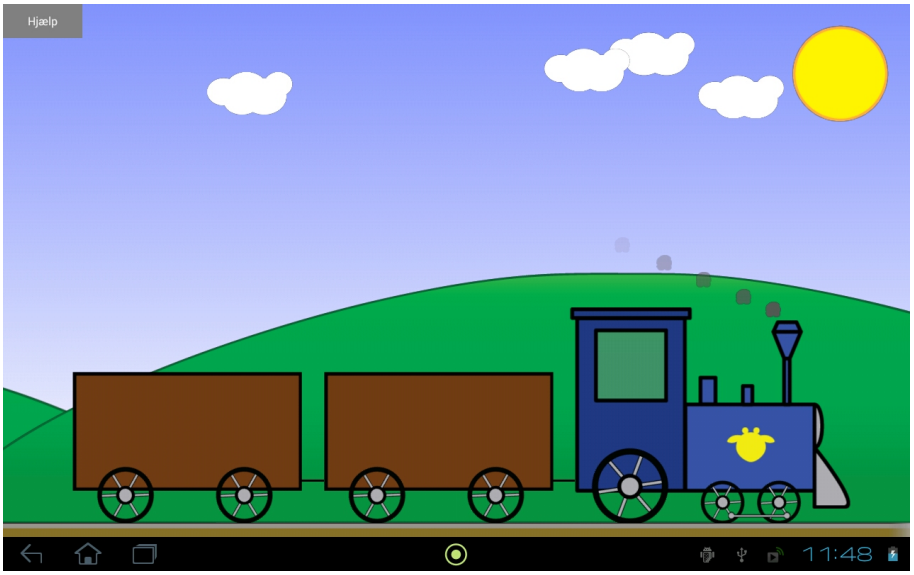
P6 Project

The Category Game

P6 Project

Group SW601F14
Software
Department of Computer Science
Aalborg University

Spring 2014



AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg East
http://www.cs.aau.dk/en

Title:
The Category Game

Project period:
P6, Spring 2014

Project group:
SW601F14

Group members:
Aleksander Sørensen Nilsson
Kasper Plejdrup
Mette Thomsen Pedersen
Niels Brøndum Pedersen

Supervisor:
Katja Hose

Total number of pages:
81

Synopsis:

Autism can be a real problem both for those who have it and those who interact with them.
For our 6th semester project we will further develop a tool, from last semester, which purpose is to ease the day to day activities of people affected by Autism.
The purpose of this semester is to teach the groups team-work on a greater scale than the 6-7 person groups we are used to. Therefore we are going to work on a composite project, where each group is responsible for a part of it. These sub-projects will in the end, work together to make a collective solution that will help people with autism learn to communicate, as well as help their guardians manage their day to day activities both on a general plan and in more specific situations.
For our part we handle the specification requirement to help guide the groups toward a common goal. We will also develop on the application called "Train"(later renamed "Kategorispillet"), which is a game the will help people with Autism learn to associate pictograms with their respective category. Beside this the group handled everything regarding the customers for the entire multi-project.

Resumé

This year we worked on a part of GIRAF(the Graphical Interface Regarding Autistic Folk) which was started in 2011 by Ulrik Mathias Nyman. The whole multi-project this year was about developing tools to people with autism and their guardians to help them in their daily lives. For our part of the project we made the specification requirement and we also worked on the application "Kategorispillet" (The Category Game). We are developing the application for Android tablets and it has been programmed in Java using Android Studio. "Kategorispillet" is a game that helps people with autism associate pictograms with their respective categories. This is done by placing the pictograms on train carts. The train will then drive to a station which has a category attached to it and here the user will have to drag the correct pictograms onto the station. The train will not be able to depart before all the correct pictograms have been dragged off. When there are no more pictograms left the train will drive to the depot and the game will be finished. The report is mostly written towards people with basic technical knowledge, and the people taking over the development of the application after us. The report is split into four sprints contained in separate chapters.

Prolog

This report is written by a group of students from the Department of Computer Science at Aalborg University (AAU) as a 6th semester software multi-project. The group members are Aleksander S. Nilsson, Kasper Plejdrup, Mette T. Pedersen and Niels B. Pedersen. This report documents and describes the process working in a multi-project and what the group worked on.

The references in the report will be in the format [Example, year] with a corresponding entry in the bibliography in the back of the report just before the appendix. Websites with no specific year will be on the format [Example]. Figures and tables will be referred to in this manner: Table 3.5, where the first number is the chapter and the second number is the index of the figure or table in that chapter.

The DVD included on the last page of the report contains a PDF this report, the code for the project together with an apk of the application we have worked on, see appendix D.

We would like to thank our supervisor Katja Hose for her supervision and guidance through this multi-project as well as thank our customers for finding time to work together with us.

Aalborg, May 27, 2014

Aleksander Sørensen Nilsson

Kasper Plejdrup

Mette Thomsen Pedersen

Niels Brøndum Pedersen

Contents

1 Contextual Analysis

Last year the multi-projects main focus was developing for children with autism, this year the focus will be extended to people with autism, not only children. Different applications to help people with autism already exists, but they are usually developed by different developers and thus there are few to no solutions that combine every tool they need. This multi-project aims to solve this by having many applications that each will help the people with autism in different ways.

Each section in this chapter will describe one of the sprints' focus for our group, the impact it had on the multi-project and what we learned from the multi-project. The section will also be used to describe the setting of the multi-project and how the other projects influenced our work.

1.1 Structure of the Multi-Project

This year we were 16 groups with four people in each, except two groups containing only two people, working on the same project. Two of the groups were working with iOS while the rest were working on the project from last year to Android. Since we are used to work together only in a group of 6-7 people, this was a completely new challenge to try to work together between different groups.

In the beginning the different groups seemed mostly to mind their own project, and taking every problem up on the status meetings, which were held once every week. Later in the process we thought that these status meetings were taking too long, approximately 60 minutes, and they were shortened with the decision to only take up the topics concerning everyone. If a problem concerned multiple groups, they would afterwards meet and discuss the matter.

This seemed to change the work routine for most groups. Now if a group was having a problem with something concerning another group, one of the group members would simple go to the other group's room and discuss with them how to solve the problem. This way of working together really helped a lot, since most groups were happy to help get the problem solved quickly, and this ensured a group would not be stuck on a problem for a long time without any progress.

We also learned that 14 groups needing to agree on many things was hard work, when you are not used to collaborating on such a large scale. We experienced that talking immediately with other groups about problems sped up the progress of the project. We also learned that a strong leader with a good overview is important to get things done,

but also that delegating some of the responsibility out to different people is important, so a single person is not crushed under the responsible for everything.

In our semester we had different people responsible for all the different tools we used, our server, the project, the sprint reviews and the customers. This ensure a single person would not be overburdened with more work than he/she was willing to take.

At the end of each sprint we held a sprint end review where we invited our customers. At this sprint end review we presented what we had worked on, how the application look liked and what we would work on in the next sprint.

1.2 Application Guide

Most of the application has been programmed using Android studio and will be described in general, to give a better overview of the multi-project and the interaction between the different projects.

GIRAF is the application/launcher meant to contain the other projects in the multi-project and where the guardian can restrict which of the applications each citizen has access to.

Oasis's purpose is three fold: it is a database, a library and an administration application. The database is local on the tablet and used to store data that is accessible to the other applications in the multi-project. The library called "OasisLib" is used by the other applications to communicate with the local database. The administration application is for handling profiles that the guardian and citizen with autism have on the tablet.

RemoteDB is a remote database to store data for the other applications in the multi-project. This database is used to synchronize the local databases on each tablet with each other.

PictoSearch is a tool used by others applications to find and fetch pictograms from the local database into their application.

Cat is a tool for the guardian to make, see, edit and delete categories of pictograms that are used by other applications. This makes it easier to search through the database of pictograms. As of sprint III this year this was renamed to "Kategoriværktøjet".

GIRAF-Admin is a web page which was made this year to administrate profiles on the Internet thus allowing administration on a computer. This was made in "PHP".

Cars is one of two games made to "GIRAF". It is meant to help citizens with autism learn how to control their tone of voice by moving a car up and down a track with obstacles. The first idea for the game mechanic was to move the car based on the pitch of the voice. The idea was fine but was not implemented very well, so this semester the "Cars" game mechanic has been based on the volume of the voice instead. As of sprint III this year this was renamed to "Stemmespillet".

Train is the other of the two games made to "GIRAF". It was meant to help citizens with autism learn the relationship between pictograms in context to categories. As of sprint III this year this was renamed to "Kategorispillet".

Parrot is a tool made to read a sequence of pictograms for the user. The citizen makes these sentences by dragging the pictograms onto the bar displaying the sequence and the

application will read it as a sentence. This can help them communicate as well as help them learn to communicate on their own. As of sprint III this year this was renamed to "Piktooplæser".

Croc is a tool made to make new pictograms and record a sound that will be attached to it. As of sprint III this year this was renamed to "Piktotegner".

Tortoise is a tool to make a life story which is a way for the person with autism to describe their day, if they cannot do it themselves. This is done by making a series of pictograms that describes that day's activities. This year Tortoise have been split into two applications. An application for describing life stories and a week scheduler. As of sprint III this year these two were renamed to "Livshistorier" and "Skema" respectively.

