LAUNCHER

Part of The GIRAF Platform

Department of Computer Science
Aalborg University
June 4th 2012

Aalborg University

Department of Computer Science

Selma Lagerlöfs Vej 300

9220 Aalborg East

http://www.cs.aau.dk

Title:

Launcher – Part of the GIRAF Platform

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: 96

Finished: June 4th, 2012.

Synopsis:

Institutions for children with ASD currently employ analogue tools which are suitable for digitalization. Such digitalized tools exist today, but are however not cost effective for the institutions. The "Launcher"-project, as part of a larger multi-project, aims to solve some of the issues by creating parts of a free platform for Android™ tablets, which contains digitalized tools suitable for the institutions in questions. The parts created in this project are a launcher, a component that provides access to other GIRAF components, and a shared library of visual elements, to improve concistency across components running on the platform.

This rapport and its content is freely available, but publication (with source) may only be made by agreement with the authors.

SUMMARY

Institutions for children with ASD currently employ analogue tools which are suitable for digitization. Such digitized tools exist today, but are however not cost effective for the institutions.

This report describes the development of such a tool to help children with ASD and their guardians in their daily routines. The solution is a platform for Android, called GIRAF, built for tablets.

GIRAF was developed using an agile work method by a multi-project group of 18 students. The multi-project consisted of five smaller groups, with each group responsible for an individual part of the system. Five guardians, consisting of educators and pedagogues, were attached as customers, and each group had one customer assigned to them.

GIRAF consists of five parts. The first group developed an app that visualizes time. The second developed an app that enables communicating through images. The third an administration module, that all GIRAF apps use in order to save their data.

The fourth group developed a server module, designed to allow data to be shared across devices. The fifth group developed an app that allows the user to run GIRAF apps, which is the part that this report documents.

Through analysis, usability was found the most important quality for the launcher, and therefore, the highest priority while developing the launcher, was to ensure high usability.

This, among other design decisions, lead to a shared library being designed and implemented, such that common UI components were available for all parts of the GIRAF system.

Lastly, usability tests were performed, to reveal usability issues in the product. The test revealed a single critical, three serious, and three cosmetic issues. We believe these provide a good foundation for further development.

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

1. Traditional analysis, design, and 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 may be more straightforward to understand the analysis, design, and implementation 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 agile work process.

The strength of the second approach is that the ordering of events and decisions is presented chronologically, making the reasoning clearer and more intuitive. The weakness is the lack of a clear and structured way to easily grasp and understand the final state of the analysis, design, and implementation.

A mix of the two have been chosen. We first focus on process oriented activities, their chronological order, and what their effects were. This is followed by a product oriented description of analysis, design, implementation, testing, validation, and lastly an epilogue.

ACKNOWLEDGEMENTS

We would like to thank our supervisor, Ulrik Mathias Nyman, for both the extensive and professional guidance as well as motivation. We would also like to thank *Naked Fruit* [3] for supplying refreshing drinks. Thanks to Henrik Sørensen, Peter Axel Nielsen, Ivan Aaen, and Michael Skov, for advice during the design process.
As a multiproject group, we wish to acknowledge the help and time spent by the educators and teachers from various institutions, whose insight has been of utmost value in the development of the GIRAF system.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

GLOSSARY

XP Extreme Programming

GUI Graphical User Interface

ADS Autistic Spectrum Disorder

MVC Model View Controller [5]

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

We In the introduction, "we" denotes the multi-project group, and every other place it denotes our individual group

Children Refers to "children with ASD"

WOMBAT Abbreviation and title of the timer application – Way Of Measuring BAsic Time

PARROT Abbreviation and title of the pictogram application – Pictogram Assisting with Rhetoric Reasoning Or Talking

Oasis Title of the administrative component

Savannah Title of the administrative server component

Half day A half day of work, typically four hours

Part I

PROLOGUE

The *Prologue* part introduces the shared parts of the GIRAF multi-project.

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[14].

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 [7], explaining what it is like to live with ASD, and an interview with an educator at Birken, a special kindergarten for children, see Appendix H 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 [12]. 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. [12]

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) [16], and Scrum [2].

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 [1].

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 has been shortened to approximately 7 - 14 half days.
- Some degree of pair programming has 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.

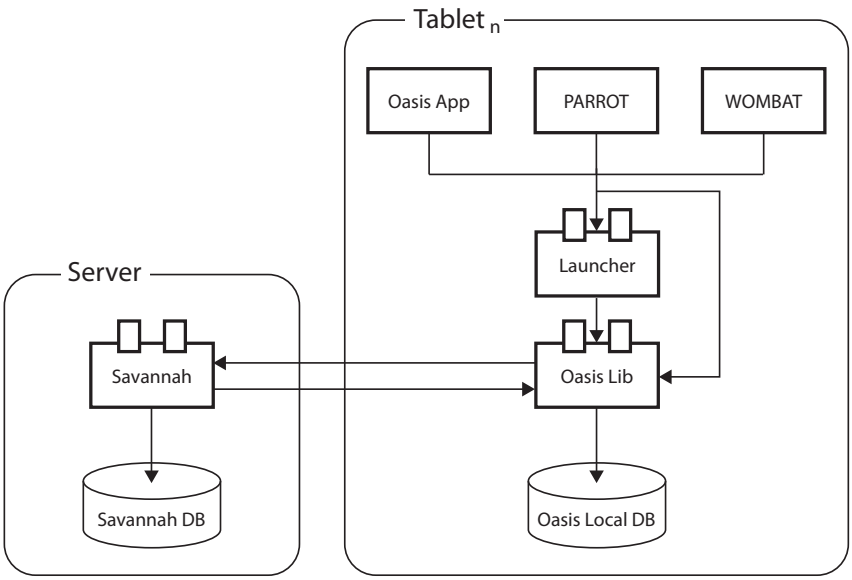

Figure 1: The GIRAF architecture

We have chosen to redesign last year's architecture [9] to make it easier to work with. We have simplified the architecture because we feel it is unnecessarily complex.

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 Figure 34 in Appendix D.

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. [10]

1.8.1.1 *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.

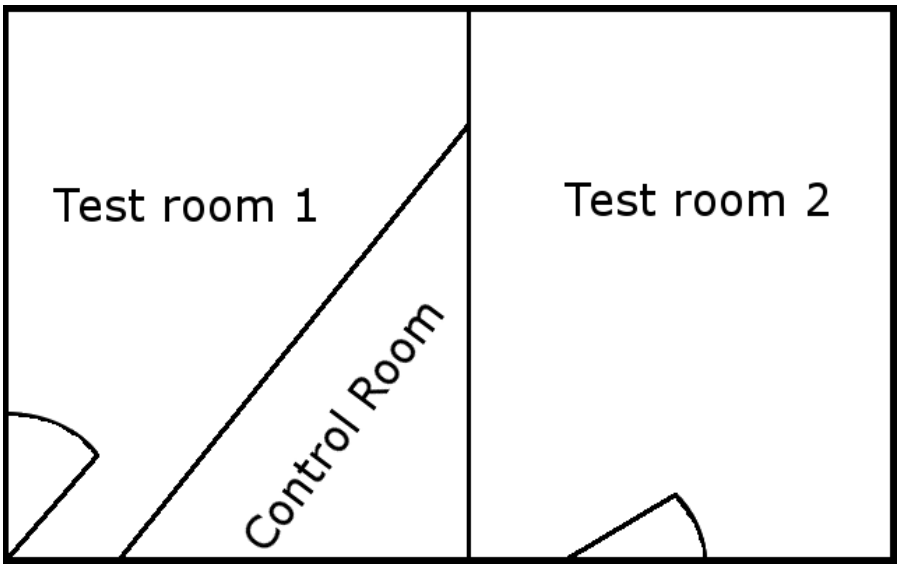

Figure 2: An overview of the usability lab at Cassiopeia, Department of Computer Science, Aalborg University.

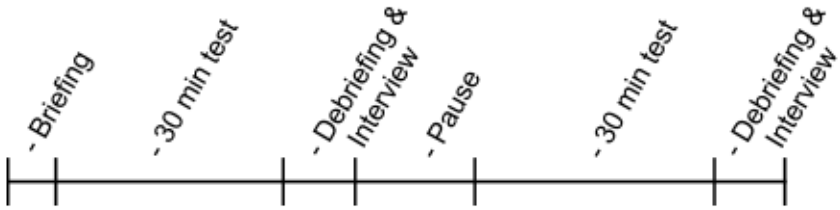

Figure 3: The schedule of the usability test.

1.8.1.2 *Execution*

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 Chapter 9.

