Aalborg University

Bachelors Project

Developing Software for Autism Institutions

28/05-2014

Group SW603F14:
Dennis Bækgaard
Benjamin Hubert
Michael Lausdahl Fuglsang
Anders Bender

Supervisor:
Xike Xie

AALBORG UNIVERSITET
STUDENTERRAPPORT

The Faculties Engineering,
Science and Medicine Software
Selma Lagerlöfs Vej 300
Telephone +45 9940 9940
Fax +45 9940 9798
http://cs.aau.dk

Title:

Developing Software for Autism Institutions

Theme:

Programming

Project Period:

Bachelor, Spring Semester 2014

Project Group:

SW603F14

Collaborators:

Dennis Bækgaard
Benjamin Hubert
Michael Lausdahl Fuglsang
Anders Bender

Supervisor:

Xike Xie

Report Page Count: 55

Appendix Page Count: 16

Total Page Count: 71

Source Code: See preface

Abstract:

People diagnosed with any form of autism have many problems in their day to day lives. Many tools have been developed to help overcome these, but most are currently analogue and could be improved upon if made digital. The ones that are digital only help small areas, but there is no all-in-one solution.

This digitalization is, and has been, the project for several years of all SW6 students. In the implementation of this software suite on Android and a web platform, we have been given the task of creating a GUI framework, to ensure common looks throughout all apps. We develop such a framework with focus on usability, both for the user as well as the developers using the framework.

In addition, we develop a web-based week scheduler for 55" touch monitors, similar to the analogue version in the institutions.

The framework is used by all applications, some more than other, and is contentiously adapted to changing requirements. The week schedulers front-end is completed, with functionality left to whomever picks up the project next year.

Signatures:

_____

Dennis Bækgaard

_____

Benjamin Hubert

_____

Michael Lausdahl Fuglsang

_____

Anders Bender

Preface

Source Code

Due to place constraints on the title-page the source code is listed below.

- giraf component: http://cs-cust06-int.cs.aau.dk/git-ro/giraf-component.git
- giraf component web: http://cs-cust06-int.cs.aau.dk/git-ro/giraf-component-web.git
- Web Scheduler: http://cs-cust06-int.cs.aau.dk/git-ro/web-scheduler.git

To get the source-code use git, `http://git-scm.com/downloads` and pull the master branch on the repositories listed above.

Notes

On this semester we, the groups, were required to read all reports (split in groups). If you, the reader, is part of a group who has taken over this project, we do not recommend reading the entire report. Instead **do read** Project Reflections, Part VII Section 2, it will contain our reflections and pointers specifically for further development.
The repositories used are giraf-components, giraf-components-web and web-scheduler.
Good luck.
Reading the report code will be mentioned and will be by using inline code like `this`.
Each of the sprint part will contain a sprint evaluation. In this sprint evaluation a list of the sprint goals will appear, and it will contain information of whether the goal was reach in following manner:

✓ Reached

✗ Not Reached

✓/✗ Partially reached

We would like to thank our supervisor, Xike Xie for good supervision, as well as the customers for good collaboration.

Contents

Part I

Role in the Multi-Project

1 | The Multi-Project Organization

For the groups to get organised a first meeting was established on the 12th of February 2014. On the meeting different points were decided:

- Which tools should be used
- Who is in charge of these tools
- What the different groups should work on

To maintain structure among the large amount of persons on the SW6F14 a student-board where created to help enhance the knowledge sharing between the groups. The project tools decided on were Redmine and Git. Furthermore there is a specialist in the group. This person has responsibility for the group-wide usage of *GIT*, which is the version control system used, and the Redmine *Issue Tracker*. If anyone has problems with either of these, it is expected for the specialist to help resolve them. This is the job of the specialist throughout the entire duration of the project.

2 | Sprint Reading Guide

At the beginning of each sprint, each group is assigned with a giraf project. These sections explain the role of this group for each sprint.

2.1 First Sprint

In the first sprint, the group was assigned to the task of creating a common framework for the *Graphical User Interface* (GUI). This project is called *Giraf Components*, or *GComponents* for short. *GComponents* will be used throughout this report. The reason for this framework was to use a common set of GUI components, such that the appearance of all applications in the entire GIRAF project would be the same.

The main task of the group was to refactor/rework the current GComponent framework and make any changes to the components that were necessary for the use of all groups. In addition it was to create a comprehensive guide such that the components would be easy to use by any other group in the future.

2.2 Second Sprint

At the meeting for the second sprint, the group was assigned to the GUI project once more (this was also the groups wish in order to continue the work). For this sprint the goal was to have issues/features reported by other groups in the first sprint completed, adding new functionality as well as adding documentation and guides for the usage of these. These goals are further detailed in Part IV Chapter 2.

The group also spent time helping other groups with issues regarding GComponents, which was not resolved by the documentation for the component.

2.3 Third Sprint

In the third sprint, the group is assigned a new project as well as maintaining support and development of `GComponents`. This new project is a web-based week scheduler that will function similar to the Week-Planner in the Tortoise project. The reason, explained in detail in Part V Section 3.3, for developing this as a web-based project is that the customers have a 55" touch monitor which can be connected to a PC[1].

The goal for this sprint was to finish up the unfinished components from the last sprint and analyse the web-scheduler and create a view for it. The group also took in requests from other groups.

2.4 Fourth Sprint

For the last sprint, the group decided to focus mainly on the week scheduler project, whilst not neglecting `GComponents`. The week scheduler took form this sprint, both in terms of its GUI as well as featuring some functionalities. This development included a visit to Birken, in order to see and test on the big touch screen. Minor additions were be made to `GComponents`, and support was provided.

The goal of this sprint was to create a prototype of the week scheduler and test it at Birken, as well as create `GLayout`.

Part II

Information Sharing

Since the GIRAF_Components project assigned to the group is a framework for the other groups to use, documentation is needed so that the other groups will use the framework correctly. This includes both code examples as well as general documentation. Two ways are considered:

- An automatically generated documentation, for example using Doxygen[2].
- Manually written documentation.

As described in Part I Section 1, we are using the project management tool known as Redmine. Here, especially the feature of a per-project wiki will be of use. It is decided that the best way for the group to efficiently share its knowledge with others is to manually write a wiki page detailing the usage of each GUI element.

A step-by-step guide for importing the framework in a project is on the front-page of the wiki. Below this will be a wiki page for each component in the framework as seen in Appendix A Figure A.1.

The wiki page for each component will include a section for:

- The purpose of the component.
- Its constructors, methods and how to use them.
- An example implementation and how it looks.

This is done with the inspiration of MSDN[3], Microsoft's documentation of the .NET framework. An example can be seen in Appendix A Figure A.2.

When other groups have trouble with the GUI elements, even after reading the wiki page for said element, we will try to assist them with their issue. This may result in a missing feature, or a bug being discovered, or that the wiki page is unclear for certain usages. This allows for continuously improving the wiki.

Furthermore Redmine also has an issue tracker and forums. Redmine enables others to add issues to our issue tracker, browse our wiki, and communicate with us using the forums. The forums for our project will receive updates in form of a change-log that the other groups can watch.

This change-log will feature which components have been added/updated as well as which wiki pages are added/updated. The forum's *watch* feature allows the interested groups to receive email notifications when new updates are posted.

Part III

First Sprint

1 | Analysis

The analysis will focus on the platform and environment the software is developed for as well as a review of the previous years work.

1.1 Runtime environment

Due to the existing applications are developed for Android, the runtime environment is the Android OS[4]. The programming language will be Java.

The Android version is Android 4.0.3, as the customers' tablets run this version. In order for the project to run on Android 4.0.3, the minimum SDK level has to be set to 15[5]. The previous semester used SDK level 19 as target SDK level, this is kept.

The target hardware for this project are tablets running Android. Other Android devices may be unintentionally compatible.

1.2 Development Environment

Previous semesters used the Eclipse *IDE* (Integrated Development Enviroment) with the *ADT* (Android Development Tools) plugin. For this semester, however, it was decided, at the first meeting[6], the project should migrate to Android Studio as this is an IDE designed for Android development.