Zebra is a tool to make, save and view sequences of pictograms, it was also possible to make choices in a sequence so a citizen with autism can choose between more than one pictogram. These are made as templates for the guardian to help tell the citizens with autism what they are suppose to do in a given situation. As of sprint III this year this was renamed to "Sekvens".

Wombat is a tool for the guardians to set a time limit on an activity for the citizen with autism. It is possible to use different views of the timer like an hourglass, progress bar or a clock. It has also been made possible to have the timer displayed in the other applications. As of sprint III this year this was renamed to "Tidstager".

1.3 Guardians and Citizens

Institutional guardians are taking care of the citizens, helping them, making their weekly schedule and teaching them different skills. Since there can be a big difference between the citizens and the institutional guardians, this difference had to be taken into account when developing. Therefore there are some applications or functions in an applications the citizens will not have access to while the institutional guardian have. An example can be that the institutional guardian have full access to the weekly schedule application, being able to create, edit and delete schedules, while the citizens only have access to the function that view the week schedule in the weekly schedule application and interact with it.

1.4 Dependencies for "Kategorispillet"

"Kategorispillet" have some dependencies on other projects that are in the multi-project, an illustration can be seen on figure 1.1. Three of them are implemented as sub-modules and they are named "GIRAF_Component", "pictogram-lib" and "OasisLib". "GIRAF_-Component" contains all the graphical user interface components. "pictogram-lib" is an library for handling Pictograms in the multi-project. "OasisLib" is a library that holds the information on how the profiles are saved in the multi-project.

The applications "GIRAF", "Picto Search" and "Kategoriværktøjet" need to be installed on the table together with "Kategorispillet" for our application to function correctly. "Kategorispillet" need "GIRAF" for getting the users to login and thereby getting their profile information. "Picto Search" is used for choosing Pictograms that are used for stations. "Kategoriværktøjet" is tool to make the categories for pictograms that is used in "Kategorispillet" to get the stations color from the category.

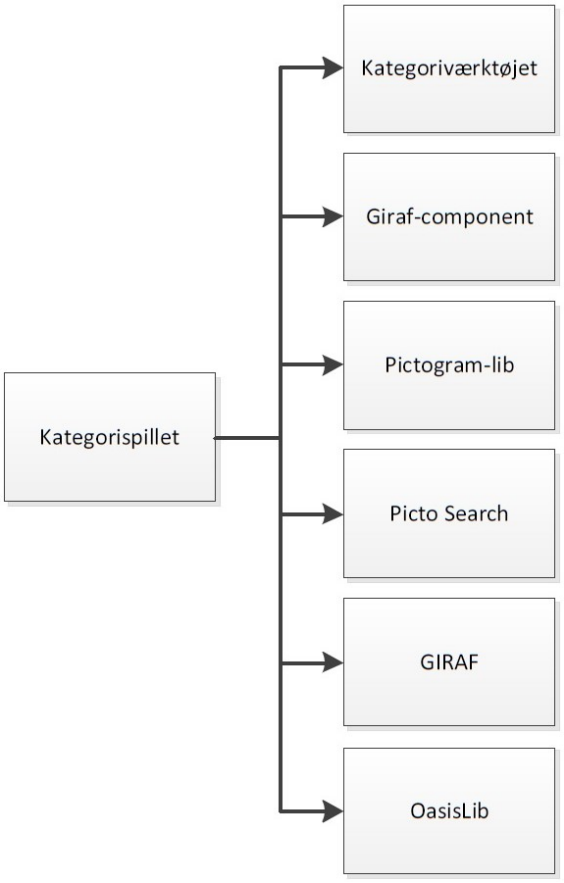


Figure 1.1: Here it can be seen what "Kategorispillet" depends on.

1.5 Multi-project in Sprint I

Sprint I's main focus was for all the project groups to get an overview of the project and figure out what should be further developed. To give the different groups a better idea of what the customers wanted, we worked on the specification requirements to describe the customers' expectations of the product.

1.5.1 Autism and its Impacts

The multi-projects target-group was citizens with autism and their guardians, so we had to learn what it meant to be a person with autism and what impact it had on their life.

Autism hinders the mental growth of the person thus they have trouble developing their comprehensive capabilities as well as their social skills. People with autism can get frustrated if they experience something they do not comprehend, which means they do not like unexpected events or changes. Therefore their daily routine are scheduled in a structured fashion to minimize the amount of stress they experience.

People with autism can also have underdeveloped linguistic skills and they train these skills with their guardian with the help of pictograms. This means there are a lot of pictograms that the guardians will have to create, administrate, maintain and use in their daily work. A picture of the guardians pictogram workspace can be seen on figure 1.2.

Figure 1.2: The thousands of pictograms the guardians have to administrate every day.

The problems with creating, administrating, maintaining and using the pictograms can be made easier by a digital solution and therefore encourage us to develop a system that includes the needed functionality to solve this problem. As the severity of autism are very different from person to person we have to reach a broad audience, thus we have to keep the design and functionality relatively simple and flexible.

Different solutions already exist, like the application "JABtalk" [LLC], that provide the same functionality as the application handling sequences (Sekvens) and the communication application (Piktooplæser), but none of them collects the functionality into one system. Therefore the stakeholders are very interested in this multi-project system as a free alternative solution, that they can use instead of the existing ones.

1.6 Multi-Project in Sprint II

In sprint II we began to look into the code of the train project to get a better understanding of how it worked. As this had little context to the multi-project, therefore this section will focus on a more relevant subject, the role as a contact person.

1.6.1 Contact Person as a Role

Group member Mette Thomsen Pedersen volunteered to handle all communication between the customers and the different groups in the multi-project, and therefore act as contact person for the multi-project. The role as contact person will persist through the

rest of the sprints and the same person will keep the role throughout the time, to make sure there are consistency in the communication with the customers.

It was the wish from the customers to have one contact person so that the mails regarding the multi-project would be combined. This was done to prevent the customers from being flooded with the same questions from different groups. The contact person role also included scheduling and having meetings between the groups and the customers, handling all relevant information, scheduling as well as being the minute taker for the usability tests. The purpose of the meeting was for the groups to be able to discuss with the customers in which direction their individual part of the multi-project should be heading. When some groups had decided that they wanted a meeting with the customers, the contact person would write to the appropriate customers and try to find a date for the meeting. When a date was agreed upon, an agenda was made for the meeting. The contact person attended the meetings to make sure the different groups kept to the schedule and wrote a summary of the meeting. The contact person was also in charge of getting all the information out to the customers regarding sprint ends or other important information.

1.6.2 Experience of Being the Contact Person

In sprint II, two meetings were held and at each meeting two customers and four project groups were attending. Each group presented their work and asked the customers for their opinion. Between each group a short break were held. In sprint IV two days of usability tests were held with two customers. At the test the contact persons responsibility was the same as at the meetings.

The role as contact person was educational, it taught the contact person how to handle meetings with customers and handling communication between 14 groups and eight different customers. It also taught her how to handle scheduling and following up on every conversation. It taught her how important it is to schedule ahead of time, and that deadlines are important to make and upheld. Another thing was that through it seemed like easy work at the start it consumed more time than the contact person expected having to handle emails and all the work. Though the work was time consuming it was exciting and educational.

1.7 Multi-Project in Sprint III

In sprint III we focused on developing the "Kategorispillet" application. During the sprint we encountered some minor issues regarding using other groups modules. Therefore we contacted the relevant groups to help us with these things. One of the groups we contacted was the GUI (Graphical User Interface) group to get help implementing the "GComponents" in the window with the settings of "Kategorispillet". We had trouble setting the background color and it turned out the function we were using for generating the backgroundcolor was being developed by the GUI group this semester and not completely done at the time. So they help us identifying this problem and provided us with an alternative that we could use.

Another group we contacted was the "GIRAF" group as we could not start the "GIRAF" application on the tablet we had just acquired at the start of this sprint. The problem was that our tablet lacked a camera on the back of the tablet which "GIRAF", the launcher, needs to be able to scan the QR-codes we use to log in. Our tablet only had a front camera. This problem caused "GIRAF" to prompt a window explaining the problem and close itself. The "GIRAF" group looked into our problem and made a fast

hot fix for us so we could continue working. Later they released this new version of "GIRAF" to every group to make sure no one would struggle with same problem that we had experienced.

1.8 Multi-Project in Sprint IV

In this sprint we focused on finishing implemented features. To ensure the application worked as the customer expected, we performed a series of usability tests to evaluate the application's features.

This sprint we learned that it is important to agree upon a deadline for changes that potentially can affect the whole system. When the remote database got integrated in the launcher, it caused a lot of issues for everybody, mere hours before the applications should be delivered to the customers. This caused a lot of unnecessary stress which could have been avoided by discussing and planning the integration better so all groups were prepared for it.

Planning testing with customers was a difficult matter, because the customers seldom had the time in their work day for testing and therefore required planning ahead to actual getting it to fit with their time schedule.