Part II

PROCESS

The *Process* part presents what has been done in preparation for developing the GIRAF launcher, together with a process oriented overview of the project, followed by a list of features, both implemented and unimplemented.

2 PRELIMINARY ANALYSIS

In this chapter, the choice of XP and Scrum as the work process is explained, how informal integration testing was conducted, how the customer domain was understood through interviews and field observations, and some usability considerations that were born from that understanding. Finally, the role of prototyping is explained.

2.1 WORK PROCESS

The work form used in our project group is based on XP. XP is built around having a customer available at all times, but this has not been the case for this project. Occasional meetings were used in the project, and found adequate. One of the requirements from the customers, was that it should be "easy to use", which we interpret as requiring high usability. We believe the occasional meetings were sufficient as the project was focused mainly on the usability of the product. This allowed us to spend time refining designs, and only needing occasional feedback from the customer to evaluate the current ideas and design. The requirements gathering, in the form of interviews, is explained later. The usability 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, e.g. pair programming. Pair programming has been a great tool in keeping the development pace, 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 more understandable to them.

The "whole team" and "sustainable pace" conventions were used as well. [15] 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, therefore test driven development was not included.

In addition to our own conventions, there is also a meeting convention used by all project groups. 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 UNDERSTANDING

We set the scope of this project to include the guardians as both customers and users, who can make demands of the product, while the children are considered users. The needs of the children are conveyed by the guardians. 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. [4, p. 152]

The results from the interviews 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 different needs.

2.2.1 *Field Observations*

As part of the interviews were conducted at Egebakken[1], observations were also made. These observations gave insight into the tools used (physical and digital), the robustness of the environment and the organization needed for the children.

2.2.1.1 *Impressions*

A concern could be that the physical environment was unsuitable for the use of tablets, however we found nothing to be concerned about.

The physical tools used at Egebakken fulfill simple tasks, and are robust, but also come with limitations. While they can be very flex-

1 An institution for children and the workplace of our customer.

ible, they can require a sizable effort to work with. Some of these tools are suitable for having digital replacements, as this could lessen the previous issue. Software solutions do however already exist, but our customers find them hard to use and overly expensive. These solutions also come with limited flexibility, limiting 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.

2.3 USABILITY CONSIDERATIONS

As stated, 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, the following considerations are relevant for the project.

> "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."
>
> - [6]

> "The results also provide further evidence that young children require interactions designed specifically for their developing motor skills."
>
> - [11, p. 8]

Another aspect that is important is consistency, especially for children [6], but also generally. [4, p. 90] This includes creating concistency with the real world, using familiar icons and behavior in the product [6].

2.4 PROTOTYPING

Prototypes and sketches were used to envision design ideas within the project group, and for gathering feedback from customers. This worked well with the iterative work process of the project, and new

prototypes and sketches were continuously built on top of existing ones.

Later prototypes were used to demonstrate the design of the product to the customers. These prototypes were implemented on paper, and showed 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

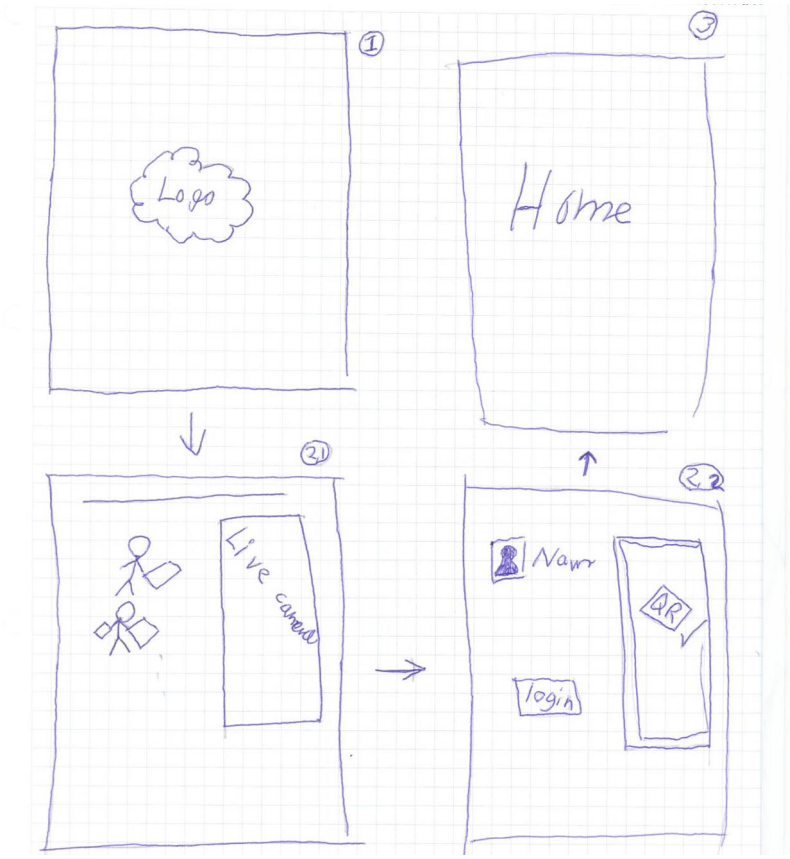

Figure 4: Sample prototype

3 ITERATIVE PROCESS

This chapter chronologically presents information about relevant events and areas of focus in each sprint. Durations of the sprints are presented in half days.

The first sprint started on March 19th and the last development sprint stopped May 11th. Following sprints are not considered as development contributions were not made to the product.

SPRINT 1

Duration: Seven half days.

The focus of this sprint was on usability. This includes creating use cases, evaluating authentication 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.

The meetings held in the multiproject group until this point had been marked by lengthy discussions and knowledge sharing, which made the meetings unnecessarily long. 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, e.g. 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 due to time limitations.

SPRINT 2

Duration: Ten half days.

The focus of the second sprint was to start implementing the base design, which was already in place. This led to building the parts of the launcher allowing apps to be arranged and loaded, and creation of an authentication system being the priority for the sprint, with no requirements yet coming from the other project groups. 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. The issue was discussed on the meeting, and rules were agreed upon to help alleviate the issue.

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 apps, providing profile information for apps, and refinement of the authentication screen. This is also where integration with the Oasis library started, increasing our reliance on the Oasis group.

SPRINT 4

Duration: Ten half days.

This sprint revolved around the presentation of the product, and worked on app management, specifically the drawer mechanism. Resources were also put into improving the integration with the Oasis library, with features like saving settings for each app.

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, during the development period, were prioritized based on their relevance for the design and the usability of the product, with the highest priority entries being implemented first. They are not presented in a prioritized order in this report.

4.1 IMPLEMENTED FEATURES

4.1.1 *Guardian Mode*

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

4.1.2 *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.

4.1.3 *Authentication*

Authentication restricts access to data, 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.

4.1.4 *Autologin*

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

4.1.5 *GIRAF GUI Components*

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

4.1.6 *App Grid*

App grid displays icons and titles of GIRAF apps, and allows the user to launch these.

4.1.7 *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 4.1.6.

4.1.8 *Logo Screen*

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

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, i.e. without profile selection, 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 in the *unimplemented*-section of the 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 has one dependency. The service 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 fulfills this dependency is provided by the Oasis lib.

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

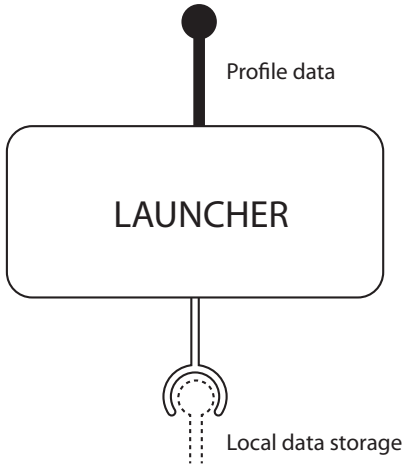

Figure 5: The GIRAF launcher component.

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.

