Bachelor Project:

GIRAF MultiProject

#Requirements

#Timer (Wombat)

Group: **SW604F13**

Date: **May 28. 2014**

Project: **Bachelor Project**

AALBORG UNIVERSITET
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:

#Requirment
#Timer (Wombat)

Theme: Multiproject in developing Android Application

Project period:
6th Semester 2014

Project group:
SW604F04

Participants:
Mads Holm
Ali Mansour Nazim
Mike Nellemann Gregersen
Stefan Holst Christiansen

Supervisor:
Scott Xike XIE (PhD, HKU)

Page count: 58

Appendix count: 30,

Finished on May 28th 2014

Synopsis:

This project is a multiproject made in collaboration with several groups of students at Aalborg University, and describes the development and task for the Timer application and the requirement analysis. The purpose of the requirement analysis is to give an overview of the product owners requirements, and to make it easier for other groups to understand the purpose of the project. The requirement analysis is written in collaboration with group SW601F14.
The Timer application is able to help the guardians limiting the amount of time a citizen has in an applications. Therefore Timer creates a timer that exists as an overlay over other applications, such that citizens can always see how much time they have left in the applications. Another function of Timer, is to help the citizen understand time, therefore it is important that the Timers correctly show the passage of time.
The application is updated and a lot of functions and interfaces changes have been made, therefore it almost fulfills the requirement analysis from the project owners.

Signatures:

Mads Holm

Ali Mansour Nazim

Mike Nellemann Gregersen

Stefan Holst Christiansen

Preface

This report is written by four 6th Semester Software Engineering Students during the spring semester 2014. The target audience of this report is Software Engineering students or similar with basic knowledge of developing android applications.

This report describe how a project requirement analysis is made and how an android tablet application is further developed, and will take the reader through an introduction of this multiproject where the vision of the GIRAF multiproject is described and throughout every sprint which includes topics like Product owner interviews, requirement analysis, Sprint overview, new features, implementation and Sprint review.

This project has run for several years and this year the focus should be to complete the applications, and therefore not add many new features.

We recommend reading the chapters in chronological order to get the best understanding of the project. When we is used in the report, it refers to the group members of this project.

We would like to give a special thanks to our project supervisor Xike Xie for his guidance in this project, and the product owners Mette Als Andreasen, Kristine Niss Henriksen of Birken, and Tove L. Søby from Taleinstituttet for collaboration.

Table of Contents

Part I

Introduction

1 Multiproject - GIRAF

Graphical Interface Resources for Autistic Folk (GIRAF) is a project started in 2011 in Aalborg University as a bachelor multiproject. This project is targeting citizen with autism and their guardians, and how we can help to ease the lives of these citizens and their guardians. In the following sections, we discuss the vision of the GIRAF project. Furthermore we present the GIRAF multiproject and how it is structured.

1.1 Vision for the GIRAF multiproject

The vision for GIRAF is to create an application based on Android that can help citizen with autism to communicate with the environment around them [1]. The application is a multi-purpose application with different programs, tools and games that can be used by the citizen with autism and their guardians. One of the purposes of GIRAF is to replace and digitalize the physical items that are used daily by the citizens and their guardians.

Having a multi-purpose application that can be customized and optimized for each individual citizen, should optimize work procedures on the different institution in such a way, that guardians will save time doing repetitive tasks such as making pictograms. This project have so far three schools and institutions for citizens with autism in Northern Jutland involved in the development, but the vision is to distribute GIRAF across all institutions in Denmark.

Groups of students was working on different parts of the multiproject and this project form have since been passed on for several years. This year the students are working in a multiproject, where every group have their own sub-project, that is a part of the GIRAF.

Problems and challenges may occur when working in a multiproject. The communication between the groups is important to ensure that the project runs smooth, and misunderstanding between groups is minimized.

Every sprint has their amount of time, in the first and second sprint we still have courses but we have more time for the project in the last sprints. A sprint is calculated in half days, where a full day is from 8:00 to 16:00, half day is the half of the full day. Sprint 1 has: 18 half days, Sprint 2 has: 25 half days, Sprint 3 has: 24 half days, and last Sprint 4: has 27 half days.

2 Our Project

This project's initial role in the multiproject, is to define the specification of requirements in collaboration with the Product Owners. This work is done in collaboration with group SW601F14, where each group will work in collaboration with specific Product Owners. Our group will work be working with the kindergarten Birken's spokespersons Mette Als Andreasen and Kristine Niss Henriksen, and also with Taleinstituttet's spokesperson Tove L. Søby.

The final role of the project is to develop the Timer application. Timer is used to let the citizens know when they have to stop doing something, like playing, drawing or anything which they only should do in some amount of time and help them getting a better understanding of time.

2.1 Developing Method

In this project we have chosen to use the Scrum method as a developing method. We do not ourselves decide the length of the sprint, as the project is separated into smaller Scrum teams and a collection of spokesmen from each team build up a new team, called the Scrum of Scrums, which decides the sprint length.

As is usual in the Scrum method, we have a daily stand up meeting, called the Daily Scrum, where we ask:

- What have you done since last time?
- What will you do today?
- Is there anything blocking you?

We are all part of The Development Team, and we have one Scrum Master. The Scrum Master's job is to be a firewall to the rest of The Development Team, he will conduct the Daily Scrum. Our Scrum Master is Mads, and he is also a developer.

We have chosen Scrum, because the requirements can change, and we wanted flexible deadlines and planning. Another reason Scrum is ideal, is because we are working in many different teams, so the Scrum of Scrums method is an ideal solution for this.

We will not have a Product Backlog, because it concerns the entire multiproject, and many of the points will not be relevant for report, instead, we simply have a Sprint Backlog for every sprint where we choose which features and what we are making in the sprint.

2.2 Projects

As mentioned before, the GIRAF Project consist of different application and games. To give you a better understanding of this report, we would like to describe some of those application and tools we are using throughout the different parts of this report.

2.2.1 Launcher

The Launcher application handles the execution of the other GIRAF applications. When the launcher executes an app, it will provide it with profile information. Furthermore the Launcher application provide safety for the user, by ensuring that the user do not get unauthorized access to applications and data.

2.2.2 Oasis

Oasis is the local database, which locally stores data and configuration on the tablet. The stored data on the tablet will later be synchronized with the Remote server. We use the Oasis to store the properties for a given profile.

2.2.3 GIRAF-Components

GIRAF-Components are standard designs for different components such as buttons, dialogboxes, checkboxes etc. in the GIRAF project. These are designed by the GUI group to make sure that they are of the same design throughout the whole application.

2.2.4 PictoSearch

PictoSearch is a tool used in GIRAF to search for pictograms. The Timer application uses pictograms when the time runs out, guardians can use PictoSearch to find a specific pictogram to appear when time runs out.

Part II

Sprint 1

3 Sprint Overview

3.1 Sprint Goal

The goal for this sprint is to define the specification of requirements in collaboration with the Product Owner. We will work in collaboration with group SW601F14, to prepare and agree on some questions for the interview with Product Owners. The two groups will combine questions that we will be asking the Product Owners. We will then interview the Product Owners individually, i.e. we interview Birken and Taleinstituttet and the other group interviews the other Product Owners, these interviews will form a better understanding of the Product Owners requirements. After we have interviewed the Product Owners we will write a combined requirement analysis.

3.2 Sprint Backlog

This section has the Sprint Backlog for this sprint, as can be seen in table 3.1. This table has a link to where we have each requirement from, each task to be done for that requirement to be completed and how many hours that were spent in each week of the sprint.

Requirement	Tasks	Week 1	Week 2	Week 3	Week 4	
Internal	Organize collaboration with SW601F14	4	4	4	4	✓
Internal	Schedule meetings with Product Owners	4	4	4	0	✓
Internal	Meet with Product Owners	0	12	12	16	✓
Internal	Evaluate meeting with Product Owners	0	20	40	12	✓
Internal	Meet with software 8	0	0	8	0	✓
Internal	Rework requirements for final draft	0	0	0	20	✓

Table 3.1: The Sprint Backlog

4 Product Owner Interviews

This chapter takes you through the individual interview with each of the Product Owners. These interviews are based on questions made in collaboration with group SW601F14. These questions can be seen in appendix A.

4.1 Birken

We met with Birken Monday 24/2/2014, where we conducted a semi-structured interview. We made a semi-structured interview because it makes it easy to understand their troubles, it is also because it should be a conversation so they could tell their opinion. From the interview we learned:

- The only tool from the multiproject they use is the Timer, called Wombat.
- They prefer the colors of the different applications should be black and white, with the possibility of a color palette so they can change the colors themselves.
- The applications do not need to have a standard look.
- The applications need to be as simple as possible.
- The only information about the children needed, is a picture and a name of each child.
- They prefer that we finish the application that have already been in development, instead of new applications or functionality.
- The exception to the above, is a weekly and daily schedule for their new 55' screen.
- They prioritize the applications like so:
    1. Pictures for all apps (Croc).
    2. Weekly and daily schedule for 55" screen.
    3. Sequencer (Zebra).
    4. Timer (Wombat).

5. The games Trains and Cars (with the possible inclusion of a new game, Barrels).

- The names of the applications should be revised so that they do not use unintuitive animal names.

4.2 Taleinstituttet

We met with Tove Søby from Taleinstituttet Thursday 13/3/2014, where we conducted a semi-structured interview. From that interview we learned:

- The only tool from the multiproject she uses is Train, but she have tried Cars but it does not function correctly.
- She is happy with Train, although it does not function as originally intended.
- She would like Cars to function, if the originally intended control scheme of voice frequency is too hard, then by a voice volume control scheme instead.
- She prefers that we finish the applications that have already been in development, instead of new applications or functionality.
- She has a lot of new ideas for Entertaining Learning Tools that could be made, if we have extra time:
    - Barrels, a game where a barrel is shown, where possibly a sound comes from it, and the child will have to guess what is in the barrel.
    - PuzzleStory, a game where a collection of specific pictograms are shown, and the child then has to organize these into a coherent story.
    - RepeatStory, a story where repetition is important. A good example of this is the song '12 Days of Christmas', where each new day repeats the previous days.
    - She suggests that we can be creative and mix these ideas together with existing games. Such as when you have completed a game of Train, a barrel will appear as a fun surprise for the child.
- The applications need to be as simple as possible.
- She would like a place where she can view all pictograms.

4.3 Requirements Prioritization

We met with Birken again on Monday 17/2/2014, where we presented the requirements as they were, and got them prioritized. We also asked a few follow up questions, which were about requirements that were not specific to any application. These questions were based on a different set of questions, which were made in collaboration with all groups in the multiproject and can be seen in appendix B.

We went though each requirements with them, and they prioritized the requirements in ranks from 1 to 3, the meaning of each rank can be seen in appendix C.

What we learned was:

- They were satisfied with the requirements.
- They want to start all applications through the Launcher.
- They want the system to work with both Wi-Fi and mobile data connection.
- They want to be able to log in both in the Launcher and in each individual application.
- When the timer ends, it should have a sound, and there should a choice between sounds to be chosen.
- Additionally, they do not want the timer to go to the next activity, but they want it to end the current activity.
- They have no direct wishes to when the local database should synchronize with global database.
- They do not mind that we use Google Analytics.
- They only want to have one Guardian profile, instead of one for each individual guardian.
- It should not be possible for the children to use pictocreator.
- As standard, a citizen profile should not have access to any applications.

5 Requirements Analysis

5.1 Introduction

In this chapter we analyse the information gathered through the interviews with the Product Owners. We in group SW601F14 and group SW604F14 have in collaboration written this chapter together and therefore appears in both the reports. This chapter is based on [2].

5.1.1 Purpose

The purpose of this analysis is to give an overview of the requirements, and to make it easier for other groups to understand the purpose of the project. This section will also explain specific requirements, i.e. user interface- and functional requirements, and we also explain the constraints for the project. This document will be presented to the customers in order to get their approval.

5.1.2 Scope

Graphical Interface Resources for Autistic Folks (GIRAF) is an application for Android devices, designed to ease the life of citizens with autism and their guardians. The exact purpose of this application is to provide a digital solution for the physical tools used by citizens with autism to communicate with their guardians and vice versa. It also teaches the citizens to use their voice through various voice-controlled games. GIRAF is an Android overlay that contains other applications, which are the digital solutions for the physical tools used by the guardians. GIRAF will need to have a local database that is accessible by all its applications so they can store pictograms. The local database then needs to be able to synchronize with a larger remote database to enable sharing across multiple devices.

5.1.3 General Description

This section will give an overview over the overall solution and its functions and the constraints we and the users have that will affect the solution.

Product Functions

There are two types of pictures in the product, pictograms and photos. Pictograms are simple sketches, which are used to communicate with citizens, whereas photos are pictures taken with a camera used for similar purposes, for citizens who need something more realistic for communication. In the product, pictograms and photos are handled the same way, and therefore we instead use the word "pictogram" to refer to both.

