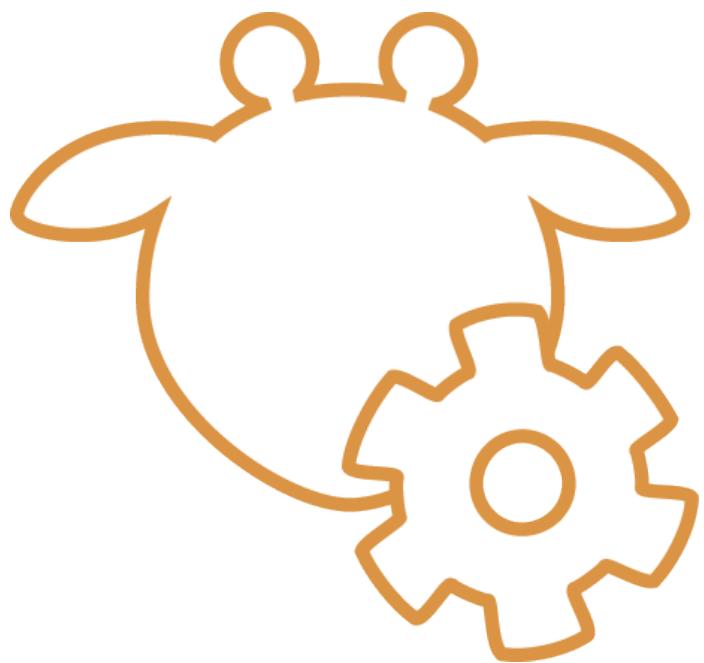


GIRAF - Admin



SW6 PROJEKT
GROUP SW601F13
DEPARTMENT OF COMPUTER SCIENCE
AALBORG UNIVERSITY
JUNE 4th 2013



AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
Telephone: +45 9940 9940
Telefax: +45 9940 9798
<http://cs.aau.dk>

Title:

GIRAF - Admin

Theme:

Developing Complex Software Systems

Project period:

P6, Spring Term 2013

Synopsis:

Project group:

SW601F13

Participants:

Jens M. Lauridsen
Johan Sørensen
Lars Chr. Pedersen
Tommy Knudsen

Supervisor:

Shuo Shang

Circulation: 7

Page count: 123

Appendix count and type: 5, Installation Guide and Configuration, Graphical Mockups of the GIRAF System, Transcript of Meeting with the head of kindergarten (1/3 2013), Usability Testing Appendix, Colour Theme

Finished on June 4th 2012

This report documents the development of an administration system for the GIRAF system as of Software 6 multiproject 2013. The administration system is designed for high maintainability and user friendliness. The back-end for the system is written entirely in PHP, the front-end is a combination of HTML, JavaScript and CSS. The administration system should enable users to administrate users, applications and pictograms. However the pictogram and application systems are not fully implemented. To administrate pictograms refer to the Parrot project under the GIRAF system. Mockups of the complete system is viewable in the appendix.

The report content is freely available, but publication (with source), only after agreement with the authors.

Preface

This report is written by four students from the Department of Computer Science at Aalborg University, and is a part of the multi project GIRAF. The four students are studying as Software Engineers at their 6th semester.

This report both serves as a documentation of the project process as well as documentation of how the system works.

Included with this report, there is a CD containing the project's Git repository. Which contains the report, source code of the system as well as the graphic assets used in the project.

The group would like to personally thank:
Ulrik Nyman for great input and support during the project.
Mette Als for her patience and valuable input into the designing of the system.

Signatures:

Jens Mikkel Lauridsen

Johan Sørensen

Lars Chr. Pedersen

Tommy Knudsen

Contents

Title Page	3
Signatures	7
Contents	12
I GIRAF Introduction	17
1 The GIRAF Project	19
1.1 Vision for GIRAF	19
1.2 Previous Years	19
1.3 Target Platform	21
1.4 Autism	21
2 The GIRAF Project 2013	23
2.1 The Goals for 2013	23
2.2 Definition of a Multi-Project	24
2.3 Group and Work Structure	24
2.3.1 Development Method	24
2.3.2 Development Tools	25
2.4 Decision Making - The Process	27
2.4.1 The Weekly Meeting	27
2.4.2 Rules of Conduct	28
2.4.3 Committees	28
3 What was developed	31
3.1 Pictograms, Morgana, and Design Guidelines	31
3.1.1 Pictogram	31
3.1.2 Morgana	33
3.1.3 Design Guidelines	33
3.2 The Project of 2013	34
3.2.1 Admin	34
3.2.2 Cars	34
3.2.3 Croc	34

3.2.4	Parrot	35
3.2.5	Tortoise	35
3.2.6	Train	35
3.2.7	Wasteland	36
3.2.8	Zebra	36
3.3	Acknowledgement	36
II	Analysis	37
4	System Design	39
4.1	Starting from scratch	39
4.2	PHP on Android	40
4.3	Folder structure and other practical issues	40
4.3.1	Folder Structure	41
4.3.2	PHP and JavaScript	42
4.3.3	Language Support	42
5	Project Progress	45
5.1	Weekly Abstracts	45
5.2	Weekly Progress	45
5.3	Conclusion of Project Progress	48
6	Problem Statement	49
III	Solution	51
7	System Overview	53
7.1	Login	54
7.2	My Profile	54
7.3	Profiles	54
7.4	Create Profile	54
7.5	Add Relation	54
7.6	Pics Manager - Make	55
7.7	Pics Manager - Add	55
7.8	Pics Manager - Remove	55
7.9	Pics Manager - Edit	55
7.10	Pics Manager - Delete	55
7.11	Dep. Information	55
7.12	QR Manager	55
7.13	App Manager	56

8 Design	57
8.1 Designing individual pages	58
8.1.1 Login	58
8.1.2 Navigation	58
8.1.3 Profile Page	60
8.1.4 Profiles	60
8.1.5 Pics Manager	60
8.1.6 Department Manager	61
8.2 App. Manager	61
9 Implementation	63
9.1 Navigation	63
9.2 QR Code Generation and Printing	66
9.3 Profile Picture Change	67
9.4 Multi-language Support	70
9.4.1 Adding a new language	70
9.4.2 Adding a new sub-site	70
9.5 Pics Manager - Make	71
9.6 PHP Session Variables Overview	75
10 Usability Testing	77
10.1 Structure of the test	77
10.2 Result of the test	78
10.3 New Features	81
IV Perspective	83
11 Conclusion	85
11.1 Evaluation of the Multi group project	86
11.1.1 Cooperation between groups	86
11.1.2 Advice for next year	87
11.1.3 General Evaluations	87
12 Future Work	89
12.1 Project Proposals	89
13 Acknowledgements	91
13.1 Technologies used in this project	92
Bibliography	94
V Appendix	95
A Installation Guide and Configuration	97
A.1 Requirements	97

A.2 Installation	97
A.3 Configuration	98
A.4 Auto Update	98
B Graphical Mockups of The GIRAF Admin System	99
C Transcript of Meeting with the head of the kindergarten (1/3-2013)	115
D Usability Testing Appendix	119
E Colour Theme	123

Danish Summary

Projektet Admin er et underproject af GIRAF sytemet der har været under udvikling siden 2011. Admin har til fokus at udvikle administrations interface til GIRAF systemet til håndtering af bruger, pictogramer og applikationer. For forårssemestret 2013 havde Admin projektet fokus på at omdesign interfacet til et system der er nemmer at vedligeholde på tablet og desktop samt mere brugervenligt.

For at skabe et projekt der samtidig også er optimal at arbejde videre på, i årene der kommer var der først fokus på at få indentifiseret alle funtionaliteter. Som der er krav på og designed disse i et samhængende system.

Hovedfunktionaliteten der skulle være funktionel efter projektets afslutning ved sommeren 2013 er bruger håndtering. Bruger håndtering er vigtigst da dette ikke kan håndteres andre steder i GIRAF systemet.

Problemformulering for projektet var:

“Currently there are two different administration interfaces for the GIRAF system. This results in a problem with maintainability and user friendliness. How can we make a single user friendly administration interface for the GIRAF system?”

og på dansk:

“På nuværende tidspunkt er der to forskellige administrations inter- face for GIRAF systemet. Dette skaber problemer med vedlige- holdelse og brugervenlighed. Hvordan er det muligt at skabe et enkelt brugervenligt adminstrations interface for GIRAF sys- temet?”

Systemet er bygget i PHP, og Apache. Målet med projektet var også at lave et system som både kunne køre på en PC, men også på en Android tablet. Et forslag på en løsning er at bruge PAW, som gør det muligt at eksekverer PHP på en Android tablet.

En stor del af fokusen i projektet har også været at gøre det så nemt som muligt for den gruppe som tager over, at fortsætte med projektet. Rapporten består af flere dele der beskriver projektets udvikling. Det samlede admin system består

af et login- , et profil- , et pics (billede)- , et QR kode- og et app-system.

Designet af systemets front er bygget ovenpå Twitters Bootstrap, som hjælper med hurtigt at udvikle hjemmesider. Det overordnet design er baseret på GIRAF design guidelines. Login systemet er opbygget ved hjælp af PHPs session. Der er også udviklet et multi-sprog system som er bygget til at efterligne Android's sprog system. Dette gør at det er nemt at tilføje nye sprog til systemet, samt at folk som er bekendt med Android's system hurtigt vil blive bekendt med systemet.

Profil systemet fungere som informations oversigt, samtidig med at den gør det muligt at holde styr på hvilke brugere der kan modifisere andre brugere.

Pics systemet er lavet sådan at man nemt fra sin PC kan tilføje nye pics til systemet, samt tilknytte pics til et barn, da det tidligere har været svært at tilføje allerede eksisterende pics til GIRAF systemet.

QR kode systemet er lavet til at hvis man taber sin QR kode, så er det muligt at få lavet en ny. QR kode systemet bruger et bibliotek kaldet phpqr code, som gør det muligt at tage en tekststreng og lave det til et billede som viser QR koden.

App systemet er designet sådan at man kan bestemme hvilke apps et barn har mulighed for at gå ind i.

Under projektet er der desuden lavet en brugervenlighedstest, hvor testpersoner blev inviteret til at afprøve systemet. Ved hjælp af brugervenlighedstesten blev der fundet en del fejl og mangler, hvor nogle af dem nåede at blive rettet. På grund af problemer med en anden gruppe nåede alle funktioner i systemet ikke at blive færdig, og disse er beskrevet i rapporten.

For at konkludere så er projektet ikke færdigt, men der er sat et godt grundlag for den næste gruppe der skal arbejde videre med projektet. Yderligere er der også givet forslag til hvad den næste gruppe kan lave videre på, for at gøre systemet bedre.

Introduction

This project is a part of the multi-project called GIRAF, which stands for: “Graphical Interface Resources for Autistic Folk” which have been an active project at Aalborg University since 2011.

As the name suggests this system is a tool made for helping children with autism to communicate. The system is to be actively used in the specialized kindergartens and schools in Aalborg.

This report contains first an introduction to the GIRAF project, which describes the GIRAF project as it is as of the day this report was written. This introduction is contained in all project reports for the GIRAF multi-project, and can therefore be skipped if the reader is already familiar with the GIRAF project as of May 2013. The place to be skipped to is then part II on page 39.

The second part of this report also contains the thoughts that went into constructing the new administration system, as well as some in depth explanation of the implementations, for the GIRAF system.

The administration system makes it possible to edit and create users for the GIRAF system and in its finished state it is possible to administrate both from a tablet and from a PC platform. It also comes with the possibility to make, edit and manage pictograms on a PC platform.

In this report we also use some terms that often is misunderstood, as well as a few special words. We therefore want to clarify them here before we begin. The definitions can be seen in Table 1 and is from [1].

Word	Meaning
Website	A set of related webpages served from a single web domain
Webpage	A single page displayed on a website
Homepage	The initial or main web page of a website
Site	Refers to the word website
Pictogram	A special element, developed for the GIRAF project. Containing sound, text and image
Guardian	A person with responsibility for a child with autism, either a pedagog or a parent

Table 1: Word Explanation

Part I

GIRAF Introduction

Chapter 1

The GIRAF Project

Graphical Interface Resources for Autistic Folk (GIRAF) started in 2011 as a semester project targeting children with autism and their guardians. In the following chapter, the overall vision for the GIRAF project will be presented, the projects from previous years will be explained briefly along with the platform for the project. Lastly a section describing autism is included.

1.1 Vision for GIRAF

The vision for GIRAF is to create a multi-purpose application based on *Android* which can simplify and ease the lives of autistic children and their guardians.

The purpose of GIRAF is to replace physical items that are being used daily by the children and their guardians with digitized versions. The idea is to gather several functionalities in one object and allowing customization for each individual child.

This will also optimize work procedures on the individual institution in such a way, that guardians will save time doing repetitive tasks such as making pictograms. This time could be spent with the children instead.

As of spring of 2013 three schools and institutions for children with autism in Northern Jutland are involved in the development, but the hope is that GIRAF will be distributed across all similar institutions in Denmark.

1.2 Previous Years

During the first year of development, four parts of the GIRAF project were developed. The four projects were developed during the spring semester of 2011 and included the projects:

Admin An administration interface used for administrating different aspects of the GIRAF system.

DigiPECS A digitized version of “Picture Exchange Communication System”[11] a system used as an aid for communication with people with special needs such as autism.

Launcher A home screen application and distribution platform for Android.

aSchedule A visual schedule for the Android platform.

During the spring semester of 2012, five new software groups continued development of the GIRAF project. The projects developed during 2012 were:

Launcher An enhancement of the launcher project developed during the spring semester of 2011.

Oasis An enhancement of the admin project from 2011. Furthermore the Oasis project developed a local database for the GIRAF system.

Parrot An enhancement of the DigiPECS project from 2011. The project was renamed because of trademark issues.

Savannah A server side database with web interface for the GIRAF system.

Wombat An Android application for measuring and visualizing time.

During the spring semester of 2012 two databases were developed, however synchronization between them was never achieved.

Problems with Initial Implementation

As the spring semester of 2013 started, an ”install party” for the students was held. The party was intended to help the students compile and deploy the projects from 2012.

Even though a representative for each of the 2012 groups were present, some compilation problems still occurred.

The repository used for distributing in 2012, was disorganized and difficult to navigate, i.e. due to:

- Multiple copies of the same project.
- Unclear dependencies among the different projects.
- Projects only meant to be compiled from Eclipse for Windows.

During the following week a working workspace was created and shared with the rest of the students, along with install instructions. The install instructions were later updated to a more clear edition.

1.3 Target Platform

Android is an open-source operating system originally developed by Android Inc, and later bought by Google Inc. The first release came in 2007, where it was launched by Google Inc. together with Open Handset Alliance (OHA), which includes companies such as Samsung, HTC, LG and Google.

Before the first students were involved in the project in the spring of 2011 Ulrik Nyman considered two platforms for the development of the project. The Android and iOS platforms. The Android platform was chosen for three main reasons:

- That the platform is open source.
- That in Android the developers can take control of the functionality of the home button.
- That distribution of the software is possible outside the official marketplace.

For the two following years it has been chosen to stay on the Android platform. This is done to be able to both reuse the source code and because Android compatible hardware is available for the students. In the very long term the system could support multiple platforms.

1.4 Autism

Autism is a spectrum disorder, meaning that it appears in different variants and not all people who are diagnosed have the same symptoms. The disorder can often be observed within the first three years of a child's life. Autism is a physical condition and is linked to abnormal chemistry in the brain, however the exact causes of these abnormalities are still unknown.[4]

Symptoms

Children with autism usually have difficulties understanding the concept of "play pretend", meaning that they have a hard time imitating the actions of others when playing and therefore prefer to play alone. Furthermore they have difficulties with social interaction and communication – verbally and non-verbally.

People diagnosed with autism may;

- Be very sensitive to light, noise, touch, and taste.
- Have a hard time adjusting to new and changing routines.
- Show unusual attachments to objects.

Autism diagnosed individuals may have a hard time starting and maintaining a conversation. They may communicate with gestures instead of words, develop language slower or faster than normal and some do not develop any language at all. Furthermore the lack of social interaction means they might have a hard time making friends, may be withdrawn and may avoid eye contact.[4]

Signs and tests

If a child fails to meet any of the following language milestones, it may be an indication that it needs to be tested for autism;

- Babbling by 12 months.
- Gesturing (such as pointing or waving goodbye) by 12 months.
- Saying single words by 16 months.

Children failing to meet any of the previous mentioned language milestones might receive a hearing evaluation, a blood test and a screening test for autism. Since autism covers a broad spectrum of symptoms, a single brief evaluation cannot predict what abilities the child has. Therefore a range of different skills are evaluated, such as:

- Communication
- Language
- Motor skills
- Speech
- Success at school
- Thinking abilities

Some parents might be scared of having their child diagnosed, however without a diagnosis, the child might not get the necessary help.[4]

Treatment

Autism cannot be cured, however an early diagnosis and treatment can greatly improve the child's quality of life. Different treatment programs usually build on the child's interests and are highly structured to their needs and routines.[4]

Chapter 2

The GIRAF Project 2013

When working in a multi-project consisting of eight groups, it is important to have a common goal for the project. This chapter describes this goal as a story. Furthermore the chapter includes description of the development process and the rules of conduct.

2.1 The Goals for 2013

Within the first couple of weeks, when all the groups had been assigned a project, a major story for the overall project was written.

The Major Story for 2013

“The guardian arrives at the institution, and turns on the tablet. The guardian is aware of the arrival of a new child at the institution after lunch. The guardian sets up and customizes a profile for the child, this includes creation of new pictograms. Furthermore the guardian prepares games and a life story for the child.

After lunch the new child and the guardian meet. The child is introduced to the communication tool Parrot. After some introduction they sit down to do some communication practice using the tool.

Afterwards the child wants to go outside to see the rest of the institution, and needs to put on some outdoor clothes. The guardian introduces the child to the Zebra tool, and together they put on the child’s outdoor clothes.

When the child comes back in, the guardian and the child play the games prepared earlier by the guardian.

When they are done playing the child and the guardian read the child’s life story using Tortoise.”

2.2 Definition of a Multi-Project

A multi-project is a project which includes multiple groups which each work on their own sub-project, which is part of a larger project. In this case, the larger project is the GIRA F system and each group works on a separate part of the system.

Compared to working on a single project in isolation, working together creates new challenges. The software produced by each group has to be integrated to ensure the entire system works properly. Some projects are more independent on the rest, while others depends heavily on some projects like the database project Wasteland described in Section 3.2.7. Groups have to be flexible and pass any requirements to other groups' projects early to prevent halts.

To ensure the project is successful and no misunderstandings occur, there must be good communication and cooperation between the groups. This requirement is amplified by the fact that there are no definitive authoritative figures, other than those chosen by project members.

2.3 Group and Work Structure

This section describes the development methods used during the spring semester of 2013, including stories and project management tools.

The section is rounded off by a description of the development tools used, including Redmine, Git, and Jenkins.

2.3.1 Development Method

Having a development method is one of the main ways to structure the work process of a project. A development method is a collection of methods and structures, from the way to have meetings, gathering requirements and structuring the development. There are many development methods, each is structured and handles issues differently, however, it is rare that one fits a development problem perfectly. Different methods are often combined and customized to fit the problem at hand.

Implemented Development Methods

This project's nature calls for agile development, due to team collaboration, user feedback, product focus, and continuous integration. Agile development focuses on a flexible but structured work progress suited for projects with many unknown variables. The agile development method has the ability to adapt to changing requirements throughout the project and focuses on having a shippable product at the end of each iteration.

Stories

User stories is one of the tools that helps streamline the work process, keeps focus on a shippable product and is the main component for management of the project. First of all the product story works as a common problem statement for all work groups. A product story is the agreement on what is necessary for the product to be finished. From product story each group can extract what is required of them to complete the story.

Management

The semester coordinator, Ulrik Nyman, has supervised the project since its beginning. Ulrik Nyman himself has a child with autism and will continue being a part of the project for the time to come, conveying his knowledge of the development process and the product. To help fit the product to the needs of guardians, for whom the product is intended, a number of representatives are included for more detailed feedback on the process and the product.