Android Studio is based on *JetBrain's IntelliJ*, an IDE for Java development[7]. Android Studio has basic features including syntax highlighting and code completion, and more interesting features such as the XML viewer specifically designed to mimic an Android device. It also has the ability to compile and install directly on an Android device or emulator. The emulator is provided by Android and is included in the IDE[8].

1.3 Preliminary research

The preliminary research is concerned with the previous work, and as such attempts to establish an understanding of the current state of the project. To do this, the following questions are answered

- Which GUI elements are used throughout the already created apps?
- How much of the code in GComponents is reusable?

In order to establish a knowledge base in terms of the requirements for the GUI framework, code is briefly reviewed in all projects in search for GUI elements. These are noted down on a list, listed in Appendix B, which serves as a rough catalogue of components which needs an equivalent in the GComponent framework.
Secondly an analysis of the GComponents' existing code is made to establish how much of the code is "reuseable", as well as how many components there have already been implemented. The following list shows all components currently in the framework:

- GButton
- GCancelButton
- GIconButton
- GVerifyButton
- GColorAdapter
- GDialog
- GList
- GProfileAdapter
- GTooltip
- GWidgetCalendar
- GWidgetConnectivity
- GWidgetLogout
- GWidgetUpdater

The difference between the already created components and the lists of components used in other applications will serve as a guide to what needs to be added to the GComponent list. The analysis revealed that several essential elements like checkboxes, grids and radio buttons are missing in the GComponent framework. Furthermore it was discovered that code base in GComponents can be improved upon, since the code is unstructured and is missing comments.

2 | Sprint Goals

The primary goals of the first sprint are to refactor/rework the existing GComponents framework.

Furthermore, creating documentation and guides will be required such that other groups can use the framework as easily as possible.

In summary, the goals are as follows:

- Refactor/rework the GComponents framework
- Create documentation and guides for the use of these components

3 | Development

A requirement given by the clients is to make a default theme which is black and white, see Appendix C.1.1. Further they want to be able to change the theme colors according to a specific child. For GComponents this requires the theming to be dynamic.

3.1 GStyler

The purpose of `GStyler` is to supply a set of methods to aid in styling the various components in the framework. This includes various base colors of the components. Currently `GStyler` contains:

- `buttonBaseColor` a white color
- `dialogBoxBaseColor` a white color
- `listBaseColor` a white color
- `listItemBaseColor` a white color
- `toastBaseColor` a black transparent color
- `toastBaseTextColor` a white color

It also contains the following methods:

- `scaleDrawable()`
- `calculateGradientColor()`
- `dpToPixel()`

The intention is that in a later sprint, this will be the class that is used for customizing colors, such that whenever a component is styled, it will get the base color from `GStyler`. This allows for easy change of the colors, in one common place, rather than hardcoding many in the components them self.

3.2 GButton

In order to get a more general button in GComponents, `GButton` and all its derivatives is refactored. Currently `GButton` is deriving from `Button` and is simply a static restyling of it. There is also `GIconButton` which is derived from `GButton`, however, it can contain an image as well as text. Further there are two buttons deriving from `GIconButton`. These are `GCancelButton` and `GVerifyButton`. Instead of having a button for pure text and one for text and an image, the two will be merged.

`GIconButton` contains a method to scale an image so it fits the button. The scale is calculated manually. Instead of this, it is refactored into using the native Android method `createScaledBitmap()`. Furthermore it is moved to `GStyler`, such that it may be used by other components as well.

Previously the image was set with a call to the method `setIcon()`. To simplify the framework, the image can now be set through XML. To do this, simply using the native `android:drawableLeft` will suffice. Images are scaled to the size of the button as well.

The yellow gradient on the `GButton` is changed to a dynamic gradient calculator located in `GStyler`.

3.3 GDialog

During research a bug was found in `GDialog`, besides the obvious fact that it does not following the general UI theme. When the image is too small the descriptive text is hidden behind the buttons, which is not intended behaviour. The fix was a simple XML update.

The component is also changed from loading a static XML layout, to instead use the flexible approach of using `GStyler`. It still loads a layout, but the colors are set through `GStyler`.

3.4 GList

The main task for `GList` is to refactor it from using a default styling to use a dynamic approach like the rest of the components. Background and border colors are set using `GStyler`.

The previous version of `GList` had problems when working on older versions of Android which resulted in `GList` becomming all black [9]. It was discovered that hardware acceleration is not supported on tablets with an older version of Android. To solve this *Software Rendering* is forced.

3.5 GToast

The `GToast` component is not present in the old code base. It originates from a request made by another group, and is essential enough to be developed this sprint. Due to this it is of higher priority than other components.

The Android `Toast` component is a short notification popping up at the bottom of the screen. It is usually used for small notification messages such as "Saving..." or "Press back to close the window".

`GToast` derives from the existing android `Toast` class. It is, however, limited in functionality in order to control the styling of the component using `GStyler`.

4 | Sprint Evaluation

The sprint goal for this first sprint, was as mentioned in 2. The goals were fully not met, but was met to following extend:

- ✓ /✗Refactor/rework/develop a base framework containing essential components
- ✓ Create documentation and guides for the use of these components

In this sprint we underestimated the time it would take to refactor and improve upon the code base. This combined with the fact that we had no previous understanding of Android resulted in the goals only being partially met. The following components has been refactored/improved/developed:

- `GButton`
- `GDialog`
- `GList`
- `GToast`
- `GStyler`

`GToast` is a new component that was developed due to it being requested by another project using the GComponents framework. `GStyler` is also a new class, which holds all the color information of the components in the framework. We intend that in a later sprint, `GStyler` will be used to create a per-child theme feature, but before that can happen the rest of the components also have to be refactored/reworked. The components refactored/improved/developed so far are ready to use by any group using our framework. Furthermore, all these components, except `GStyler`, have had their wiki-pages created to document their usage. `GStyler`

Part IV

Second Sprint

1 | Analysis

The group is assigned to the same project as in the first sprint, the GUI project. As such the problem domain will remain the same. The problem domain is described in Part III Chapter 1.

1.1 Differences from Last Sprint

In the student-board meetings a common significant part of the agenda is to report the status for each individual groups project. Through this it was learned that much of the first sprint was delegated to refactoring code and fixing bugs from the previous year. An example of this can be seen in the summary of the student-board meeting of the 25th of February 2014[10]. The expectations for the coming sprint was to develop new features and implement new features on a working code base. Due to this, it is expected that this sprint contains more requests from other groups as they start to implement more GComponents components as well as discover they need elements which do not yet exist. It is therefore necessary to find a reasonable way to handle these requests.
There are various options which can be grouped:

- **E-mail, Redmine Forum**
  - *Pros: Easy to use for the requesters, merely send a mail with what you need*
  - *Cons: Hard to maintain a standardized form which requesters are to follow. Can be hard to get a full understanding of the request from the text*
- **Google Forms, Redmine Issue Tracker**
  - *Pros: Somewhat easy for the requesters, they have to follow a more rigid form than what they could in emails. Easy for us to formalize*
  - *Cons: More work required from the requester. Can be hard to get a full understanding of the request from the text*

Further we expect many requests to follow the trend of communicating closely across groups and projects, resulting in requests being made in-person through a short conversation between our group and the requester.

We want to track all requests directly on the issue tracker, so any request regardless of where they come from will be formalized into an issue on our project page on redmine. This means that groups can either directly add their requests in our issue tracker, or the issues can be created as a result of a conversation between us and the requesters.

1.2 Customers

Additional requirements have been made by the customers, which have been compiled into a list of requirements by the groups in charge of retrieving these. As can be seen in the excerpt of these requirements in Appendix C, support must be added for creating and saving a theme for each app, for each profile. Additionally each application should per default be black-and-white.

2 | Sprint Goals

As the goals from the first sprint was not met, the development on the GComponents components will continue. The components which are the goal for completion for this sprint are:

- `GDialog`
- `GTooltip`
- `GToggleButton`

Furthermore development on the theming system will start. The theming will enable the user to define their own color scheme for the applications.