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

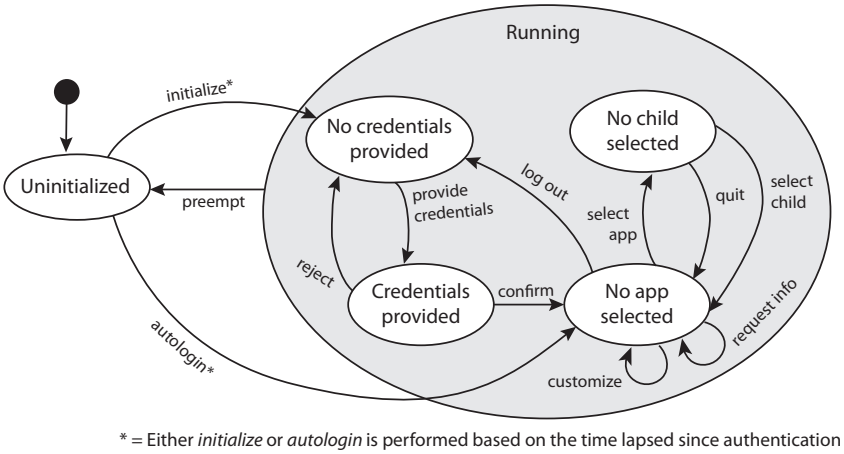

Figure 6: State diagram

Figure 6 shows all possible states and transitions, which the launcher can be in. These states grouped together represent four functionalities of the launcher:

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

These functionalities are shown in Figure 7, along with the states they represent, and the arrows between them. The arrows represent dependencies between each functionality, e.g. to reach the app management functionality, the launcher must have been through the initialization and authentication functionalities.

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

In order to highten usability, as requested by the customers in Section 2.2, it is decided to standardize elements in the GUI. These standards together are referred to as the *design language*. Furthermore, it is decided that the concrete elements are to be shared, such that other GIRAF components can utilize the standards.

5.1 DESIGN LANGUAGE

As stated, the design language is a collection of design standards, specifically for the GIRAF system. The design language is based on usability concerns found in Section 2.3.

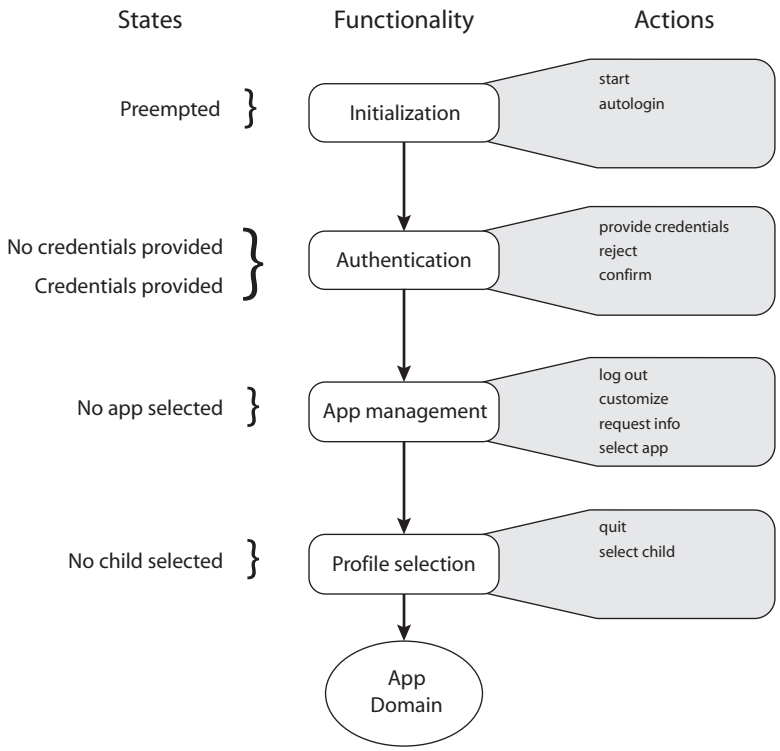

Figure 7: Functionality diagram

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 so the user knows the system is loading.

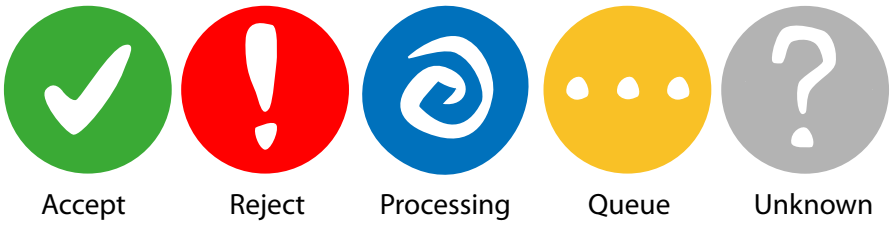

Figure 8: Status icons in the GIRAF system

5.1.2 *Colors*

Colors where chosen for the GIRAF system, these can be seen in Figure 9. These colors are designed for use in two different cases. The first four yellow and brown colors are chosen for use together, and

the last four colors, white and grey, are made for use together. These color groups were chosen, such that they contained colors with contrast, and colors that could be used for gradients.

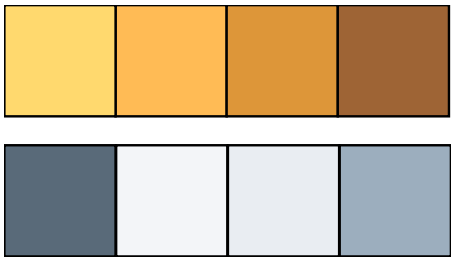

Figure 9: The color theme of the GIRAF system

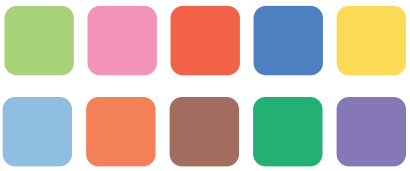

Figure 10: App colors of the GIRAF system

The colors shown in Figure 10, are the app colors available in the GIRAF system. These colors are chosen to be dark, to allow white silhouettes to be placed on top of them, while letting the silhouettes remain clearly visible.

5.1.3 *Interactive elements*

The idea is for all interactive elements in the GIRAF system to have round corners, and if an element is not interactive, it should have square corners.

An example can be seen in Figure 11, where two button illustrations are shown, but the system also includes draggable objects and other interactive objects.

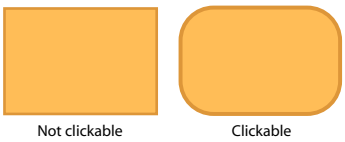

Figure 11: Interactive elements: Buttons illustrations

5.2 INITIALIZATION

The Android operating system can preempt any app at any time [8], and it might be cumbersome to authenticate often, if the GIRAF

launcher is executed after a preemption. To improve the user experience, the launcher must be accessible in a sufficiently fast manner after a preemption.

One could argue that this feature introduces a security issue, as an attacker, if able to physically get a device which has been authenticated within the last eight hours, could start the launcher in order to get full access.

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

In Figure 6, *Uninitialized* is the first state reached. There are two transitions from *Uninitialized*:

- Initialize
- Autologin

The transition *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 *No credentials provided*.

The transition *autologin* is taken in case authentication was done less than eight hours ago, and brings the launcher to *No app selected*, 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

Validation is needed to ensure privacy. Confirmation is a requirement as error prevention, in case validation with a wrong identity is performed. As stated, 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 modelled child and guardian contains private data and therefore needs to be protected. QR-codes were chosen as the means of authentication, as they provide some level of security.

An alternative to QR-codes could be a *username-password* combination, where each user has their own username, with a private password. However, as the launcher is developed towards being a tool usable by both guardians and children, a username-password combination is inappropriate, as it can diminish usability for children. This requires the user to remember their credentials, and some children have problems with this.

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

QR-codes provide a analogue way of storing the user credentials, the user's certificate, 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 a tablet 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. To sum up, QR-codes are chosen because they improve usability, despite of their ability to be copied.

A flow chart of the authentication functionality can seen in Figure 12.

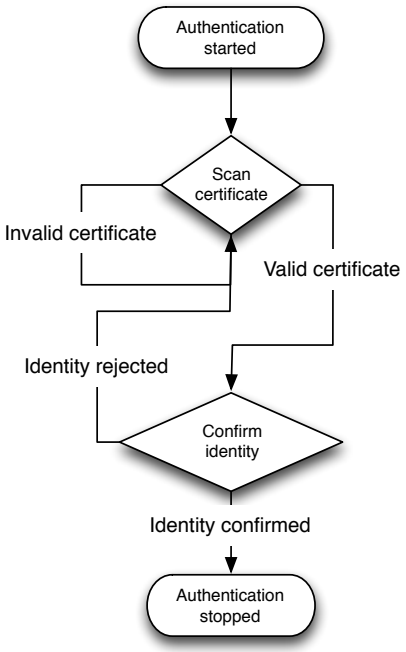

Figure 12: Flow chart of the authentication functionality

Upon scanning a QR-code, there are 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 the QR-code

is valid, the device vibrates to notify the user, and the process enters its second step. The second step is the confirmation of the identity, or rejection if the identity represented on the system does not match the user's desired identity.

5.3.1 *GUI*

The GUI components of the authentication design can be seen in Figure 13.