The product should be a closed environment with several smaller programs that should have a specific function, also most of these smaller programs should be able to work together. Besides working together these smaller programs should handle the following:

- All communications with the central database.
- All communications with the local database.
- A timer, which should be able to visualize that time passes.
- The creation of pictograms.
- Drawing sequences of pictograms.
- Categorizing of pictograms.
- Helping the citizens with autism explaining their life stories.
- A communication tool, which can play sound files attached to pictograms.
- A planning tool, which can help planning the daily and/or weekly activity of the citizens with autism.
- A new launcher to the Android tablets, that collect all the applications in one place and locks out citizens with autism from generic Android functions.
- Furthermore, the product should also include some helpful learning games.

User Constraints

The users are citizens with autism and their guardians. This means the applications should be as simple as possible, so that the citizens can use them.

Some citizens cannot manage certain colors, so we will have to be mindful of what colors we utilize, and in addition, red is seen as a thing that is not allowed or an event that have been canceled, so we should avoid that color in functions the citizens could have access to.

The citizens are all at a different development level, so this adds to the need of simplicity. This also means that there is a need for different accessibility for the different citizens, so that those with low development level, do not have access to certain things while those with a higher level of development do have access to certain functions.

Constraints

Our constraints in this project are mainly time. However, because there are more groups working on this project than there were in the previous semesters, it is possible for some to work on new functionality while others continue the previous work.

5.1.4 Specific Requirements

This section will describe the entire requirement given to us by the customers. This section has been separated into the different applications as well as a general section, general GUI section and a profile section. The general section contains the overall requirements that should hold for every application. The general GUI section contains the user interface, e.g. colors of the applications and other requirements to the design of the application.

Requirement Example

The following section is an excerpt of our full specification requirement, the full version can be seen in appendix C. The user stories that are used in this example can all be found in appendix D. Each individual requirement denoted with a number, which indicates the priority of the requirement.
(1) - Requirements that must be fulfilled.
(2) - Requirements that must be fulfilled, but which are not essential.
(3) - Requirements that must be fulfilled, but only if there is additional time available.

General

- No applications must close, stop or crash unexpectedly. (1)
- All applications must synchronize with the global database, once a day, if there is access to the Internet. (1)
  - It must be possible to synchronize manually with the global database. (2)
- All applications must be able to save settings for each individual profile. (1)
  - It must be possible to copy settings from one profile to another. (2)
- All applications must run in landscape mode (horizontal). (1)
  - Sequences and Week Schedule must be able to use portrait mode. (1)
- All applications, except the Launcher, must have a 'close' button. (2)
  - When the button is pressed, it must close the application, if the user has permission to close applications. (1)
  - If the button is pressed and the user does not have access to close the application, the application must be able to close with the help of a guardian, e.g. by scanning QR code. (1)

- When and only when there is a permitted input, there must be some kind of response, which makes sense in context, but otherwise this is up to the individual developer. (2) see user story number 4.
  - Applications with functions targeted towards citizens must only use singleclick, drag and drop, and swipe input. (2)
  - If there is a function, which should only be usable by a guardian, then all Android gestures are permitted. (2)
  - All applications must be named with meaningful Danish names. (1)
- If a pictogram is not yet saved to the global database, then all applications must show a 'pending' icon, so the guardian can see that it is not yet saved to the database. (2) see user story number 8.
  - Pictograms must be able to be marked as private, so they are not uploaded to the global database. (2)
- All applications must be able to play a pictogram's associated sound. (2)

User Stories

This section has example user stories, which are either used to elaborate specific hard to explain in a single line requirements, but also just as example use of the system. The full list of user stories can be seen in appendix D.

1. Application responding to the user
   - A citizen is playing one of the learning games in the system. As the citizen I expect the application to respond to the input I give the system so I see that something is happening. This should be general for the whole system.
2. Editing an existing pictogram
   - A pictograms caption does not fit and it needs to be changed. The pictogram also have sound attached. As a guardian I select the specific pictrogram and open it in pictogram creator. I now replaces the caption with a new, record a suitable sound and stores the edited pictogram. I expect that the changes to the pictogram now are store and is changed for all application using this specific pictogram. I also expects an indication of whether a pictograms changes have been uploaded to the database or if it still to be done.
3. Different views of sequences
   - A child cannot always understand if there are to many sequences in a week schedule, and a child sometimes pick another sequences instead of the one that was meant to be. Therefore the child want to have as few as possible sequences, and it is possible as a citizen to create sequences for month, week, day or simply a half day schedule.

6 Sprint Review

We have been working in collaboration with group SW601F14 in making the requirements specification. We had an informal meeting with group SW601F14 where we discussed how the interview with the Product Owners should be, we agreed on making a semi-structured interview, we then discussed what questions we should ask the Product Owners. After the meeting with the Product Owners we presented the answers to group SW601F14 and we started working on the requirements. After having a first draft of the requirements we sent it out to the other groups of the multiproject, in order to get feedback on the requirements, this feedback lead to an almost final draft of the requirements that we took out to the Product Owners to get prioritized. This helped us making a final requirements specification.

Overall the collaboration with group SW601F14 was good and efficient and we had no internal issues.

6.1 Collaboration Issues

One of the biggest issues in this sprint was different expectations. We did not understand the other groups of the multiprojects expectations, so after we presented the requirements the expectations from all the other groups was very different than what we had understood. The other groups expected a more in-depth requirements specifications that would include all requirements from the previous years. Instead we had expected to create a requirements specification that was only focused on the new requirements that we had received from the Product Owners. This forced us to rework the requirements specification in order to satisfy the other groups, however we did not include all the previous requirements because of time limitations.

6.2 Future Work

In a next sprint we want to make sure that our expectations and the other groups expectations are the same. We want to try to have another role in the multiproject, but we still want to finish our work with the requirements.

Part III

Sprint 2

7 Sprint Overview

In this sprint we want to try another task than the requirement task, we had in the first sprint. But due to lack of time in the first sprint, we spend extra time to complete the first sprint. Therefore we spent the first week of the second sprint to discuss the report structure and to complete the first sprint.

During the first sprint we were working in collaboration with another group, and the last part of the assignment required extra time to complete which took some time away from this sprint.

Because we wanted to try a new task we have chosen to work on the timer application called Wombat.

7.1 Wombat (Timer)

The GIRAF project is a collection of applications designed to be used by either the Guardians or citizens. As mentioned before we will focus on Wombat during this sprint.

The purpose of the Wombat application is to help the guardians limiting the amount of time a citizen has in an applications. Therefore Wombat needs to create a timer that exists as an overlay over other applications, such that citizens can always see how much time they have left in the applications. Another function of Wombat, is to help the citizen understand time, therefore it is important that the timers correctly show the passage of time, however, because this application has been in development for several years, and the Product Owners have tried it, and they are satisfied with it, we assume that no further testing is needed for this functionality.

Because Wombat is an existing application, the second week of this sprint will be dedicated to reading and understanding the code. In addition, because the first sprint was used to analyse requirements, and not coding, this week will also be used to install the necessary tools: Android Studio and setting up Git repositories.

The product owners also wanted to have more meaningful names to the different applications, therefore Wombat will henceforth be called *Timer*.

7.2 Sprint Goal

The goal of this sprint is to continue work on the *Timer*. We will be focusing on fulfilling the requirements and also add new functionality in an overlay of the entire software solution, an overlay that will show the remaining time of the timer. The standard GComponents will also be implemented to give a standard for all the applications.

7.3 Sprint Backlog

This section has the Sprint Backlog for this sprint, as can be seen in table 7.1. The table shows all the task that should be developed, who is responsible for each task, how many hours that were spent in each week of the sprint and also the priority of the requirement from the product owners. See appendix C.

Requirement	Tasks	Priority	Week 1	Week 2	Week 3	Week 4	
C.4.2	Use GComponents in project	2	-	0	0	10	
C.13.2	Timer overlay	2	-	25	25	10	✓
C.13.2	Decrease timer size	2	-	15	15	0	✓
Internal	Add Google Analytics	3	-	0	0	0	
C.3.3a	Copy saved timers to other profiles	1	-	0	0	0	
C.5.3c	Update timerLib to the new Oasis	1	-	0	0	41	✓
C.13.5b	Add a 'stop' button	1	-	0	0	2	✓
C.13.1	Make the 'start' button a 'restart' button once the timer has started	1	-	0	0	15	✓
C.13.5a	Go to next activity when the timer has ended	2	-	0	0	0	
C.13.5a	Go back to Launcher when the timer has ended	3	-	0	0	0	
C.13.5d	Add PictoSearch	1	-	0	0	0	
C.13.5c	Add a sound when the timer ends	1	-	0	0	0	
C.13.2	Make all timers transparent	2	-	0	0	0	
Internal	Attached pictograms should be visible in the overlay	3	-	0	0	0	

Table 7.1: The Sprint Backlog. The gray column is the week that we spent on sprint 1

8 New Features

8.1 Timer Overlay

The purpose of the overlay was to have the timer to always be visible. This gives you the possibility to see the timer no matter which application you used. In Android applications this feature is called a *system overlay* [3]. This was a key feature of the timer application, because there were a limited use of the timer if it had to be open. Furthermore the system overlay should appear in the top-right corner of the screen.

8.2 GIRAF-Components

The GIRAF project uses some standard design components called GIRAF- Components (called GComponents from now on), that should make all applications look consistent across the GIRAF project. These standard design components should also be implemented in the Timer application.

One of the groups in the semester had made these components, and one of our task was to implement these components. The other group redesigned the buttons, checkboxes, and listViews. The new design standard, is a standard for colors and frames which must be used by all groups of the semester, so that all applications have a similar look and feel.

8.3 Update Local Database

Timer, like all other applications of GIRAF, used a shared local database (called Oasis from this point forward via a library called TimerLib. Oasis was maintained by another group, so we were only able to update how TimerLib uses Oasis. Because of frequent updates to Oasis, we initially decided to wait with updating Oasis, till it was less frequently updated. However, because GComponents used an updated Oasis, we needed to update Oasis for the entire project. These updates caused problems, because functions and parameter were changed, which meant using extra time to understand the changes.

8.4 New Buttons

Since we have implemented an overlay in the Timer, we need to change the functionality of Start button, and also implement a Stop button. In the following subsection we will explain why we need to implement a new button and change the functionality of the start button.

8.4.1 Stop Button

Since the timer now is an overlay, we needed a button that could stop the timer as the name suggest. If the the timer is not running the stop button will be gray, and the stop button is not click-able otherwise the button is gray and not click-able, which means that the timer is running. We explain how this is done in section 9.4.1.

8.4.2 Start Button

Since the user now is able to see the timer application while the timer is running, we need to change the functionality of the start button, this button should change name and design as soon as the timer is running, and it should do as the name suggest, reset the timer. We explain how this is done in section 9.4.2.

9 Implementation

9.1 Timer Overlay

In order to make an overlay for the timer we created a class that would contain all the code needed for the overlay. This class we called *Overlay*, and since this overlay had to be visible when running other applications it needed to be a service [4]. The code for this can be seen in listing 9.1.

```java
public class Overlay extends Service {
```

Listing 9.1: Creating the overlay service

When creating the overlay we needed to set the layout parameters for the window. This is done through the *WindowManager*, where we set all the layout parameters that would make the window act like an overlay. The three layout paramteres that would ensure this is called *TYPE_SYSTEM_OVERLAY* (this parameter makes the windows sit on top of everything else), *FLAG_NOT_FOCUSABLE* (this parameter ensures that the user cannot click on the window) and *FLAG_NOT_TOUCH_MODAL* (this parameter is a ensures that all clicks on the window is sent to the application underneath). The code for setting up the layout parameters can be seen in listing 9.2.

Furthermore we needed to make sure the overlay would sit in the top right corner of the screen, this was also done with a layout parameter called *gravity*, the code for this can be seen in the bottom of listing 9.2.

```java
WindowManager.LayoutParams params = new WindowManager.↩
    LayoutParams(
     DrawLibActivity.frameWidth,
     DrawLibActivity.frameHeight,
     WindowManager.LayoutParams.TYPE_SYSTEM_OVERLAY
     WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
             |WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL↩
    ,
     PixelFormat.TRANSLUCENT);

```

```
params.gravity = Gravity.RIGHT | Gravity.TOP;
params.setTitle("Load Average");
```

Listing 9.2: Setting up the service's layout

9.2 GIRAF-Components

In this sprint we wanted to implemented the GComponents. Due to some troubles updating the Oasis, we could not implement the GComponents because they were not compatible with the version of the Oasis that we were using, which blocked the development process. After we had updated Oasis, we finally got the buttons implemented. We used the wiki site [5], which had guides for implementing the components.

In the next sprint the group working on Timer should implement the last components which are the checkboxes, listViews, buttons and dialogboxes.

Here are an example of the code for a button:

```
1   <dk.aau.cs.giraf.gui.GButton
2                   android:id="@+id/customize_donescreen"
3                   android:layout_width="110dp"
4                   android:layout_height="125dp"
5                   android:layout_gravity="center"
6                   android:background="@drawable/btn"
7                   android:drawableTop="@drawable/↩
                        thumbnail_attachment"
8                   android:paddingTop="20dp"
9                   android:text="@string/↩
                        donescreen_button_description"
10                  android:textSize="14dp" />
```

Listing 9.3: Codelisting for GButtons

The most important changes is at line one in listing 9.3, where we reference the GComponents button. Due to the time limitation we have not implemented all GComponents, which should be done in the next sprint.