Additionally it is expected that more requests is received, as the framework will be used by more groups in this sprint. These requests get highest priority, since these requests are wanted by other groups and our own list is a general list of components which are nice to have.

Finally it is still the intention to update the wiki to document new components.

This gives the following sprint goal list:

- Development of GComponents
- Theme-system development
- Develop component requests from other groups
- Further addition of documentation to the wiki

3 | Collaboration

3.1 Component Requests

In this sprint it is expected that GUI features are requested by other groups. Therefore a way to formalize these requests is required. The reason for this is for the documentation but also to ease the structuring of the requests. This is done with the issue tracker[11], which is available for all the other groups. The issue tracker makes it easy for both our group and the requesting group to follow the progress of the request. This includes which components have been released, the progress on each component and the priority of each components. Examples of such requests can be seen in Appendix G.1, following the formula described in Part IV Section 3.2.

3.2 The Request Formula

A request is formalized together with the requesting group, such that the actual formalized request reflects what is really needed. The requesting group will explain what they need, preferably through a prototype and a description of the functionality they need. It can also be old components they need updated to GComponents with equal functionality. The requirements are logged to the issue tracker. This is done by creating a new issue which contains a general description of the request, including:

- The requesting group's number
- Their group room location
- What project they are working on
- The functionalities they requested

Furthermore a sub-issue is added for each component and another sub-issue for creating documentation, this is illustrated in Figure 3.1.

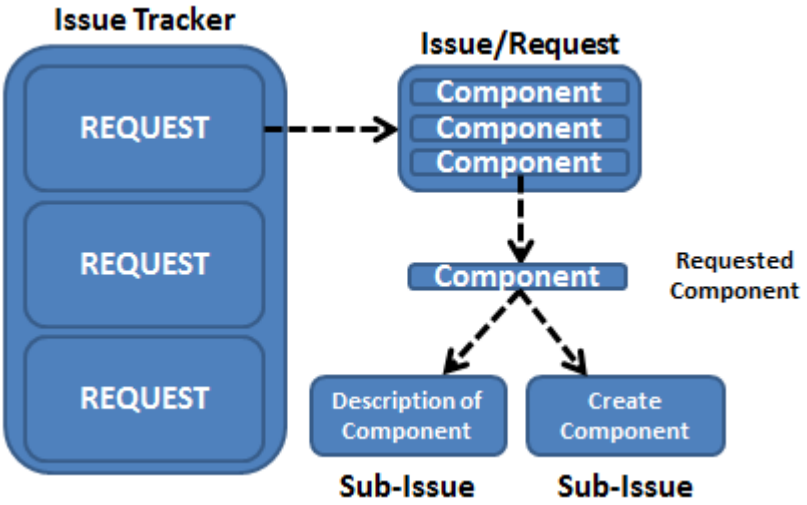

Figure 3.1: The flow of how requests is handled in the issue tracker

4 | Development

This chapter contains the continued development of the GComponents framework, this includes new components, as well as maintaining the old ones. Following is a list of all new and altered components as well as new features in the framework. A chosen subset of these will be described.

- GButton
- GButtonPlay
- GButtonSettings
- GButtonTrash
- GCheckBox
- GComponent
- GDialog (described in Part IV Section 4.1)
- GDialogAlert
- GDialogMessage
- GProfileSelector
- GGridView
- GList
- GProfileAdapter
- GRadioGroup
- GRadioButton
- GSpinner
- GSpinnerAdapter
- GStyler
- GTextView
- GToggleButton (described in Part IV Section 4.2)
- GTooltip (described in Part IV Section 4.3)
- GTooltipBasic

4.1 GDialog

The `GDialog` version from the previous sprint is refactored and renamed to `GDialogMessage`. This is done because many groups had specific uses for the

dialog which the old `GDialog` could not satisfy.

A new base class is created. This base class is essentially a frame around a `View`. This `View` can be anything, as it is supplied by the user, either in the constructor or through a method call. This is illustrated in Figure 4.1, where the area marked green is the `View` which holds content. This increase the flexibility of the `GDialog` as requested.

Figure 4.1: Illustration of `GDialog`. The area marked green is a `View` intended to hold content.

4.1.1 Derivatives

The following classes are derivatives of `GDialog`, and utilize the `View` to hold their content. These serve as a default set of standardized dialogs. The default versions are ready to use without having to alter the `View`.
The specifications for the dialogs:

- `GDialogMessage`, contains a `GVerifyButton`, whose functionality is determined by a callback function, a `GCancelButton`, two `TextViews`, and an `ImageView`.
- `GDialogAlert`, contains a `GVerifyButton`, whose functionality is determined by a callback function, two `TextViews`, and an `ImageView`.
- `GProfileSelector`, contains a `GList` coupled with a `GProfileAdapter`↩ which holds all profiles, a `GTextView` which displays the name of the currently selected profile, and a `LinearLayout` for the profile picture of the selected profile.

4.2 GToggleButton

A toggle button is requested by group *SW602F14*. The functionality they request is a button that, when pressed, toggles on and off. They also need to be able to get the current state of the button.

To do this a new class is created, `GToggleButton`, which extends `GButton`. The reason for not inheriting from the native `ToggleButton` in android, is that it would require either redundant code or an auxiliary class for both `GButton` and `GToggleButton`.

A method `isToggled()` is implemented to get the current state of the button. Furthermore, it is possible for the developers to respond to when the button is pressed. This is set through either a function call or XML.

4.3 GTooltip

`GTooltip` is quite similar in regards to the way it is used code-wise to the `GDialog`, described in Part IV Section 4.1. Its use, however, is to provide information to the user, rather than to have the user make a decision. It is an entirely new class, as something like this is not present in the default Android library. The `GTooltip`, like the `GDialog`, is essentially a frame around a View, as illustrated in Figure 4.2. Tapping somewhere on the screen outside of the tooltip will hide it.

Figure 4.2: Illustration of `GTooltip`. The area marked green is a `View` intended to hold content.

4.3.1 Usage

When instantiating a `GTooltip`, the first argument you parse is an Android `View` to which it will anchor itself. That way users of the framework need not worry about the placement of the tooltip, as it will do this automatically according to the view it has been provided with. It will attempt to be closer than the view to the center of the screen, and it will point its arrow towards the view, as illustrated in Figure 4.3.

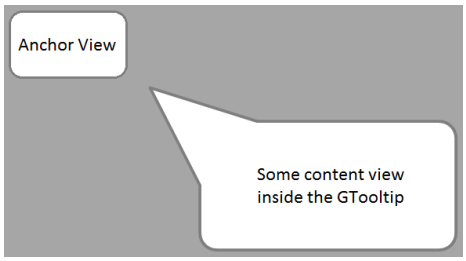

Figure 4.3: A `GTooltip` anchoring to the `View` *Anchor View*, placed in the top left of the screen.

To anchor the `GTooltip` correctly to the view, two reference points are established. These points are located at the vertical center of the screen. One is located at a third of the width from the left and the other is two thirds from the left. From these two points, six regions are created as illustrated in Figure 4.4. The region in which the `View` is located is then calculated. Using this information, the `GTooltip` can now be placed in accordance with the `View`.

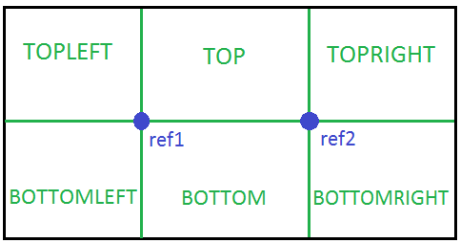

Figure 4.4: `ScreenRegion`s. The six regions are calculated using the two points, *ref1* and *ref2*, marked with blue.

4.3.2 GTooltipBasic

The `GTooltipBasic` is a derivative of `GTooltip`, and features an image and a scrollable text. If an image is present, the number of lines the image uses is calculated using the size of the text and the height of the image. This is used for wrapping the text around the picture. The text itself is a `GTextView`, and as such has the capability of scrolling should the text take up more space than what the tooltip allows.

4.4 Theming

According to the requirements in Appendix C, and more specifically the requirements in Appendix C.1.1, the clients want to be able to choose a theme for each application. This theme further has to be customizable for each child, as some

of them are biased positively and negatively towards some colors.