To keep as many work hours in development and to keep a good overall management, common meetings were held weekly. The common meetings had focus on sprints and team cooperation. Problems that needed further discussion and/or development were discussed by a committee consisting of a few representatives from each group.

The common meeting and committee meeting are further specified in sections Section 2.4.1 and Section 2.4.3.

2.3.2 Development Tools

A number of tools were used in order to optimize team collaboration and to make the projects more accessible. These tools will be further explained in the following sections.

A dedicated Linux server was commissioned for the entire GIRAF project and several services installed to facilitate collaboration and agile development. Common to all current services are their free, open-source nature and support of LDAP authentication, allowing all students and supervisors to log in using their AAU credentials.

Redmine

Several tools were audited for use in the project management aspect of development, including Trac, PivotalTracker and Github. Redmine, a Ruby-On-Rails web application, was selected owing primarily to its support of multiple projects and support features such as wikis, forums, milestones and various charts. The features most broadly used will briefly be described here.

Projects All projects live in a shared project space, and can be placed in a hierarchy under a super project. In this regard, the primary multi project served as the base of each of the eight groups' underlying projects.

Issue handling Redmine's primary feature is its issue handling. Project members can create and react to issues within custom-defined domains. For GIRAF, this was primarily development tasks, but could just as well be used for report-related tasks or general maintenance in an attempt to manage time usage.

Burndown Charts Redmine does not have native support for burndowns, but does support it through a Free and Open-Source Software (FOSS) third-party plugin. Burndowns are a visual aid of each subproject's progress throughout a sprint, giving quick summary of development speed and whether proactive action may need to be taken.

Milestones A generic milestone feature in Redmine is Versions. Versions are simply markers with a set date, and can be open or closed for attachment of issues. The burndown plugin couples a version's end date with attached issues and their progress to generate the related charts.

Wiki A per-project wiki module exists in Redmine. The basic wiki markup has been expanded to allow referencing of almost any other element in the project hierarchy, such as projects, issues, files and VCS revision.

Redmine has many more features not directly applied during this project period. However, many could be applied to create a more centralised and structured development experience in future projects. Examples include file and document hosting, advanced issue workflows, permission management and VCS integration. Future multiprojects may consider expanding into these fields if they feel proficient in Redmine's basic usage.

Version Control System

The university's IT services offers only a single version control system, Subversion. Although centrally supported and backed up regularly, Subversion's shortcomings were challenged before main development had begun. Most notably, the system's centralised workflow and high operation cost. Many of SVN's actions require access to the central server. Two alternatives without these issues were suggested: Git and Mercurial (Hg). The former was chosen as a general question of broad platform support and popularity. A primary strength of these systems is their support of separate branches of development without the constant need to connect to a central server. This allows developers of each project to synchronize with a main branch while maintaining several development branches on their own workstation.

Most groups used Github as hosting solution for development of their projects, as a git hosting solution was not immediately forthcoming (contrary to Subversion and Mercurial, Git does not have a default server implementation). At the conclusion of the project period, a solution was configured using Apache-based LDAP authentication, deferring authorisation and repository management to Gitolite, a low-footprint open-source offering.

In the interest of easier cross-project code contribution and inspection, an improved web solution may prove a better choice. Due to time constraints, a few solutions were briefly audited but ultimately discarded in preference of Gitolite. Gitlab should be mentioned as it featured an interface and features very close to those of Github itself, but proved difficult to install and maintain.

Jenkins

A principal element of agile development is continuous integration, the automated concurrent building of new code as it is pushed to central repositories which ensure constant availability of newest binary packages while catching coding errors before pushing them to the public. Jenkins, a fork of Oracle's Hudson, was suggested early and, given no proponents, was implemented. Build jobs were set up for each project, polling their origin repositories for new Git builds to main branches. If a repository has new code, it is downloaded and built. In case of build errors, the project developers are notified by email. To facilitate the deployment phase of each sprint, all projects are rebuilt every Thursday night and pushed to a public FTP server as well as making them publicly available by HTTP.

Git support is not part of Jenkins' core feature set, but is available as a plugin. During development, unhandled exceptions in the plugin code resulted in thousands of superfluous builds as a failed build due to unexpected circumstances was not marked as failed.

2.4 Decision Making - The Process

The following section will describe the decision making process, set in place to ensure that everyone would be heard on an equal and democratic footing. The decision making process during this semester's multi-project consists of two different steps.

2.4.1 The Weekly Meeting

It was strongly recommended by the semester coordinator, Ulrik Nyman, to hold a weekly meeting for all software students on the bachelor semester of 2013. The meeting's agenda consists of a few points of formalism at the very beginning, in which a secretary and a moderator are chosen by means of voting. Candidates for these roles are entirely self-appointing and a vote is issued to pick one of the candidates.

Though the weekly meeting is established to ensure a higher level of communication between students, as well as to ensure that decisions will be taken on a multi-project level scale, not all points are actually discussed at this meeting. Instead, a committee approach is agreed upon, see Section 2.4.3. The purpose of establishing committees is to ensure that relevant discussions to a given topic can be had, but within a smaller audience.

Committees are discussed at the weekly meeting where voting determines which committees are established. A chairman for a committee is self-appointed and a vote determines if there is consent to let the given person be chairman.

The meeting will then proceed and discuss the ideas and suggestions agreed upon within each committee from the previous week and at the multi-project level determine, by voting, which ideas are okay, or if any of the points concluded by one of the committees are subpar and should be reworked.

2.4.2 Rules of Conduct

During the first weekly meeting some general rules of conduct were established, including decisions on how voting should be done. A number of ways to do this were suggested. Ultimately it was decided that every person present at the meeting has an individual vote, and the idea of a group based voting system was therefore discarded. Furthermore in the event that there is a 50/50 split, the vote will have to be reissued. There must be majority 'for' or 'against' a decision. Guidelines for when a decision should be taken at the weekly meeting were established as well. If a decision involved only two or three groups, then it would not be necessary to discuss it at the weekly meeting. If, however, the decision impacted everyone, a committee would be established to make these decisions.

During a committee meeting every group has a single vote. It is possible to send as many group members as it is deemed necessary to the committee meetings, however, it does not increase the number of total votes a group has.

2.4.3 Committees

A committee ideally consists of a representative from each multi-project group and a chairman agreed upon at the weekly meeting. The chairman is responsible for setting up the meeting, time, place, agenda as well as writing down the details of what is agreed upon during the committee meeting.

The resulting work product of the committee is a document, that potentially answers every question on the agenda, ready to be presented at the next multi-project meeting.

Important Committees

The following section describes an extract of some of the most important committees, that were established during one of the first weekly meetings.

- Wiki: *Ensures that the multi-project wiki page on Redmine is created in a uniform way by establishing guidelines for new articles.*
- Design Guidelines: *Ensures that the User Interface design of the GIRA application is uniform (e.g. in regards to font, color scheme and various buttons - green for 'yes' and red for 'no').*

- Common Report: *This committee is responsible for the creation of the common-report chapters, which you are reading now, which are at the beginning of every semester report.*
- Pictogram Class: *Because every group requires a common pictogram class, it was decided to create a Pictogram Class committee to determine the functionality that this class needed.*
- GIT: *The GIT committee is responsible for working out a common structure across all repositories to create uniformity and make it easier to continuously integrate.*
- Public Pictogram: *Determines guidelines for how pictograms are handled in the database (e.g. who has access rights to what and why?).*
- Story: *The story committee is responsible for creating a story to follow every sprint. It puts the sprint's tasks into an overall context.*
- CI/Git: *This committee is responsible for coming up with solutions to potential issues that might occur as part of the Continuous Integration step when using GIT.*

Chapter 3

What was developed

This chapter describes the work done for GIRAF in the year 2013 and is rounded off by acknowledging the involved contacts and the semester coordinator.

3.1 Pictograms, Morgana, and Design Guidelines

In this section the notion of a pictogram will be presented followed by how pictograms are currently being used and why they should be digitized. Furthermore the section includes a description of the Morgana library.

The section will be rounded off with the overall design guidelines for the entire GIRAF system.

3.1.1 Pictogram

In the context of this report a pictogram is defined thus: *A pictogram is an image representing a living being, a physical object or some form of action.* Pictograms can contain a text-label, describing the respective images, for clarification. There is currently no standard for the layout or contents of pictograms, due to the specific needs and opinions of the users. User **A** might like to have black and white images with text labels whereas user **B** might want colorful images without text. The images can themselves vary from cartoons to photographic representations. Pictograms are commonly used as means of communication, especially by those requiring assistance with communicating, including but not limited to individuals with autism.

Current Use

During the spring semester of 2013, when this report was written, the use of pictograms is mostly in the form of physical images. The images need to be drawn and/or edited, printed, cut out and then laminated to extend their lifespan. After this process the pictograms are ready for use, generally for one individual, making this repetitive and tedious for the guardians. When the required amount



Figure 3.1: Pictograms in use 2013

of pictograms have been created for an individual, they need to be organized and made accessible with the help of some sort of container. This container can be a folder with a pocket for the pictograms and a velcro-like strip for arranging the pictograms. For communication an individual can choose to form sentences by arranging the pictograms accordingly or use a single image to simply express needs and wants. Another purpose of the pictograms can be to graphically represent instructions for various tasks, in the form of “do **A**, followed by **B** and lastly do **C**” for individuals requiring special assistance.

Digitizing the Pictogram

The GIRAF project focuses on simplifying and digitizing a medium used by individuals with autism and their guardians. This includes digitizing the pictograms, making them available on devices running Android with added functionality. Added functionality includes the option to make the pictograms play a sound, dynamically change the layout of text-labels and editing images. Digitizing the pictogram also makes it possible to share them easily, carry them between devices and make backups of them. Previously, with the same idea in mind, it was attempted to digitize the pictogram. It was considered unsatisfactory (see section below) and therefore the re-implementation in this semester’s project.

GIRAF Pictogram Design

The digitized pictogram consists of an image, a text-label and a sound. With all elements included, it can be presented as each of the three, two parts combined or all three in union. This viewable container is designed as an extension of the *Android* view class, making it easy for developers to include and present in their applications. The idea is to have users sharing the same pictograms, with the option to customize their contents without affecting the pictogram itself. The previous GIRAF pictogram design lacked documentation, portability and functionality such as text-labels. Therefore a new design was implemented, which hopefully fits the needs of both future GIRAF developers and GIRAF users.

3.1.2 Morgana

The Morgana library project was initially intended to make it possible for all the GIRAF applications to use both the Wasteland database, see Section 3.2.7, and the local Oasis database seamlessly, however in the time allotted it was not possible to finish this functionality, so the focus was shifted to making it parse and write JavaScript Object Notation (JSON) objects for use in calls to the Wasteland database.

The library implements a Java class for each value object documented in the Wasteland Application Programming Interface (API), each class parses a JSON object and turns it into an object which can be used by GIRAF applications, it is also able to create JSON objects from the stored Java object.

3.1.3 Design Guidelines

The purpose with the guidelines is to get a consistent look and feel across all of the different applications included in the GIRAF system. The design guidelines have been discussed among all of the project groups, and they are as follows:

- Keep the existing color palette
- Font: Helvetica
- Font size: use common sense. *Android* offers extra small/small/medium/large/huge
- Minimize the use of text, use images instead of text
- Graphical User Interface (GUI) in vector graphics
- Green and red are universal colors for ‘accept’/‘cancel’
- Applications have animal icons
- Icons are non-customizable
- Every application should be locked in landscape mode

The color palette will be the same as in the 2012 version of GIRAF. With regards to font type and size, Helvetica has been chosen and developers need to keep in mind, that the text has to be readable on the tablet.

The aim is to use more images and less text as the target audience are mostly children, many of whom have communication and/or reading difficulties and some have problems imagining objects purely from text.

The GUI will be in vector graphics, because it scales well, which makes it possible to reuse some of the images. Green and red are universal colors for ‘accept’/‘cancel’. It may sound obvious but other applications have been developed with different colors. Tool-applications should have animal icons.

Lastly everything will be in landscape mode as this eliminates additional implementation for responsive layout, when the tablet is rotated.

3.2 The Project of 2013

3.2.1 Admin

This project focuses on the creation of an administration interface for the GIRAF system. The Admin system consists of two parts, one for a desktop computer and one for *Android*. The desktop part will run on a Linux, Apache2, MySql and PHP (LAMP) stack and communicate with the database using the database API provided by the Wasteland (see Section 3.2.7) project. The *Android* part will run on the tablet using the same code base as the desktop part, using a web server application. The main focus of the project is for department managers and guardians to be able to administrate the GIRAF system.

3.2.2 Cars

The aim of the Cars project is to develop an application, which will help children with infantile autism to be more comfortable in using their voice. To ensure that the children learn how to use their voice in creating different types of sounds, and not just speak in a monotone way, the application will require the children to create sounds covering different sides of the frequency spectrum.

Cars is a game in which the player has to lead a car through a street into a garage, controlling it with high or low frequency sounds. The car has a matching coloured garage at the end, which when entered completes the game successfully. Randomly placed obstacles are used to force the player to avoid them to reach the end.

3.2.3 Croc

The Croc project aims to create an application for creation of pictograms for use in the GIRAF system.

Pictograms can be created in a number of ways:

Camera take a picture with the camera and turn that picture into a pictogram.

Drawing draw a pictogram.

Audio record sounds to attach to pictograms.

3.2.4 Parrot

Parrot is an enhancement of the Parrot project of 2012 and is an application for communication between guardian and child. Its development is based around the currently used physical system Section 3.1.1. The original Parrot application from 2012 also included the administration of categories. It was therefore technically possible for a child using Parrot to access these administration tools, and it is for this reason, that the currently developed version has relocated the administration to a separate application named Category Administration Tool. The version developed during this project will focus on making improvements to the GUI design, adding subcategories (such as breakfast item under the food category) and handle the interaction with pictograms. The primary focus for Parrot remains the same; providing an easier way for children to communicate with a guardian in a way that they are familiar with.

Category Administration Tool

Category Administration Tool (CAT) focuses on administrating categories and subcategories. Currently CAT is also responsible for communicating with other applications that need specific pictograms, such as the Tortoise (Section 3.2.5) and Zebra (Section 3.2.8) applications, by providing search/deliver functionality.

3.2.5 Tortoise

The Tortoise application focuses on helping children to learn about their own lives and strengthen their social skills. The hope is, that by letting the child interact with pictures and sentences, which are associated with their life, the child can develop an identity. By developing their own identity, the child will learn how to interact with other people by learning what kind of topics to talk about in a conversation with others.

3.2.6 Train

The inspiration for Train comes from an exercise, which one of the guardians practices with the children. The purpose of the game is to create a dialogue between the child and the guardian. The child has to drag pictograms from a train station onto the train wagons and make the train drive. When the train arrives at the next station, the child has to drag the correct pictograms from the train and onto the station. The correct pictograms are decided by the station category.

The category for each station is chosen by the guardians by clicking the category picture frame and browsing CAT (Section 3.2.4) for the picture they want to use. After selecting a category, they select which pictures they want associated with the station.

3.2.7 Wasteland

The purpose of the Wasteland project is to handle all of the data for the GIRAF system. In order to achieve this goal, a database will be implemented on a central server and a local database will be kept on the tablet. The two databases will synchronize data on a regular basis.

3.2.8 Zebra

The aim of the Zebra project is to create a software application aiding guardians in their work. The application should aid the guardian in situations where a child is to perform an ordered sequence of actions. These actions are typically represented by pictograms. Zebra should replace the current paper based version of this system. The guardian should be able to create and manage digital versions of such sequences specific to each child. Upon selecting a sequence for the child to follow, the child should be able to mark actions as done when they are completed to illustrate their progress.

3.3 Acknowledgement

The group of students working with GIRAF during the spring semester of 2013, would like to thank the contacts, who were;

Tove Søby - speech therapist, and contact for three groups.

Mette Als Andreasen - kindergarten teacher at Birken Langholt, and contact for two groups.

Kristine Niss Henriksen - kindergarten teacher at Birken Vodskov, and contact for one group.

Drazenko Banjak - teacher at Egebakken Vodskov, and contact for one group.

Mette Frost - teacher at Egebakken Vodskov, and contact for one group.

In addition the group would like to thank Ulrik Nyman, semester coordinator, for his help, guidance and engagement during the project.

Part II

Analysis

Chapter 4

System Design

As explained in section 3.2.1 on page 34, the Admin system is based on a single code base running on a LAMP stack, which means we use Apache, MySQL and PHP for the system to operate. However the MySQL part is handled by another group, the Wasteland group. The Admin system simply interface with the database through the API that the Wasteland group makes. A description of the WASTELAND project can be read in section 3.2.7 on page 36.

Given that the system is written in PHP, the product of this project will be a website. Which has the ability to manage users, pictograms and applications.

This chapter will give an insight into the initial choices made by the Admin group at the beginning of the project. Choices such as, starting the project from scratch for the 3rd time, how to run PHP code on a Android device, and how we organize the source code.

4.1 Starting from scratch

Though there have been 3 GIRAФ administration projects before this one, the decision to create a completely new one came fairly easy.

The choice was made based on several problems with the two admin projects from the GIRAФ project year 2012. The projects was named Oasis and Savannah.

Oasis was a system mainly concerned with constructing a database for the tablet, and every application run on the tablet. The administration part of this project did therefore not get far. It could more or less only display the information about users in the local database which it was constructed with.

The Savannah admin interface came a little further than Oasis. However they too were also concerned with constructing a database, this time for the website version. It was supposed to be able to match that of the Oasis project. However

this was not the case when the two projects was done. The database schemes did not match and were not able to synchronize. This lead to the start of the Wasteland project.

The Wasteland project was supposed to have been mainly about the synchronization of the two databases, but ended up having to start the database project from scratch because the available code from the Savannah project was outdated and that of Oasis was unusable as a central database setup.

By the same two reasons the Admin project could not be continued from the Savannah or the Oasis project. If we had chosen to continue the Savannah and the Oasis project, we would first of all have to get the Savannah code back up to date and then make both the Savannah as well as the Oasis project complete, while making both ready for a new database implementation. This would also mean that it would be more difficult then necessary to maintain the Admin system when the project was finished, since this would result in two entirely different code bases, one in Java and another in Java Servlet.

All these reasons made the decision to start this project from scratch easy. Developing the system in PHP would make it possible to keep a single code base. And the time it would have taken to take Savannah and Oasis up to a state where they could both be worked on properly, both with a new DB and with their original features, would take at least the same time as it would to rewrite the system in a single code base.

The next problem which this presents is that, PHP code is not normally something Android systems can interpret on their own. How we intend to solve this problem is what the next section is about.

4.2 PHP on Android

PAW stands for Pro Active Webfilter and is an application that makes it possible to run a website, written in PHP code on an Android device. This makes it possible, in combination with a web-browser for the user to use the Admin system without being online, which can be the case for the target group to use the GIRAF system.

However we did not have the chance to implement this ourselves, since halfway through the project our tablet went haywire and was not fixed before the end of the project. This also meant that nearly nothing was finished in regard of this feature. Instead we focused on finishing the code base.

4.3 Folder structure and other practical issues

In this section we briefly explain the folder structure as well as why we chose to implement language support and use PHP and JavaScript as the coding languages.

4.3.1 Folder Structure

When designing a system which is supposed to come with a high maintainability it is important to design a usable folder structure. The folder structure of the admin project can be seen in figure 4.1.