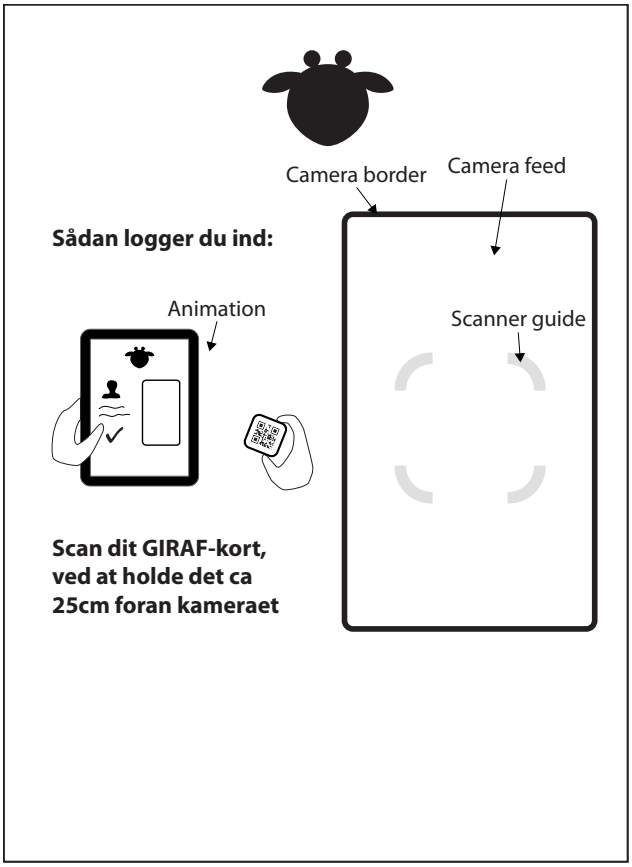

Figure 13: Illustration of the authentication components

The camera border helps distinct the camera feed from the rest of the layout, but it also provides feedback upon scanning by changing colors — this can be seen in Appendix F. The backwards facing camera feed is surrounded by the camera border. A scanner guide is overlaid on top of the camera feed to remind the user of scanners in general and the behavior that is expected of them. Figure 14 shows the complete graphical design.

On the left hand side an animation is shown, to guide the user through the scanning process.

Figure 14: GUI of the authentication functionality

5.4 APP MANAGEMENT

When the user launches a GIRAF app, the transitions between *No app selected* and *No child selected*, are taken, as seen in Figure 6.

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

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

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 given, 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 Figure 5.

Furthermore, additional pieces of information are available:

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 an app is installed on a device running GIRAF. Since the GIRAF platform uses both local and remote storage, there might be latency in synchronizing these storages. It is therefore important for the user to know if both storages are synchronized.

Figure 15 shows a flow chart of the interactions the user can perform, and the actions the launcher takes upon the interactions, in order to fulfill the requirements listed earlier.

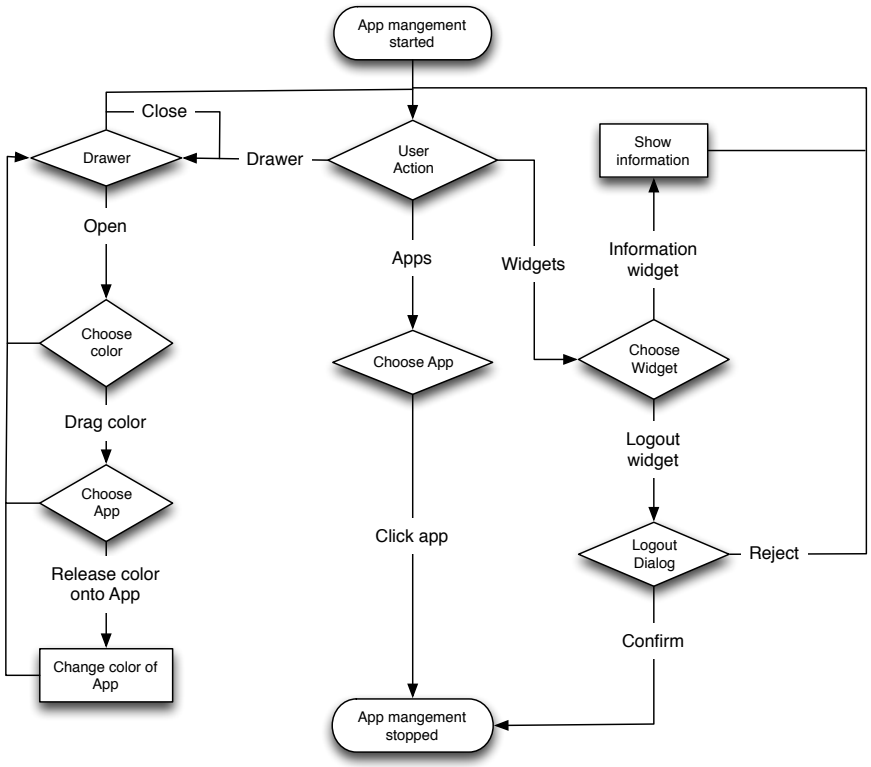

Figure 15: Flow chart over the app management functionality

Figure 15 shows three branches of user interactions and system actions:

DRAWER For a user to change app settings, he or she must open the drawer, a component explained below. In the drawer, there is a color picker, also explained later, from which the user chooses a color and drags it to the desired app. The user can "cancel" the color change at any point, either by not choosing a color, or releasing the color over something that is not an app.

APPS When launching an app, the user chooses an app to launch and clicks it. This brings the launcher out of *App Management*, and into *Profile Selection*, explained later. The user can "cancel" the action by not choosing an app.

WIDGETS Finally, the user can request information, which happens through widgets, explained below. There are different widgets in the system, and every widget contains unique information. The user chooses which one to recieve information from, and clicks the widget containing the desired information to have it shown. If the user does not choose a widget, the action is cancelled.

5.4.1 *Drawer*

As stated, the drawer was designed to hide functionality, that is not always needed, in a convenient place. As seen in Figure 15 everything that has to do with changing app settings is placed in the drawer, and the user has to open the drawer to get to this functionality. The drawer is shown in Figure 16 in context.

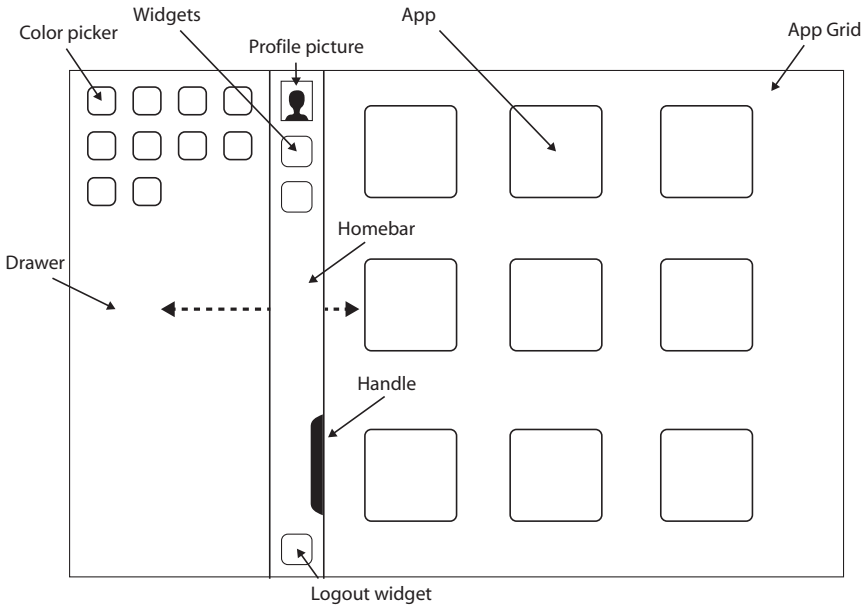

Figure 16: Illustration of the app management components

To highten consistency, the handle and the inside of the drawer have the same color. The handle has rounded corners, see Section 5.1.3, to tell the user that this element is interactive. An illustration with the drawer closed can be seen in Figure 18, and one with the drawer open in Figure 17.

Figure 17: Design of app management, with drawer opened

Figure 18: Design of app management, with drawer closed

5.4.1.1 *Widgets*

Widgets in the GIRAF system are used to display information to the user, and might let the user take an action. An action can e.g. be to present additional information to the user, or to produce a side effect in the system. Figure 16 shows the placement of the widgets in context.

