eBogreolen.dk - Android App
Anden Delrapport

Tobias Hallundbæk Petersen - 081092
Nikolaj Dybdahl Rathcke - 180692
Ola Rønning - 200190
Victor Petrén Bach Hansen - 130892
Projektgruppe-id: A6

Instruktor : Andreas Frisch

April 25, 2013

Contents

1 Abstract

We are making an Android application for the company Ebogreolen.dk. This application needs features that enables their customers to search for e-books and audiobooks. The application should make the users able to buy e-book and audiobooks. This means that there needs to be a payment method for the e-books and audiobooks. Additional features, if possible, would be to implement a way of reading the e-books and hearing audiobooks though this is not a requirement for the finished product.
The collaboration is done in the group with meetings each week where we discuss what subjects should be completed in near future and what subjects that can wait until later. These tasks are fulfilled in get-togethers on weekends and weekdays if needed, this article specifies how far we are in development, how the group work is going and the progress in general.

More specifically, the functional requirements of the project described with different diagrams. It discusses the user interface with screens of the user interface as well as a description of the interaction between the user and the application.

2 The purpose and constraints of the IT-solution

This is based on the FACTOR-concept, and explains the scope of the project.

2.1 Functionality

1. A way of buying and paying for books.
2. Ability to download and use the bought books.
3. A search and browse functionality.

2.2 Application domain

The application domain is Android phone users who are customers at Ebogreolen.dk.

2.3 Conditions

The specification of android devices vary greatly and the application should run evenly on all devices.

2.4 Technology

The system will be developed entirely in Java with the Android SDK and will be run on Android smartphones with Android version 2.3 or higher.

2.5 Objects

The objects of the Application are the Ebooks, Audiobooks, an online bookshelf, and a user account.

2.6 Responsibility

Making the user able to browse, buy, read and listen to ebooks and audiobooks.

3 Requirements specification

This section explains what is to be expected of the application when it is completed, furthermore it also specifies different use cases and the problems that might appear. 1

3.1 a)

Functional requirements:

1. Login capability.
2. Purchase ebooks and audiobooks in the Android application.
3. Show purchased ebooks and audiobooks in an account specific bookshelf.
4. Download purchased ebooks and audiobooks available from the eBogreolen website.

Non-functional requirements:

1. Stability and reliability. We need a stable application, for example, to make sure all transactions are atomic.
2. Usability. We want to make sure that the application is as easy and intuitive as possible, for the best costumer experience.
3. Security. Since we are dealing with others peoples money, we will need a secure system.
4. An offline-mode, for reading books when not connected to the internet.

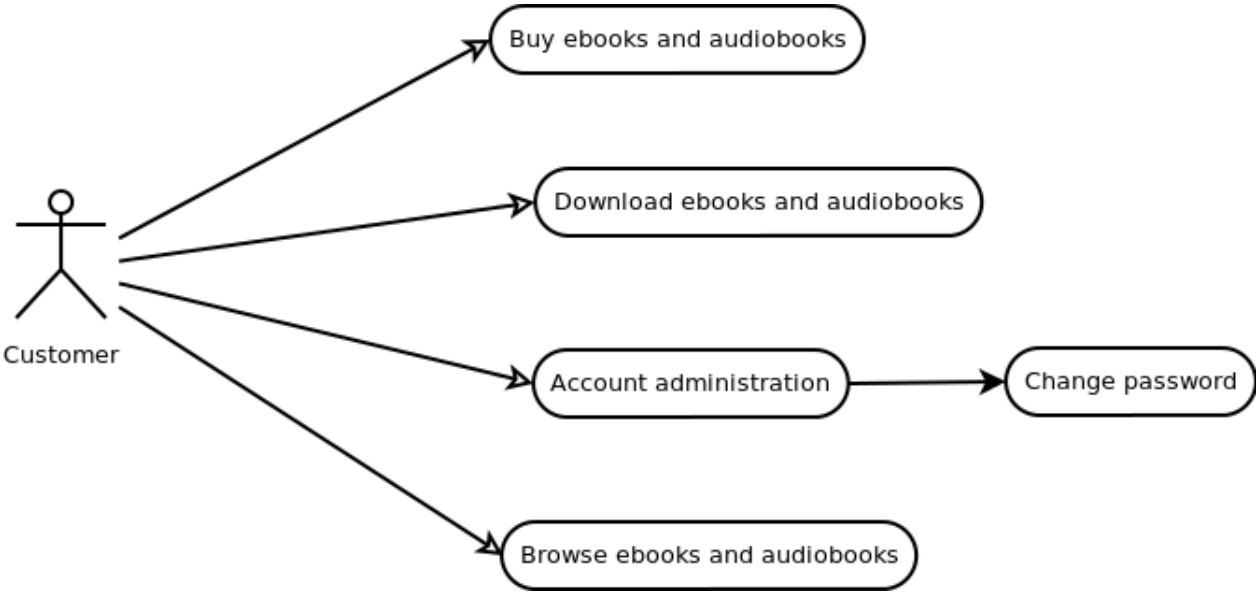

Figure 1: A case model describing the use cases of the Application.

