LIST OF CORRECTIONS

ACKNOWLEDGEMENTS

We would like to thank our supervisor, Ulrik Mathias Nyman, for both the extensive and professional guidance as well as motivation. We also like to thank *Naked Fruit*[1] for supplying refreshing drinks. Thanks to Henrik Sørensen and Michael Skov, for advices during the design process. [1]
As a multiproject group, we wish to acknowledge the help and time spent by the educators and teachers from various institutions, whos insight have been of utmost value in the development of the GIRAF system.

1 FiXme Note: add peter og Ivan for advice

Frontpage!!!

Aalborg University

Department of Computer Science

Student report

Selma Lagerlöfs Vej 300

9220 Aalborg East

http://www.cs.aau.dk

Title:

Project title²

Theme:

Applikationsudvikling / Application Development

Project Term:

P6, spring 2012

Project Group:

sw605f12

Students:

Kasper Møller Andersen

Magnus Stubman Reichenauer

Rasmus Steiniche

Thomas Kobber Panum

Supervisor:

Ulrik Mathias Nyman

Copies: 6

Pages: 91

Finished: June 4th, 2012.

Synopsis:

The ""title"" project involves an Android™ application called the launcher. This launcher is made for use by both guardians and children with ASD. It tries to solve some of the everyday problems a child with ASD can have and make the typically high costprice go away, it is free! This launcher is made as the base for all interaction with the GIRAF system and all the apps in it. It includes the GIRAF GUI library which is an library for all GUI elements used in the GIRAF system.

Kasper Møller Andersen

Magnus Stubman Reichenauer

Rasmus Steiniche

Thomas Kobber Panum

PREFACE

In deciding upon a report structure, two main approaches were considered.

1. Traditional analysis, design, & implementation-structured product oriented report.
2. "Diary" iteration-structured and process oriented report.

The strength of the first approach is the clear way the product would be presented, as it would be easy to understand the analysis, design, and implementation done of the product. The weakness is the ability to represent the reasons for the design choices and the implementational details, as the project has alternated between analysis, designing, and programming activities, as is customary in the iterative work process used.
The strength of the second approach is that the ordering of events and decisions is chronological in the report, making the reasoning clearer and more intuitive to follow. The weakness is the lack of a clear and structured way to easily grasp and understand the final state of analysis, design, and implementational work done.
We have chosen a mix of the two, where we first focus on process oriented activities, their chronological order, and what their effects were, followed by a product oriented description of analysis, design, and implementation.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

GLOSSARY

XP Extreme Programming

GUI Graphical User Interface

ADS Autistic Spectrum Disorder

Guardians Parents, teachers, caretakers, and educators of children with ASD.

we In the introduction, "we" denotes the multi-project group. Every other place it denotes our individual group.

Children "children" refers to "children with ASD".

MVC Model View Controller [**?**]

WOMBAT name of the timer application – abbreviation of Way Of Measuring Basic Time.

PARROT name of the pictogram application, Pictogram Assisting with Rhetoric Reasoning Or Talking

Part I

PROLOGUE

1 INTRODUCTION

In order to describe the context of the system, we – as a multi project group – will in the following state the motivation of the project, the group of people we are aiming at helping, the technological platform chosen, the used development method, followed by a problem definition, a system description and architecture, and the conducted usability test.

1.1 MOTIVATION

As this is a student report written as part of a learning project, we are required to comply with the study regulation. The main areas of focus, according to the study regulation, are: multi-project management and quality assurance in the form of requirements analysis, requirements management, and testing. The goal is to create a comprehensive software system, across multiple project groups, in order to enhance our competences in analysis, design, implementation, and evaluation of software applications in regards to the system requirements[**?**].

This project builds on top of a previous project, and is further developed, with the aim of having other students continue the development. The goal of the project, we are building on top of, is to create a touch based tablet system to support children and their guardians in everyday scenarios.

1.2 TARGET GROUP

Our target group is both children and their guardians. These guardians have certain needs for special tools and gadgets that help to ease the communication between them and the children.

Five teachers and educators, who work with children, act as customers. They will provide requirements and information about the institutions' way of working to give us an insight into their daily struggles.

1.2.1 *Working with Children with ASD*

This section is based upon the statements of a woman with ASD [**?**], explaining what it is like to live with ASD, and an interview with an educator at Birken, a special kindergarten for children, see appendix **??** for interview notes.

People with ASD are often more visual in their way of thinking. Rather than visualizing thoughts in language and text, they do it in pictures or visual demonstrations. Pictures and symbols are therefore an essential part of the daily tools used by children and the people interacting with them. Also, children can have difficulties expressing themselves by writing or talking, and can often more easily use electronic devices to either type a sentence or show pictures, to communicate with people around them. Another characteristic of children is their perception of time. Some of them simply do not understand phrases like "in a moment" or "soon", they will need some kind of visual indicator that shows how long time they will have to wait.

Different communication tools for children with autism already exist, but many of them rely on a static database of pictures, and often these has to be printed on paper in order to use them as intended. Other tools, such as hourglasses of different sizes and colors, are also essential when working with children, and these tools are either brought around with the child, or a set is kept every place the child might go, e.g. being at an institution or at home.

There exists tools today which helps the guardians in their daily life, although – as stated in Drazenko's quote – none of them are cost-effective enough to be used throughout the institutions. From the quote, it is clear that there is a need for a more cost-effective solution.

> The price of the existing solutions are not sufficiently low such that we can afford to buy and use them throughout the institution.
>
> - Drazenko Banjak, educator at Egebakken.

1.3 TARGET PLATFORM

Since we build upon last year's project, we are bound to use the platform they used, which is tablets running the Android operating system.

In this project we have been provided with five Samsung Galaxy Tab 10.1 devices[**?**]. The firmware on the tablets is version 3.2. This version, as of project start, is the latest stable version available for these specific tablets. [**?**]

1.4 DEVELOPMENT METHOD

As a part of the study regulation we have been required to use the same development method in each individual group. Two methods have been considered, *XP* (eXtreme Programming) [**?**], and Scrum [**?**].

With the knowledge of both XP and Scrum, we decided in the multi project to use Scrum of Scrums, which is the use of Scrum nested in a larger Scrum project [**?**].

The reason for choosing Scrum of Scrums is that everyone, at all times, will be able to know what the vision of the project is, and how close every group is to achieving their individual goals of the vision.

Another element of the Scrum method is that a close contact with the customers is maintained. This helps keep the product backlog up to date and correctly prioritized. The customers are presented with the vision of the project, as well as showing the latest release when we have meetings with our customers.

We customized Scrum to fit our project. The changes are as follows:

- The sprint length have been shortened to approximately 7 - 14 half days.
- Some degree of pair programming have been introduced.
- There is no project owner because this is a learning project.
- Everyone is attending the Scrum of Scrums meetings.
- The Scrum of Scrums meetings are only held once at sprint planning.

1.5 PROBLEM DEFINITION

The problem statement is as follows:

> How can we ease the daily life for children with ASD and their guardians, while complying with the study regulation?

This problem statement is necessarily vague to allow the individual groups some freedom in their projects, while we maintain the overall structure of the multi project, however there are limiting factors. We are limited by resources and time available, as we are only working on this project for a single semester. However, all work done in this multi project will be passed on to the next line of students, which means we can make a full system design and pass on anything we do not have the time or resources for. This also requires that our work need to be of such quality that it is understandable by students of the same educational level as ourselves.

1.6 SYSTEM DESCRIPTION

GIRAF is a collection of applications, either fully or partially interdependent, for the Android platform, designed to be used by guardians

and children. GIRAF consists of five projects with various degree of interaction. These projects are named Launcher, PARROT, WOMBAT, Oasis, and Savannah. Each of the groups have produced individual products, which are parts of a greater project, GIRAF.

Launcher handles execution of GIRAF apps, and at the same time it provides safety features to ensure that a user that is not authorized to interact with the rest of the system will not be able to do so. When the launcher executes an app, it will provide it with profile information, specifying which child is currently using the app, as well as which guardian is signed in.

PARROT is an app which provides access to pictograms – pictures with associated information such as sound and text – which can be used for communication. PARROT also gives guardians functionality for adding additional pictograms, as well as organizing the pictograms into categories for ease of access, based on the needs of the individual child.

WOMBAT is an app which purpose is to help the children to understand the aspect of time, by visualizing it. WOMBAT provides different ways of displaying time, as well as the possibility to configure the app for the needs of individual children.

Oasis locally stores the data and configuration of the GIRAF platform, and provides an API to access it. The stored data and configurations are synchronized to the Savannah server, if available. In addition, an app is provided for the guardian to access the stored data and configurations.

Savannah provides Oasis with a way to synchronize tablets running GIRAF. Furthermore, a website is provided to ease administration of the synchronized data.

1.7 ARCHITECTURE

Our System architecture – shown in Figure 1 has been designed with simplicity in mind and was greatly inspired by the MVC pattern. This means that the architecture is divided into three layers. The lowest layer is the database where the information is stored. Above this layer is the controller layer which, in the GIRAF platform, is known as Oasis. The controller is responsible for querying the database for information needed in an app and the controller is also responsible for storing information in the database. The last layer is the apps. This division of layers give the GIRAF platform a low cohesion which makes it easier to work with individual parts of the platform independently.

We have chosen to redesign last year's architecture **[?**] to make it easier to work with. We have simplified the architecture because we feel it is unnecessarily complex.

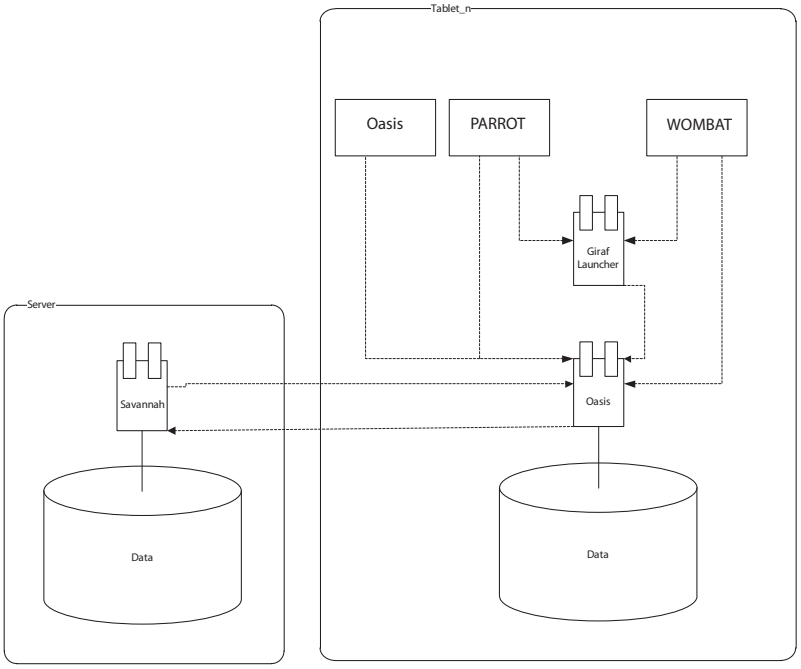


Figure 1: The GIRAF architecture

1.8 USABILITY TEST

As stated in the motivation, quality assurance through testing of the system is required. Therefore a usability test was conducted in order to measure the current usability of the GIRAF platform as a whole, as well as of the individual parts of the platform. Furthermore, the next wave of developers will immediately be able to start correcting the found usability issues.

1.8.1 *Approach*

The test group consists of the five contact persons. We assess that they, as a test group, are representative. We base this on them being a mix of educators and teachers, with varying computer skills.

They have prior knowledge about the overall idea of the GIRAF platform, and although some of the contact persons had previously informally used some aspects or parts of the system, they had not been exposed to the platform as a whole, and therefore still are of value.

The invitation sent to the test persons can be found in **??**.

The Instant Data Analysis (IDA) method for usability is chosen. A traditional video analysis method could be used, but since IDA is designed for small test groups, this approach is used. [**?**]

Setup

The usability test is divided into two tests: A test of three user applications, and a test of two administrative applications. The user applications are: The launcher, PARROT, and WOMBAT. The administrative applications are: The Oasis app and the Savannah web application. Each test is assigned a team to accommodate the need to run two tests simultaneously. The teams are made with respect to the criteria of the Instant Data Analysis process.
Each team consisted of:

- 1 x Test Coordinator
- 1 x Test Monitor
- 1 x Data Logger
- 2 x Observers

The usability lab at Aalborg University is designed with two rooms for usability testing and a control room to observe and record the tests. The two test chambers are assigned a test each and the control room is used to observe both tests as seen in figure 2.