All widgets are designed with round corners to signal interactivity, in accordance with Section 5.1.3. The widget *logout* is kept seperately from the other widgets, as it allows the user to change the state of the launcher to *No credentials provided*, as seen in Figure 6, while the other

widgets keep the launcher *No app selected*. The widgets can be seen in Figure 17.

5.4.1.2 *Color picker*

All colors in the color picker have rounded corners as they are interactive. They have white edges to keep them distinguishable from the background. The color picker consists of ten predefined colors. The reason for using predefined colors in the color picker is to ensure contrast. If developers of GIRAF apps make their app icons monocolored white, the contrast would always be sufficient. This should ensure that users will be able to see the icon, no matter what icon background color was chosen.

Figure 16 shows the placement of the color picker in context, and Figure 17 shows the complete graphical design.

5.5 PROFILE SELECTION

As explained, 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.

No child selected in Figure 6 shows the state of the launcher when the system is waiting for the user to select a child profile to launch the previously selected app with. From *No child selected*, there are two transitions:

1. select child
2. quit

Regardless of the transition taken, the launcher will be brought back to *No app selected*. The difference is in the side effects of these transitions.

The side effect of the transition *select child*, is the execution of the previously selected app, whereas the transition *quit* unselects the selected app.

The flow chart in Figure 19 shows the steps involved in the profile selection process.

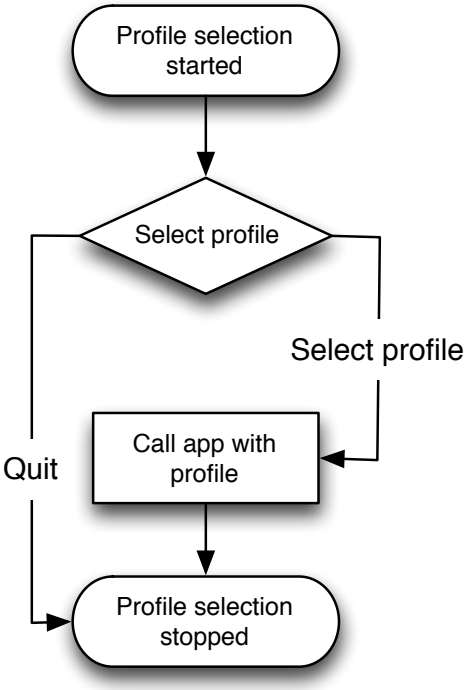

Figure 19: Flow chart of the profile selection process

6 IMPLEMENTATION

The *Implementation* chapter describes the implementation of the designs presented in the previous chapter. The functionalities in the design are represented with Android activities. The structure of these activities are first described, followed by an explanation of – in our opinion – non-obvious details of each activity. Furthermore, examples of the implementation of the shared visual components are given.

6.1 ACTIVITY STRUCTURE

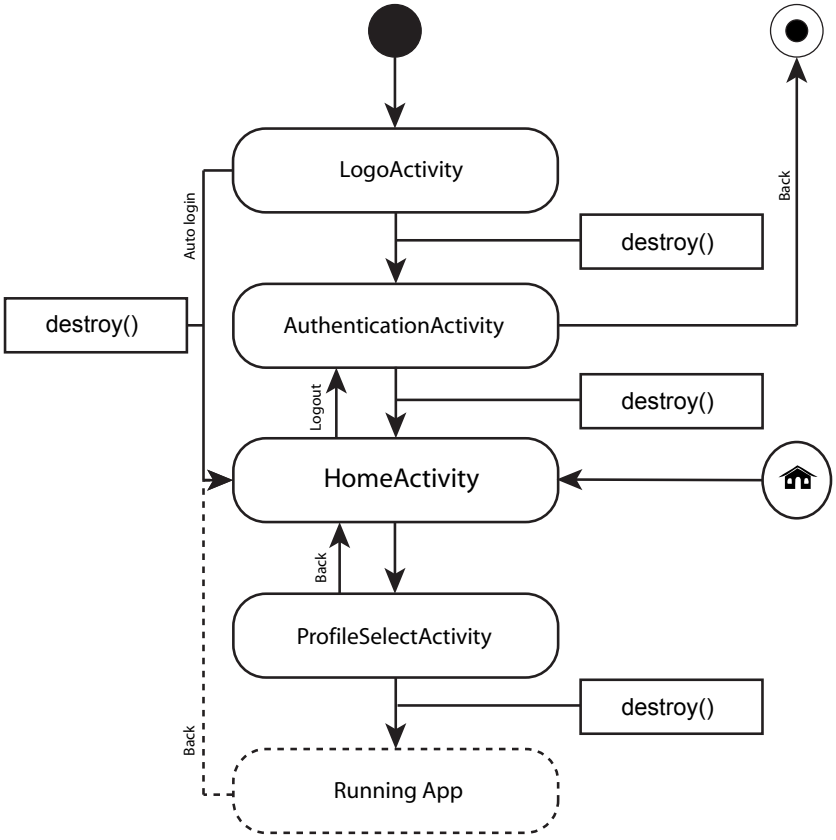

Figure 20: Activity diagram of the launcher

Figure 20 shows the activity structure of the GIRAF launcher. Every path with a `destroy()` box on it gets the activity starting the path destroyed. Pressing the back button will also the destroy the activity it was pressed, and the *HomeAcitivity*, which implements the app management functionality, is also destroyed on log out. Every time the user is outside the launcher and hits the home button, they will end

up in the *HomeActivity*, if the GIRAF launcher is set to the default launcher. Note that this is a feature in Android. The dotted section symbolizes that the launcher does not have any control in this area, as it belongs to the launched app.

When the launcher is started, the user is presented with the *LogoActivity*, which implements the initialization functionality, from which they will be redirected to the *AuthenticationActivity*, which implements the authentication functionality, to perform the authentication. In this process, the *LogoActivity* is destroyed. If the user authenticates, the *HomeActivity* is called. If an app is clicked in the *HomeActivity*, the user will be presented with the *ProfileSelectActivity*, which implements the profile selection functionality, and must choose a child before they can start this app.

6.2 GIRAF GUI COMPONENTS

GIRAF GUI Components is the name of the library, which is the implementation of extensions to Android components, such that they satisfy the standards explained in Section 5.1.

The library consists of extensions of standard Android classes:

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

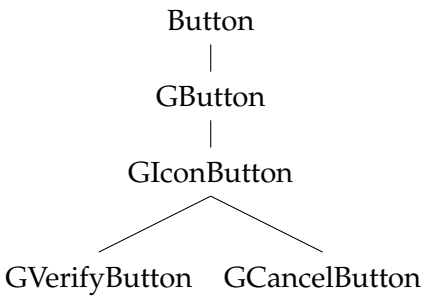

Figure 21: Inheritance of GIRAF buttons

Figure 21 shows the following classes, which directly or indirectly implement extensions to `android.widget.Button`.

GBUTTON implements coloring, according to the the design language, presented in Section 5.1.

GICONBUTTON implements functionality which allows for the use of an icon inside the button. An icon could be a status icon, as described in Section 5.1.1.

GVERIFYBUTTON simply uses functionality from `GIconButton` to set the icon to the *accept* status icon, from Section 5.1.1.

GCANCELBUTTON sets the icon to the *reject* status icon from Section 5.1.1, by using functionality from `GIconButton`.

The principle of extending android classes is applied to all the classes listed earlier in the itemize, and the the inheritance for all components in the library are shown in Section A.1.

6.3 LOGOACTIVITY

The *LogoActivity* is a splash screen, which shows the GIRAF logo while the system is loading. When the *LogoActivity* is running, when looking at Figure 6, the launcher is in *Uninitialized*.

As the launcher might spend more than a reasonable amount of time to load the *AuthenticationActivity*, and thus might give the impression that the system has crashed or is not responding. The *LogoActivity* is implemented to let the user know the launcher is loading. Having a loading animation could improve the *LogoActivity* further, as the user would not be in doubt about whether the system had frozen or was simply taking a while to load, as seen in Section 4.2.6.

The *LogoActivity* implements the autologin feature, explained in Section 5.2, by using the *sessionExpired* method, found in the *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

6.4 AUTHENTICATIONACTIVITY

This activity implements the authentication functionality described in Section 5.3. It is also used, as a proof of concept, as the current lock screen by WOMBAT, as presented in Section 4.2.5.

An open source QR-code scanning library released under *Apache License 2.0*, called ZXing [17], is used for the QR-code scanning functionality. Changes are made to the original ZXing library to make it suit our needs, in accordance with the license.

There is also implemented vibration, and an animation has been implemented to guide users through using the scanner, as mentioned in Section 5.3.