3.2 b)

The casemodel shown in figure 1, shows what different cases the user can experience using the Android application. This is a rather simplified diagram, showing the key features that a user will experience.

3.3 c)

1. Title: Logging in
   Main Success Scenario:
   Step 1: The user presses the log in button.
   Step 2: Then the user enters his/her username and password.
   Step 3: He/she then submits the information by pressing a button.
   Step 4: His/her books are now loaded to the device.
   Extensions:
   Extension 2a: The user writes a wrong username and/or password.
   Extension 3a: There is no internet connection.

2. Title: Buying book
   Main Success Scenario:
   Step 1: The user presses the buy button.
   Step 2: Then the user enters his/her credit card details.
   Step 3: He/she then submits the information by pressing a button.

Step 4: The book is added to the users account.
Extensions:
Extension 2a: The user writes wrong card details.
Extension 3a: There is no internet connection.
Extension 3b: The transaction was denied.

3. Title: Search for a book
Main Success Scenario:
Step 1: The user presses the search bar.
Step 2: A search query is written.
Step 3: The query is committed.
Step 4: The results are shown.
Step 5: A book is chosen.
Extensions:
Extension 3a: There is no internet connection.
Extension 4a: The query returned no books.

3.4 d)

The class diagram shown in figure 2, describes the outline of the solution-domain. This is not the final design, as there are many uncertainties, which is described in depth in section 4.

3.5 e)

The sequence diagram shown in figure 3 shows the chain of inner workings that is executed when the user tries to log in.

The sequence diagram shown in figure 4 shows the chain of inner workings that is executed when the user tries to buy a book.

The sequence diagram shown in figure 5 shows the chain inner workings that is executed when the user tries to search for books.

4 Systemdesign summary

As of right now, the system is a number of classes and methods that was foreseen would be needed in the final design. Since the way we will receive the data from the queries to the database is still unknown, data specific classes and methods are yet to be made and