9.3 Updating Oasis

There were some problems with Oasis updating, with small updates. The reason these updates, caused us problems, was because some class' names were changed. For example, the class 'App' was changed to 'Application'. Furthermore, some functions were removed from these classes. Other functions had changed parameters, such that the old function call no longer worked.

All these changes, were not big in themselves, however, put together and in addition there being no change logs, meant that we had to figure out ourselves how to change TimerLib to work with the new Oasis. The function 'AddApplicationToProfile' was removed, and with no change logs to tell us what to use instead, we instead went to the group that works with Oasis, they did not know what to do instead. This meant that we used unnecessary time trying to figure out what to do to achieve the old functionality.

Here are some examples of what we would do instead, if we had to update Oasis.

9.3.1 Changing Class Name

Changing a class' name should be avoided at all cost. However, if the class has a name that makes no sense, you can change it to something a little more sensible. In listing 9.4 is an example of a class that changed name, but already had a name that made perfect sense.

```
//Old Class
import dk.aau.cs.giraf.oasis.lib.models.App;

//New Class
import dk.aau.cs.giraf.oasis.lib.models.Application;
```

Listing 9.4: Change of class name

9.3.2 Changing Parameters

Sometimes, changes in the code, means that new parameters are needed, or old parameters are now redundant. Listing 9.5 shows an example where they removed a parameter. Instead of removing a parameter they should either keep the parameter and just not use it, or even better, they should create a new function that has the same name but different parameters and keeping the old function, because you don't have to convert to the new function immediately.

```
//Old Parameters
public Profile(String name, long phone, Bitmap picture, ↩
    String email, Roles role, String address, Setting<String↩
    , String, String> settings, int userId, int departmentId↩
    , int author)

//New Parameters
public Profile(String name, long phone, Bitmap picture, ↩
    String email, Roles role, String address, Setting<String↩
    , String, String> settings, int departmentId, int author↩
    )
```

Listing 9.5: New function parameters

9.3.3 Removing Functions

Like changing a class' name, removing a function should be avoided, however, sometimes you want to avoid users to use this function and want them to use another function instead. But you should always create a changelog that describes how to handle a change like that. An example of this is shown in 9.6.

```
//Old method
oHelp.appsHelper.attachAppToProfile(m_app, m_oGuard);

//New method
ProfileApplication profApp = new ProfileApplication();
profApp.setProfileId(m_oGuard.getId());
profApp.setApplicationId(m_app.getId());
oHelp.profileApplicationHelper.insertProfileApplication(↩
    profApp);
```

Listing 9.6: New method to attach application to a profile

9.3.4 Changelog

One thing that you should always have when other project depend on your own project, is to create a detailed changelog, describing exactly what was changed and what the users should do instead.

9.4 New Buttons

As mentioned before in section 8.4 we implement a new Stop button and we change the functionality of Start button. Furthermore the start button is able to start the overlay which shows the timer.

9.4.1 Stop Button

The purpose of the Stop Button was to stop the timer overlay. This is done by stopping the overlay service. First of all we check if the overlay service is running or not, if it does we make the stop button click-able, this is done in line 1 to line 3 in listing 9.7.
If the overlay service is running, and we click the stop button, then the service should stop. This can be seen in line 8 and 10 in listing 9.7.

```
1   if (MainActivity.svc != null) {
2               stopButton.setText(R.string.stop_button);
3               stopButton.↩
                    setCompoundDrawablesWithIntrinsicBounds(null↩
                    , getResources().getDrawable(R.drawable.↩
                    thumbnail_stop), null, null);
4
5               stopButton.setOnClickListener(new ↩
                    OnClickListener() {
6
7                   public void onClick(View v){
8                       if (MainActivity.svc != null){
9                           getActivity().stopService(↩
                                MainActivity.svc);
10                          MainActivity.svc = null;
11                          Log.d("Overlay", "Stopped");
12                          initBottomMenu();
13                      }
14                  }
15              });}
```

Listing 9.7: Show how we stop the MainActivity service

9.4.2 Start Button

The Start button switches to a restart button as soon the timer begin. Before we restart the timer, we check if the overlay service is running or not. If it does we stop the service and start a new overlay service again at once. This is done in line 11-12 in 9.8.

```
1  startButton.setOnClickListener(new OnClickListener() {
2
3          public void onClick(View v){
4                  currSubP.addLastUsed(preSubP);
5                  guard.saveGuardian(currSubP);
6                  currSubP.select();
7
8                  if (DrawLibActivity.scale == 1){
9
10                     if (MainActivity.svc != null){
11                         getActivity().stopService(↩
                               MainActivity.svc);
12                         MainActivity.svc = new Intent(↩
                               getActivity().↩
                               getApplicationContext(), ↩
                               DrawLibActivity.class);
13                         startActivity(MainActivity.svc)↩
                               ;
14                         Log.d("Overlay", "Restarted"); ↩
                               }
```

Listing 9.8: Code for how restart the MainActivity service

10 Sprint Review

At the end of this sprint we can conclude that we have completed the sprint goals we had this sprint. We finished the timer overlay which was one of the main features we wanted to develop. During this sprint end, clients tested the Timer application, this resulted to a new requirement from the clients. The clients required that the timer should be able to fill the entire screen instead of only appear on the right corner of the screen. This requirement was very important to the clients because they could now use the Timer application also to keep track of the time. Besides the full screen function, we need to update to the new Oasis. We also implemented new buttons including a restart and stop button, and the GComponents, but there are still some components that needs to be changed to the new standard.

10.1 Experience

In this sprint we had some issues in our process because we were dependent on another group who updated their project and made a lot of changes. These changes made it difficult for us to continue working because our project couldn't compile after we updated. We needed to comment out and change a lot of code because of some changes in names and variables. The result of this was a bad mood, double work and other important features was not implemented due to the time limitations. We introduced another way to do all these updates, instead of that method the group used which caused problems.

10.2 Future work

In the next sprint this application must be further developed. There are more fixes and bugs after the updates. Because Oasis is getting updated almost every week, we need to keep tracking it, and make sure that our project follows.

Part IV

Sprint 3

11 Sprint Overview

11.1 Sprint Goal

In this sprint we wanted to continue with the tasks from last sprint, and complete the application. This is why we have the same backlog except we removed the tasks that already are fulfilled.

11.2 Sprint Backlog

This section has the Sprint Backlog for this sprint, as can be seen in table 11.1. This table has a link to where we have each requirement from, each task to be done for that requirement to be completed and how many hours that were spent in each week of the sprint.

Requirements	Tasks	Priority	Week 1	Week 2	Week 3	
C.4.2	Use GComponents in project	1	-	30	30	✓
Internal	Add Google Analytics	3	-	1	3	✓
C.3.3a	Copy saved timers to other profiles	1	-	0	0	
Wiki [5]	Add Profile Selector button	1	-	0	0	
Wiki [5]	Add Settings button	-	-	-	-	
C.13.5a	Go to next activity when the timer has ended	2	-	0	0	
C.13.5a	Go back to Launcher when the timer has ended	3	-	2	7	✓
C.13.5d	Add PictoSearch	1	-	0	0	
Internal	Attached pictograms should be visible in the overlay	-	-	-	-	
C.13.5c	Add a sound when the timer ends	1	-	3	7	✓
C.13.2	Make all timers transparent	2	-	10	0	✓
Last Sprint End [5]	The timer should work in fullscreen	2	-	4	0	✓
Internal	Fix bugs	1	-	40	40	✓

Table 11.1: The Sprint Backlog. The gray column is the week that we spent on Easter holiday

11.2.1 Rejected Requirements

Add Settings button

During one of the status meetings we had the semester, we all agreed on a requirement stating that all applications should include a settings button with an associated setting window. But due to the limited selection of setting in our application and after a discussion about in the weekly status meeting, we decided that we would reject this requirement for applications who had a limited selection of settings, as we did.

Attached pictograms should be visible in the overlay

During our last sprint when we implemented the overlay, we did not have time to implement the requirement *Attached pictograms should be visible in the overlay*, but after reevaluating this requirement we found that to make the system overlay fit in the top-right corner of the screen, we also needed to decrease the size of the current timer. The decreased size of the current timer, means that if an image was to be attached to the timer, it would either be too small, or the overlay would have to be increased in size. However, that would cause the overlay to be too large and hide away too much of other applications' functions. Therefore we have decided to not attach images to the overlay, and leave this function to only work when the timer is in fullscreen mode.

12 New Features

12.1 Sound

In this sprint we will add a sound when the timer ends. As other timers, for instance, a kitchen timer, plays a sound when the time has passed. The customers want a sound, and the possibility to choose between some different sounds. We added six different sounds, a standard beep sound, two different horn sounds, a chicken sound, a cow sound and at last a water drop sound.

These sounds are just chosen by ourselves because we want to give examples, the customers wanted a kitchen timer sound. The future group should find the actual wanted sound, because we just focused on the functionality.

An easy way to implement this sound feature is to have a button that shows a dialog box where the different sounds is possible to choose between.

12.2 GIRAF-Components

To have a unified design throughout the whole GIRAF project, the GUI-Group have developed and designed some standards for several components that are used throughout the whole GIRAF application. During this sprint we have updated and changed some of the components to GComponents. The following list show which components have been updated to GComponent.

- **Dialog Boxes:** Have been updated to GDialog, which define the standards for a dialog box, this update resulted in a change in colors of the dialog box, which fits better in the whole GIRAF application.
- **Buttons:** Every single button in the Timer have been updated to G-Button, which define the standards for a button used throughout the whole GIRAF application. Small changes in color is the result of this update.
- **Check Boxes:** Have been updated to GCheckBox, which define the standards for a CheckBox used throughout the whole GIRAF application.
- **Lists:** Have been updated to GList, which define the standards for a List used throughout the whole GIRAF application, this resulted in small change in colors.

- **Colors:** Background colors and different colors from the interface have been changed to some specified color that are used throughout the GIRAF application.

12.3 Google Analytics

In this sprint we implemented Google Analytics. Google Analytics is a service that generates statistics over the usage of the application, i.e. you can see what the users use the most in your application and in case something went wrong you can trace the problem.

We followed the guide that is on our *Redmine page* [6].

One of the requirements of using Google Analytics was that we needed a Google Gmail, but unlike what the guide suggested we created a new Gmail just for our application. All the information for the Gmail can be seen below:

Name: GIRAF Tidstager
Mail: tidstager@gmail.com
Password: AAUGiraf
Birthday: 1 feb 1943
Sex: Other

12.4 Transparent Timer Overlay

When the timer is started in Overlay mode, the background of the timer should be transparent If the background was not transparent, it would take up a big part of the screen and cover whatever the other applications, have in the top right corner. By making the background of the timer transparent, we reduce this inconvenience.

12.5 Bugs

In this sprint we found several bugs

1. When the timer finished, and the user went back into Timer, it would not use new profile information from the Launcher, instead keeping the old profile information.
2. If a timer profile was selected, and then deleted, and the user then tried to save, nothing would happen.
3. The background color of the colorpicker buttons still does not appear in thecorrect color when the application first starts or 'switch layout' is pressed.
4. When the 'gradient' checkbox was checked and the Standard Timer was chosen, the color of the timer would rapidly switch back and forth from the time left colors, instead of slowly fading from one to the other.
5. When the button 'New Timer' was pressed, the application would crash.

6. The application would crash if a user tried to attach a timer, but had no saved timers.
7. When the 'attachment' button was pressed, but no profile was selected, the application would crash.
8. When a timer had an attachment, the 'attachment' button would keep its standard icon, instead of switching to an icon that would indicate what kind of attachment was attached to the timer.
9. Each time a new dialog window was made, the background would get a darker shade, meaning if a new dialog window opened another new dialog window, the background would get darker and darker.
10. The background color of dialog window was white, instead of 'GIRAFOrange'.

13 Implementation

13.1 Sound

For the sound function we implemented it as we wanted from the earlier description. We have this GButton and when this is pushed, we show a GDialogbox. In the dialogbox we show all the sounds we wanted to be able to play. For the code see listing 13.1. We instantiate the mediaplayer first and set our standard beeb *R.raw.song;*.