The *AuthenticationActivity* currently contains a work around for an administration issue that has not been solved yet. The snippet showing this can be seen in Listing 3, and needs further explanation:

```
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: Code snippet from the AuthenticationActivity

When a GIRAF component saves settings regarding a user, the component needs to be attached to the user in question. During the development, this has not always been the case, and therefore, a workaround has been introduced, which is shown on line three in Listing 3.

6.5 HOMEACTIVITY

The *HomeActivity* implements the app management described in Section 5.4. As seen in Figure 15 in App Management, there are three branches of interaction. One of these is the *Launch app* functionality. A prerequisite of launching an app, is that the app is available to the user. To find out which apps to load, two lists of apps are retrieved. The first is the apps attached to the user in the Oasis database, and the second is the apps installed on the device. The apps to be loaded are computed by taking the intersection of these two lists.

This ensures users are not allowed access to apps on the device they are restricted from.

The launcher does not ensure that apps are installed. A consequence hereof is that even if a user has permission to access an app, it will not be accessible if not installed.

6.5.1 *Drawer*

The drawer is implemented as described in Section 5.4.1. The implemented design can be seen in Figure 22 and Figure 23.

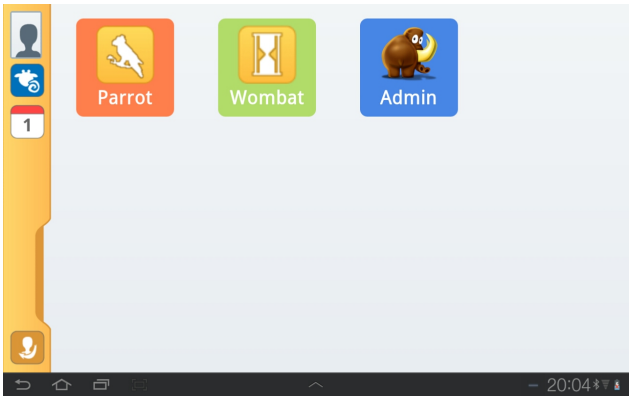

Figure 22: Shows the launcher with the drawer closed

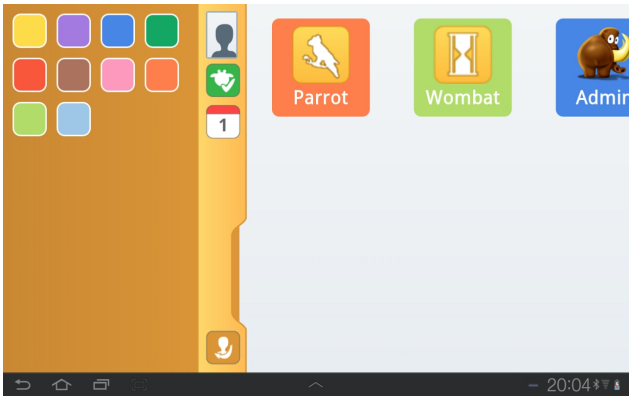

Figure 23: Shows the launcher with the drawer opened

6.5.2 *Color picker*

We believe that usability of the color picker can be heightened by further development on the class shown in Listing 4. The drag cases can be further developed to improve usability, as they can e.g. be used to indicate that the apps are able to recieve the dragged view, e.g. a color.

```
1  public class GAppDragger implements OnDragListener {
2      @Override
3      public boolean onDrag(View view, DragEvent drawEvent) {
4          switch(drawEvent.getAction()){
5          case DragEvent.ACTION_DRAG_STARTED:
6              break;
7          case DragEvent.ACTION_DRAG_ENTERED:
8              /*
9               * Future todo: implement highlighting
10              */
11             break;
12         case DragEvent.ACTION_DRAG_EXITED:
13             /*
14              * Future todo: implement dehighlighting
15              */
16             break;
17         case DragEvent.ACTION_DROP:
18             long id = Long.parseLong((String)view.getTag());
19             int color = (int) Integer.parseInt(drawEvent.getClipData().
                   getItemAt(0).getText().toString());
20
21             AppAdapter.saveAppBackground(view.getContext(), view, color,
                   id);
22             break;
23         case DragEvent.ACTION_DRAG_ENDED:
24             /*
25              * Future todo: implement dehighlighting
26              */
27             break;
28         }
29         return true;
30     }
31 }
```

Listing 4: The GAppdragger class

6.5.3 *Widgets*

The widgets are implemented in the GIRAF GUI Components library, and built as isolated classes such that they can be utilized in differents contexts. As widgets might need updates, we have implemented a periodic update mechanism, that updates GIRAF widgets in a uniform manner. This manner refers to all widgets being updated by a

central updater, instead of running multiple clocks to update widgets seperately. This behavior is enforced by having all widgets in GIRAF implement the IGWidget-interface, and any widget to be updated should be added to the central updater.

6.6 PROFILESELECTACTIVITY

The *ProfileSelectActivity* implements the behavior described in Section 5.5, and takes care of providing the profile service, described in Chapter 5.

The profile service is implemented using intents, as shown in Listing 5.

```
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 5: Code snippet showing the provision of profile data

The launched GIRAF app can retrieve the information by following the "Launcher Services" guidelines, see Appendix G, provided to the project groups.

Part IV

TESTS

The *Tests* part explains the prioritization of the different software qualities, what the unit testing would have provided had time permitted, the usability testing that was performed, and the currently known bugs in the product.

7 PRODUCT QUALITIES

This chapter explains why, amongst others, usability was highly prioritized, and why e.g. performance was not.

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 procured, but there are other quality aspects to take into account. As the product is meant to be developed on by other students later, static software qualities like having maintainable and structured code are very important as well. 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. Static qualities refers to qualities that are observable without using the product e.g. looking at the code, while dynamic qualities are only observable while using the product.

Because there is limited time for development however, not all relevant qualities can be prioritized equally, while still building a functioning product. Therefore, 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 HIGH PRIORITY QUALITY ASPECTS

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

7.1.1 *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.

7.1.2 *Consistency*

Concistency refers to consistent design and component use, which helps simplify the user experience, increasing usability [4, p. 90]. This

aspect was promoted to a high priority, as it has an impact on the usability of the system. To achieve consistency, the GIRAF GUI Components library, as mentioned earlier, was created, not only to create consistency in the launcher, but to allow consistency throughout the system.

7.1.3 *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 guardian users. To achieve reliability, the system has been recurringly, albeit informally, tested and we have been pragmatic about fixing bugs.

7.1.4 *Structured Code*

Structured code aims to help increase code readability and understandability, which we believe 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 reduce code duplication.

7.1.5 *Maintainable Code*

Having maintainable code, refers to 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.

7.1.6 *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 un-

derstand 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.

7.2 LOW PRIORITY QUALITY ASPECTS

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

7.2.1 *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.

7.2.2 *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, 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.

7.2.3 *Performance*

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

7.2.4 *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 specifi-

cation and developing with that in mind, but rather on developing features as time permits with the customer requirements in mind.

7.2.5 *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. There is a dilemma between usability and flexibility, as usability improves by heightened concistency. Flexibility works against concistency, and achieving both flexibility and usability requires, in our opinion, significant effort.

8 CODE TESTING

Though having testable code has not been a priority, see Chapter 7 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 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 that 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 the set of test cases should be rewritten to further exercise the code.

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 for more details on the working process.

Pair programming and the refactoring employed resulted in static white-box testing of the project, as both of these have code reviewing as an essential component.

9 USABILITY TESTING

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) as a disadvantage of their current solutions, and making functionality easy to discover would help reduce the need for such courses.

9.1 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.

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.

Table 1: Results from the usability test

9.1.1 *Dissection of Results*

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 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, e.g. by changing the icon to a more telling one.

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 by giving additional instructions, e.g. visual clues.

In 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 front 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 to 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 5.4.1.1.

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 one or both of the following reasons:

1. Lack of knowledge about the system

2. Users did not understand the task that was tested

Users having the right information, e.g. additional visual clues, could therefore make an important difference for the usability of this component.

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.

9.1.2 *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.

10 KNOWN BUGS

This chapter describes all bugs confirmed to exist.

10.1 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.

10.2 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 the development period had ended for us.

10.3 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 the edge of the screen than the design calls for. This bug is not critical, but could potentially confuse the users of the product.

Part V

EPILOGUE

The *Epilogue* part ends the report by discussing issues, such as a lack of a global backlog, and the total democracy of the multiproject group, providing remarks to hopefully future developers, and finally concluding upon the project.

11 DISCUSSION

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

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 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 [13], 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. This could be accomodated for by reducing the scope of the project.

