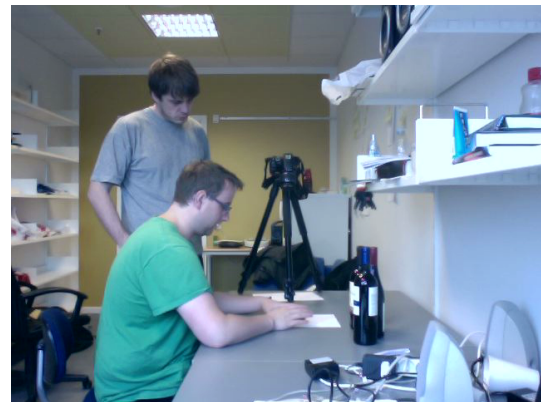
6.2.2 Test Results

The test results will be divided into three sections to make it more readable:

- **Usability.** Observations and comments about what is hindering a friction free interaction with the application.
- **Additional Features.** Comments about features or functions that could be added to the application to improve the experience.
- **Errors.** Typos and other minor errors.
- **General.** All other comments that do not fit into the above categories.



(a)



(b)

Figure 6.3: (a) A camera with a close up of the interaction. (b) A shot of the evaluator and observer.

Usability

- More space between the list entries' black separator lines to make it easier to separate each entry.
- The current implementation of the barcode scanner is confusing because of the big button. The button need to be small to resemble an actual pressable button.
- Textfield margins needs to be increased to avoid Text hugging the lift side of the device. It makes it hard to read.
- The panes has a hard look. Make them with more rounded corners.

- Maybe place the quick actions in the bottom of the interface to avoid possible occlusion caused by the user's hand. It may be too much with the quick actions at the top because of the shallow architecture. You are not deeper than two presses on the back button.
- The 'Bedøm' functionality and position on the dashboard not meaningful. The actual function is available when searching. Maybe just remove from dashboard.
- The recent icon confusing a redesign would be good.
- Switch the 'favorites' icon with the 'Popular' icon.
- Maybe call 'Recent' for 'History'.
- User ratings might need some kind of adjustment weighting to avoid few ratings to effect the overall rating too much while the number of overall ratings are low.
- Implement all physical buttons to avoid breaking consistency between apps.
- Show only the important information. Maybe get rid of entries like 'proptype' to avoid cluttering.
- Hard to see the change of the button text from 'Bedøm' to 'Bedømt' when rating. Need some kind of visual cue.

Additional features

- When the application is in background maybe have a shortcut in top bar for quick access.
- Profile need to have a log out function, and a lot of different stats about use like total number of rated wines etc.
- Be able to write public comments when rating a wine.
- The possibility to post a links to e.g. recipes on the dish suitable to the wine.
- Show the number of times a wine has been rated. It gives a possibility for the user to estimate how liable the rating is.
- Have some sort of compare function
- Show the average score and your own score at the same time for easy comaparison.

Common Errors

- The letter 'ø' was missing in some messages.

General

- The barcode reader was slow and took a long time recognizing the barcodes.
- The barcode reader's yellow and green dots can be confusing
- The popular will need a way to handle newcomers to the top 10 so you don't accidentally keep the same 10 wines in top 10.

Observations

Every evaluator used the rate button on the dashboard when they were asked to rate a wine. There were a difference in how iPhone users and Android users interacted with the interface. The biggest difference between the two approaches are the use of physical interface on Android devices compared to iOS devices.

6.3 Test Conclusion

The ability to scan the barcode of a wine and get the correct result from the database server worked as expected. Problems with the smartphone's ability to properly focus and thereby the time it took to scan a barcode seemed to be a technical problem. The application was installed on a HTC Legend after end test and the Barcode reader application was a lot more robust and faster which suggest that it could be a faulty camera. Further tests will reveal if it is the case or if it can be the curvature of the bottle or a weakness of the HTC Desire.

The Facebook login and user created login worked flawlessly. A possible problem occurred when trying to logout of Facebook. It where necessary to delete the application cache the completely logout. This is considered a minor problem at the current state because you can claim the user of the application may also be the main user of the entire phone and thereby it is rare you need to logout, but it still need to be examined further when time allows.

The placement and how to access to the scan and rate feature need some altering. The evaluators hesitated when trying to locate the scan feature because of the nesting of text search and barcode scan. When asked to rate a wine the evaluators used the rate button on dashboard which actually turned out to be a shortcut to two of the other features placed on the dashboard namely favorites and recent which is a shortcut to the wine info screen where you make the actual rating of the wine. The rating feature needs to become a lot more intuitive because it is an essential part of the application.

It were also discussed how the non implemented features represent on the dashboard could or should be used. One particular issue was the similarity of popular and favorites that could be confusing.

A redesign of the User Interface and some of the functions of the application will be described in Part III.

The test videos can be found on the attached DVD.

7 First Iteration Conclusion

The first iteration has undergone several phases. First the wine application was conceptualized and designed in chapter 3 on page 49. A lo-fi interface was laid out, based mainly on the seven design principles in section 3.3.2 on page 56 and the design patterns in 3.6 on page 66. The main application features were depicted in section 3.4 on page 61. Lastly it was analyzed how information about wine could be handled in this context and what information to cover. All together it formed the basis of the wine app, which leads to the first iteration implementation of the whole solution.

In the first iteration implementation in chapter 4 on page 87 the barcode technology of encoding and decoding an EAN-13 number was covered and it was decided to outsource the actual barcode reading to a 3rd party application available on the Android platform, namely the ZXing Barcode Reader. Relational databases were examined and it was covered how the wine database is implemented on the server, so wine information can be contained properly and how the database is maintained with a backend for administrators so adding wine is less error-prone. The client-server relationship was described with the use of the REST style utilizing the request-response communication on the HTTP protocol with the data format called JSON. Further it was also covered the various requests that the server supports. Lastly, the Android platform was examined from a developers point of view and the applications most significant functionality was described from a technical point of view. In figure 5 on page 121 a quick overview over the implemented features of the Android application can be examined.

In chapter 6 on page 127 a usability test was conducted with the heuristic evaluation method to find the most critical errors of the implemented user interface and core features. The test concluded that some restructuring of the user interface is required in order to maintain a more intuitive and pleasant user experience.

Part III

Second Iteration

8 Revised Interface Design

In this chapter, the application developed during the first iteration, will be evaluated and changed according to the suggested changes presented by the evaluators during the final test of iteration one. The changes will be held up against the design principles presented in the first design chapter of iteration one, section 3.3 on page 53.

This redesign will again be followed by an implementation, a test and a conclusion.

To sum up on the design principle a list is shown below:

- Clarity
- Aesthetic Integrity
- Consistency
- Direct Manipulation
- Feedback
- Metaphors
- User Control

8.0.1 Clarity

According to the usability test in chapter 6 the clarity of the dashboard needs improvement. There are especially confusion about the search's dual functionality, rating button and the popular functionality. To sum up, the dashboard from the first design can be seen in figure 8.7.



Figure 8.1: *An overview of the Dashboard as it looked after the first iteration.*

Search, Rate and Popular

To increase the clarity, search will be split up into two separate dashboard icons, namely one for the scanning barcode functionality and one for the text search functionality. This will make the two most important functions of the application a lot more visible and easy to find.

Popular will be removed as a standalone functionality and dashboard icon, but its functionality is incorporated into text search as a 'sort by' feature. The not yet implemented feature 'Catalog' will be merged together with text search bringing forth the possibility of criteria based text searches. This merged function will now be called Index.

The Rate icon will be removed as a standalone icon from the dashboard. The actual rate functionality is accessible from the wine info screen and therefore not needed as an icon. An illustration of this new dashboard can be seen in figure 8.2b. In figure 8.3, a sketch of the new Index functionality can be seen.

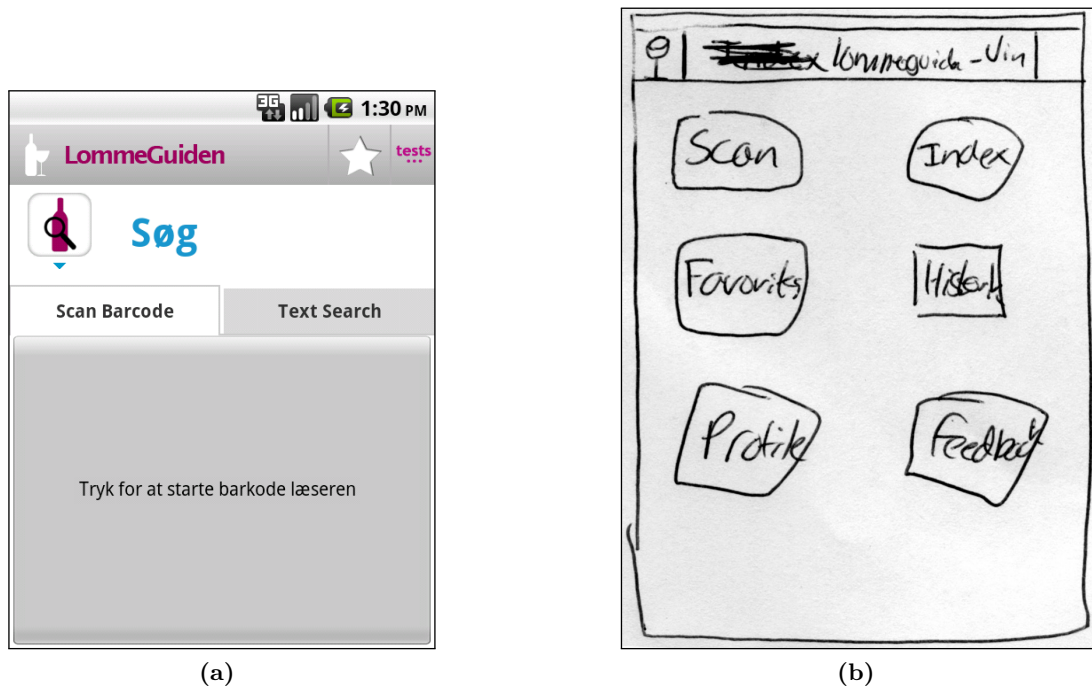


Figure 8.2: (a) The Search screen containing the "Scan Bar Code" and "Text Search" tabs, as it was implemented in the first prototype. (b) A mockup of the new dashboard layout after merging several functions. Top left icon called Scan, will lead directly to the scanner. The top right icon called Index is a merging of, text search, popular. It will feature a search based on search criteria.

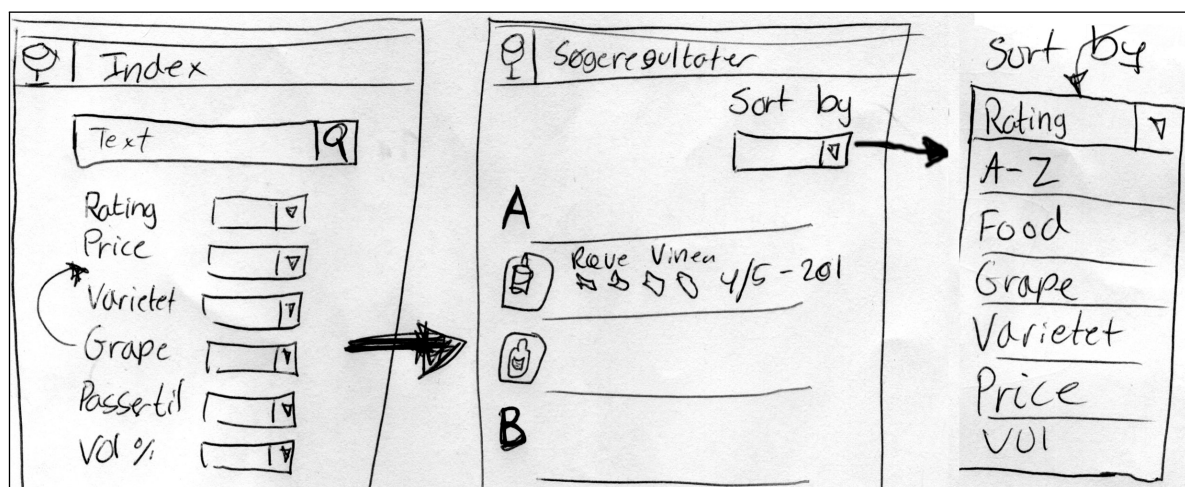


Figure 8.3: This illustrates the concept of the new Index functionality. It makes it possible to search via a keyword, and add criteria to the search. By sorting the search results, the functionality of the old popular functionality can be incorporated into the search.

Wine Info Screen and List

The ability to quickly see your own rating and compare it to the average rating, needs a redesign. During the test it was clear that it was a cumbersome task to press rate to see your own rating and then on cancel to see the average rating again.

To make it easier to see your own rating while in the wine information screen, in the redesign, two rows of stars will be present; one showing the total average of ratings, and one showing the users own rating.

A number of how many users that have rated a given wine were also suggested, which would give the user a much better feel of the reliability of the rating. A sketch can be seen in figure 8.4.

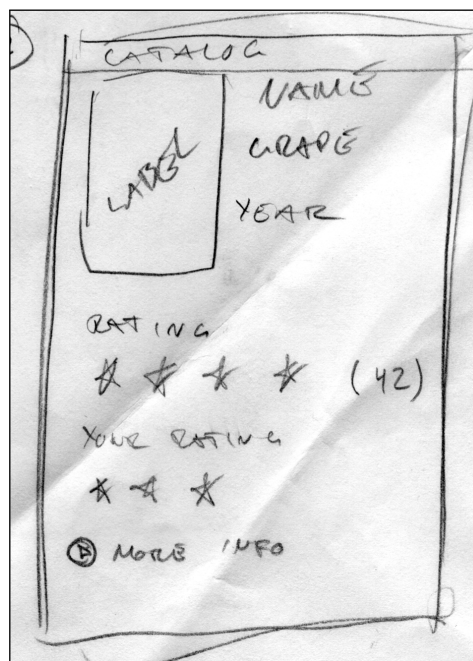


Figure 8.4: The new rating system with a separate user rating row of stars and an indication of how many people have rated a given wine.

The list view worked as intended. There were some small adjustments pointed out by the evaluators like the space between entries needed a little more air etc. To indicate if a user have rated a wine or added it to the favorite, two small icons is shown; a little green star if you have rated the wine, and a small black glass if it has been added as a favorite. These icons is added to help the user see if he has already rated or added the wine he is looking at in the list view.

Rate - Rated

The rate and rated button is difficult to read. This is changed from a push button with a label, to a row of stars which display the user's current rating. If not rated the row of stars

are empty and if rated some of the stars are filled. See figure 8.5 for the old design, and figure 8.4 for a sketch of the new design.

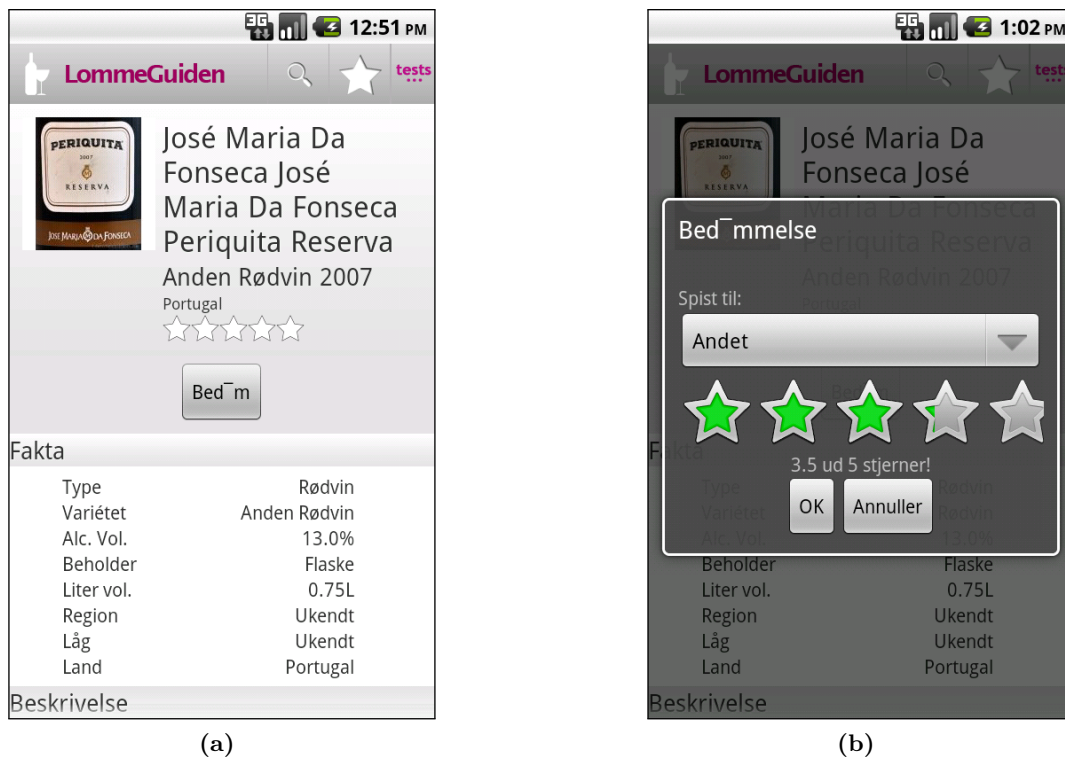


Figure 8.5: (a) The wine info screen with the rate button (old design). (b) When the rate button is pushed the rate dialog pops up (old design).

8.0.2 Aesthetic Integrity

The selection of color theme and general design idea of the icons got a lot of praise. The theme will therefore be continued in the coming iterations, but will be updated and have more polish.

8.0.3 Consistency

Among the evaluators were some seasoned Android users. They pointed out that there was a break in consistency. Some of the hardware buttons was not implemented which broke the flow and confused them a little. In later iteration they will be implemented to avoid breaking consistency.

It was suggested to place quick actions in the bottom of the screen to avoid occlusion by the users own hand, but to keep the application consisting to common Android design patterns the request isn't followed.

8.0.4 Direct Manipulation

The navigation through the application and rating wines by using of direct manipulation worked as intended. Every evaluator had no problems in grasping the idea and it seemed intuitive to all.

8.0.5 Feedback

The general feedback from the application did not get any response from the evaluator. This is taken as a positive sign meaning that the amount of feedback is sufficient at the current state. There was a single comment regarding feedback from the Barcode Scanner application during scan. When scanning a number of yellow dots forms and move across a red line indicating a laser beam. They can seem confusing, but we do not have control over that aspect in the current implementation and will therefore not look further into it.

8.0.6 Metaphors

Some of the metaphors used for icons did not work as intended. It was suggested to swap the favorite icon with the popular icon, but a group discussion lead to completely drop the two icons and make the favorites a wine glass to symbolize that a favorite is a wine you want to put in your glass. A more difficult task is to make a more understandable recent icon. There is no clear metaphor so the overall design of the recent icon will be kept but might be refined slightly.

8.0.7 User Control

No comments were given about feeling unsure of what the application was doing.

8.0.8 Additional Improvements and Features

Vertical Space

When using the application we discovered how crucial vertical space is on a small screen device. The current implementation has a separate title when in a sub-screen. To save vertical space the title bar will be removed as the overall application title bar will dynamically change title to the currently selected sub-view and thereby save a lot of vertical pixels. The removal of the screen title bar is illustrated in figure 8.6.