Functionality already exist in the launcher to color an application. In order to figure out how this was implemented, a meeting was held with the group which has the launcher project, group *SW605F14*. A flowchart illustrating the current implementation is show in Figure 4.5.

Figure 4.5: Old version to get an app colored

The inherit problem with this approach is that there is no actual way to set the theme/color for a specific child. In order to fix this, a new meeting was held with the launcher group discussing the approach in Figure 4.6.

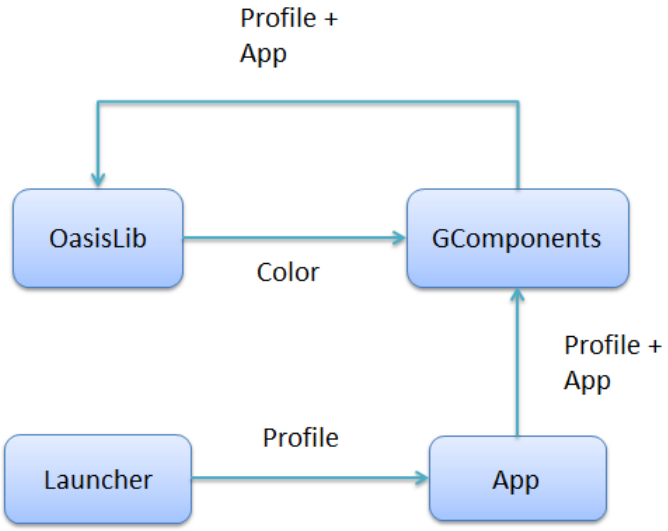

Figure 4.6: New version to get an app colored according to a child

The flowchart shows that instead of sending the color to an app, it sends a profile, which is the profile currently logged in. The app will then relay this information to a new component in GComponents, fittingly called `GComponent` along with the name of the app actually being launched. With this information GComponents can make a lookup in the database table where this information is saved, and find the appropriate color to the app opened, for this particular user. One problem with this, however, is that the colors of GComponents *must* be set prior inflating any UI elements. In order to do this, it is required by all apps to initialize GComponents in their first activity.

5 | Plan Change

The requirements unexpectedly changed with regards to the theming. The customers no longer care about the ability to be able to change the color theme of each application[12]. It was deemed unnecessary and as such the work already done is now redundant. The functionality is kept however, but with a different flow. Instead of `GComponent.initialize()` which takes an app id and a profile id, a new method `GComponent.SetBaseColor()` is implemented. It simply takes a color in the form of an integer, and sets that color as the base color for all components. All components in the framework use this base color to calculate their own colors. `GComponent.initialize()` is removed. The new flow can be seen in Figure 5.1.

Figure 5.1: New theming flow after requirement change.

6 | Porting to Web

During the sprint a large request has come from group *SW606f14*. They are making a web-based administration interface where the guardians can see and alter settings for all children. It is meant to run on a computer rather than a tablet for easier use. Seeing as they wish to have a GUI resembling what is on the android devices, our role in this is to port the GUI elements to a web-based version.

6.1 Analysis of port

It is not trivial as to which elements should be ported to web. The android GUI framework emphasizes as much on layouts (`GGridview`, `GList`) as elements such as buttons and dialogs. This is not the case with web development. In web development the layout is much more freely defined by the developers. For this reason the list of resources to be imported is:

- `WGButton`
- `WGVerifyButton`
- `WGCancelButton`
- `WGToggleButton`
- `WGRadioButtons`
- `WGCheckBoxes`
- `WGDialog`
- `WGToast`
- `WGTooltip`
- `WGTabs`

These are mostly actual components, however, there is one layout specific element added which is `WGTabs`. It is been supported because it is not a trivial layout to make for web, especially if it has to have the same feel as the rest of GComponents.

6.1.1 Work flow differences

One of the main strengths of android is that most of the UI is handled in android. That is, how `LinearLayout` works is defined in android and as developers you do not need to touch this definition. In web, non of these are defined. All classes and styles are made for a specific website. This limits one particular feature that there is in the android version of the GUI, which is quick and dynamic theme changing. On android we can compute all colors from one base color. In the web version, the entire theme is composed of many files, images and loads of css classes. One of these bundles is one, static, theme. In order to have multiple themes, they must first be created. This is done easily by using the theme roller[13] on jQueryUI's website - this, none the less, has to be done every time. We choose to make one theme for the web version, which is based on the default theme used in the android version.

6.2 Choosing an implementation

For developing the actual web-version, a couple of different solutions are explored. The option of creating everything from scratch is rejected, due to the fact that there are a lot of really good frameworks created for this particular task. The chosen framework is jQuery[14] with an additional framework called jQueryUI[15]. The benefit of using this framework is that it is thoroughly tested, complies with new HTML5[16] and CSS3[17] standards, as well as it is cross-browser compatible with most desktop browsers and mobile/tablet browsers. On top of that it has graceful degrading when older browsers are used. Further using jQuery and jQueryUI makes development of the port much faster, which is a priority.

6.3 Implementation

The implementation of the framework is limited. However, due to the sheer size of jQueryUI much work was put into selecting relevant features to use. Additional documentation in form of several wiki pages contains *how-to*'s of all the components available for the web GUI.

7 | Sprint Evaluation

The second sprint was set with a modest sprint goal, because the possibility of a lot of requests was very high, as mentioned in Part IV Section 1.1, and as it was learned from the first sprint refactoring/reworking/developing components can require a good amount of time.
The sprint goals were met as following:

✓ Development of GComponents

✗ Theme-system development

✓/✗ Develop component requests from other groups

✓ Further addition of documentation to the wiki

The goal of completing `GDialog`, `GTooltip` and `GToggleButton` was met, but the initial goal of developing a theme-system was abandoned. This was due to a new meeting which revealed that the customers had no desire for the theming feature. This gave some extra time and it was therefore decided to expand the GComponents to contain web-components too.

This decision was made when a request from sw606f14 was received and the theming system was abandoned.

Furthermore the following requests were implemented and documented:

- GGridView: New component
- GButton: A drawable should be able to be centered
- GSpinner: New component
- GCheckbox: New component
- GRadioButton: New component
- GButtonTrash: New component (generic trash button)
- Port GComponents to Web: make the theme available for web applications

The GComponents request list was almost empty towards the end of the sprint, with the remainder mostly added at the end of the sprint. The following requests are left for the next sprint

- GDivider: Used to divide sections of the screen
- GSwitch: A two-state switch with both states shown, an active and an inactive
- GSlider: Allows the user to choose a value from a range
- FancyListSelector: Tabs whose background 'melt together' with the selected item
- Abilwarna: Make Abilwarna a part of GComponents, which allows the user to select a color from a palette

To get an overview of the usage of the GComponents the other groups were asked if they were using GComponents, and it revealed that *11 of 16* groups are using GComponents. Two of the groups not using GComponents are iOS so it makes perfect sense that they are not using the Android library. One of the groups not using GComponents is a database group, so they are not using a GUI. Lastly, as our group has no application project, we are not using it ourselves. As such the adoption of GCompontions from the other groups has been a great success.

Part V

Third Sprint

1 | Sprint Goals

The goals for the third sprint will inherit what was left over from the second sprint in terms on uncompleted requests, see Part IV Chapter 7). The group has taken on an additional project in the form of a week scheduler, and prioritization between this and GComponents will be required. New requests will still be accepted and their priority will be evaluated in accordance with the goals for this sprint.

Given the group has two projects this sprint, it is required to prioritize the time spent between the two. For this sprint, the prioritisation will be determined by two things:

1. Other groups dependency of a request
2. Time frame of individual tasks in accordance with sprint goals

If another group requires a component, or a modification to a component, to continue their development, it will be of the highest priority. However, if it is a request for cosmetic changes, or similar non-crucial changes, it will be of low priority.