The backlog is a crucial element in agile development and it is used to encourage developers and help communication across the multi-project. Backlogs were created in each project group, but a global one was not created, for the multiproject group. As seen in Chapter 3, 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, the Savannah group might have prioritized synchronization between Oasis and Savannah 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.

A goal in XP is to motivate the team members by letting them choose their own tasks, so they can pick the tasks they feel motivated about. This can however reduce motivation, e.g. bad mood, when no one in the team feels compelled to do a certain task. Working in pairs might solve the problem, as our experience with pair programming makes us believe that working in pairs could lessen the burden of completing a tasks which otherwise noone would be compelled to

do.

Communicating the progress in each project group was done by using burn down charts, amongst other things, e.g. meetings. The burn down charts were hard to get accurate readings from, as each project group had their own measurement of work load. It might not be needed to have burn down charts for further development.

The essence of working iteratively is the fact that 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 the report after the development process ends, for us, creates an issue which also occurs in the waterfall development model. The issue is in regards to scheduling, as delays in one part of the schedule reduces the time available to work on remaining parts.

11.1 REMARKS AND FUTURE WORK

This section is for the group who might continue the development of the GIRAF launcher.

We suggest that a decreasing number of meetings is held throughout the project, e.g. one weekly meeting at the start, and one every other week by the end.

We believe it is important to keep an open mind about the needs of each project group, and staying respectful, while maintaining communication, and socialization.

It is also important to define the scope of the project early and make it clear to the other project groups what will be implemented and what will not.

As concistency increases usability, adapting common visual elements is also something we consider important, for example through usage and development of the GIRAF GUI Components library.

12 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?"

Together with four other project groups, we have developed the GIRAF platform, an Android platform for children with ASD. We have specifically developed the launcher component, which deals with: authenticating users, launching apps and allowing the user to read, and change platform settings.

We began by spending time 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. Usability was found as one of the key aspects of the project during this period. This lead to envisioning through the creation of prototypes and sketches, prior to the actual design of the launcher. 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 remaining sprints focused on implementing functionalities.

We implemented a library containing GUI components for the GIRAF system, in order to enhance usability by adding consistency. In the implementation, we also focused on creating maintainable and readable code, to ensure that the development of the product 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 GIRAF GUI COMPONENTS INHERITANCE TREES

These trees show how the components in the GIRAF GUI Components library extends from each other.

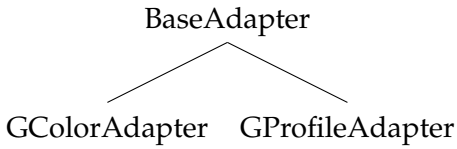

Figure 24: Inheritance of GIRAF adapters

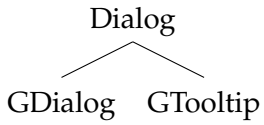

Figure 25: Inheritance of GIRAF dialogs

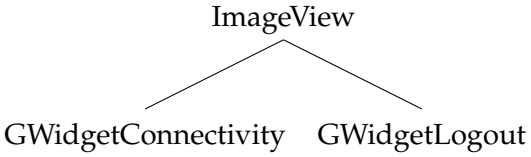

Figure 26: Inheritance of GIRAF ImageView

A.2 LOGO ACTIVITY

A.3 PROFILE SELECT ACTIVITY

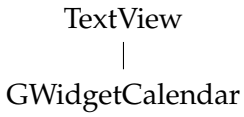

Figure 27: Inheritance of GIRAF TextView

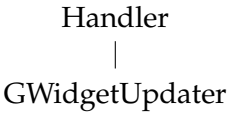

Figure 28: Inheritance of GIRAF Handler

Figure 29: Inheritance of GIRAF ListView

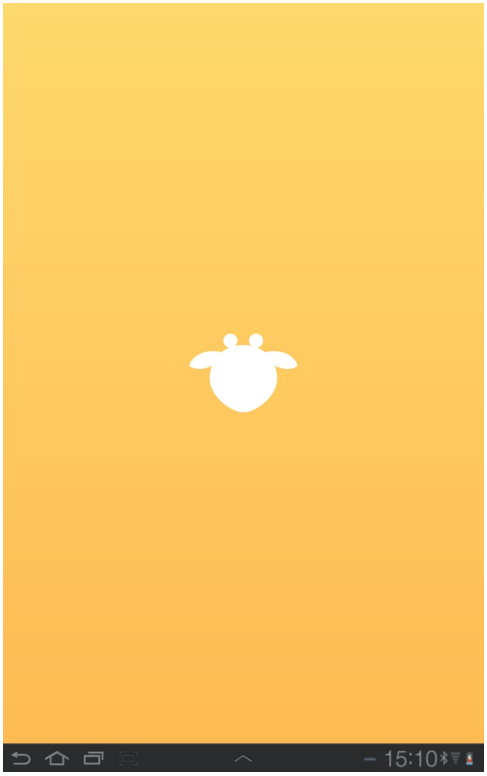

Figure 30: LogoActivity in portrait mode

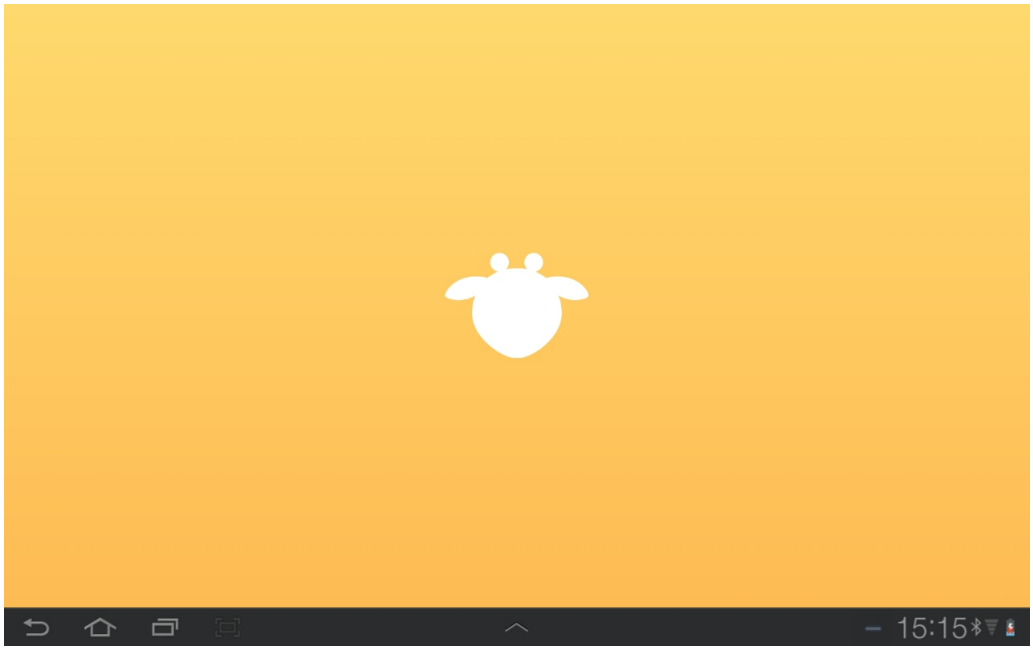

Figure 31: LogoActivity in landscape mode

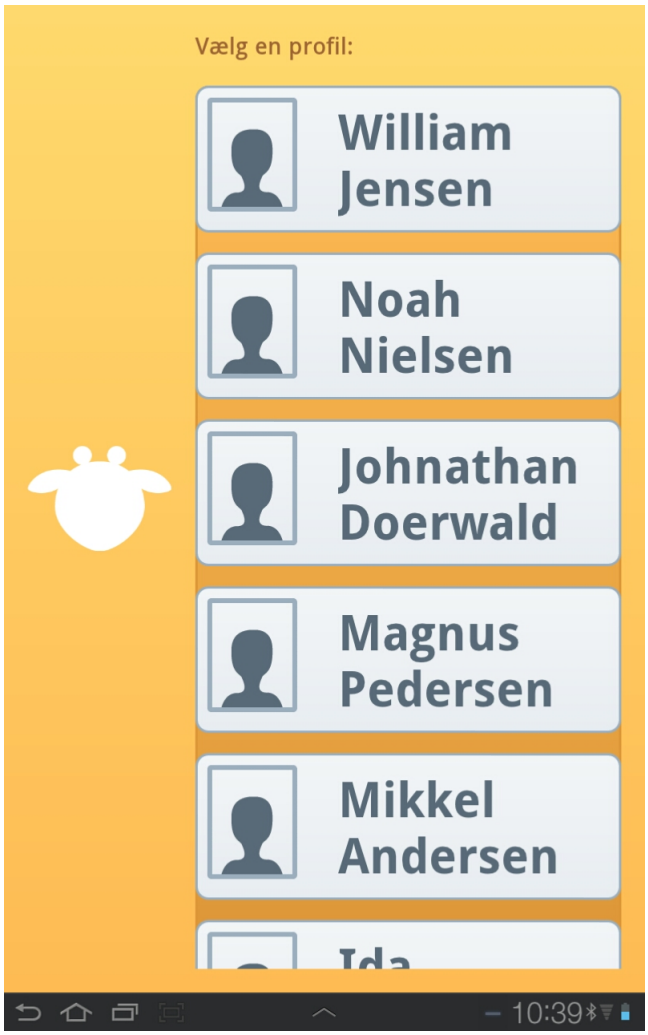

Figure 32: ProfileSelectActivity in portrait mode

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 33: 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 34: Invitation asking for customer participation in a usability test