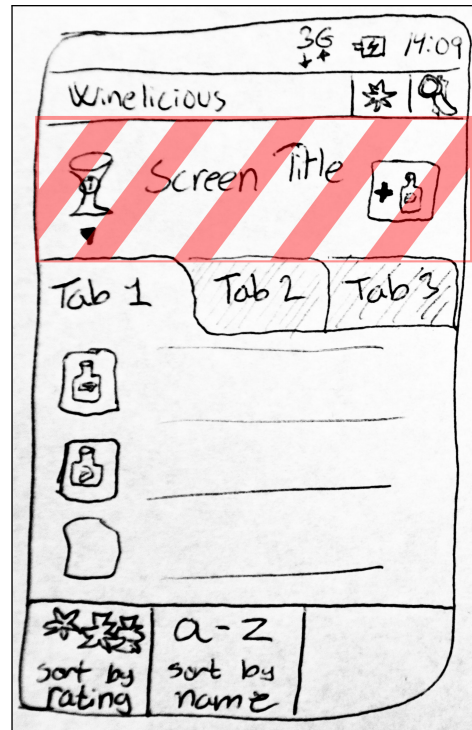


Figure 8.6: *In the redesign the screen title bar is removed to save space. The actual navigation of the app is not that complex, and therefore the screen title is not a severe loss.*

Add Notes

The ability to add notes to wines is a highly wanted feature among the evaluators. This feature will be accessible in the wine info screen. The note should be private for each user, in contrast to public comments, which will be described further on. The add note button can be seen in figure 8.7.

Public Comments

Another highly wanted feature by the evaluators is the ability to attach a comment to their rating to justify their decision like you know it from e.g. Amazon, or the different App Stores. This is a feature that should be added to the design.

Change of Android Platform

It came to our knowledge that nearly 25% of all Android users, see source [23], use Android version 2.1. The current implementation of our application is made to run only on Platform 2.2 and above. This will exclude nearly a quarter of all potential users. This will be changed to 2.1 in the forthcoming iteration.

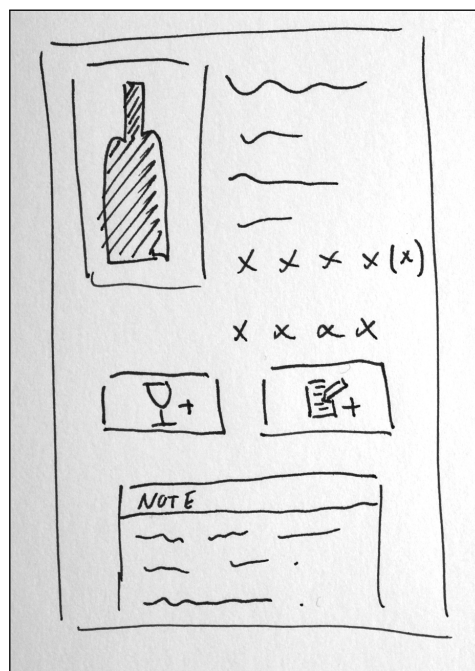


Figure 8.7: The add note functionality will be incorporated in the wine info screen. It is a button that will be placed along with the 'add to favorite' button, and is a button with some sort of note icon.

9 Implementation

The implementation process in the second iteration originates from the redesign in chapter 8 and will cover the most significant issues. Implementation details about every change or feature will not be covered as many of those are general and similar in nature of the implementation in first iteration in chapter 4. By that only the notes, favorites and advanced search interface will be covered in the following.

9.1 Implementing Notes and Favorites

The note functionality requires that a new table in the database are created that will keep track of the notes. The information needed in the table for a note is the user that owns the note, which wine the note belongs to and the note itself. The table and its relations are depicted in figure 9.1.

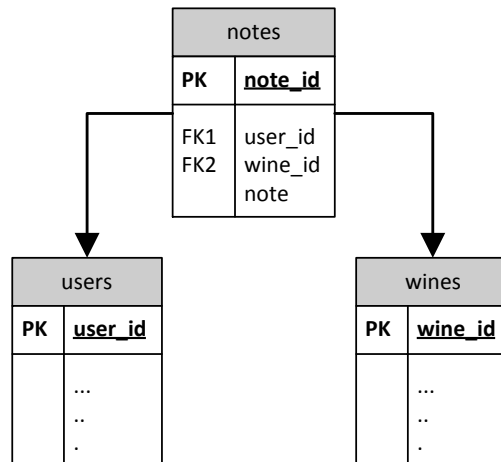


Figure 9.1: A database model diagram showing the notes table and its relations with the “users” and “wines” table.

9.1.1 Note requests

A note can be added, edited and deleted, but the requests can be reduced to two, because add and edit can be used interchangeably. If a note already exists for a wine when an add/edit request is initiated it should simply update the note and if not create a new one. The process is depicted in figure 9.2.

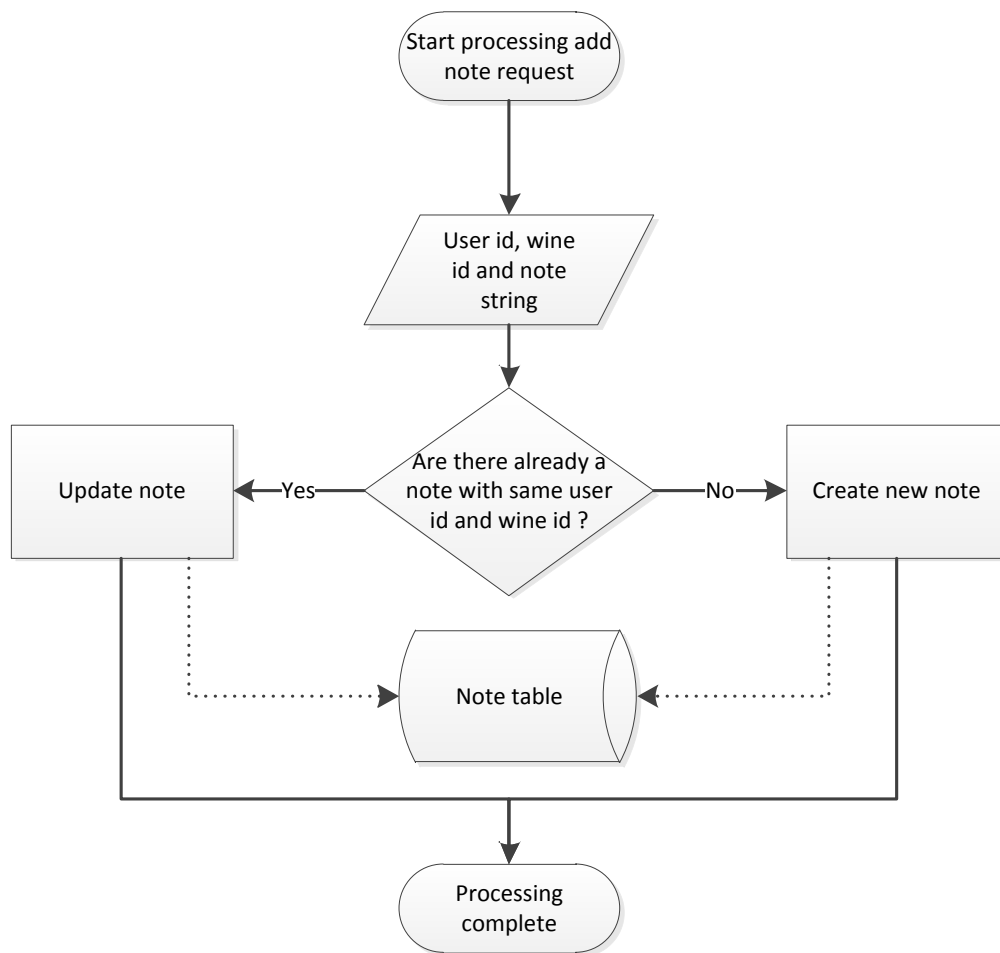


Figure 9.2: Shows the processing of the add/edit note.

The favorite's implementation is almost identical, but in the favorites table we only track the user and the wine's id. The request needed is also two: add to and remove from favorites.

9.2 Implementing The Search Screen Interface

In the app, the search screen is called Index, because it resemble a place where the user can browse through all the wines in the database. The user have the possibility to set up a list of criteria, for which wines they want to be displayed. The following lists the different criteria. There may be more criteria to choose from in future iteration.

- Match a text string

- Match rating range
- Match price range
- Match grape varietal
- Match year
- Match country
- Match alcohol percentage

In the left screenshot in figure 9.3, it is displayed how the final search screen look like. If the user wants to edit a criteria the corresponding edit button is pressed. This is the "Redigér" button on the figure. The button activates a popup dialog where the user can use two sliders to set a desired range of rating, price, year, etc.



Figure 9.3: (left) The search screen when opened from the dashboard (right) When editing for example the rating criteria the user, uses two sliders to set the desired range, in this case rating. The adjustment screen is a dialog popup which is activated when the users pres the edit (Redigér in Danish) button from the search screen.

The type and country criteria is a bit different, because it is not possible to adjust through sliders. Instead a popup with a range of checkboxes appears.

9.2.1 Creating Criteria Popups

As seen there need to be four popups that have sliders to adjust criteria, as the one seen in the right picture of figure 9.3, namely rating, price, year and alcohol percentage. The last two is popup witch a list of checkboxes.

As the two groups of dialogs is identical respectively, except from the text and numbers, it should be possible to reuse some code in a clever manner. It will now be described how the popup with the sliders, were made for optimal use.

9.2.2 The Dialog Class

To create a dialog, the already build in dialog class in the Android SDK, is used. Among many things, the dialog class can be provided with a title, an icon, its content, and its buttons. As standard, the content of a dialog is a text string which would be the text that the dialog will display. In this case however, a more complex content is wanted in the shape of the adjustments sliders.

To make custom content, the dialog is provided with a custom View, see section 4.9.2 on page 113 for a description of Views and ViewGroups. Inside the custom View, the sliders is defined. The sliders used is a standard widget from the Android SDK. A sketch of the dialog and the custom View can be seen in figure 9.4. The content from the dialog in the left of the figure is replaced with the custom View in the right of the figure.

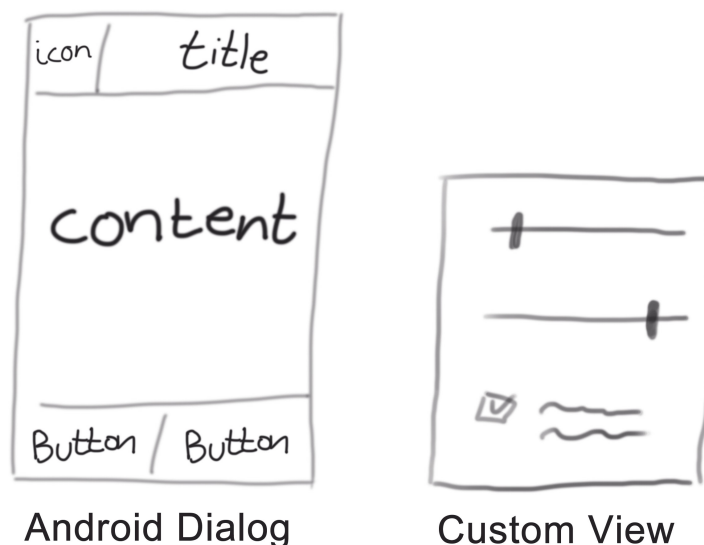


Figure 9.4: To the left is the Android dialog. The normal content in the dialog, is replaced with a custom View to the right. The custom View contains the adjustable sliders, a checkbox and a text field.

9.2.3 The Dialog Creator Class

As mentioned there is four nearly identical dialogs which contains adjustable sliders. To not have to repeat code creation of a dialog, a dialog creator class was established. The dialog creator class have a create method which returns a new dialog to be displayed. The create method takes in the required arguments; title, icon and the content view. A conceptual class diagram can be seen in figure 9.5.

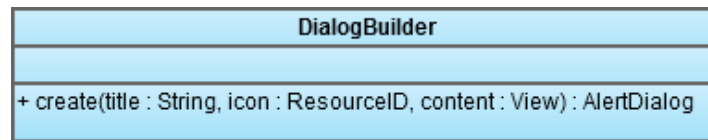


Figure 9.5: Conceptuel class diagram of the dialog builder class. The class returns a `AlertDialog` which can be showed as a popup dialog.

9.2.4 Custom View Base Class

For the four custom Views, an abstract superclass is defined which every custom view should derive from. By using an abstract class it is possible to define all the default layout and implementation shared by all the view. At the same time, it is possible to have abstract methods to ensure that the differnet views implement their unique properties. A conceptual class diagram of the custom view can be seen in figure 9.6.

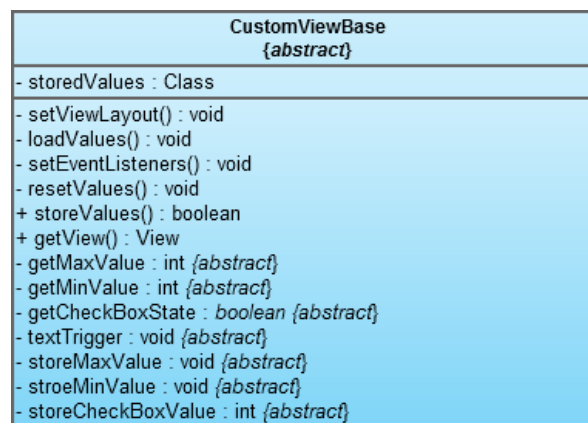


Figure 9.6: Conceptual class diagram of the superclass used for the costum view.

The storedValues variable, is a a global singleton class, which among other things, holds the current values of search criteria.

When the users pres the edit button to define a search criteria, a custom view should be created for the dialog popup. For the custom view, first the layout is defined, i.e. the positions of the sliders, etc., using the setViewLayout method. Next, if the user already have defined search criteria before, these values is loaded back to the view using the loadValues method. Lastly, event listeners is instantiated so that the sliders, etc., responds to user input.

The method, `resetValues`, is called when the event listener for the reset search criteria is pressed. The method resets the criteria to default.

The abstract methods implemented by its four subclasses, all process the data for each of the respective situation. For example, the range of the rating is between 0.0 and 5.0 floating point values, while the alcohol percentage should range between 0 and 100 interger values.

10 Second Prototype Overview

Similar to the prototype overview in first iteration, chapter 5 on page 121, this overview chapter is made to get a better understanding of the implementation process. The layout of the chapter is similar to the first overview chapter.

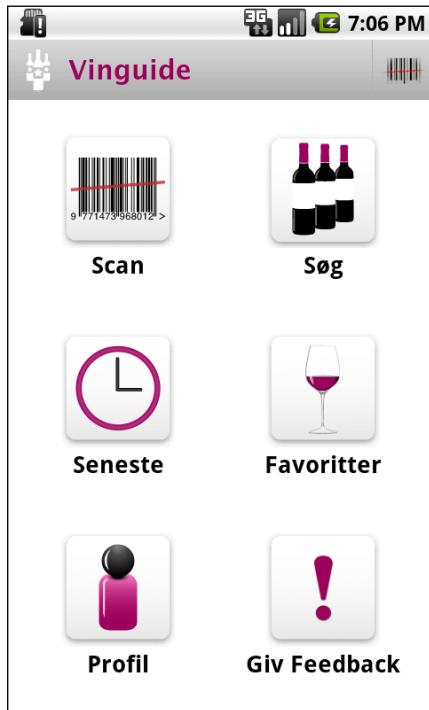
In figure 10.4 on page 155, the state of the application, as it is present after the second implementation iteration, is illustrated. The diagram is based on the same diagram that was presented in the low fidelity design in figure 3.15 on page 74. The new progress diagram illustrates two important things:

First it illustrates which screens that have been implemented, and which screens that has not yet been implemented. Screens colored in green are implemented, which means that the navigation works, the layout is implemented, and tabs have been set up. Screens that are grayed out in the diagram, are screens that are not yet implemented or visible in the application.

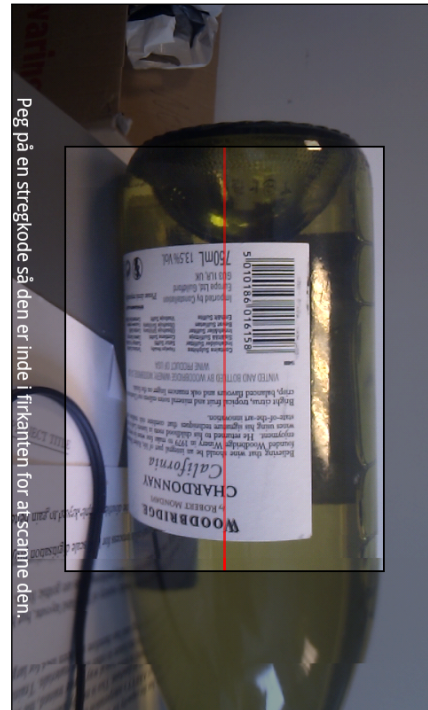
Second, each feature on its respective green colored screen, is marked whether it has been implemented or not. Those features marked red and with a minus sign (-), are features that are missing or not fully functionally. Features marked in black and with a plus sign (+), are features that are more or less functionally.

Screenshots

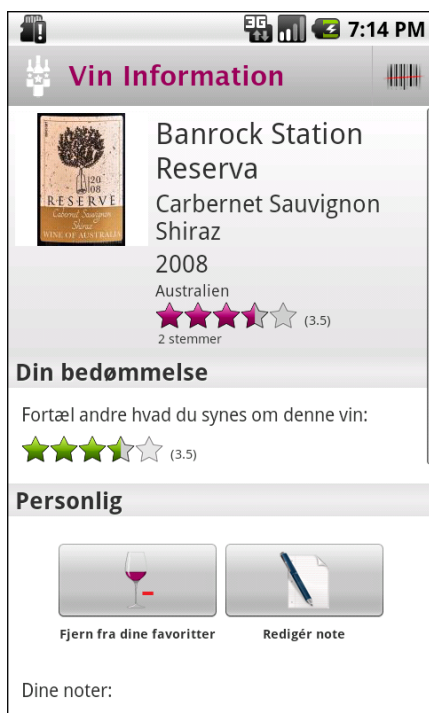
On the following pages, screenshot taken from the wine application will be shown. It should give a clear view on the implementation process.



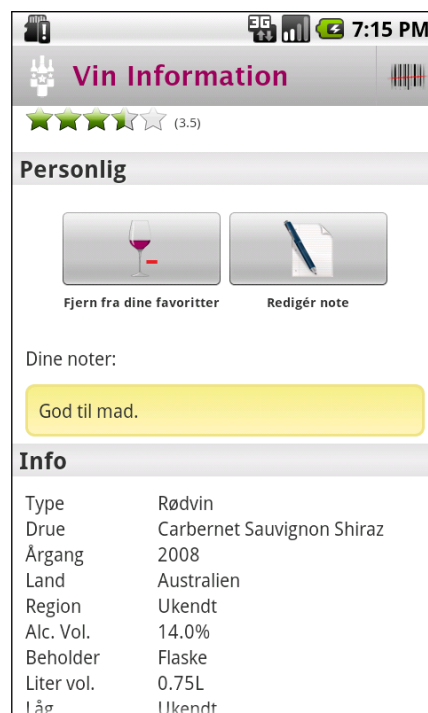
(a)



(b)



(c)

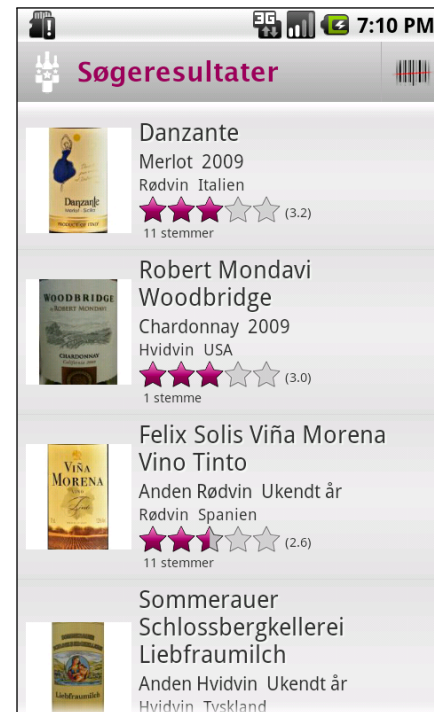


(d)

Figure 10.1: a) The redesigned dashboard layout. The Action bar now only consist of a single shortcut for the scanner. The feedback button in the lower right corner is for user feedback only and not a part of the design. (b) When scan is toggled you go straight to the scanner application ready to scan a barcode. (c) The wine information screen with basic information, user and average rating and the wine added to favorites. (d) The wine information screen with the additional info and a note added.