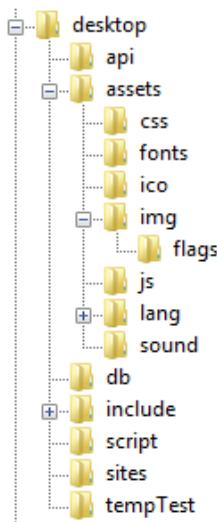


Figure 4.1: The Code Folder Structure

desktop: This is the main directory for the desktop version. The reason for not having a single root directory is that there might be some variations between the desktop and the Android version of the system.

api: Is intended to contain API which are not JavaScript or PHP code. And is empty as of the end of this project.

assets: Contains every asset in the system. Images, JavaScript, sound, fonts, CSS and language files.

assets - css: Contains the CSS files for the system.

assets - fonts: Contains special fonts for the system.

assets - ico: Contains the icon files for the system.

assets - img: Contains the static image files for the system. Pictograms and Profile Images are loaded from the database.

assets - img - flags: Contains images of flags, used for changing language.

assets - js: Contains all JavaScript in the system, files are named as they fit to PHP files, as far as this is possible.

assets - lang: Contains all language files to the system. Each webpage and some JavaScripts, have their own language sub-folder . Right now they contain English and Danish language files.

assets - sound: Contains all static sound files to the system. Sounds for Pictograms are loaded from the database.

db: Contains the database files. All active database functions are stored in the file “new.db.php”

include: Contains any JavaScript or PHP code that is not developed by the Admin project, that is included in the system.

script: Contains PHP scripts which do not display anything, but rather work to execute some functions such as changing a profile image.

sites: Contains .html and .php files which correspond to a webpage on the website.

tempTest: A folder for any need of generating files during development.

4.3.2 PHP and JavaScript

As already mentioned in this chapter we chose to use PHP as the back-end language. And in order to make some more advanced user interface we decided to use JavaScript, HTML and CSS as front-end languages.

We chose to use PHP because most of the Admin group already had experience with the language, and also because we found PAW which would make it possible to execute the PHP code on the tablet.

Also another bonus, which was not considered much when we made the choice, is that PHP is natively able to work with JSON objects which is how we communicate with the database API.

JavaScript, HTML and CSS were chosen as front-end because of the product being a website, for which there is no real alternative.

4.3.3 Language Support

It was in the beginning of the project suggested that the system could be operated in multiple languages, as a start in Danish and English. This was suggested by the semester coordinator. The admin group thought, if this was to be imple-

mented at any point, it should be in the beginning of the system, which meant we would have to implement it into our system.

The way it is implemented is much like how it is in Android applications, and can be read in further details in 9.4 on page 70.

The reader should now have an understanding of the most basic decisions of the Admin project, and be able to navigate in our system.

The next chapter will go through the project progress, followed by the project's problem statement.

Chapter 5

Project Progress

This chapter concerns the work progress of the project and the documentation here of. The work progress was split into weekly progress which was documented with a weekly abstract which is to be read in conjunction with sprint story. At the end of this chapter the reader should have the knowledge of the development structure which is used and have the overview of the work that was done in this project.

5.1 Weekly Abstracts

Weekly abstracts are a small resume on what have been developed over a week and what problems occurred during that time. Usually the weekly abstract is between 5-10 lines of text which is written at the end of each week. Writing weekly abstracts is an easy way to ensure that the development move forward because if there is nothing to write, nothing was made and one should reflect upon that. Also newcomers or an involved third person would be able to quickly get updated on project status by reading the abstracts.

5.2 Weekly Progress

In the rest of the section is a description of the weekly progress in conjunction with sprint story. The conclusion of the work progress will be at the end of this chapter.

The first two weeks of the project were spent to create the groups and determine which projects should be worked on this year. Internal in the group the expectations were aligned and work conditions were established. Graphic figures, including logo, and design guidelines were also established.

Sprint stories were first used in week 3 (04/03/2013-08/03/2013).

Week 1 (18/02/2013-22/02/2013) Designs and features for the Admin interface were discussed. Low-fi board designs were made to give abstract view

of our ideas regarding the meeting with our contact person. The first considerations to incorporation of the multi language support. The structure of the PHP code, the project and the report had been made so it was ready for the first sprint. Setup GIT and Redmine. It was agreed upon to make a web based administration system which is convertible to both desktop as well and tablet. *For further information this choice is readable in chapter 4 on page 39.*

Week 2 (25/02/2013-01/03/2013) The focus has been on finishing the preliminary interface design and web application structure. All of the interface designs which were drawn on whiteboards were converted into Balsamiq mock-ups for a coherent look and feel. At the meeting with the contact person there was feedback on the design and the features. Work in committees started. *The feedback, questions and answers regarding how the day care works can be found in appendix C on page 115. The Balsamiq mock-ups and web application structure is found in appendix B on page 99.*

Week 3 (04/03/2013-08/03/2013) This weeks work has been on implementing the design and the functionality of the login screen and main navigation. We have also edited our mockup designs with the feedback we got from our contact person. Bootstrap is included to streamline the design.

The first SCRUM sprint started with the main story being:

"The guardian is in the launcher and starts all the applications one after another, the guardian can freely move from application to launcher at any given time. The guardian also enters the administration."

which lasted from 04/03/2013-18/03/2013.

From that story we concluded the demand for our sprint was for a user to log in to the system and view his/her profile page.

Week 4 (11/03/2013-15/03/2013) The first scrum sprint is completed. Following sites are somehow functional implemented in the Admin system:

- Log In/Out
- Own Profile
- View own information
- Edit own information

The site can still not be shown outside the campus network, and therefore can it not be shown to the contact person. Regarding functionality the site is not connected to the final database yet but all the test data are there. The first parts of the common report are created.

Week 5 (18/03/2013-22/03/2013) Most progress was in implementing the design and edit the mock ups. The committee for the DB-API is started. This committee is important as it will define connection to the database.

The new sprint is: "The guardian is in a application and is working with a picture"

The focus is therefore on implementing profile pictures.

Week 6 (25/03/2013-29/03/2013) Time was short because of Easter. Following features were integrated the Profile Picture Edit and ability to Change

QR. Mock-ups of Profiles, Create Profile and Department Management were also worked on.

Week 7 (02/04/2013-05/04/2013) This week there were only were 4-8 hours of free work time, because of the many lectures and the holiday of this week.

The new sprint is as follows: “The guardian creates a pictogram, and imports the pictogram to an application. The guardian personalizes the application.”

Week 8 (08/04/2013-12/04/2013) This week an auto update script that fetches the git-repository for our web-server was made. Uploading the profile picture was reworked into a faster and more data efficient way. But still missing the database to store the data and to give a report when the image fails to upload or succeeds. The QR generators is now fully functional. There is now a printer function to send our QR-images to the browser or the OS printing service.

Week 9 (15/04/2013-19/04/2013) The QR system is changed, so that it is secure. Language support now works for all the developed sites.

Week 10 (22/04/2013-26/04/2013) Uploading of profile picture is finished. The work is now on getting the Admin interface down to the tablet. We have scheduled a meeting with Mette Als (our contact person), to test our system.

New sprint: “The guardian can navigate between applications and use them”
Week 11 (29/04/2013-03/05/2013) This week was spent on solving the whole problem with the missing database. A front end was created for the Picto Admin. Features in our system cannot be supported in IE9 and below. Therefore a warning were added at the log-in page that informs the user that they are using an old browser if the browser does not support the “FileReader” system in JavaScript.

Week 12 (06/05/2013-10/05/2013) The week was spent to preparing for the usability test(Monday 13th). Profile Create and Make Relations were finished, with exception of the DB create implementation. Also Picto Admin Create is fully finished. But it does not support categories. The weekend between the 10th and the 13th was used to ensure that the system could work with the newly developed database API. The system was not ready in time for the usability tests.

Week 13 (13/05/2013-17/05/2013) The tests were completed and analysed. All the bugs and the wanted changes were documented and set to be fixed. Most of the week was spent on implementing the DB API in the system and making fixes. The report structure was made and writing of the report has been started. *The finished functionality can be found in chapter 7 on page 53.*

At the end of the project the program was tuned to be fully functional and include notes. Until the report delivery was writing documentation.

5.3 Conclusion of Project Progress

The weekly abstracts were a good point to acknowledge the progress and evaluate the work effort. It was also used to inform our supervisor of the progress we had made during the weeks. The group work process was more a natural development process forced by the sprint story. The one thing that did complicated the work was the lack of cooperation with Wasteland which should have been much closer.

Chapter 6

Problem Statement

The general purpose for this project has been explained both in this reports introduction and in the GIRAF introduction, see the Introduction on page 15 and 3.2.1.

The first GIRAF Admin project was started in 2011, then in 2012 it was split into two projects, called Savannah and Oasis. The two last named projects were focused on constructing database systems as well as user interfaces to support it. Savannah was implemented as a web version and Oasis as a tablet version. When the two projects were finished, the databases could not synchronise and was at this project's start ultimately scrapped due to incompatibility and low maintainability.

In 2013 the projects were split into constructing an administration interface and constructing a proper synchronisable database. The database project this year was named Wasteland [8].

Since Wasteland chose to redesign the database in order to accommodate the use on several platforms and synchronisation, it was a logical choice to also redesign the administration interface, seeing that a database design would render previous projects work useless. This also accommodated a way to heighten maintainability and user friendliness. This will hopefully ensure that this administration system will not be discarded next year.

In order to fulfil the general need of user creation in the GIRAF system, the focus will have to be on creating all the profile management tools first. This means that if some of the features cannot be implemented in time, as explained in 7 on page 53, then the focus should be on the user management interface. This will leave all management of pictograms to the tablet projects.

This has lead to the following problem statement:

“Currently there are two different administration interfaces for the GIRAF system. This results in a problem with maintainability and user friendliness. How can we make a single user friendly administration interface for the GIRAF system?”

Part III

Solution

Chapter 7

System Overview

In this chapter we give an overview of the features which the system contains, their status and why they are necessary.

Below is shown a table containing each feature which we have designed. As can be seen in table 7.1 we have designed a few features which we think is necessary for the online PC system. It can also be seen that we only managed to finish 3 out of 13 features and 5 out of 13 are only in need of DB implementation. That leaves 5 out of 13 features unstarted, where 4 of these are related to the Pics Manager.

Feature Name	Status
Login	Done
My Profile	Needing DB
Profiles	Needing DB
Create Profile	Done
Add Relation	Needing DB
Pics Manager - Make	Done
Pics Manager - Add	Unstarted
Pics Manager - Remove	Unstarted
Pics Manager - Edit	Unstarted
Pics Manager - Delete	Unstarted
Dep. Information	Needing DB
QR Manager	Needing DB
App Manager	Unstarted

Table 7.1: Feature Table - The status explains if they have been finished or not

As it also can be seen in the table we have focused on implementing features which makes it possible to handle Profile related issues. This have been done because it is already possible to handle pictograms from the tablet systems. We did implement one of the Pics Manager features, Make, this is due to it not being system wise directly connect to the remaining 4. The 4 remaining

features of the Pics Manager are all based on the same search and select system and should therefore not prove difficult to implement at a later point. More can be read about this in chapter 12 on page 89.

7.1 Login

Status: Done

Resume: This feature, makes it possible to select the desired language, and log in to the GIRAF Admin system. This is also where rights is discovered and stored in a session, which will be explained in further detail in section 9.6 on page 75.

7.2 My Profile

Status: Needing DB

Resume: This feature displays the logged in users information as well as the users relations and makes it possible to change these information. My Profile also has a sub feature to change the Profile Image of a user, which will be explained in further detail in section 9.3 on page 67.

7.3 Profiles

Status: Needing DB

Resume: This feature is an admin feature¹, it displays all profiles in a department which have been related to a pedagog in some way. It also makes it possible to navigate to these profiles.

7.4 Create Profile

Status: Needing DB

Resume: This feature is an admin feature, it makes it possible to create new profiles for a department. But is not able to give admin rights to a user. Admin rights can only be given directly from the Database.

7.5 Add Relation

Status: Needing DB

Resume: This feature is an admin feature, it makes it possible to form relations between children and pedagogs, as well as children and parents.

¹This means that only a designated admin can use this feature.

7.6 Pics Manager - Make

Status: Done

Resume: This feature makes it possible to create pictograms.

7.7 Pics Manager - Add

Status: Unstarted

Resume: This feature makes it possible to link a pictogram to one or more profiles.

7.8 Pics Manager - Remove

Status: Unstarted

Resume: This feature makes it possible to remove a link to a pictogram from one or more profiles.

7.9 Pics Manager - Edit

Status: Unstarted

Resume: This feature makes it possible to edit a pictogram which have already been created in the database.

7.10 Pics Manager - Delete

Status: Unstarted

Resume: This feature makes it possible to remove a link to a pictogram from a user, and when there are no more links to a pictogram it will be permanently deleted.

7.11 Dep. Information

Status: Needing DB

Resume: This feature is an admin feature and makes it possible to edit the information about a department.

7.12 QR Manager

Status: Needing DB

Resume: This feature is an admin feature and makes it possible to change the QR code for a user, which will be explained in further detail in section 9.2 on page 66.

7.13 App Manager

Status: Unstarted

Resume: This feature is for adding or removing the rights to use an application for a child, and it is the only feature still in the design step.

This chapter should now have given a proper overview of the systems features, and it should be clear what the system is intended for and how. The next chapter will explain how we designed the system, followed by a deeper explanation of some of the more advanced features in the system.

Chapter 8

Design

This chapter will describe the design process of the administration system. Seeing as the work from previous years will not continue in this project the opportunity to make a new design philosophy rose. In 2012 the Launcher group made a design guide. This guide was extended in the Design committee 3.1.3 and this project will follow that guide. Basically it says that one should follow the colour theme, which can be found in appendix E, and use vector graphics as much as possible.

Looking at other web administration interfaces like WordPress [15] the general idea of having a navigation bar at the left side of the screen and the content for a given menu on the right side as figure 8.1 shows. This idea also came from the Android systems settings app, which looks like WordPress' admin interface but include a more neutral way of displaying categories. The login page, which is the only page that does not follow the general idea, is inspired directly from Twitter's Bootstrap [13] sign in layout.

Every menu-item went through the process of using a whiteboard as a screen and then draw the design by hand. For better preservation, the hand-drawn mockups were created digitally in Balsamiq Mockups [3]. Next at a meeting showed to our contact person and changed to reflect the issues risen by her. Minutes of the meeting can be found in appendix C. The complete set of final mockups are available in appendix B.

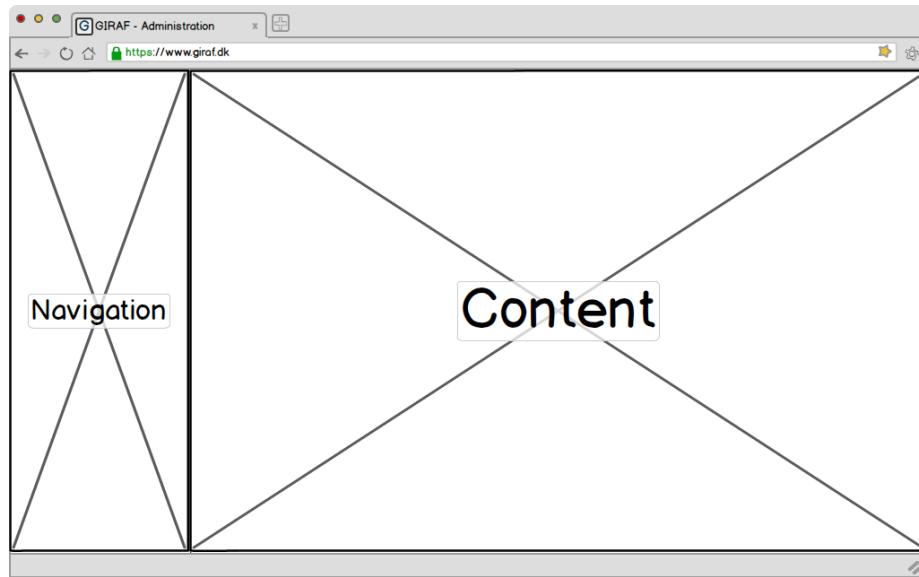


Figure 8.1: General idea of building web-pages.

8.1 Designing individual pages

8.1.1 Login

The Login page is designed to be as minimal as possible. There should be no cluttered information and with a single exception no other option than to login. The exception is to change language. The mockup is viewable in figure 8.2.

8.1.2 Navigation

Navigation was one of the design items that went through a lot of small changes. The general idea was that it should look and feel natural to an Android user. Meaning that the navigation bar conforms to a principle about accessibility so that there are no foldout points or other elements that would be considered difficult to go to on a touch screen. On the mockups each menu category has a little image next to it, this is changed so that now the first letter is capitalized and uses a bigger font than the rest of the text. If a user with administrator rights login he will be able to see all menu items but if a user with degraded rights logs in he will only see Own Profile, Pics Manager and App Manager further information about department as an example is then accessible through own profile without editing rights. A mockup showing the navigation and the profile page is figure 8.3.

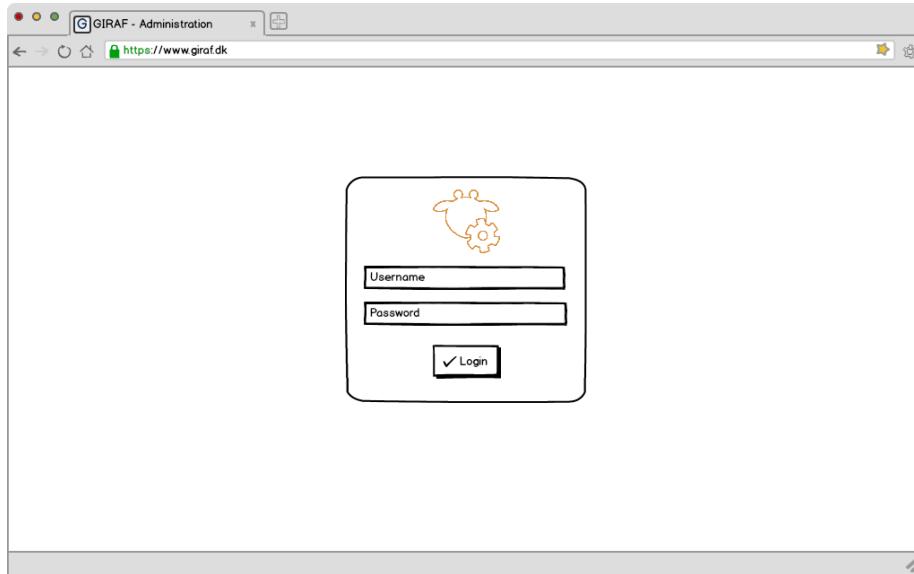


Figure 8.2: Login

A screenshot of the GIRAF Administration interface. On the left is a vertical navigation menu with icons and labels: Egen Profil, Profiler, Tilstøj relation, Pics Manager (with sub-options Opret, Tidel, Fjern, Edit, Slet), Dep. Manager (with sub-options Dep. Information, QR Manager), App Manager, Log ud, and a logo at the bottom. The main content area has a title "Egen Profil". It displays personal information: Navn: Hans Hansen, Tlf: 48768598, Mobil: 32980298, Adresse: Bredgade 1, 9000 Aalborg, Afdeling: Bjælken. To the right is a placeholder for a user profile picture with an "Ændre" button below it. Below the profile area is a table titled "Børn" showing three children: Kim Larsen (6), Jette Berg (14), and Marianne Holm (12). To the right of the table is a QR code with a "Print QR Kode" button below it.

Figure 8.3: Navigation and Profile page

8.1.3 Profile Page

The profile page is the page that is shown when a user logs in. It should have information about the user as well as information about linked profiles such as attached children, parents or pedagogues. A user should also be able to edit his own information as well as attached children information. As a result of our usability test, which can be seen in chapter 10, the current design used, does not include the ability to change ones QR-code. Instead only the designated admin has that ability. A mockup showing the navigation and the profile page before usability testing is figure 8.3.

8.1.4 Profiles

It gives the administrator a complete overview of how users are linked. This sometimes gives a better overview and therefore it is designed so that all links between pedagogues, children and their parents are displayed in an easy to comprehend way. To do this we agreed upon a single table approach which gracefully full-fills the comprehension wanted. This is even more underpinned when colour coding is applied to guide the admin. This idea was given to us by our contact person.

Create Profile

The title say everything. The admins are able to create other profiles. The design consists of a number of input fields as well as the option to select which type of user one wants to create.

Add relation

Here a privileged user should be able to create relations or links between other profiles. A scenario would be that a child profile has just been created and it should now be linked to its already created parents. This procedure should take place here.