At last we create the player when we show the donescreen, and we play the standard sound if we have not chosen anyone else, and then we start it.

```
    private MediaPlayer mediaPlayer;
    public static int soundindex = R.raw.song;

    mediaPlayer = MediaPlayer.create(DoneScreenActivity.this↩
        , soundindex);
    mediaPlayer.start();
```

Listing 13.1: Show how we start the sound service

13.2 GIRAF-Components

As mentioned in section 12.2, we have updated and changed the design of some of the components to GComponents. The following listing shows the xml-code of the GComponent we have used. In our application we use the GCheckbox component for changing to fullscreen mode. As we see in listing 13.2 we also call the id FullScreenCheckBox.

```
<dk.aau.cs.giraf.gui.GCheckBox
                                android:id="@+id/↩
                                    FullScreenCheckBox"
                                android:layout_width="40dp"
                                android:layout_height="40dp"
```

```
                                  android:layout_gravity="↩
                                      center_horizontal"/>
```

Listing 13.2: xml code for the GButton

The next GComponent is the WDialog which extends the GDialog component. We used this one to prompt the user with a Dialog window, when the user click for example the sound button. In listing 13.3 we see the options for the Dialog window, and the xml-code.

```
public class WDialog extends GDialog {
```

Listing 13.3: xml code for the GDialog

```
<dk.aau.cs.giraf.gui.GList android:id="@+id/↩
    profile_list_dialog_listview"
         android:layout_width="400dp"
         android:layout_height="300dp"
         android:padding="20dp"/>
```

Listing 13.4: xml code for the GList

We were following the GUI requirement for all groups and we also followed the colors for the application. The main color is 'GIRAFOrange', which we used in most of our background colors. In listing 13.5 we show how we set the color.

```
getResources().getColor(R.color.GIRAFOrange);
```

Listing 13.5: xml code for the GIRAF Colors

The last component we implemented was the GColorPicker. This is a dialog and the main purpose was to make the user able to change color the timers. In our application it was used for changing the color of the timer skin. As we see in listing 13.6 we instantiated a new GColorPicker which get the current context, and when we click the OK button in the dialog window, we change the color of the timer, to the color that the user picked in the GColorPicker dialog.

```
GColorPicker diag = new GColorPicker(v.getContext(), new ↩
    GColorPicker.OnOkListener() {
                        @Override
                        public void OnOkClick(GColorPicker diag,↩
                             int color) {
                             currSubP.timeLeftColor = color;
                             setColor(colorGradientButton1.↩
                                 getBackground(),
                    currSubP.timeLeftColor);
                        }
                    });
                    diag.SetCurrColor(0xFF000000);
                    diag.show();
```

Listing 13.6: xml code for the GColorPIcker

13.3 Google Analytics

When implementing Google Analytics we first have to set some permission for the application to allow this service. The code in listing 13.7.

```
<uses-permission android:name="android.permission.INTERNET" ↩
    />
<uses-permission android:name="android.permission.↩
    ACCESS_NETWORK_STATE" />
```

Listing 13.7: Permissions for Google Analytics

Furthermore we needed an analytics.xml file that is used to setup Google Analytics. The code for the xml file can be seen in listing 13.8.

```
//----analytics.xml----//

<?xml version="1.0" encoding="utf-8" ?>

<resources xmlns:tools="http://schemas.android.com/tools"
    tools:ignore="TypographyDashes">

  <!--Replace placeholder ID with your tracking ID-->
  <string name="ga_trackingId">UA-48608499-7</string>

  <!--Enable automatic activity tracking-->
  <bool name="ga_autoActivityTracking">true</bool>

  <!--Enable automatic exception tracking-->
  <bool name="ga_reportUncaughtExceptions">true</bool>

</resources>
```

Listing 13.8: Show what is in the analytics.xml file

And as a last bit we needed to use the *EasyTracker* to tell Google Analytics to keep an eye on this class. The code can be seen in listing 13.9.

```
@Override
  public void onStart() {
    super.onStart();
    EasyTracker.getInstance(this).activityStart(this);
  }

  @Override
  public void onStop() {
    super.onStop();
    EasyTracker.getInstance(this).activityStop(this);
  }
```

Listing 13.9: Show how we start Google Analytics

13.3.1 Transparent Timer Overlay

As can be seen in listing 13.10 to make the background of the timers transparent, we simply check if scale is not equal to 1, because if it is, that means the timer should be started in fullscreen. If it is, we change the background color's alpha value to 0, which makes the color transparent.

```
if(DrawLibActivity.scale != 1) {
            paint.setColor(background & 0x00);
        }
```

Listing 13.10: Implementation of transparency

13.3.2 Go Back to Launcher When Timer Ends

This feature was primarily made by the Launcher group, and happens after we call their 'AUTHENTICATE' intent. This is done, as can be seen in listing 13.11, by creating the authenticate intent and then starting it. Once it has been started, our application is done, and the guard should be reset, so that when the user next time starts Timer, it will correctly use the profile information from the Launcher. Afterwards the application should finish.

```
frame.setOnClickListener(new OnClickListener() {

      public void onClick(View v) {
        // Start
        Intent i = new Intent("dk.aau.cs.giraf.launcher.↩
            AUTHENTICATE");
                startActivity(i);
                guard.reset();
        finish();
      }
    });
```

Listing 13.11: Implementation of Launcher's Authenticate

13.3.3 Bugs

In this sprint, we fixed some of the bugs that were found, but not all. This section will describe how we fixed those bugs. Each bug will be referenced by a number, that is a reference to the number seen in section 12.5.

1. We fixed bug number 1 by resetting the guard information whenever the timer was done, and when Timer was exited without the overlay started.

2. Bug number 2 was fixed by checking if the selected timer profile was in the list of saved timers, and if it was not, then save a new version of it, instead of just modifying the non existing saved timer.

3. Bug number 3 was fixed, but our testing showed that it only happened when using GButtons and not regular buttons. However, because of the requirement

to use GComponenents, and the fact that the GUI group was very busy, this was not fixed.

4. We found bug number 4 was caused by a division error. It was set to change the colors each 1/1000th second, instead of each second. Correcting the division error, it then functioned correctly.

5. Bug number 5 was because of incorrect use of GComponents, since we have two different layouts, we accidently only switched one layout to use GComponents for this button, and when in the other layout, and clicked the button, there would be a casting error, trying to cast a Button to a GButton. Fixing this, fixed the bug.

6. We fixed bug number 6 by first checking if there was any saved timers for the current profile, and if there wasn't, remove the ability to attach another timer.

7. To fix bug number 7 we simply checked if there was a selected user, and if there was not, removed the ability to attach things, when no profile was selected. This will need a better fix in the future.

8. Bug number 8 was caused by the function 'setCompoundDrawablesWithIntrinsicBounds' of the GButton, does not work with a layer drawable, which we were using. By calling a function that converted the layer drawable to a single drawable, 'setCompoundDrawablesWithIntrinsicBounds' functioned as it should and the background image of the buttons changed correctly again.

9. We fixed bug number 9 and number 10 by asking the GUI group to fix this, and they made a function to disable the function that made the background darker. We used this function if any new dialog windows would open other new dialog windows. They also changed the background color of the dialog windows to 'GIRAFOrange'

14 Sprint Review

In this sprint we have made some new features, and the application really came together.

We have finished the timer overlay, it is transparent as we wanted. It is also possible to run the timer in fullscreen mode, so now we have two possibilities: fullscreen mode, or overlay mode. All the GComponents are fixed and inserted in the design now including dialogboxes, buttons, checkboxes, lists and all colors.

The customers wanted a sound to mark that the timer has stopped so we added a standard sound beep, and the opportunity to choose between different sounds. At last we implemented Google analytics to the timer app, it helps the next group that wants to work on the app, to see what makes the application crash, and see which kinds of features the users uses the most.

14.1 Experience

We also had some annoying experiences with Oasis as we had in sprint 2. Every time the Oasislib was updated, it caused problem to our project, as an example we were confused because the class names were change, and functions were removed. But we also had good experiences in this sprint. Our project team was very good in developing and working together.

14.2 Future work

In the future there are still some features to work on. One thing that needed to be implemented is the PictoSearch, we also needed to implement a way to copy the timer between citizens, make sure the profile selector is working. We also needed to make sure that you cannot change to fullscreen mode when the timer is running.

Part V

Sprint 4

15 Sprint Overview

15.1 Sprint Goal

In this sprint we wanted to continue with the tasks from last sprint, and complete the application. This is why we have the same backlog except we will remove the items that are already fulfilled. We will also make a code review in collaboration with group SW611F14.

15.2 Sprint Backlog

This section has the Sprint Backlog for this sprint, as can be seen in table 15.1. This table has a link to where we have each requirement from, each task to be done for that requirement to be completed and how many hours that were spent in each week of the sprint.

Requirements	Task	Priority	Week 1	Week 2	
C.3.3a	Copy saved timers to other profiles	1	10	0	✓
C.13.5a	Go to next activity when the timer has ended	2	-	-	
C.13.5d	Add PictoSearch	1	5	15	✓
Internal	Fix bugs	1	15	20	(✓)
Wiki [5]	Add Profile Selector button	1	0	20	✓
Internal	Code Review	0	0	5	✓

Table 15.1: The Sprint Backlog. The gray column is the week that we spent on Easter holiday

15.2.1 Rejected Requirements

During the start of the semester we got the requirement that the timer should be able to go to the next activity when the timer has ended. But this was of low priority.

This would require collaboration with the Launcher group as it would be their responsibility to open the next activity, and also collaboration with the group that is creating the Week Planner, because it is them who decide what the next activity will be. This application was started this semester, and thus only a working copy was available at the Sprint End of sprint 4. At this point in time we do not have the time to complete this requirement. Therefore we have chosen not to reject this requirement.

16 New Features

16.1 PictoSearch

In order to have all pictogram incorporated into the application, we implemented PictoSearch, which is an application developed by another group. The PictoSearch application enables the user to search through the complete database and load the pictograms the user wants.

16.2 Checkbox Fullscreen

One of the requirements we got from the customers, was that they wanted the timer to work in both fullscreen and as an overlay. This was because the fullscreen would be used to teach the citizens how time work and if the citizens were going to an activity outside of the tablet, but still needed a timer, such "Read a book for the next 30 minutes", in such a case, a fullscreen timer would be preferable to a small overlay.

16.3 Profile Selector

The profile Selector button will make the users able to choose between different profiles. It could be implemented as a button with a logo of people and could be placed next to the new timer button.

16.4 Copy Saved Timers to Other Profiles

It should be possible to copy a saved timer, to another profile, so that the guardians does not have to make the same timer several times, in case multiple profiles have use of this single timer.

16.5 Bugs

In this sprint we found a couple of bugs.

1. When the user checked the Fullscreen checkbox, while the timer was running, it would change the size of running timer.
2. A bug found in this sprint was: If there existed a citizen profile that did not have access to Timer, the application would crash.

17 Implementation

17.1 PictoSearch

When implementing the PictoSearch application we have to create an intent that will run the PictoSearch application, also we have to send it some parameters that will determine what type of pictograms will be available for the user to see and also to limit the amount of pictograms the user can select. The last thing to do is to start the intent, this is done with the function *startActivityForResult*, this function enables us to get the result from the PictoSearch application, also we include a request code that we use to identify how we called the PictoSearch application. The code for this can be seen in listing 17.1.

```
Intent i = new Intent();

i.setComponent(new ComponentName("dk.aau.cs.giraf.↩
    pictosearch", "dk.aau.cs.giraf.pictosearch.↩
    PictoAdminMain"));
i.putExtra("purpose", "single");
i.putExtra("currentChildID", guard.profileID);
i.putExtra("currentGuardianID", guard.guardianId);

startActivityForResult(i, 1);
```

Listing 17.1: Show the start of PictoSearch

When the user is done using PictoSearch and have selected one or two pictograms we have a function that will load the pictograms into the right place. The code for this can be seen in listing 17.2. This function first checks if everything went as is should and runs the code according to request code. The next thing is to load all the pictogram IDs for all the selected pictograms, and after that we load in the pictograms and uses them according the request code.

```
public void onActivityResult(int requestCode, int resultCode↩
    , Intent data) {
super.onActivityResult(requestCode, resultCode, data);
```

```

if (resultCode == Activity.RESULT_OK && requestCode == 1) {
    int[] checkoutIds = data.getExtras().getIntArray("↩
        checkoutIds");
    if (checkoutIds.length == 0) {
        GToast t = GToast.makeText(MainActivity.context, "↩
            Ingen pictogrammer valgt.", Toast.LENGTH_LONG);
        t.show();
    }
    else {
        PictogramController pichelp = new ↩
            PictogramController(getActivity());
        Pictogram pictogram = pichelp.getPictogramById(↩
            checkoutIds[0]);
        pictogram.getImage();

        Attachment att = new SingleImg(pictogram.getImage())↩
            ;
        setAttachment(att);
    }
}
```

Listing 17.2: Show how we load in the pictograms

17.2 Checkbox Fullscreen

We have inserted a checkbox in order for the user to be able to switch between fullscreen and an overlay. If the checkbox is checked we set the timer scale to 1, this means the timer will fill the screen, and otherwise we set the scale to 8. The code in listing 17.3 shows how the checkbox work.

```
GCheckBox FullScreenCheckBox = (GCheckBox)getActivity().↩
    findViewById(R.id.FullScreenCheckBox);
FullScreenCheckBox.setOnCheckedChangeListener(new ↩
    CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, ↩
        boolean isChecked) {

        TempChange = isChecked == true ? 1 : 8;
    }
});
```

Listing 17.3: Show how we set scale variable

When we start the timer we check if the scale variable, if it is 1 we start the fullscreen timer, and otherwise we start the overlay. The code for this can be seen in listing 17.4

```
if (DrawLibActivity.scale == 1){
    //start Fullscreen
}
```

```
else{
    //Start overlay
}
```