(a)



(b)

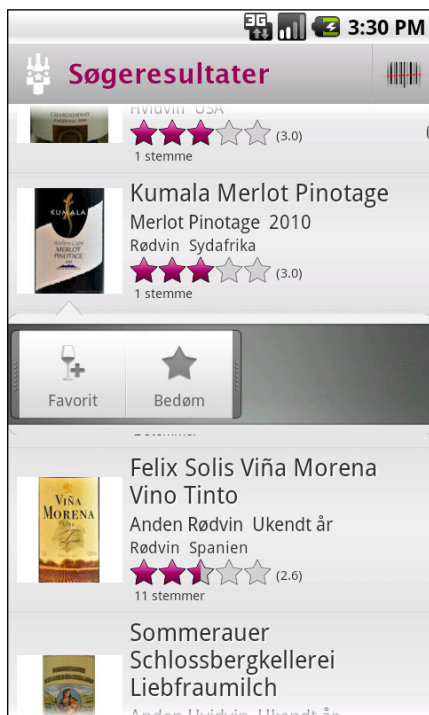


(c)

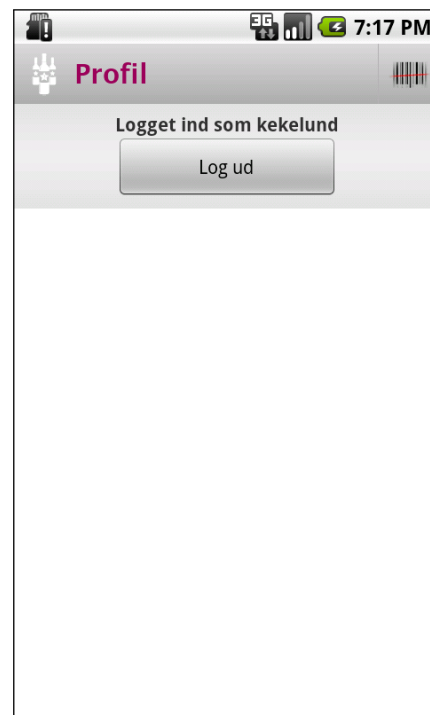


(d)

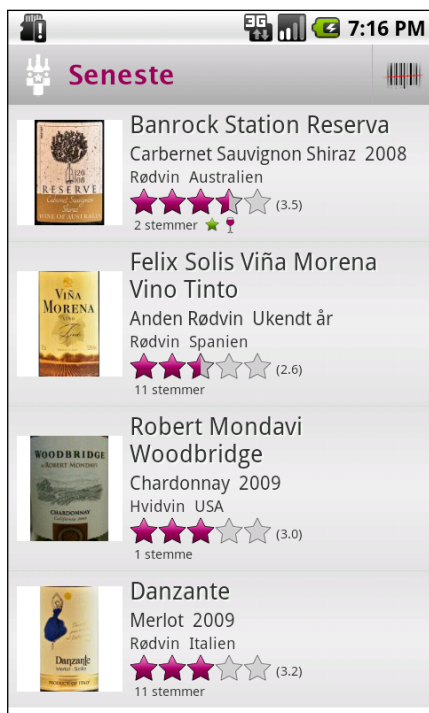
Figure 10.2: (a) Rating a wine swiping the finger back and forth to in- or decrease the number of stars. (b) The list view when more than one match occurs. (c) Text search and the possibility to narrow the search for example, rating, price or region. (d) The dialog popup when defining special search criteria.



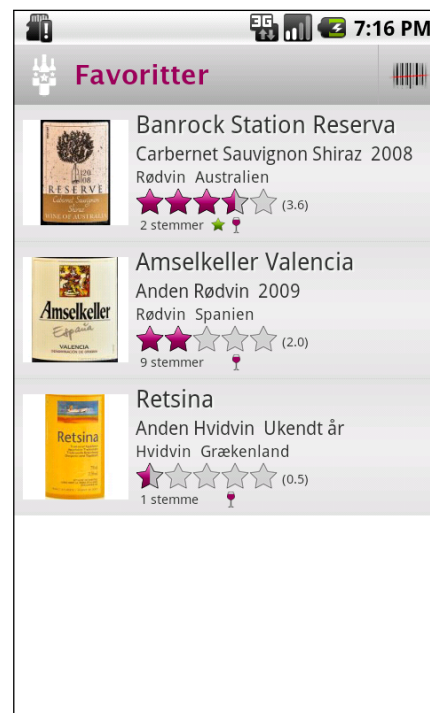
(a)



(b)



(c)



(d)

Figure 10.3: (a) How quick action look. The quick action pop ups underneath or above the wine when pressing the picture of the wine (b) It is now possible to log out of your profile. (c) A list of the recent scans or text searches. (d) A list of wines on the favorite list.

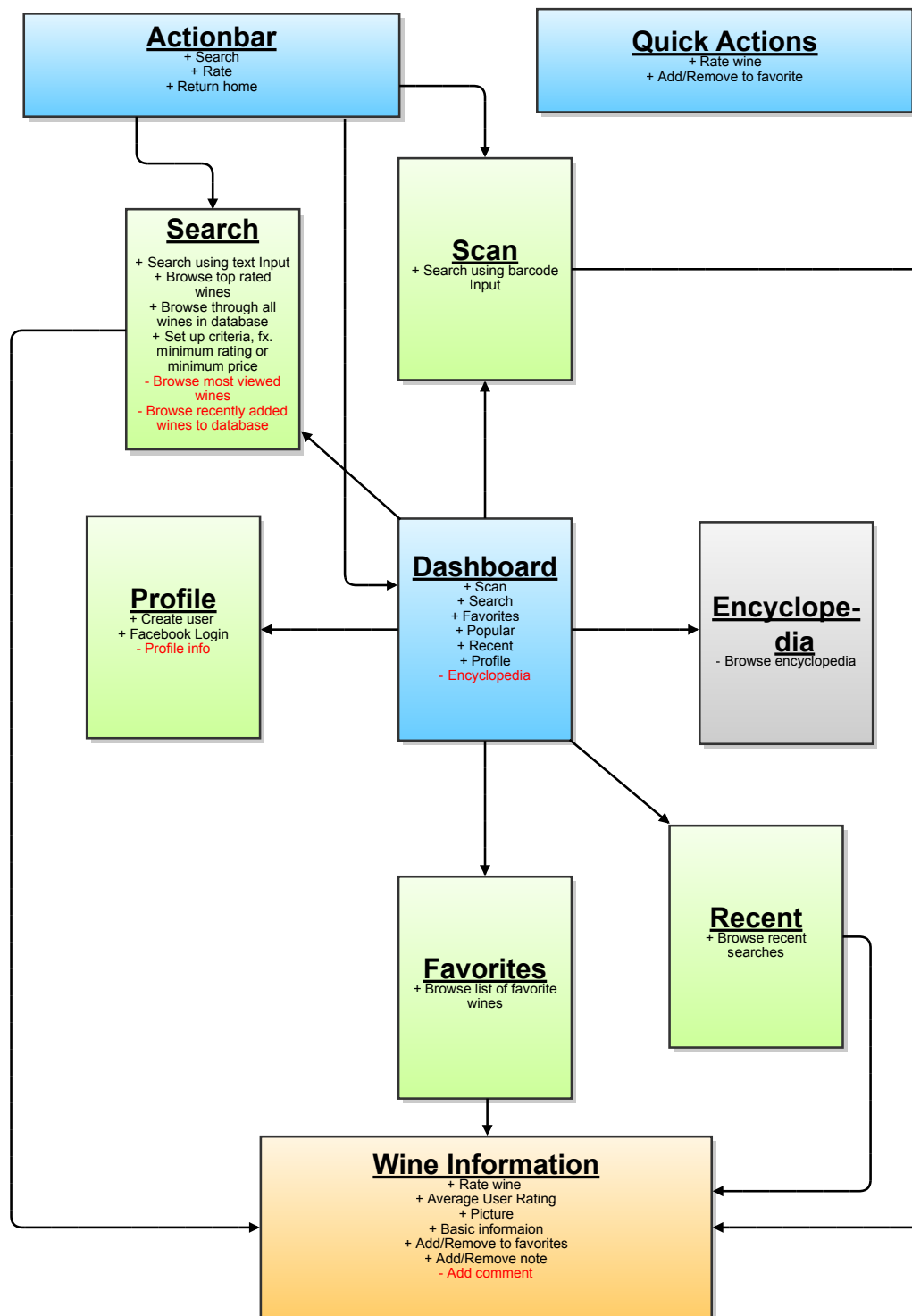


Figure 10.4: An overview of the implemented features following the first implementation iteration. Features marked red with a minus sign (-), indicates planned features. Features marked black with a plus sign (+), indicates implemented features. Gray boxes indicates features and screens not implemented at all.

11 Evaluation

This chapter will describe the second test that was made during the development of the wine app. The prototype used for the test is prototype number two, which inherit all implementations that have been done through the second iteration.

Test Scope

The test serves to prove that the observations made in the first test, section 6.1, and the following initiative on improvements, made during the design and implementation of the second iteration, has been successful. Many observations were made on faulty operations during the first test, and hopefully the new improved interface and interaction will do better.

Besides of testing the improved interface and interaction design, this test will also have a great focus on functionality and usefulness of the app. Many more features are working and fully functionally on the second prototype, and it is actually possible to use the wine app as a working product.

In addition the test will look at what the users think about the aesthetic design, general feel, and the speed of the app.

Test Participants

As this test will try to imitate real situations where the product could be used, people from the target audience is needed for the test. At least a few "real people" should be used, that is, people that is no experts. Just plain normal users, users that you could expect to get their hold on the app when the app hits the market.

Test Surroundings

The product is still in its prototype stage, especially the database, which only have about 150 wine entries. This constrains the test to be held in real surroundings, which would be at a wine store. Instead the test will be made at some more or less random locations where it is possible to get hold of a lot of test participants.

Though locations with very limited light are avoided to make the barcode scanner function probably, it was not sought to find perfect light conditions, as different lighting condition should be part of the test.

Test Equipment

The test was done on a HTC Desire, running Android version 2.2, with 3,7 inches screen. The HTC Desire has been a very popular smartphone, and hence a very good candidate for testing purpose. Connection to the Internet, for database access, was done through 3G connection on Telia's network.

11.1 Two Part Test

The test should try to imitate a real situation where the app could come in use. But with less than 150 wine entries in the database, and not a single valid rating of a wine, it is hard to imitate real use at the moment. Therefore the test will be split into two separate tests, where test iteration one will clear the way for test iteration two.

First to sum up on the scope of the test, here is a list of what is tried investigated through both of the tests:

- Interface and interaction
- Aesthetics and design
- Understandability
- Speed
- Functionality
- Usefulness

11.2 General Test Structure

Both parts of the test will be based on the same structure, but with different content and purpose.

Starting Information

As the first thing, each participant are told about the purpose of the wine app, not going in to much detail, and the purpose of the test. It must be assumed, if it has been a real life scenario, they would know what kind of app they had downloaded to their phone.

After the starting information, the participant is handed out the test phone. The participant is told that, at some point they will have to create a user, and that they can type in, any information they want, valid or not.

The phone starts at the phones home screen, and the wine app is not opened, hence the user have to open the app by tapping the launch icon. Although the participant are told they have to open the app, this serves to give the participant the full experience of opening the app and see the welcome screen. This way they won't feel like starting in the middle, or that the app has been tailored for the specific test.

Tasks

First the participants will be given different tasks that they have to complete. During the execution of the tasks, the participants is being observed to understand how they perform, and maybe asked a few questions about the task and the product.

Think Aloud

The participants will not directly be instructed to say loud everything they think, but is encourage to express their feelings and opinions when something is confusing or something is well done in the product.

Semi Structured Interview

Closing of the tests, the participants will take part in a semi structured interview. Semi structured interviews contain both open and closed questions and is thereby more strict compared to a normal unstructured interview. There is a script to follow, but the interviewer is allowed to come up with new questions. In this type of interview it is important not to preempt the answers by phrasing questions that suggest a particular expected answer as it could bias the interviewees answer. The benefit of this type of interview is that there is certainty that all topics are covered while still leaving space for unexpected topics from the interviewees.

Wine Tasting

Both tests was performed as a part of a free wine tasting. The wine tasting part was chosen because, that way it would resemble a real life situation the most. In addition it should feel much more natural and intuitive to use the functionalities of the wine app, contrary to using complete fictitious scenarios.

Five different wines were used for the test. They were chosen at random, but should range in price. The cheapest cost 16 Danish kroner, and the most expensive cost 68 Danish kroner. In the middle there were two wines of equal value that cost 30 and 31 Danish kroner respectively. The five wines can be seen in figure 11.1.



Figure 11.1: The five different wines used for the wine tasting in both parts of the test. They are arranged from lowest price to highest price. From the left in Danish kroner: 16,- 30,- 31,- 48,- 68,-

11.3 Test Part 1

The first part of the test will focus on the first four items in the scope listing, namely, interface and interaction, aesthetic and design, understandability, and speed. The part will however not disregard the rest of the items in the scope listing.

This part will also act as a database builder, where users begin submitting opinions about wines for the user generated content. The data generated for the app in the first part will be used to create a more realistic environment for the second part of the test.

The test should have about 5-10 test participants. That way, most interface and interaction issues should be found [19], and should give a good indication if the new improved interface and interaction work better. In addition, this amount of participants should give a fairly good amount of submitted content for the second part of the test.

11.3.1 Test Setup and Tasks

The five wines used for the test was lined up, so that the participant could study the wines as they please, at any time. This can be seen in figure 11.2.



Figure 11.2: *The lineup of wine as they were used during the test*

Task 1, Taste and Rate

For the first task, each participant is asked to take a taste of all the five wines and rate them using the wine app. The participant are also encouraged to play around with the app as they please.

Hidden task: When the participant want to rate a wine, or use any other functionality that require login, the participant have to create a user in database.

Purpose of task:

This task serves to prove that the user find the search intuitive, that the wine information screen is understandable, and if the rating mechanism is intuitive.

In addition the user creation is tested, especially if it is an annoying necessity.

Task 2, Mark Two as Favorites

When the participant want to proceed, they are asked to choose the two best wines and mark them as their favorite, and then add a note about the wine.

Purpose of task:

This task would prove if the favorite and note mechanism is intuitive and working.

Another very important purpose of this task however, is to see if the intent of having a recent functionality is working. The goal is to have the user intuitively, use the recent functionality, instead of searching for the same two wines twice.



Figure 11.3: *Picture taken during part 1 of the test*

Task 3, Use the Text Search

The last task, the participant is asked to find the best wine from Spain with a minimum of 10% alcohol.

Purpose of task:

The task is done to test understandability of the search function, using text and different criteria.

11.3.2 Questions and Observations

In the following, the result of the test will be presented. Every question is written in *italic*, and after each question, the interpretation of the answers and observations will be written.

Gender, age and occupation.

This part of the test had 6 participants, all students aged in the mid twenties. They were all studying a programme equivalent to Medialogy.

Although the group of participants is not the best distribution inside of the target audience, they should be able to track down most interface and interaction flaws, and certainly they can create valid ratings for the wines.

Which mobile phone do you have?

This is important, because their experience with a smartphone can have influence on the performance of the test.

Half of the participant had a smartphone, and two had smartphones running Android operating system. During the test, this did not cause much of a problem though. Only minor problems that you would expect, such as the user not knowing how to go back.

Do you know of any other wine app for smartphones?

No one of the participants knew of other wine apps.

What do you think about the design, colors and the general theme of the app?

Generally all participant were very positive about both, design, colors and theme. The colors in particular were praised as being nice and pleasant.

What do you think of the icons? Does they make sense?

All participant thought the icon design was very self-explanatory. They all liked the design of the icons and one even praised the icons for being "fantastic!". One thought the icon for the favorite was a bit unclear, but it should not be a problem after being accustomed to it. Half of the participants used the scan button placed at the actionbar intuitively after a couple of searches.

Did you have any problem in navigating the app?

All participants found the app very easy to navigate. No major problems at all during the test were observed, only minor confusions which you should expect when trying an all new interface. One wondered why pressing the headline on the actionbar made him return home.

Do you understand all the information given on the wine information screen?

All participants found the wine information very understandable and clear. The order at which the information is presented was also very good, and overall the participants agreed that the most important information was placed at the top. One thought the rating mechanism was a bit confusing, as he thought the area to activate the rating pop up was a bit too small.

Do you understand all the information given on the wine list? Do you understand the small icons?

To recall, the wine list is the list occurring when having multiple matches, and the small icons indicate if a user have rated or made a wine his/her favorite.

All participants found the information understandable and all could tell what the small icons indicated after a short amount of time. One participant though, found the small wine glass a bit odd for the purpose of a favorite symbol.

How was the speed of the barcode scanner?

The scanner performance varied a lot during the test, which lead to half of the participant thinking it was a bit too hard to focus and the other part of the participant really excited of the good performance.

How was the speed of the app, when retrieving information and when navigating?

The speed of retrieving information from the database also varied in performance. Usually the downloading was very fast, but sometimes it would halt for too long time before showing a result. This again lead to some of the participants thinking it was a bit too slow at moments, while other said it was "astounding fast!".

Is there some functionality that you think is redundant? Do you miss any functionalities?

Two participants asked for price information. Two thought it should be possible to see where the wines can be bought. Two wanted to have food pairings.

Two participants thought the functionality was great and that the most important was present. One would like a wish-list, like the YouTube "Watch later" functionality. One thought it could be cool to have a functionality that could show what friends liked.

One actually asked if there could be some sort of wine encyclopedia.

Did you have any problem when, rating, adding to favorites, adding notes?

All participants found it understandable, and no problems were observed during the test, all participants completed the tasks very fast.

If you had a smartphone, presume that the database contained all possible wines, and there was a good amount of ratings on the wines. Would you use such a program in your everyday life?

Three participant thought they would certainly use the app. One would use it, if it was free. One would use it if there was a bit more information, such as price comparing and food pairing. And one would only use it on occasions where he could not find the wines he would normally look for.

Would you trust such ratings?

All the participant think they would be influenced by the ratings and would trust them. However, it seems important that good ratings on a wine actually results in a good experience. Otherwise the credibility for the ratings will fall. In addition, for most participants, the rating will mostly be used as a basis and will not be paramount.

Do you think it was acceptable that you had to create a user?

One thought it might stop some from using the program, but overall they all thought it was okay.

Other Observations For Part One

The first part of the test was clearly a success and no participants seemed to have any problems navigating or understanding the app.

For task number 1, where participants have to taste and rate wines, they were also told that they could play around with the app as they liked. It was very interesting to see how much they actually played around on their own. Most of the participants seemed very interested in the capabilities of the app, and many of them actually completed task number 2 before it was even laid to to them.