8.1.5 Pics Manager

Pics Manager is based upon having the capability to create, add, remove, edit and delete pictograms which all are designed to the same principles of easy accessibility as the rest of the system. Pics Manager should not be accessible on tablets. This is mostly due to the fact that separate applications have been developed for its primary purpose. It should instead open these applications and the user should not be bothered by this. The different components design should have the same look and feel as the Android application but take advantage of the fact that they are run on a desktop computer and not a tablet.

Create

Originally named Make this tool should supply the user with the capability to create pictograms in the database.

Add

Add pictograms to users.

Remove

Remove pictograms from users.

Edit

Enables the user to edit pictograms which the user has available.

Delete

If a user own a pictogram he can delete it permanently from the database.

8.1.6 Department Manager

Enables a privileged user to edit department information and view a short list of attached pedagogues.

Department Information

Essentially the same as Department Manager but displays the information as an unprivileged user would see it. An unprivileged user accesses the page through a link on his profile page.

QR Manager

Enables a privileged user to change users QR-codes. This is also one of the pages which have gone through a number of design iterations, as viewable in appendix B it originally had three sub items but was refined to a single page with a much more comprehensive layout. This enables the user to complete a given task much more easily. A screenshot of the current layout is figure 8.4

8.2 App. Manager

The app manager, has only gone through the initial design step. This means that the only thing considered was what it was supposed to represent. The app manager should make it possible to enable or disable the use of certain apps within the GIRAF system, as well as look up description of these.



Figure 8.4: Current design of QR manager

Chapter 9

Implementation

In this chapter the most complex features will be explained in details. This chapter also serves as a guide for anyone wanting to continue the work on the Admin project.

9.1 Navigation

The navigation system which is implemented in the system is based on JavaScript and the event `window.onhashchange`, with the addition of Ajax. Ajax is used to fetch the sites without ever navigating away from the index site. This is done to minimize the amount of data transfer from the server. However this does present itself with a few challenges. In order to handle this, a special solution had to be made for sending the PHP `$_POST` and `$_GET` data around. In this section it will be explained how this is implemented and how it should be used in the future.

There are at least two other solutions that could have been implemented with different advantages. The old HTML method could be implemented, which means that the user would have to navigate directly to the file names. This would be an easy way to build the website, but would leave it very hard to maintain, since the design of the website would have to be written more than one place, or at least use an include at the top and bottom of each page. Another alternative would have been the use of a PHP based switch. The method is commonly used in larger website systems, because it makes for an easy design and heightens maintainability.

Listing 9.1: A PHP switch Example

```
1 $site = $_GET['site'];
2 switch ($site) {
3     case "ownProfile":
4         include "sites/ownProfile.php";
```

```

5      break;
6  case "picsMake":
7      include "sites/picsMake.php";
8      break;
9  case "":
10 default:
11      include "sites/home.php";
12      break;
13 }

```

As seen in listing 9.1 it is easy to create the switch method, and maintain it, since all what is needed is to add another case when a new site is made. What listing 9.1 contains has to be included in the index.php file, and the links, or hyperlinks, will have to set the `$_GET` variable `site` to a fitting name for the site, and it will then include the content of that site.

However this PHP switch solution has one downside to it. It still sends the data of the index file from the server to the user each time the user presses a link. This might not be much data, but it becomes so in the long run. We therefore went on with a third alternative, which is much like the PHP switch method. We simply took the same idea and wrote it in JavaScript, this however means that we must use Ajax to perform the fetching of new data.

Listing 9.2: The JavaScript switch

```

1  switch(destination)
2  {
3      case "":
4      case "#ownProfile":
5      case "#otherProfiles":
6          destinationPath = "sites/own_profile.php";
7          break;
8
9      case "#profiles":
10         destinationPath = "sites/profiles.php";
11         break;
12
13     case "#profilePicUpload":
14         destinationPath = "script/profilePicUpload.php";
15         break;
16         ...
17     }
18     ...
19     $.ajax({
20         type: "POST",
21         url: destinationPath,
22         data: postData,

```

```

23     success: function(result) { // result is the content ←
24         // that the php file 'ECHO's.
25         $("#content").html(result);
26     }
26 );

```

Listing 9.2 contains the main contents of the JavaScript switch that we created. `destination` is the variable which we use for storing the actual hash value¹ and the `postData` is created in a unique way, so that we can send the PHP `$_POST` data to the site which the JavaScript is Ajax'ing to.

Listing 9.3: The POST transform code

```

1 echo "<script>
2     var postData = "";
3     echo json_encode($_POST);
4 echo "</script>";

```

As seen in listing 9.3 we convert the `$_POST` data from PHP into a JavaScript variable that then again is send on to the next PHP site as seen in listing 9.2.

However we also wanted to be able to use the `$_GET` variable from PHP on other sites that we call with Ajax. In order to do this we created a rule as can be seen in listing 9.4. The rule says that instead of using the usual syntax of "?" after the hyperlink, there needs to be a "/" instead. We do this because we think it gives a better look on the link itself.

And the PHP site called with Ajax would not be able to read the `$_GET` variables without the script parsing it through the Ajax call.

Listing 9.4: The GET transform code

```

1     var hashInfo = location.hash;
2     var hashArray = hashInfo.split("/");
3     // We use / instead ←
4     // of ? in our URL's (for $_GET), they do the exact same,←
5     // but gives a different look
6     var destination = hashArray[0];
7     var info = hashArray[1];
8     var destinationPath = "";

```

Then when the code in listing 9.4 and the switch in 9.2 has been executed we append `info` to `destinationPath` with the normal syntax. Then the Ajax call does the rest.

But this method also has a bad side. It requires a more complex way of calling scripts which is dependent on large amounts of data from the user. For example

¹Hash value is what JavaScript interprets as the value followed by the hash symbol # in hyperlinks.

when the user wish to upload an image for a pictogram.

As seen in listing 9.5 which is a cutout of the `headInclude.php` file, which is always included in our `index.php` file, we include the script directly into the index file instead of using our special Ajax JavaScript function. If we did not do this, the user would have to send the data to the server twice. First in order to send it to the index file, then the user will receive it again, and then call the Ajax function with the same data.

Listing 9.5: The handling of big PHP POSTs

```

1 if(isset($_POST['picsManagerMakeSubmit'])) {
2     //Call upload script
3     require "script/picsManagerMakeUpload.php";
4 }
```

And then the upload script must always make sure to use a `PHP header` call to navigate the user to the right site, depending on whether he got an error or not.

If the reader wants to learn more about the navigation script the files used for this script is: `/include/headInclude.php` , `/assets/js/navigation.js` and `index.php` .

9.2 QR Code Generation and Printing

The multi-group last year decided that QR codes would be used as the only login method on tablets, which meant that the GIRAF administration system would need a system for generating and printing new QR codes, in the event a person lost their QR code. In collaboration with Wasteland it was decided that it would be a security risk to allow people to print out existing QR codes, which meant that our system only has to facilitate the generation of new QR codes and the printing of these new QR codes. The last multi-group also decided that the QR codes would be based on a 512 character long string. The system generates the new QR codes using the code displayed in Listing 9.6. The QR code generation uses the function `microtime` in order to get data to hash. The function `microtime` relies on the system call `gettimeofday`, which means that this implementation will only work on UNIX and Windows based systems (PHP supplies its own implementation in Windows). The time from `microtime` is then hashed with the `sha512` hashing function which generates a 128 hexadecimal long string. This is then repeated 3 more times in order to get a 512 hexadecimal long string, which is then the newly generated QR code.

Listing 9.6: QR Code Generation

```
1 function generateNewQr()
```

```

2  {
3      $qr = "";
4      for ($i=0; $i < 4; $i++) {
5          $time = microtime();
6          $qr .= hash("sha512", $time);
7          usleep(100); // sleep for 100 microseconds (0.1 ←
8          milliseconds) to get a different time from microtime
9      }
10     return $qr;
11 }
```

The newly generated QR code is then inserted into the GIRAF database using the Wasteland database API. Now the user is prompted to print out the new QR code, as this is the only chance the user has, without generating yet another new QR code. The PHP library `phpqrcode`[6] is used to generate the QR code itself into an image. In order to support the scalability of the QR code, the image is generated as a SVG, but because the original implementation of `phpqrcode` does not support SVG output, a modification of `phpqrcode` is used[2] which adds support for SVG and EPS. Due to a bug with Internet Explorer, `phpqrcode` was further modified so that the colour of the QR code is statically black. The bug with Internet Explorer was that `phpqrcode` would truncate the hex code of the colour black to #0 which in Internet Explorer would display as white, while in other browsers it would display as black.

After `phpqrcode` has generated the SVG, it is then added to a hidden iframe, which only contains the SVG and a separate CSS file which contains the style for printing the QR code. JavaScript is then used to open the printing dialogue of the browser with the iframe as its focus, so that only the QR code is printed and not the whole page.

9.3 Profile Picture Change

It can normally be done with a simple uploading of a file and transforming it, into the right size with the build in PHP functions.

But in our system it is a bit more complex. Both because we store the images in the Database, so that they can be retrieved on the tablet as well as our PC version. And also because of the before mentioned way we move data around, see section 9.1.

As again, mentioned in one of the former sections, see section 9.1, we have made our system this complex in order to minimize the data transferal from the server. This forces us to make some complex solutions to anything that requires big data transferals.

As for the database, it is generally bad practice to store images in a database, because of the encoding which is needed to do so. But exactly because we want to be able to use the image both on the tablet as well as on the website, this is actually a good solution. There might have been some alternatives to this, but

this is not a part of our project. For more information on this the reader should read the Wasteland project report, which is also a project under the GIRAF project.

As explained at the end of section 9.1, we have made it so that the upload script is called on submit, from the index files, see listing 9.5.

In order to submit the user must go to the profile page, press the "Edit" button underneath the display of the profile picture, they will then be met with a modal window² where they are asked to chose a file.

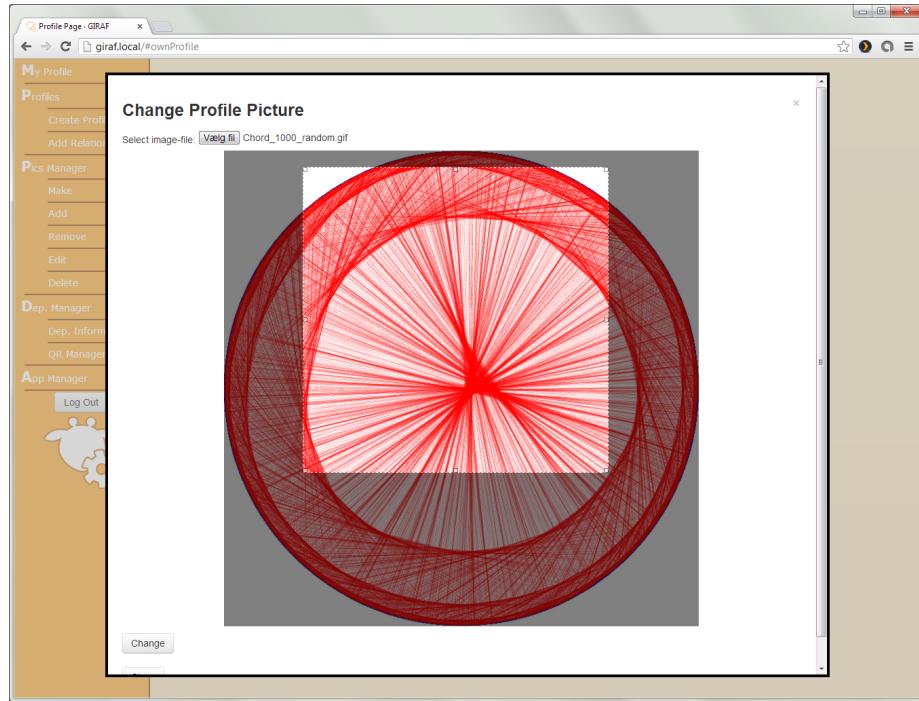


Figure 9.1: Changing a profile picture

When that file is chosen they have to crop the image, as shown on figure 9.1, and finish by pressing the "Change" button. The cropping here is made with the help of an open source project called imgAreaSelect [14].

When this is done, the user will be send back to the profile site, with a success or error message and also be able to see the new profile image on the profile.

But what the user does not see is that on submit they are actually navigated to our profilePicUpload script, which then navigates them back with the error or success message in the URL.

²A modal window is a sort of pop-up window on a website.

This is the script which handles the encoding and the database query for updating the profile picture.

When the profile picture is uploaded the script first checks for any possible upload errors, as well as making sure that the file is actually an image and it was send from our form. This is to prevent security issues. Then when that is handled, the script will load the image into PHP and will transform it by the help of a specialized cropping tool inspired by a PHP project called SimpleImage [9].

```

1      function resizeCordsColor($width,$height,$x1,$y1,$x2,$y2,←
2          $red,$green,$blue) {
3          $new_image = imagecreatetruecolor($width, $height);
4          $white_image = imagecolorallocate($new_image, $red, ←
5              $green, $blue); //Change image color to: White (←
6              RGB format)
7          imagefill($new_image,0,0,$white_image);
8          imagecopyresampled($new_image, $this->image, 0, 0, $x1←
9              , $y1, $width, $height, $x2-$x1, $y2-$y1);
10         $this->image = $new_image;
11     }

```

Figure 9.2: The function used for cropping a profile image.

The function seen in listing 9.2 is what we use for reshaping the image into a more manageable size. What it does is first to create a new fully white image (L. 2-4), because it is called with the RGB value 255,255,255. Then it proceeds to copy the selected area, which the user selected before onto this white image (L. 5), in a 300 x 300 pixel area, which is the size we decided a profile picture always will be. Finally it stores itself in its own class (L. 7).

As for implementing the database query, we have actually designed it and according to the Wasteland API [7] it should work. But when we try to update the image it only returns an SQL error in the console. We have asked the Wasteland group why this is so, but they are as clueless as we are. So when the Wasteland group manages to find the error, this feature will be fully implemented.

This concludes the explanation of the Profile Picture Changer, if the reader wants to learn more about the Profile Picture Changer the files used for this feature is: `/include/SimpleImage.php` , `/assets/js/profileEdit.js` and `script/profilePicUpload.php` .

9.4 Multi-language Support

Android has simple multi-language support, which means that in order for GIRAF Admin to stay consistent with the GIRAF suite of apps, it should also support multiple languages easily. In order to make it as easy for developers to add support for more languages in GIRAF Admin a simple language system was developed. The system consists of separate PHP files for each sub-site and language. Each sub-site has a sub-folder in `assets/lang` in which all of the language files should be. Each language has a separate file with the format `sub-site.language.php`. These files then contain an associative array of strings which contains all of the static strings of the given sub-site. The language of the site is chosen when the user first logs in. Each language currently has a flag below the login box on the login site, and so to change language the user just has to click on the flag of the given language. The chosen language is set in the users PHP session, and so all sub-sites has to do is check the session variable and then include the language file for that language. This can be read in detail in section 9.6 on page 75.

9.4.1 Adding a new language

Described here is the procedure for adding new languages into the multi-language system

1. For each folder(sub-site) in `assets/lang` a new file should be created with the format `sub-site.language.fileext` where fileext is either `php` or `js` depending on if the language file is for PHP or Javascript.
2. Copy all of the variables from an existing language file into the new language file and translate all of the strings into the new language
3. In `login.php` add the new flag for the new language and give it a link to `login.php?lang=language` where language is the ISO 3166-1 alpha-2 code of the given languages country (although `en` is used to designate English). When the number of flags increase to a number that can no longer be shown visually pleasing, the selection should switch an alternative selecting method more suited for large amounts of languages
4. For each sub-site's PHP file (`sub-site.php`) add the new language to the switch statement and include the language files for the new language for that given sub-site

9.4.2 Adding a new sub-site

Described here is the procedure for adding a new sub-site to the multi-language system

1. On the new sub-site locate all of the static strings, and replace them with references to an associative array like this `$SUB-SITE_STRINGS['STATIC-STRING-NAME']`
2. Add code to get the current language from session and a switch case statement to include the language files at the top of the PHP file in the format shown in Figure 9.3.
3. Create a folder for the new sub-site in `assets/lang`
4. Create a new file for each of the currently supported languages in the format `sub-site.language.php`.
5. In each of the new language files create the associative array referenced earlier and add all of the strings with their translation

```

1  session_start();
2  if (isset($_SESSION['lang'])) { $lang = $_SESSION['lang']; } ←
   else {$lang = 'en';}
3
4  switch ($lang) {
5    case 'en':
6      include($_SERVER['DOCUMENT_ROOT'].'/assets/lang/sub-site←
         /sub-site.en.php');
7      break;
8    case 'dk':
9      include($_SERVER['DOCUMENT_ROOT'].'/assets/lang/sub-site←
         /sub-site.dk.php');
10   break;
11  default:
12    include($_SERVER['DOCUMENT_ROOT'].'/assets/lang/sub-site←
         /sub-site.en.php');
13  break;
14 }

```

Figure 9.3: Including language support on a sub-site

9.5 Pics Manager - Make

The way the Pics Manager - Make feature is implemented is very similar to how the Profile Picture Changer is implemented, see section 9.3.

They are both called the same way and both rely on the HTML Form Element to receive their data. They also give back their error message in the same way.

But what is more interesting is their differences. Looking at figure 9.4 it can be seen how the user interface is designed. The top left part is for entering the

general data, and the right side is for the image and the sound file.

We have designed it so that when a user chooses a sound file, or an image file, they can hear, or see, a preview of it before uploading. We also did this so that the design could be reused for the editing feature.

The preview is possible because of a JavaScript function called `FileReader` which simply loads the file into the browser, and makes it possible to work with it without uploading the file. But this is only available in some of the newer browsers which means we have had to disregard the use of IE9 [5]. In doing so we have created a warning on our login screen which checks if the `FileReader` is available, and if not the user will be met with a warning, telling them some features will not function for them.

This is done with the help of an open source JavaScript library called modernizr [10].

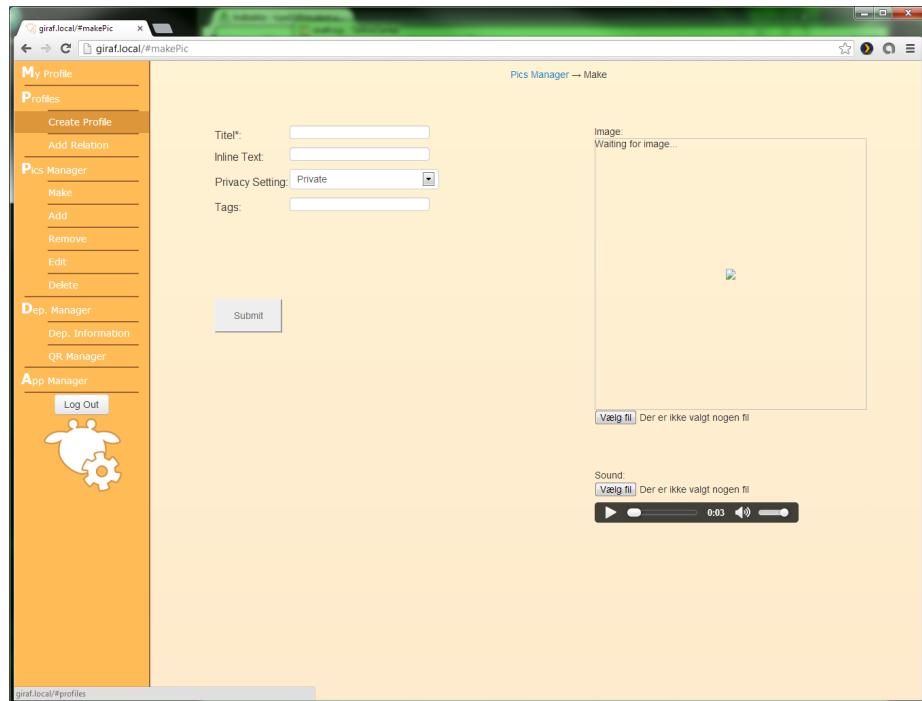


Figure 9.4: The user interface for the Pics Manager - Make

Again when speaking about the backend of this feature, it is very similar to the Profile Picture Changer Feature, where the main difference is that Pics Manager - Make has a specialized sound file recognizer, since this is not natively supported in PHP.