Sprint I 2

This multi-project was a continuation of last semester's project, thus we had a lot of data to work with and it had allowed the customers time to try the product on their own. With this in mind we have been collaborating with group SW604F14 in interviewing our customers so we could get a better idea of their thoughts on the product in its current state and their expectations for the product. Together with SW604F14, we formulated a series of interview questions which were used when we conducted the interviews. The collected knowledge was apportioned to relevant topics and together with last years user stories we developed the specification requirements. We and our collaboration group evaluated the requirements together with the customers and performed the necessary changes to reach an agreeable state. The result of the specification requirements should guide the multi-project and the different groups to make a product in accordance to the customers wishes and needs. The first sprint started on the 24th of February 2014 and ended on the 26th of March 2014. It was necessary to continue with the specification requirements in sprint II since extra time was needed to have a chance to meet with the customers and get the specification requirements approved.

2.1 Preparations for Customer Dialog

To gain more insight into the problem domain, we decided to engage in early collaboration with the customers. This collaboration was conducted as a dialog where we had prepared a series of questions, which was used to guide the dialog with the customers and to ensure that specific areas of the problem domain were discussed. The areas that we wished to research were: which tools the customers uses, which features they find essential for the used tools, missing functionality and at which stage of the development of the individuals with autisms our tools would help the most. See appendix A for the translated version of the interview questions.

The information we attained from the customers was used to design a draft of the specification requirements. We used the specification requirements and the user stories from last semester to write new user stories for the project. These user stories were used to describe the customers needs and wishes formally to the other projects groups in the multi-project and can be seen in appendix C.

2.1.1 Interview Form

We have looked at different methods to conduct our interviews to ensure that the needed information would come to light. These different methods are: Fully structured-, semi-structured and unstructured interviews [Benyon, 2010].

Fully structured interviews uses questions that are developed before the interview and are followed word for word. This format is good for interviewing a large number of people, but limits the variation of responses. The limitations obstructs possible new insights in the specific field because it focus on an answering to the interview questions and thereby limited by the preparatory work done by the interviewer. Semi-structured interviews are a more used format were the interviewer writes some general questions for the interview and if, through the interview, a new topic arise it can be more thoroughly debated and explored together with the customer. Semi-structured is however more demanding of the interviewer because they have to be more aware to not get side tracked, but still follow up on relevant topics, compared to a fully structured interview. The additional information gained from semi-structured is usually worth the effort if it is still relevant to the subject. Unstructured interview is used when the preconceptions of the interviewer is minimal and little to no background information is known before the interview. Therefore no questions are made before the interview but the general subject is found and used to guide the interview in the wanted direction [Benyon, 2010].

We determined that the interview should be semi-structured because this format is good for starting a dialog with the customers but still allow us to select which fields we would like to hear more about. The dialog can help with answers on questions that came up in the interview that was not thought of when the questions were made. The dialog will also help us discover needs of functionality which the customers might have trouble expressing or rather thought of them as trivial, something we should know of but might not have because of lacking insight in the particular field.

A consequence of the interview form is that the discussion with the customers were fluid and coherent. This characterized the recording of the interview which resulted in the summary having no structure. The unstructured summary meant that the results needed to be sorted in some way to gain a overview.

2.1.2 Interview Questions

For the first interview we wrote a questionnaire which was used when conducting the semi-structured interviews. We and the collaboration group drafted an early version of the questionnaire. These drafts were combined and we discussed whether the combined questionnaire were sufficient for gathering the necessary information. The questions we formulated were split into different categories to get a better structure on the interview:

- Customers work flow.
  We wanted to know about their general work flow, such as what tools and techniques they utilize. This was to get a good baseline for the products requirements.
- Experience with the product.
  We wanted to know which of the apps from the product they have tried, since that would give a good basis to ask further questions. Depending on their experience with the product we would ask for suggestions on improvements.

- General improvements across the board.
  We wanted to ask them about different features that might be implemented to make it more user friendly for the citizens as well as make the product easier for the customers to use.

- Customers expectations of the product.
  These questions were generally about the customers expectations for this semester, and what they would prefer got developed. This would tell us what to focus on this semester.

The full list of the translated questions can been seen in appendix A.

For the second interview we mainly used the developed requirements and discussed it together with the customers. Based on discussions with the customers we performed corrections to the requirements so they better fit the customers wishes.

2.2 Data Processing

To get a better understanding of the customers needs, we processed the data we gathered from the interviews and presented the result in a more formal way. With our collaboration group we drafted the first specification requirements which contained all the key points we discussed with the customers. When we were content with the draft, we presented it to the multi-project groups for feedback. With this feedback we made corrections to the draft and expanded upon the description making it more explanatory. The modified draft was then presented to SW8(The software students one year above us) for a more advanced evaluation as they had a specific course for this. With the feedback from SW8 we made some additional modifications before we presented our work for the customers to approve. We asked the customers if they could spot any obvious shortcomings and noted their feedback. Along with our collaboration group we made the final modifications to the draft based on our collected feedback. The final edition of the specification requirements can be seen in appendix B.

2.3 Requirements Analysis

In this section we analyze the information gathered through interviews with the product owners. We in group SW601F14 and group SW604F14 have written this section in collaboration and therefore it will appear in both of our reports with slid differences. The structure of this section is inspired by the structure of the article "Software Requirements Specification" [Geagea et al., 2010].

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.3.3.1 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 needs 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.

2.3.3.2 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 can not 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.

2.3.3.3 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.

2.3.4 Specific Requirements

This section will describe all the 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.

2.3.4.1 Requirement Example

The following section is an excerpt of our full specification requirement, the full version can be seen in appendix B. The user stories that are used in this example can all be found in appendix C. 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)

2.4 Reflection

In this section we reflect on the overall multi-project and the process of written a chapter together with another group.

2.4.1 Group Collaboration

In this sprint we have collaborated with group SW604F14 in order to produce the specification requirements which the customers could use to influence the project and its different parts. We think that the collaboration with the other group went as expected, both group contributed to the course of creating the specification requirements.

Before the interview both group formed a series of questions that were found relevant in order to collect information regarding the overall project. Together with the other group we combined the questions into one document that was used when conducting the interviews. The group of customers were split between each group to be more efficient. We think this was the most constructive method for collecting information because both group got the chance to talk with the customers and gained experience from the interview practice.

After the interview each group processed the information which we then combined into a draft of the specification requirements. Because of lecture arrangements we had to write in turns on the specification document. This way of working introduced some communication problems because the group working could not talk with the group having lectures

without disturbing the lecture session or exercises which slowed down the productivity. Therefore as an alternative we tried to allocate time after normal work hours where both groups had the possibility to work. We found the simultaneous working flow to be more productive than the staggered working flow.

2.4.2 The Multi-Project

In the multi-project we have experienced communication difficulties regarding the expectation of what the specification requirements should contain and the level of detail. We had the idea that the specification requirements should act as general guidelines for other projects which they then should use to develop their own more specific requirements. This was not the case for the other project groups as they expected a more detailed specification of the whole system. This difference in expectation led to unnecessary conflict between us and other project groups which could have been avoided. We think time and effort was wasted on these conflicts which could have been avoided if we and the other groups had discussed the task and come to collective agreement on what was expected from the start.

We also experienced other group exceeding deadlines for their input on the specification requirements. After the deadline a lot of changes were submitted to us which forced the time schedule to be delayed. We were partly to blame because the developed specification requirement was not specific enough to describe how the sub-systems should interact and even certain sub-systems were not described at all, but we also think that the other groups could have helped with this issue by providing their feedback within the deadline so we had more time to fix it. We think this could have been avoided by better use of the calender and communication between the groups.

Sprint II 3

It was determined in the group that working with "Kategorispillet" was interesting and therefore it was chosen as the project for sprint II which started on the 27th of March 2014 and ended on the 14th of April 2014. In sprint II the time was used on finishing the specification requirements from sprint I, getting an insight into the code for "Kategorispillet" and getting it up-to-date with the new local database scheme and removing dependency for profiles from "Timer-lib" that was deemed unnecessary and moving profile handling to "OasisLib".

3.1 Finalizing the Specification Requirements

We got a lot of feedback on our specification requirements in the end of sprint I which we did not have time to include. So we spent part of the first week of sprint II on incorporating all the feedback we got from the customers and the different groups in the multi-project into the specification requirements. When we were done correcting the specification requirements, we put it on the multi-project's wiki making it available to the other groups and thus finished our work on it which allowed us to begin work on our new topic for sprint II.

3.2 Kategorispillet

The idea behind the game is to teach a citizen with autism that there can be a connection between different pictograms, e.g. that both the "apple" and "banana" pictograms can be grouped in the category "fruits".

First a guardian sets up a game by creating a number of stations. The guardian then sets the categories together with the pictograms that should be placed in the given category for each of the stations. An example could be the pictograms "apple" and "banana" should be in the "fruit" category while "bike" and "car" could fit in the "transport" category. When the game is started, the train starts at the main station where all the selected pictograms are waiting to be dragged onto the train carts. When all the pictograms are on the train, the whistle can be pressed which will make the train take off. The train then drives to the next station and stops there.

This station has a category and the citizen needs to drag off the pictograms belonging to the stations category. After all the correct pictograms have been dragged off, the whistle will be clickable again and the citizen will be able to press it to continue the train's journey.