The curiosity and their independent handling of the app, indicates that, as much as being easy to use the app is astounding intriguing and funny.

For task number 2, where the participants should add two wines to his/her favorites, it was also a success. All but one, used the recent functionality pure intuitively, to find wines multiple times and add them to the favorite.

For task number 3, where the participants have to use the advanced search mechanism, all seemed to more or less understand how it functioned.

Problems and Enhancement Observed in Part One

The biggest flaw revealed by the test, was the inability to access all personalized actions. For example, all participants but one, wanted to have their own rating displayed when a list of wine is present. That was especially true when showing the list of favorite wines, not having to open every wine entry to see personal ratings. Another example, was the inability to see all the wines that the participant had rated.

The speed performance was another big flaw. Although, most of the time, both barcode scanner and retrieving information executed extremely fast, there was a varying performance, and sometimes the wait time was tedious.

Another flaw, arise when the user had to create a user, and the "Create User" button is placed at the bottom of the screen. Many of the participants wrote their email and user name in the login field, before having a user, thinking that they were creating one. This results in the user having to write it twice.

The participants also reported it annoying, when submitting ratings, adding to favorites, etc., they had to wait for the operation to finish. These operations could potentially operate as a processing-thread in the background, and just give a notification when it was done.

Many of the participants asked for some more social aspects. For example the ability to read comments made by other users. Another very exciting feature would be the ability to match the likings of wines across users, meaning if two users have rated the same or more wines high, they might share the same taste of wine. A system could then recommend wines to one user based on what users that share his/her taste of wine have rated high.

11.4 Test Part 2

The second part of the test will make use of the data created in the first part, to focus on the two last items in the scope listing, namely, functionality and usefulness.

Since there now is a valid average rating on the wines used in the test, it is possible to get closer to imitating a real life scenario. The second part will try that, and by that hopefully reveal the true usefulness of the app.

11.4.1 Test Setup and Tasks

For the second part of the test, the five wines are again lined up at random at a table. Their respective price is written on a paper and placed in front of each wine. The wines on the table should depict a wine store.

After each task the participant answers questions marked in each different segment corresponding to the task. The questions will be explained in the following section of test questions, 11.4.2.

Task 1, Choose a Wine as Normal

The participant is asked to think of the last occasion where he/she was about to buy a wine. Subsequently, the participant is asked to choose a wine, from the five wines on the table, matching the criteria for the specific occasion.

Purpose of task:

First, we wanted to know how the participant choose their wine on normal occasions. This would indicate if the wine app could be useful for them. Second, the choice on wine will be compared on the choice of wine in the subsequently task.

It was important to make the participant create the scenario for the choice on wine. In that way they were not influenced on their choice in any way.

Afterwards the participant is asked the questions from segment 1, see questions afterwards.

Task 2, Explain the Four Topmost Icons

The participant is asked to explain the expected functionality of the four icons in the top, namely, Scan, Index, Recent and Favorites.

Purpose of task:

The primary purpose of this task is not to confirm the understandability of the icons, though this is of course a side effect, but really is to secure that the participant acknowledge the possibility of the app. This way the participant will be aware of the recent and the favorite functionality, and hopefully the participant will use those functionalities out of pure intuition in later tasks.

Afterwards the participant is asked the questions from segment 2, see questions afterwards.

Task 3, Make New Choice on Wine Using the App

The participant is asked to make the choice on a wine again, using the wine app as a guide, with same criteria as in the first task.

Purpose of task:

It is sought to see if the participant embrace the new tool, the wine app, and uses the ratings and information to choose the wine. It does not matter whether the participant choose the same wine or not.

Afterwards the participant is asked the questions from segment 3, see questions afterwards.

Task 4, Taste and Rate

If the participant want to, he/she can taste the different wines, rate them, and play around with the app.

Purpose of task:

First, it is not all participants that want to drink wine during their work hour, hence they should have a choice. Second, the amount of time the participant use to play around on their own, can indicate how much interest they have in the app.

Afterwards the participant is asked the questions from segment 4, see questions afterwards.



Figure 11.4: *Pictures taken during the second part of the test*

11.4.2 Questions and Observations

In the following, the result of the test will be presented. Every question is written in *italic*, and after each question, the interpretation of the answers and observations will be written.

After the initial mandatory questions, questions in each segment will be presented. Each segment of questions were asked after the corresponding task.

Gender, age and occupation.

This second part of the test had 5 participants, 4 students and 1 professor. The students were aged mid twenties to late twenties and studied the humanities. The professor was 53 years old.

It was important for the second part of the test to have some less technical participants.

Which mobile phone do you have?

Three participants had a smartphone, while only one was running Android. As with the first part of the test, it did not seem to cause any problems using the app.

Do you know of any other wine app for smartphones?

No one of the participants knew of other wine apps.

Segment 1, asked after task 1:

How did you make your choice of wine step by step? How do you normally choose your wine?

Nothing new here. The participants all have similar patterns where price, own experience and recommendations, played the biggest role.

Segment 2, asked after task 2:

Do you understand the 4 topmost icons? Can you explain them?

All participant easily explained the functionality of each icon.

Segment 3, asked after task 3:

Did the wine app and the ratings have any influence on your choice?

All participant agreed that the rating influenced their choice on wine, and helped them. It was also clearly that, the wine that was a bit cheaper than the original choice, suddenly became interesting. But the participants also indicated that there need to be a good amount of votes before it would really matter.

Did you feel better qualified to make a choice?

All participants felt they were more qualified to make the choice when using the wine app.

Do you generally trust ratings as those seen in an app like this?

In general the participants would trust user ratings, but a good number of votes is again important.

Segment 4, asked after task 4:

What do you think about the design, colors and the general theme of the app? Did you find any problems navigating?

Again, all participants found the app easy to navigate and the design and aesthetic feel was thought of as pleasant and nice.

Was there anything in the app that you did not understand?

There was a slight confusing when searching and rating for one of the participants, though it probably had to do with that she never has used a smartphone.

Were you satisfied with the speed of the app?

One thought it was a bit too slow at some points, while the rest reported very good speed.

As in the first part, the speed of the app was a bit varying.

Would you use the app if able?

The general response were really positive and all reported they would use it. The responses was more positive than with the first part, which probably has to do with the less technical mind of the participants in the second part.

Other Observations For Part Two

As in the first part, the participants seemed to navigate the app very easily. It was clear that they think it was very exiting concept, and they thought it was funny to be able to just scan a wine to find it. Overall the participant seemed to have fun and were smiling.

For task number 3, it was observed that the user was very interested in what rating the different wines have gotten, and whether they should change the choice of wine.

For task number 4, nearly all participants played around the app on their own, adding wines to their favorites and rating wines, even with the knowledge that they would hand over the phone in a couple of minutes.

Problems and Enhancement Observed in Part One

Of new problems raised other than what was experienced in the first part, it seemed a bit confusing to add wines to favorites. It was observed that when trying to make a wine a favorite, the participant would open the favorite screen from the main screen. From the favorite screen the user cannot add favorites, only show those already added.

11.5 Conclusion Whole Test

Both test were very successful. The navigation seemed to work flawless for all the participants, and almost all the information and functionality was understood by all participants. Even

those who were not accustomed to a smartphone, did not seem to have many problems using the app.

Things that needs to be addressed in relation to the app are:

- Possibility to see personalized actions. For example showing personal ratings when showing a list of wines, i.e. in the favorite screen and recent screen.
- The User screen need a small redesign, so that users not accidentally write information in the log in field, thinking that they are creating a new user.
- Have submitting's run as a thread in the background, so that it does not interrupt navigation.
- Maybe a button from the favorite screen to add a wine directly.
- The varying speed issue needs to be addressed.
- There were minor technical issues that needs to be addressed, but overall, all functionalities operated very promising.

The conclusion of the test as a whole, is that the app as it is presented now, works very well, and the usefulness is present to a great extent.

But, to be a perfect app, the app will need more functionalities and need to display more information which from the result of the test could be:

- Price comparing
- Food pairing
- Available stores
- A "Wish List"
- More social aspects, like what friends like, and taste pairings across users, based on ratings.

12 Second Iteration Conclusion

The second iteration may seem like a small one compared with the first iteration when reading, but in reality the wine app have undergone some major improvements. The theories and methods used in the second iteration is much similar with the first iteration, and therefore does not need just as much documentation. For an overview, here is a list on what has been made through the second iteration.

- Update and polish of the aesthetics and icons
- Update and polish of the general layout for a pleasant feel
- Update and polish of the wine information screen
- Update and polish of the wine list screen. For example the indication of a wine that have been rated or added as favorite
- Ability to add, delete and edit notes
- Ability to show recent lookups and searches
- Ability to add and remove to favorites
- Ability to make criteria based searches
- Ability to log out from app.
- New interface and interaction as per the redesign
- Primitive chache search which caches previous searches for faster identical look ups
- Quick actions implemented and functioning
- Welcome screen. Added help and info screens to guide the user.

The evaluation of the second prototype was a clear success with only minor problems to the interface and interaction. People seemed to like the application and wanted to use the application. This means that the redesign have been a success.

Still there are planned features that were not implemented. It is learned that polish take a huge amount of time. To get an application that have the feel of being finished and polished, a cut down on features was necessary. This means that the encyclopedia feature for example, again have been postponed to a future iteration.

The most crucial missing thing, is the lack of information in the database. Without a full database, the app can still only be used as a preview prototype.

Part IV

Closing

13 Release on Android Market

The second iteration was ended with the release of the wine app to the Android Market. In this process it was also decided for a final brand name for the wine app, namely “Vinguide”. The release marks a milestone in the development as it was part of the success criterias for this thesis. It was released as a beta version on the Android Market, and contains an extra feature where users can give feedback on what they think about the concept, UI or what ever they might want to add as feedback. Also the Android Market, features detailed statistics about the released applications which might be interesting.

13.1 Statistics

Vinguide was released on 19. April 2011 and the following numbers was captured 26. May 2011 and will be referred to as release and capture date respectively.

Since the release there has been a steady download rate with an average of about 15 downloads per day and a total of 544. Further it can be tracked how many installs are active, meaning how many who has not removed it again. There is 333 active installs or 61% of the total. The total active installs from release to capture date can be seen in figure 13.1.

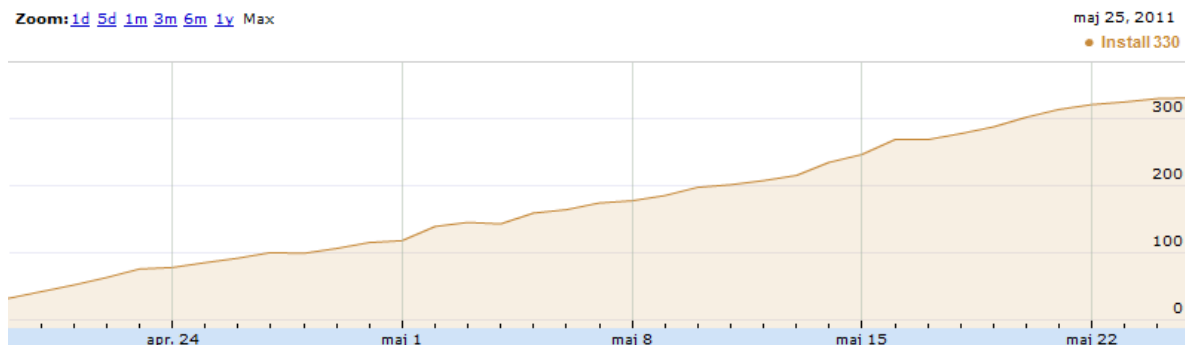


Figure 13.1: Shows a graph illustrating the total active installs from 19. April 2011 till 26. May 2011

The release was done without any additional advertisement, so Vinguide would only be found if users searched or browsed specifically for “Vinguide” or related searches which might be wine related words or part of the Vinguide description. From this it might be inferred that there is some kind of interest for this kind of application on the Android Market, even without any advertisement.

In contrast, the feedback received has not been overwhelming and the received can be said is issues that we already are aware of, namely that the wine database is not sufficient enough, and results in empty matches most of the time when the user scans a wine. The number of

search requests made to the server with Vinguide is 1100+ and indicates that users are trying to find matches, but unfortunately the majority returns no match.

In addition, devices and countries distribution among active installations can be seen in 13.2 and 13.3, respectively.

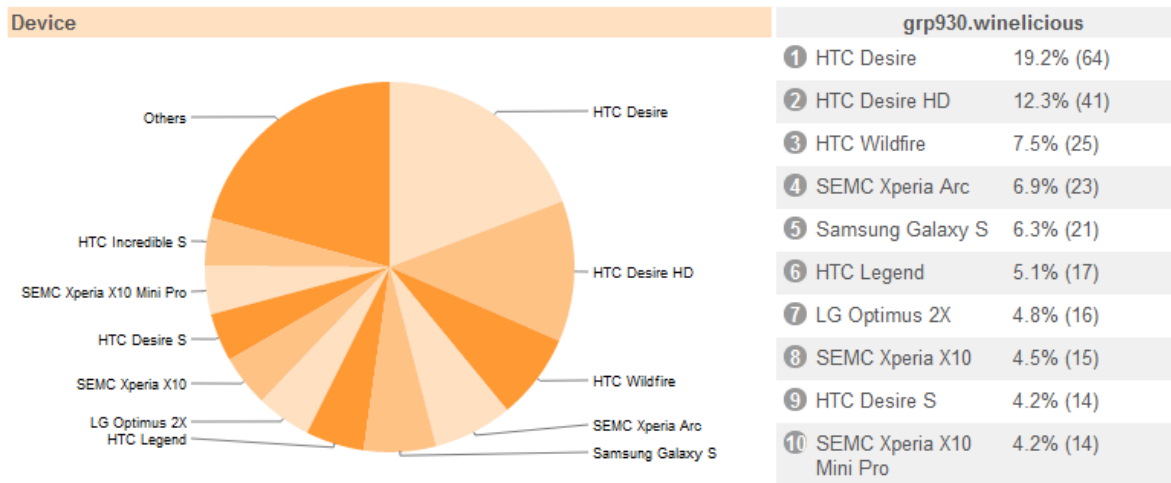


Figure 13.2: Shows the distribution among devices with an active installations of Vinguide

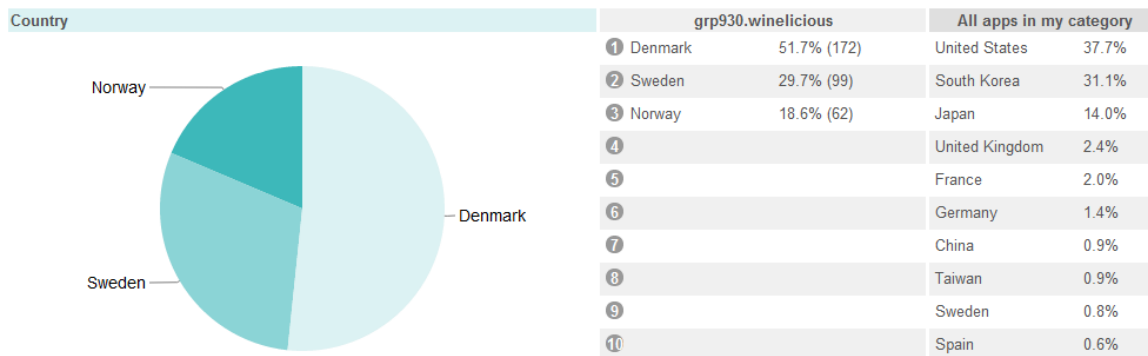


Figure 13.3: Shows the distribution among countries with an active installations of Vinguide

14 Future Improvements

Because of the time constraint of this project a third iteration is not possible. New ideas, suggested changes, features and improvements gained during the second user test, see chapter 11, will be described in this chapter. Note that this isn't a strict design chapter, but more a sum up.

14.1 User Rating in List View

During the test it became clear that your own rating needed to be visible in the list view as well, as on the wine info page. This addition would require a bit of rearrangement of the current list layout. A suggestion to the rearrangement can be seen in figure 14.1.

14.2 Add to Favorites

To address the possible confusion of how to add a wine to the favorite list, as discovered in the second iteration test, chapter 11, two possible solutions have been discussed. The first solution would be to add a 'add to favorites' shortcut within the favorites view. When you press the shortcut, the scanner starts and the wine you scan is automatically added to your favorites list. The second solution is to preset the user with a 3 step guide the first time they enter the favorites view, or as long as there are no entries in the favorites view, see figure 14.2

14.3 Login and User Creation

During test, subjects were asked to create a profile. They accidentally began typing their information in the wrong fields, namely in the login field and not the "create new user" field. They simply did not see the "create new user" button further down the screen. On small screen devices it is a cumbersome task to enter a lot of text like creating a user. It is absolutely essential that this kind of mistakes is avoided because it is likely to scare off potential users. See figure 14.3 for two alternative methods.

14.4 Wine Buddy

Taste is a highly individual sense just like people's individual taste in movies or music. A function called Wine Buddy would recommend wines that might fit your taste by looking at relations among users. That could be looking at the users favorite lists to find users similar to

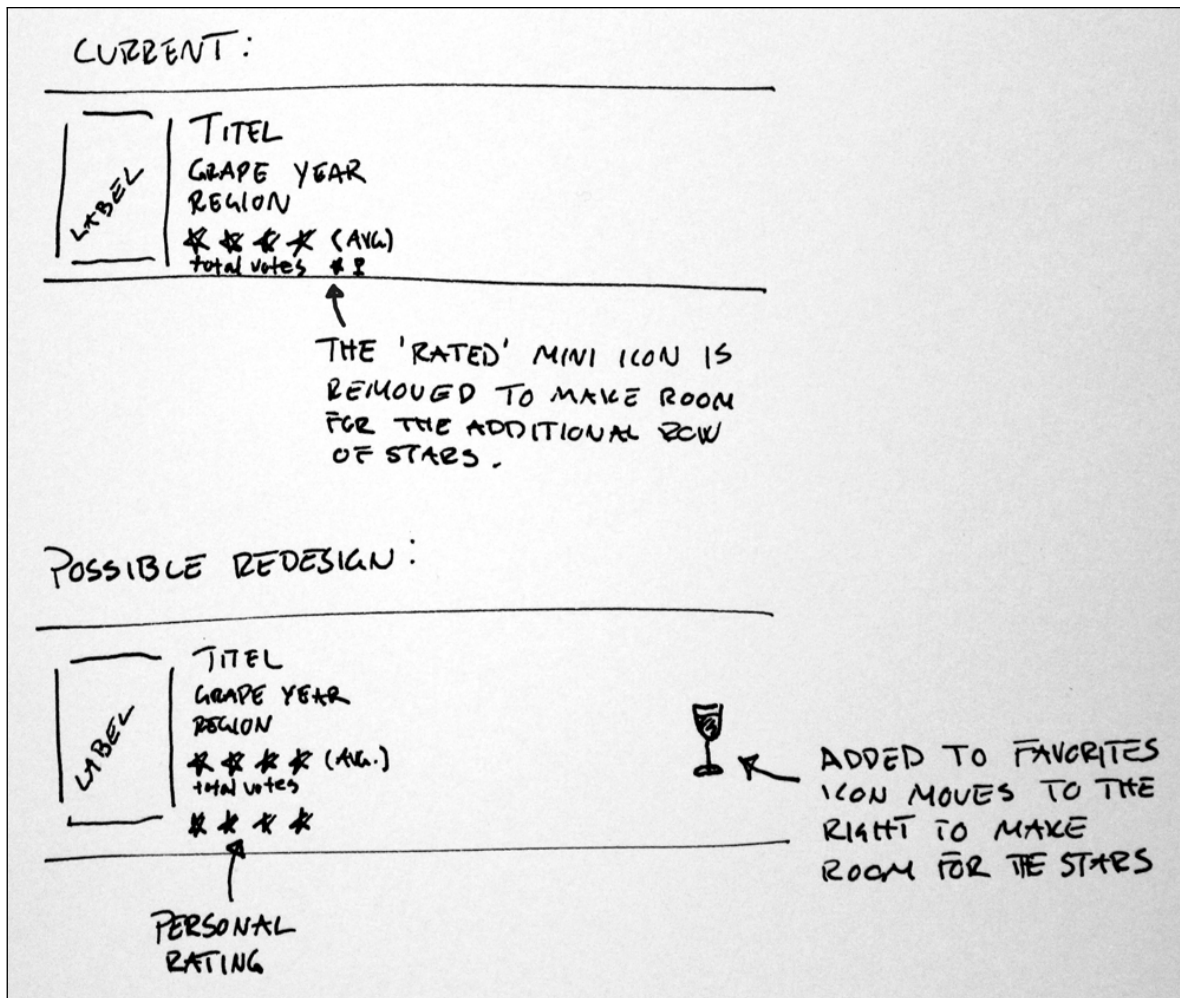


Figure 14.1: To make room for an additional row of stars representing the users individual rating the mini icon for added to favorites gets moved out right and the row of stars obviously replaces the rated indicator.

each other and then recommend wines from each other favorite list. Amazon.com uses a kind of similar feature to suggest e.g. books. When browsing books there is a field in the bottom of the screen that show books that other people bought who also bought this particular book. A feature like this could encourage users to try wines they may not know, but still be quite sure that if they buy a wine suggested by wine buddy they would buy a wine they most likely would enjoy.