As such, the sprint goals are the following:

- Finish requests remaining from sprint 2
- Accept new requests and prioritize appropriately
- Analyse week scheduler
- for week scheduler

2 | Development

In this sprint, a lot of components have been updated and added. An example of a request made this sprint can be seen in Appendix G.2. Below is a compiled list of components modified this sprint:

- Added - GSwitch
- Added - GSeekBar
- Added - GColorPicker
- Added - GSelectableContent
- Added - GDividerHorizontal
- Added - GDividerVertical
- Added - GHorizontalScrollViewSnapper
- Added - GVerticalScrollViewSnapper
- Updated - GStyler
- Updated - GToggleButton
- Updated - GTextView
- Updated - GDialog
- Updated - GButton
- Updated - GProfileSelector

2.1 Updating Buttons

The groups task with developing a GUI framework is to provide a common look and feel throughout the different apps. This not only provides a similar code base for all apps, so they are easier to maintain, but it is a usability concern as well. Therefore, having a clean user interface is important. All components in the GComponents framework has been made individually, and designed to follow the theme individually. As more and more groups were using a reasonable part of the framework, it was found when the components were used in conjunction with each other, the visuals could be improved such that each component were more unique within the same theme. Due to this, the buttons design was updated to make them stand out more.

(a) The old button style

(b) The new button style

Figure 2.1

2.1.1 The Refactor

The change to the button class, was not as simple as first anticipated. The visual change to the button is a modification to the border, or stroke, of the button. It is now changed to be a stroke with 2 colors, as seen in figure 2.1b, from the old version with only a single color seen in figure 2.1a. Furthermore, the width of the border has been decreased to 2 pixels. This creates the effect of the button standing out as something you can press. Further it creates a less blurred view when using many components together, as they all shared the same border style.

2.2 Background Colors

In GComponents it was only possible to get one background type, which was a solid color. Having multiple types of background colors which follows the theme makes it possible to adhere to the theme, but still having diversity. This requirement resulted in a new method `GetBackground` which returns the following types of backgrounds:

- *Solid*
- *Gradient*
- *Gradient inverse*

It further makes it possible to easy update and add backgrounds to the framework, and they can be added as requests come in.

3 | Analysis of Week Scheduler

3.1 Informal Meetings

Since there are two other groups working on week schedulers, group SW602F14 on the Android version and SW610F14 on the iOS version, it makes sense have ours be consistent with theirs. The other groups have had their projects for multiple sprints, so to share knowledge of the scheduler and possible requirements a meeting was held.

3.2 Scheduler Requirements

The formal requirements for the week scheduler for the tablets are as written in the requirement specification document, which was created by the groups in charge of the requirement in Sprint 2.[18] The requirements:

- Citizen profiles must be able to mark how far in a sequence they are.
- Colors for the week schedule must follow the international color standard for days.
- It must be possible to turn on a feature, such that days are switched upon swiping to either side, when you are in the day tasks.
- This application must function as a tool to plan daily activities for citizens.
- This application must be able to help citizens to formulate their day in pictograms.
- This application must be able to allocate periods where citizens can choose between preselected activities.

- Guardian profiles must be able to define how many pictograms are shown at a time.
- Sequences of pictorgrams must have the same size and be locked in a grid.
- When a citizen has a choice to make, other functions must be locked, until a choice has been made.

The target of this application is a 55" Touchscreen that can be connected to a PC[1]. This leaves a lot of opportunities regarding the design and framework.

3.3 Platform

As the monitor can be hooked up to any PC, making the application work cross-platform is valuable so that it will work independant of OS. Two options are immediately available, a web-platform or java. Java is cross-platform and will run on most operating systems, as the code runs on the JVM. We also already have a code-base in java, but as there is little in terms of algoritms for this application, as it is mostly about the GUI, the code-base we have is largely unusable as it is specific to Android. Alternatively, using a web-application will enable it to run on anything with a browser, though cross-browser compatability will need to be considered. The choice here fell on a web-application, as it is the easier place of the two to make a custom GUI, as this is an essential part of any web-application.

Selecting the right foundation to build the web application on is important. Partly because we have to develop working content, but also because future developers has to be able to pick up the project and continue development. It was clear from the beginning that in order to support this a framework would be a good start. It has a large working code base which supplies functionality to deal with a large amount common tasks and problems in web development. Seeing the current server supports PHP, the application will be written in this, and the framework will have to support this.

There are two major categories of frameworks currently which are widely used in web development. The first category are *Content Management Sytems* such as Wordpress[19] and Joomla[20]. Common for these is the focus on developing websites from a template-like system where it is possible to use modules and plugins to achieve the functionality required, rather than making your own. It is, however, still possible. The other category are *development frameworks* like Laravel[21] and CodeIgniter[22]. They tend to be more focussed at developers seeking a framework of functions to build upon and achieve the functionality they require, rather than using already made plugins.

The best solution for the web-scheduler is the second category. During development we would like to have full control of the website, and develop all features independently, whilst still having a solid framework to rely on for many of the common tasks.

A good candidate for this is *Laravel*. It features new development techniques such as routing as well as complicated filters on routes. Further it has an extensive Object-Relational Mapping system, *Eloquent* which makes working with databases easier. Compared to *Codeigniter* Laravel is the continued development of the same ideas, just with more recent techniques and technology used.

Due to the time constraints of this project, we have chosen to select Laravel for our project. It has a comprehensive documentation which is good for future developers, as being a huge framework with many features to use for faster and better development.

4 | Sprint Evaluation

The goals for the third sprint are as listed in Part V Chapter 1. These goal are not met. The sprint goals for this sprint was met to following extend:

- ✓ Finish requests remaining from sprint 2
- ✓ Accept new requests and prioritize appropriately
- ✓ Analyse week scheduler
- ✗ Create view for week scheduler

The reason for this, is that the new requests made throughout the sprint, as well as the remaining requests from the previous sprint was too consuming in regards to completing every goal. These requests were prioritized higher as detailed in Part V Chapter 1, and as such the creation of a view for web-based week scheduler has not been accomplished. The tasks which required the most time was creating the new requested components, which are listed in Part V Chapter 2. A good foundation for the web-based week scheduler was made in the web analysis.

Part VI

Fourth Sprint

1 | Analysis

During the sprint end of Sprint 3, the customers continuously noted that they would like content, especially pictograms. They will like to have a number of states, which indicate a number of cases:

- Content should be highlighted when selected (GSelectableContent was already created in Sprint 3, as mentioned in Part V Chapter 2)
- Content, specific to schedules, should be cancelable
- Content should have the same delete button

Two options are available. Every application can implement a solution themselves, and cooperate to create a similar design, or it can be implemented through the GComponents framework. As the group in charge of the GComponents framework, we thought it made most sense to implement it here, as this would ensure a similar implementation across all applications. This potential new component, GLayout, should be able to handle all three cases. Requests to the GComponents framework are usually made by the application developers rather than the customers. To find out which solution to pick a survey is created. The survey, which can be seen Appendix D, has the following results:

- Interested: 5
- Not interested: 7
- No answer/not present: 4
- Total: 16

5 out of a total of 16 groups have an interest in getting this component from the GComponents framework, and the rest was either not present or had no use for it. Since 5 groups is interested in GLayout the decision is to create it - as an extension to the already created GSelectableContent.

2 | Sprint Goals

With the analysis for the week scheduler completed in Part V Chapter 3, development will now begin. This development should include a test to get more information about the usage of the week scheduler as well as a more detailed view of features the customers wants. The goal is to have a presentable view to show the customers. This was a goal from the third sprint that was not met, as described in Part V Chapter 4. Building on top of this view, the week scheduler should be able to use data created on the tablets. Without neglecting GComponents, the primary focus of this sprint will be on the week scheduler.

As described in Part VI Chapter 1, from the sprint end meeting of Sprint 3, a new component, GLayout, is to be made. Furthermore, we will be providing support for the groups using GComponents and complete simple requests.

This gives the following sprint goals

- Develop view of week scheduler
- Deploy a test at Birken
- Develop the model the of week scheduler
- Create GLayout component
- Provide support for GComponents

3 | Development

The web based week scheduler will be deployed as a prototype. It will be based on the analysis in Part V Chapter 3. It will be developed as an evolutionary prototype such that we can continue development on it after testing, compared to a paper prototype.