The function used for this can be seen in listing 9.7. The first 2 lines (L. 2-3)

creates two arrays, one for accepted mime-types and the other for accepted file extensions.

Next we do the actual check on file extension and mime-type. (L. 6-23), this is done by simply pulling out the information from the uploaded sound file, and see if it matches any of the variants in the arrays. This means that the user can upload a file with a `mp3` mime-type and a `.wav` file extension. - But since this is still an accepted sound file it is no security risk.

As it can be seen two variables which are used as booleans, `$fileExtOkay` and `$fileMimeTypeOkay`, these are used when we want to return the information of whether or not this was an acceptable sound file (L. 26-31).

Listing 9.7: The function used for checking sound files.

```

1  function isAllowedSoundFile($fileName, $fileTmpName){
2      $supportedExtensions = array('3gp', '3gpp', 'flac', 'mp3', '←
3          'mid', 'xmf', 'mxmf', 'rtttl', 'rtx', 'ota', 'imy', 'ogg', '←
4          'wav');
5
6      $supportedMimeTypes = array('audio/mpeg', 'audio/mp3', '←
7          'audio/mid', 'audio/wav', 'audio/x-wav', 'audio/rtx', '←
8          'audio/3gpp', 'audio/ogg', 'audio/mobile-xmf', 'audio/←
9          mxf'); //Could not find mime type of .rtttl, .ota ←
10         and .imy
11
12
13
14     //Check file extension
15     $ext = pathinfo($fileName, PATHINFO_EXTENSION);
16     if(in_array($ext, $supportedExtensions)){
17         $fileExtOkay = true;
18     }
19     else{
20         $fileExtOkay = false;
21     }
22
23
24     //Check Mime Type
25     $finfo = finfo_open(FILEINFO_MIME_TYPE); // return mime ←
26         type ala mimetype extension
27
28     if(in_array(finfo_file($finfo, $fileTmpName), ←
29         $supportedMimeTypes)){
30         $fileMimeTypeOkay = true;
31     }
32     else{
33         $fileMimeTypeOkay = false;
34     }
35     finfo_close($finfo);
36
37
38     //return
39     if($fileMimeTypeOkay && $fileExtOkay){
40         return true;
41     }
42 }
```

```

28     }
29     else{
30         return false;
31     }
32 }
```

When the sound and image files have been verified they are sent on to the database query, the main part can be seen at listing 9.8. But as it also can be seen it is called with a JSON object, which we have another function for. It simply constructs a JSON string from what information the user sent us, we call this function `makeJsonPictogram`.

When we use the function `makeJsonPictogram` the image and the sound file is encrypted to base64, which is needed in order to ensure we do not mess up the actual database query. By the database query we are referring to the SQL query that the Wasteland API produces from our JSON call.

Listing 9.8: The main part of the DB query for creating Pictograms

```

1 function db_uploadPictogram($jsonPictogram){
2     global $session,$username,$password;
3     $data = '{
4         "action": "create",
5         "auth": {
6             "username": "' . $username . "' ,
7             "password": "' . $password . "'"
8         },
9         "data": {
10             "type":"pictogram",
11             "values":['.$jsonPictogram.']
12         }
13     }';
14
15     $result = db_query($data);
16
17     if ($result[ 'status' ] == 'OK')
18     {
19         return $result[ 'data' ];
20     }
21     else
22     {
23         return false;
24     }
25 }
```

When the database query have been completed, the feature will as the Profile Picture Changer, send the user to the creation page with a message of success.

9.6 PHP Session Variables Overview

The login system of the GIRAF Admin system uses PHP sessions to check if a user is logged in. The PHP session is also populated with various variables that help the rest of the GIRAF Admin system. An overview of the different variables can be seen in Table 9.1 Many of the variables stored in the session is used

Session Variable	Description
<code>session_id</code>	Stores the PHP session id of the user
<code>username</code>	Stores the username of the logged in user, Temporary variable until Wasteland implements API session, should be removed before deploying anywhere!
<code>password</code>	Stores the password of the logged in user, Temporary variable until Wasteland implements API session, should be removed before deploying anywhere!
<code>userId</code>	Stores the user id of the logged in user, and is used for database calls
<code>profileId</code>	Stores the profile id of the logged in user, and is used to reduce database calls (you can get profileId from userId)
<code>lang</code>	Stores the chosen language (chosen at login) of the logged in user, and is used on all sub-sites to determine what language file should be used
<code>dbsess</code>	Stores the database API session, which is used to make database calls after the initial login, this is here for future use and does not contain anything at the moment, as the database API have not implemented session yet
<code>department</code>	Stores the department id that the logged in user is attached to, and is used to reduce database calls
<code>role</code>	Stores the role of the logged in user, and is used to determine what pages to show
<code>update</code>	Stores the update value of the logged in user, and is used to determine rights
<code>delete</code>	Stores the delete value of the logged in user, and is used to determine rights
<code>isAdmin</code>	Stores whether the logged in user is an admin or not, which determines what menu items should be available

Table 9.1: PHP Session Variables

to reduce database calls, which could help the system scale better if that would be necessary. Because API session is not yet implemented by Wasteland, it has been necessary to store the username and password of the logged in user, and is then used to make database calls instead of the session. When API session

is implemented the username and password variables should be removed from PHP session and the authentication method in `new.db.php` should be replaced with API session in all calls except `db_getSession`. At this moment it is not clear what exactly the update and delete variables signify, and the reader should refer to the WASTELAND report or source code for more details.

This is the end of all the advanced and complicated solutions we have managed to complete, the next chapter will focus on the results of our usability testing of the somewhat finished system.

Chapter 10

Usability Testing

One of the project's goals is to create an interface which has a high degree of user friendliness. This coincides with that software projects are usually tested against users before eventual release.

This chapter will describe the usability test conducted for the developed interface.

10.1 Structure of the test

The test was performed in Cassiopeias usability lab, shown on Figure 10.1, where a computer, in subject room one, was logged onto our website. Each test person was lead into the test room and offered a cup of coffee, after which they were given a short briefing and asked to sign a consent form in which they agreed to that they were being filmed for research purposes.

After the introduction the test person was asked to perform a set of short assignments which were designed in such a way that they would try the complete set of implemented features in the system. When the test person had completed all the assignments, he was given a set of follow-up questions about some of the ways that the test person used the system during the test.

During all completed tests it was the same person performing the role as instructor, as to not get biased test results.

The assignments as well as the introduction given to all participants can be found in appendix D on page 119.

The usability test was performed with four test persons. Even though that number is not very high the actual number of people which will be using the advanced features of the system such as creating profiles will not be much higher when the system is deployed. One of the test persons, which job include daily leadership of a kindergarten, was before the test considered one of those who will actually work with the advanced features but during the follow-up questions

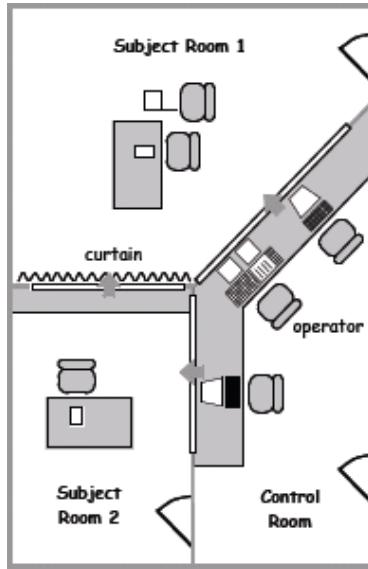


Figure 10.1: Usability Lab

it became clear that this might not be the case. The test can therefore be seen as a little biased as to having test persons without the appropriate level of technological knowledge. As this person might work with the system's advanced features in the future this person is viewed as a proper user and the other three as confirmation subjects. Two of these confirmation subjects work with children which has autism, every day and the last is an educated computer scientist as well as a parent to a child with autism. The latter person is considered to have a higher level of technological knowledge than the eventual users of the advanced features and was mainly used for confirming critical bugs and give suggestions for new features.

Upon eventual release all of the test persons would work with the non-advanced features, if not on a daily then on a weekly basis.

As the developed system at the time of the test was not in a state were it was stable enough to test on, an older version of the administrative interface was used during the test. The version which was used during the test can be collected from GitHub [12].

10.2 Result of the test

Table 10.1 displays the errors or bugs, which was found during the usability test. They are described by a category, the error itself, its severity and lastly if it is corrected in the final version before hand-in.

They have been sorted by the severity of the error, and here we use three categories, critical, serious and cosmetic. Critical depicts an error which either makes a general task impossible or made the user stop altogether. Serious depicts an error which made the user stop for some time, or seemed to be unnecessarily distracting. Cosmetic depicts an error of low calibre, which even some of the users informed us was minor issues which did not affect the flow.

Description	Severity	Corrected
Pictograms - It is not possible to create new categories of Pictos	Critical	
Profile - Restrict QR editing to department manager	Critical	
DB Problem - Not possible for all pedagogues to fix Pictos for all department children	Critical	
Navigation - The site "Profiles" should link to each profile	Critical	
Missing - Unable to remove relations	Critical	
Profile Pic - Accept button is hard to find	Serious	
Profile Pic - Word "change" is misleading	Serious	
Navigation - "Add" and "Make" under Pics Manager is confusing	Serious	
Navigation - Language Support on navigation did not change to Danish	Serious	X
Missing - No Danish language support on the site "Profiles"	Serious	X
Profile - Department should be a link, not editable	Serious	(X)
Profile - Links from Own Profile relations is missing	Serious	
Profile Pic - GIRAF Logo as Place holder is misleading	Cosmetic	X
Profile Pic - Word "Edit" is not informative	Cosmetic	
DB Problem - Own Profile takes too long to load	Cosmetic	
Navigation - "Add Relation" is before "Create Profile"	Cosmetic	X
Design - Standardize button names	Cosmetic	
Logout - Can navigate in system without session	Cosmetic	

Table 10.1: Bugs/Errors Found Under Usability Testing

Some of these errors was nearly immediately rectified, or had already been solved in the system which was up to date. These can be seen in table 10.1 as "X". If the X, is marked with parentheses, it is because the problem have been partially solved.

As mentioned before there is errors marked critical, as can be seen by the first

5 entries in the table. These are errors which must be fixed for the system to be able to use the implemented feature set, as listed in chapter 7 on page 53. Next an explanation of the critical problems, and in short how they can be rectified.

Pictograms - It is not possible to create new categories of Pictos

This error arose because the task of creating categories was not thought through. This is one feature which was overlooked, even after having had a meeting with the contact person, and showing the intended feature set.

Solving this error requires the creation of a minor tool which makes it possible to create and assign categories by a simple entry of a text string. For navigation purposes this tool should be under the Pics Manager category in the navigation menu.

Profile - Restrict QR editing to department manager

After the first test person, it was made clear that the test person thought highly of the QR code, which they are given for the system. It is supposed to be as important as your house key, and should therefore not be something which can easily be changed.

To accommodate for this request the QR changer should not be available from the Own Profile site. Instead the procedure will be to contact the administrator of the system and have him generate a new key with the QR Manager tool and then send this out.

DB Problem - Not possible for all pedagogues to fix pictos for all department children

This problem most likely erupted from a misunderstanding between the DB group, Wasteland, and ourselves. The problem is that in order for a pedagogue to edit anything about a child in its department they need to have a relation between them. But this is not how the user intends to use the system. The relation should only be thought of as a responsibility link, and only adds the feature to change this child's profile data.

There is no easy way to rectify this problem from the admin projects side. This has to be fixed inside the database, because of its security mechanisms. For a solution to this it is suggested to look at the database project, Wasteland [8].

Navigation - The site "Profiles" should link to each profile

This is simply a matter of a feature which has not yet been implemented. It is shown here as an error, because it is necessary for the system to be operated correctly.

It can be fixed by adding normal HTML anchor links to each name in the tables.

Missing - Unable to remove relations

This is again a matter of a feature which have not yet been implemented. This feature is thought to be usable only for the department manager and the administrator of the system and it should be used directly from each users pro-

file. This means that the administrator or department manager, navigates to the user's profile, find the person he or she is related to and clicks the button to remove the relation.

The rest of the errors is not of a nearly as urgent calibre, and will therefore not be discussed further in this report.

10.3 New Features

Instead the attention is turned to the possible extra features found during the test, both inspired by the way the users used the system, and their commentary afterwards.

Description
Make the modal window customizable
Profile - Press enter to save change
Pics Manager Make - Add recording feature
Input forms - Automatic first letter uppercase for name and address
Navigation - Add link to "add relations" and "create profile" in Profiles
Pics Manager Make - Directly create picto for child from profile page
Pics Manager Make - Auto add "inline-text" to picto preview

Table 10.2: Features Found Under Usability Testing

As seen from table 10.2 there is a few features which could improve the system. Some of the features which might be a bit hard to understand without having been on the development team is explained in detail in the following text.

Make the modal window customizable

During this project a modal window was designed because of some issues with the bootstrap library. It seemed that the user sometimes found it confusing with the many cancel options which is offered for closing the modal window, even though it follows the way bootstrap displays their modal windows.

But also in the long run it could be useful to have the ability to customize the modal window with more than a title and a text.

Input forms - Automatic first letter upper case for name and address

While watching the users work in the system, multiple times it was witnessed that they deleted a full text string because they entered the first letter of an address or a name in lower-case. therefore to save the user time, making each word in the address and name field automatically convert to upper-case since

that is the custom with all names and addresses.

Navigation - Add link to "add relations" and "create profile" in Profiles

Some of the users in the test meant that the most meaningful way to handle profiles, would be directly from the menu point "Profiles", and did not bother to look just below this menu point, where these options lay. This behaviour was seen in 3 out of 4 cases, and should therefore be considered.

It is a simple addition and does not hinder the system in any way. So if it even only helps a few people in the beginning of the learning of the system, this is feature worth having.

Pics Manager Make - Directly create Picto for child from profile page

While watching the users work in the system, and also afterwards, when they were interviewed. We came to understand that some of the pedagogues, in our test case, 2 out of 3. Thought of the child as the central element in the system and therefore wanted to create everything out from the child.

This meant that when they had to add parents or pictograms to the child, they went to try and find the child's profile. therefore in order to accommodate this behaviour we should implement a feature which makes it possible to automatically relate a child and a Guardian as well as a child and a pictogram, when the action is started from that child's profile.

Pics Manager Make - Auto add "inline-text" to picto preview

Some of the users did not quite understand what the inline-text field in the Pics Manager Make function meant, before we asked them. Which after they immediately understood what it was for, without us telling them. During the test, we even saw one of our test persons fill the field, and then delete the entry again.

If this was a result of the assignment regarding the use of pictograms did not specify that the inline-text field needed to be filled, or something else, we are not certain.

But a way to avoid this problem would be to simply add a JavaScript feature which automatically writes on top of the temporary display of the new pictogram.

The reader should now be as well informed of the structure and shortcomings of the GIRAF Admin system, as the developer group was. But in case the reader wants more information about the system, the reader should try to contact Ulrik Nyman, Associate Professor at Aalborg University and ask about the GIRAF Admin group of the summer semester, year 2013.

In the next chapter we draw conclusions about the system as a whole, and after that we make an evaluation of the process and the multi-project as a whole.

Part IV

Perspective

Chapter 11

Conclusion

This project's focus was on making an administrative interface for the GIRAF system on both a desktop as well as an android platform. Although the developed system was never deployed on an android platform the system is running on a server and is accessible through common web-browsers.

As it is already shown in chapter 7 on page 53 some of the features in the system were not completed. This was due both to trouble with the Wasteland API as well as unforeseen trouble with different aspects of the already created system. This project did however not set out to make a 100% complete system, but rather a system that, included completed parts, which would not require additional work. With that in mind the most important parts of the administration system to complete would be the user management parts, since the pictogram features would be implemented by other projects on the tablet.

By looking at table 11.1, which displays all the errors and bugs we found during the usability test described in chapter 10 on page 77. It is clear that there are a number of critical errors in the system which in term makes the system unable to fulfil its goal. In order to fulfil the goal these errors must be corrected. 4 out of 5, of the critical errors can be corrected by a couple of days hard work, but the last critical error regarding the database problem, can not be corrected within the administration system.

Concluding on the usability test the system is quite user friendly. If the problems in table 11.1 were corrected the system should be free of slowing effects in the matter of daily use. More of the test persons did praise the system on the simple structure during the interview after the test.

The system only has one codebase and the maintainability for the system should therefore be a lot higher than that of the former two systems. Also the group has made an effort to comment on the written code, as well as in this report to describe some of the more complex solutions, for an overview of these see chapter 7 on page 53. This report also comes with an install guide in appendix

Description	Severity
It is not possible to create new categories of Pictos	Critical
Profile - Restrict QR editing to department manager	Critical
DB Problem - Not possible for all pedagogues to fix Pictos for all department children	Critical
Navigation - The site “Profiles” should link to each profile	Critical
Missing - Unable to remove relations	Critical
Profile Pic - Accept button is hard to find	Serious
Profile Pic - Word “change” is misleading	Serious
Navigation - “Add” and “Make” under Pics Manager is confusing	Serious
Profile - Department should be a link, not editable	Serious
Profile - Links from Own Profile relations is missing	Serious
Profile Pic - Word “Edit” is not informative	Cosmetic
DB Problem - Own Profile takes too long to load	Cosmetic
Standardise button names	Cosmetic
Logout - Can navigate in system without session	Cosmetic

Table 11.1: Bugs/Errors Found Under Usability Testing without the fixed bugs

A, so that this project should be as painless to continue, as possible.

Within all of this the initial goal is not obtained. But much has been done to ensure that continuing this project would be easy. A goal we thought higher of than simply completing the system.

In the next sections we will evaluate on the multi-project itself and after that make a note of what could be done in the future to improve this system as well as the GIRAF system as a whole.

11.1 Evaluation of the Multi group project

This semester is a special semester for software students at Aalborg University. This semester is the first time we work as teams on a bigger project. In connection with this we wanted to evaluate the multi project as a whole, so that future students might benefit from our findings.

11.1.1 Cooperation between groups

First of all, it is always hard to get along when you are 8 groups consisting of nearly 30 people all in all, who each have their own agenda in mind. We did all agree on one common goal, but it seemed that most people preferred to isolate themselves to their own projects. - This might have to due with the fact, that

we until now have worked that way.

But the few groups which were entirely dependent on each other, for example our group and the database group, at least tried to work together. However the database group was faced with much hardship, both from their project and some of the other groups. As a result of this they ended up being difficult to contact during an important period of time.

However this was an interesting experience, which have taught us much about how group work functions outside the university. How big groups of people have to work together in order to complete a vast system. Both with its ups and downs.

11.1.2 Advice for next year

As we have been working with the multi project for about 6 months, we have some advice and suggestions for the coming year.

Advice

In order to end with a more satisfiable product, decide on the focus of the project, earlier. We decided the focus for our project, in the middle of it, which turned out to be too late. We feel that we made an okay product, but if we had decided on a focus earlier, that might have been better.

This multi project, was so lucky to have someone that was able to set up a starting suit of software. We started with having a server with CI, Redmine and a Wikipedia. As well as the decision to force all projects on GitHub, have made it very easy to find and work with each others project. We highly recommend keeping this going for the next year's students.

Suggestions

There were a few things that we would have liked to try this semester, but we did not, for whatever reason. Which we here list.

- Make it a habit to show off your system, so that the other groups might get inspiration from it.
- More work together between the groups. Not necessary on the same thing, but in the same room.
- When nearing the end of the project try to strive for the ability to make a joint usability test, between the groups.

11.1.3 General Evaluations

There were also was a few things, that did not fit any category. These are explained here.

In this multi project, we used burndowncharts, which we ended up thinking of more as a chore than as an beneficial work progress. We believe that this is due to the fact that we are not educated properly in their practical use. So if the students next year are going to use burndowncharts, they might want to consider having a joint session about, how to use burndowncharts.