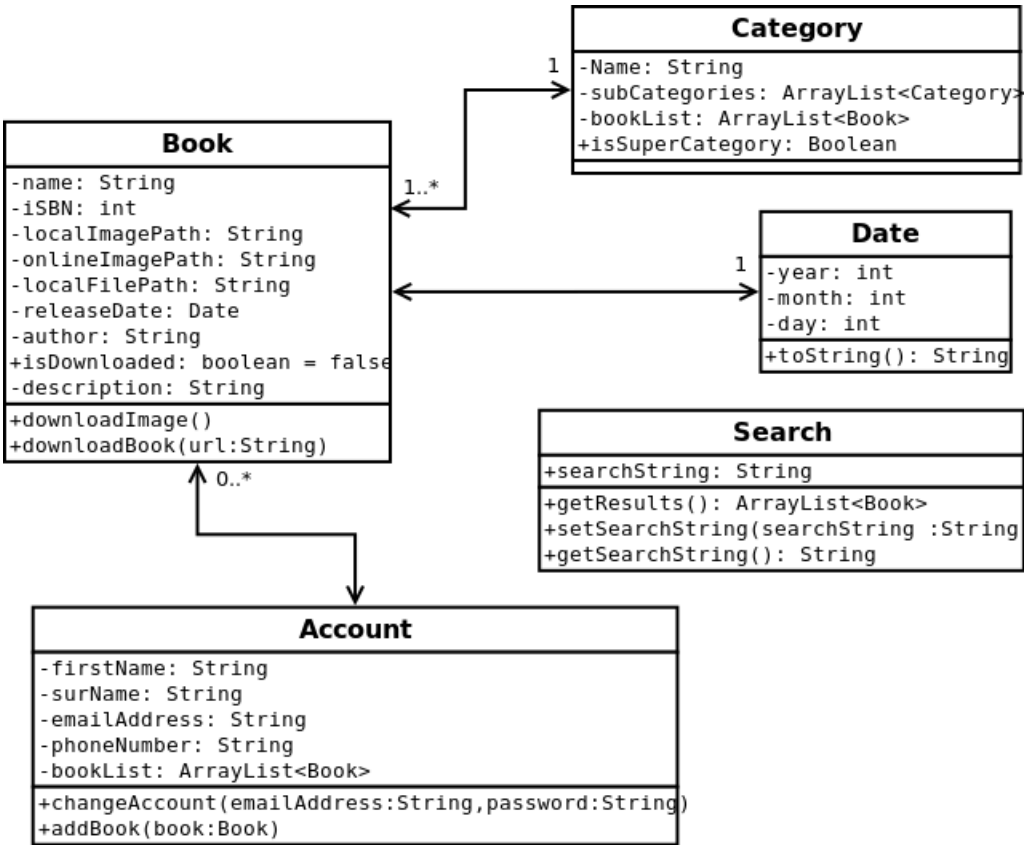

Figure 2: A uml diagram over the Application

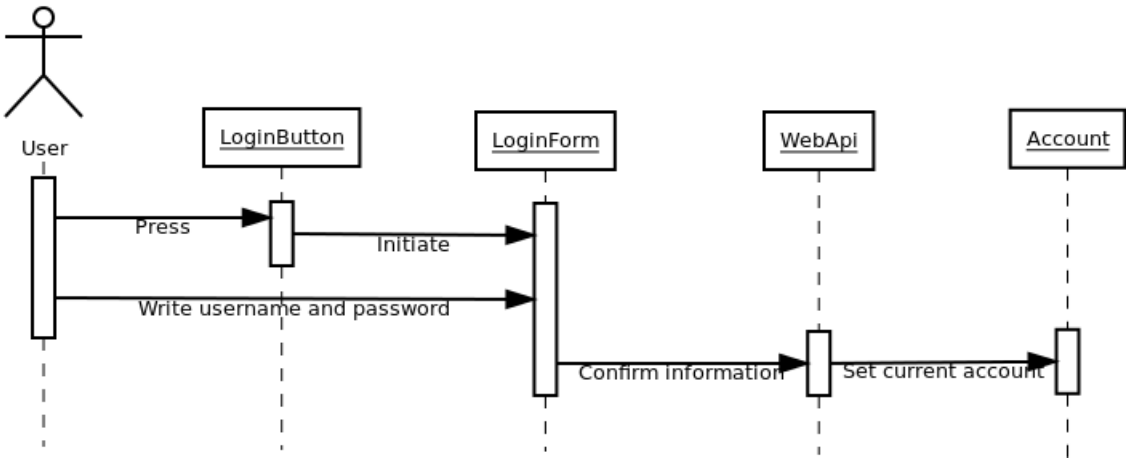

Figure 3: A sequence diagram for when a user logs in

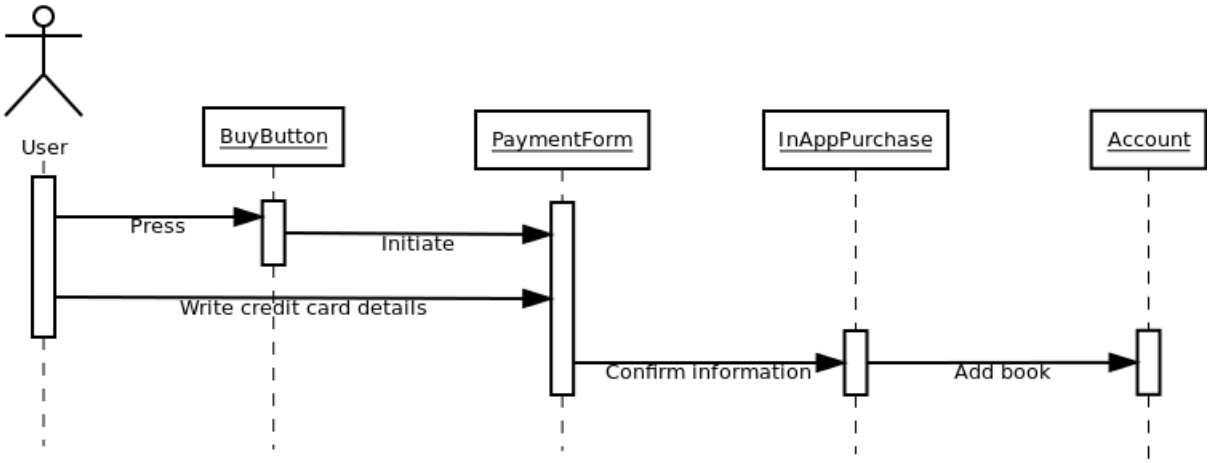

Figure 4: A sequence diagram for when a user buys a book

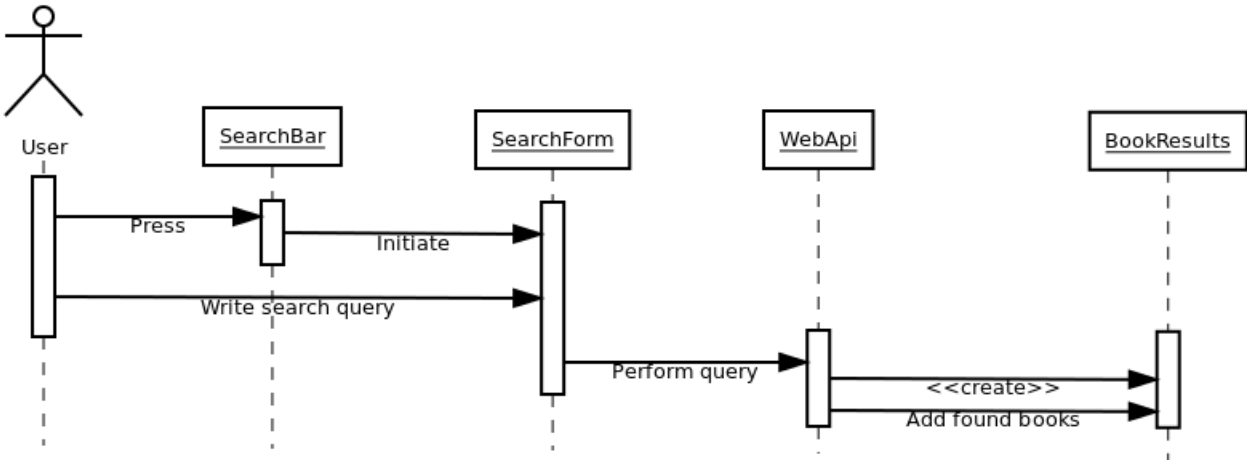

Figure 5: A sequence diagram for when a user searches for a book

implemented.

Some of the most important remaining design and implementation task are as follows:

GUI:
The Android application user interface and the various action listeners that is needed to get this to work satisfactory on all Android platforms of version 2.3 or higher.

A payment solution:
When a customer wants to buy a book, the system for handling this is needed. The Android SDK provides a tool for this, but it has yet to be implemented and fully understood.

The WebAPI:
This will be implemented by eBogreolen.dk's developer Anders Hyldig and will facilitate the interaction with the database. We expect that this will return all information about a book in the store by providing the ISBN number. Furthermore, we expect that, from a search string, it will return a list of books as a result of the search query.

Database interaction:
The queries that the user makes regarding bought books and browsing for new books needs to be taken care of, the implementation of this depends on the WebAPI

5 Program and system test

Our intent is to use think-out-loud testing using friends and family as our test subject. We have deemed this appropriate as the Android Application market, along with our client, has a wide range of users. We have not started planning the usability testing yet, as a working version of our application is not developed yet.

We are setting up a JUnit testsuite for the backend as part of the next sprint.