Listing 17.4: Show how we set fullscreen

17.3 Profile Selector

The implementation of the Profile Selector was done by following the instructions on the wiki page [5] on the semesters Redmine page. [7]. Furthermore, as can be seen in listing 17.5, a few more things were added. When the new profile is selected, we change the profileID of the current guardian (the variable 'guard'), which contains the ID of the currently selected citizen. Once this is done, we find the citizen in the Array called 'Children' and change which citizen is selected and afterwards we change the 'Child' variable to be the new citizen. Once this has been done, we change the name that is shown at the top of the application to the name of the newly selected citizen and finally we load in that citizen's saved timers.

```
 private void initProfileButton(){

        profileButton = (GButtonProfileSelect) getActivity()↩
            .findViewById(
                 R.id.customize_profile_button);

        //Call the method setup with a Profile guardian, no ↩
            currentProfile (which means that the guardian is↩
             the current Profile) and the onCloseListener


        profileButton.setup(guard.m_oGuard, null, new ↩
            GButtonProfileSelect.onCloseListener() {
            @Override
            public void onClose(Profile guardianProfile, ↩
                Profile currentProfile) {
                //If the guardian is the selected profile ↩
                    create GToast displaying the name
                if (currentProfile == null) {
               //If another current Profile is the selected ↩
                    profile create GToast displaying the name
                }
                else {

                    if (currentProfile.getRole() == Profile.↩
                        Roles.CHILD) {
                         guard.profileID = currentProfile.↩
                             getId();
                         if (children == null || !children.↩
                             isEmpty()) {
                              for (Child _child : children) {
                                   if (_child.getProfileId() ==↩
                                        guard.profileID) {
```

```
                                    children.get(children.↩
                                        indexOf(_child)).↩
                                        select();
                                    child = _child;
                                    break;
                                }
                            }
                        }

                        GTextView tv = (GTextView) ↩
                            getActivity().findViewById(R.id.↩
                            customizeHeader);
                        CharSequence cs;
                        if (child != null) {
                            cs = child.name;
                        } else {
                            cs = "Ingen Valgt Profil";
                        }
                        tv.setText(cs);
                        SubProfileFragment spf = new ↩
                            SubProfileFragment();
                        spf = (SubProfileFragment) ↩
                            getFragmentManager()
                                .findFragmentById(R.id.↩
                                    subprofileFragment);
                        spf.loadSubProfiles();
                        setDefaultProfile();
                    }
                }


            }
        });
    }
```

Listing 17.5: Implementation of Profile Selector

17.4 Copy Saved Timers to Other Profiles

The implementation of the Copy button was done similarly to Profile Selector, and will be placed next to the delete button on each saved timer. When pressed, it will open the profile selector window, and when a new profile is chosen, will add the timer to be copied to the selected profile. This is done, as can be seen in listing 17.6, by finding the citizen in the Array 'Children', and when found, save as if the save button was pressed while that profile was selected.

```
final GButtonProfileSelect copyButton = (↩
    GButtonProfileSelect) v.findViewById(
                    R.id.customize_copy_button);
            copyButton.setup(guard.m_oGuard, null, new ↩
                GButtonProfileSelect.onCloseListener() {
                @Override
    public void onClose(Profile guardianProfile, Profile currentProfile) {
        //If the guardian is the selected profile create GToast displaying the name
        if(currentProfile == null){
            GToast w = new GToast(MainActivity.context, "Du kan ikke kopiere til en personale profil", 2);
            w.show();
        }
        //If another current Profile is the selected profile create GToast displaying the name
        else{
            ArrayList<Child> children = guard.Children();
            if(children == null || !children.isEmpty()) {
                for (Child _child : children) {
                    if(_child.getProfileId() == currentProfile.getId()) {
                        GToast w = new GToast(MainActivity.context, "Tidstageren " + finalsp.name + " er blevet kopieret til " + currentProfile.getName().toString(), 1);
                        w.show();
                        _child.save(finalsp.copy(),false);
                        break;
                    }
                }
            }

        }
    }
});
```

Listing 17.6: Implementation of Copy button

17.5 Bugs

This section will describe how we fixed each bugs. Each bug will be referenced by a number, that is a reference to the number seen in section 16.5.

1. We fixed bug number 1 by making sure that the scale would only be changed when the 'start' button was pressed, and the checkbox only told the start button what the scale should be.

2. We made a quick-fix to bug number 2 by checking if the application was not attached to the current selected profile. However, this should be checked upon further, by future groups working on Timer, as it is likely to still cause problems, but since this bug was found and fixed on the day of the last Sprint Review, there was not time to make a better fix.

18 Code Review

This chapter is made in collaboration with group SW611F14. In this review we will each review a single class that is chosen by the non reviewing group.

18.1 Their Review

The first section will describe the feedback we got from the other group and which files they reviewed.

18.1.1 The Class

We have chosen our Overlay class because we have build this class from the beginning, and this is the class that sets up the overlay for the timer.

18.1.2 Their Feedback

After the other group have reviewed our code, they gave us a document with all their thoughts and comments. The summery of this was that overall, the code was neat, organized, well-documented and they found it difficult to point to any improvements. The only criticism was that we might have too many comments and that we might not take into account of the system is running a non-existing guardian and that this would hinder maintenance or similar.

The full review call be read in appendix F.

18.1.3 Our Class Reworked

After getting the other groups feedback we removed some of the unnecessary comments, and improved the comments for the Runnable. As for the feedback about the non-existing guardian, we take care of this before we call the overlay class.

The complete code for the overlay can be seen in appendix F.1.

18.2 Our Review

The first section will describe the feedback we made for the other group and which files they wanted us to review.

18.2.1 The Class

The class that they wanted us to review was a search class, that includes five search function. The full code can be seen in appendix F.2.

18.2.2 The Feedback

The code overall is pretty good, great separation. But some of the comments are todo's and some variable names could be described much better. They also have a function DoSearch_Tags, that seemingly does nothing, and inside this function they have a for loop that also does nothing, except making the running time much slower.

Other than that there is not much to say, the code is fine.

19 Sprint Review

In this sprint we had to complete the tasks from last sprint and try to complete the application. From the backlog we needed to add PictoSearch, Checkbox Fullscreen and Profile Selector, Copy saved timer to other profiles and Go to next activity when timer ends.

We rejected the 'Go to next activity when the timer has ended' because it was of low priority, and required collaboration with the Launcher group and Week Planner group. We implemented the PictoSearch application and we had to create an intent that could run the app, and we needed to send some parameters to it, to determine which pictograms the user had selected.

We also implemented the checkbox to give the user the opportunity to change between fullscreen mode and overlay mode. We simply change the scale in the timer from 1 to 8.

At last we implemented the profile selector button and the Copy saved timer to other profiles as we wanted. The profile selector button is placed at the top next to the new timer button, and it was implemented by the instructions on the wiki page under GUIs Redmine page. The Copy saved timer to other profiles feature was implemented similarly to the Profile Selector, and it is placed next to the delete button on each saved timer.

We can therefore conclude that we implemented all the features from our backlog except one function the Go to next activity when the timer ends feature. Everything works as it should and therefore a successful sprint.

19.1 Future Work

In the future the application can be fine-tuned, but as it is now it works as intended. There are still few bugs and other things that should be implemented, such as:

- Citizen profiles that do not have access to Timer can be chosen in Timer's profile selector.

- If a citizen profile that does not have access to Timer is chosen, and tries to save a timer, the application crashes.
- The background color of the colorpicker buttons still does not appear in the correct color when the application first starts or 'switch layout' is pressed.
- A button that closes Timer should be implemented.
- The Timer application should have the ability to play the sounds the Product Owners wished, they were:
  - The sound of a standard Kitchen timer.
  - During the last Sprint End, the Product Owners expressed a wish for the ability to play the sound of the last pictogram's which appear when the time is finished.
- The Timer application should have the ability to copy different timer to several profiles at once. Hint: Use the copy method from the Sequence application.
- The attachment button should not be click-able when the checkbox for fullscreen is not activate, or it should at least in some way indicate that the attachment will not be shown if the timer is not in fullscreen mode.

Part VI

Evaluation

20 New Features Complete

In this chapter we list the new features we have implemented in last three sprints. The result can be seen in figure 20.1 and compared to figure 20.2 there are a lot of changes. Below you can see some of the major changes in the Timer application.

Overlay: We have implemented an overlay, which appears when the timer begin.

Restart Button: We have also changed the functionality of the Start Button, so when you click the Start Button, the button changes to a restart button.

Stop Button: We have also implemented a stop button, this stops the timer, when it is running. This button is not available (gray) when the timer is not running.

GIRAF-Components: We have implemented the interface components to GComponents, which is standard design components for the whole semester.

Stop sound: The Timer application was required to have a stop sound. When the timer runs out of the application plays the chosen sound to indicate that the time is finished.

Google Analytics: We have also implemented Google Analytics to make sure every time the application fails or when error appears, we can get information about the errors from Google.

Updated Oasis: We also updated the Oasis to communicate with the database.

Bug fixes: We started to work with bug fixing at the beginning of this project, also at the end of this project we used many hours to fix bugs.

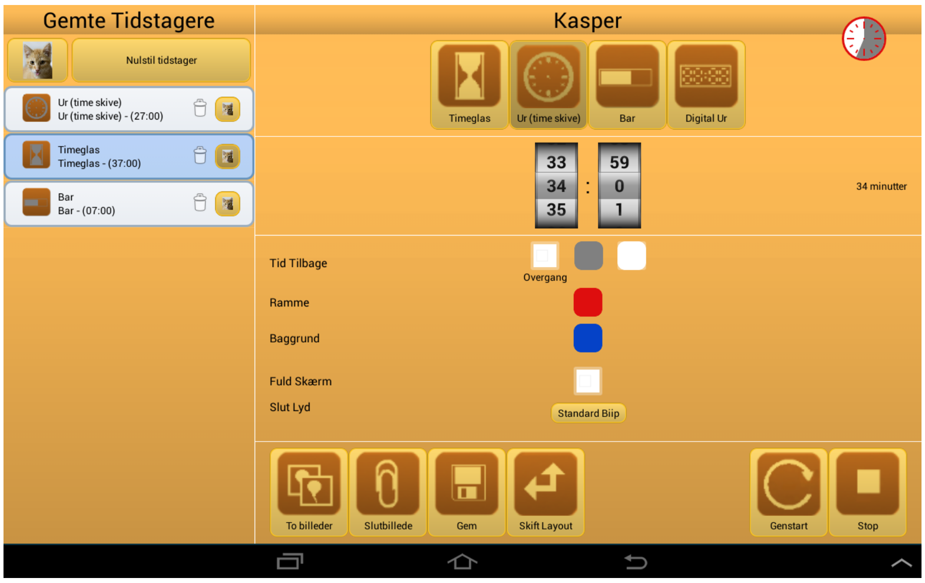

Figure 20.1: Wombat Application

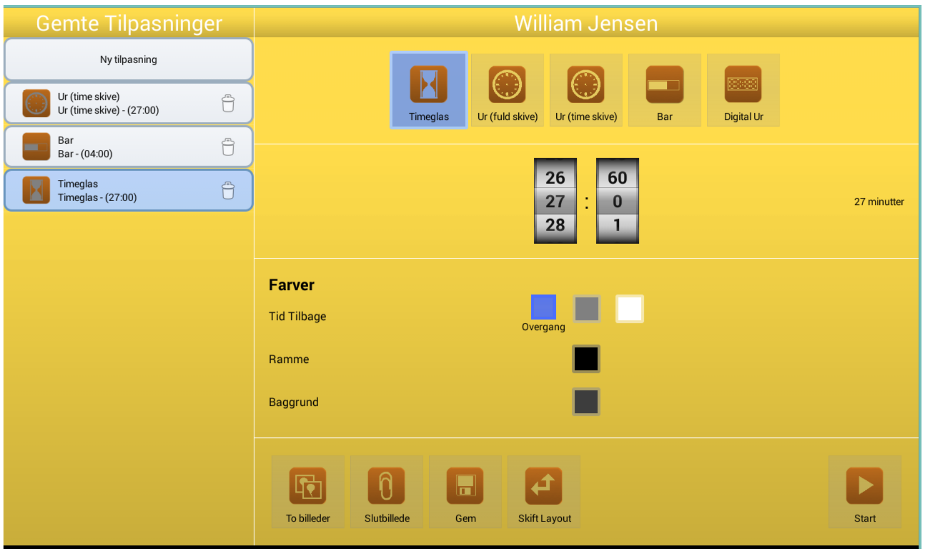

Figure 20.2: Old Wombat Application

21 Evaluation

21.1 Conclusion

During the first part of this project, we have worked with the requirements. We had several meetings with different clients, where we interviewed the clients to collect the most important requirements for the application. We had also a good collaboration with group SW601F14, to help us collecting those requirements. This collaboration resulted in a list with prioritized requirements, which was given to the other GIRAF groups. Some misunderstanding occurred at the end of the sprint, which resulted in a detailed requirement specification.

In the last three sprints we have worked with the Timer application. This application was a tool for helping the guardian to setup time limitation for citizens to use the GIRAF application.

In the second sprint a new overlay was implemented, so every time the timer begins, the application opens a new overlay. Later on at the third sprint we have implemented sounds, which allowed the person to choose a sound when the time was finished. We have also updated the GUI-components, which have been redesigned and updated to GComponents. We have also updated the whole timer application, so it is able to communicate with Oasis.