We found it peculiar, the way we had to construct our report. It felt as a documentation of the system, but in the form of a project report. We are not entirely sure how to fix this problem, but we would suggest that the semester coordinator explained to the students how to think of this report.

He did try to do so, this semester by giving a list of things that needed to be in the report, however this did not help on the mindset of the report.

In our group, we were so unfortunate that our tablet broke down. We had to wait for more than a month of the development time, before receiving it back from repair. It might have been lucky, that it was our group that had this problem since we were able to work on our PC version of the system. - However it will become problematic if there is no better backup solution, if a tablet fails in the future.

This semester, we the multi group, decided to have a meeting each week. Sometimes it has felt like a waste of time, however it also gave a better overview of the other group's progress. Also it made it harder for problems to go unnoticed. We also sat down committees, which could make some major solutions over the span of the individual groups. We appreciated this much, because it felt as an effective way to make decisions which had great impact.

Chapter 12

Future Work

This chapter will contain all of the things that could be made to the system in order to improve it. This will also serve as a guide for the group next year that has to continue developing on this system.

- Correct all of the bugs/errors found in the usability test
- Create all the new features found in the usability test
- All of the features described in chapter 7 that are not done yet, should be completed.
- The system should be ported to Android using PAW
- A calendar system could be incorporated into the system
- A messaging system could be added so that messages could be pushed to the system

12.1 Project Proposals

To keep a continues progress of GIRAF there is need for further development opportunities. Here we have written a number of items which we have discovered during our work with GIRAF.

Event Calender After meeting Mette Als at Birken it was clear that there is a need for a digital organizational tool. To plan the guardian and the children's daily schedule.

The schedule should also be able to print for parents.

Child Mode Child mode is needed to ensure children are able to play with the tablet without supervision from guardians. Child mode means that the tablet should be totally restricted to a special app or apps in launch mode.

Data and Analysis Create application which gathers information from apps to analyze the child's development.

Co-op Games Develop a game which has in focus for children to interact with other children to accomplish a task.

Development Game Tool Create a tool for easy game development, so that guardians, are able to design games for the children.

Language Stimulation Create an app which stimulate children's ability to spell and pronounce words.

Chapter 13

Acknowledgements

Reports within the GIRAF project can possibly be received through Aalborg University.

Savannah

Authors:

- Jesper Bromose
- Sebastian Lybæk
- Martin Fjordvald
- Thørbjørn Kvist Nielsen

Title:

Savannah - part of the GIRAF system

Oasis

Authors:

- Henrik Klarup
- Jens Mohr Mortensen
- Dan Stenholt Møller

Title:
Oasis: Del af GIRAF Systemet : Application Development

Wasteland

Authors:

- Simon Jensen
- Hilmar Magnussen
- Jeppe Tarp
- Barbara Flindt

GIRAF Project <http://giraf.cs.aau.dk/>

13.1 Technologies used in this project

- Balsamiq
- Apache
- PHP
- HTML, CSS, Javascript
- Git
- Python
- Phpqr code
- SimpleImage
- PAW
- Bootstrap
- jQuery
- jQuery.imgareaselect
- Flags
- Modernizr

Bibliography

- [1] Wikipedia, 2013. Looked up: 26-05-2013.
- [2] Alexandre Assouad. *PHP QR Code - SVG/EPS Update*. URL: <https://github.com/t0k4rt/phpqrcode>, 2012. Looked up: 20-05-2013.
- [3] balsamiq. Balsamiq studios. Website, May 2013. <http://www.balsamiq.com/>.
- [4] American Accreditation HealthCare Commision. Autism, May 2012. <http://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0002494/>.
- [5] Alexis Deveria. Can i use... Website, May 2013. <http://caniuse.com/>.
- [6] Dominik Dzienia. *PHP QR Code*. URL: <http://phpqrcode.sourceforge.net/>, 2010. Looked up: 20-05-2013.
- [7] The Wasteland GIRAF group. Wasteland api. Website, May 2013. http://htmlpreview.github.io/?https://github.com/Ezphares/giraf_database/blob/master/source/api/documentation.html.
- [8] The Wasteland GIRAF group. Wasteland project. Website, May 2013. https://github.com/Ezphares/giraf_database/.
- [9] Simon Jarvis. Simpleimage php libary. Website, November 2006. <http://www.white-hat-web-design.co.uk/blog/resizing-images-with-php/>.
- [10] Modernizr. Modernizr. Website, May 2013. <http://modernizr.com/>.
- [11] Pyramid Droup Management Services. Picture your student learning. Website, Visited 21.03.2013. <http://pecs.com>.
- [12] SW601F13. Giraf - admin: The test version. Website, May 2013. <https://github.com/Zucka/girafAdmin/tree/46e19108a6ca0413e25426212552ac6da99b8962>.
- [13] Twitter. Bootstrap. Website, May 2013. <http://twitter.github.io/bootstrap>.
- [14] Michael Wojciechowski. jquery img libary. Website, May 2013. <http://odyniec.net/projects/imgareaselect/>.

[15] WordPress. Wordpress.org. Website, May 2013. [http://wordpress.org/.](http://wordpress.org/)

Part V

Appendix

Appendix A

Installation Guide and Configuration

This section will describe how to install and configure the GIRAF Admin system.

A.1 Requirements

- A suitable operating system (Ubuntu, Windows and OSX was used during development)
- Apache (Not tested on nginx)
- PHP5 with the socket module enabled
- A suitable browser (Chrome,Firefox,Safari,IE10+ is recommended)
- GIT (The project is hosted on GIT)
- Python if auto update is used

A.2 Installation

- Install Apache, PHP5, GIT and Python
- Pull the most current version from <https://github.com/Zucka/girafAdmin> (although forking the project is advised)
- Copy the contents of source/desktop into the www root of Apache (usually /var/www/ on UNIX systems)

A.3 Configuration

- Enable the socket module of PHP5 if not already enabled
- Enable the header module of Apache if not already enabled
- Modify the Apache configuration file (httpd.conf) to auto-inject a meta tag into every header (used to force Internet Explorer into standards mode). The configuration fragment to be added is listed in Listing A.1.

Listing A.1: The configuration fragment to add to httpd.conf

```
1 <IfModule headers_module>
2   Header set X-UA-Compatible: IE=Edge
3 </IfModule>
```

- Change the variables address and port in source/desktop/db/new.db.php to reflect the IP address and port of the server that hosts the database API

A.4 Auto Update

A python script was made to auto update the server from a GIT. It can be used to auto pull new changes from a GIT and then move the contents of a folder into another folder. To get it to work you need either Python 2.X or 3.

- Copy autoupdate.py from source/scripts/autoupdate.py to a seperate folder, to be used as an intermedidate folder
- Change all references to `/home/neo/autoupdate` to point to the seperate folder created for auto update
- Change all references to `https://github.com/Zucka/girafAdmin.git` to point to the new GIT for the project (the GIT installed on the system needs to have read rights for the GIT repository)
- Give autoupdate.py execute rights (`sudo chmod +x autoupdate.py`)
- Make sure that the www folder and all subfolders have the same owner
- Start autoupdate.py as the user that owns the www folder (example: `sudo -u www-data python autoupdate.py`)
- Alternatively start autoupdate.py in a screen (the UNIX program) instance so that auto update keeps running after the ssh connection has been terminated
- The www folder will now be updated every 60 seconds with the newest changes from the specified GIT

Appendix B

Graphical Mockups of The GIRAF Admin System

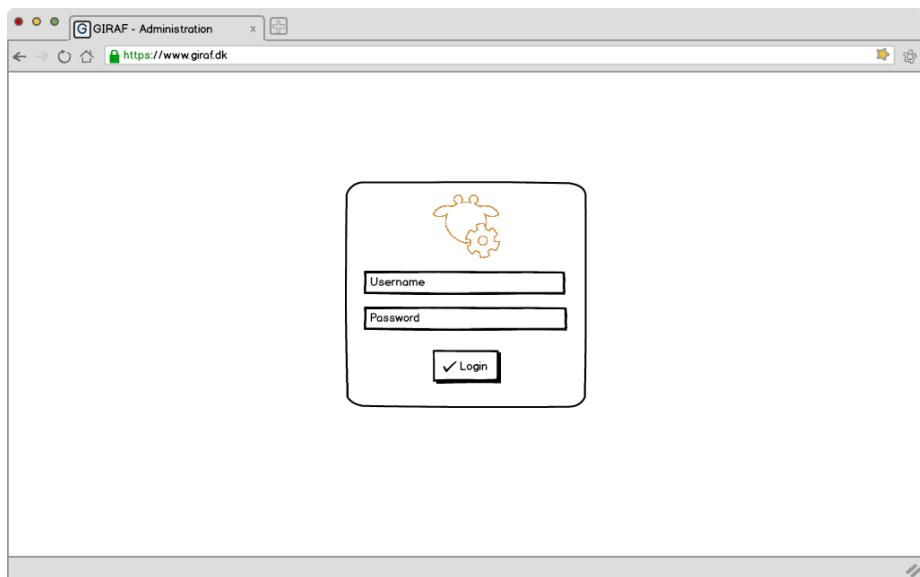


Figure B.1: Login

Graphical Mockups of The GIRAF Admin System

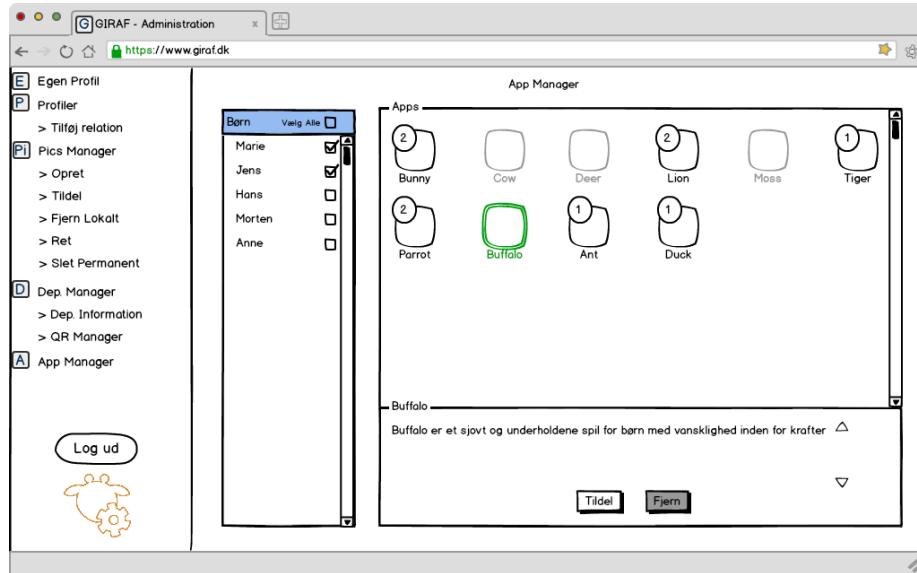


Figure B.2: App Manager

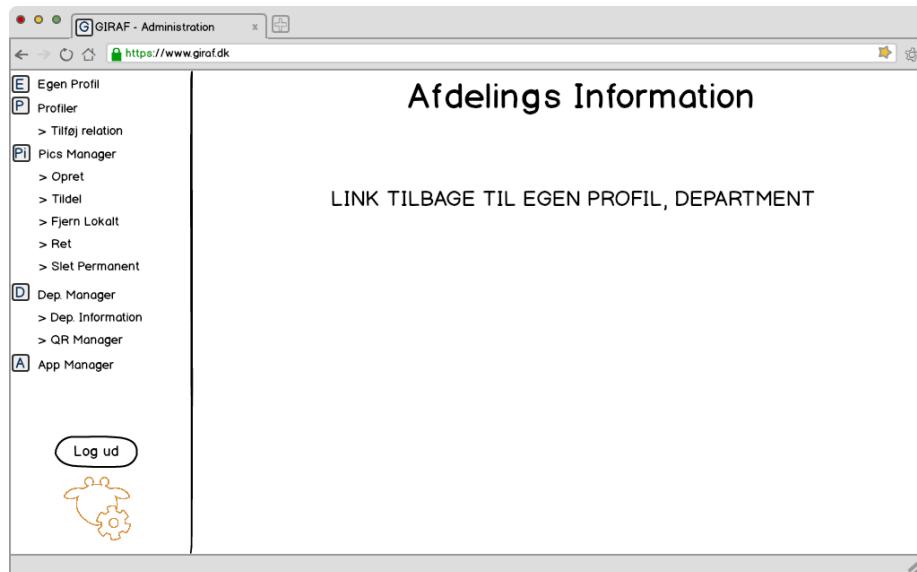


Figure B.3: Department Information

Graphical Mockups of The GIRAF Admin System

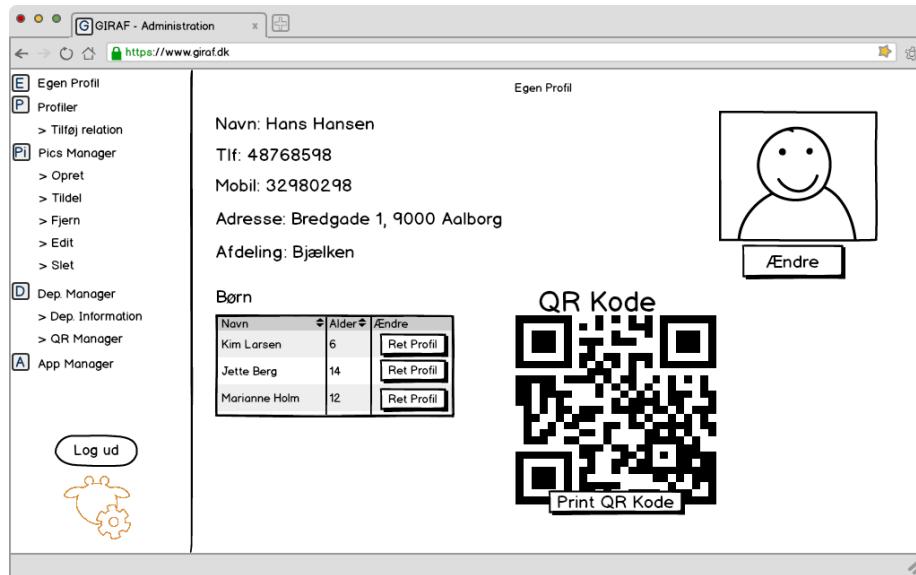


Figure B.4: Own Profile

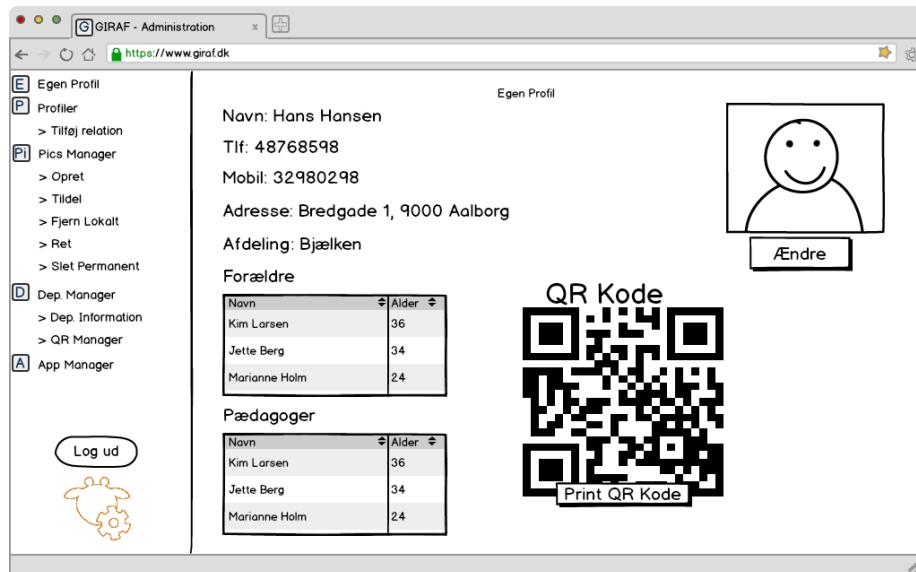


Figure B.5: Own Profile Child

Graphical Mockups of The GIRAF Admin System

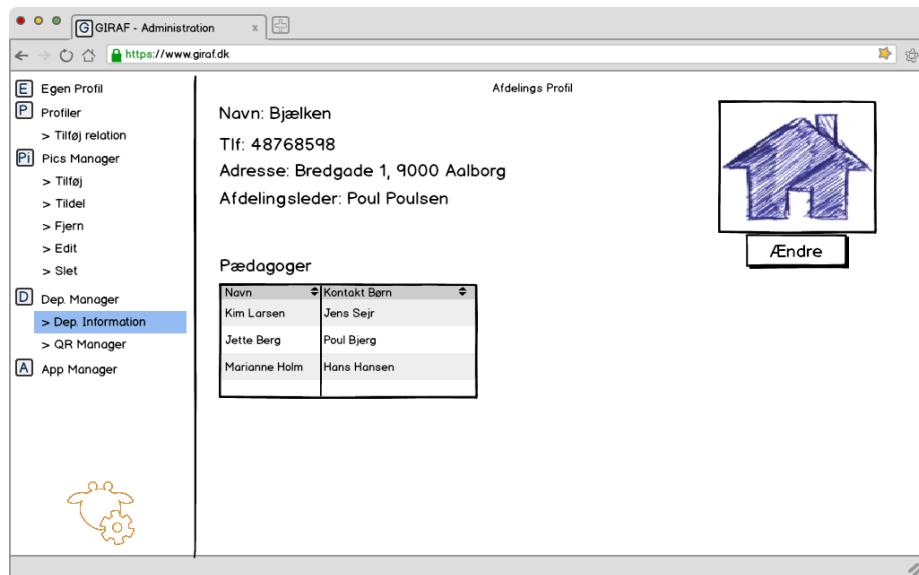


Figure B.6: Own Profile Department

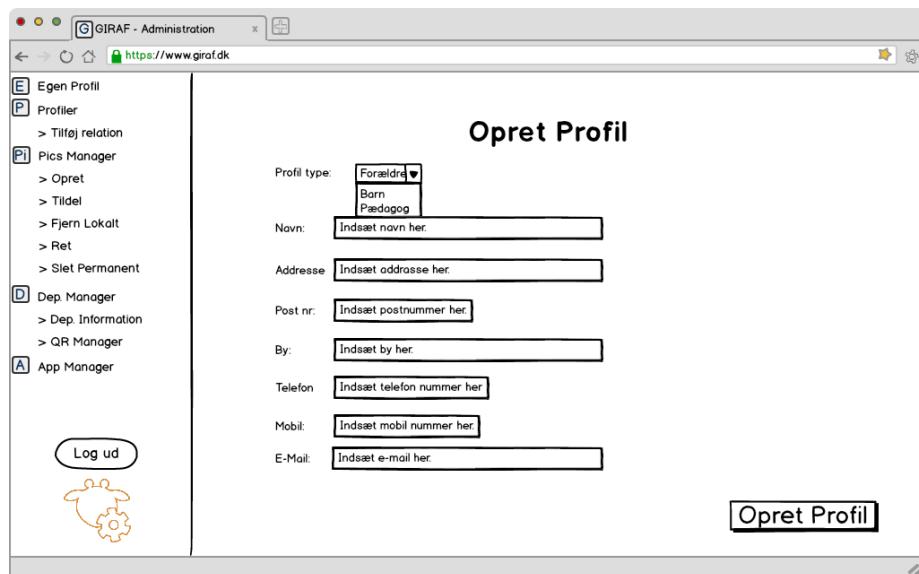


Figure B.7: Own Profile Guardian

Graphical Mockups of The GIRAF Admin System

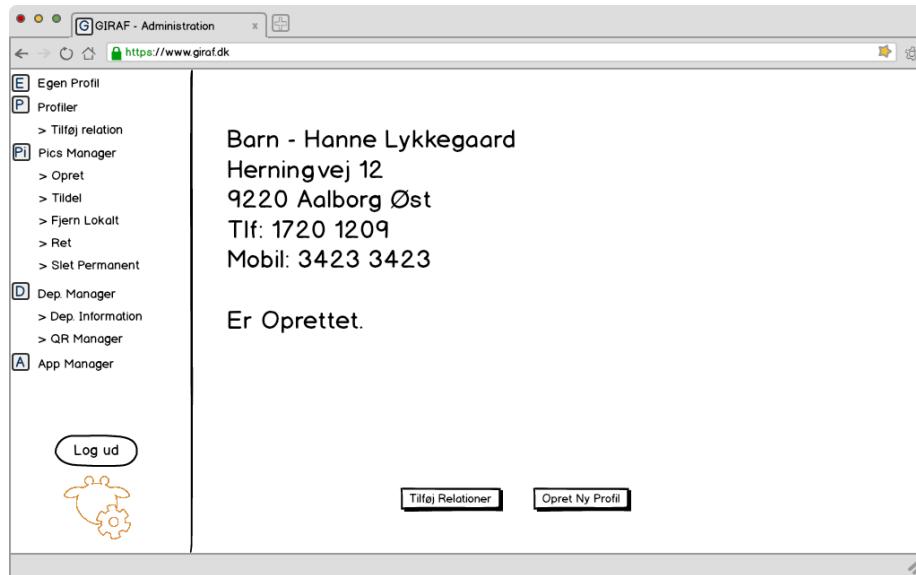


Figure B.8: Create Profile Accept

The screenshot shows a web browser window titled "GIRAF - Administration". The URL is https://www.giraf.dk. The left sidebar contains navigation links: Egen Profil, Profiler (with sub-links: Tilføj relation, Opret, Tildel, Fjern Lokalt, Ret, Slet Permanent), Pics Manager (with sub-links: Opret, Tildel, Fjern Lokalt, Ret, Slet Permanent), Dep Manager (with sub-links: Dep Information, QR Manager), and App Manager. On the right, there is a form titled "Opret Profil" for creating a new profile. The form fields are: Profil type (dropdown menu showing "Born" and "Forældre Pædagog"), Navn (text input field "Indsæt navn her"), Adresse (text input field "Indsæt adresse her"), Post nr. (text input field "Indsæt postnummer her"), By (text input field "Indsæt by her"), Telefon (text input field "Indsæt telefon nummer her"), and Mobil (text input field "Indsæt mobil nummer her"). At the bottom right is a large "Opret Profil" button.