6 Userinterface and interaction design

6.1 Userinterface and interaction

The external interfaces are here understood as the finished product GUI, for which we have a course outline as it follows in Figure 6,7,8 and 9. The application GUI for the login page has not been developed yet.

Figure 6:
1. Open a page containing the information of a purchased book. This will lead to Figure 7.

2. Search/browse after more books to purchase. This will lead to Figure 8.

3. Swipe to the side to look at more of your books.

4. This button will always go one page back. If there are no more pages to go back to, the application will be closed.

A. This book is darkened, because the book has been purchased, but not downloaded.

Figure 7:
1. Swipe up and down to read a description of the book.

2. This button can be a: Buy, Download, Read or Listen button, this depends on whether or not you own the material and/or if its an audiobook or ebook.

3. Tapping the image enlarges it to a full screen view, tapping the image while enlarged will return to the displayed page.

Figure 8:
1. This will access the search function, when a search is complete it will direct you to Figure 9

2. The will open the category in this interface if it is a super category, and go to Figure 9 it is a sub category.

3. Swipe up and down to browse the categories.

Figure 9:
1. This will take you to the information of the given book, this will lead to Figure 7.

2. Swipe up and down to browse the results.

6.2 Audio-visual presentation

On this stage we do not have a prototype of our product, but the final product is expected to look like what's given in section 6.1, therefore an audio-visual presentation would be pointless.

6.3 Latest think-out-loud results

As described in section 5, a think-out-load test has not yet been performed, and therefore no results are present.