When all the pictograms have been dropped off and the whistle is pressed the train will drive to the train depot and the game ends. This can be seen on figure 3.1.

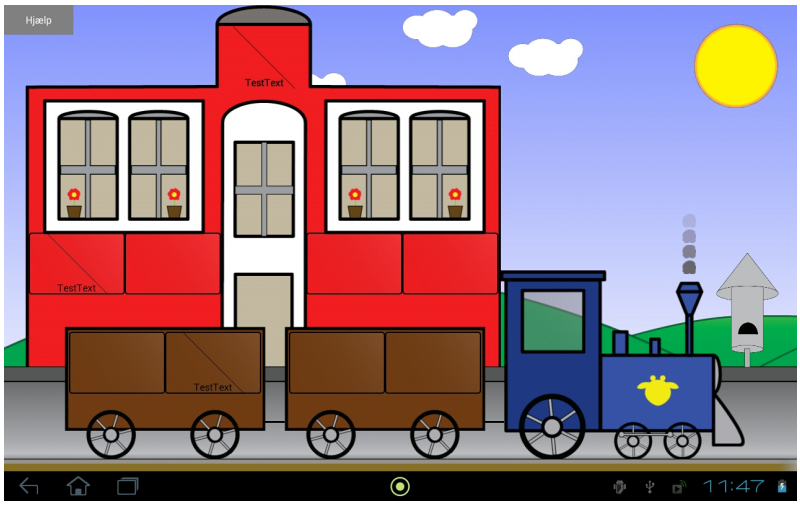


Figure 3.1: A picture of the train at a station, with a couple of test pictograms.

We are working with people with autism and therefore it have been decided there should not be a time limit which the player have to finish within to complete the game or a score point system. The reason behind this is that these mechanics can expose the citizens to unnecessary stress and therefore is actually not a wanted feature by the customers.

3.3 Initiation of the Development

Before we could begin development, we had to connect to the "Train" repository to actually access to the code. The whole multi-project uses GIT as source control which meant we had to learn how to work with this new tool.

GIT is a tool for version control that emphasis optimistic version control because it allows multiple developers to work on the same file concurrently and tries to merge each user's changes together. Conflict-handling is done through a merge manager where the user have to use the tool to merge the parts that causes the conflict. Another very useful feature that GIT provides is branching which, if used correctly, ensures that there is always a "master" version of the application that actually runs while under development. The developer can, when developing, create a branch from the master branch and use the newly created branch independently of the master branch to develop and bug fix. When working on a branch it is still possible to get the newest version of the master branch by pulling the changes and merge them into the branch that is being worked on. When the newly implemented features are finished being developed, the branch is merged into the master branch. This is the general work flow we have learned and follows by working with GIT as a version control tool.

After setting up GIT and learning how to work with it, we fetched the code from the repository. Before developing on the code we tried to pull all new changes to the "Kategorispillet" project and its submodules and get it to run. After the new changes we experienced errors and crashes that hindered "Kategorispillet" in running correctly.

3.4 Development

In this section we look more into the problems mentioned in section 3.3 and how we corrected them.

3.4.1 The Analysis

The database had been changed which meant we got errors regarding profile selection, because some of the functions no longer got the correct input. We noticed that part of the problem with our code stemmed from functions in "Timer-lib", a library from "Tidstager" our code depended on. The initial solution which depended on "Timer-lib" used a class which they called "Guardian". "Guardian" implemented "Oasis-lib" functions to fetch guardian profile information and store it in an object of the "Guardian" class. A "Guardian" object contained a list of citizen profiles which were associated with the specific guardian, that could be used to select the wanted profile. This setup suited the design where each sub-application would handle the profile selection themselves. The "Guardian" object were used to fill the "Data" object with information about the citizen and guardians IDs and the applications background color. The "Data" object were then used through out the program to access this stored data.

The changes to the database scheme also introduced problems with types across the application. We had trouble with data not being parsed correctly between activities which meant that the defined station could not be created correctly in the game and caused a system shutdown.

3.4.2 The Development

"Timer-lib" originated from another project, which another group handled, so we decided not to try and fix the errors in "Timer-lib". Instead we decided to use "Oasis-lib" directly instead of going through "Timer-lib". The reasoning for this decision was that "Oasis-lib" already provided functions for fetching profiles and other data which were meant to be used for interaction with the local database. Since the launcher application have taken the responsibility of handling the profile selection it is no longer necessary for this application to do it.

To directly interact with the local database we make use of a class called "Helper" from "OasisLib", which was designed for this exact purpose. The function "getProfileById" finds and returns the profile identified by the supplied profile ID, given as a parameter in the function call. We decided to redesign the "Data" class so it contains profiles for both the guardian and the citizen instead of just for the guardian. The "Data" class now also have a constructor to set the values when declaring an object of it. The class can be seen on listing 3.1.

```
1 public final class Data {
2
3   public int appBackgroundColor;
4     public Profile guardianProfile;
5     public Profile childProfile;
6
7     public Data(int guardianID, long childID, int backgroundColor,
          Context context){
8         Helper localDataFetcher = null;
9         try {
10            localDataFetcher = new Helper(context);
11        } catch (Exception e) {
12            //TODO:lav en ordenlig exception
13        }
14
15        appBackgroundColor = backgroundColor;
16
17        guardianProfile = localDataFetcher.profilesHelper.
              getProfileById(guardianID);
18        childProfile = localDataFetcher.profilesHelper.
              getProfileById((int)childID);
19    }
20 }
```

Listing 3.1: The "Data" class with its constructor.

Android features a data transportation method that uses parcels, a form of data packaging, that efficiently transports data between activities [Android, e]. We discovered the origins of the problem in the data package (the parcel) containing the game configuration. Some of the data was written like it was still of the type "long" while the same data later was read as type "integer". This resulted in a null pointer exceptions that caused the crash but through debugging we discovered the cause and patched it so the data were transported correctly. We also changed parameters, return types and variable of the type long to integers so they followed the new database scheme.

3.5 Reflection

In this section we will reflect on the collaboration with the other groups. We will also reflect on the work that we have done in this sprint.

3.5.1 Collaboration

In this sprint there was a proposal for collaborating with the group that was working on the "PictoSearch" application. They got an error when trying to load a pictogram from "PictoSearch" through "Kategorispillet" and therefore proposed to collaborate in finding and fixing this problem. While researching the problem we experienced other building and runtime errors occurring which hindered "Kategorispillet". Through the process of understanding the code and working with these errors, we came to the conclusion that the error originated from the problem that the "Kategorispillet" project were not adapted to the new database scheme and therefore there were no foundation for the proposed collaboration.

The remaining time of sprint II went by trying to adapt "Kategorispillet" to the new database scheme and fix the errors that occurred after the new changes. Therefore we did not find any time to collaborate with any of the other groups in sprint II.

3.5.2 The Multi-Project

We used the first week of the sprint to finish the last requests we got from the groups about the specification requirement from sprint I. The next week we spent most of our time on a mini-project as it took longer than our lecturer anticipated, it was not only our group that struggle with the assignment, from what we gathered most of the groups on the semester had trouble with it and had to spent additional resources on it. After the mini-project was finished we spent the rest of the sprint on understanding and re-factored the code to work with the newly implemented database that another group worked on in the first sprint. The code would not compile and we had some difficulties finding our way around the different files, this lead to a lot of frustration as we could not deduce the problem. During this process we also made suggestions to what could be worked on regarding Train in the next sprint. We did not interact so much with the other groups this sprint, but the interactions we had were a lot more pleasant compared to the last sprint.

3.6 Further Development

As mentioned in section 3.5 we had problems getting the code to compile and work properly. Thus we spend the time re-factoring and trying to understand the code. For the next sprint we have suggest some prominent features, that we think would be suitable start implementing. These features can been on the following list that will be worked on in sprint III:

- The color-theme of the application needs to be the same as "GIRAF" and use the GUI components which was made by the GUI group.
- Make it possible to adjust how long the train has to drive before reaching a new station.
- Make game instructions for the application.
- Make it possible to save a game setup.
- The stations' colors should adapt to their assigned category.

Sprint III 4

We continued on "Kategorispillet" ("The Category Game") which in this sprint had its name changed from "Train" to its new name because of the requirement stating that all applications should be named with a more describing Danish names, see number 6c section B.3 in appendix B. This renaming of all relevant applications was done during the last sprint end review. This sprint started the 25th of April and ended on the 7th of May. In sprint III the time was used to change the self-made graphical user interface for train from last year into using "GComponents" which is used in the whole multi-project, making it possible to have different distance between stations in a game and starting on making it possible to chance the color of stations based on their category.

4.1 Work Based on the Requirements

We have looked at the requirements about "Kategorispillet" from appendix B and feedback from the customers, and composed a specific list of functionalities to give us a better overview of which features that are missing and should be implemented.