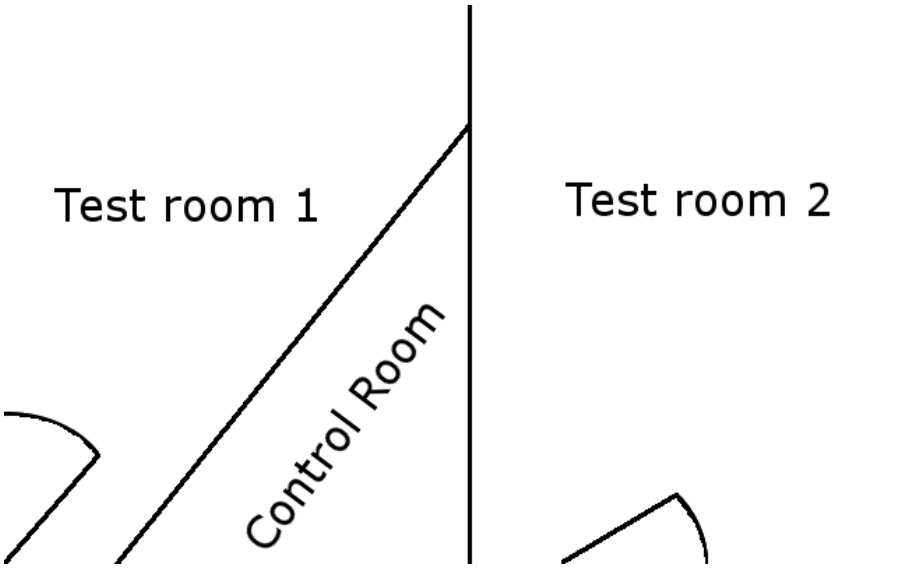


Figure 2: An overview of the usability lab at Cassiopeia, Department of Computer Science, Aalborg University.

Execution

Figure 3: The schedule of the usability test.

The tests are conducted according to the schedule in Figure 3.

Briefing, debriefing, and questionnaire documents can be found in Appendix D, and the results of the test can be found in Section 7.3.

Part II

PROCESS

2 PREANALYSIS

[1]

2.1 WORKING PROCESS

The working process of this project is based on agile development methods, mainly Scrum and XP. There are two work forms used by us in this project, the one used by the entire multiproject group and the one used by our project group.

2.1.1 *Multiproject work form*

The work form of the multiproject group is designed around Scrum of Scrums. With it, the work for all the project groups is split into sprints, and each project group works in synchronization with the other project groups. Each sprint is started with a meeting, where each project group presents their progress so far and what they are going to accomplish in the coming sprint. These meetings are also used to vote on the length of the coming sprint, knowledge sharing and decision making for decisions that affect the entire multiproject group.
Each sprint is ended with an evaluation meeting, used to briefly sum up and evaluate on the sprint, such that each project group recieves feedback on how they might improve their next sprint, before that sprint is planned. When two or more project groups need to work on something not related to the multiproject group, no formal meeting is required, and open discussion amongst the project groups is encouraged through an open door policy. This has been important, as many project groups rely on each other to provide services. This has also led to continuous, informal integration testing of the system, as each project group increased their reliance on the others. [2]

2.1.2 *Launcher work form*

[3] The work form used in our project group is primarily based on XP. XP is built around having a customer available at all times, but

1 FiXme Note: lav ny head - husk at den skal vÃŠre konkret og opsummerende om hvad der staar i kapitlet
2 FiXme Note: Flet sammen med common report. Meget af det der staar her burde egentlig staa i common report
3 FiXme Note: Find god overskrift

this has not been the case for this project. We found it unnecessary to compensate for this however. This was due to the nature of the project, which, in the case of the launcher, was focused mainly on the GUI. The only requirements made by the customers in relation to the GUI, was that it should have high usability. This allowed us to spend time refining designs, and only needing occasional feedback from the customer to evaluate the current ideas and design. The GUI focus also made it easier to fill out the backlog without the customer, and planning poker, a Scrum practice, was used to determine the size of each task.
Other XP conventions did make it into the work form though, e.g. pair programming. Pair programming has been a great tool in keeping the development pace up, as assisting each other in this manner makes it easier to discover problems and solutions early, while also reducing overhead in communicating code to the rest of the team.
Another XP convention that helped reduce this overhead was refactoring. This involves going through existing code to rewrite parts that are complex from a readability point of view, in order to simplify the code and make it easier to understand. This is essential, as the project will be handed over to a new team later on, and high readability helps ensure that the project is useful to them.
The "whole team" and "sustainable pace" conventions were used as well. The reason for this was to ensure that our work remained high in quality, and led to fairly stringent work rules, where the team agreed to work through the day together, but not work at home. Exceptions were made in case of illness. Lastly, "collective code ownership" was also employed. However, this is a demand from the study regulation, and not something that was deployed based on personal judgement.
One final note is that test driven development was considered as a possible convention to use as well. We found that GUI programming does not lend itself well to writing tests ahead of the actual code however, and so this convention was not included.

2.2 UNDERSTANDING

In order to understand the customer domain, interviews were performed.

2.2.1 *Interviews*

The customer domain contains two kind of users: Guardians and children. We set the scope of this project to include the guardians as primary users, with the children being only peripheral users. For this reason, the guardians are the customers of the GIRAF launcher, and are at the same time expert users of the customer domain. To gain good understanding of the customer domain, each of the customers

available in the project, were interviewed. The interviews were semi structured, in order to create a solid base of questions, while having flexibility, in case exploring additional topics would become relevant [2, page 152]. Two types of interviews were performed: One conducted by non-fixed subgroups of the multi-project with the customers divided between each of the subgroups, and the other type of interviews were conducted later by each of the project groups with their assigned customer. The first type of interviews were done at the university, in separate rooms. The second type of interviews were performed at the customer's workplaces.

The results from the first interview round showed that usability and flexibility are critical for the customers. Usability was requested based on experience with previously used solutions, including expensive learning courses, and flexibility was deemed critical, due to individual children potentially having very different needs.

2.2.2 *Field Observations*

As part of the interviews were conducted at Egebakken[4], observations were also made while at the site. These observations gave insight into the tools used (physical and digital), the robustness of the environment and the organization needed for the children.

2.2.2.1 *Impressions*

Egebakken is a robust environment, in many ways resembling public schools, though with a greater focus on serving the needs of each child. The physical tools used at Egebakken fulfill simple tasks, and are robust, but also come with limitations. While they can be very flexible, they can require a sizable effort to work with. Some of these tools are suitable for having digital replacements, as proper software could lessen the previous issue. Software solutions do however already exist, but our customers find them hard to use and overly expensive. [5] These solutions also come with limited flexibility, hampering the work of the guardians, as they try to optimize the solution to best enhance the understanding and communication skills of each child. The guardians are also in charge of organizing the daily schedule of each child, with the child having some limited influence.
[6]

4 An institution for children and the workplace of our customer.
5 FiXme Note: Find reelle tal.
6 FiXme Note: Mangler TAIL

2.3 USABILITY CONSIDERATIONS

This section describes general usability concerns that are relevant for the product.

The GIRAF system is designed to be usable by both children and adults, making the age range of potential users very large, as it can be used by young children all the way up to elderly guardians. This is amplified by the fact that the children using the system have ASD, which can result in their mental capacity being below that associated with their physical age.
While children are not directly customers for the product this semester, it is expected that the product will be expanded to accomodate them later on. For this reason, considerations like "'The product needs to accommodate children who click madly around a screen as well as those who sit back and wait to be told what to do."' [4] and "'The results also provide further evidence that young children require interactions designed specifically for their developing motor skills."' [3, p. 8] are relevant for the project. Another aspect that is important is consistency, as children "'... are also better at recognition over recalling"' [4], but concistency generally increases usability, regardless of age [2, page 90]. This includes creating concistency with the real world, using familiar icons and behavior in the product [4].

2.4 PEOPLE

There are two diffrent groups of people which is intented to use the GIRAF system. The first is the guardians which should have their own mode because guardians can have more than one child and therefore need, in their everyday for helping the children, to have some kind of choosing mecanisme for what child to help. They should also have the posibility for seeing all information and data in the system. The second group is the children and they should have less information i.e. only have access to their own information and data. It was decided only to work with the guardians because it was the most comprehensive. But there will still be incorporated the usability found in Section 2.3. The part of the launcher for children is called child mode and the part for guardians got the name guardian mode.

2.5 PROTOTYPING

This section will explain how important prototyping and sketching was in the project and how it helped in the preanalysis.

Prototypes and sketches were used to quickly communicate and refine design ideas within the project group, and for gathering feed-

back from customers. Many sketches and prototypes were produced over the course of the project, to keep communication about design effective. This also worked well with the iterative work form of the project, and new prototypes and sketches were continuously built on top of existing ones.
Late prototypes were used to demonstrate the design of the product to the customers. These prototypes were implemented on paper, and showcased the different screens of the product while demonstrating the control flow of the system in specific use cases. This was important in evaluating the design, and gave reassurance about the quality of the design.
An example of a prototype can be seen in Figure 4

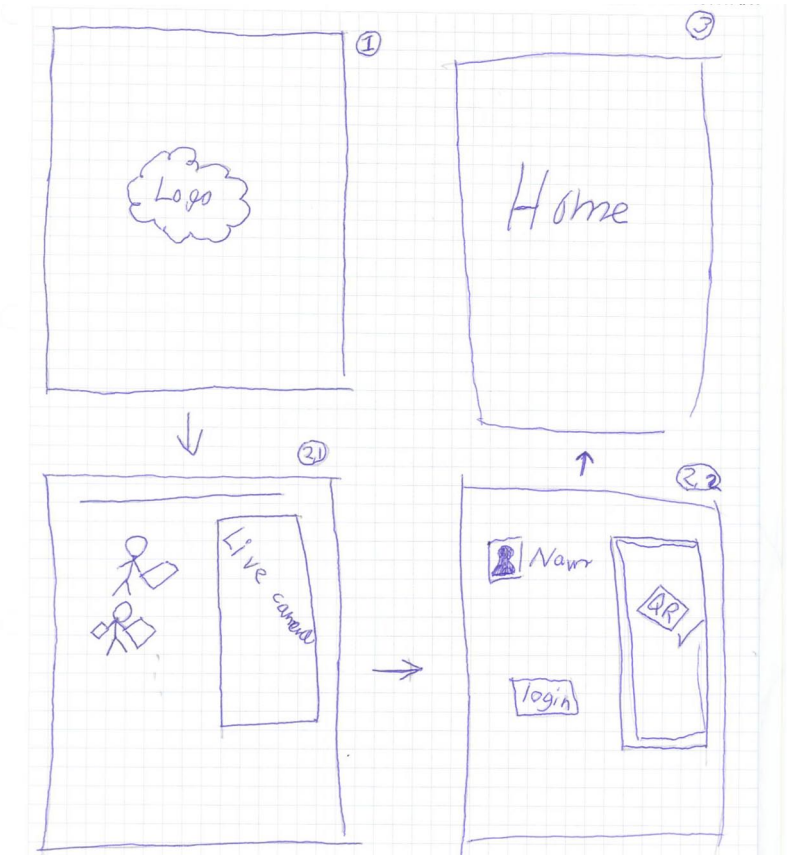


Figure 4: Activity overview page from one of the first prototypes.

7

7 FiXme Note: lav tail

3 ITERATIVE PROCESS

This chapter presents information about relevant events and areas of focus in each sprint. Durations of the sprints are presented in half days, as a lecture and associated exercises often last half a day. This makes the format convenient for describing the time available for the project in a given time interval.

3.1 SPRINT 1

Duration: Seven half days.

The focus of this sprint was on usability. This includes creating use cases, evaluating login methods, and designing visual elements of the system. By the end of the sprint, a usable design had been created. Some of the use cases created can be seen in Appendix B.
Managementwise, the meetings held in the multiproject group until this point had been marked by lengthy discussions and knowledge sharing, which made the meetings bloated and ineffecient. This improved by now, as most major decisions about the project had been made and knowledge sharing had been done for most relevant subjects. The meetings also improved as the organization of the multiproject group improved, and better restrictions were put in place to help alleviate unnecessary discussion.
There were issues with miscommunication however. For example, we presented ideas at a meeting in this sprint, but did not clearly communicate which parts we would aim to implement, and which we considered part of our vision for the system, but would not pursue.

3.2 SPRINT 2

Duration: Ten half days.

The focus of this sprint was to get implementation started. As a base design was already in place, ressources were mainly put into getting to know the system and the requirements for making a launcher. This led to features like app arrangement, app loading and a login system being the priority, with services for the other project groups not yet becoming important. At the end of the sprint, these features had been roughly implemented.
An issue came up at a meeting, as one of the project groups had trouble with their members showing up to the meetings on time. A quick

effort was made to solve the issue, and though some disagreement occurred, several possible resolutions were presented by the multi-project group.

3.3 SPRINT 3

Duration: Ten half days.

The work done in this sprint revolved around improving the features that were implemented in the previous sprint, while adding new features and services. Tasks worked on in this sprint include the ability to launch applications, providing profile information for applications, and refining login screen and design. This is also where integration with the Oasis library has started, increasing our reliance on the Oasis group.

3.4 SPRINT 4

Duration: Ten half days.

This sprint revolved around the presentation of the product, with features like the home bar recieving polish and the drawer mechanism being implemented. Ressources were also put into improving the integration with the Oasis library, with features like saving settings for each app.

3.5 SPRINT 5

Duration: Six half days.

This was the final sprint where implementation took place. Refining existing features was therefore the highest priority, and scope adjustments were made to cut features that would not be ready before the sprint was over. The sprint was ended with a refactoring period, where the entire code base was checked and complex parts were rewritten to increase readability.
The planning meeting for this sprint was also the first meeting where each project group made a formal presentation of their near final products.

4 BACKLOG

This chapter contains the entire backlog of the project, with the backlog entries divided into two groups, *implemented*, and *unimplemented*. The backlog entries have been prioritized based on their relevance for the design and the usability of the product.

4.1 IMPLEMENTED FEATURES

4.1.1 *Guardian mode*

Guardian mode allows guardian users to utilize the GIRAF system. This is currently the only mode in the system.