At the end of the fourth sprint, we have tested the application to check the new functionality and to understand how reliable the new functionality were.

As mentioned before we have had many problems through this project, but we still think that it has been a very exciting project, specially working in a multiproject have been exciting. This semester has also been different than most other semester, which has given us a lot knowledge about customer relations, cooperation and management of both clients and co-developer.

Although we had a lot of trouble during this project, specially when we were working with the timer application, we have fulfilled the most major requirements for the Timer application, which was required by the clients.

21.2 Perspective

Through this chapter we will evaluate the process of this semester including the cooperation with other groups. Furthermore we discuss some of the major issues we have met during this semester.

During the first sprint, we have collaborated with another group to collect the most important requirements for the GIRAF project. We had meetings with different clients and we collected requirements and recorded them in the form of a list with prioritization. The biggest issues we faced during this sprint were that we had some misunderstanding between different groups. Some of the other groups had greater expectation of how we should handle the interviews with the clients and how we should have structured the requirements.

In the second, third and fourth sprint we have worked with the timer application. This application needed to be redesigned and some missing functionality had to be implemented. Implementing a new overlay, sound and GComponents was some of the new functionality we did during those sprints.

One of the issues we faced during the first sprints was that we used a lot of time on understanding the code from the last year. Fixing bugs was also an issue we frequently faced through the last three sprints, before we could implement new functionality to the application. Due to the time limitations for each sprint, we did not have time enough to work on the report. Therefore the documentation for the each sprint was done in the beginning of the next sprint.

Another issue we needed to take care of was, that the timer project was deeply depending on Oasis, which means that every time Oasis was updated, we needed to make changes the timer application to work with the new Oasis update.

But overall the process was fine, and we have learned a lot during this project.

21.3 Setup Guide For Next Semester

We have made a setup guide, which will walk you through the essential parts of setting up the project, in order to continue Timer's development. We have made a guide for GIT, Android Studio and a guide for understanding the code. E

Bibliography

[1] Nikolaj Andersen, Anders Frandsen, Rune Jensen and Michael Lisby, "Android Application Management System for Children with Disabilities," tech. rep., Department of Computer Science.

[2] Sarah Geagea, Sheng Zhang, Niclas Sahlin, Faegheh Hasibi, Farhan Hameed, Elmira Rafiyan and Magnus Ekberg , "Software Requirements Specification," 2006.

[3] "System overlay description." `https://sourceware.org/binutils/docs/ld/Overlay-Description.html`. Accessed: 27-05-2014.

[4] "Android service." `http://developer.android.com/reference/android/app/Service.html`. Accessed: 27-05-2014.

[5] "The wiki webpage." `http://cs-cust06-int.cs.aau.dk/projects/sw6f14/wiki`. Accessed: 27-05-2014.

[6] "Google analytics." `http://cs-cust06-int.cs.aau.dk/projects/sw6f14/wiki/Google-analytics`. Accessed: 28-04-2014.

[7] "Profile selector guide." `http://cs-cust06-int.cs.aau.dk/projects/gui/wiki/GProfileSelector`. Accessed: 23-05-2014.

[8] Anders Jensen and Danial Bøcker Sørensen and Tom Petersen and Jakob Albertsen, "Cars A Voice Controlled Game," tech. rep., Aalborg University, 2013.

[9] Søren Knudsen and Christian Lykke and Søren Enevoldsen and Sam Olesen, "GIRAF: Zebra," tech. rep., Aalborg University, 2013.

[10] Christian Mortensen and Brian Holbech and Nedim Cokljat, "Tortoise An Application for Managing Autistic Children's Life Stories in GIRAF," tech. rep., Aalborg University, 2013.

Part VII
Appendix

A Questions for First Meetings

These questions were made in collaboration with group SW601F14, and were used for the initial interviews with the Product Owners.

- Which tools do you use in every day life?
- Which tool functionality do you like the best?
- Which of the previous applications have you worked with, if any?
  - How was your experiences with these applications?
  - Is there anything you feel is missing or could be improved?
  - Is there anything in the application that was not important?
- How important is it to partition each application from other applications?
- How important is flexibility in the system?
  - Should colors, text and/or pictures be flexible?
- Is it important that the different applications look alike? I.e. follow the same general theme.
- How much information about the citizens is needed in the system?
- What is most important for you, new functionality or the existing solution to be completed?
- How will you prioritize your current wishes?
- If you did not have to consider us, how should the product be?
- Do you have any specific wishes for the 55" screen? (Only for Birken)

B Follow Up Questions

These questions were made in collaboration with all groups in the multiproject, and were used as follow questions in the second meeting with the Product Owners.

- Should applications be opened in the GIRAF Launcher or in the Android standard Launcher?
- Should the applications run on 3G or Wi-Fi?
- Where should you choose your profile? In the Launcher or in each application?
- How should the timer notify when the time has passed and what should it do afterwards?
- May we use Google Analytics?
- Should it be possible to have more than one Guardian profile?
- Should it be possible to switch between Guardian profiles?
  - If yes, from the GIRAF Launcher or from each individual applications?
- Should it be possible for citizens to use Croc to create pictograms or to take pictures and save them locally?
- As standard, how many applications should the citizens be authorized to use?

C Specification Requirements

Full version of the specification requirement made in the first sprint.

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 must be fulfilled.
(2) - Requirements that must be fulfilled, but which are not essential.
(3) - Requirements that must be fulfilled, but only if there is additional time available.

C.2 Profiles

1. There are two types of profiles:
   They are divided into organizations, this means that a profile from one organization does not have access to another organization's profiles. (1) See user story number 6.
   (a) Guardian profile:
      i. Has access to all functions in all applications. (1)
      ii. Can give authorization to which functions in the applications, a citizen profile has access to. (1)
      iii. Can add new guardian- and citizen profiles on both tablets and web. (1)
      iv. Can edit existing profiles, both guardian- and citizen profiles. (1)
      v. Can remove existing profiles, both guardian- and citizen profiles. (1)
   (b) Citizen profile:
      i. Has access to functions and applications, which are authorized by a guardian profile. (1)

ii. As standard, citizen profiles does not have access to any functions. (1)
iii. Cannot add new profiles. (1)
iv. Citizen profiles can never use the three androids buttons (back, home and menu). (1)

2. It must be possible to transfer a citizen profile to another organization, i.e. that a citizen profile's settings and private pictures transfers to another organization so that they can be used there. (3)
3. Shifting between profiles requires approval, e.g. by the use of QR code. (3)

C.3 General

1. No applications must close, stop or crash unexpectedly. (1)
2. All applications must synchronize with the global database, once a day, if there is access to the Internet. (1)
   (a) It must be possible to synchronize manually with the global database. (2)
3. All applications must be able to save settings for each individual profiles. (1)
   (a) It must be possible to copy settings from one profile to another. (2)
4. All applications must run in landscape mode (horizontal). (1)
   (a) Sequences and Week Schedule must be able to use portrait mode. (1)
5. All applications, except the Launcher, must have a 'close' button. (2)
   (a) When the button is pressed, it must close the application, if the user has permission to close applications. (1)
   (b) If the button is pressed and the user does not have access to close the application, the application must be able to close with the help of a guardian, e.g. by scanning QR code. (1)
6. When and only when there is a permitted input, there must be some kind of response, which makes sense in context, but otherwise this is up to the individual developer. (2) see user story number 4.
   (a) Applications with functions targeted towards citizens must only use singleclick, drag and drop, and swipe input. (2)
   (b) If there is a function, which should only be usable by a guardian, then all Android gestures are permitted. (2)
   (c) All applications must be named with meaningful Danish names. (1)
7. If a pictogram is not yet saved to the global database, then all applications must show a 'pending' icon, so the guardian can see that it is not yet saved to the database. (2) see user story number 8.

   (a) Pictograms must be able to be marked as private, so they are not uploaded to the global database. (2)

8. All applications must be able to play a pictogram's associated sound. (2)

C.4 General GUI

1. The general color scheme must be black and white (3)
   (a) It must be possible to choose a color theme for each application. (2) see user story number 3.
   (b) Each profiles must be able to have their own chosen color theme. (2) see user story number 3.
2. All applications must use the same GUI components. (2)
3. All applications must indicate if there is a sound attached to a pictogram. (2)

C.5 Database

1. Pictograms, photos, sound clips, profiles and categories must be able to be saved in the database. (1)
   (a) Citizen profiles are split into groups. (1)
   (b) Profiles must be split into organizations. (1)
   (c) Pictograms and photos must be handled the same way in the database. (1)
   (d) Sound clips must be handled separately. (1)
   (e) Categories must be handled separately. (1)
2. Global database:
   (a) It must be possible to synchronize the local database with the global database. (1)
   (b) Must contain all standard pictograms. (2)
   (c) Guardian profiles must be able to add new pictograms to the database. (1)
   (d) Guardian profiles must be able to remove and edit pictograms they have added to the database. (2)
   (e) Guardian profiles must be able to copy standard pictorgrams, edit them and then save the copy to the database. (2)
   (f) Guardian profiles must not be able to remove standard pictograms. (3)
3. Local database:
   (a) All contents of the local database must be able to be saved in the global database, this includes settings. (1)
   (b) It must be possible to synchronize with the global database. (1)

i. It must be possible to synchronize manually. (1)
    A. There must be a synchronize button for the guardian in all applications. (2)

ii. It must be possible for the synchronization with the global database to happen automatically. (2)
    A. As standard, automatic synchronization should happen once a day when there is wifi. (2)

(c) All applications must use the same local database, e.g. different application does not have their own local database. (2)

C.6 Launcher

1. The launcher must change the functionality of the three android buttons (back, home and menu), depending on the user's permissions. (1)
2. The launcher must handle switch of color themes for each application. (3)
    (a) If you are logged in as a guardian, it must be possible to change launcher. (1)
    (b) If you are logged in as a citizen, it must not be possible to change the launcher. (1)
3. The launcher must keep track of which applications should be visible. (2)

C.7 Pictocreator

1. Must be able to use the camera to create a pictogram. (1) see user story 26.
2. Must have a function to draw, save and edit pictograms. (2) see user story 8 and user story 6.
3. Must have a function to record, save and edit sound clips for a pictogram. (3) see user story 8 and user story 6.
4. It must be possible to mark a pictogram as private, so that it will not be saved to the global database. (2)
5. It must be possible for a pictogram to have a title, which can be freely placed on the pictogram. (2)

C.8 PictoSearch

1. Must be able to delete pictograms. (1)
2. Must be able to show all pictograms, in their specific categories. (1)
3. Must be able to search through pictograms. (1)

C.9 Categorizer

1. Must be able to create new categories of pictograms. (1)
2. Must be able to view, add and remove pictograms from existing categories. (1)
3. Must be able to delete existing categories. (2)
   (a) When a category is deleted, a warning dialog box must be shown asking the guardian if they are sure they want the category to be deleted. (2)
4. Guardian profiles must be able to connect a specific category to a specific citizen profile. (1)
   (a) It must be possible to copy a category from one profile to another. (2)

C.10 Communication Tool

1. Must be able to view pictograms.
2. Must be able to play the sound of a corresponding pictogram, when the pictogram is pressed. (1)

C.11 Sequences

1. Citizen profiles must be able mark how far they are in a sequence. (1) see user story number 14
2. The guardian can change whether sequences are shown horizontally or vertically for the citizens. (2)
3. Guardian profiles must be able to view, create, edit and remove sequences. (1)
4. There must be multiple ways to view sequences. (1)
   (a) It must be able to show one pictogram, three pictograms or a complete sequence. (1)
5. It must be possible for a guardian profile to authorize a citizen profile to be able to make sequences. (2)

C.12 Life Stories/Week Schedule

1. Citizen profiles must be able to mark how far in a sequence they are. (1) see user story number 14.
2. Colors for the week schedule must follow the international color standard for days. (1)
   (a) These colors must always be viewable, even if you have scrolled down in a sequence. (3)

3. 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. (1)
4. This application must function as a tool to plan daily activities for citizens. (1)
5. This application must be able to help citizens to formulate their day in pictograms. (2)
6. This application must be able to allocate periods where citizens can choose between preselected activities. (1) see user story number 11.
7. Guardian profiles must be able to define how many pictograms are shown at a time. (2) see user story number 18.
   (a) It must be able show one pictogram, three pictograms or a complete day. (2)
8. Sequences of pictograms must have the same size and be locked in a grid. (1)
9. When a citizen has a choice to make, other functions must be locked, until a choice has been made. (2)

C.13 Timer

1. This application must be able to reset, edit, start and stop the timer. (1)
2. This application must be able to make the timer visible in all other applications that the citizen profile has access to. (2)
3. Citizen profiles can only see the time. (2)
4. The timer must always count down. (1)
5. When the time has run out, there must be a notification and a sound, that gives the user two options (1)
   (a) It must be able to move on to the next planned activity (this function must be able to be turned off). (2)
   (b) A button that stops the timer, but which requires a guardian to continue. (1)
   (c) A notification must have a short iconic/recognizable sound, which indicates that the time has passed. (1)
      i. There must be a possibilities for a guardian profile to choose which sound that is used as the notification. (1)
      ii. One of the possibilities for the sound must be the sound of a kitchen timer. (1)
   (d) A notification must contain a pictogram, that explains to the citizen that the time has passed. (1)
6. The list of buttons must be able to be listed horizontally or vertically. (1)

7. There must be a representation of the Product Owners' timers. (1)
    (a) The clock face must always represent 1 hour. (1)
    (b) The time indicator (that which shows how much time there is left) is pulled from 0 and is pulled clockwise. (1)
    (c) When the timer is counting down, it will go counterclockwise. (1)
8. There must be three alternatives for their timers. (1)
    (a) Hourglass. (1)
    (b) Progress bar. (1)
    (c) Digital watch. (1)

C.14 Train

See user story number 30 and 32.