14.5 Adding Wine and Suggesting Changes

There is one big thing missing from the application, and that is the ability to add a wine that does not exist in the database. This has been confirmed by the feedback given by our users of Vinguide that was released on Android Market since April 2011. This feature is an extremely important feature of the application. It was decided early on, in the first lo-fi design that

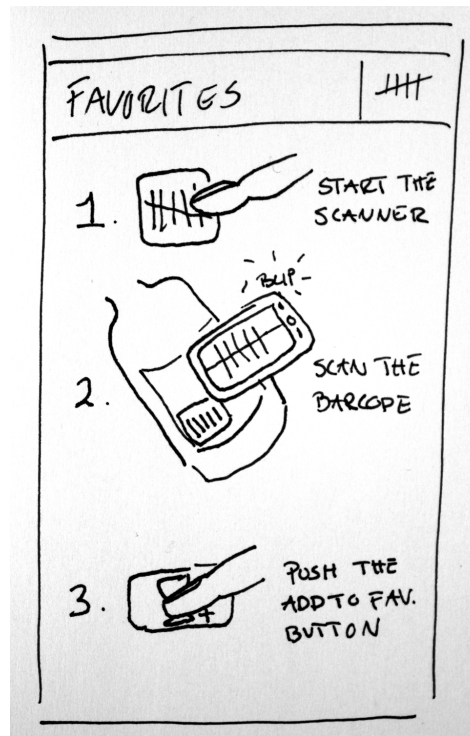


Figure 14.2: One image

this feature had to wait because of its complexity. Waiting with this feature gave us enough time to implementation and testing on a more complete application containing the majority of features like recent, favorites, barcode search, text search and criteria based search.

The possibility to add wine to the database from within the application is obviously the next feature that has to be designed and implemented. With the feature in the application it can be a lot more self-sustainable giving the users the possibility to improve on their own experience and contribute to the database.

If the user gets a 'no match' message when scanning a wine, he should be presented with two options. The first option would be to scan again, the second option will be to add the wine to the database. If the user choses to add the wine it is crucial that it is fast and fluent otherwise the user simply decide that it is not worth the effort and quit the attempt. In figure 14.4 and figure 14.5 is a sketch with an example of how i could be done.

Opening up for the possibility to add information to the database need certain precautions. There is a lot that can go wrong when opening up the database for the public to edit. There are the more obvious issues like misspelling, missing or misplaced information, and bad quality photos of labels and so on. To counter that it should be possible for the users to make changes in existing wine entries to correct such errors.

To avoid the more serious issues like vandalism the system need a way to handle such actions. It is clear that this feature contains many layers and therefor is one of the more complex implementation in the application.

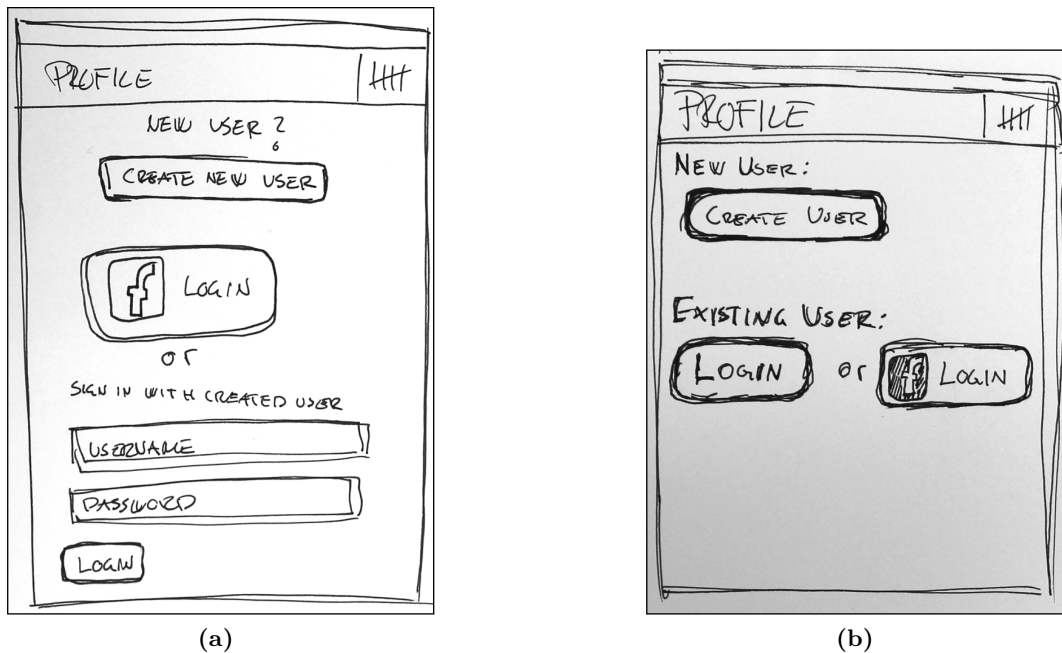


Figure 14.3: (a) This proposal positions the “create new user” button at the top instead at the bottom. To clarify that you need to create a user to login. (b) This proposal eliminates any text fields where text can be wrongfully submitted and time and effort wasted until either create new user, login or Facebook login are selected.

There is different ways that can be thought of to optimize the feature. Auto fill when typing is one possibility. Another thing could be that other users have to accept corrections and submitting's from users. It could also be a rating like system where user can like or dislike the information submitted from other users.

Lastly, because the barcode is used, many information can actually be extracted from the bacode itself. Recall from the section 2.12.4 on page 42 about barcodes, that the numbers depict the country, manufacturer and product. This means that it might be possible to auto fill some information solely based on the barcode.

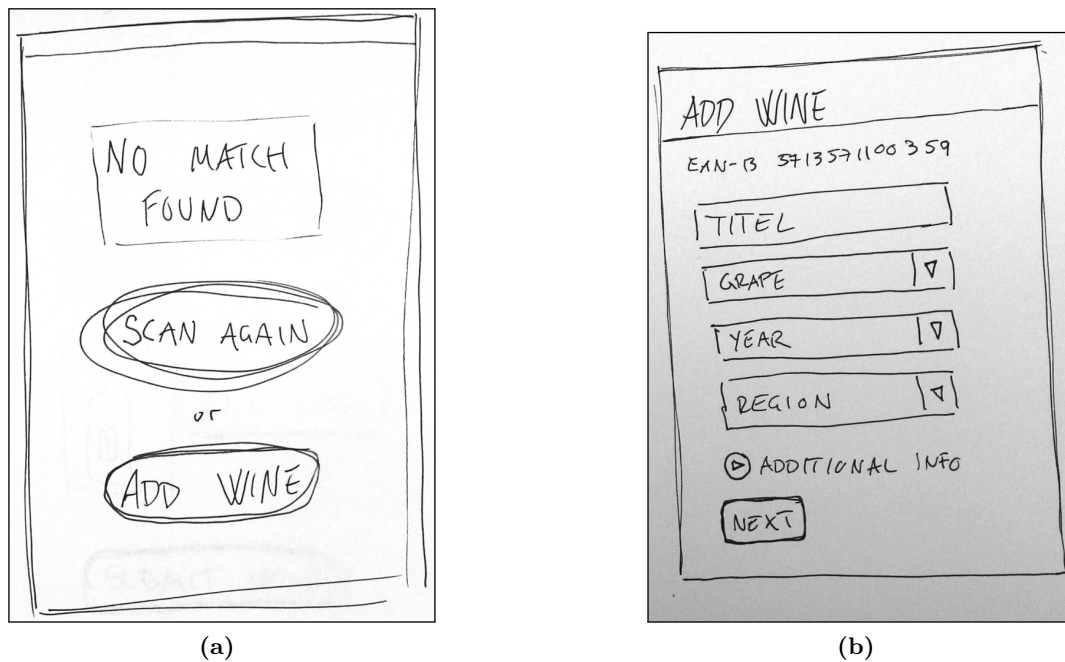


Figure 14.4: (a) When there is no match when scanning a barcode the user is presented with two options. Either scan a new barcode, or add the missing wine to the database. (b) The amount of information required to add a wine needs to be kept at absolute minimum to avoid users getting overwhelmed. The possibility to add more information should of course be possible, but it should be optional

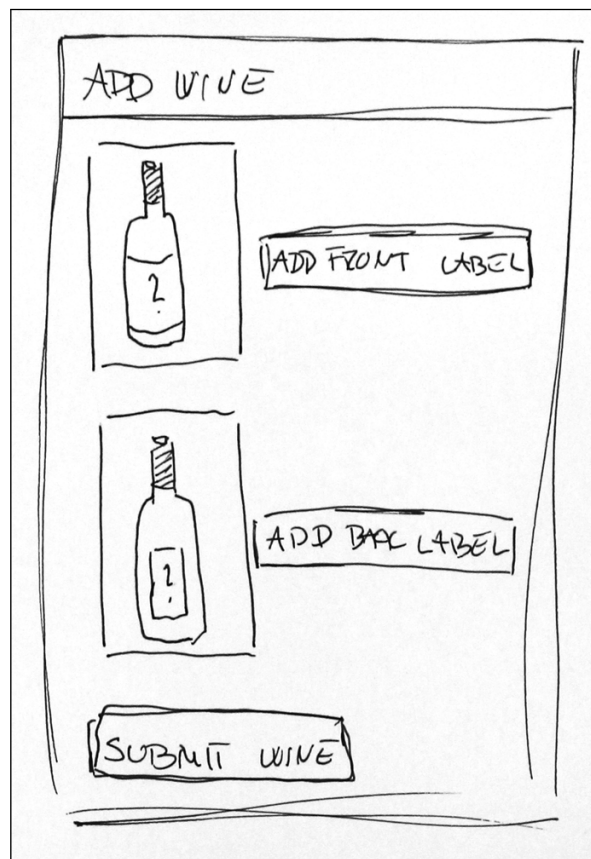


Figure 14.5: *You can use the built in camera to add both front and back label to the wine.*

15 Field Study

In an attempt to verify the two initial tests conducted on different sites on campus, a field study to a wine department in a store will be conducted. A full-fledged field test would be the optimal solution, but due to the deadline of this project and the limited amount of wines in the database, makes the possibility of a big uncontrolled test impossible, without the risk of getting to many false answers due to the current limitations. The small field study's purpose is to investigate if the answers we have from the in house tests corresponds to those we will get from the potential users. One aspect is clear and that is the age of the target group is going to rise significantly. The users on campus were naturally around the mid to late twenties and a trip to a wine department will alter that average and maybe yield some different answers.

15.0.1 Location

The field study will be conducted at Bilka Skalborg's wine department, the same location where the photos of front and back labels were shot for the database.



Figure 15.1: *The wine department at Bilka Skalborg Hobrovej 450, 9200 Aalborg Sv.*

15.0.2 Setup

The possible test participants will be found by asking customers, looking for wine if they will participate in a show and tell with a few follow up questions.

To make sure that the test subjects does not feel uneasy there will be no photos or video shot during the interview. During the interview and use of the application the participants will be observed and any interesting observations that might occur during the test will be written down.

15.0.3 Interview

First the participant will get a short show and tell about the application. When the participant have seen the application demonstrated and tried it themselves they will be asked a few questions which can be seen below. If additional questions seems relevant they will of course be asked and the answers noted just like a semi-structured interview.

- Do you find such an application interesting?
- Would you use such an application if possible?
- Would you use a similar application if it were for other products than wine?
- Would you actually rate wines or just use as a wine look-up app?
- Do you already use similar types of applications?

The possible test subjects will be approached and asked if they want to participate in a short interview and a quick try of the application. If they accept they will be taken through the following questions and tasks below. The test subject will not be let loose in the wine department because the database does not contain a sustainable amount of wines yet and there will be too many wine searches with no matches. Therefor the test participants will only be able to choose from three pre-selected wines which have already been rated.

15.0.4 Answers

The test was conducted a Monday between 17.00 and 18.00. A total of six people: two men, two women and a pair, a man and a woman, were interviewed. The age was around 35 and up to late 50 years old.

Everyone except one found the application interesting or very interesting.

No one used similar applications on their phones.

Everyone except one would use the application if it was finished and they had a smartphone.

Asked if they wanted to use the application as a look-up only or if they also would rate wines, everybody except one, said that they would primarily rate wines mostly for their own sake in combination with the favorites list etc.

When asked about what other topics a simpler system could be used on electronics in general were a common answer. One wanted to be able to check cleaning detergents and soaps for possible risk of allergy.

15.0.5 Conclusion

Even though the age difference was significant as expected compared to the campus tests, the answers corresponds to each other. It is clear that the scope for this application namely wine hit spot on. People said that it was the perfect topic for this kind of application because of the multitude, amount and complexity wines have to offer.

16 Discussion

16.1 Barcode Reader Error

During the development of the application the used barcode reader from ZXing has misinterpreted barcodes sporadically. A barcode there has been especially prone for misinterpretation is the UPC-A barcode 085200000258. The result returned by the reader when misread, instead of the correct code was instead the EAN-8 barcode 85200258. Comparing the numbers it is clear that it has skipped four zeros. The leftmost zero outside the actual barcode, a zero just before the center guard in the left half and two zeroes immediately after the center guard in the right half.

With the knowledge gathered, about encoding and decoding barcodes, it seems strange that such an error can occur. Dropping zeroes on both sides of the center guard without noticing any space gap in the input image might indicate that the algorithm are not as strict as it should be regarding accuracy. It could be a deliberate choice to maybe speed up the read times, but as mentioned above we can only guess. In further iterations, this issue should be looked into more deeply and a reader should be created from scratch to make an implementation with a lower error rate. It must be stated that the error rate has been very low in general.

16.2 Global Wine Rating

The current implementation of the global wine rating is simply the arithmetic mean of the total number of ratings for a given wine. You can argue that the mean of a small number of ratings are misleading. A solution to this could be to add, so called, dummy ratings. A dummy rating would be a rating with a value of 2,5 which is the mean of the lowest and highest score. The number of dummy ratings added could e.g. be 5. The idea is that having 5 dummy ratings with an average score will pull low and high votes against the middle for some time until the number of total ratings begin to make the dummy ratings insignificant and you will have enough ratings to make a more significant rating mean.

16.3 Prospective

The concept of scanning a barcode or label of a product and retrieve relevant information, let it be electronics, cosmetics, cleaning agents or pet food, reach far beyond the scope of the project. The feedback gotten from the test participants has been so positive that it is planned to continue developing on Vinguide and possibly expand the concept into other product categories. To make it easier to include different product categories, a template application will be made from Vinguide that quickly can be fitted to a category.

As a startup it has been discussed how to best proceed. To survive as a startup company it is necessary to generate cash flow. One way to generate cash flow with Vinguide is to somehow incorporate advertising. Advertising can be done in many different ways, where the most common is banner advertising with random ads where you are paid per click. This approach will however not generate that much cash and is seen as a last resort. Another more subtle way and possibly more giving is to incorporate advertising as a feature. A deal could be made with grocery stores to show wines of the week or month etc. The last possibility would be to give it away for free. Make an exclusive deal with e.g. Bilka and thereby get the company name exposed.

17 Conclusion

The initial vision for the project was a smartphone application capable of assisting users when selecting products. This should be accomplished by using image based search with the phone's camera, instead of the more cumbersome normal text based search. The information available in the app should rely on user generated content and user based ratings.

To delimit the project, the app was developed for Android and using barcodes as the image based search mechanic. The product category was chosen to be wine. Wine is a perfect candidate as it is a product used by a big percentage of the danish poplation dispite its complex nature.

The success criteria for the project were a product that feels finished and polished, and users should feel that they, easy and fast, are able to make a more qualified choice. The product should be published to the Android Market for public availability, and at least one unrelated user should download it.

As concluding on the product as a whole, it has been a great success. All the criteria for a successful project have more or less been accomplished. The wine app has been published as a polished and finished app to the Android Market. With over 500 downloads divided between the Scandinavian countries, the release have been a success. Though, without any use of advertisement it might seem that it is not a major achievement to have unrelated downloads from users. However it must indicate that the topic of the app is sought and needed.

The different tests made through the project have also been a great success. The majority of the test participants have showed highly interest in the app, and all reported a pleasant and easy experience while using the app. The design and interface seem to appeal to most of the test participants and did overall not seem confusing. The test participants also reported that they generally felt informed by the information given by the app, and that they felt more qualified when choosing wine.

As an application to be released to the market, it still needs some slight improvements. The database used to store information and products, still only holds about 150 entries, which is not nearly enough to make the product useable. Furthermore there should be added slightly more features to accommodate the general use of the app. But in its current state with its current capabilities, the most important features are present.

The use of user generated content has been applied to some extent. The users can rate wines, but the feature where users have the opportunity to add or edit missing information, and add wine entries to the database is still missing.

Closing, the project as a whole is a success with many positive comments coming from external viewers and observers.

Bibliography

- [1] Wikipedia. Smartphone, February 2011.
- [2] Wikipedia.org. Application software, May 2011.
- [3] Till Quack, Herbert Bay, and Luc Van Gool. Object recognition for the internet of things. In *Proceedings of the 1st international conference on The internet of things*, IOT'08, pages 230–246, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78730-5, 978-3-540-78730-3.
- [4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.
- [5] Wichian Premchaiswadi. An image search for tourist information using a mobile phone. *WSEAS Trans. Info. Sci. and App.*, 7:532–541, April 2010.
- [6] Google. Google goggles, May 2011.
- [7] Ebay. Red laser, May 2011.
- [8] ShopSavvy. Shopsavvy, May 2011.
- [9] Konrad Tollmar, Ted Möller, and Björn Nilsved. A picture is worth a thousand keywords: exploring mobile image-based web search. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, pages 421–428, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-862-6.
- [10] Elsebeth Lohfert. Meininger's wine business international - the magazine for the global wine trade, October 2007.
- [11] FDB Analyse. Nordsjællænderne topper det danske vinkort, February 2011.
- [12] Børsen. Salget af smartphones eksploderer i danmark, November 2010.
- [13] comScore. Age demographic profile for mobile, smartphone and ipad owners, April 2011.
- [14] Google. www.google.com, May 2011.
- [15] Google. Android market, May 2011.
- [16] Apple, May 2011.
- [17] Theis og Heini. Vivino, April 2011.
- [18] dkapps.dk. Din vintjener, April 2011.
- [19] Jakob Nielsen. How to conduct a heuristic evaluation, April 2005.
- [20] Andre Domine. *WINE*. hf ULLMANN, 2010. ISBN 9783833146145.