4.1.2 *Authentication*

Authentication provides data protection, and allows the launcher to know the identity of the current user. QR-codes are used to store the credentials of the users, and allows for authentication via the tablets camera, scanning a provided QR-code.

4.1.3 *Autologin*

The autologin feature enables guardians to start the launcher without authentication, if authentication was performed within the last eight hours.

4.1.4 *GUI components*

GUI components are shared using a public library, allowing GIRAF apps to use the same dialogs, buttons, tooltips, widgets etc.

4.1.4.1 *App grid*

Launchable GIRAF apps are launchable through a grid of apps, called "'app grid"'.

4.1.5 *Homebar and drawer*

Information is accessible through a bar – called "'homebar"' – as text, images and widgets. Additional information and functionality is hidden within the homebar, in a drawer, simply called "'drawer"'. The

homebar and the drawer is shown together with the app grid, described in Section 4.1.4.1.

4.1.6 *Logo screen*

While the launcher is initially loading, a splash screen containing the GIRAF is shown, showing the user that the system is loading.

4.1.7 *Profile select*

Upon launching an app from the app grid, child profiles are listed, such that the user is allowed to select one of them, prior to app execution.

4.2 UNIMPLEMENTED FEATURES

4.2.1 *Child Mode*

Another mode to implement in the launcher is called "'child mode"'. This mode is only for children and is minded on them using it alone without a guardian. In this mode, all settings are removed and everytime an app is clicked, it launches directly with the child's preferences instead of showing the profile selection screen.

4.2.2 *Custom Icons*

This feature can change the icon of an app in the app grid. This is important, as it allows children to use familiar icons, which was deemed important in Section 2.3. Recognition is important for children, and if they associate an app with another thing e.g. PARROT with a picture they know from their everyday, it should be possible for them to use the same in GIRAF.

4.2.3 *Add and Remove Apps*

This feature was meant to be integrated in the drawer **??**. It should allow guardians to customize what apps are accessible in the home screen, both for them or for a child (when Child Mode has been implemented).

4.2.4 *Home Screen Modes*

The home screen should work in both portrait and landscape mode.

4.2.5 *Lock screen*

The lock screen was a feature that WOMBAT needed for when a timer ran out and the tablet should not be usable any more. Currently, the authentication screen is available for this, but no functionality has been implemented to actually make it lock the device.

4.2.6 *Logo Screen Loading*

The logo screen is currently static, with no indication that the launcher is loading. This can make users unsure of whether they need to wait or do something else, something that might be fixed by implementing a loading animation on the logo screen.

Part III

PRODUCT

The *product* part explains the state of the product as of our delivery of this report without any of the features of in the *unimplemented*-section of backlog.

5 DESIGN

As part of a greater platform, the launcher is required to provide services to other components of the GIRAF system, and may also depend on others. The required services and dependencies are outlined below.

Figure 5 illustrates the GIRAF launcher component. The component provides one service and have one dependency. The services it provides is based on demands from the surrounding components of the GIRAF platform, which can be seen in Figure 1. The illustrated service provides a way for launched GIRAF apps to determine which guardian launched the app in question, with which child profile, together with color data of the specific app.

Knowing which child profile the app in question was launched with is required, as the GIRAF platform as of writing, is designed for *guardian mode* – described in Section 4.1.1 – instead of *child mode* – described in Section 4.2.1 – or a combination hereof.

The illustrated dependency represents the need for being able to read and update profile data. The service which fulfill this dependency is provided by the Oasis lib.

As seen on Figure 1, Oasis is available for the launcher to use. Oasis stores the modulated data, including the guardian and child profiles on the device in a local database.

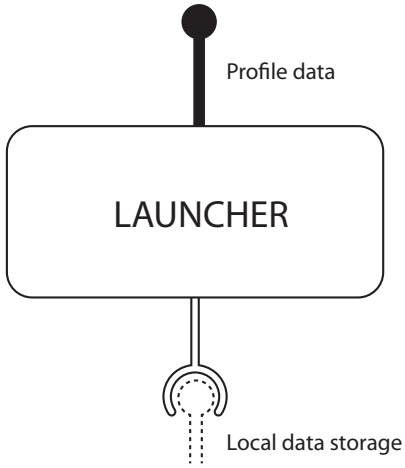


Figure 5: The GIRAF launcher component.

Being able to launch an GIRAF app as a specific guardian requires the user to interact such that the launcher knows which guardian the user represents.

Authentication was chosen, as each modulated child and guardian contains private data and therefore needs to be protected.

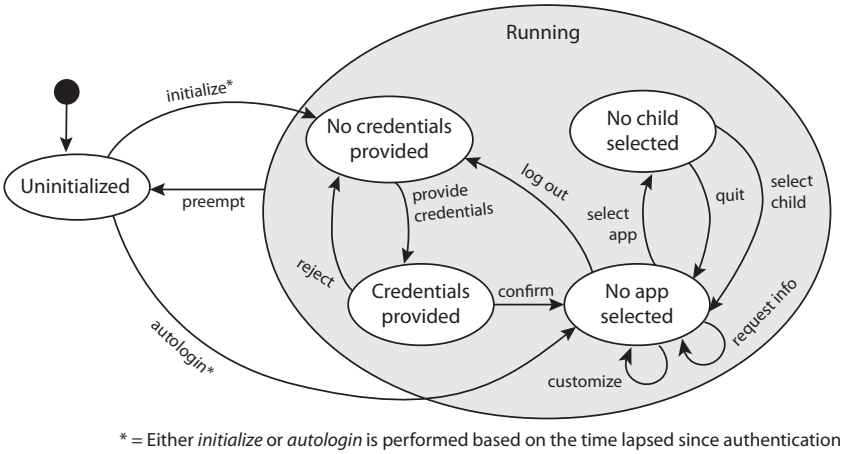


Figure 6: State diagram

Figure 6 shows all possible states and transitions, which the launcher can be in. The states of the state diagram represent four functionalities of the launcher:

1. Initialization
2. Authentication
3. App selection
4. Profile selection

These functionalities are shown in Figure 7, along with the states of which they represent, and the transitions between them.

The states, transitions and functionalities, are explained below in their appropriate section.

In order to highten usability which was requested during the interviews, found in Section 2.2.1, it is decided to standardize elements in the GUI. This standardization is called *design language*. Furthermore, it is decided that the concrete elements are to be shared such that other GIRAF components can utilize the standardization.

5.1 DESIGN LANGUAGE

5.1.1 *Status icons*

In the GIRAF system there are different status icons for showing the user what an action will do or is doing. All these icons can be seen in Figure 8. If an action accepts the leftmost icon will be used and if a window is loading the fourth leftmost icon will be shown maybe

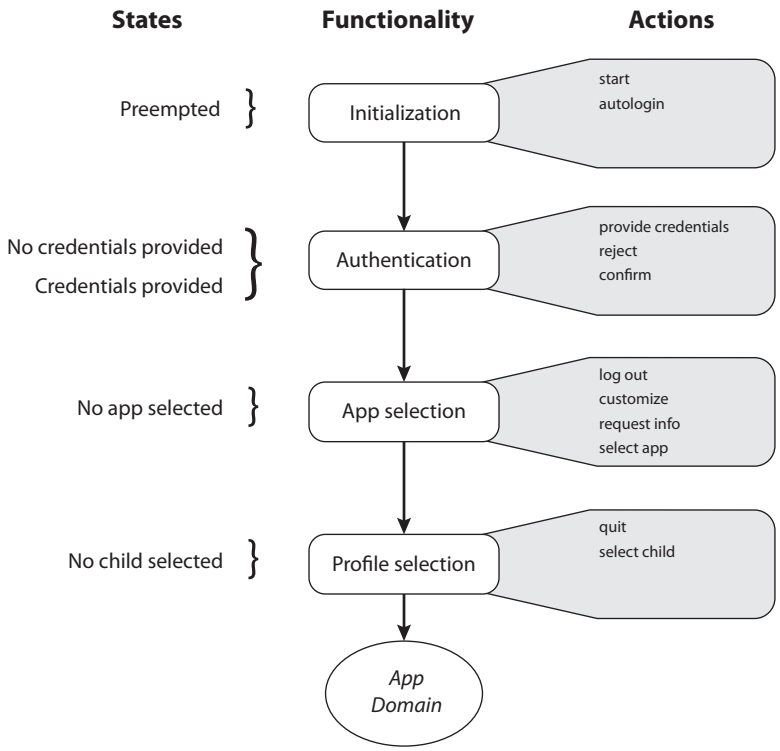


Figure 7: Functionality diagram

with an animation so that the user knows for sure that the window is not crashed.

Figure 8: Status icons in the GIRAF system

The icons in Figure 8 includes: Accept, reject, synchronizing, loading and not found, seen from the left.

5.1.2 *Colors*

Colors where chosen for the GIRAF system these can be seen in Figure 9. These colors is designed for use in two different cases. The first four yellow and brown colors is made for use together and the last four colors white and grey is made for use together. These color groups is chosen such that they have contrast and can make gradients.

Figure 9: The color theme of the GIRAF system

5.1.3 *Interactive elements*

In the GIRAF system all interactive elements should have round corners. This was decided to make the overall consistency and usability in the GIRAF system better this was done because of Section 2.3, where it states that recognition is better. If an element is not interactive they should have square corners. These guidelines also includes colors of buttons. All buttons should have the brown color on all sides and the middle should be yellow.

This section will use buttons as an example of an interactive element but the interactive elements in the GIRAF system also includes dragable objects and other touch objects. The example can be seen in Figure 10

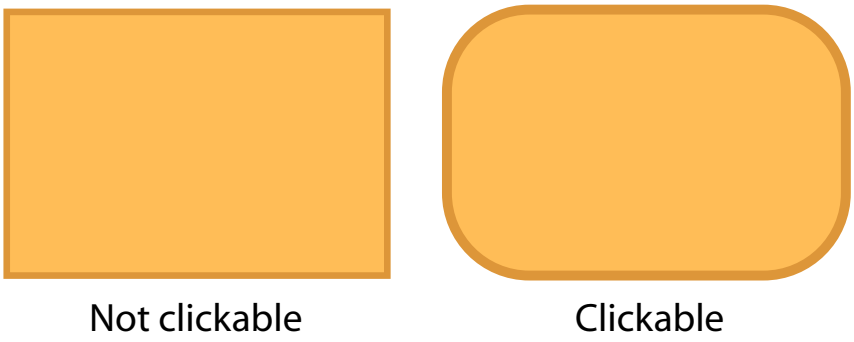


Figure 10: Interactive elements: Buttons illustrations

5.2 INITIALIZATION

It might be cumbersome to authenticate every time the GIRAF launcher is executed, if the user is switching between multiple launchers. Therefore, the launcher must be accessible in a sufficiently fast manner, such that the launcher is not perceived to be cumbersome.

In order to speed up the process of switching between different android launchers – should more than one be installed – we choose to allow users to access the launcher with the last logged in user, if authorization happened within the last eight hours.

One could argue that this feature – also described in Section 4.1.3 – introduces a security issue, as an attacker, if able to physically get

a locked device, running the GIRAF launcher in an authenticated session, could simply reboot the device in order to get full access.

We deem that this is not critical, since a simple workaround is to always logout whenever the device is placed in an location where unauthorized users might get a hold of it.

In Figure 6 the "Uninitialized"-state is the first state reached, when running the launcher for the first time. There are two transitions from the "Uninitialized"-state:

- Initialize
- Autologin

The *initialize* is taken, if one of the following conditions are met:

- Authentication has never been done before on the device
- Authentication was done more than eight hours ago

Taking the *initialize* transition brings the launcher to the "no credentials provided"-state.

The second transition – *autologin* – is taken in the case authentication was done less than eight hours ago, and brings the launcher to the "no app selected" state, such that the authentication process is omitted for the user, although the last authenticated user is still authenticated.

5.3 AUTHENTICATION

Authentication consists of two steps:

1. Validating
2. Confirmation

These steps are required to have in order to make authentication successful. The need for validation is to ensure privacy. Confirmation is a requirement as error prevention, incase of validation with wrong identity. Being able to launch a GIRAF app as a specific guardian requires the user to interact such that the launcher knows which guardian the user represents.

As stated in Section 5.3, privacy is required since: each modulated child and guardian contains private data and therefore needs to be protected. QR-codes was chosen as means of authentication, as they provide a level of security. An alternative to QR-codes could be a *username-password* method, where each user have their own username, with an private password. The launcher is developed towards being a tool usable by both guardians and children e.g. the "child mode"

feature, described in Section 4.2.1. A username-password combination requires the user to remember their credentials, whereas some children have problems with it.

> Some children can have problems remembering a username and password
>
> - Drazenko Banjak, educator at Egebakken.

QR-codes provides a physical way of storing the user credentials and allows for other users to take responsibility of the QR-code, such as a guardian carrying a QR-code of a child. They can be scanned by a built-in camera on tablets and can be printed using standard paper and printer equipment. They can be copied, by e.g. a copy machine, and therefore must be kept away from untrusted users, if they should not be used by people for which they were not intended. To sum up, QR-codes are chosen because of they improve usability, despite of their ability to be copied.
[1]

This leads to the functionality solution seen in Figure 11.