1. Ensure that the game data is saved correctly to the database.
2. Indicate that an input has been accepted and is currently been worked on, for instance by displaying a 'pending' icon.
3. Make it possible for a pictogram to play its associated sound when clicked while the game is running.
4. The color-theme of the application needs to be the same as "GIRAF" and use the GUI components which was made by the GUI group.
5. Make it possible to adjust how long the train has to drive before reaching a new station.
6. Use the profile picture of the citizen profile in the cab of the train.
7. Make game instructions for the application.
8. Make it possible to name the carts.
9. Make an option to change the color of the train.
10. The stations' colors should adapt to their assigned category.

11. Make a 'close' button that leads to the launcher GIRAF.

Because the list includes all the requirements we have decided to extract the most important features that should be implemented first. The features we deemed important to implement were number 4, 5, 7 and 10 in listing 4.1. We have chosen number 4 because our application should follow the general theme set for the overall multi-project to make it a consistent experience for the customers to use the applications. Number 5 and 10 were chosen to make "Kategorispillet" more flexible for the customers to use. Feature number 5 provides the guardians with a tool to help train the citizens patience, which the customers requested. Feature number 10 also help citizens identify a category based on the category's color. Feature number 7 was chosen to make it easier both for the guardian and the citizen to learn the mechanics of the game. We chose these features because they provided some features that would improve the overall user experience.

4.2 Development

In sprint III the three major features we have been working on was, implementing distance between stations, "GComponents" and the color of the stations. Each feature will be described in detail in the following sections. As seen on figure 4.1 we have a overview of the segment of the code we have deem relevant to our work. An arrow going from one class to the other, means that the class the arrow goes to depends on the class it comes from. We have selected the relevant functions and attributes for our work.

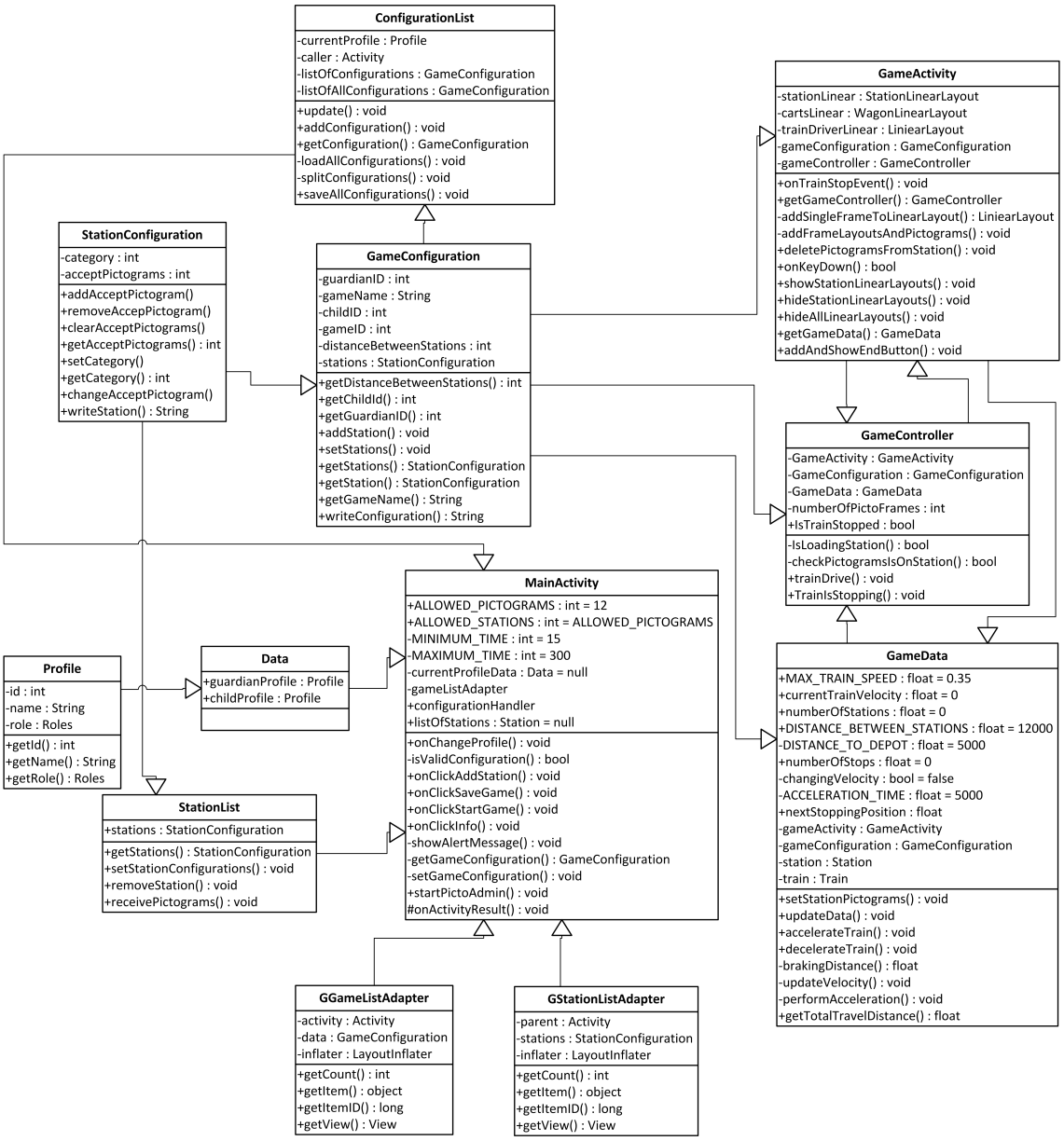


Figure 4.1: Overview of the dependencies between classes.

4.2.1 Train Travel Distance

As can be seen in the specification requirement number 2 in appendix B section B.14, the customers want to be able to change the travel time for the train between stations.

4.2.1.1 Analysis

When a guardian makes a new game they start in the main menu, the "MainActivity" class, where they can edit games. When they add stations with pictograms to the specific game, all the information is stored in an object of the class "GameConfiguration". This object will be parsed to the game part when they start the game and stored in the class called "GameData". This information is then used to set up the game layout and the total distance is calculated to determine the length of the course, so the background can be rendered. The first station is set on the same position as the train's start position and each time a new station is added, the "x" value of the station before it, is increased by

a static value. This value represents an amount of pixels and was preset by the precious group to ensure that the distance between stations is the same each time.

4.2.1.2 The Development

We decided to add an "Edit Text" box to the main menu to allow guardians to adjust the distance between stations. To accommodate this change the "GameConfiguration" class has gotten an extra variable called "distanceBetweenStations" to hold this new value as well as a way to parse it along with the other data. To be able to receive and save the new value the constructor of "GameData" needed to be changed. When the "GameData" constructor is called it will get the variable "distanceBetweenStations" from "GameConfiguration" and overwrite the preset value. We did not change the distance between the last station and the depot so the citizen should not wait on the reward for having completed the game.

The number written in the "Edit Text" box is the approximate amount of seconds the train will use to move from one station to the next. We chose to use seconds instead of pixels here because it would be easier for the guardians to understand. When the "Start Game" button or the save game button is pressed, a function called "isValidConfiguration" from the "MainActivity" class is be called to see if the configuration is acceptable. For the configuration to valid, there have to be a value and that value have to be between 15 seconds and 300 seconds. 15 seconds was the shortest acceptable time because if it got any shorter the stations would overlap, and we reasoned that waiting 5 minutes between stations would be enough. If the value is accepted, the value of the box is saved in the variable "distanceBetweenStations" and is first calculated to the corresponding amount of pixels when used in the game. This variable is then used to set the distance between the start station and all stations afterwards except between the last station and the depot.

4.2.2 GUI

We spent time redesigning the GUI and implementing "GComponents", which is GUI components designed for this project, in the "Kategorispillet" to achieve unity with the other applications in this project.

4.2.2.1 Analysis

A general problem was, that each group designed their layout differently across the applications, for instance different visuals for buttons with the same functionality. This made the customers think there was no internal communication between the different applications. The problem was reflected in the questions the customers asked at the sprint review of sprint I and this was a problem because the applications should feel like there were a connection between them. As a potential solution we decided across the project-groups to have a group design a set of graphical components to use for the GUI. These "GComponents" were to be implemented across all applications that made use of a GUI. This decision was also reflected in the specification requirements, which says: "All applications must use the same GUI components", see specification number 2 section B.4 in appendix B. A project group were assigned to design and administrate these graphical components and make them available as a sub-module for the other project groups.

The idea behind this collective pool of graphical elements was to streamline all the applications into an overall similar design so the users would quickly get familiar with the

applications and feel a correlation between them. Therefore we have spent time in sprint III to follow up on the requirement of a collective GUI and implement "GComponents".

To load the background color of the application before implementing the "GComponents", the variable "background", which was a drawable that was loaded in the "onCreate" method in the "mainActivity" class. The "background" drawable could be found in the location "res/drawables". The background color was set by filling the loaded drawable with the desired color, specified in the "APPLICATIONBACKGROUND" variable. The drawable was then set as the background in the "main_activity" layout. This can be seen in code snippet 4.1.

```
1 super.setContentView(R.layout.activity_main);
2 :
3 Drawable backgroundDrawable = getResources().getDrawable(R.drawable
      .background);
4 backgroundDrawable.setColorFilter(APPLICATIONBACKGROUND, PorterDuff
      .Mode.OVERLAY);
5 super.findViewById(R.id.mainLayout).setBackgroundDrawable(
      backgroundDrawable);
```

Listing 4.1: A code snippet of how the background was set before.