- [21] Wikipedia.org. Wine, May 2011.
- [22] Wikipedia. Open handset alliance, May 2011.
- [23] Wikipedia.org. Android (operating system), May 2011.
- [24] Android.com, May 2011.
- [25] Wikipedia.org. ios (apple), May 2011.
- [26] Apple. Developer, May 2011.
- [27] Apple, May 2011.
- [28] Stephen V. Rice, George L. Nagy, and Thomas A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Norwell, MA, USA, 1999. ISBN 079238492X.
- [29] Wikipedia.org. Optical character recognition, May 2011.
- [30] Adrien Auclair, Laurent D. Cohen, and Nicole Vincent. How to use sift vectors to analyze an image with database templates.
- [31] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150 –1157 vol.2, 1999.
- [32] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf.
- [33] Global trade item number, May 2011.
- [34] Apple. ios human interface guidelines, Marts 2011.
- [35] Google. User interface guidelines, May 2011.
- [36] Google. Google i/o 2010 - creating positive user experiences, June 2010.
- [37] Google. Android ui design patterns, May 2010.
- [38] pocketnow.com. Htc desire gets reviewed early, Marts 2010.
- [39] PureMobile.com. The android froyo to be available soon on the htc desire, legend and wildfire, July 2010.
- [40] Facebook. Facebook app for android, April 2011.
- [41] IMDB The Internet Movie Database. Imdb movies & tv app for android, May 2011.
- [42] Google. Gmail for android, February 2011.
- [43] Social Bakers. Denmark facebook statistics, May 2011.
- [44] Wine.com. Wine.com, May 2011.
- [45] <http://www.vinlex.dk/>. Vinlex, May 2011.

- [46] D. Chai and F. Hock. Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras. *Information, Communications and Signal Processing, 2005 Fifth International Conference on*, pages 1595–1599, 2005.
- [47] GS1 Australia. Gs1 australia user manual - bar code technical details. January 2010.
- [48] TALtech. How a barcode reader works, May 2011.
- [49] Wikipedia. Relational database, May 2011.
- [50] Paul Litwin. Fundamentals of relational database design. 1994.
- [51] Gregor Roth (InfoQ). Restful http in practice, May 2011.
- [52] Alex Rodriguez (IBM). Restful web services: The basics, May 2011.
- [53] Richard H. Carver & Kuo-Chung Tai. *Modern Multithreading*. Wiley, 2006. ISBN 9780471725046.
- [54] Google. Android developer guide, May 2011.
- [55] Google. Google developer guide - activities, May 2011.
- [56] ZXing ("Zebra Crossing"). Zxing ("zebra crossing"), 2011.
- [57] Facebook. Facebook sdk for android, May 2011.
- [58] Tiresias.org. Think-aloud method, November 2009.

Appendices

A List of Figures and Tables

List of Figures

1.1	Some of the features included in a modern Smartphone. Each feature has a standalone equivalent slowly being killed.	4
2.1	Client software for the city guide application: the user snaps a picture, waits a few seconds, and is redirected to the corresponding Wikipedia page [3]	8
2.2	Image-based mobile tour guide [5]	9
2.3	Screenshots from different mobile applications using the scanner principle. 2.3a: Google Goggles. 2.3b: ShopSavvy. 2.3c: Red Laser	10
2.4	A View of the wine department in Bilka Skalborg with its abundant selection and a forest of yellow price tags.	13
2.5	The main feature of the smartphone application is the ability to snap a picture of any given wine and quickly display relevant information	15
2.6	The illustration show the overall system of the wine application with a smartphone application used on the go and a web portal for better overview when time is to it	16
2.7	Gender distribution on mobile phones compared with smartphones. (Based on subscribers, Europe numbers, 2011) [13]	19
2.8	Age distribution on mobile phones compared with smartphones. (Based on subscribers, Europe numbers, 2011) [13]	19
2.9	While every third above the age of 55 drinks wine on an average day, the same only apply to every tenth below the age of 34. [11]	20
2.10	Screen shots from the Drync Wine app. [15]	21
2.11	Screen shots from the Vivino app. [15]	22
2.12	Screen shots from the Din Vintjener app. [18]	22
2.13	Screen shots from the Hello Vino app. [15]	23
2.14	Age distribution of respondents	29
2.15	How many of the respondents drinks wine	29
2.16	Shows how many of the respondents make their own choice of wine	30
2.17	Why do some respondents not make their own choice of wine	30
2.18	This diagram shows whether the respondents that do not make their own choice of wine, if they would find a wine application useful or not	31
2.19	Do the respondents that drink wine, tell others about their experiences	31
2.20	This diagram shows the respondents own judgment of knowledge on wine	32
2.21	The diagram shows how often the respondents confront the staff to get help to buy the right wine	32
2.22	The respondents were asked to pick the three most important criteria when choosing a wine	33

2.23	This diagram show how confident the respondent feels after having bought a wine. Is their pick on wine normally just a wild guess, or do they know what they choose	33
2.24	The diagram shows whether the respondents would find a wine app useful or not. This only shows the result from those who actually make their own choice of wine, which is different from figure 2.18.	34
2.25	The chart shows whether the respondents use the Internet to search for information about products in general beforehand a buy	34
2.26	The ripe grape is crushed and the juice is collected. When making red wine the seeds and skin is mixed in with the juice during the fermentation process.[21]	36
2.27	(a) An example of a vineyard with its long straight lines of vines.[21]. (b) A vine with a series of ripe blue grapes.[21].	37
2.28	Some wineries are casking their wine in a wine cellar before bottling. The casks usually made of oak affects to affect the taste that typically oak can give.[21]	37
2.29	Smartphone sales worldwide from fourth quarter 2010 based on operating system.[1]	38
2.30	Android logo.[24]	39
2.31	Apple's iPhone 4[27]	40
2.32	An example of a front label (a) and a back label (b).	41
2.33	Server-client model. A server with a database serving data for its clients.	44
3.1	A user despairing looking for the right wine to buy, not having a clue how any of them tastes.	50
3.2	A user that have found a wine of interest, snaps a picture with his phone, and quickly find out the rating of the wine.	51
3.3	The diagram outlines all the possible functionalities thought of for the wine app.	54
3.4	As an example of good feedback, all interactive interface design elements should have at least 4 states. [34]	58
3.5	The diagram outlines what specific features that are going to be implemented in the first design.	62
3.6	A HTC Desire smartphone[38].	65
3.7	A closeup of the HTC Desire. At bottom is placed five buttons used for navigation in Android. This five buttons is standard on Android mobile phones[39].	66
3.8	Example of a Dashboard from the Twitter app on Google Android. Top screen shows the main features of the program, while the bottom gives the user updates on content and application updates. [37]	67
3.9	An example of the ActionBar from the Twitter app for Google Android. The ActionBar is the bar covered in blue at the top. To the left the application name, or the application status is shown alongside the application icon. To the right the actual actions are located. The actions here are refresh, compose, and search. [37]	68
3.10	An example of the Quick Action design pattern used in the twitter app for Google Android. It is a popup that makes for quick and intuitive interaction with a specific element. [37]	68
3.11	Sketch of the wine app main screen using the Dashboard design pattern. The Dashboard consists of the eight most important features. The name "Winelicious" is used as a dummy name for the application name. "Catalog" refers to the database search functionality. "Scan" refers to the search functionality.	70
3.12	The ActionBar replaces the application title bar and provide shortcut to the most common features of the program which is the rate and the search functionality.	70
3.13	All sub screens from the main screen follow a common layout as shown in this figure.	71

3.14	Both left and right show how the Quick Action design pattern used in the app. The quick actions are triggered either when tabbing the screen identification icon or a content item icon. The quick actions enables the user to quickly change screen location or get access to most common item interaction.	72
3.15	This flowchart display screen overview of the app. The main screen, the dashboard, is placed in the middle and all its sub screens emerge from it.	74
3.16	This flowchart display the actual application flow and what actions the user can do, and how the app responds to the users actions.	75
3.17	Sketch of the search screen, called scan in the app, which consists of two tabs. First tab enables the user to search by using an image, while the second enables the user to do a normal text search.	76
3.18	Left is shown a sketch of the rating screen. The user can browse through a list in the recent tab and the favorites tab to do rating. As described Quick actions are available when tabbing either the screen identification icon as shown in right, or tabbing an item icon in the list.	77
3.19	Left a sketch of the recent screen is shown. It consists of a list the user can browse through. To the right a sketch of the favorite screen is seen. It consists of a list containing wines that are added to the favorites. At the bottom of the favorite screen it can be seen how the menu pop up is displayed when the dedicated menu button on the device is pressed.	78
3.20	Left is a sketch of the popular screen which consists of three tabs for listing the top rated wines, the most popular wines and recently added wines. Right is shown how quick actions are available when tabbing appropriate icons in the different screen such as icons from the list.	78
3.21	A sketch of the profile screen. The content of the profile screen is not completely defined, but will be like an account settings where the user can change information about her.	79
3.22	A sketch of the encyclopedia screen. Here the search tab is open and the user can type in a search.	80
3.23	The encyclopedia with different tabs opened to show how listing of items is done. The small triangle on the item indicates to the user that it can be expandable by tabbing it. On the left it is shown how the alphabetical view should look like. . . .	81
3.24	A sketch of the wine information screen. This screen contains all the information of a specific wine. The icon that looks like a nut is actually an icon of a hunk of meat to indicate that the wine goes good with that.	82
4.1	The digits marked with blue is the country code, green the manufacturer code, pink the product code and yellow the check digit.	88
4.2	The EAN-13 Barcode's 5 parts. [47]	88
4.3	It takes seven modules to create a symbol character. A symbol character has two white and two black bar differing in thickness where each bar has a maximum thickness of four modules. [47]	89
4.4	The EAN-13 Barcode's total number of modules. [47]	89
4.5	Shows the encoding of digit 0 - 9 in Number set A, B and C. 0 Represents white modules and 1 represents black modules.[47]	90
4.6	The sequence of number sets used to create the left half of the barcode is determined by the leftmost number in the barcode. If the country code is 57 the sequence is: ABBAAB.[47]	90

4.7	The figure shows the number of modules and the encoding of the barcode guards. [47]	91
4.8	A possible photo diode response of reflected laser light.[46]	91
4.9	The thresholded result from figure 4.8.[46]	92
4.10	The terminology describing a relation in a relational database. A relation is typically referred to as table and attributes and tuples are commonly referred to as columns and rows like in an ordinary table. [49]	93
4.11	A database model diagram showing the main table of the wine database. An almost replicate of table 4.2 and it needs further optimizations. The final database model diagram can be seen in figure 4.12	96
4.12	Database model diagram showing the tables in the wine database and their relationships. The main table (wines) contains foreign keys to other tables which define the relationship with those.	97
4.13	Shows an example of an error / warning to the administrator, so bad or duplicate data are not inserted to the database.	98
4.14	A flow-chart depicting the back-end when an administrator wants to add a wine.	99
4.15	Users will login on the client with their username and password and the client initiates a request and ask the server if the user is authorized. If so, user id and a token will be send back to the client that will use it to make requests as a specific user.	103
4.16	Shows the general flow of a request sent from the client to the server.	104
4.17	The SQL that will query the database in a search are generated from features and its constraints if they was contained in the filter parameter in the query string sent as part of the request.	105
4.18	The processing of the search request.	106
4.19	The processing of the rating request.	106
4.20	Shows the Android system architecture and its layers. [54]	109
4.21	An overview of all the activities identified for the application.	114
4.22	The view hierarchy in Android used to define interfaces. The ViewGroup act as parent, defining the positions of its children. A View is child of a ViewGroup and is a actionable element called a widget, that the user can interact with.	115
4.23	The diagram shows how to create the Dashboard interface used for the main screen of the wine app. It is a hierarchy of ViewGroups, the LinearLayots, and View Widgets, the ImageButtons	116
4.24	The final result of the defined interface from figure 4.23	117
4.25	Shows a flowchart depicting the search in the application.	119
4.26	An Android ListView with some items.	120
5.1	(a) The dashboard with the six icons and the actionbar at the top of the screen. The top rightmost icon named test located int the actionbar, is for test purpose and not a part of the design. (b) The search view with the big ' Push to start barcode reader' button.	122
5.2	(a) The placeholder screen for Rate. (b) The placeholder screen for Recent. (c) The placeholder screen for Favorites.(d) The placeholder screen for Popular.	123
5.3	(a) The Login screen with the ability to use and existing Facebook account for login or create a new user and login. (b) The placeholder screen for Profile (c) List (d) Wine Info	124

5.4	You rate a wine by dragging your finger across the stars to fill them. When you have filled the number of stars you think the wine deserves. To submit your rating you press ok or press cancel if you do not want to rate the wine.	125
5.5	An overview of the implemented features following the first implementation iteration. Features marked red with a minus sign (-), indicates planned features. Features marked black with a plus sign (+), indicates implemented features. Gray boxes indicates features and screens not implemented at all.	126
6.1	A curve showing the percentage of usability problems per evaluator found in an interface using heuristic evaluation[19].	127
6.2	A drawing of the test setup. Camera 1 is filming the whole scene and Camera 2 is filming the screen of the smartphone.	128
6.3	(a) A camera with a close up of the interaction. (b) A shot of the evaluator and observer.	129
8.1	An overview of the Dashboard as it looked after the first iteration.	138
8.2	(a) The Search screen containing the "Scan Bar Code" and "Text Search" tabs, as it was implemented in the first prototype. (b) A mockup of the new dashboard layout after merging several functions. Top left icon called Scan, will lead directly to the scanner. The top right icon called Index is a merging of, text search, popular. It will feature a search based on search criteria.	139
8.3	This illustrates the concept of the new Index functionality. It makes it possible to search via a keyword, and add criteria to the search. By sorting the search results, the functionality of the old popular functionality can be incorporated into the search.	139
8.4	The new rating system with a separate user rating row of stars and an indication of how many people have rated a given wine.	140
8.5	(a) The wine info screen with the rate button (old design). (b) When the rate button is pushed the rate dialog pops up (old design).	141
8.6	In the redesign the screen title bar is removed to save space. The actual navigation of the app is not that complex, and therefore the screen title is not a severe loss. .	143
8.7	The add note functionality will be incorporated in the wine info screen. It is a button that will be placed along with the 'add to favorite' button, and is a button with some sort of note icon.	144
9.1	A database model diagram showing the notes table and its relations with the "users" and "wines" table.	145
9.2	Shows the processing of the add/edit note.	146
9.3	(left) The search screen when opened from the dashboard (right) When editing for example the rating criteria the user, uses two sliders to set the desired range, in this case rating. The adjustment screen is a dialog popup which is activated when the users press the edit (Redigér in Danish) button from the search screen.	147
9.4	To the left is the Android dialog. The normal content in the dialog, is replaced with a custom View to the right. The custom View contains the adjustable sliders, a checkbox and a text field.	148
9.5	Conceptual class diagram of the dialog builder class. The class returns a AlertDialog which can be showed as a popup dialog.	149
9.6	Conceptual class diagram of the superclass used for the custom view.	149

10.1	a) The redesigned dashboard layout. The Action bar now only consist of a single shortcut for the scanner. The feedback button in the lower right corner is for user feedback only and not a part of the design. (b) When scan is toggled you go straight to the scanner application ready to scan a barcode. (c) The wine information screen with basic information, user and average rating and the wine added to favorites. (d) The wine information screen with the additional info and a note added.	152
10.2	(a) Rating a wine swiping the finger back and forth to in- or decrease the number of stars. (b) The list view when more than one match occurs. (c) Text search and the possibility to narrow the search for example, rating, price or region. (d) The dialog popup when defining special search criteria.	153
10.3	(a) How quick action look. The quick action pop ups underneath or above the wine when pressing the picture of the wine (b) It is now possible to log out of your profile.(c) A list of the recent scans or text searches. (d) A list of wines on the favorite list.	154
10.4	An overview of the implemented features following the first implementation iteration. Features marked red with a minus sign (-), indicates planned features. Features marked black with a plus sign (+), indicates implemented features. Gray boxes indicates features and screens not implemented at all.	155
11.1	The five different wines used for the wine tasting in both parts of the test. They are arranged from lowest price to highest price. From the left in Danish kroner: 16,- 30,- 31,- 48,- 68,-	160
11.2	The lineup of wine as they were used during the test	161
11.3	Picture taken during part 1 of the test	162
11.4	Pictures taken during the second part of the test	168
13.1	Shows a graph illustrating the total active installs from 19. April 2011 till 26. May 2011	177
13.2	Shows the distribution among devices with an active installations of Vinguide . . .	178
13.3	Shows the distribution among countries with an active installations of Vinguide . .	178
14.1	To make room for an additional row of stars representing the users individual rating the mini icon for added to favorites gets moved out right and the row of stars obviously replaces the rated indicator.	180
14.2	One image	181
14.3	(a) This proposal positions the “create new user” button at the top instead at the bottom. To clarify that you need to create a user to login. (b) This proposal eliminates any text fields where text can be wrongfully submitted and time and effort wasted until either create new user, login or Facebook login are selected. . .	182
14.4	(a) When there is no match when scanning a barcode the user is presented with two options. Either scan a new barcode, or add the missing wine to the database. (b) The amount of information required to add a wine needs to be kept at absolute minimum to avoid users getting overwhelmed. The possibility to add more information should of course be possible, but it should be optional	183
14.5	You can use the built in camera to add both front and back label to the wine. . . .	184
15.1	The wine department at Bilka Skalborg Hobrovej 450, 9200 Aalborg Sv.	185

D.1	Broken Dimensions.	225
D.2	The raycasting directions for checking player surroundings.	227
D.3	Broken Dimensions screenshot	228
D.4	Broken Dimensions screenshot	228
D.5	Broken Dimensions screenshot	228
D.6	Boxcover for Blendimals	229
D.7	Screenshot from Blendimals	230
D.8	Screenshot from Blendimals	230

List of Tables

2.1	Summary of recognition rates for city guide [3]	8
3.1	All features that was found and if they can be enumerated in lists.	86
4.1	A table representing a relation with three columns (attributes) and four rows (tuples). Each attribute has a permitted set of values and each row describe a relation between the data across it.	93
4.2	All features and information that should be contained in a wine entry with examples of each.	95
4.3	The different versions of Android and their corresponding API level as of May 2011 [54]	113

B Initial Questionnaire

B.1 Questions

The next 8 pages include the questionnaire from the initial pre study. The questionnaire is in Danish. It is also included on the DVD.

Spørgeskemaundersøgelse

Dette spørgeskema vil spørge ind til dit forbrug og købsvaner af vin. Derudover vil der blive spurgt ind til din brug af smartphones og programmer på telefonen.

Spørgeskemaet består af max 21 spørgsmål.

Dine svar forbliver anonyme og vil blive brugt i sammenhæng med et videnskabeligt projekt på Aalborg Universitet

*** Required**

Køn *

- ☐ Mand
☐ Kvinde

Alder *

Indtægt *

- ☐ 0 - 99.000 kr.
☐ 100.000 - 199.000 kr.
☐ 200.000 - 299.000 kr.