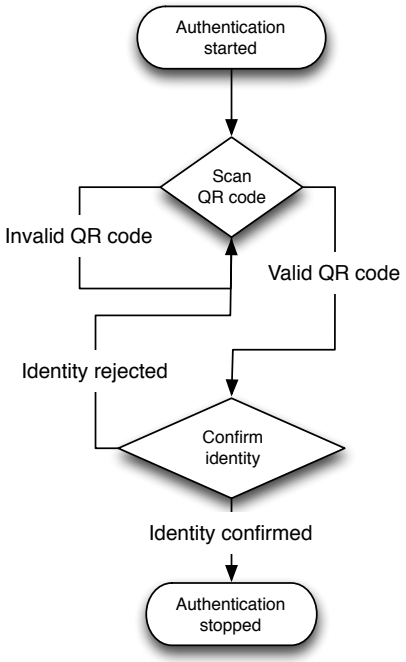


Figure 11: Flowchart of the authentication functionality

Upon scanning a QR-code, there is two possible outcomes: The QR-code is invalid, as the credentials are not recognized, or the QR-code contains credentials which are recognized. In case of the QR-code

1 FiXme Note: Ulrik, er dette i orden? Bullet points plox

being valid, the process then enters it second step. The second step consist of the user needs to confirm the identity, or reject if identity represented on the system does not match the user's identity.

5.3.0.1 *GUI*

The GUI components of the authentication design, can be seen in Figure 12. The camera border helps distinct the camera feed from the rest

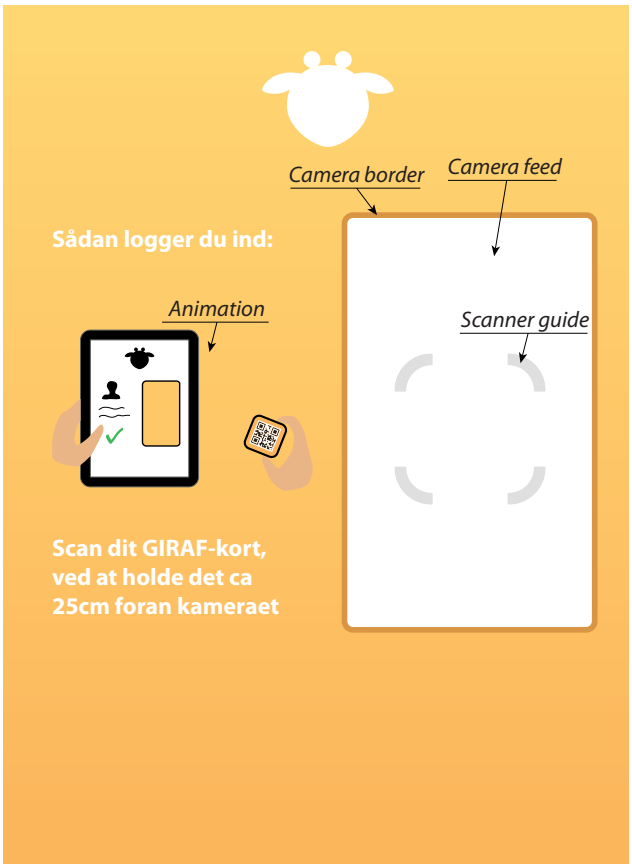


Figure 12: Graphical User Interface of the authentication functionality

of the layout, but it also provides feedback upon scanning by changing colors — this can be seen in Appendix F. To scan, the backfacing camera feed is shown inside of the camera border. A scanner guide is shown on top of the feed, to help the user aim during the scanning process and possibly enhance the feel of the design being a scanner[2]. On the left hand side an animation is shown, to guide the user of how he or she should interact with the system.

5.4 APP SELECTION

In Figure 6, the "no app selected"-state and the "no child selected"-state, and the actions inbetween, show the transitions taken, when

2 FiXme Note: OK sentence?

the user launches a GIRAF app.

Three pieces of information are needed in order to launch an app:

1. Current authenticated guardian
2. App to launch
3. Selected child to launch the app with

The first requirement is already given, since the launcher cannot be in any of the two states without the user already being authenticated. The second requirement is straightforward, as it is not possible to launch an app without knowing which app to launch. The first and third requirements are needed in order to fulfill the services which the launcher needs to have in order to be a functional part of the GIRAF platform, as shown in the component diagram, Figure 5.

Furthermore, additional pieces of information must be known to the user:

1. Date related data
2. Network status

As the GIRAF platform is thought to be the primary electronic tool of guardians, date related data is provided. Date related data is thought to be the day in characters the day in number, the month in characters and the week number. Later date related data can be a calendar app if such app gets into the GIRAF platform. Since the GIRAF platform uses both local and remote storage, there might be latency in synchronizing and keeping the data up-to-date in these storages, and therefore it is important for the user to know if both storages are synchronized, up-to-date or not.
[3]

Figure 13 shows the flowchart over the interactions the user can perform, and the actions the launcher takes upon the interactions, in order to fullfill the requirements listed in Section 5.4.

Figure 13 shows three branches of interactions and actions:

- Change app settings
- Launch app
- Request information

3 FiXme Note: check for system vs. platform

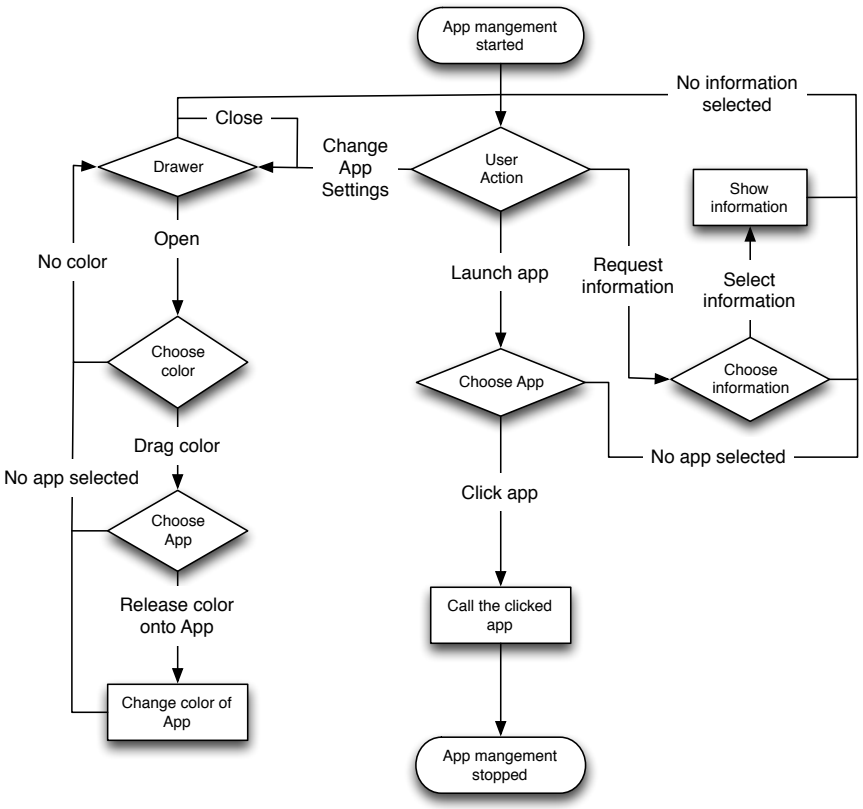


Figure 13: Flowchart over the app management functionality

5.4.0.2 *The drawer*

The drawer was designed to hide functionallity when not needed in a convenient place. As seen in Figure 13 everything that have to do with changing app settings is placed in the drawer which the user have to open to get to it. Another thing the drawer did was to make a place for the widgets. The handle of the drawer was designed to have the same color as the inside of the drawer because it was important to make consistensy in the way that the user should feel when they saw the color of the handle the thought of the functionality inside of the drawer. It was also rounded to tell the user that this element was interactive.

WIDGETS Widgets in the GIRAF system should help the user request information. As seen in Figure 13 the user can take the action to request information. If the user chooses an information the information will be selected and shown to the user. The reason it was designed this way is that if all the information should be around the drawer handle it would probatly confuse the user more than it would help them. All widgets is designed so they need to be unique, meaning that there can never be two of the same widgets. This was done for making widgets more consistent so the user always would get the

same information from the same widget. All widgets is also designed with round corners to seem interactive see Section 5.1.3. The logout widget should be in the bottom of the drawer because it should not confuse the user while they are looking for information.

COLOR PICKER All colors in the color picker is rounded because they are an interactive element. They also have white sides so they not fall into one with the background. The color picker consists of ten predefined colors. The reason for predefined colors in the color picker is to make contrast. If developers of GIRAF apps made their app icons mono-colored in white the contrast would always be sufficient. This should help that all users will always see the icon no matter what icon background color was chosen.

5.5 PROFILE SELECTION

As explained in Chapter 5, the purpose of the profile selection functionality is to inform the launcher of which child profile the user wishes to launch the previously selected app with.

The "no child selected"-state in Figure 6 shows the state of the launcher when the system is awaiting user input on which child profile to launch the previously selected app with. From the "no child selected"-state, there are two transition:

1. Select child
2. Quit

Regardless of the transition taken, the launcher will be brought back to the "no app selected"-state. The difference lies on the side effects of these transitions, and therefore the behavior they produce.

The side effect of the *select child*-transition, is the execution of the previously selected app, whereas the *quit*-transaction wipes the previously select app out of memory.

The flow chart in Figure 14 shows the steps involved in the profile selection process.

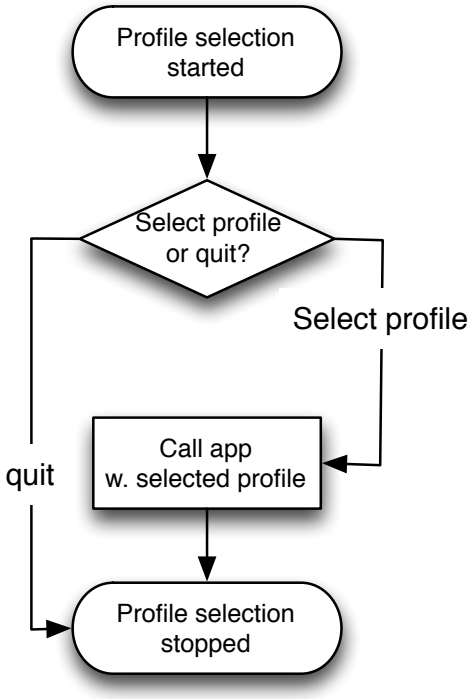


Figure 14: Flow chart of the profile selection process

Part IV

IMPLEMENTATION

4

4 FiXme Note: skriv ny head, skriv opsummerende hvad der konkret staar i implementation, nok bedst naar det er faerdigt

6 GUI COMPONENTS

GUI components is the name of the library which is the implementation of the concrete standardized components, as explained in Chapter 5.

The library consists of extensions to standard android classes:

- android.widget.Button
- android.widget.BaseAdapter
- android.app.Dialog
- android.widget.ListView
- android.widget.TextView
- android.widget.ImageView
- android.os.Handler

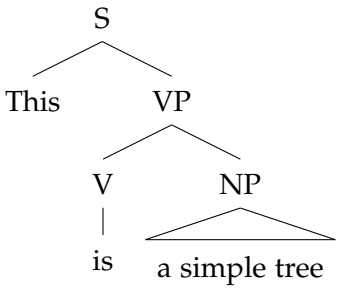
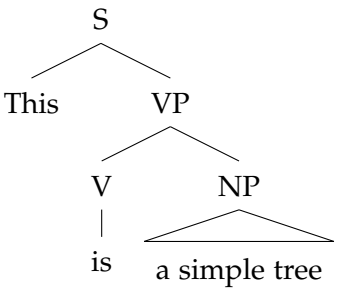


Figure 15: sample tree



6.1 ACTIVITY STRUCTURE

Figure 16 shows the activity structure of the GIRAF launcher. Every path with a `destroy()` box on it gets the activity starting the path destroyed. Everytime the user is outside the launcher and hits the home

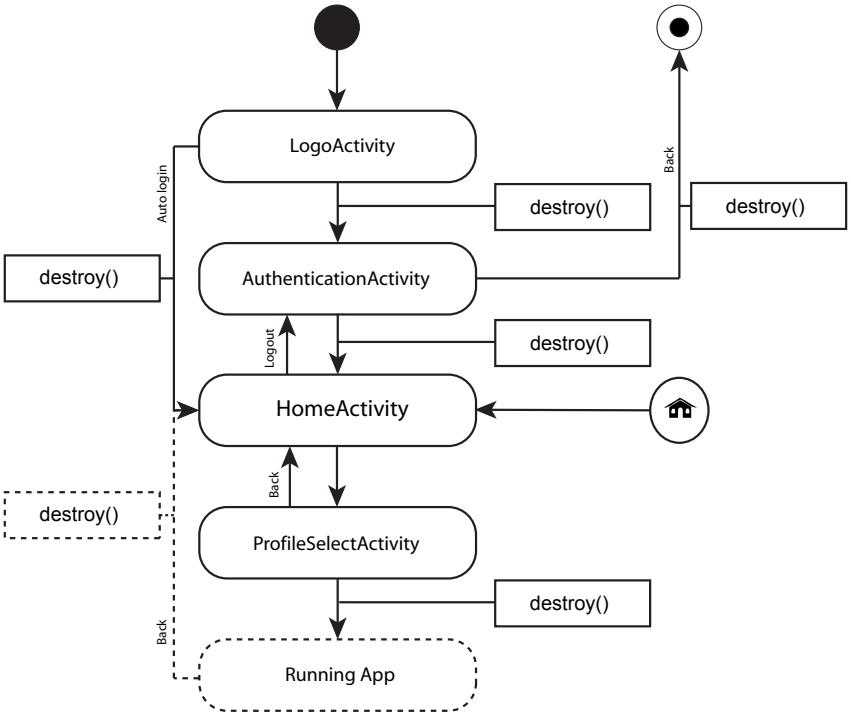


Figure 16: Activity diagram of the launcher

button they will end up in the *HomeActivity*, if the GIRAF launcher is set to the default launcher. The dotted line from Running App to *HomeAcitivity* symbolizes that the app itself can handle the back press but if there is not done something active to handle this action the default action is to return to the *HomeAcitivity* and destroy the app from the backstack. The backstack is a stack implemented by Android which holdes information about previous activities and the ongoing activity.