Another problem was the way the lists in the main activity's view was constructed. A view takes care of drawing the components declared in its layout and handling the events of the respective components [Android, d]. The problem was that a class had been constructed to both handle the data and contain the methods for updating and inserting data into the lists displayed in the view. This construction made the code complicated and introduced issues with the layout being tightly connected to the class used to store configuration data for the game.

4.2.2.2 The Development

Before we could start implementing the GUI components we spent some time to include the library with all the elements we had to use throughout the application. Once the components had been included we used the rest of the sprint on remaking the list setup so it could make use of the new components and changing the background to use the correct color theme which was also provided by "GComponents".

To make the library available to our project we first had to attach the "GIRAF_Components" as a sub-module to the project. This process required changes to the Gradle build file and the Gradle settings file in the project. This was done in order to make Gradle use the newly added "GIRAF_Components" sub-module when building the project. Gradle is a tool for building your projects that the Android studio uses [Grandleware].

With the implementation of the "GComponents" we changed the way we set the main view of the activity. Instead of setting the view directly from the layout file, we inflate the layout into a view and on that view, we set the background color to the color stored in variable "APPLICATIONBACKGROUND" through the method "setBackgroundColor". The new code replacing the old we see in code snippet 4.1 can be seen in code snippet 4.2.

```
1 GComponent.SetBaseColor(APPLICATIONBACKGROUND);
2 LayoutInflater li = LayoutInflater.from(this);
3 View mainView = li.inflate(R.layout.activity_main,null);
4
5 //Set the background
6 mainView.setBackgroundColor(APPLICATIONBACKGROUND);
7 setContentView(mainView);
```

Listing 4.2: A code snippet of how the background is set.

To change the components in the view, we had to change the layout file which tells the system how the view should be constructed. To change the components of the layout we have to change the class of the specific components, to use the custom classes provided by the "GIRAF_Components" library instead of using the standard classes. "GComponents" takes care of setting the components background and text color, so we no longer need to explicitly express it in the layout file but let the "GComponent" handle it. This general procedure covers the changes of buttons, textviews and dividers.

The lists were a bit of a different matter to convert to "GComponent" because of the second problem mentioned in section 4.2.2.1 concerning the construction of lists in the main activity's view. The way each item on the list was constructed made it difficult to convert to the new components because each item in the list have a very high coupling with the layouts around them. This meant that changes to the outer layout would cascade all the way to inner layout and would require all the affected layouts and items to be changed as well. We started off by changing the "scrollview" to a "GList". The "scrollview" was populated with items using the custom class "GameLinearLayout" which contained methods for updating the list but also for storing or getting game configuration data. The guide produced by the GUI group suggested that we should use an adapter to populate the "GList" with items. This was supported by the general guidelines by Android for using listviews [Android, c]. An adapter is a class that handles the insertion of data into a list [Android, a]. Therefore with a new list, the proceeding class "GameLinearLayout" were refactored into two new classes "GGameListAdapter" and "ConfigurationList". The "GGameListAdapter" class is an adapter for the lists in the layout and the "ConfigurationList" class contains all the data the list should have.

The constructor of "GGameListAdapter" receives the data contained in an object of the type "gameConfiguration" and uses a "layoutinflater" to inflate the layout so it is ready to be populated with the data. The method "getView", as seen in code snippet 4.3, is responsible for creating the items that should be inserted into the list in the layout. An item in this list contains a "GTextView" and an "ImageView". Where the "GTextView" will contain the name of the station and the "ImageView" will contain the pictogram of the category selected for that station. The changed view is then returned.

```
1 public View getView(int position, View convertView, ViewGroup
      parent) {
2   View vi = convertView;
3
4     if(convertView == null){
5       vi = inflater.inflate(R.layout.game_list_item,null);
6     }
7
8     GTextView categoryView = (GTextView) vi.findViewById(R.id.
          configurationName);
9     categoryView.setText(data.get(position).getGameName());
10
11    ImageView categoryPic = (ImageView) vi.findViewById(R.id.
          categoryPic);
12    categoryPic.setImageResource(R.drawable.default_profile);
13
14    Bitmap bitmap = PictoFactory.INSTANCE.getPictogram(activity.
          getApplicationContext(),data.get(position).getStation(0).
          getCategory()).getImageData();
15    categoryPic.setImageBitmap(bitmap);
16
17    return vi;
18 }
```

Listing 4.3: A code snippet of the "getView" method in the "GGameAdapter".

The class "ConfigurationList" handles loading and saving "gameConfigurations" from a save file. A "gameConfiguration" is saved to two lists, one list for storing the configurations for a logged in user profiles, called "listOfConfiguration" and another list for storing all configurations, called "listOfAllConfiguration". "listOfAllConfiguration" is used to secure that none of the other user's configurations are lost when saving. The method "loadconfiguration" and "splitConfigurations" handles loading "gameconfigurations" from the lists. The method "saveAllConfigurations" is used for storing the configurations to the save file. These methods were located respectively in the "GameLinearLayout" and "MainAcitivity" classes before, but are now located in the new "ConfigurationList" class. We did this in order to collect all handling of the game configurations into one class and encase the game configuration data.

To setup the list we create an object of "ConfigurationList" which loads all configurations through its constructor. We then create an object of "GGameAdapter" with a reference to the "listOfConfiguration" in the "ConfigurationList" object. The created adapter is then used to as the adapter for our lists, this together with how we create a "GList" can be seen in code snippet 4.4. When data is inserted or removed from "listOfConfiguration" we simply have to call the "notifyDataSetChanged" method on the our adapter to refresh the actual view. We believe than this design is easier to comprehend compared the proceeding design and it follows the Android's standard.

```
1 GList saveConfigurationList = (GList)this.findViewById(R.id.
      savedConfig);
2
3 this.configurationHandler = new ConfigurationList(this, this.
      currentProfileData.childProfile);
4 gameListAdapter = new GGameListAdapter(this,this.
      configurationHandler.getGameconfiguration());
5 saveConfigurationList.setAdapter(gameListAdapter);
6 saveConfigurationList.setOnItemClickListener(new AdapterView.
      OnItemClickListener() {
7   @Override
8   public void onItemClick(AdapterView<?> parent, View view, int
        position, long id) {
9       MainActivity.this.setGameConfiguration((GameConfiguration)
            parent.getAdapter().getItem(position));
10    }
11 });
```

Listing 4.4: Code snippet illustrating how we set up a list in the code. On this code snippet you can also see how the listener is created in the new design.

With the new lists we also changed the way the event listeners are created. Instead of creating an event listener to every item on the list, we create an event listener to the list itself. The idea of the event listener is to take the game configuration stored in the pressed item and set its values throughout the main activity. This idea is similar to the old design but differs in the implementation, the old event listener can be seen on code snippet 4.5 while the new event listeners can be seen on code snippet 4.4.

```
1 gameListItem.setOnClickListener(new OnItemClickListener(
      gameConfiguration));
2 .
3 private class OnItemClickListener implements OnClickListener {
4   private GameConfiguration gameConfiguration;
5
6     public OnItemClickListener(GameConfiguration gameConfiguration)
          {
7       this.gameConfiguration = gameConfiguration;
8     }
9
10    @Override
11    public void onClick(View v) {
12      ((MainActivity) GameLinearLayout.this.getContext()).
            setGameConfiguration(gameConfiguration);
13    }
14 }
```

Listing 4.5: Code snippet of how the listener was made in the old design.

4.2.3 Station Color

The customers expressed a desire for the stations to have the color of the category attached to them as can be seen in the specification requirement 7 section B.14 in appendix B. They

wanted this feature so the citizen could easier associate the given category with that color, along with the different pictograms that belongs to it.

4.2.3.1 Analysis

"Kategorispillet" had already implemented the stations when we started working on it. Pictures of the stations had been drawn and then imported. The pictures were then added to a list so they could randomize the order in which the stations appeared in the game layout. Being a static picture meant that it would not be possible to dynamically change the color of the station. If we were to follow that design choice we would have to, for each type of station picture, draw a new picture in a different color for each category. This would not be very efficient as it would be necessary to manually update the code each time a new category color was introduced. In code snippet 4.6 it can be seen how the pictures of the stations are loaded into their respective containers. They are then added to a list which is then randomized and then drawn in the for loop where the coordinates are added.

```
1         //Load the textures
2         this.redTrainStation.loadTexture(super.gl, super.context, R
              .drawable.texture_red_train_station, Texture.AspectRatio
              .BitmapOneToOne);
3         this.yellowTrainStation.loadTexture(super.gl, super.context
              , R.drawable.texture_yellow_train_station, Texture.
              AspectRatio.BitmapOneToOne);
4         this.blueTrainStation.loadTexture(super.gl, super.context,
              R.drawable.texture_blue_train_station, Texture.
              AspectRatio.BitmapOneToOne);
5         this.platform.loadTexture(super.gl, super.context, R.
              drawable.texture_platform, Texture.AspectRatio.
              BitmapOneToOne);
6
7         //Add stations to list and randomise
8         ArrayList<StationContainer> stations = new ArrayList<
              StationContainer>();
9         stations.add(new StationContainer(redTrainStation, 364f +
              (640f - 588.64f) - 16f, 583f));
10        stations.add(new StationContainer(blueTrainStation, 364f +
              (640f - 588.64f) - 16f, 583f));
11        stations.add(new StationContainer(yellowTrainStation, 364f
              + (640f - 588.64f) - 16f, 583f));
12
13        //Collections.shuffle(stations);
14        LinkedList<StationContainer> stationsQueue = this.getQueue(
              stations);
15
16        //Add stations to the matrix in the randomised order
17        float xPosition = -364f; // first platform position
18
19        for (int i = 0; i < super.gameData.numberOfStations; i++) {
20            StationContainer nextStation = stationsQueue.pop();
21            stationsQueue.add(nextStation);
22            this.stationPlatformMatrix.addRenderableMatrixItem(
                  nextStation.station, new Coordinate(xPosition +
                  nextStation.xOffset, nextStation.yOffset, 0f));
23            this.categoryMatrix.addRenderableMatrixItem(new Square
                  (0.5f, 0.5f), new Coordinate(xPosition + nextStation
                  .xOffset, nextStation.yOffset, 0f), Color.White);
24            this.stationPlatformMatrix.addRenderableMatrixItem(this
                  .platform, new Coordinate(xPosition, 0f, 0f));
```