-
- ☐ 300.000 - 399.000 kr.
 - ☐ 400.000 - 499.000 kr.
 - ☐ 500.000+ kr.

Spørgsmål om vin

Drikker du vin? *

- ☐ Ofte
- ☐ Sjældent
- ☐ Aldrig

Spørgsmål om vin

Når du har smagt en vin, fortæller du efterfølgende om den til andre? *

Det kan være hvis du har smagt en god vin og synes andre også skal prøve den

- ☐ Ja, fortæller gerne om mine oplevelser med vine til andre
- ☐ Nej, snakker sjældent om de vine jeg har smagt til andre

Vælger du selv den vin du køber? *

- ☐ Ja, jeg vælger som regel selv når jeg køber vin
- ☐ Nej, andre vælger eller køber altid for mig når jeg skal have vin

Spørgsmål om vin

Hvorfor vælger du ikke selv dine vine? *

- ☐ Jeg er ligeglad eller gider ikke bekymre mig om det
- ☐ Jeg ved ikke nok om vin
- ☐ Ved ikke / er bare aldrig med til at vælge vin
- ☐ Other:

Når du står overfor at skulle købe vin, kan du som regel kun få

information ved at kigge bag på flasken eller spørge personalet. Forestil dig at have adgang til en applikation/program på din mobiltelefon, der på en nem måde kunne give dig yderligere information og hjælpe dig til et bedre valg af vin. Yderligere informationen kunne være anmeldelser, brugervurdering, hvilket mad vinen passer sammen med, etc. I hvor høj grad tror du at du ville finde sådan et program nyttigt? *

- ☐ Har ikke mobil / benytter ikke applikationer/programmer på min mobil
- ☐ Finder i ingen grad sådan et program interessant / ville ikke anvende det
- ☐ Finder i lille grad sådan et program interessant
- ☐ Finder i mellem grad sådan et program interessant
- ☐ Finder i høj grad sådan et program interessant
- ☐ Finder i megen høj grad sådan et program interessant
- ☐ Ved ikke

Er der noget du gerne vil vide om en vin som normalt ikke oplyses på flasken?

Spørgsmål om køb af vin

Hvor foretrækker du typisk at købe din vin? *

- ☐ I en discount butik såsom Fakta, Rema 1000 eller lignende.
- ☐ I et supermarked som Kvickly, Super Brugsen, Føtex eller Bilka.
- ☐ I specialbutikker som Skjold Burne og lignende.
- ☐ Other:

Hvilket prisleje ligger de vine du køber normalt i? *

1 2 3 4 5

Det absolut billigste ☐ ☐ ☐ ☐ ☐ De absolut dyreste**Synes du selv du er god vinkender? ***

1 2 3 4 5

Jeg ved næsten ikke noget om vine ☐ ☐ ☐ ☐ ☐ Jeg er meget erfaren når det kommer til vin**Hvor ofte spørger du personalet om hjælp når du skal købe vin? ***

1 2 3 4 5

Spørger aldrig om hjælp til køb af vin ☐ ☐ ☐ ☐ ☐ Spørger næsten altid om hjælp til køb af vin**Hvad er de 3 vigtigste faktorer når du skal vælge vin? ***

Hvad får dig normalt til at vælge en vin? Vælg op til 3 svar

- ☐ Etikettens visuelle udseende
- ☐ Etikettens beskrivende tekst
- ☐ Alkohol procenten
- ☐ Pris
- ☐ Anbefaling fra venner og familie
- ☐ Tidligere egne erfaringer
- ☐ Anmeldelser fra f.eks. avis eller internettet
- ☐ Druen
- ☐ Årgang
- ☐ Hvilken mad vinen passer til
- ☐ Oprindelsesland
- ☐ Other:

Når du har købt en vin, hvor sikker føler du dig normalt på du har købt noget godt? *

Er dit valg af vin et sats, eller føler du at du har godt styr på dit valg?

- ☐ Jeg er ofte i tvivl om den bliver en succes eller ej, men håber det bedste
- ☐ Jeg er som regel rimelig sikker på mit valg af vin
- ☐ Jeg er næsten altid sikker på at jeg har fået valgt den rigtige vin til formålet
- ☐ Other:

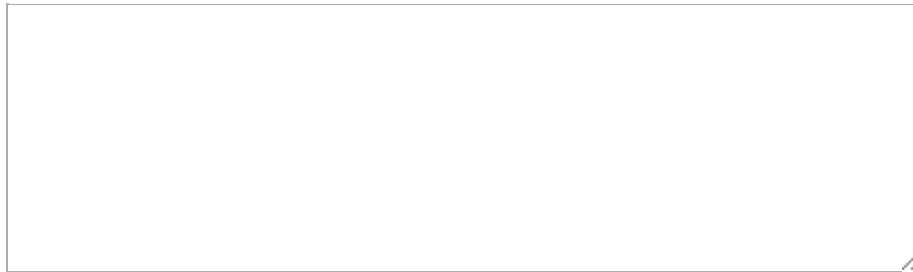
Når du køber vin, går du så efter de vine som du kender og ved smager godt, eller prøver du noget nyt? *

- ☐ Jeg går næsten altid efter de vine jeg kender og er sikker på smager godt
- ☐ Jeg er ikke bange for at købe en vin jeg ikke har prøvet før, men køber det jeg kender hvis der ikke er andet
- ☐ Jeg går sjældent efter en bestemt vin som jeg kender, men køber som regel det jeg lige føler for
- ☐ Other:

Når du står overfor at skulle købe vin, kan du som regel kun få information ved at kigge bag på flasken eller spørge personalet. Forestil dig at have adgang til en applikation/program på din mobiltelefon, der på en nem måde kunne give dig yderligere information og hjælpe dig til et bedre valg af vin. Yderligere informationen kunne være anmeldelser, brugervurdering, hvilket mad vinen passer sammen med, etc. I hvor høj grad tror du at du ville finde sådan et program nyttigt? *

- ☐ Har ikke mobil / benytter ikke applikationer/programmer på min mobil
- ☐ Finder i ingen grad sådan et program interessant / ville ikke anvende det
- ☐ Finder i lille grad sådan et program interessant
- ☐ Finder i mellem grad sådan et program interessant
- ☐ Finder i høj grad sådan et program interessant
- ☐ Finder i megen høj grad sådan et program interessant
- ☐ Ved ikke

Er der noget specielt du gerne vil vide om en vin som normalt ikke oplyses på flasken?



Telefonbrug

Er din mobiltelefon en 'smartphone'? *

En smartphone kan gå på internettet, og der kan hentes programmer til den osv.

- ☐ Ja
☐ Nej

Smartphones

Har du hentet programmer til din smartphone? *

F.eks. programmer om vejret eller rejseplanen

- ☐ Ja
☐ Nej

Applikationer til smartphones

Hvad betaler du normalt for et nyttigt program? *

- ☐ Har aldrig brugt penge på et program til min telefon
☐ 0 - 5 kr.
☐ 6 - 10 kr.
☐ 11 - 15 kr.
☐ 16 - 20 kr.
☐ 21 - 30 kr.
☐ 31 - 50 kr.

☐ 50+ kr.

☐ Other:

Generer reklamer i programmer til din smartphone dig? *

☐ Ved ikke / har ikke prøvet programmer med reklamer på min telefon

☐ Jeg lægger som regel ikke mærke til dem og de generer mig ikke betydeligt

☐ Jeg lægger mærke til dem men kan leve med dem

☐ De er lettere generende og vil helst være fri

☐ Reklamer i programmer er så irriterende at jeg kan finde på at droppe programmet

☐ Other:

Diverse

De følgende spørgsmål gælder dit generelle forbrug, og er altså IKKE begrænset til dit brug af vin eller smartphones

Bruger du internettet til at finde information og anmeldelser af produkter inden du køber? *

Dette gælder ikke kun vin, men er generelt for alle de produkter du køber

☐ Ja stort set altid når det er muligt

☐ Det gør jeg ofte

☐ Det gør jeg ikke så ofte

☐ Stort set aldrig

☐ Other:

Bruger du sociale medier på nettet i hverdagen? *

Igen, dette er dit generelle forbrug, og gælder ikke kun vin eller smartphones

☐ Nej

☐ Facebook

☐ Twitter

☐ Forums

☐ Blogs

☐ Other:

Tak for din deltagelse - Husk at trykke på knappen send for at afslutte spørgeskemaet

Powered by [Google Docs](#)

[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)

B.2 Answers

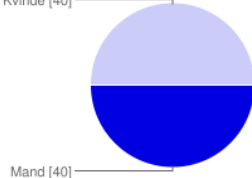
The next 6 pages include the answers from the initial pre study. The answers is in Danish. It is also included on the DVD.

80 [responses](#)

Summary [See complete responses](#)

Køn

Kvinde [40]



Mand

40

50%

Kvinde

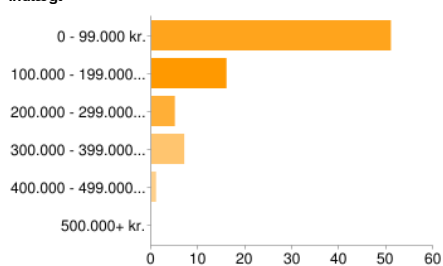
40

50%

Alder

25 28 23 25 25 24 26 28 24 25 25 50 22 26 22 28 55 28 21 26 50 34 24 26 30 24 24 29 25 30 25 26 24 25 20 20 25 :

Indtægt



0 - 99.000 kr.

51

64%

100.000 - 199.000 kr.

16

20%

200.000 - 299.000 kr.

5

6%

300.000 - 399.000 kr.

7

9%

400.000 - 499.000 kr.

1

1%

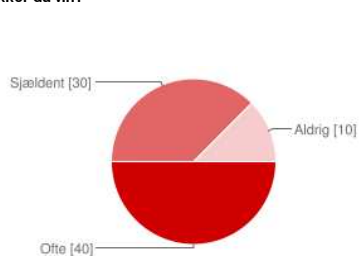
500.000+ kr.

0

0%

Spørgsmål om vin

Drikker du vin?



Ofte

40

50%

Sjældent

30

38%

Aldrig

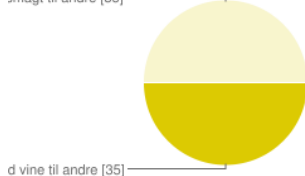
10

13%

Spørgsmål om vin

Når du har smagt en vin, fortæller du efterfølgende om den til andre?

smagt til andre [35]



Ja, fortæller gerne om mine oplevelser med vine til andre

35

44%

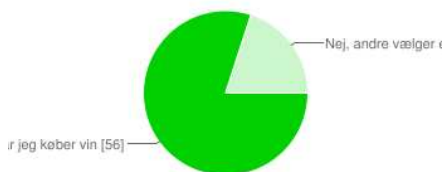
Nej, snakker sjældent om de vine jeg har smagt til andre

35

44%

d vine til andre [35]

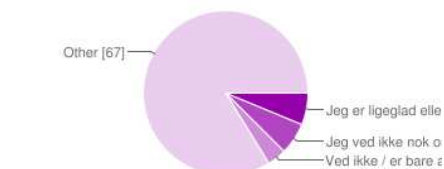
Vælger du selv den vin du køber?



Ja, jeg vælger som regel selv når jeg køber vin	56	70%
Nej, andre vælger eller køber altid for mig når jeg skal have vin	14	18%

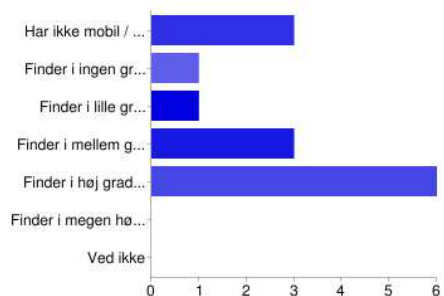
Spørgsmål om vin

Hvorfor vælger du ikke selv dine vine?



Jeg er ligeglad eller bryder mig ikke om vin	5	6%
Jeg ved ikke nok om vin	5	6%
Ved ikke / er borte	3	4%
Other	67	84%

Når du står overfor at skulle købe vin, kan du som regel kun få information ved at kigge bag på flasken eller spørge personalet. Forestil dig at have adgang til en applikation/program på din mobiltelefon, der på en nem måde kunne give dig yderligere information og hjælpe dig til et bedre valg af vin. Yderligere informationen kunne være anmeldelser, brugervurdering, hvilket mad vinen passer sammen med, etc. I hvor høj grad tror du at du ville finde sådan et program nyttigt?



Har ikke mobil / benytter ikke applikationer/programmer på min mobil	3	4%
Finder i ingen grad sådan et program interessant / ville ikke anvende det	1	1%
Finder i lille grad sådan et program interessant	1	1%
Finder i mellem grad sådan et program interessant	3	4%
Finder i høj grad sådan et program interessant	6	8%
Finder i megen høj grad sådan et program interessant	0	0%
Ved ikke	0	0%

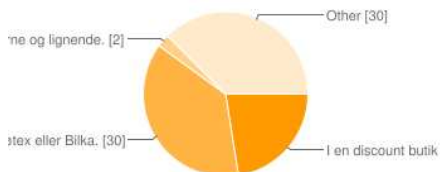
Er der noget du gerne vil vide om en vin som normalt ikke oplyses på flasken?

Om den er pengene værd
stærk i smagen den er

Hvilket mad den passer til og hvor

Spørgsmål om køb af vin

Hvor foretrækker du typisk at købe din vin?



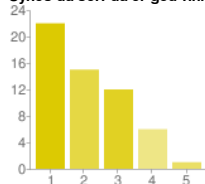
I en discount butik såsom Fakta, Rema 1000 eller lignende.	18	23%
I et supermarked som Kvickly, Super Brugsen, Føtex eller Bilka.	30	38%
I specialbutikker som Skjold Burne og lignende.	2	3%
Other	30	38%

Hvilket prisniveau ligger de vine du køber normalt i?

1 - Det absolut billigste	7	9%
2	26	33%
3	20	25%
4	3	4%
5 - De absolut dyreste	0	0%



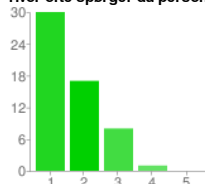
Synes du selv du er god vinkender?



Jeg ved næsten ikke noget om vineJeg er meget erfaren når det kommer til vin

1 - Jeg ved næsten ikke noget om vine	22	28%
2	15	19%
3	12	15%
4	6	8%
5 - Jeg er meget erfaren når det kommer til vin	1	1%

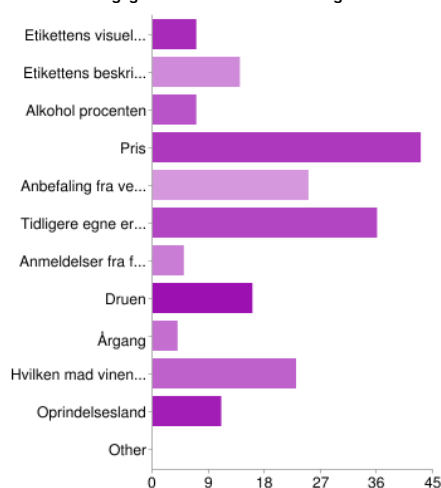
Hvor ofte spørger du personalet om hjælp når du skal købe vin?



Spørger aldrig om hjælp til køb af vinSpørger næsten altid om hjælp til køb af vin

1 - Spørger aldrig om hjælp til køb af vin	30	38%
2	17	21%
3	8	10%
4	1	1%
5 - Spørger næsten altid om hjælp til køb af vin	0	0%

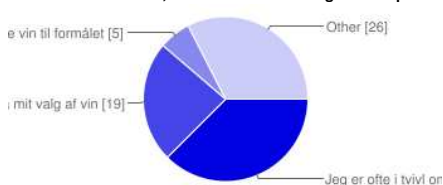
Hvad er de 3 vigtigste faktorer når du skal vælge vin?



Etikettens visuelle udseende	7	13%
Etikettens beskrivende tekst	14	25%
Alkohol procenten	7	13%
Pris	43	77%
Anbefaling fra venner og familie	25	45%
Tidligere egne erfaringer	36	64%
Anmeldelser fra f.eks. avis eller internettet	5	9%
Druen	16	29%
Årgang	4	7%
Hvilken mad vinen passer til	23	41%
Oprindelsesland	11	20%
Other	0	0%

People may select more than one checkbox, so percentages may add up to more than 100%.

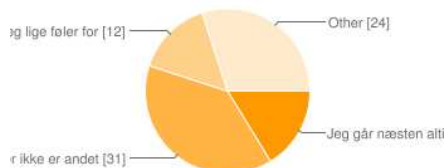
Når du har købt en vin, hvor sikker føler du dig normalt på du har købt noget godt?



Jeg er ofte i tvivl om	30	38%
Jeg er som regel rimelig sikker på mit valg af vin	19	24%
Jeg er næsten altid sikker på at jeg har fået valgt den rigtige vin til formålet	5	6%
Other	26	33%

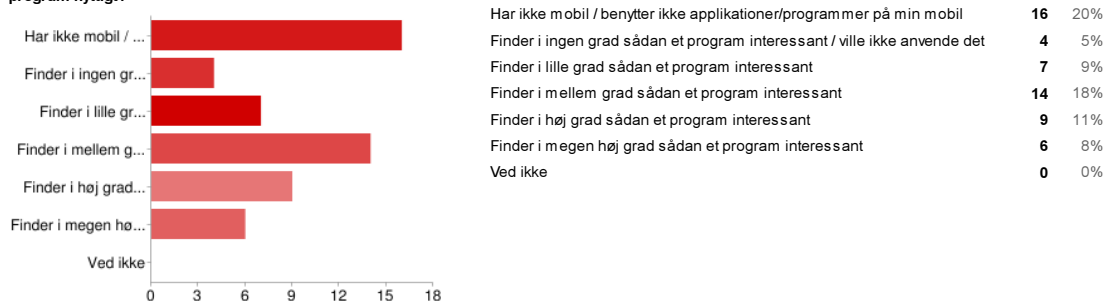
Når du køber vin, går du så efter de vine som du kender og ved smager godt, eller prøver du noget nyt?

Jeg går næsten altid efter de vine jeg kender og er sikker på smager godt



Jeg er ikke bange for at købe en vin jeg ikke har prøvet før, men køber det jeg kender hvis der ikke er andet	31
Jeg går sjældent efter en bestemt vin som jeg kender, men køber som regel det jeg lige føler for	12
Other	24

Når du står overfor at skulle købe vin, kan du som regel kun få information ved at kigge bag på flasken eller spørge personalet. Forestil dig at have adgang til en applikation/program på din mobiltelefon, der på en nem måde kunne give dig yderligere information og hjælpe dig til et bedre valg af vin. Yderligere informationen kunne være anmeldelser, brugervurdering, hvilket mad vinen passer sammen med, etc. I hvor høj grad tror du at du ville finde sådan et program nyttigt?

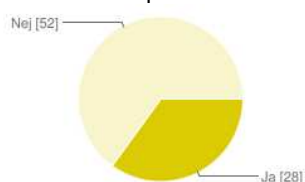


Er der noget specielt du gerne vil vide om en vin som normalt ikke oplyses på flasken?

Måske 'sammenligning' med andre vine. Lidt søgt metafor men lidt ala youtube's anbefalinger baseret på hvad jeg normalt drikker. Smagsprøver er jo ikke at kim se af. Om den egner sig til lagring Restsukkerindholdet kan til tider være interessant lignende vine, så man har et sammenlignings grundlag Jeg er særligt interesseret i andre brugeres anmeldelser af vinen, gerne inklusiv sjove fortællinger om hvad indtagelsen af netop denne vin indebar af skægge oplevelser - selvom det jo ikke er en konkret årsags-sammenhæng, men det ville være fornøjeligt at høre om :) Som I også foreslår, så er det meget ...