The prototype will attempt to fulfil as many of the requirements, listed in Part V Chapter 3, as possible. The development will focus on the front-end interacting with the user since the remote database is not yet able to synchronise with the local database. As such, it is not possible to retrieve the data from the Android week scheduler, which would be the ideal data to work on. Instead we will create relevant test data.

3.1 Making the Layout

The design of the week scheduler should look similar to the existing schedulers. The Android version, made by group *SW602f14*, will be the main inspiration since this has been developed for more than one year and is further in the development process compared to the iOS version. The most characteristic visual feature in the Android week scheduler, is the color convention from the international color standards[18] used for the various days.

The first task was to recreate a schedule layout complete with a profile selector. This was done with relatively new CSS3 techniques, such as `display: flex`, to cut down on workload at the expense of backwards compatibility. We were, however, told by the clients they had Windows 8 PCs, which meant they had newer browsers. All newer browser versions supports this CSS3 feature, and as such the backwards compatibility is not a problem.

It was a priority to prevent the page from updating in terms of a browser refresh, such that the page did not move as is often seen while content loads. This was, however, only partially obtained due to time constraints.

A general rule of thumb has been to provide clear and concise positive feedback when the child interacts with the system, such that they are never in doubt

in terms of what has happened.

3.2 Keeping in Touch with Touch-Screens

Although the application is web based, and developed on laptops, the target is a large touch screen. In order to make it feel natural to use the scheduler with a touch screen, all functionally was based on intuitive touch features. As such, javascript with jQuery is used to make pictograms *click-able* such that the child can ”complete” a task as seen in Figure 3.1.

Figure 3.1: Screen snippet of a checked pictogram signifying that the activity has been completed.

4 | Week Scheduler Prototype

The prototype, which can be seen in Figure 4.1, has the following features:

- Multiple profiles with individual schedules
- Colors following the color convention
- Highlighting of the current day
- Checkable pictograms (To mark when a task is done)
- Choice-pictogram - a pictogram which opens a dialog where a child can choose between multiple pictograms

Figure 4.1: Screenshot of the first week scheduler prototype, illustrating some of its features.

4.1 Prototype testing

The prototype was tested on-site at Birken, and all feedback is formalized in Appendix E. Most notably are the following features:

- Cancellation - One should be able to drag a cross onto a pictogram to cancel the activity
- Multiple pictograms in one slot - Sometimes activities require more pictograms, such as when a child has to play, but with another child. Then there will be a pictogram of "Play" and a picture of the other child.
- Dragable profile picture that can be dragged to each day

It was also learned that the touch screen is very sensitive to right-clicks, selections and zooming. Attempts should be made to fix this.

5 | Continued Development

The feedback on the prototype, which is formalized in Appendix E, provides a number of changes which the clients found desirable.

5.1 Improving Key Features

The most important aspect of the scheduler is to be able to show pictograms/-tasks in the same way as it is in the institution.

This meant representing pictograms on a schedule for each day. Further pictograms should be able to be adjacent to each other, for example if a child has to play with a specific person, then a pictogram of "Play" and a pictogram/picture of the other person would show in the same row. This is illustrated in Figure 5.1.

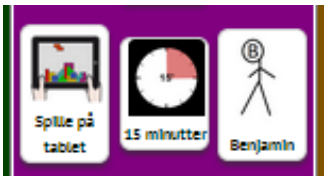

Figure 5.1: Screen snippet of adjacent pictograms symbolising playing with a tablet for 15 minutes with the friend Benjamin.

Two features were implemented:

- The picture of the child which can be dragged from day to day, signifying that the child has started the day
- The ability to cancel a task through the "cancel task"-feature, which is a red cross which is dragged over a pictogram to cancel it

5.2 Data Persistence

The state of each pictogram should be stored in database. This way the data is equivalent on both tablets and pc's as they are synchronised. Unfortunately the databases developed only synchronise from a local database on a tablet, to the remote master database, and not the other way. Due to this, it was opted in to use cookies to save the states, simply because the system should be rewritten once a group has finished work on the database synchronisation.

6 | Sprint Evaluation

The goals for sprint 4 outlined in Part VI Section 2 were partially met. The sprint goals were met as following:

- ✓ Develop view of week scheduler
- ✓ Deploy a test at Birken
- ✗ Develop the model of the week scheduler
- ✓ Create GLayout component
- ✓ Provide support for GComponents

In the sprint a lot of the time was used to support other groups when they had trouble with GComponents.

Even thought the web-scheduler view was created and uses a data model. This model is not based on the database though. This is because the group developing the database structure did not finish their work which we were dependent on.

The meeting with Birken went very well and while not all feedback features have been implemented, due to time restraints, the web-scheduler still had some significantly improvements. The following components were also updated/added to GComponents:

- added GButtonProfileSelect
- added GLayout
- added GMultiProfileSelect
- updated GSwitch

Part VII

Project Reflections

1 | Current Project Standings

This project consists of two sub-projects, the GComponents and the web based week scheduler.
The GComponents part of the project is almost up to date with the requests from the other groups, only thing left on the issue tracker is a request on a special list for Parrot/*Pikto-oplæser* and a refactoring of the elements using a normal `TextView` instead of the `GTextView`.

The week schedulers front-end have a lot of the high priority features which the costumers wants and, as mentioned before, at the last sprint end it also received positive feedback on the functionality and UI, but it has not been prototyped after the rework, which would be a good idea. The main task is still to make it functional which it is not since the back-end is far from finished, nor consistent with the database.

2 | Knowledge Sharing and Advice

This chapter is dedicated to future groups which will be assigned to either the GComponents project or the web based week scheduler.

2.1 GComponents

When working with GComponents it is important to acknowledged that different groups requires components at different times in the process. Which means that not only should there be a good amount of time allocated to assist people and create the components, but also try to get a feel for what the different groups are working on. More often than not our group had multiple project-repositories pulled and would look regularly in them. Because GComponents is a huge part of the user experience for most apps, if you know what is likely to be requested before it is requested, then it is much easier to predict work loads. Further, when a group comes with a request, if you are aware of their project at a basic level, and know where they inflate views, buttons and so on, then it will be much easier to relate to what they are requesting. Often times, people are not aware of the actual component-type they want, but only aware of what functionality they want.

A structured approach to tracking requests, and good documentation is essential. Our experiences, are quite unanimous in this regard; *people do not read forum updates or wiki pages.* This resulted in various groups not updating to the latest version of GComponents. Perhaps going around to the groups and notifying them that a new version of GComponents has been released upon major releases would be a good idea. That being said, keep writing on the forum and in the wiki. Furthermore it is important when creating a component to test the appearance of the component in an application context. That is, the component should get visualized inside the application of the requested group.

This is important because it is possible to make an element which looks good by itself but it might not look as good when applied to given application.

So in bullet point form:

- **Learn the Android life cycle!**
- **Read the GComponents wiki!**
- Test components in the application they are requested for to ensure that they look good.
- Make time in the sprints for taking in requests.
- Make sure to document new components and updated wiki pages.
- **Make sure the other groups are aware of the wiki pages and their function!**
- Make sure the other groups know when there's a new version of GComponents.

Lastly, if you are working with git the following year, make *absolutely sure to use development branches*. GComponents is used in almost all projects, if you push app-breaking code to the master branch, you will hinder development further down the pipeline. If someone needs a new component, make a branch with it. Make then pull that specific branch and test it. When it works, merge it into the master branch.

2.2 Week Scheduler

The week scheduler is a very important project to the customers at *Birken*, and we would therefore recommend that a group is assigned to this project from the beginning instead of being a side-project in Sprint 3 and 4.

About the project itself it is important to emphasize the importance of collaboration when working with this project. Since there will most likely be a dependency on the remote database's ability to synchronize from the local database to the remote database. There will also be collaboration with the customers since it is a good idea to test the scheduler as often as possible on the big touch screen.