Listing 4.6: Code snippet illustrating how the stations were drawn in the game when we got the code.

4.2.3.2 The Development

We decided to turn the pictures of the stations into templates by removing the color of the walls from them, so they became transparent. On figure 4.2 and 4.3 you can see the old design and the new design side by side.

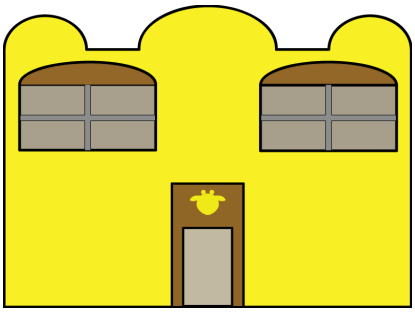


Figure 4.2: The previous station design.

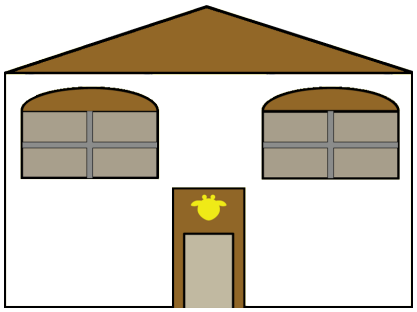


Figure 4.3: The new station design.

Being a template with no color allowed us to put a square behind it in the layout. The color of the square could change to the color of provided category. We also redesigned the roofs as they were not very practically design, as well as to get them to the same height, 132 pixels. The reason we made them the same height was that the square's dimensions were calculated by measuring the size of the picture of the station. So we had to subtract the roof's height from the height of the square or else the square would be too tall and would protrude from boundaries of the station template. Codesnippet 4.7 shows the new way we add the templates to their containers and also shows a new function that made to remove redundant code as it was just same thing written multiple times. The loop for adding the stations to the game layout still function the same as described in the analysis, see section 4.2.3.1. The function "LoadStation" can be seen on code snippet 4.8 which simply input the parameter into the formula needed to save the picture and returns it.

```
1    //Load the textures and add them to a list
2    ArrayList<StationContainer> stations = new ArrayList<
         StationContainer>();
3    stations.add(LoadStation(R.drawable.
         texture_template_train_station_first));
4    stations.add(LoadStation(R.drawable.
         texture_template_train_station_second));
5    stations.add(LoadStation(R.drawable.
         texture_template_train_station_third));
6    this.platform.loadTexture(super.gl, super.context, R.drawable.
         texture_platform, Texture.AspectRatio.BitmapOneToOne);
```

Listing 4.7: Code snippet illustrating the changes to the stations changing how they are drawn in the game.

```
1 private StationContainer LoadStation(int station){
2      Texture tempStation = new Texture(1.0f, 1.0f);
3      tempStation.loadTexture(super.gl,super.context, station,
           Texture.AspectRatio.BitmapOneToOne);
4      return new StationContainer(tempStation, 399.36f , 583f);
5 }
```

Listing 4.8: Code snippet illustrating the function used to make a station.

The square is not able to change color yet as it needs to receive a color from the "MainActivity" class, section 5.1.2 will describe our further development on this matter.

4.3 Reflection

In this sprint, most of the time was used on implementing new features for the "Kategorispillet". One of the new features was the distance between stations where we made some calculations, so that the users can input the distance between the stations as seconds instead of pixels. The calculation is needed because the game measures the distance between stations as a length of pixels instead of time. Another feature was changing our GUI in the settings window to use "GComponents" which is a requirement for all groups to implement in the multi-project. We had a problem with using the function for getting the background color, since at the time we implemented it, it was not completely done. We also encountered a problem with "GIRAF" on the tablet we got, for debugging in this sprint, since our tablet lacked a camera on the back. The problem was however easily fixed and after some time "GIRAF" pushed the hot-fix to their master branch so it also could be accessed from there too.

4.3.1 Collaboration

The group which had the responsibility for "PictoSearch" contacted us and reported a problem that occurred when running "Kategorispillet" on their developing device. We agreed to look into the problem, find the cause of it and a possible solution. The problem occurred when opening the "PictoSearch" activity inside the "Kategorispillet" and returned from it. The "PictoSearch" group reported that it was a null pointer exception. The group gave us their developer device for a time to use for finding the problem.

We started with trying to reproduce the problem at other developing devices to see if the problem in someway was connected to the device. We tested it on two other devices and experience that problem only occurred on their device. After this observation we looked into what caused the null pointer exception. Through the debugging we saw that it was the instance "pictogramReceiver" in the "mainActivity" class that caused the exception. Through further debugging we saw that it was upon returning to "Kategorispillet" activity that the "pictogramReveiver" instance turned null. Based on this our hypothesis was that the root of the problem was connected to the life cycle of applications in the Android system. A detailed diagram of an activity's life cycle can be seen on figure 4.4. From the diagram it can be seen that applications can be paused with the "onPause()" state which simply saves the current data so it can be easily restored with the "onResume()" state. We believe that our application on their tablet went to the "onStop()" state which meant our application got killed and had to be created again, which meant we lost the current data we were working on and there was no longer anything for "onActivityResult" from the "pictosearch" call to point at.

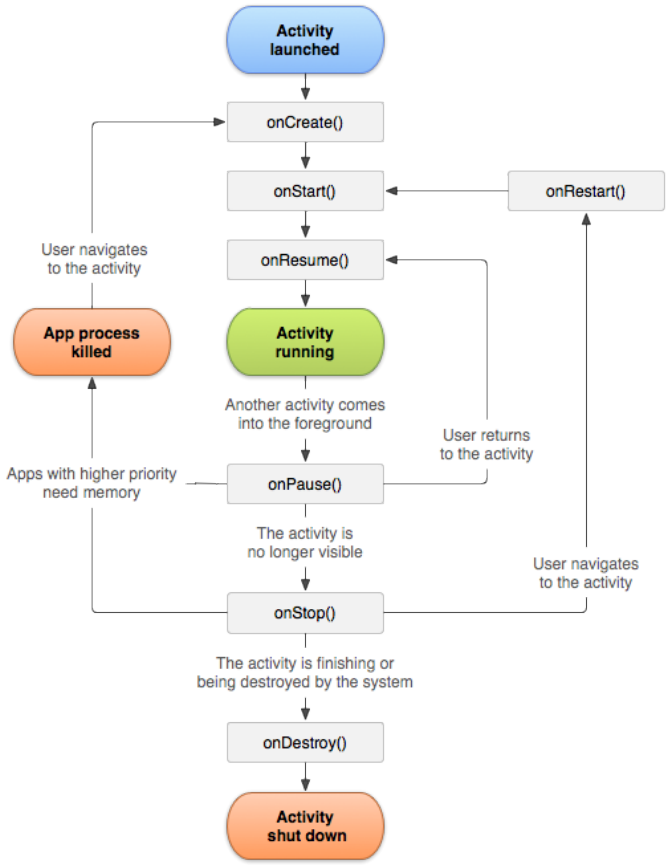


Figure 4.4: Detailed description of an activity's life cycle taken from the Android's developer page about activities [Android, b].

To test our hypothesis we created two variables, which we stored the same values in just before calling "PictoSearch". The difference was that one of the variables was saved in the "savedInstanceState" bundle through the method "OnSaveInstanceState" which is called when an application is stopped. In the "onRestoreInstanceState" method, which is called when an application starts, the previous stored variable is restored by using the value saved in the "savedInstanceState" bundle. With this setup and the two methods implemented we again tested the hypothesis and monitored the variables value. Before the "PictoSearch" call the values were the same, but upon returning the variable not stored in the "savedInstanceState" bundle had turned null while the other had kept its value.

Based on these result we concluded that the reason for the null pointer was caused by the device's system destroying our application just after calling "PictoSearch". A potential solution would be to write all the necessary information to the "savedInstanceState" bundle and used it to restore the application to its previous state. We spent a couple of days working with this problem and finding the suggested solution.