1. A train leaves the main station after it have been loaded with pictograms and drives to one or more stations where some of the pictograms will need to be unloaded before it reaches the train depot at the end. (2)
    (a) A pictogram must be unloaded on a station with a category the pictogram belongs to. (2)
    (b) The train cannot leave a station before all the right pictograms are unloaded from the train. (2)
2. It must vary how long a train takes to reach a station. (2)
3. The picture of the driver must be the picture associated with the citizen profile. (2)
4. There must be instructions for the game. (2)
    (a) In the beginning of the game, there must be instructions with arrows that indicate that a pictogram should be dragged down into a cart. (2)
5. Carts must be named. (2)
6. The train must be able to change colors. (2)
7. The stations must be able to have the color of their category. (2)

C.15 Cars

See user story number 31.

1. You must be able to steer the car by the use of your voice volume. (1)
2. While driving there must be obstacles that is avoided by changing the volume of your voice. (1)

C.16 New Entertaining Learning Tools (Not requirements)

1. Barrels, a game where a barrel is shown, where possibly a sound comes from it, and the child will have to guess what is in the barrel. (3)
2. PuzzleStory, a game where a collection of specific pictograms are shown, where the child then has to organize these into a coherent story. (3)
3. RepeatStory, a story where repetition is important. A good example of this is the song "12 Days of Christmas", where each new day repeats the previous days. (3)

The Product Owner suggests that we can be creative and mix these ideas together with existing games. Such as when you have completed a game of Train, a barrel will appear as a fun surprise for the child.

D User Stories

We have used the user stories from report [8], [9] and [10] from the earlier years as a foundation to create the user stories for this semester. The user stories will be used to describe the application of the system in a common language which the customers 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 customers seeks.

1. Adding profiles
    - A new citizen has started at the institute so as a guardian I want to able to create a new autistic profile for that citizen.
2. Deleting profiles
    - A citizen is no longer part of the institute so I as a guardian want to be able to remove his profile from the system. I want to be able to remove his personal pictograms from the system, but be able to save them to a local storage device if needed before they are deleted.
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 stored for the given citizen profile.
4. Application responding to the user
    - A citizen is playing one of the learning games in the system. As the citizen I expect the application to respond to the input I give the system so I see that something is happening. This should be general for the whole system.

5. Opening an application through the launcher
    - As a guardian or citizen I want to work with many of the applications from the system. I open the launcher and select the application I want to work with. When I am done, I go back to launcher and simply select the next application I want to work with.
6. Creating a pictogram
    - As a guardian I want to create a pictogram to describe an action. I open the pictogram creator, where I draw the pictogram as I want it. I add a caption and record myself pronouncing the action. I then assign it into a suited category and save the pictogram. I expect that the pictogram will be accessible for other guardians from the institute I work as well as myself.
7. Deleting a pictogram
    - A citizen has a personal pictogram that is no longer needed, so I as a guardian want to remove it from the system.
8. Editing an existing pictogram
    - A pictograms caption does not fit and it needs to be changed. The pictogram also have sound attached. As a guardian I select the specific pictrogram and open it in pictogram creator. I now replaces the caption with a new, record a suitable sound and stores the edited pictogram. I expect that the changes to the pictogram now are store and is changed for all application using this specific pictogram. I also expects an indication of whether a pictograms changes have been uploaded to the database or if it still to be done.
9. Adding a new pictogram to a specific category
    - As a guardian I am in the categorizer application and I notice that there is not a pictogram for the activity I need. So I press the button for adding a new pictogram to a category and I am sent to the pictocreater so I can either take a photo or draw my own pictogram. When I am done I save it and I am returned to the categorizer application where I can choose the new pictogram which I then add the category. I expect that the newly created pictogram is not accessible to any other institutions except my own.
10. Removing a pictogram for a specific category
    - As a guardian I have accidentally added a pictogram to a category it did not belong to, so I want to be able to remove it from that category.
11. Planning activities
    - A citizen needs a schedule to tell him what he is doing a specific day. I as a guardian want an application that lets me put together a schedule that I can attach to the autistic profile. When making the schedule I make a sequence of pictograms one for each activity, though I can also

make an activity free. A free activity is an activity where the citizen has a choice between multiple activities. Each activity can lead to a stored sequence describing how to complete that activity.

12. Keeping track of the schedule
    - As a citizen I look at the schedule to see the activities planned for the day. I check the color of each day to find which day I have to look at in the schedule. I then mark the activity that have been done and move on to the next activity.
13. Creating a new sequence
    - A new citizen is starting at the institute. As a guardian I want to create a new sequence for this citizen because the stored sequences does not fit this specific citizen. I can then use this new sequence to explain activities for the citizen. I expect that the newly created sequence is stored and accessible only to my institute.
14. Mark actions as complete
    - As a citizen I want to be able to mark actions as completed so I can keep track of my current progress in a sequence. There should be multiple options for how to mark actions as complete so I can choose one I find appropriate.
15. Different views of sequences
    - A citizen cannot always understand if there are to many sequences in a week schedule, and a citizen sometimes pick another sequences instead of the one that was meant to be. Therefore the citizen want to have as few as possible sequences, and it is possible as a citizen to create sequences for month, week, day or simply a half day schedule.
16. Edit a sequences title and images
    - A citizen would like a new brand of cereal instead of the old brand for breakfast. As a guardian I want to open the old breakfast sequence and replace a specific pictogram with the new pictogram. Then I want to save the sequence with a new title so I still have access to both. I can then help the new citizen with breakfast by using the new sequence while my colleagues can still access the old sequence.
17. Deleting a sequence
    - A specific sequence is no longer necessary in the system. As a guardian I select a sequence and delete it. I expect that the deleted sequence is removed from the system.
18. Opening a sequence
    - The entire institute is going for a walk. As a guardian I want to show a sequence for a citizen so I can help them with the correct order of actions for putting on outdoor clothes. I expect the application to provide the possible to view the sequences displaying all, 3 or 1 pictogram in sequence, so I can help the citizen.

19. Helping a citizen communicate
    - As a citizen I can have a difficult time speaking, so I want an application that can play the sound of a pictogram when I press it.
20. Using the timer application
    - A citizen needs to do an activity for a set amount of time. As a guardian I open the timer application and allocate the appropriate time. I select a suited visual representation of the timer and start it. I expect the timer application to show the representation for the allocated time without the screen turning off. When the time runs out I expect the timer application to notify with a sound that the time has passed and the citizen shall move on to the next activity.
21. One application available in another
    - A citizen has a free time slot and wish to play the "Train" application. As a guardian I want to able to set a timer so the citizen knows how long he can play. I want the timer to be visible in the "Train" application, so he always knows exactly how long he can play. This should be general for the entertainment and learning applications.
22. Creating a category
    - A large collection of pictograms have been added to the system. As a guardian I want to create suitable categories so can I can add these pictograms to the category I deem they belong to. I expect that the created categories persist until they are deleted. I also want to be able to link the created category to specific profiles.
23. Deleting a category
    - As a guardian I have accidentally made a category that will not be used, so I want to able to delete it.
24. Viewing the collection of categories
    - When I as a guardian enter the categorizer application I expect to only see the categories associated with my profile or the autistic profile I am currently helping. I also expect to be able to press a button to see all the categories and pictograms that are available to me. When looking at all the categories I expect I am able to see what each category contains.
25. At the end of the day
    - When a citizen gets home they want to tell their parents about their day. As a guardian I want to able to help them make a life story of their day. I make a template for them where I access the categorizer and choose a set of pictograms for each activity they have done this day. When the citizen then use this template to create their life story they can choose between the pictograms and place them as they like. When they get home they can show off their life story on a tablet.
26. Take a photograph and use it as a pictogram

- Some citizens need a real photo before they can understand the meaning of a pictogram and link it to the real world. As a guardian I want to be able to open the application for creating pictograms, take a photo and use it to create a new pictogram. I go to the sequence application and use the newly created pictogram in a sequence.

27. Changing to another citizens profile in a given application

    - As a guardian I want to change the settings of some citizen profiles in a given application. I open the list of profiles and select the profile which settings I want to change. I expect that all the settings appears as they are for a given profile and changes to these settings will be stored in place of the old settings. I adjust the settings for all the profiles that I wanted to change and then return to the given application.

28. Remember what I do

    - As a guardian I expect that anything I do or change will be stored so I can use it on any tablet I choose when I am logged in on my profile.

29. Safe navigation

    - As a citizen I want to be in an application mode where I cannot accidentally navigate to parts of the system I am not supposed to use, so I do not get lost or loose focus on the task I should do.

30. Training citizens sequences view

    - For helping citizens in a fun way they use the "Train" application to understand how they can place something in a sequences. They have to drop the things at the right place.

31. Training a citizens voice

    - Some citizens cannot control the level of their voice. As a guardian I open the training game to help the citizen control their voice and give them an understanding of how to use it. I expect the game to use the citizens voice to achieve some kind of goal so they find it entertaining while using it and thereby help motivate them to complete the training.

32. Training a citizens understanding of pictograms

    - As a guardian I want to train a citizens understanding of pictograms and their relation to each other. I open the learning game application for understanding pictograms, if this is not the first time then I expect that the selection of categories is the same as from last session. I can edit the list of categories that should be present in the game and start the game. I expect the game to arrange a selection of pictograms from each selected category and choose one category which some of the pictograms are associated with. The citizen then tries to find the pictogram associate with the chosen category.

E Setup guide for the next semester

This section will explain how to setup the Timer application for further development. We made this section because we had a lot of problems getting started.

E.1 Git

In order to start development on the Timer project, you need to download the project from the git server.

Installing Git Bash

Start by downloading and installing git bash: *http://git-scm.com/downloads*.

When you have installed git bash, open the application and navigate to the folder that should contain your project folder, use the following commands:

- "cd *folder path*".
- "cd .." to go up a folder.
- "pwd" to view the current folder path.

Navigating to Project Folder

After navigated to the right folder, you are ready to download the project from the git server. Use the following commands to get all the project files, this will create a new folder called *wombat*:

1. "git clone "http://cs-cust06-int.cs.aau.dk/git/wombat""

2. "cd wombat"
3. "git submodule init"
4. "git submodule update"

Updating

After you have downloaded the project files you are ready to develop and you can go to the next section to see what to do now. The next part of this section will explain how to update you files and how to commit you changes. In order to update the files you can use the following commands:

- "git pull" this command will download all changed files

If you get an error telling you that you are not on a branch, the you can use the following commands:

- "git checkout master" to select the master branch

If you have to update a sub project you have to *cd* to that folder and pull in there. Also remember if you update a sub project you have to commit that you have updated a sub project in the parent folder.

Discard Changes

If you have any changes that you want to discard you can use the following command:

- "git checkout *file path*"

Commit

In order to commit the changes you can use the following commands:

1. "git add *file path*" to select a file that should be added to the commit command, you can use "–all" to select all files
2. "git commit -m"*commit message*"" it is important to leave a good commit message, so that others and yourself in the future can see what has been changed in this commit.
3. "git push origin master"

If you have to commit a sub project you have to *cd* to that folder and commit in there, and afterwards you have to commit that you are using a new version of that sub project.

E.2 Android Studio

When you have downloaded the project files you need to download android studio and all the tools you need:

Installing Android Studio

You will need the following things:

- Android studio 0.4.6 (you need this version) *http://tools.android.com/download/studio/canary/0-4-6*
- Java Developer Kit *http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html*
- Gradle 1.9 *http://www.gradle.org/downloads* in the bottom there is a menu to choose version, download gradle-1.9-bin.zip.

Remember to unzip the android studio file, and unzip the gradle file and installing the JDK before opening android studio.

Setting Up the Android SDK

Start by downloading and installing the Android SDK at *http://developer.android.com/sdk/index.html?hl=sk* in the bottom you can choose *DOWNLOAD FOR OTHER PLATFORMS* under *SDK Tools Only* select the recommended installer.

After installing the SDK the SDK Manager will appear, here you have to select the following:

- Tools
- API 19
- API 17
- API 15
- Under "extras" select
  - Google USB driver
  - Intel x86 Emulator Accelerator (HAXM)

Importing Project

When you have installed both the JDK and android studio you are ready to import the project:

1. Import the project (browse to the project folder)

2. Select use custom gradle (browse to the gradle folder)

After importing you might get an error saying that android studio cannot find you SDK. Open *File* and select *Project Structure* under Android SDK you can set the path to your SDK. If you don't remember where you installed the SDK it might be under *C:/Users/User Name/AppData/Local/Android/android-sdk*.