Telefonbrug

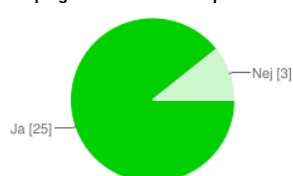
Er din mobiltelefon en 'smartphone'?



Ja	28	35%
Nej	52	65%

Smartphones

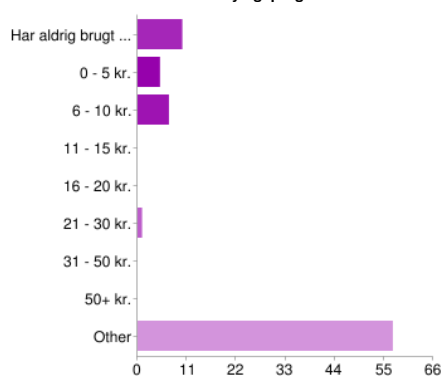
Har du hentet programmer til din smartphone?



Ja	25	31%
Nej	3	4%

Applikationer til smartphones

Hvad betaler du normalt for et nyttigt program?



Har aldrig brugt penge på et program til min telefon

10 13%

0 - 5 kr.

5 6%

6 - 10 kr.

7 9%

11 - 15 kr.

0 0%

16 - 20 kr.

0 0%

21 - 30 kr.

1 1%

31 - 50 kr.

0 0%

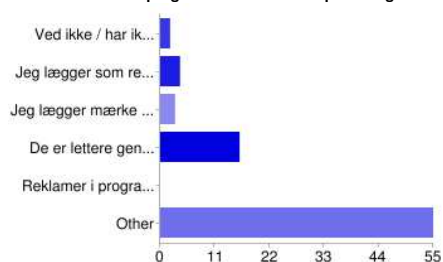
50+ kr.

0 0%

Other

57 71%

Generer reklamer i programmer til din smartphone dig?



Ved ikke / har ikke prøvet programmer med reklamer på min telefon

2 3%

Jeg lægger som regel ikke mærke til dem og de er generer mig ikke betydeligt

4 5%

Jeg lægger mærke til dem men kan leve med dem

3 4%

De er lettere generende og vil helst være fri

16 20%

Reklamer i programmer er så irriterende at jeg kan finde på at droppe programmet

0 0%

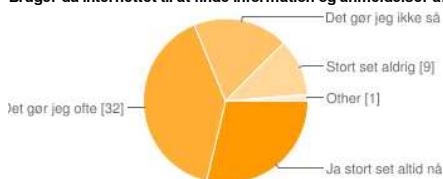
Other

55 69%

Diverse

De følgende spørgsmål gælder dit generelle forbrug, og er altså IKKE begrænset til dit brug af vin eller smartphones

Bruger du internettet til at finde information og anmeldelser af produkter inden du køber?



Ja stort set altid når det er muligt

23 29%

Det gør jeg ofte

32 40%

Det gør jeg ikke så ofte

15 19%

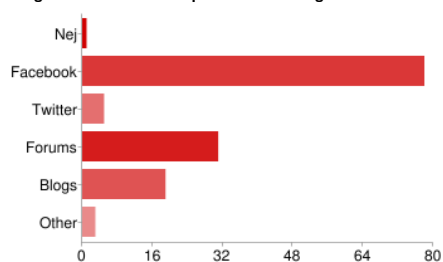
Stort set aldrig

9 11%

Other

1 1%

Bruger du sociale medier på nettet i hverdagen?



Nej

1 1%

Facebook

78 98%

Twitter

5 6%

Forums

31 39%

Blogs

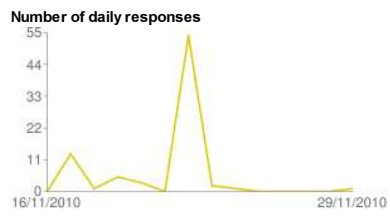
19 24%

Other

3 4%

People may select more than one checkbox, so percentages may add up to more than 100%.

Tak for din deltagelse - Husk at trykke på knappen send for at afslutte spørgeskemaet



**C Android Market Statistics 26. May
2011.**

D DADIU Summery

D.1 Broken Dimensions - Kasper Larsen Ekelund

D.1.1 Introduction

I was on the game Broken Dimensions as a Game Programmer as part of my second production at The National Academy of Digital, Interactive Entertainment (DADIU) in March 2011. The game is a 3D puzzle game where you play the role of a little boy named Theodor which is trapped in a big old Aztec pyramid. In here, the ghostly woman Tonantzin tries to kill and take over the poor little Theodor's soul. But of course he has magic super powers which allow him to rotate the world around him, like every other child in the center of the universe. He will now fight his way though the pyramid and escape by using these super powers! Rotating the world around the player that makes things fall down and apart will be the only game-mechanic. The player will with this simple game-mechanic solve puzzles in a 3D world where physics also play a role.

The game can be played at: <http://broken-dimensions.dadiugames.dk>



Figure D.1: *Broken Dimensions*.

D.1.2 Management

The whole team of the production consisted of 13 people where we were 3 programmers: 1 lead and 2 programmers. The management of the whole team was done by the project manager

and for the development management the Scrum approach was used which is an iterative management process. Every morning at 9 AM a scrum meeting was held which summed the progress of the last day work, goals of today and clearing any problems that one might have. The production team was split into smaller teams or departments such as Programmers, Art and audio departments, each having a lead. This approach worked well, and the daily scrum meeting in the morning successful clear problems that you might sit with. Also it makes the whole team updated on what is going on in the different departments, which without the meetings might go unnoticed.

D.1.3 Tasks

As a game programmer I had many tasks. Many tasks was small in magnitude such as polishing, setting up sounds, bug-fixing etc. Some of the other major tasks will briefly be covered in the following.

Player Going Though Walls

The way that the rotating mechanism was implemented made the player sometimes go though walls when the player was rotated. This was because the rotating was not done in the physics engine with forces and so on, so no collision detection would get performed in that step. When the player was rotated beside a wall the head of the player could accidentally be outside the ceiling or if on the floor the player could even fall through it. This was a very serious problem because it can ruin the whole gameplay. To avoid this some solutions were considered. The first one was to implement a custom collision detection against the vertices of a wall, but it quickly seemed that this was too complex, considering the limited time frame this task was given. The chosen solution was instead to do a manual check about the surroundings of the player and then move the player away from the wall before the player starts rotating at all, so completely preventing the player to be near any walls. This can be done simply by shooting rays from the player in all 6 degrees of the players bounding box and then decide where to move the player if any are blocked, see figure D.2. For example if the player faces a wall he should be moved away from the wall or if he is standing in a corner he should be lifted away outwards from that corner. The perfect case would be that no directions are blocked and if all are blocked, we cannot do anything, but hope for the best. However, that is very unlikely that the player should find himself in that kind of situation, but it is a drawback because if it can happen it will happen. But considering out time limits this was the best way to do it on such a short production.

Game Manager

To hold global state in the game such as what level the player has reached and other global settings of the game, a GameManager class was defined. It was written with the singleton design pattern so only one instance can exist.

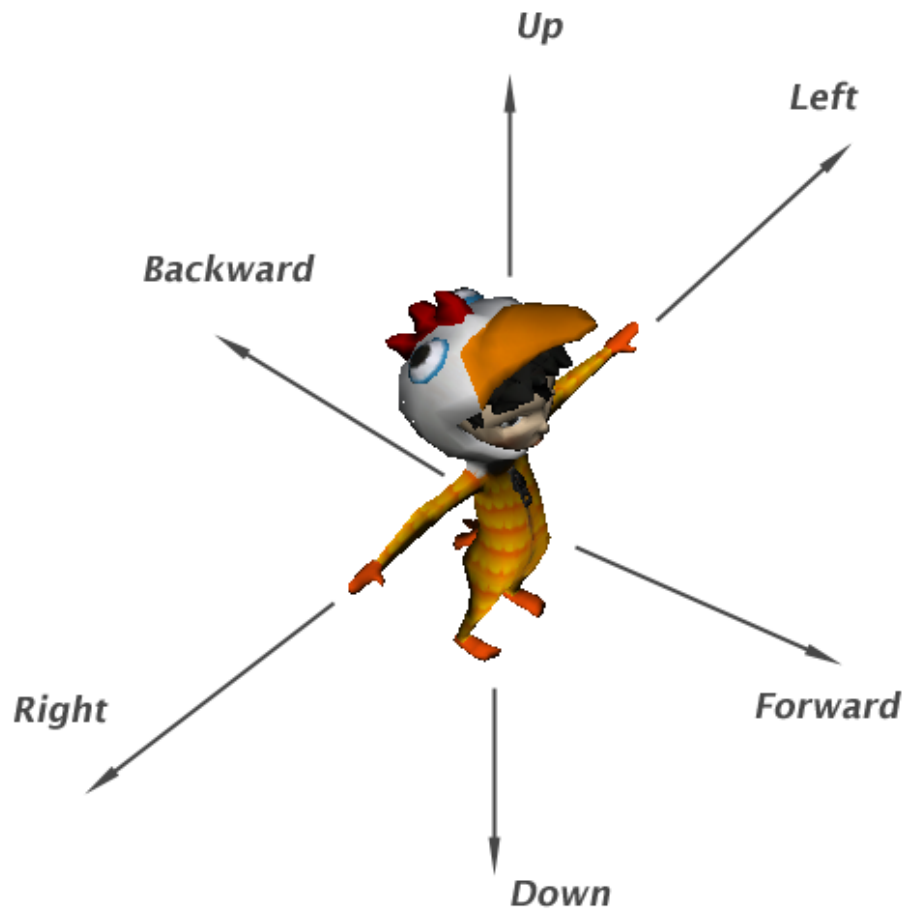


Figure D.2: *The raycasting directions for checking player surroundings.*

Audio Programmer

On the production I was assigned many of the audio implementation tasks. The audio-tasks is most of the time quite small because they do not require big coding issues, because sounds most of the time are triggered by something that is already implemented, and therefore already triggered. For example the footstep sound is a loop which are played every time the player holds down the move button and when he is grounded. Both of these conditions are already implemented and therefore its not a task that you have to rethink for a long time. I designed a class named SoundManager that would keep track of ambiance and music in the game. It is basically a state machine that when ambiance is turned on, it will keep looping the ambiance loop and if the music is on it will keep fading in and out random music tracks from an defined array of sound tracks. The fading between music tracks is done in a separate thread.

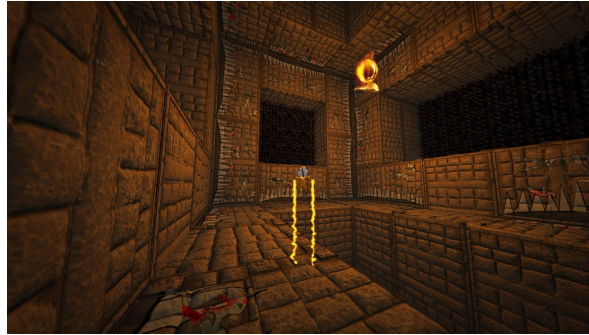


Figure D.3: *Broken Dimensions* screenshot



Figure D.4: *Broken Dimensions* screenshot



Figure D.5: *Broken Dimensions* screenshot

D.2 Blendimals - Jacob Sloth Laursen

D.2.1 Introduction

One big blending machine plus a whole bunch of animals = Blendimals!

Inspired by classic puzzle games like Lost Vikings and Lemmings, Blendimals takes you back to the roots of puzzle action while adding a new twist to the genre. Set in an iconic and classically styled game world, Blendimals mixes something completely new with something very familiar.

Blendimals is a slow paced, humorous and challenging game experience. As the player you will guide your Blendimals through an obstacle course filled with deadly traps and puzzles. You'll have to figure out how to solve the challenges and time your actions to complete the level. With indirect controls the game is a relaxed platform game with puzzle elements. It's a game you can enjoy while sitting at home having a cup of hot cocoa or sitting on the train with your iPad.



Figure D.6: *Boxcover for Blendimals*

Selling Points

- Create weird and funny creatures as you mix different animals in the blending machine.

- Dig, jump, smash, float and explode your way to victory as you combine different animal skills to complete the challenges.
- Feels new and fresh, but also comfortable and familiar.
- Replayable levels
- Think ahead and figure out which animal parts are best to use in a given level.

The game was designed for mobile device deployment such as iPad, iPhone and Android.



Figure D.7: Screenshot from *Blendimals*



Figure D.8: Screenshot from *Blendimals*

Gameplay Sum Up

It is very recommended that the reader of this document have tried the game before reading the following sections. This will give a better understanding when reading.

On the DVD in the folder called Blendimals, is placed a executable of the game, a long with the game trailer. The game can also be found on the website www.blendimals.com.

If it is not possible to play the game beforehand, here is a quick guide to playing a level:

The player starts a level. Before beginning the actual game, the player will pan around in the level to get a good overview. The player will notice any traps and hinderings, and on the basis of the, plan which animals to use for the level.

Now the player will bring forth the blender and begin dragging animals to the blender. The animals are blended in a non-violent manner, and out pops a new animal called a Blendimal. The blendimal will inherit all properties from the blended animals.

For example, blending a rhino and a bunny, gives the player a cloning between a bunny and a rhino. A cloning not just visual but also in abilities. The rhino-bunny blendimal will be able to hop over gaps, and smash rocks.

My Role

As technology is my specialization, my role in the production was set as a programmer. Of course i did take a minor part in almost every aspects of the project, but my main concern was to implement the following:

- Blendimal state machine
- Blendimal abilities
- Camera control
- User interface
- Sound

D.2.2 State Machine

The behaviour of a blendimal is pretty simple. In default mode the blendimal walks until it hits something and then turn around. The blendimal can be assigned different behaviours depending on the abilities.

There are many ways to implement a decision making behavioral artificial intelligence. But as there is only one thing the blendimal can be doing at a given time, a state machine is a good choice for handling the behaviour. A state machine provides a much better overview when coding and enables for strict controlled rule based behaviour. Especially a lot of conditions

is saved, e.g. when the blendimal do that i cannot also do that. With a state machine, the blendimal can be in one and only one state.

Implementing the State Machine

The implementation of the state machine was done by having an overall state machine class. This class had a list of all possible states, and handled the switching between different states. The state machine class was not supposed to determine when to change to a new state, and it only handled the changing when change was already decided. The actual change happened in the different states.

For the actual different states a class was written that should work as a parent class that all the other states were supposed to derive from. In this way it was secured that the states would all have the same basic functionality.

The parent state class contains references to different game objects like, the navigation control, the camera control, the mouse control, the sound control, etc. In this way each state would have access to all necessary game controls. It also contains reference to the rigid transform, to give all states the possibility to activate different animations. Lastly the parent class handles all collisions to be able to pass them to the each state subclass if needed.

In addition the parent class contains different abstract methods, where the most useful is: *EnterState*, *StateRunning* and *ExitState*. This secures that these methods are implemented in the different states. As suggested the three methods are called whenever a state starts for initialization of the state, are running for the behaviour, and exits the state for cleaning up.

A Typical State - Charge

Charge is the ability of a rhino used to smash rocks. When charge is activated through the user interface the charge state is set to the next state. In the next frame the state machine controller exits the current state, for example walk, and cleans up everything from that state.

Next the state machine runs the *EnterState* method. In the charge example, the enter state activates different things: A dust particle emitter, crossfading the animation to the charge animation, plays the charging sound, and starts a coroutine which is used to take care of the charging time.

In the following frames while the charge lasts the state machine continuously runs the *StateRunning* method. The running method ensures a velocity is added to the rigid body of the blendimal to make it run faster, and then it checks when to switch to the default state again.

When the charge is over, the default state is activated and the charge's *ExitState* method is run by the state machine. The method stops the dust particle emitter, and stops the playing of the charge sound.

D.2.3 Sound Manager

Although it is very easy to get sound to play in a game using unity, the need for a sound manager is understandable.

To have a sound played in unity, a audio source object is needed. The audio source depict where the sound originate from. This can for example be used in conjunction with 3D sound, and fading of volumen when getting closer or farther away from the sound.

A audio source can only play one sound at a time and does not support simultanious plays. There is a way around this by using something called *OneShot* method. What it does is that a audio source is spawned together with the sound that wants to be played. The drawback by using the *OneShot* method, is that it lacks control. You cannot acces the audio source object, and when the sound have played once, the object and the sound is destroyed, not allowing for looping.

There was also the need for playing random sound from a bucnh of sounds. For example when smashing a rock, it should not be the same sound that playes every time, but should be a random chosen among multiple.

Implementation of the Sound Manager

The sound manager is a superclass, that more specific sound mangers derive from. There is three subbblasses in blendimals.

- Global sound manager. Takes care of music and other global sounds that should be played across the game across the levels.
- Blendimal sound manager. Manges sounds that is emitted from the blendimal.
- Environment sound manger. Takes care of all environment sounds.

The super class has all the basic methods that the subclasses use to play its sounds. It consists of three groups of methods that all have various overloaded methods to be able to server different needs:

- Playing a sound. This plays a single audio clup which is passed to the methods.
- Playing a random sound. This plays a random sound from the passed array.
- Playing a sound using the *OneShot* method. This just plays a sound. The force is that it does not need a audio source.

Every group of methods have various overloaded methods. For example for normal play of a sound there is four methods as listed here:

1. `protected void PlaySound(AudioClip clip)`
2. `protected void PlaySound(AudioClip clip, AudioSource audioSource)`
3. `protected void PlaySound(AudioClip clip, AudioSource audioSource, float pitch)`
4. `protected void PlaySound(AudioClip clip, AudioSource audioSource, bool silenced)`

The sound manager makes it really easy to play a sound to the specific need. For example when starting gameplay music the `PlaySound` method is passed the audio clip and a audio source. In that way we have control over the audio source and can for example make it loop or fade out and in.

When play a rhino smash the we dont need to control the audio source, and to save code instantating the `OneShot` method is used and an array of smash sound is passed for a random pick.

Sub Classes

The sub classes of the sound manager class, hold all references to the different sounds. And holds methods that ensure that the right methods from the super class is called.

D.2.4 Camera Control

The camera follows the blendimal wherever it goes. When a blendimal is not present on the screen, either because it is dead, or because the player have just started the level, the player can freely move the cammera around.

A iPad/iPhone feel was wanted when the user pans the 2D level:

The player clicks the screen and drag like if he was grapping the level. The level should move excatly the same distance as was dragged. What it means is that if the player presses on a point in the level and drag the screen, when the player releases the level has been panned with the same amount so that the point still is at the mouse position.

To make a one to one panning of the cammera some steps have to be done, to make the right calculations. In every frame the delta mouse position, in game world coordinates at the level plane, is calculated. This is the velocity of the camera.

To make the camera slide in a nice movement, a build in lerp method is used, to interpolate between the cameras start position and the new position.

The palyer should also be able to add velocity to the camera by releasing while dragging. This means that the camera continues to to pan after releasing, slowly decreasing the velocity of the camera. It gives a smooth and nice feel when dragging the camera around in the level.

To achive the sliding simply continues with the last calculated velocity, and slowly decreasing. To prevent from making abnormal velocities, the velocity wil have a max.

D.2.5 Camera Shake

To make a nice shake of the camera when for example smashing rocks, a simple camera shake method was made. It works by alternating between moving the camera a bit left, right, up and down. The amount of movement is dependent of the time duration. This means that the shake is most intense at initiation of the shake and slowly degrades.