The base of the Week Scheduler is quite solid. Unfortunately we were very much limited on time when working with it, and especially the two last days were extremely tight in terms of development. As such, some of the code is "make it work" code rather than "correct code". Most significant of this is the actual model used in the project. We were waiting on an update from the

database group which never was completed, and due to this we had to make a testable model to show at the clients and in the last sprint review. It is likely this model needs to be changed and updated according to the actual database. This refactor is likely to pull ripple-effects all the way through the code out to the view.

Lastly, the Week Scheduler is based upon Laravel 4 and jQuery. If you wish to develop rapidly and use the power of these two frameworks, we strongly recommend you to use some time at their respective documentations, it pays off.

If more time on this project was available the highest priority tasks would be to:

- Create back-end, enabling the customers to actually use the scheduler. It is very important that a collaboration is established with both the database groups and the other week scheduler groups to ensure coherence and compatibility.
- Refactor parts of the front-end, because some of the code can get more structured and reworked (namely *day.blade.php*).
- Create multiple schedule views as requested in Appendix E.
- Create an admin/edit mode in which the schedules can be edited (for the guardians, not the children).

Bibliography

[1] 55 inch touchscreen. http://cs-cust06-int.cs.aau.dk/projects/requirements/wiki. [Online; accessed 2014 May].

[2] Doxygen. Doxygen: Main page. http://www.stack.nl/~dimitri/doxygen/index.html. [Online; accessed 2014 March].

[3] Microsoft. Msdn - stopwatch class. http://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch(v=vs.110).aspx, 2014. [Online; accessed 2014 March].

[4] Google. Android. http://www.android.com/, . [Online; accessed 2014 March].

[5] Google. Android 4.0.3 apis. http://developer.android.com/about/versions/android-4.0.3.html#Honeycomb, . [Online; accessed 2014 March].

[6] Summary of first project meeting. http://cs-cust06-int.cs.aau.dk/projects/sw6f14/wiki/Meeting-2014-02-12. [Online; accessed 2014 April].

[7] Katherine Chou Xavier Ducrohet, Tor Norbye. Android studio: An ide built for android. http://android-developers.blogspot.in/2013/05/android-studio-ide-built-for-android.html, 2013. [Online; accessed 2014 March].

[8] Android. Android emulator. http://developer.android.com/tools/help/emulator.html. [Online; accessed 2014 March].

[9] Google. Android | hardware acceleration. http://developer.android.com/guide/topics/graphics/hardware-accel.html, . [Online; accessed 2014 March].

[10] Student-board metting. http://cs-cust06-int.cs.aau.dk/projects/sw6f14/wiki/Meeting-2014-02-25. [Online; accessed 2014 April].

[11] SW603F14. Issue tracker. `http://cs-cust06-int.cs.aau.dk/projects/gui/issues`. [Online; accessed 2014 April].

[12] Meeting with customers. `http://cs-cust06-int.cs.aau.dk/attachments/download/258/Kundemoede03-04-14.pdf`. [Online; accessed 2014 April].

[13] jQueryUI. jqueryui. `https://jqueryui.com/themeroller`, . [Online; accessed 2014 March].

[14] jQuery. jquery. `https://jquery.com/`. [Online; accessed 2014 March].

[15] jQueryUI. jqueryui. `https://jqueryui.com/`, . [Online; accessed 2014 March].

[16] W3Schools. Html5. `http://www.w3schools.com/html/html5_intro.asp`, . [Online; accessed 2014 March].

[17] W3Schools. Css3. `http://www.w3schools.com/css/css3_intro.asp`, . [Online; accessed 2014 March].

[18] Specification requirements. `http://cs-cust06-int.cs.aau.dk/attachments/download/193/SpecificationRequirement.pdf`, . [Online; accessed 2014 May].

[19] Wordpress. Wordpress. `http://wordpress.com/`. [Online; accessed 2014 April].

[20] Joomla. Joomla. `http://www.joomla.org/`. [Online; accessed 2014 April].

[21] Laravel. Laravel 4. `http://www.laravel.com/`. [Online; accessed 2014 April].

[22] EllisLab. Codeigniter. `http://ellislab.com/codeigniter`. [Online; accessed 2014 April].

[23] Anders Bender. [request] from 615 (parrot). `http://cs-cust06-int.cs.aau.dk/issues/954`, . [Online; accessed 2014 April].

[24] Anders Bender. [request] from 602 (tortoise). `http://cs-cust06-int.cs.aau.dk/issues/992`, . [Online; accessed 2014 April].

[25] [request] from 607 (zebra)]. `http://cs-cust06-int.cs.aau.dk/issues/1362`, . [Online; accessed 2014 May].

Part VIII

Appendix

A | Information Sharing

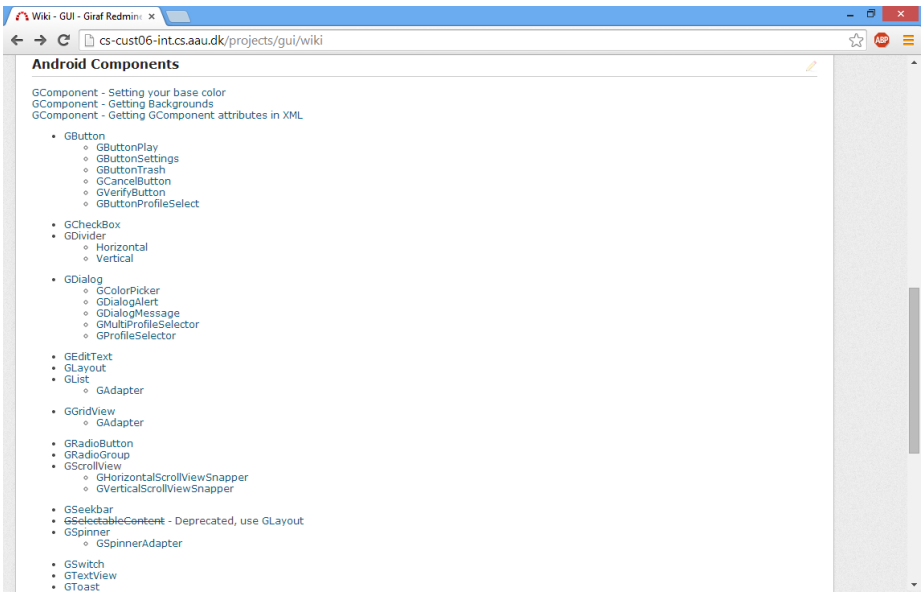

Figure A.1: Screenshot of the list of components on the wiki.

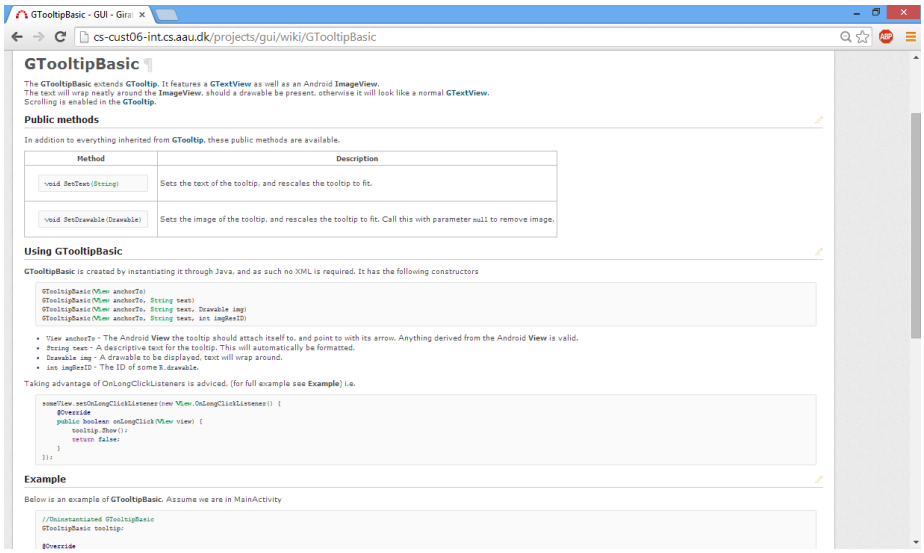

Figure A.2: Screenshot of the page of a component.