Figure B.9: Create Profile Child

Graphical Mockups of The GIRAF Admin System

The screenshot shows the 'Tilføj Relation' (Add Relation) page. On the left, a sidebar menu includes 'Egen Profil', 'Profiler', 'Pics Manager' (selected), 'Dep. Manager', and 'App Manager'. Under 'Pics Manager', there are options for 'Opret', 'Tildel', 'Fjern Lokalt', 'Ret', and 'Slet Permanent'. A 'Log ud' button and a gear icon are also present. The main content area has two tables: 'Vælg Barn' and 'Vælg Guardian/Forældre'. The 'Vælg Barn' table lists 'Lille Lotte', 'Hans', and 'Peter' with checkboxes; 'Peter' has a checked box. The 'Vælg Guardian/Forældre' table lists 'Anders', 'Rasmus Holt', and 'Knud Top' with checkboxes; 'Rasmus Holt' and 'Knud Top' have checked boxes. A large 'Tilføj' button is centered at the bottom.

Figure B.10: Create Relation

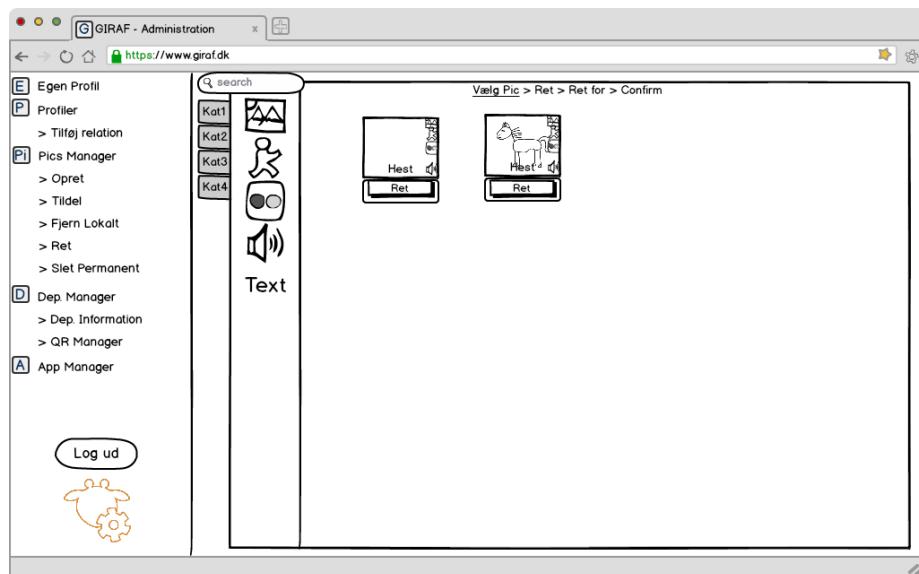


Figure B.11: Pics Manager Edit

Graphical Mockups of The GIRAF Admin System

Figure B.12: Pics Manager Edit 2

Navn	Vælg
Mads	<input checked="" type="checkbox"/>

Navn	Vælg
Mads	<input checked="" type="checkbox"/>
Mikkel	<input type="checkbox"/>
Frederik	<input checked="" type="checkbox"/>
Maria	<input type="checkbox"/>
Mette	<input type="checkbox"/>
Lotte	<input checked="" type="checkbox"/>
Gustav	<input type="checkbox"/>