7 Versioncontrol

The following is the commitlog for our repository at https://github.com/VixQIT/EBogreolen where our code also is accessible:

```
$ git log --pretty=oneline

258566b216827dd09a6186042069f4ecb563faf5 Merge branch 'master' of
https://github.com/VixQIT/EBogreolen
5176586754383ad696ecc8da6e3b1a089b611ad1 First commit
86bf9e9a7090a87a986ecf6bf7ec34ed094a56f7 Initial commit
```

As this is the first time committing any code to the repository, we have not made a whole lot of changes to it yet.
Some of the classes are dependant of other classes which have not been created or fully implemented yet, due to the fact that we do not have a WebAPI to access the clients database currently.

8 Project collaboration

We have developed a 'game' using "experience' points and the opportunity to 'level' up. These point are distributed to the group participants by a point generator on: `http://www.VixQIT/XP/XP.php` that generates a random amount of point in a certain range. The range is calculated from tasks completed. Tasks are worth points equal to the hours of work estimated during our scrum and these translates into points that provides the range from your random gain of experience.
The experience needed to level up increases each time using the Fibonacci sequence.
This game a motivation for us since there are small benefits (bragging rights, beers etc.) and the idea of making it competitive is appealing to all of us. While keeping the simple benefits, it won't ruin the dynamics in the group. Keeping it on a low level, a fun sidekick, we keep the ambitions and motivation high while still maintaining our good teamwork.

As for the communication with the client, we have not met physically in the last month, but we have a email correspondence established, where we contact each other weekly. We are currently still discussing design matters when it comes to the WebAPI, that is under development by Anders Hyldig.

9 Review: Designing for usability: key principles and what designers think

In the article, Lewis and Gould describes, what they believe is. The three principles for system design that is to be followed, if you want to make a useful and easy to use computer system. The three principles they recommend are: Early focus on users and tasks, empirical measurements and iterative design. Even though these principles may seem intuitive to many, designers tend to avoid them, even if they think they're following them. Lewis and Gould points out many times, that user testing from a very early stage is crucial for the developing process, if a good system is to be made.
The article then proceeds to present an example, where their philosophy is being used successfully, in the development of IBM's ADS.

The article correctly predicts that usability will be a key aspect of future software, which we can confirm. Software developed today for the mainstream masses is heavily focused on 'easy of use', in order to appeal to an audience as wide as possible. An example of this is the success of almost all Apple products.

The article suggests a design process that extends throughout the development of the software project. This design method, as described by the article, differs from the book's scrum model in that the final design is fine polished before a new sprint starts, hence leaving the final product a tuning of the initial design. As we understand the article, the iterative design is continuously redesigning the product based on user input, thereby significantly remodelling the product from the initial design.

A development process with focus on usability seems rather logical. The article makes the case that the ideas they are presenting are not intuitive. Our group was under the same impression as many of the interviewed developers and, though the article makes a compelling case, are still not convinced that user friendly development is something that programmers naturally stride towards.

10 Review: A rational design process: How and why to fake it

The article describes how we should strive to create an ideal rational design process, when developing computer system, and how this systematic approach, in their opinion, always will be unattainable. Instead, we can fake this process by showing the system to others so that it would appear to be designed rationally. They go into depth of explaining why the process is useful, what the process actually consists of, the importance of good documentation and how to fake relational design.

Rational design process should encourage the use of legacy software. In the listing reasoning why the rational design process is an idealization, the authors argue that the use of legacy software inhibits the ideal software ref(1 II 7), since the legacy software is not written based on the requirements alone. Assuming that the process of faking the ideal design is adopted as a standard, identifying legacy software, that can be implemented according to the requirements, would be in most cases, not just economically sound, but the very ideal of rational design. The legacy software would come with the documentation needed, hence minimizing the amount of new documentation, that the developers would have to produce. Furthermore adoption of terminology from the legacy software documentation into new production of documentation, would further standardization both within an organization and with the software development community as a whole. The apparent contradiction between the use of legacy software is also indirectly acknowledged by the authors in their conclusion "... the result is a product can be understood, maintained, and reused".

The OOSE book and the article differ greatly in their treatment of requirement documents. The article argues that this should be a formalized document, preferably developed by the client or representatives of the client. The OOSE book separates this document into two stages, a requirement specification produced by the client, users and developers in a natural language, which is then formalized by the developers in an analysis model. The iterative model of the OOSE book is for us, a group of novices in designing and developing software, much more appealing, as the task at hand is not as overwhelming. Overall the rational design process requires extreme understanding of the design process, which we have yet to obtain.