When the launcher is started the user is presented with the *LogoActivity* from where they will be redirected to the *AuthenticationActivity* to perform the login. In this process the *LogoActivity* gets destroyed from the backstack. It is possible for the user to easily quit in the *AuthenticationActivity* if they do not have their QR code at hand. If the user makes a valid login they will be sent to the *HomeActivity*. As the main activity it is possible to come from here and to *AuthenticationActivity* or *ProfileSelectActivity*. If the user clicks on an app they will be presented with the *ProfileSelectActivity* and have to choose a child before they can start the app.
[1]

1 FiXme Note: REFACTOR GIRAF COMPONENTS!

6.2 LOGO ACTIVITY

The *LogoActivity* is a splash screen, which shows the GIRAF logo while the system is loading.
This activity is justified by the possibility that the device running the launcher might spend more than a reasonable amount of time to load the *AuthenticationActivity*, and therefore might give the impression that the system has crashed or is not responding. The goal is to increase responsiveness.

However, *LogoActivity* does not indicate whether or not the activity itself has crashed. This is a feature which is on the backlog Section 4.2.6.

The logo activity implements auto login – explained in Section 4.1.3 – by using the *sessionExpired* method, found in *Tools* class, seen in Listing 2. The usage of the *sessionExpired* method can be seen in Listing 1

```
1 if (Tools.sessionExpired(mContext)) {
2     intent = new Intent(mContext, AuthenticationActivity.class);
3 } else {
4     intent = new Intent(mContext, HomeActivity.class);
5     SharedPreferences sharedPreferences = getSharedPreferences(Data.
          TIMERKEY, 0);
6     long guardianID = sharedPreferences.getLong(Data.GUARDIANID, -1);
7     intent.putExtra(Data.GUARDIANID, guardianID);
8 }
```

Listing 1: Snippet of LogoActivity.java

```
1 public static boolean sessionExpired(Context context) {
2     SharedPreferences sp = context.getSharedPreferences(Data.TIMERKEY, 0)
          ;
3     Long lastAuthTime = sp.getLong(Data.DATEKEY, 1);
4     Date d = new Date();
5
6     return d.getTime() > lastAuthTime + Data.TIME_TO_STAY_LOGGED_IN;
7 }
```

Listing 2: Snippet of Tools.java

6.3 AUTHENTICATION ACTIVITY

This activity implements the design described in Section 5.3 and is the first activity the user will face when using the GIRAF system. It is also the current lock screen which is used by WOMBAT Section 4.2.5.

This activity was implemented using ZXing [6], pronouced Zebra crossing, which is a libaray under Apache 2.0. This library is made for use with Android for scanning 1D and 2D QR codes. There have

been made changes to the original ZXing library to make it fit into the implementation of this activity.

There is implemented instructions and illustrations as mentioned in Section 5.3. There is also implemented a gesture, a 500ms vibration, for showing the user that the QR code was scanned correct because some of the users had trouble seeing the login button when it appeard.

```
1                // If the authentication activity was not launched by the
                      launcher...
2                if (!getIntent().hasCategory("dk.aau.cs.giraf.launcher.
                     GIRAF")) {
3                    Tools.attachLauncher(mContext); // should not be
                         called
4                    Tools.saveLogInData(mContext, mPreviousProfile.getId
                         ());
5                    startActivity(mHomeIntent);
6                } else {
7                    finish();
8                }
```

Listing 3: This is code

2

6.4 HOME ACTIVITY

This section will explain what the *Home Activity* provides of services to the user.

This activity is the center of the launcher and implements the design described in **??**. As the center it also have a lot of functionality some of it is the drawer which is explained in **??**. Some of the more basic functionality is to show the user which apps they can open. This is done by showing all apps which is installed on the device, is certified as an app in the database and the user has a relation to the apps in the database. Only apps which has all of these three properties will be shown to the user. If this number of apps exceeds nine and or if the drawer is extended so much that some apps will be place to the right of the screen the user will be able to scroll horizontally and will be guided by a scroll indication bar in the bottom of the screen.

6.4.1 *The drawer*

The drawer mentioned in Section 5.4.0.2 was also implemented.

The drawer is a place for functionality, this could be e.g. a way to put apps in the drawer if they are not being used by the user or if they need them they can drag them out of the drawer Section 4.2.3.

2 FiXme Note: Forklaring til lstlisting

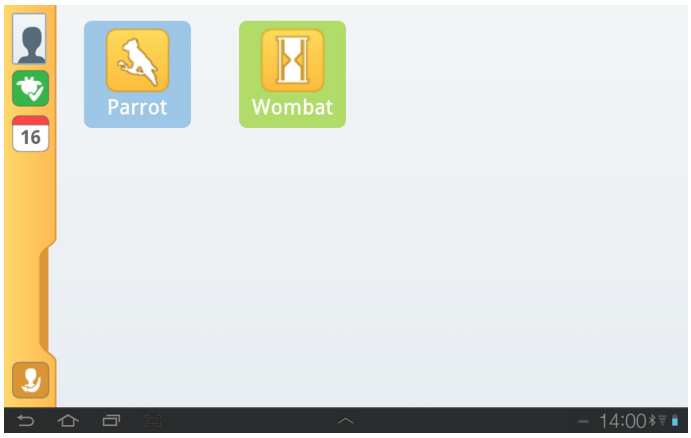


Figure 17: Shows the launcher with the drawer closed.

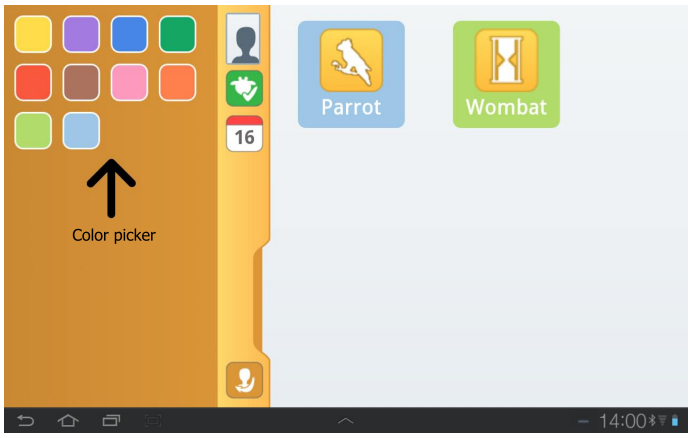


Figure 18: Shows the launcher with the drawer opened. The color picker can also be seen here.

6.4.2 *Color picker*

To show the functionality of the drawer a color picker for apps were implemented and hidden in the drawer. The color picker can be seen in Figure 18 as mentioned in Section 4.1.5 and explained in **??**. The *color picker* is a tool that was made to change colors on apps, this have been extended so far that if the logo of e.g. WOMBAT is choosen to be red then the background in WOMBAT will also be set to red when the application is loaded.

The color picker is implemented with a color board with ten predefined GIRAF colors. These colors can be assigned to any app that are in the launcher and there is no limit on how many times the user can use one color. This means that if the user want they can make every app e.g. red. The color picker is made with drag and drop functionality as described in Section 5.4.0.2. When the user have choosen a color and assigned it the app icon will change to this color and the color is saved in the local database so that if the user restarts the launcher the

apps will still have the colors the users have assigned for them. The color picker can be seen in Figure 18

6.4.3 *Widgets*

The widgets which is described in Section 5.4.0.2 and can be seen in Figure 18is between the app board and the drawer, this is called the *home bar*. Here it is also possible to see a picture of the user curretly logged in. The user can logout with the orange button in the button of the home bar. There are two widgets: One that shows the connectivity to the Savanah server. This widget has three states: Connected and updated, connected and synchronising and not connected. Connected and updated tells the user that all data is synchronized with the server and they are good to go. Connected and synchronising means that either the tablet needs data from the server or to push some changes to the server. Not connected tells the user that they are not connected to the server and therefore can not update or synchronize with the server. If the user clicks on the connectivity widget they will get a description to the currently showing status telling them what this icon means. The second widget is a calendar widget which tells the user what day it is in the month in a numeric format. If the user clicks this widget they will get a description with day, d. dd, month, week. This is done so the user easily can access and see what day it is.

Listing 4: This is code

6.5 PROFILE SELECT ACTIVITY

The *ProfileSelectActivity* activity implements the behavior described in item 5.5, and takes care of providing the profile service, described in **??**, to the launched GIRAF app.

As explained in **??**, the guardian should be able to launch apps with all profiles which is attached to the guardian. Listing 5 shows how all profiles are found, including those in attached departments and their subdepartments.

The profile service is provided with intents, as shown in Listing 6.

The launched GIRAF app can retrieve the chosen profile, by following our specification, as seen in [3].

[4]

3 FiXme Note: indsaet ref til vores speficikation af, hvordan man laeser hvilken profile der er valgt

4 FiXme Note: skriv tail, skriv opsummerende hvad der konkret staar i implementation, nok bedst naar det er faerdigt, og hvad vi fik ud af det.

```
1        ...
2        mChildren = new ArrayList<Profile>();
3        Profile guardianProfile = helper.profilesHelper.getProfileById(
             mGuardianID);
4        List<Profile> guardianChildren = helper.profilesHelper.
             getChildrenByGuardian(guardianProfile);
5        List<Department> guardianDepartments = helper.departmentsHelper.
             getDepartmentsByProfile(guardianProfile);
6        List<Profile> totalChildren = new ArrayList<Profile>();
7        totalChildren.addAll(guardianChildren);
8
9        for (Department department : guardianDepartments) {
10           List<Profile> childrenFromDepartments = helper.profilesHelper
                 .getChildrenByDepartmentAndSubDepartments(department);
11
12           totalChildren.addAll(childrenFromDepartments);
13       }
14
15       ...
```

Listing 5: Snippet of the *loadApplications* method

```
1        // When a child is selected, launch the app that was chosen with
             the correct data in the extras.
2        listOfChildren.setOnItemClickListener(new OnItemClickListener() {
3            public void onItemClick(AdapterView<?> parent, View view, int
                  position, long id) {
4                final long childID = ((Profile) parent.getAdapter().
                     getItem(position)).getId();
5
6                Intent intent = new Intent(Intent.ACTION_MAIN);
7                intent.addCategory(Intent.CATEGORY_LAUNCHER);
8                intent.setComponent(new ComponentName(mPackageName,
                     mActivityName));
9                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
10                       | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED);
11
12               intent.putExtra(Data.CHILDID, childID);
13               intent.putExtra(Data.GUARDIANID, mGuardianID);
14               intent.putExtra(Data.APP_COLOR, mAppColor);
15
16               startActivity(intent);
17           }
18       });
```

Listing 6: Snippet of the *loadApplications* method

7 TEST AND VERIFICATION

7.1 PRODUCT QUALITY

As seen in Section 2.2, the customers wished for the product to have high usability and flexibility. These are the only explicit user requirements that were procurred, but there are other quality aspects to take into account. As the project is meant to be continued by other students later, static software qualities like having maintainable and structured code are very important as well. And there are also plans to make the product usable by children later on, which puts heavy demands on dynamic software qualities such as reliability and consistency.

Because there is limited time for development however, not all relevant qualities can be prioritized equally, while still building a functioning product. To this end, the different quality aspects have been split into two pools; those that needed to be achieved within the development time, and those that did not. Each aspect is described, detailing why it is or is not important to achieve within the given time frame, and how we sought to achieve it.

7.1.1 *High Priority Quality Aspects*

These quality aspects are described in a prioritized list, with a descending priority.

7.1.1.1 *Usability*

Usability refers to the ease of use of the product, which was a key concern for the customer. Usability was thus promoted to the highest priority concern in development. To achieve high usability, all design work was centered around usability, seeking to simplify user interaction. Some relevant usability concerns for the product can be seen in Section 2.3.

7.1.1.2 *Consistency*

Concistency refers to consistent design and component use, which helps simplify the user experience, increasing usability[2, page 90]. This aspect was promoted to a high priority, as it has a impact on the usability of the system. To achieve consistency, the GIRAF GUI

Components library was created, not only to create consistency in the launcher, but to allow consistency throughout the system.

7.1.1.3 *Reliability*

Reliability refers to failure free operation, and is important in making the user feel confident about the system. If the user knows the system occasionally crashes, they may prefer an inferior, but reliable, tool to get the job done. This is important as children are especially sensitive to change, and so unreliable operation can be even more disruptive to them than to regular users. To achieve reliability, the system has been recurringly, albeit informally as well, tested and we have been pragmatic about fixing bugs.