Emulator Setup

Open the AVD Manager, and create a new Android Virtual Device, change the following settings:

- Device - 10.1" WXGA (Tablet) (1280 x 800 mdpi)

- Target - API 19

- CPU/ABI - Intel Atom x86

- Skin - No Skin

- Memory Options - RAM 768

- Internal Storage - 1 GB

- Emulation Options - Use Host GPU

There are two apks that you need to run Timer.

- Launcher - https://www.dropbox.com/s/wrff3pqc09z07wd/launcher.apk
- Local DB - http://cs-cust06-int.cs.aau.dk/attachments/download/269/local-db-debug-unaligned.apk

If the emulator does fails to start or you have any other problems you might have to install Intel HAXM from *https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager*. If you cannot install HAXM you have to turn of Hyper-V. Open your *Control Panel*, click *Programs*, click *programs and features*, and in the left side click *Turn windows features on or off*, and the uncheck *Hyper-V*.

Installing APKs

Start by navigating to your Android platform-tools folder, mine is located at: *C:/Program Files (x86)/Android/sdk/platform-tools* otherwise it might be located at *C:/Users/User Name/AppData/Local/Android/android-sdk/platform-tools*.

Now open a Command Prompt and type the following:
adb install *apk filename*

E.3 Understanding the Code

In this section we will briefly go though the different folders and describe what goes on in them.

E.3.1 Res

The res is the folder that contains all of your layout, i.e. the graphical part of the application.

Layout

The layout contains all the xml files that markup the application.

Values

Contains files that specifies global values, e.g. if you have a button with a specific content you will create a string variable in the strings.xml file and reference the variable where you set the buttons content.

E.3.2 Src

The src contains all the java code.

MainActivity

MainActivity is in control of starting the application and setting everything up, furthermore it loads all the data from the database.

CustomizeFragment

This is where the major part of the application is handled. All of the button events is handled here by event listeners. This is also the file that loads the graphical part of the application.

Overlay

This the service that is in charge of creating the overlay, this class does not contain a lot of code, but it calls many other functions from other classes.

E.3.3 DrawLib

This is the project that is in charge of controlling all the timers.

E.3.4 TimerLib

Unlike what the name suggest, this project handles all communication with the local database.

E.3.5 Wheel

This is the project that is in charge of drawing all the timers.

E.3.6 Sub Projects

These sub projects are project that you should not change any code in. These are project that are worked on by other groups.

GIRAF_Components

This project handles the unified design throughout the whole GIRAF project.

OasisLib

This project handles all the data that needs to be saved locally, furthermore it also manages all communication with the remote database.

pictogram-lib

This project contain all classes that is used for the pictograms.

E.3.7 Debugging Timer

In order to debug the Timer and you don't plan on using a tablet, you can start the Timer without the need of the Launcher, the code from *MainActivity* in listing E.1 is the code that loads profiles, if you comment out the code it will run without the need of the Launcher.

```
Bundle extras = getIntent().getExtras();
if (extras != null) {
  guardianId = (int)extras.getLong("currentGuardianID");
  childId = (int)extras.getLong("currentChildID");
  color = extras.getInt("appBackgroundColor");
} else {
    new AlertDialog.Builder(this)
            .setTitle("Timer")
            .setMessage(R.string.launch_from_giraf)
            .setPositiveButton(android.R.string.yes, new ↩
                DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, ↩
                    int which) {
                    finish();
```

```
                        Process.killProcess(Process.myPid());
                    }
               }).show();
  color = getResources().getColor(R.color.Black);
}
```

Listing E.1: Code listing that loads profiles

F Code Review

Overall the code is neat, organized and well-documented, and it's difficult to make any sort of pointer for improvement on well-documented code. Sometimes there is a bit excessive use of comments (ex. commenting that frameWidth sets the frame width is a bit too much, we can read it out of context). However most of the time it is well placed, in particular for stuff like TYPE_SYSTEM_OVERLAY to force the overlay to always be visible.

A clear indication of what Runnable is would be appreciated (currently I am assuming it is a tick-able timer?)

The code is also not taking into account if the system is running a non-existent guardian. In the target environment this naturally won't be the case, but it means the system can't be debugged without first running through the Launcher? This could be a potential problem if Launcher becomes unavailable (due to maintenance or similar).

Consider abstracting the Overlay class and creating several subclasses for each overlay type; one for progress bar, one for hourglass etc.. Doing so will better allow future developers to expand the types of overlays available by creating their own subclass of Overlay.

F.1 Overlay Class

```
package dk.aau.cs.giraf.wombat;

        import android.app.Service;
        import android.content.Intent;
        import android.graphics.PixelFormat;
        import android.os.Handler;
        import android.os.IBinder;
        import android.view.Display;
        import android.view.Gravity;
        import android.view.View;
        import android.view.WindowManager;
        import android.content.Context;
```

```java

        import java.util.ArrayList;

        import dk.aau.cs.giraf.TimerLib.Guardian;
        import dk.aau.cs.giraf.TimerLib.SubProfile;
        import dk.aau.cs.giraf.wombat.drawlib.*;

public class Overlay extends Service {
    View view;
    private Handler mHandler;
    private Runnable mRunnable;

    @Override
    public IBinder
    onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();

        //Get the needed profiles
        Guardian guard = Guardian.getInstance();
        SubProfile sub = guard.getSubProfile();
        sub.getAttachment();

        //Find the size of the screen
        WindowManager wm = (WindowManager) getSystemService(↩
            Context.WINDOW_SERVICE);
        Display display = wm.getDefaultDisplay();

        DrawLibActivity.frameHeight = display.getHeight()/↩
            DrawLibActivity.scale;
        DrawLibActivity.frameWidth = display.getWidth()/↩
            DrawLibActivity.scale;
        view = GetWatchViews(sub, DrawLibActivity.frameWidth↩
            );//Finds the watch that will be shown

        //Layoutparams decides how the overlay behaves
        WindowManager.LayoutParams params = new ↩
            WindowManager.LayoutParams(
                DrawLibActivity.frameWidth,
                DrawLibActivity.frameHeight,
                WindowManager.LayoutParams.↩
                    TYPE_SYSTEM_OVERLAY,//System overlay ↩
                    forces the overlay to always be visible
                WindowManager.LayoutParams.↩
                    FLAG_NOT_FOCUSABLE//Makes the overlay ↩
                    non focusable
                        |WindowManager.LayoutParams.↩
                            FLAG_NOT_TOUCH_MODAL//Sends any ↩
                            touch events to the activity ↩
                            underneath
                        |WindowManager.LayoutParams.FLAG_WATCH_OUTSIDE_TOUCH,//Checks for touch events outside the overlay
                PixelFormat.TRANSLUCENT);//Makes the overlay transparent


        params.gravity = Gravity.RIGHT | Gravity.TOP;//Places the overlay in the top right corner of the screen
        params.setTitle("Load Average");//Sets the title of the overlay
        wm.addView(view, params);//Adds the watch to the overlay

        //Intent for starting the donescreen
        final Intent overlayIntent = new Intent(Overlay.this, DoneScreenActivity.class);
        overlayIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
        //This runnable is a container that contains code, that will be executed when it's called
        mRunnable = new Runnable() {
            public void run() {
                getApplicationContext().startActivity(overlayIntent);
                getApplicationContext().stopService(MainActivity.svc);
                MainActivity.svc = null;

            }
        };

        /* Set the delay of the intent to the time of the timer + 1 second
         * otherwize the user will have a hard time seeing the timer reach 0
         * Also the code in the runnable is delayed, so it's called when the timer ends*/
        mHandler = new Handler();
        mHandler.postDelayed(mRunnable, (sub.get_totalTime()+1)*1000);

    }

    @Override
    public void onDestroy() {
        super.onDestroy();

        //When the timer ends, it removes the watch
        if(view != null)
        {
            ((WindowManager) getSystemService(WINDOW_SERVICE↩
                )).removeView(view);
            view = null;
        }

        //Stops the runnable from being called
        mHandler.removeCallbacks(mRunnable);
    }

    //Finds the appropriate watch that will be shown in the ↩
        overlay
    public View GetWatchViews(SubProfile sub, int frameWidth↩
        ) {
        switch (sub.formType()) {
            case ProgressBar:
                return new DrawProgressBar(↩
                    getApplicationContext(), sub, frameWidth↩
                    );
            case Hourglass:
                return new DrawHourglass(↩
                    getApplicationContext(), sub, frameWidth↩
                    );
            case DigitalClock:
                return new DrawDigital(getApplicationContext↩
                    (), sub, frameWidth);
            case TimeTimer:
                return new DrawWatch(getApplicationContext()↩
                    , sub, frameWidth);
            case TimeTimerStandard:
                return new DrawStandardWatch(↩
                    getApplicationContext(), sub, frameWidth↩
                    );
            default:
                return null;
        }
    }
}
```

Listing F.1: The Overlay Class

F.2 SearchClass

```
package dk.aau.cs.giraf.pictosearch;

//import dk.aau.cs.giraf.categorylib.CategoryHelper;
//import dk.aau.cs.giraf.pictogram.Pictogram;
//import dk.aau.cs.giraf.pictogram.PictoFactory;
import java.util.ArrayList;
import dk.aau.cs.giraf.oasis.lib.models.Category;
import dk.aau.cs.giraf.oasis.lib.models.Pictogram;
import dk.aau.cs.giraf.oasis.lib.controllers.↩
    PictogramController;

```

```
/**
 * Created by Tobias on 25-03-14.
 */
public class SearchClass
{
    private PictoAdminMain Outer;

    SearchClass(PictoAdminMain ou)
    {
        Outer = ou;
    }

    public ArrayList<Object> DoSearch(String tag, String[] ↩
        input, ArrayList<Object> AllPictograms)
    {
        // ToDo: DoSearch_Category currently receives an ↩
            empty array, fill it with cats pls
        if (tag.equals("Tags"))
        {
            return DoSearch_Tags(input, AllPictograms);
        }
        else if (tag.equals("Pictogrammer"))
        {
            return DoSearch_Pictogram(input, AllPictograms);
        }
        else if (tag.equals("Kategorier"))
        {
            return DoSearch_Category(input, AllPictograms);
        }
        else
        {
            return DoSearch_All(input, AllPictograms);
        }
    }

    // PICTOGRAM IMPLEMENTATIONS

    private ArrayList<Object> DoSearch_All(String[] input, ↩
        ArrayList<Object> AllPictograms)
    {
        // ToDo: DoSearch_Category currently receives an ↩
            empty array, fill it with cats pls
        ArrayList<Object> Result = new ArrayList<Object>();
        Result.addAll(DoSearch_Pictogram(input, ↩
            AllPictograms));
        Result.addAll(DoSearch_Category(input, AllPictograms↩
            ));
        return Result;
    }

    private ArrayList<Object> DoSearch_Pictogram(String[] ↩
        input, ArrayList<Object> AllPictograms)
    {
        ArrayList<Object> lst = new ArrayList<Object>();
        for (Object o : AllPictograms)//review: Bad naming of o, instead use object
        {
            if (o instanceof Pictogram)
            {
                Pictogram p = (Pictogram)o;//review: Bad naming of p, instead use pictogram
                if (p == null || p.getName() == null) continue;

                for(int i = 0; i < input.length; i++)
                {
                    if (p.getName().toLowerCase().contains(input[i]))//review: should input not also be toLowerCase?
                    {
                        lst.add(p);
                        break;
                    }
                }
            }
        }
        return lst;
    }

   //review: Does this function do anything?
    private ArrayList<Object> DoSearch_Tags(String[] input, ArrayList<Object> AllPictograms)
    {
        ArrayList<Object> lst = new ArrayList<Object>();

       for (Object o : AllPictograms)
       {
           if (p == null || p.getName() == null) continue;

       //review: Does this loop do anything?
           boolean added = false;
           for(int i = 0; i < input.length; i++)
           {
               /*
               for (String tag : p.getTags())
               {
                   if (tag.toLowerCase().contains(input[i]))
                   {
                       lst.add(p);
                       added = true;
                       break;
                   }
               }
               */
               if (added) break;
           }
       }
        return lst;
```

```java
    }

    // CATEGORY IMPLEMENTATIONS

    private ArrayList<Object> DoSearch_Category(String[] input, ArrayList<Object> CatSearchList)
    {
        ArrayList<Object> lst = new ArrayList<Object>();

        for (Object o : CatSearchList){//review: Bad naming of o
            if(o instanceof Category)
            {
                Category pc = (Category)o;//review: Bad naming of pc, instead use pictogram

                if (pc == null || pc.getName() == null) continue;

                boolean added = false;

                for (int i = 0; i < input.length; i++)
                {
                    if (added) break;
                    else if (pc.getName().contains(input[i]))//review: Should pc.getName() not contain toLowerCase? and the same goes for input.
                    {
                        lst.add(pc);
                        added = true;
                        break;
                    }

                }
            }
        }

        return lst;
    }
}
```

Listing F.2: The Search Class