B | Research List

Views:

- TextView
- ListView
- View
- Button
- ImageView
- EditText
- GridView
- ImageButton
- ToggleButton
- ViewGroup
- CheckBox
- RadioButton
- AdapterView
- ScrollView
- HorizontalScrollView

Dialog:

- Dialog
- AlertDialog
- ProgressDialog

Widget:

- Toast
- RadioGroup
- TabHost
- ExpandableListAdapter
- ExpandableListView
- Spinner
- ScrollView
- SeekBar

3rd Party:

- kankan.wheel.widget.WheelView

Wombat Specific:

- SubProfileFragment (a saved "configuration" of a timer)
- CustomizeFragment
- SubProfileAdapater (handles list of saved "configurations" of timers)
- WDialog (This class is a custom dialog which is used for all dialogs in WOMBAT)
- WCheckbox (This class is a custom button which is used for the gradient button)

Zebra Specific:

- BounceListView
- SequenceViewGroup
- RoundedImageView

Cars Specific:

- Settings1Fragment extends ListFragment
- MicSetupDialogFragment extends DialogFragment
- CarCrashDialogFragment extends DialogFragment

Cat Specific:

- CreateDialogFragment
- DeleteDialogFragment
- IconDialogFragment
- MessageDialogFragment
- SettingDialogFragment
- TitleDialogFragment

Oasis Specific:

- TabManagerProfile extends Fragment
- TabManagerMedia extends Fragment
- TabManagerChildMedia extends Fragment
- TabManagerChildApp extends Fragment
- TabManagerChild extends Fragment
- TabManagerApp extends Fragment
- TabManagerAllProfiles extends Fragment
- TabManagerAllDepartments extends Fragment
- ProfilesFrag extends ExpandableListFragment
- ChildDepartmentsFrag extends ListFragment
- ChildAppsFrag extends ListFragment
- AppListAdapter extends BaseAdapter
- tmpListAdapter extends ArrayAdapter
- DepartmentListAdapter extends BaseAdapter

Parrot Specific:

- OptionFragment extends Fragment
- AmbilWarnaDialog (from AmbilWarna)
- SpeechBoardFragment extends Fragment
- PARROTCategoryAdapter extends BaseAdapter

PictoCreator Specific:

- DrawView
- PreviewButton
- DecibelMeterView extends View

PictoSearch Specific:

- PictoAdapter extends BaseAdapter
- MessageDialogFragment extends DialogFragment

Train Specifics:

- ChildrenListView extends ListView
- ChildAdapter extends ArrayAdapter
- CustomiseLinearLayout extends LinearLayout
- PictogramButton extends LinearLayout
- Pictogram extends FrameLayout (from pictogram-lib)
- AssociatedPictogramsLayout extends LinearLayout

C | Requirements Excerpt

The following is an excerpt from the Specification Requirements from sprint 2, produced by the groups in charge of communicating with the customers. The full specification can be found here [18].

C.1 Requirement Prioritization

The requirements in this section is split into the different application they belong to. Each individual requirement has a number in end, which shows the priority of this specific requirement.
(1) - Requirements that has the number (1) is a requirements that must be fulfilled.
(2) - Requirements that has the number (2) is a requirements that must be fulfilled, but which are not essential.
(3) - Requirements that has the number (3) is a requirements that must be fulfilled, but only if there is additional time available.

C.1.1 General GUI

- The general color scheme must be black and white (3)
  - It must be possible to choose a color theme for each application. (2) see user story number 3.
  - Each profiles must be able to have their own chosen color theme. (2) see user story number 3.

C.2 User Stories

We have used the user stories from the earlier years as a foundation to create user stories for this semester. The user stories will be used to describe the application

of the system in a common language which the stakeholders can understand and relate to. The stories can then be used as guidelines when developing the specification requirements, which is a more formal way the developers can base their design on. The user stories will also be used to check that all the necessary specification requirements are described to develop the solution the stakeholder seeks.

3. Changing color scheme

    - A new profile for citizen does not have the appropriate color scheme. As a guardian I want to change the color scheme to a color that suits the specific citizen. I select the citizens profile, selects the desired color for scheme of an application and drags it over the application. I expect that the new color scheme for the given application is store for the given citizen profile.

D | GLayout Tally

This tally was performed by asking the groups in each group room of their interest, after a brief description of the issue and possible implementation.

Tally for interest of a RelativeLayout that enables a common selected/deletable/-canceled graphic.

- Room 2.1.01
  - Kategorispil: n
  - Livhistorier/skema: y
- Room 2.1.03
  - GUI: n
  - Timer: n/a
- Room 2.1.05
  - Launcher: y
  - Giraf Admin: n (reason: web)
- Room 2.1.10
  - Sekvens: y
  - Pikto tegner: n
- Room 2.1.12
  - Kategoriværktøjet: y
  - Visual schedular (iOS): n/a
- Room 2.1.46
  - Remote DB: n
  - Picto Search: y

- Room 2.1.48
  - Stemmespillet: n
  - Adminværktøjet: n
- Room 2.1.50
  - Piktooplæser: n/a
  - iOS tal for mig: n/a

As such, the final result is:

- Yes (y): 5
- No (n): 7
- No answer/not present (n/a): 4
- Total: 16

E | Feedback from First Prototype

Place: Special Kindergarten - Birken
Time of Day: 15/5-2014 10:00
Visit Time: 2 hours

Missing Features:

- Multiple pictograms in one slot - Sometimes activities require more pictograms, such as if a child has to play, but with another child. Then there will be a pictogram of "Play" and a picture of the other child.
- Cancellation - One should be able to drag a cross onto a pictogram to cancel the activity
  - After cancellation: The ability to select a new pictogram to be inserted between the canceled item and the next item
- Ability to redo a choice
- "Text only"-pictograms
- Different buttons to select different views representing the schedule (Daily view, weekly view, 2 pictograms etc.)
- Coloring of individual day headers following the international color convention
- Draggable profile picture which can be dragged to each day (See Appendix F user story 1)
- Figure which can show how far the child has progressed in the current day (See Appendix F user story 2)

Other notes:

- Touch screen is very sensitive, both regarding selecting text and pictures
- Touch screen is very sensitive to hand gestures such as zooming or panning
- The costumer learned to control the touch screen in the visiting time
- The costumer does not desire any special way to create the schedule as long as it works (creating on Android and showing on web is fine)
- Pictograms are VERY big on the huge screen

Low priority requests:

- Make parents able to work with the week scheduler
- Add sounds for when a task is done, canceled etc

F | Use stories from Birken

Use story:

1. Child 1 arrives in the morning at Birken. Child 1 goes to the week scheduler and moves his/her picture to the current day. This helps the child get into a ”ready”-state to face the activities of the day.

2. Child 2 has completed a task. Child 2 now takes his/her ”Lightning McQueen” car and moves it to the next task.

G | Group Requests

G.1 Sprint 2 - Group Requests

Group SW615F14: [23] This request includes a large list of components which will cover all the needs of the Parrot makeover. The list contains:

- ListView with items which contain a picture with a text underneath, and highlighting when selected
- GridView with items which contain a picture with a text underneath
- Three Buttons (Play, Delete and Trash-can)
- Single-Lined GridView with static number of fields
- Radio Buttons
- CheckBox
- Spinner (pop-down list with multiple choices (Combo-box in C#))
- TextView
- Region Divider
- Switch

Group SW602F14: [24] This request includes a single component, a *toggle button*.

G.2 Sprint 3 - Group Requests

Group SW607F14 (Zebra)[25]
The request contains two components:

- A horizontal `ScrollView` which automatically "snaps" to the nearest `View`↩ as illustrated in Figure G.1

- A vertical `ScrollView` which automatically ”snaps” to the nearest `View`.

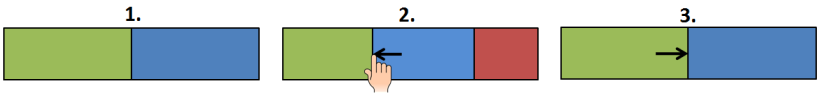

Figure G.1: Horizontal `ScrollView` snapping functionallity