7.1.1.4 *Structured Code*

Having structured code, refers to one's code being structured in an intuitive fashion that makes it easy to find desired parts of the code. Structured code helps make code more readable and understandable, which is crucial when handing the code over to a new team of programmers. To ensure that the code of the launcher is structured, the code has been continuously refactored, with deprecated code being removed, and functionality moved to the correct parts of the program. Classes have been created to accomodate functionality that is reused across the system, such that the functionality is available globally, to avoid code duplication.

7.1.1.5 *Maintainable Code*

Having maintainable code, refers to one's code being structured in a fashion that makes it easy to replace or add code to the program. Maintainable code is important when a new team is supposed to take over the project, and allowing them to easily make changes, makes it easier for them to be effective. To make sure the code of the launcher is maintainable, the refactoring process of the project has, among other things, focused on reducing functionality into smaller functions. This makes the code more modular, allowing components in the code to more easily be replaced, while also reducing code duplication.

7.1.1.6 *Code Documentation*

Code documentation refers to having documented code, for example in the form of comments in the code, or models or charts describing the behavior of the code. This is important for helping a new team understand the code, and help them be efficient in their work when they take over the project. To make sure the code documentation was adequate, the refactoring process has also dealt with commenting code, and breaking the code into more understandable pieces.

7.1.2 *Low Priority Quality Aspects*

These quality aspects have not been prioritized relative to each other, and are not described in any particular order.

7.1.2.1 *Testable Code*

Having testable code refers to tests being easy to set up and run for the code. This was not deemed important enough for the product, as much of the focus of this product is on the GUI, rather than functionality. As GUI creation is a key component of Android, we decided it was not necessary to make the GUI code testable, as the system would already have been tested extensively, formally or informally, at this point in time. Having informal testing of the GUI through use of the product was deemed necessary.

7.1.2.2 *Completeness*

Completeness refers to how many features made it into the product, compared to those specified in the product specification. Due to the short development time available, and the inexperience of the team with Android, it was decided early on to work in a way that would let features be implemented as time permitted, rather than creating a specification that had to be implemented in the time available.

7.1.2.3 *Performance*

Performance refers to the program responding to user input in a timely manner. While performance was deemed important due its impact on the user experience, it was found that the launcher did not tax the system in any meaningful way, and optimizing for performance was therefore deemed unnecessary.

7.1.2.4 *Correctness*

The correctness aspect refers to comparing the product with the requirement specification and determining if the product meets the specification. This was considered a low priority, as the project does not include focus on creating a comprehensive requirement specification and developing with that in mind, but rather on developing features as time permits with the customer requirements in mind.

7.1.2.5 *Flexibility*

While flexibility is not considered an aspect of software quality, it was requested by the customer. It was decided to focus on the other customer requirement, usability, to create a solid foundation for the product, and with the short development time, having flexible functionality was deemed unnecessary for the launcher in this semester. It

was important to focus only on either flexbility or usability, as adding features to a product (as flexibility does) can make the product more complex and less user friendly.

7.2 CODE TESTING

This section presents the testing that had been planned for the product.

Though having testable code has not been a priority, see Section 7.1 for more information, preparations were made in case there was time for doing dynamic testing. These preparations revolved around dynamic white-box testing, and includes some test cases for many different functions in the code that would allow for some unit testing to take place. These test cases can be seen in Appendix E. A few test cases were run as a proof of concept, but these were simple cases, that proved we did not have the time to run more extensive testing.
Had there been time to run these unit tests, the next step would have been to run mutation analysis on them to see if the existing cases exercised the code well enough, or more cases would have to be added.

But while there has not been much time for dynamic testing, static testing has been employed vigorously, though not in a formal fashion. Static black-box testing has been employed when the design of the product has been discussed, and the design has gone through many iterations and refinements because of it. Discussions about the design have been common, though more so early in the project, and iteration has been important for all parts of the product, as is also supported through the agile working process employed. See Section 2.1.2 for more details on the working process.

Static white-box testing has been used as a part of the pair programming and the refactoring employed in the project. Both of these have code reviewing as an essential component. Pair programming is about creating higher quality code by having real time reviews of the code as it is written, and refactoring is about reviewing code and improving it when bugs are found, the readability of the code is too low, or the complexity of the code is too high.

7.3 USABILITY TESTING

This section describes the usability testing that has been performed, and the results obtained.

7.3.1 *Test Circumstances/Something*

[1] One usability test was performed on the product, see Section 1.8 for more details on the test set up. This was deemed adequate, as the customers had provided continuous positive feedback on the product. As such, the test was also performed late in the project as a means of verifying what level of usability had been achieved, rather than using the test to find problems that would then be fixed as the project came underway.
The test was set up to test for both discoverability and usability. Discoverability was tested as the customers had mentioned the high costs associated with certified learning courses during interviews (see Section 2.2.1) as a disadvantage of current systems, and making functionality easy to discover would help reduce the need for such courses.

7.3.2 *Test Results*

The issues found in the launcher have been classified by the time it took the user to complete a given task, and can be seen in Table 1. Cosmetic issues only held the user back very briefly, serious issues had the user confused and trying to complete the task for a few minutes before succeeding, and critical issues were the ones where the user was stuck.

Table 1: Results from the usability test

ID	Criticalness	Issue Description
#001	Cosmetic	Log out button not obvious.
#002	Cosmetic	QR code obscured by fingers.
#003	Cosmetic	Wrong camera usage when logging in.
#004	Serious	Unresponsive controls after opening an app with a profile.
#005	Serious	Calendar widget not intuitive.
#006	Serious	Connectivity widget not understood.
#007	Critical	Drawer functionality not discoverable.

7.3.2.1 *Dissection of results*

This section details the different issues presented in Table 1.

In issue #001, users were asked to log out, and a common pattern was for users to press their profile pictures to find log out information, rather than pressing the log out button. After the profile picture

1 FiXme Note: Lav overskrift

was pressed and nothing happened, users tried the log out button and succeeded. This issue could be alleviated by making it clearer what the log out button does.

Issue #002 was a matter of users covering a part of the QR code with their fingers, not realizing this was an issue at first. This is an issue we deem to be best solvable through user education.

With issue #003, it was not clear to users how to use the QR scanner, as there was some confusion about whether the camera used for scanning was the frontal facing or backwards facing camera. This could be improved through communication on the authentication screen, better guiding the user on how to use the scanner.

Whether issue #004 belongs in the launcher is not clear, as the issue is tied to the system taking a long time to load apps when they are opened. The issue could therefore belong to the individual apps, but it would also be possible to create a universal loading screen in the launcher, to signify to the user that the system is working. Using faster hardware would also improve load times, though this is not considered a sustainable solution, as slower loading apps can always be created.

Issue #005 relates to the calendar widget, as in the current system, there is a conflict between the calendar widget in the GIRAF launcher and the native system clock, which provides similar functionality. This confused the test subjects, as most tried to use the system clock to complete the task regarding the calendar widget using the system clock. The issue could be alleviated by removing either of the two conflicting components, or making it clearer to the user what the calendar widget does. There is however reasons for keeping the calendar widget, see more details in Section 5.4.0.2

In issue #006, the users had trouble finding the connectivity widget, but also understanding just what it did. Some tried to look under the native system clock, as it also holds information regarding Wi-Fi connectivity. We believe this issue stems from a lack of knowledge in users, as they did not know what they were checking for, when asked to check the connection status with the server. User education could therefore make an important difference for the usability of this component, but clearer visual clues about its purpose would probably also be helpful.

The final issue, issue #007, refers to the fact that several users needed help to discover the drawer. This is a critical issue assuming that users do not hold prior knowledge of the drawer, and could

be solved by creating more cues to its existence. For example, if the home bar is pressed, make the drawer bounce out slightly, signifying that there is more content behind the home bar.

7.3.2.2 *Feedback*

Aside from the actual test, users also answered a questionnaire about the test after they had completed it. In this questionnaire, the test subjects were asked to rate the ease of use of the launcher, on a scale of 1-5, where lower is easier. The average rating of the launcher was 3, which translates to medium difficulty. One user rated the ease of use 2, and another rated it 4, though the subject who rated it 4 had not used a tablet before.

During debriefing, many of the test subjects noted that they were not entirely familiar with standard touch interactions, such as long-clicking, even though they had used a tablet before. They noted that this impacted their performance with the product, as the test also became a learning experience for them. The test subjects also noted that their performance was hampered by a lack of prior knowledge of the product. For example, though the drawer in the GIRAF launcher was hard to find for the test subjects, see issue #007 in Table 1, the users noted that the drawer itself was easy to use once they knew of its existence.

7.4 KNOWN BUGS

This section describes all bugs confirmed to exist in the product.

7.4.1 *Apps not updated*

This bug is that the home screen is not automatically updated when a new app is added to a user. To make the app appear, the launcher must either be restarted or the user must log in again. This bug is not critical at this point, as it is not currently explicitly supported that a user can add new apps, though the bug can manifest itself in the following scenario.
For the scenario, the following conditions are true.

- User X is logged in on device Y.
- User X has app Z attached in the database.
- App Z is not installed on device Y.

The bug occurs when app Z is then installed on device Y, as app Z will not show in the home screen until steps have been taken to show it, as described before.

7.4.2 *Incorrect color data sent to apps*

The launcher will currently send data about the color chosen for the app in the launcher to each app. This data is however not changed immediatly when the user chooses a new color for the app, as the data is only brought up to date either when the launcher is restarted or a user logs in. This is because the color data is only inserted into the database when the user changes a color, whereas the data held in memory by the launcher is not. This bug is important to fix, as it creates inconcistency for the user, but the bug was not discovered until after implementation had finished.

7.4.3 *Camera feed is too big*

Authentication takes place on a single screen, where the camera feed used for QR scanning is an important element. The camera feed is however a bit too big to fit in properly with the rest of the elements on the screen. As such, there is currently some overlap with the camera feed and the animation on the left side of the screen, and the camera feed is also closer to edge of the screen than the design calls for.
This bug is not critical, but could potentially confuse the users of the product. Correcting it is however not trivial, which is why it has not been fixed.

Part V

EPILOGUE

8 DISCUSSION

This section will discuss the decisions made throughout the project, and their relevance and ramifications.

[1] The management of the multi-project was conducted by the project groups, and no single project manager was chosen. Having no leader role in the project meant that excessive time was spent making decisions, as a majority of the multiproject group had to be convinced of the quality of a proposal before it could be accepted. In this scenario, a leader might have been able to cut down the time spent discussing, and force through a decision, when excessive discussion hampers decision making. A possible drawback of having a leader, is if he or she overrules too many discussions and diminishes the involvement of the multiproject group as whole.

In this semester, the project groups have a hard limit member count of four [5], instead of the previous limit of six. The lowered amount of group members demands more effort from each member. Absence also affects the group in a greater way, as group size lowers.

The backlog is a crucial element in agile development and it is used to encourage developers and help communicating across the multi-project. Backlogs were created in each project group, but there were not created a global one, for the multiproject group. As seen in Section 3.1, there were confusion with what tasks each project group were handling. An effort to develop a global backlog might have helped unite and create a clearer target for the multiproject group, and helped promote cooperation. For example, Savannah might have prioritized synchronization higher. This might have eased the process of testing, as each group would not have to create their own dummy data for their local Oasis database.

Using XP, a goal is to motivate the team members by letting them choose their own tasks, so they can pick what they feel motivated for. This can however create some difficulties, when no one in the team feels compelled to do a certain task. Working in pairs to solve non-programming tasks might have solved the issue, as each member in the pair help motivate each other.

Communication the process of each project group were done by using burn down charts, amongst other things. The burn down charts were hard to get accurate readings from, as each project group had

1 FiXme Note: Check for multi-project vs. multi project

their own measurement of time. It might not be needed to have burn down charts for further development.

The essence of working iterative is the fact your previous work might change. To accommodate this we decided to write our report after implementation was done. This reduced overhead of keeping the link between the implementation and the report up-to-date. However, writing after the implementation creates a sort of traditional working process. Traditional processes have issues if something goes out of schedule, and this issue can also be problematic in this case.

8.1 REMARKS AND FUTURE WORK

This section is made for the group that comes after us and gets to develop on the launcher. Here it will be possible to find information on what future work we would have done but also which information and experiences we made working in an multi-project group and using agile development.

8.1.1 *Iterative process*

When working in an iterative process it is important too meet often in the start of the project periode and later on in the project periode it is important not too have too many meetings. The reason for this is that in the early stages it is really important that all groups have the same vision and know what all the others are doing so dependencies etc. can be figured out and backlogs can be made! Later on it is important that the groups have time to work with their own project, but they should have open door policy, so they get the job done and do not waste their time with meetings which does benefit anything. We found out that in the startup and onto the rapport writing it was great with sprint with a length of a week or two where often there would be sprint meetings every monday and friday. We also figured that having people incharge was very important because then the job gets done and everyone knows who will take care of a subject and how to talk to if they have problems with this subject. This could be SVN, commen rapport, test manager, specification manager, design manager etc.

8.1.2 *Development*

The forward development of the launcher is very important because it provides functionality both to the users of the GIRAF system but also all other apps in GIRAF. The first steop is to fix all bugs found in Section 7.4 and in the usability test results Table 1 in test Section 7.3. Maybe even run an early test giving the guardian the tables in Ege-

bakken to try for a week and see if this will reveal even more bugs or bad design decisions.

8.1.2.1 *Modes*

The two modes represented in the GIRAF system is the guardian- and child mode. As seen in Chapter 4 only the guardian mode was implemented. Therefore it is possible that the next step the launcher should take is to include child mode so children can use the tablet alone without a guardian.