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 35: 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 36: Demographic 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

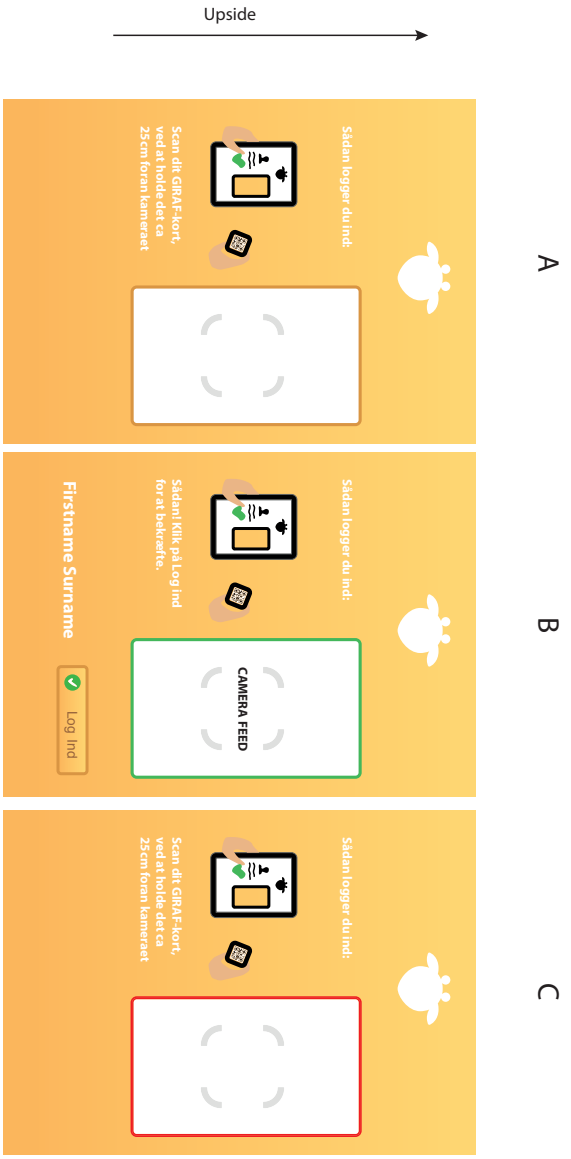

Figure 37: Authentication design

G

LAUNCHER SERVICES

"Hvad vi stiller til rådighed"

Child ID (v001) – Det barn en app skal køres for får sit ID (en Long) sendt med i det Intent, der starter appen. For at få fat i IDet skal følgende gøres inde i main activity:

```
Long id = getIntent().getExtras().getLong("currentChildID");
```

Guardian ID (v001) – Den guardian, der starter en app får sit ID (en Long) sendt med i det Intent, der starter appen. For at få fat i IDet skal følgende gøres inde i main activity:

```
Long id = getIntent().getExtras().getLong("currentGuardianID");
```

Authentication (v002) – Vores QR authentication activity stilles til rådighed, og kan kaldes med følgende kode (den returnerer ikke data):

```
Intent i = new Intent("dk.aau.cs.giraf.launcher.AUTHENTICATE");
i.addCategory("dk.aau.cs.giraf.launcher.GIRAF"); startActivity(i);
```

OBS! Authentication håndterer ikke tryk på Home- og Back-knapperne, og kan derfor nemt omgås i den nuværende version. Vi har pt. ingen planer om at gøre noget ved dette, så mener man at den funktionalitet er vores ansvar er man velkommen til at komme ind og snakke med os om det.

Background color (v003) – I launcheren har hver app en baggrund og farven på den baggrund bliver sendt med i et intentet, der åbner apps. Hent det således:

```
int color = getIntent().getExtras().getInt("appBackgroundColor");
```

H

CUSTOMER INTERVIEW

This is notes from an interview with Mette Als Andreasen, an educator at Birken in Langholt, Denmark.

Når tiden løber ud (kristian har tage et billede):
Færdig - symbol
Gå til skema - symbol
Taget fra boardmaker

Kunne være godt hvis man kunne sætte egne billeder ind som start/stop symboler.

Rød farve = nej, stop, aflyst.

De har sådan et ur på 60 minutter hvor tid tilbage er markeret med rød, og så bipper den lige kort når den er færdig.
Det ville være fint hvis de kunne bruge sort/hvid til dem der ikke kan håndtere farver, men også kan vælge farver.

Stop-ur:
en fast timer på 60 minutter + en customizable som ikke ser helt magen til ud, som f.eks, kan være på 5, 10 eller 15 minutter for en hel cirkel.

Timeglas:
skift farve på timeglassene, men ikke nødvendigvis gøre dem større. Kombinere med mere/mindre sand. Eventuelt kombinere med et lille digitalt ur, til dem der har brug for det, skal kunne slåes til og fra.

Dags-plan:
ikke særlig relevant til de helt små og ikke særligt velfungerende børn. Men kunne være rigtig godt til de lidt ældre.
En plan går oppefra og ned, og hvis der så skal specificeres noget ud til aktiviteterne, så er det fra venstre mod højre ud fra det nedadgående skema.

Til parrot:
Godt med rigtige billeder af tingene, som pædagogerne selv kan tage, eventuelt også af aktiviteter, så pedagogerne kan have billeder af aktiviter som de kan liste efter skeamet.

Der var mange skemaer rundt omkring, og der henviser det sidste billede i rækken til næste skema, som hænger f.eks. på badeværelset eller i garderoben.

BIBLIOGRAPHY

[1] Scrum Alliance. Advice on conducting the scrum of scrums meeting, May 2007. URL http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting. Last visisted 3/6/2012.

[2] Scrum Alliance. Scrum alliance, 2011. URL http://www.scrumalliance.org/.

[3] Naked Fruit A/S. Naked fruit. URL http://nakedfruit.dk/. Last visisted 3/6/2012.

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

[5] Microsoft Corporation. Model-view-controller. URL http://msdn.microsoft.com/en-us/library/ff649643.aspx. Last visisted 3/6/2012.

[6] Microsoft Corporation. *The Role of Usability Research in Designing Children's Computer Products*. October 1998. ISBN 978-1558605077. Last visisted 3/6/2012.

[7] Temple Grandin. Teaching tips for children and adults with autism, December 2002. URL http://www.autism.com/ind_teaching_tips.asp.

[8] Google Inc. Activity. URL http://developer.android.com/reference/android/app/Activity.html. Last visisted 3/6/2012.

[9] Steffan Bo Pallesen Jacob Bang, Lasse Linnerup Christiansen. Administration module for giraf, May 2011. URL http://people.cs.aau.dk/~ulrik/Giraf/Admin.pdf.

[10] Jan Stage Jesper Kjeldskov, Mikael B. Skov. Instant data analysis: Conducting usability evaluations in a day. 2004.

[11] Juan Pablo Hourcade Michael Crowther, Lisa Hunt. Does mouse size affect study and evaluation results? 2007.

[12] Samsung. Samsung galaxy tab 10.1, 2011. URL http://www.samsung.com/global/microsite/galaxytab/10.1/index.html. Last visisted 3/6/2012.

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

[14] Aalborg University. Studieordning for bacheloruddannelsen i software, September 2009. URL http://www.sict.aau.dk/digitalAssets/3/3331_softwbach_sept2009.pdf.

[15] Don Wells. Extreme programming: A gentle introduction. URL http://www.extremeprogramming.org/. Last visisted 3/6/2012.

[16] Don Wells. The rules of extreme programming, 1999. URL http://www.extremeprogramming.org/rules.html. Last visisted 3/6/2012.

[17] ZXing. Zxing ("zebra crossing"). URL http://code.google.com/p/zxing/. Last visisted 3/6/2012.