Figure B.13: Pics Manager Edit 3

Graphical Mockups of The GIRAF Admin System

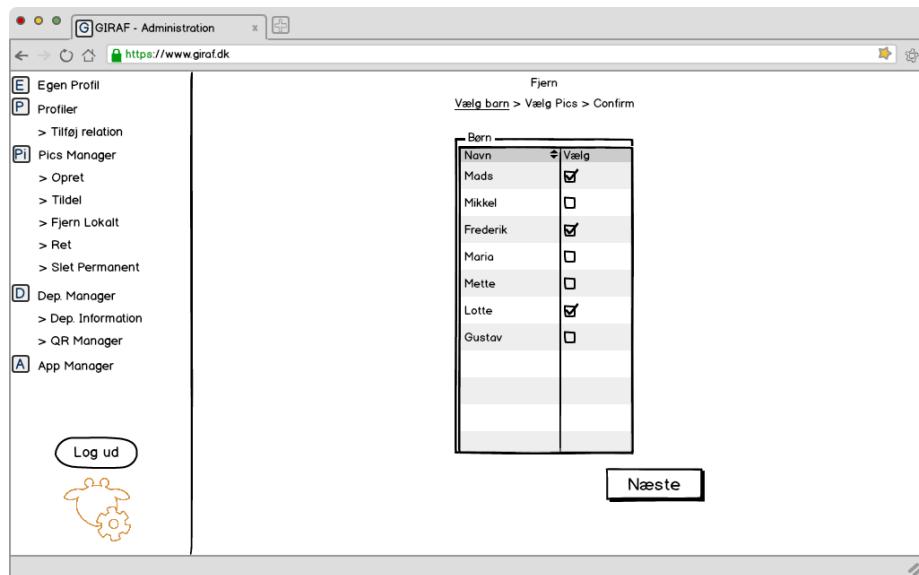


Figure B.14: Pics Manager Remove

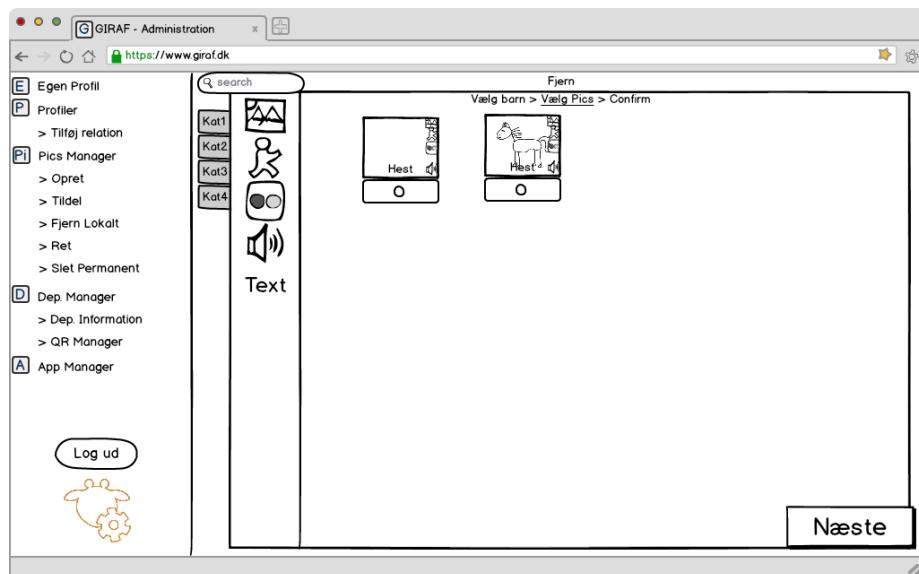


Figure B.15: Pics Manager Remove 2

Graphical Mockups of The GIRAF Admin System

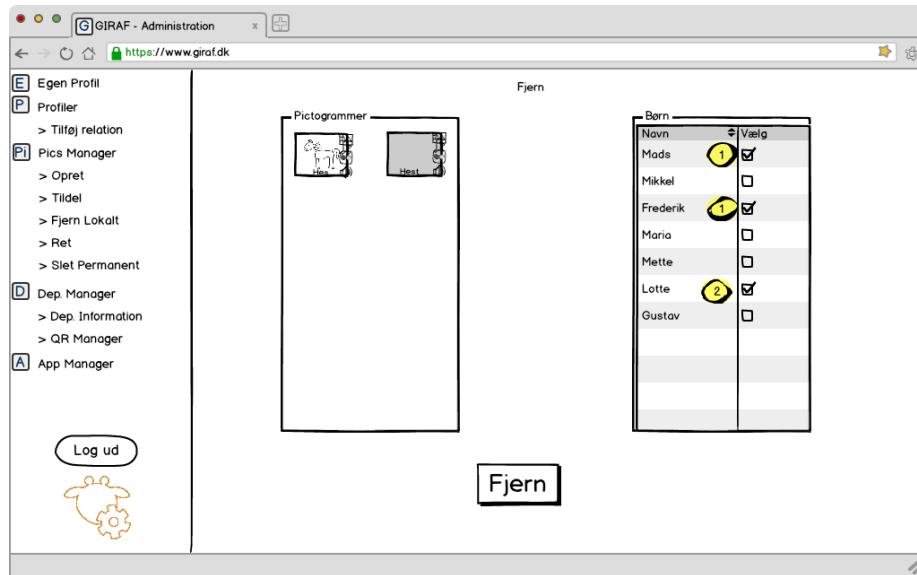


Figure B.16: Pics Manager Remove 3

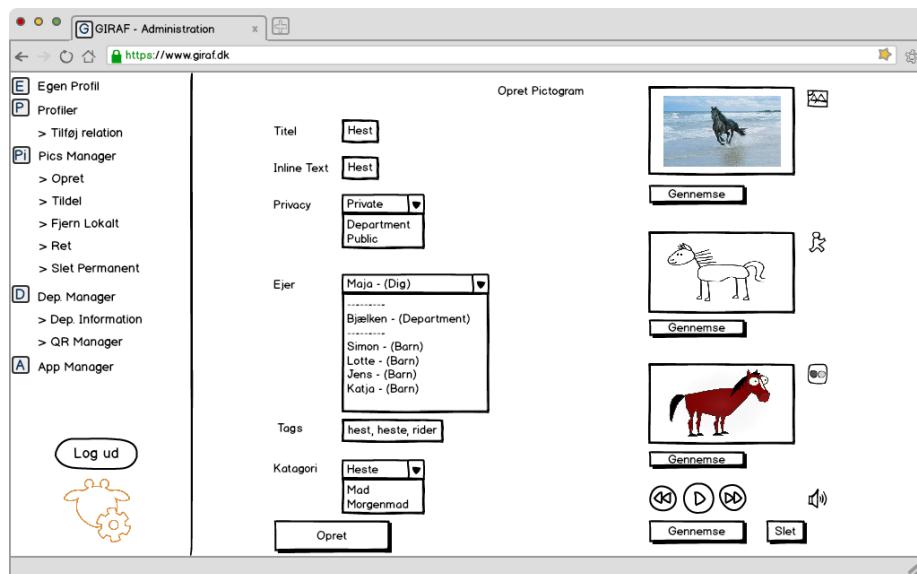


Figure B.17: Pics Manager Create

Graphical Mockups of The GIRAF Admin System

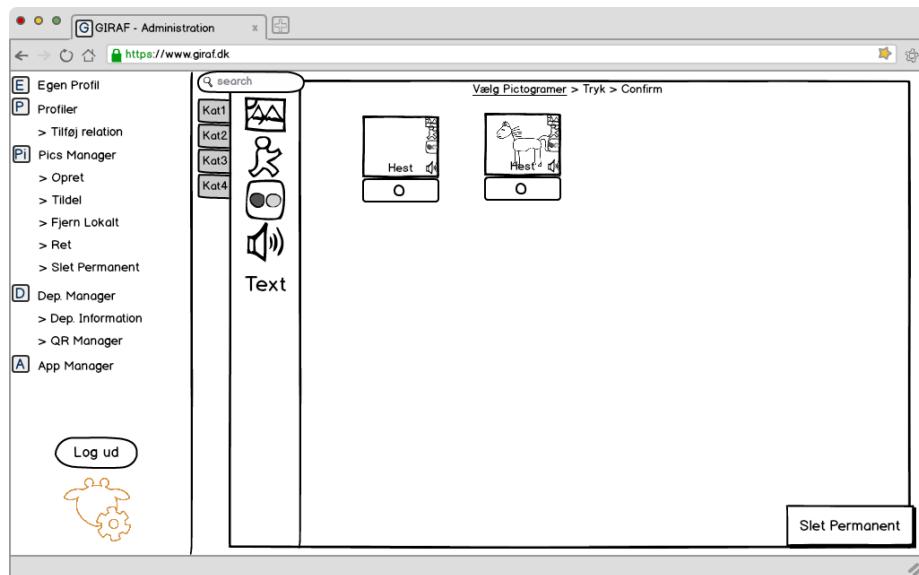


Figure B.18: Pics Manager Delete

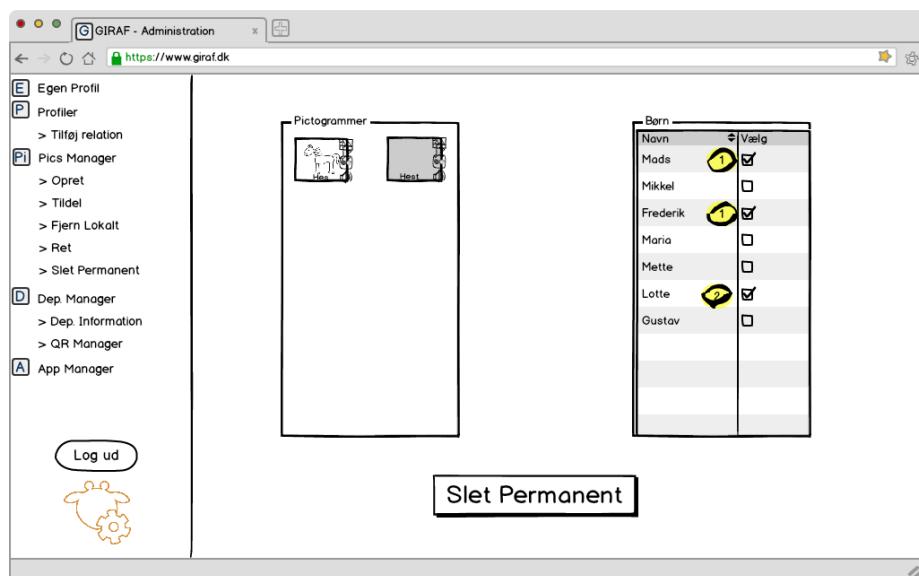


Figure B.19: Pics Manager Delete 2

Graphical Mockups of The GIRAF Admin System

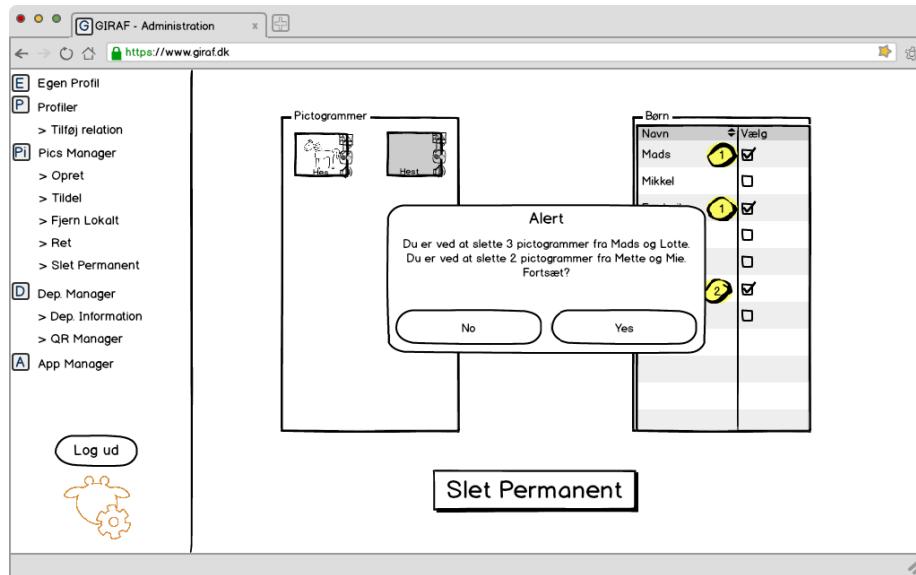


Figure B.20: Pics Manager Delete 3

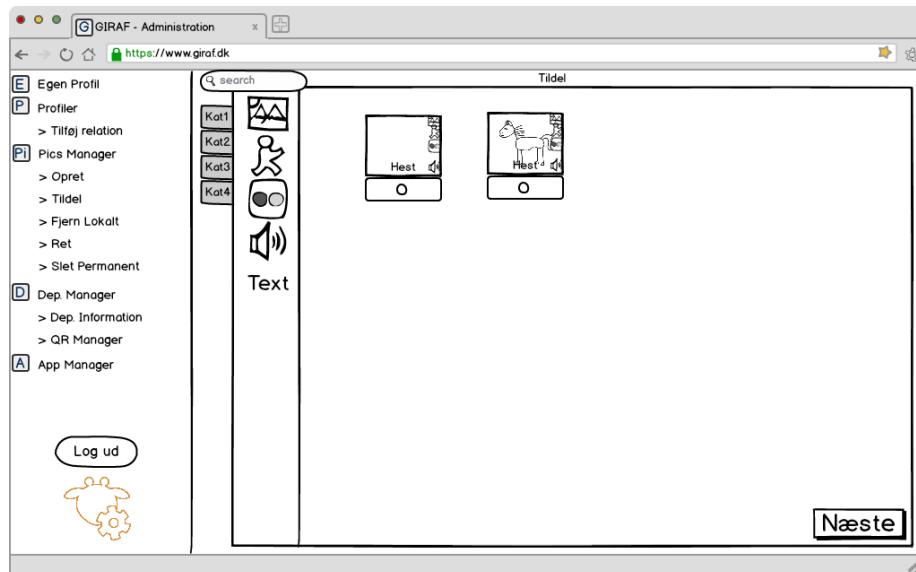


Figure B.21: Pics Manager Assign

Graphical Mockups of The GIRAF Admin System

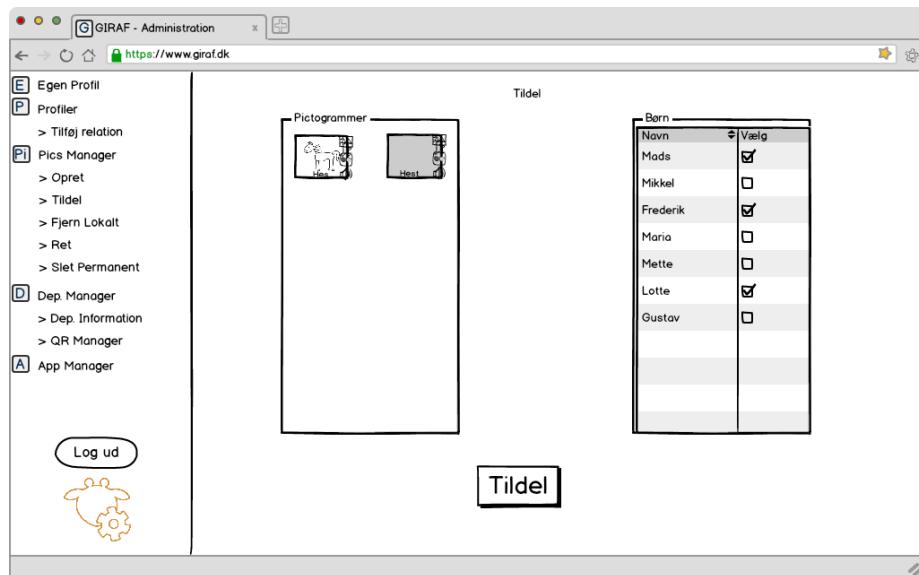


Figure B.22: Pics Manager Assign

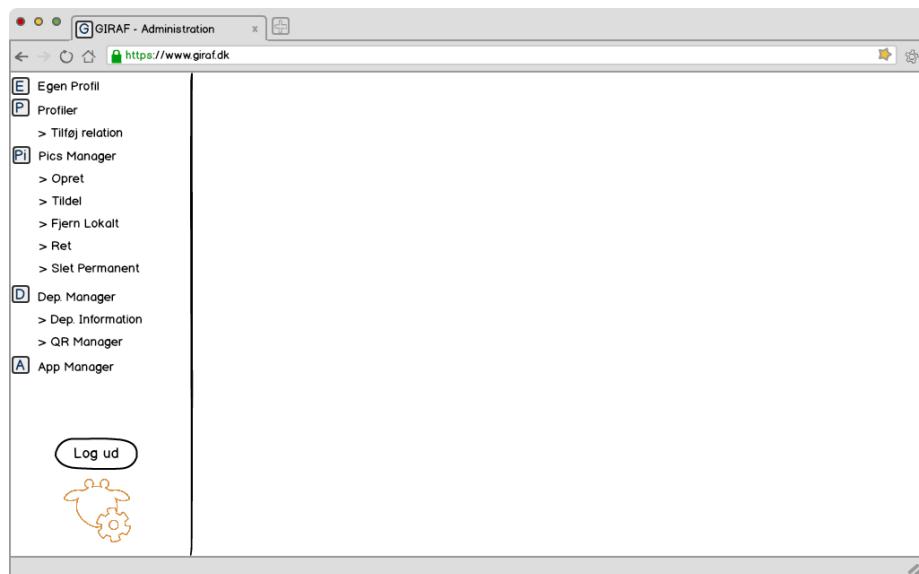


Figure B.23: Profiles

Graphical Mockups of The GIRAF Admin System

Pedagog	Børn	Forældre
Kim Larsen	René Poulsen	Anna Sejer
Holly Hansen	Inge Karp	Henriette Nielsen
Jens Sørensen	Gurli Hansen	Carsten Karstensen
Poul Ravn	Mark Sørensen	Morten Mathiasen

Opret Ny Profil

Figure B.24: Profiles Department Manager

Pædagoger	Børn	Forældre
Dagmar Krognevang	Sigurd Fredriksen Jasmine Holmgaard	Annmarie Fredriksen, Dorden Fredriksen Lucetta Holmgaard, Otto Holmgaard
Cælan Mikkelsen	Lana Louritsen Gull Jensen Ida Mortensen	Sandie Louritsen, Michael Louritsen Elisabet Jensen, Mogens Jensen Edith Mortensen, Lennart Mortensen
Brita Olesen	Vita Jørgensen	Heidi Jørgensen, Isabella Jørgensen

Opret Ny Profil

Figure B.25: Profiles Department Manager 2

Graphical Mockups of The GIRAF Admin System

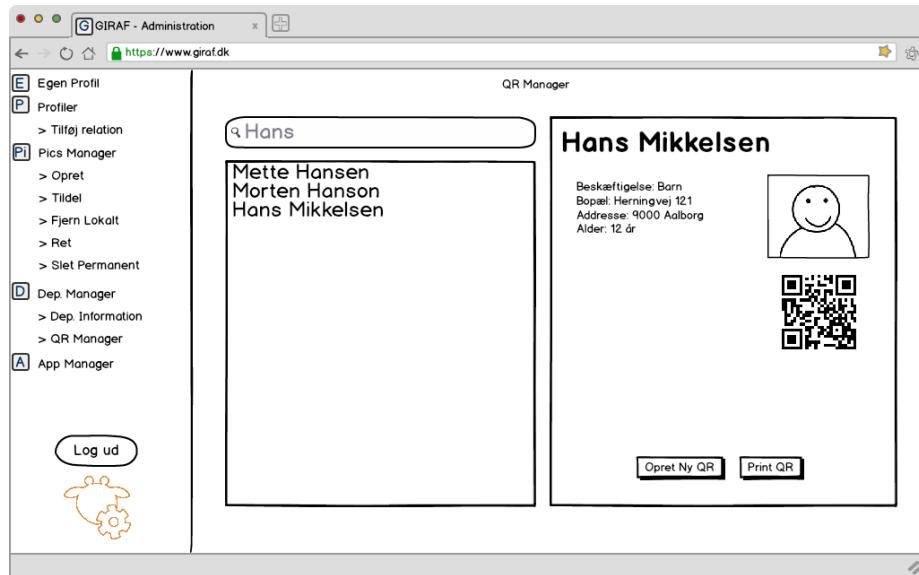


Figure B.26: QR Manager

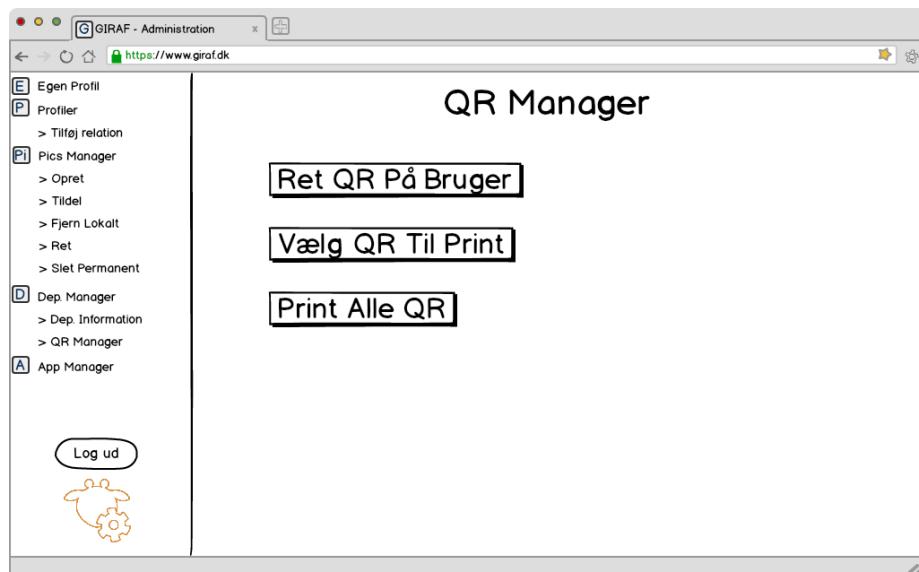


Figure B.27: QR Manager Frontpage

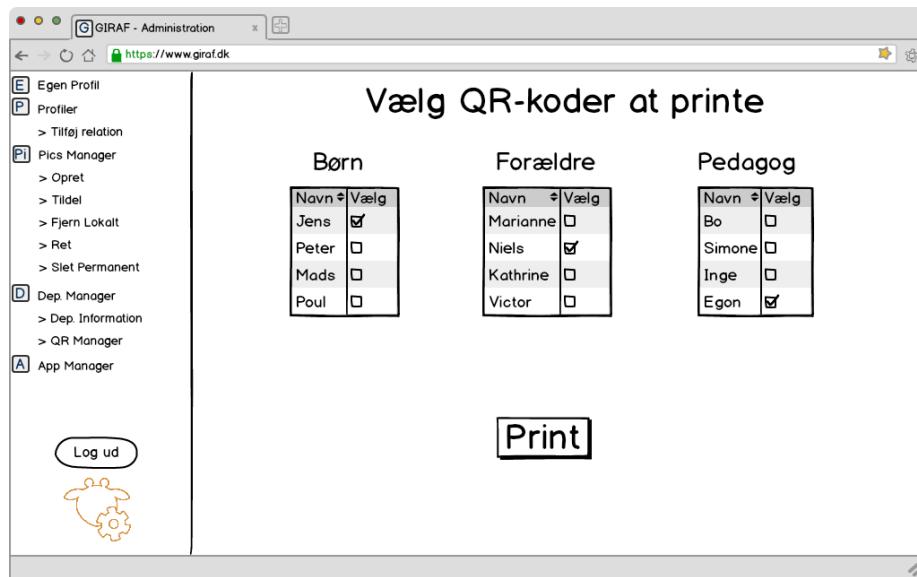


Figure B.28: QR Manager Pick

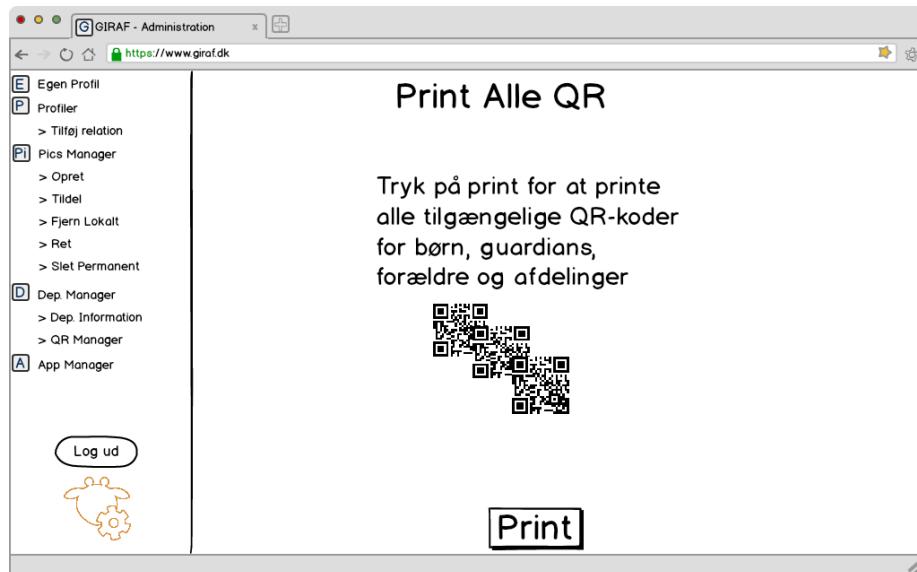


Figure B.29: QR Manager Print All

Appendix C

Transcript of Meeting with the head of the kindergarten (1/3-2013)

The questions:

A: Do the parents have a common contact list between each other, like the ones used in the Danish Elementary School?

B: How often do you change the children's pictograms? - As in all the way out of the scrap book

C: How many kids do a pedagogue take care of, on average?

D: How is the parents allowed to influence the daily life of their kids, when the kids are at the kindergarten?

E: Does the department manager have a list of the e-mails of each parent and are they easy to find? - If not, how easy would it be to gather them?

Answers to these questions:

A: The parents are not allowed to know anything about each other or the other parents kids.

B: It doesn't happen that often, but they still need to be able to change the pictures really fast. Since sometimes the kids come in with a new jacket, and cannot put their jacket back on, or take it off because of the wrong visual stimuli from an old picture.

The same goes if the milk carton suddenly changes to a Christmas theme or such.

C: A pedagogue usually has the responsibility for 2-3 kids. - These are called "kontakt børn".

D: The parents have no influence. Actually if their kids cannot participate in outdoor activities because of illness or tiredness, the parents are told to keep

their kids at home if the plan for the day involves outdoor activity, (which they at Bækken do almost every day).

E: She has their e-mail, but they are not public accessible. This still makes it possible for her to send invites to the system. - We were informed that the registration forms for when a kid enters the kindergarten has been updated so that the parents are asked to fill in their e-mail address.

Information gathered for PictoCreator group:

- The text should not be stationary.
- There should be an option for drawing lines and removing parts of a pictogram. These two lines were derived from what she told us of their current pictogram editor system.

Possible future work:

1. A planning tool. The department manager uses a certain amount of time on scheduling the kids and pedagogues day. One of the most serious issues could be that one of her pedagogue had to take the day off, then she must make sure that all her schedules are taken care of. A feature that they could really use would be that of fixing an assignment for each second week.

This planning tool should also be able to print two different versions of the schedule. One for the parents and one for the pedagogues. - The parents are not supposed to know which kids do what, or with whom. They only need to know what general activities is taking place in the kindergarten.

She also suggested that the taxa schedule could be incorporated in this system.

Other observations:

"Profiler" - We noticed that the tab "Profiler" from our original design was of no use to anyone but the department manager. Since everyone else only had access to a certain list of persons, which we already displayed on "Egen Profil".

"Profiler" - Should be ordered in a table, where first the pedagogues name is, the next <td> should then contain the children they are responsible for, the next <td> should then contain the parents of this kid, and the last <td> should then contain other relevant persons or information about the kid. (<td> is the same as a cell in a table)

"Parent/Pedagogue contact" - They don't communicate all that much in person. Since a department can have kids from basically all over the country. The kindergarten does host some 'Parent, Come and See the Kindergarten Nights'. But besides that they only communicate via phone and a black book the kids bring to and from the kindergarten. It contains important information like, if the kid did something special today, or the reason why he arrived home with a new pair of pants on, and so forth. - The pedagogues always check this book when the kid arrives and so do the parents.

Transcript of Meeting with the head of the kindergarten (1/3-2013)

"Privacy setting" - The privacy setting private is known as "mine tavler" by the pedagogues.

"Privacy setting" - We need 5 different privacy settings:

Pedagogues only

Parents only

Guardians ("værger") - We need to confirm this phrasing with the head of the kindergarten

Department

Public

"Searching" - When searching through tags or categories, it should search on synonyms as well.

"Breadcrumbs" - There should be breadcrumbs on every screen, or if there is only one action on the screen, it should contain its title.

"Pictogram 'Edit' and 'Opret'" - We forgot to add category and tag adding to these functions.

"App Manager" - She found it intuitive to perform special settings on the apps in the app manager. - This could be an interesting design idea, but hard to implement.

Transcript of Meeting with the head of the kindergarten (1/3-2013)

Appendix D

Usability Testing Appendix

This chapter contains the introduction given to the test persons and the tasks they had to perform. This chapter is written in danish, as was the test.

Opgaver:

- Ændre egen profils information, her i blandt: Navn, mail, tlf., profil billede
- Opret nyt barn
- Opret barns forældre
- Opret pædagog
- Opret relation mellem forældre og barn
- Opret relation mellem pædagog og barn
- Ret information om afdeling
- Opret pictogram
- Opret kategori

Første Login:

Du har nu fået en helt ny bruger i Giraf systemet, og opdager at dit navn er stavet forkert og der mangler tlf. nummer og et profil billede.

Vi vil derfor bede dig om at opdatere følgende: Dit navn, dit tlf. nummer og uploadet dit profil billede.

Vi har lagt et billede på skrivebordet med navnet "profilBilledePige.jpg" og "profilBilledeDreng.jpg"

Nyt barn i afdelingen:

I morgen starter det nye barn, Thorsten Jensen, i afdelingen og han har derfor

brug for at få oprettet en profil. Hans forældre hedder Lotte og Mads Jensen. For at Lotte og Mads også skal have mulighed for at benytte systemet derhjemme skal de også oprettes i systemet.

Lottes e-mail: lotte.jensen@test.dk

Mads' e-mail: mads.jensen@test.dk

Vi vil bede dig om at oprette en profil til Thorsten, Lotte og Mads.

Hvems forældre er det?

Nu har Thorsten, Lotte og Mads fået oprette en profil. Men før at Lotte og Mads kan bruge deres profiler til noget skal de forbindes med Thorsten.

Vi vil derfor bede dig oprette en forbindelse mellem Thorsten og Lotte, samt en forbindelse mellem Thorsten og Mads.

Den nye Pædagog:

I dag starter den nye pædagog Henriette Poulsen også i afdelingen, hun skal have ansvaret for Thorsten.

Henriette Poulsens informationer:

E-mail: henriette.poulsen@test.dk

Tlf: 88 44 55 66

Add.: Mark Poulsen Vej 23 – Aalborg Øst 9220

Vi vil derfor bede dig om at oprette en profil til Henriette og dernæst oprette en relation mellem Henriette og Thorsten.

Ny Tlf. i afdelingen

I har fået besked ovenfra på at skifte jeres telefon nummer i afdelingen. I har i dag modtaget det nye telefon nummer.

Det nye nummer er: 87 78 23 44

Vi vil nu bede dig om at rette informationen om jeres afdeling så den passer.

Pictogram af de nye shorts:

Thorsten har lige fået nye shorts og I er blevet tilsendt et billede af Thorsten hvor han er iført sine nye shorts.

I vælger for at være på forkant med Thorstens pictogram kartotek at oprette et pictogram af hans shorts.

Vi vil derfor nu bede dig om at oprette et pictogram som hedder Shorts.

Billedet til pictogrammet kan findes på skrivebordet: "shorts.jpg"

Introduktion:

Velkommen til testning af GIRAF admin systemet.

Vi vil stille dig en række opgaver, skrevet ned på papir, som vi gerne så dig udfører på den fremstillede PC. Testens formål er for os at drage indsigt i din måde at arbejde på, vi vil derfor bede dig tænke højt imens du udfører opgaverne. – Vi vil her gøre opmærksom på at intet er for småt til at vi gerne vil hører det.

Igennem testen vil <navn> sidde med dig i testrummet mens du udfører op-

gaverne. Hvis du har spørgsmål til opgaverne vil <navn> hjælpe dig. Han vil også give dig den næste opgave når du har gennemført den du var i gang med. Vi optager testprocessen på film og har derfor brug for at du skriver under på at dette er okay. <Udlever underskrifts side og giv tid til at underskrive>

Appendix E

Colour Theme

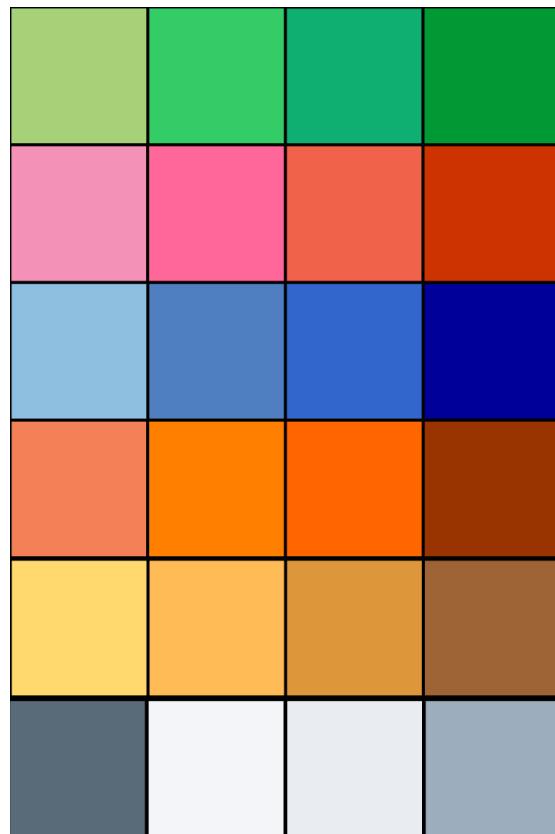


Figure E.1: Colour theme for the GIRAF system.