8.1.2.2 *GIRAF GUI components*

The GIRAFGUI library was implemented and works great, maybe it needs to be enforce to some of the other apps in the system, and should definatly still be used to place all GUI elements which every group can benefit from. One step to take in the future is to include even more elements in this library so, in the end, it will include all kinds of graphics from buttons to color pickers and replace the Android standart ones.

9 CONCLUSION

In this project, several groups have shared a problem statement, and worked in a multi-project, where they each built their own part of a system that would solve the issue presented. The problem statement for the project was:

> How can we ease the daily life for children with ASD and their guardians, while complying with the study regulation?

We have designed and implemented a launcher component for the multi-project's Android solution called *GIRAF*. The launcher component deals with: authenticating users, launching apps and allows the user to read and change platform settings.

We began by gaining understanding of the customer domain. This gave us insight into how the software quality aspects should be prioritized, and their importance to the customers. This lead to envisioning through the creation of prototypes, and sketches, prior to the actual design of the launcher. Usability was found as one of the key aspects of the project during this period. Based on this analysis work, development began with the first sprints focused on the design of the user interface, to ensure that usability was prioritized in the product. The later sprints were focused on implementing core features, and integrating the previously created design.

We implemented a library containing UI components for the other project groups to use, in order to enhance usability by creating consistency. In the implementation, we also focused on creating maintainable and readable code, to ensure that the project could be used by other developers.

To verify the quality of the product, we conducted a usability test. The test subjects consisted of the customers of the multi-project. The test highlighted some issues in the launcher, which could be corrected by the next group of developers.

Part VI

APPENDIX

A

IMPLEMENTATION

A.1 LOGO ACTIVITY

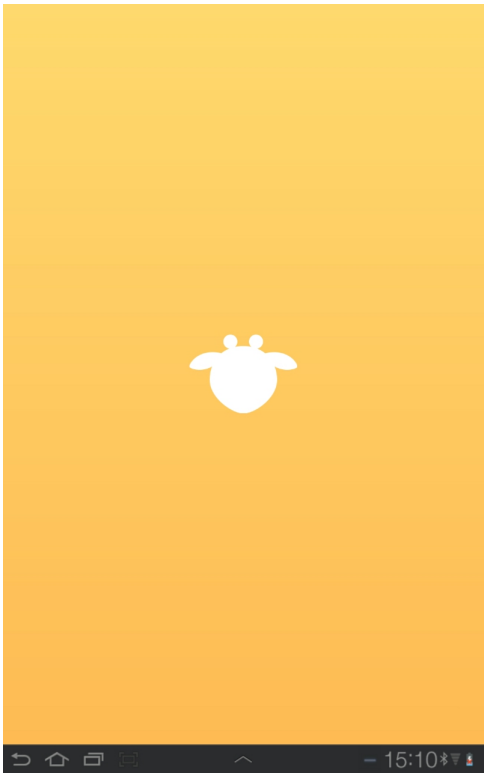


Figure 19: Portrait Logo Activity only shown when user do not have a valid session.

A.2 PROFILE SELECT ACTIVITY

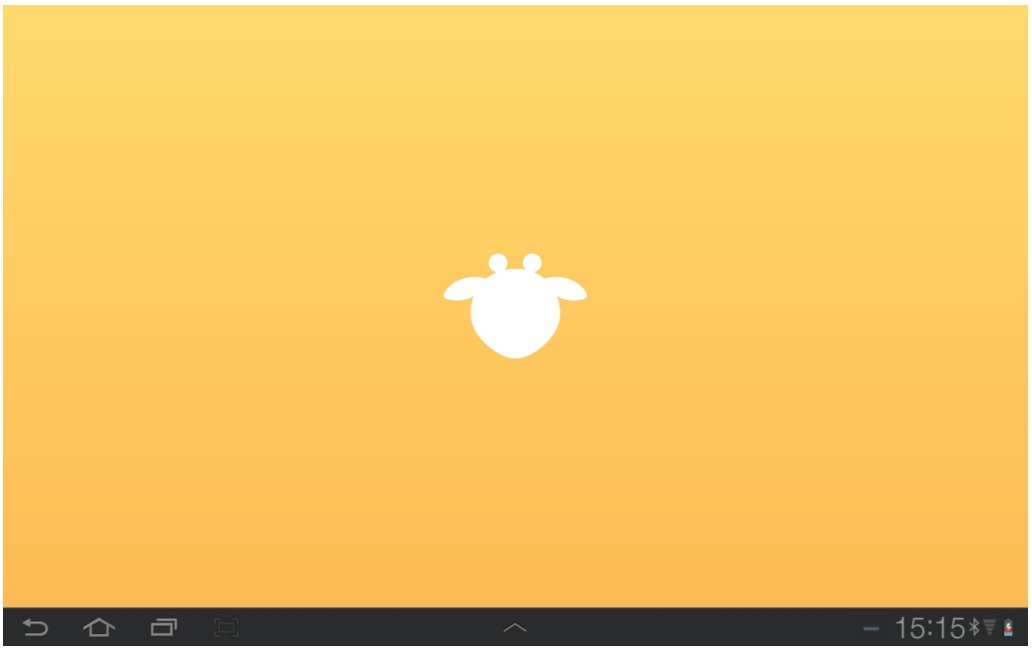


Figure 20: Landscape Logo Activity only shown when user has an valid session in and start the application again.

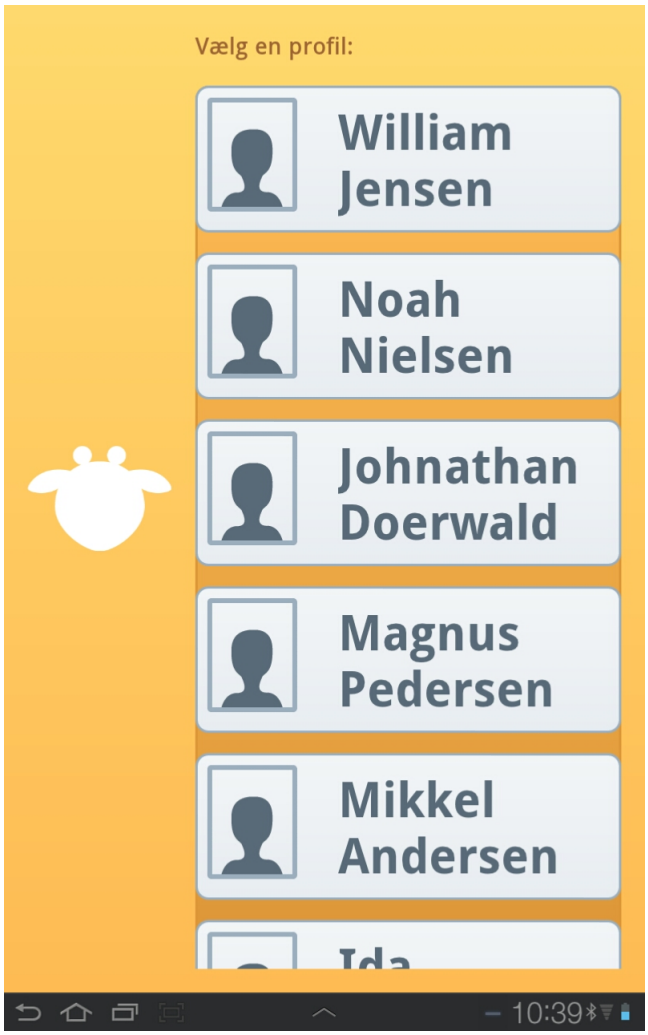


Figure 21: Portrait profile select activity screenshot

B

USE CASES

B.1 NEW GUARDIAN LOG IN

Karen is new at her workplace where she has had her GIRAF profile created already and is now trying to use a GIRAF tablet for the first time. She has recieved her authentication QR code, and after she has booted the tablet up, it prompts her to scan her QR code. She presses accept and is sent to the QR code app, where she scans her code and is sent back to the GIRAF system after it has scanned her code. She is now in guardian mode and can access the children she has been authorized for.

B.2 CONFIGURING AN APP FOR A CHILD

The GIRAF systems where Karen works have recently received a new app that she wants to use with one of the kids. To prepare, she has the tablet she is logged in on and wants to open the settings app to personalize the app ahead. When she opens the app, she is given a list of profiles where she chooses the profile for the child.
The settings app then becomes active, and displays a list of all apps that are available to the child, and the bottom the app has a button for adding more apps to the list. She presses that button and receives a list of all available apps on the system. She then finds the new app, selects it and confirms that she wishes to make this app available to this child. She is then sent back into the settings app with the list of apps where she can now choose the new app. She does so, and the settings app opens a page next to the list where she is asked to change settings in "'text"' mode or in snapshot mode.
She first chooses "'text"' mode, and receives a list of the settings she can change in the app. She wants to change the background color of the app, so she finds the setting labelled "'Baggrund"', presses it and is given the option of choosing a picture or a color as background. She selects color and is given the option of choosing the color from a color wheel. She marks the desired color and presses "'OK"' (which means her new settings have been saved).
She wants to change other aspects of the app as well, but prefers doing so in snapshot mode, so in the list of apps available to the child, she presses the app again, and now selects snapshot mode when prompted. An interactive snapshot of the given app is now opened next to the list of apps, and pressing elements of the app allows her to configure them. She presses an element she would like to resize

and does a pinching motion to make the element smaller. She then presses the "'OK"' button (which means her settings have been saved) at the bottom of the screen and is returned to the interactive snapshot's starting state. She is finished editting the settings of the app and so she presses the Home button and is returned to her launcher.

B.3 LAUNCHING AN APP FOR A CHILD IN GUARDIAN MODE

Karen is trying to work with a child and wants to open an app with that child's profile. In guardian mode, she presses the icon for the given app and receives a list of profiles to launch the app with. The child's profile is not directly visible, so she scrolls down the list until she finds the correct profile. She selects the profile and the app then launches with the settings specified for that profile.

B.4 LETTING A CHILD USE AN APP FOR A LIMITED TIME

Karen wants to let a child use an app in the child's break. With the system in guardian mode, she presses the icon for the app. A list of profiles then pops up, and by swyping in from the left, a list of apps that can interact with the app becomes visible. With a time app installed, it is visible on the list and allows Karen to choose a period of time where the app it is used with will become unavailable after this time runs out.

C

QR CODES

C.1 QR CODE TEST

CHARS	10	30	50	100	150	200
	3035	2339	3719	2713	3151	3189
	4209	4326	3145	2939	3227	3565
	3958	5387	3022	4175	2940	3030
	3569	3206	2751	3901	2737	3334
	2886	2880	2768	3298	3278	4930
	3958	3182	2820	2338	3917	3766
	2514	2645	3490	2140	2965	3889
	5588	3087	3119	2214	2104	3306
	2940	3869	3669	3443	2420	2668
	2880	4089	2663	2364	2169	2544
Average (ms)	3553.7	3501	3116.6	2952.5	2890.8	3422.1

Figure 22: The results from the QR code test, all times are in ms.

D

USABILITY TEST

D.1 TEST INVITATION

Kære ekspert

Vi vil gerne inviterer dig til at deltage i den første brugervenligheds test af GIRAF, en Android applikation bygget til børn med autisme. Formålet med denne test vil være at undersøge hvor brugervenlig applikationen er og hvor nemt eller svært det er at bruge den. Derfor er det helt fint hvis du aldrig har set eller hørt om denne applikation før nu, da vi gerne vil observerer, hvordan første gangs brugere så vel som brugere med kendskab til applikationen, har det med denne applikationen.

Bemærk venligst at vi er ikke tester din kendskab til applikationen eller evner med en tablet, men derimod om GIRAF applikationen er nem at bruge, vi har kun interesse i at kende til de svagheder der ville være i applikationen. Dette betyder også at du ikke kan give nogle forkerte svar, da du er eksperten.

Derfor vil vi gerne inviterer dig ud i vores brugervenligheds laboratorie, hvor vi kan studere din brug af applikationen. Under brugervenligheds testen vil du blive givet en række opgaver, som skal udføres. Yderligere vil du blive bedt om at tænke højt og fortælle alle tanker, indtryk og valg du tager ved brug af applikationen under testen. Under testen af applikationen vil der blive optaget både video og lyd, til at studere testen senere.

Dagen kommer til at bestå af:

Vi vil meget gerne høre fra dig hvis du har lyst og tid til at deltage i denne brugervenligheds test, den 22/5 - 2012, på Aalborg Universitet.
For at vide hvornår på dagen du kan komme vil vi gerne, at du går ind på denne side (http://www.doodle.com/d2h6swgbtsdf6z2b) skriver dit navn og vælger det tidspunkt på dagen du helst vil komme, dette er svar nok for at vi ved du gerne vil komme.

Kommentarer og spørgsmål kan sendes retur til den mail invitationen kom fra.

På forhånd tak,
Android projektet
Software 6. semester
Aalborg Universitet
Selma Lagerlöfs vej 300, 9220 Aalborg

AALBORG UNIVERSITY

Figure 23: Invitation asking for customer participation in usability testing.

D.2 TEST BRIEFING

Briefing

Goddag og velkommen til denne brugervenlighedsundersøgelse.

Vi vil gerne starte med at takke dig for, at du vil hjælpe os med at gennemføre denne brugervenlighedsundersøgelse. Vi læser op fra dette dokument for at sikre os, at alle personer som deltager i vores studie for samme introduktion. Hvis du har spørgsmål undervejs, er du naturligvis meget velkommen til at stille disse spørgsmål.

Vi har i dette semester bygget et system til Android til at hjælpe børn med autisme og deres pædagoger og forældre, og det er nu nået til et stadie hvor vi gerne vil teste systemet. Denne test handler udelukkende om at finde problemer og mangler i systemet, og ikke om at teste jeres viden af systemet, så alle tanker I må have om produktet vil vi meget gerne høre.

Før vi starter første del af testen, vil jeg bede dig om at underskrive denne samtykkeerklæring for at sikre, at du er indforstået med rammerne for studiet. Derudover skal du også svare på et demografisk spørgeskema inden testen går i gang.

Testen består af fire dele:

- Test af applikationer (20 min)
- De-briefing og spørgeskema (5 min)
- Test af Administrations applikation og web applikation (20 min)
- De-briefing og spørgeskema (5 min)

Undervejs vil der være en pause.

I de to tests vil du blive stillet en række opgaver som du skal løse. Læs opgaveformuleringen grundigt og fortæl så test hjælperen hvad du mener opgaven går ud på. Derefter skal du forsøge at løse opgaven så godt som muligt. Opgaverne skal løses i den rækkefølge de står således at du starter med opgave 1 og arbejder dig ned af.

Det er meningen at du skal tænke højt mens du løser opgaverne. Dvs. at du siger hvad du har tænkt dig at gøre for at løse opgaven, hvilke ting du synes virker uklare eller komplicerede og hvordan du tror systemet virker. For eksempel vil det være godt hvis du nævner hvad du forventer en knap gør inden du trykker på den.

Når testen er færdig vil der være nogle afsluttende spørgsmål som du skal besvare omkring hvordan du synes testen er forløbet og hvad din opfattelse af systemet er.

Figure 24: Briefing given to test subjects prior to testing.

D.3 DEMOGRAPHIC QUESTIONNAIRE

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

__ Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
__ Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

__ Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

__ Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

__ Ja __ Nej

Figure 25: Questionnaire filled out by the test subjects before the test. Afterwards, each subject was asked to rate the difficulty of each application on a five scale rating.

E

TEST CASES

Overall requirements:
Launcher is installed on a tablet running Android 3.2.x

The tests are grouped based on the class they exist in, with the identifier of each test case being the name of the function tested and the number of the test for that function.

For the second part of the tests of Tools-functions, the following apps must be present in the system as noted for each of them.

WOMBAT (is a GIRAF app):

- is in the database.
- is installed on the device.
- is attached to the current user.

PARROT (GIRAF):

- is in the database.
- is installed on the device.
- is **not** attached to the current user.

ViewDB (GIRAF):

- is **not** in the database.
- is installed on the device.
- is **not** attached to the current user.

Lion (GIRAF):

- is in the database.
- is **not** installed on the device.
- is **not** attached to the current user.

Leopard (GIRAF):

- is in the database
- is **not** installed on the device.

- is attached to the current user.

Calculator (is an Android app):

- is in the database.
- is installed on the device.
- is attached to the current user.

Camera (Android):

- is in the database.
- is installed on the device.
- is **not** attached to the current user.

Gallery (Android):

- is **not** in the database.
- is installed on the device.
- is **not** attached to the current user.

Internet (Android):

- is in the database.
- is **not** installed on the device.
- is **not** attached to the current user.

Mail (Android):

- is in the database
- is **not** installed on the device.
- is attached to the current user.

Table 2: Test cases for the AppAdapter class

Name:	Test:	Pass Criteria:
setAppBackground-001:	Call setAppBackground with the color 0xFFFFFFFF.	The background of the app is the color 0xFFFFFFFF.
saveAppBackground-001:	Call saveAppBackground with the color 0xFFFFFFFF.	The launcher settings are changed so the color for the given app is 0xFFFFFFFF.

Table 3: Test cases for the AppInfo class

Name:	Test:	Pass Criteria:
setGuardian-001:	Call setGuardian with a non-guardian profile.	The guardian of the AppInfo is null.
setGuardian-002:	Call setGuardian with a guardian profile.	The guardian of the AppInfo is the guardian used as input.
getShortenedName-001:	Call getShortenedName on an AppInfo with a name of five characters.	The return value is equal to the name of the AppInfo.
getShortenedName-002:	Call getShortenedName on an AppInfo with a name of six characters.	The return value is equal to the name of the AppInfo.
getShortenedName-003:	Call getShortenedName on an AppInfo with a name of seven characters.	The returned string consists of six characters with "'..."' concatinated.

Table 4: Test cases for the AuthenticationActivity class

Name:	Test:	Pass Criteria:
changeCamerafeed BorderColor-001:	Scan an invalid QR code.	The camera feed border changes to red.
changeCamerafeed BorderColor-002:	Scan a valid QR code.	The camera feed border changes to green.
handleDecode-001:	Scan an invalid QR code.	The camera feed border changes to red and no log-in button and name is shown.
handleDecode-002:	Scan a valid QR code.	The camera feed border changes to green, a log-in button appears, the name of the user attached to the QR code appears and the device vibrates.

Table 5: Test cases for the HomeActivity class

Name:	Test:	Pass Criteria:
onBackPressed-001:	Press the device back button while on the home screen.	Nothing happens.
calculateNumOfColumns-001:	The user should have nine apps visible to them. Call calculateNumOfCollumns().	The returned number is four.
calculateNumOfColumns-002:	The user should have ten apps visible to them. Call calculateNumOfCollumns().	The returned number is four.
calculateNumOfColumns-003:	The user should have thirteen apps visible to them. Call calculateNumOfCollumns().	The returned number is five.
appBgColor-001:	A color for a given app already exists in the settings. Call appBgColor().	The color returned matches the one from the settings.
saveNewBgColor-001:	Call saveNewBgColor() with a real color and the ID of an app in the database.	This color has been saved in the settings.
saveNewBgColor-002:	Call saveNewBgColor() with null as a color and the ID of an app in the database.	The settings are not changed.
saveNewBgColor-003:	Call saveNewBgColor() with a real color and null as the ID.	The settings are not changed.

Table 6: Test cases for the ProfileSelectActivity class

Name:	Test:	Pass Criteria:
loadProfiles-001:	Call loadProfiles() for a guardian with children attached and different departments that also have children attached.	The returned list of children contains all children attached to the given guardian and the children attached to the departments of the guardian, but no duplicates.

Table 7: Test cases for the Tools class, part one

Name:	Test:	Pass Criteria:
saveLogInData-001:	Login as an valid guardian.	The correct ID and timestamp is saved in shared preferences.
findCurrentUser-001:	Call findCurrentUser().	The currently logged in profile is returned.
findCurrentUserID-001:	Call findCurrentUserID().	The currently correct profile ID is returned.
clearAuthData-001:	Call clearAuthData().	The ID is set to -1 and time to 1 in the shared preferences.
sessionExpired-001:	Let the session to expire after 1 minute. Log in as a valid guardian. Turn off device. Wait one minuted. Turn on device.	The guardian is logged out.

Table 8: Test cases for the Tools class, part two (where the app requirements are enforced)

Name:	Test:	Pass Criteria:
getVisibleGirafApps-001:	Call getVisibleGirafApps() for the current user.	Only Wombat is returned.
getVisibleAndroidApps-001:	Call getVisibleAndroidApps() for the current user.	Only Calculator is returned.
getVisibleApps-001:	Call getVisibleApps() for the current user.	Only Wombat and Calculator are returned.
getHiddenGirafApps-001:	Call getHiddenGirafApps() for the current user.	Only Parrot is returned.
getHiddenAndroidApps-001:	Call getHiddenAndroidApps() for the current user.	Only Camera is returned.
getHiddenApps-001:	Call getHiddenApps() for the current user.	Only Parrot, ViewDB, Lion, Camera, Gallery and Internet are returned.
getAvaliableGirafApps-001:	Call getAvaliableGirafApps() for the current user.	Only Wombat and Parrot are returned.
getAvaliableAndroidApps-001:	Call getAvaliableAndroidApps() for the current user.	Only Calculator and Camera is returned.
getAvaliableApps-001:	Call getAvaliableApps() for the current user.	Only Wombat, Parrot, Calculator and Camera is returned.
getDeviceGirafApps-001:	Call getDeviceGirafApps() for the current user.	Verify that Wombat, Parrot and ViewDB is shown.
getDeviceAndroidApps-001:	Call getDeviceAndroidApps() for the current user.	Only Calculator, Camera and Gallery is returned.
getDeviceApps-001:	Call getDeviceApps() for the current user.	Only Wombat, Parrot, ViewDB, Calculator, Camera and Gallery is returned.
subtractAppsList-001:	Call subtractAppsList with a[wombat,Calculator] and b[Calculator].	Only Wombat is returned.

packageRegistered-001:	Call packageRegistered() with Wombat.	True is returned.
packageRegistered-002:	Call packageRegistered() with ViewDB.	False is returned.
insertAppInDB-001:	Call insertAppInDB() with ViewDB.	ViewDB is now in the database.
attachLauncher-001:	A relation between the launcher and given user does not currently exist. Call attachLauncher() with the given user.	A relation between the launcher and user now exists in the database.
appsContain_RI-001:	Call appsContain_RI with list of apps installed on the device as ResolveInfo's and Parrot.	True is returned.
appsContain_RI-002:	Call appsContain_RI with list of apps installed on the device as ResolveInfo's and Lion.	False is returned.
appsContain_A-001:	Call appsContain_A with a list of apps in the database and Wombat.	True is returned.
appsContain_A-002:	Call appsContain_A with a list of apps in the database and ViewDB.	False is returned.

F

AUTHENTICATION DESIGN

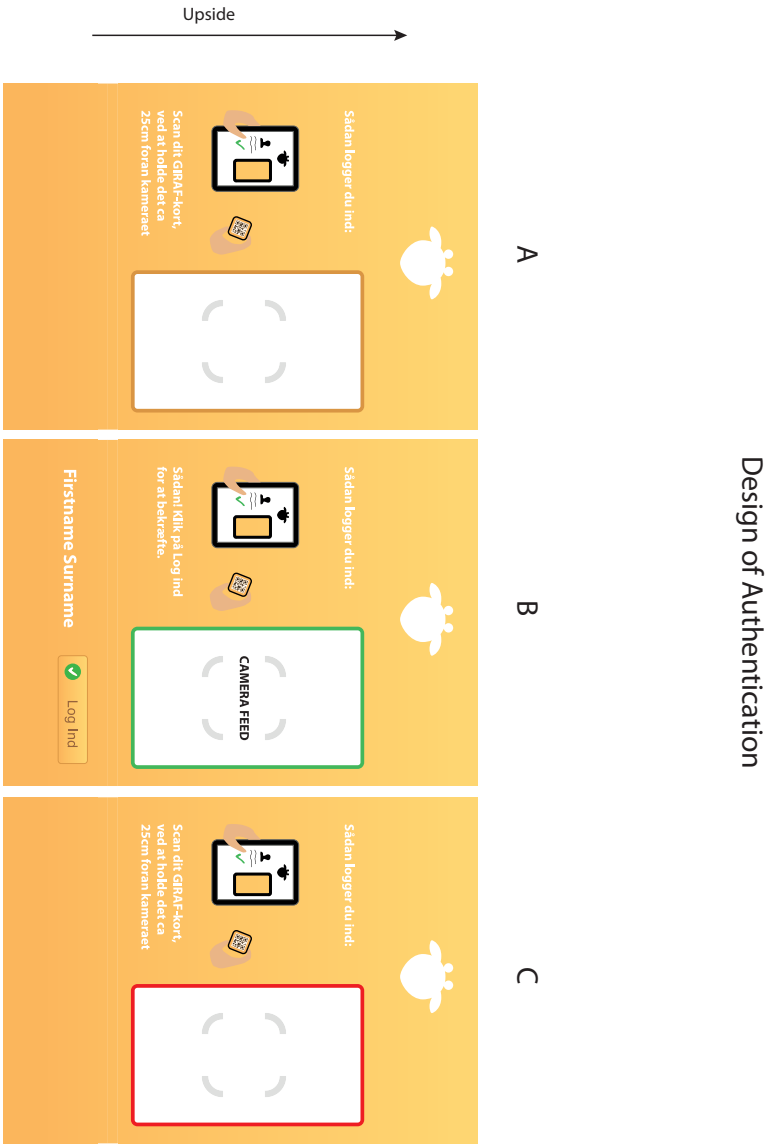


Figure 26: Authentication design

BIBLIOGRAPHY

[1] Naked Fruit A/S. Naked fruit, May 2012. URL http://nakedfruit.dk/.

[2] David Benyon. *Designing Interactive Systems*. Pearson Education Limited, second edition, 2010. ISBN 978-0-321-43533-0.

[3] Lisa Hunt Michael Crowther and Juan Pablo Hourcade. *Does mouse size affect study and evaluation results?* 2007. ISBN 978-1-59593-747-6.

[4] Microsoft Corporation. *The Role of Usability Research in Designing Children's Computer Products*. August 1998. ISBN 978-1558605077.

[5] Aalborg University. Rammestudieordning, 2008. URL http://www.tek-nat.aau.dk/digitalAssets/12/12667_rammestudieordning2008_12_08.pdf.

[6] ZXing. Zxing ("zebra crossing"), 2012. URL http://code.google.com/p/zxing/.