We only experienced this problem on a single device out of all the working devices that we tried and it could very well just be a setting placed on that device but since there were more critical issues to take care of we decided not to start implement the solution due lack of time. We informed the group working on "PictoSearch" about what we have learned and suggested them to used another application in the meantime to test their own application in.

4.3.2 The Multi-Project

As stated we experienced a few problems throughout this sprint using others functions but the setting of the multi-project allowed us to quickly interact with the groups responsible for the functions troubling us and provide us the necessary support for solving the problems. A less desirable feature of multi-project setting which we experienced was the dependency on one group's functions which we used but the group responsible for it was difficult to reach and the problem seemed to persist. One of the other groups meanwhile fixed the problem so we actually could use the function. This should have been handled differently, maybe confronting them earlier and see what the problem was could have helped avoid this kind of problem.

4.4 Further Development

Based on section 4.1 and what have been made in this sprint we have made a list containing the work we want to work on in the next sprint:

- Allow for saving on to the database
  - Make the option for copying games from one profile to a another profile
  - Inset a profile selection button from the GUI group that allow for switching between profiles
- The profile photo should be used to represent the driver of the train
- The stations must be able to chance color
- Make visualized instructions for playing "Kategorispillet"
- Make a close button so you do not have to use the hardware back button
- Make general help to the game

Sprint IV 5

In the last sprint we continued working with "Kategorispillet". The sprint started on the 8th of May and ended on the 28th of May where we turned in the report. With the last sprint review on the 20th of May we had until the 19th of May to finish up our code but due to a delay we finished the code the 20th instead. In this sprint we worked on getting the stations to change colors to the color of the category. We also implemented the "profile selector" making it possible the change the profile from within the application and instructions for the game. We planned and performed some usability tests of the system.

5.1 Development

As previously mentioned the programming part of this sprint was shorter then other sprints. This meant the main focus was finishing up any loose ends in the code and to get the features we have been working on to function correctly together. We also implemented a way to delete a saved game, simply by adding a listener to a button on the list item, which when triggered removed the data from the save file.

5.1.1 Alert Messages

The development sections centers around changing the alert messages to "GComponent".

5.1.1.1 The Analysis

When we got the code all alert messages were using Android standard alert messages, and this did not fit the requirement stating that we must use the developed "GComponents". We therefore decided to change them all to use "GComponents" to make "Kategorispillet" more consistent with the rest of the project.

5.1.1.2 The Development

Making the alert messages use "GComponents" required changing a single function. This function was called "showAlertMessage" and codesnippet 5.1 shows its structure before the change. To fit the new components the function had to get a "View" as an extra parameter. This caused some problems since the function "isValidConfiguration" in "MainActivity" was using "showAlertMessage" and did not get a view. The "isValidConfiguration" function is used to show the user if something is wrong with their game configuration. The problem was solved by giving this function the extra parameter "View"

as well, since all functions using "isValidConfiguration" already contained a "View". This fixed the problem and the new code can be seen on codesnippet 5.2.

```
1 private void showAlertMessage(String message) {
2         this.errorDialog.setMessage(message);
3
4         this.errorDialog.show();
5   }
```

Listing 5.1: A code snippet of how the alert message was handled before.

```
1 private void showAlertMessage(String message, View view) {
2         new GDialogAlert(view.getContext(), message, null).show();
3   }
```

Listing 5.2: A code snippet of how the alert message was handled after the change.

5.1.2 Category Color

This development section centers around the feature of getting a category's color and setting the station according to it.

5.1.2.1 Analysis

In sprint III we made the stations able to change color but we did not have the color of the categories at that time to change to. Therefore we looked into getting this color and changing the stations to the found color. The root of the problem was, that a stations category was set as an ordinary pictogram which does not have a color specified. This meant that the category was not linked to a user created category but instead to an illustration of the wanted category. Therefore we could not find the color of a category with this implementation.

5.1.2.2 The Development

As we discovered from section 5.1.2.1 we know that a user do not choose a category but a pictogram instead which did not yield a color that we could use. So the first step was to implement a way for the user to choose between the categories. For this purpose we created a new activity which would handle the selection of a category.

To create the new activity we need an activity class, which we called "CategoryDialogActivity". In this class' "onCreate" method we inflate a simple layout file we have created for this purpose. The layout specifies two TextViews containing a headline and a sub headline, which are used to indicated the user whats the purpose of this activity is. The layout also specifies a "GList" we use to contain and display the categories, we get the categories by making an object of the class for fetching data from "OasisLib" which

contains methods to retrieve data from the local database. The result of the mentioned layout can be seen on figure 5.1.

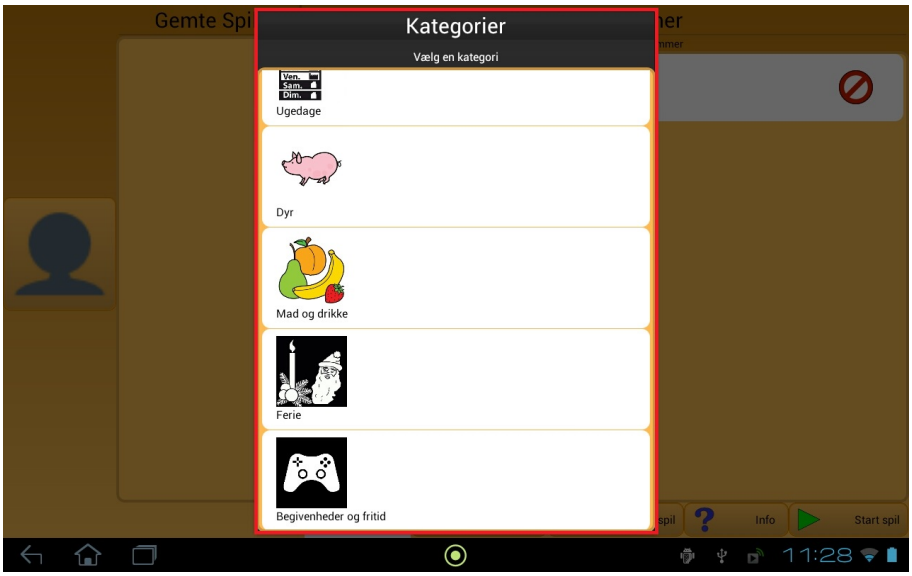


Figure 5.1: On this figure we see the layout of the described layout file.

After we have inflated our described layout we used an adapter to fill in elements in the "GList". These elements contain an "ImageView" with a category's image and a "GTextview" with the name. For filling the "GList" we use data that we retrieved from the local database. We set a listener to the "GList" which appends data to the return bundle. In the return bundle we append the selected category's id and color. This can be seen on code snippet 5.3.

```
1 LayoutInflater inflater = LayoutInflater.from(this);
2 View catelogview = inflater.inflate(R.layout.
      activity_category_dialog, null);
3 setContentView(catelogview);
4
5 GList categoryList = (GList)catelogview.findViewById(R.id.
      categoryList);
6
7 Helper localDataFetcher = null;
8 try {
9   localDataFetcher = new Helper(this);
10 } catch (Exception e) {}
11
12 List<Category> temp = localDataFetcher.categoryHelper.getCategories
      ();
13
14 for (Category i : temp){
15   this.listOfCategories.add(i);
16 }
17
18 catAdapter = new CategoryAdapter(this,listOfCategories);
19 categoryList.setAdapter(catAdapter);
20
21 categoryList.setOnItemClickListener(new AdapterView.
      OnItemClickListener() {
22   @Override
23     public void onItemClick(AdapterView<?> parent, View view, int
          position, long id) {
24       CategoryDialogActivity.this.resultIntent.putExtra("
             categoryColor",((Category)parent.getAdapter().getItem(
             position)).getColour());
25         CategoryDialogActivity.this.resultIntent.putExtra("
             categoryId",((Category)parent.getAdapter().getItem(
             position)).getId());
26         CategoryDialogActivity.super.setResult(Activity.RESULT_OK,
             CategoryDialogActivity.this.resultIntent);
27         CategoryDialogActivity.super.finish();
28     }
29 });
```

Listing 5.3: Code snippet containing code from the "onCreate" method for the "CategoryDialogActivity". The object "catAdapter" is of the class "CategoryAdapter" and listOfCategories is a ArrayList of "Category" class.

With the activity done, we had to handle the data returned by the activity. Upon returning to the main activity we set the selected stations "color" and "category" to the values we got from the "CategorydialogActivity". To know which station is selected we store the position of the station, containing the category being selected, in the variable "selectedStation". This storing of the position is done through the method "startCategory" located in the "mainActivity" class. This method is called through the listener appended to the specific category button. The code for receiving the data can be seen in code snippet 5.4.

```
1 case MainActivity.RECEIVE_CATEGORY:
2   this.listOfStations.stations.get(selectedStation).setCategory(
        data.getExtras().getInt("categoryId"));
3   this.listOfStations.stations.get(selectedStation).setColor(data.
        getExtras().getInt("categoryColor"));
4   break;
```

Listing 5.4: Code snippet containing code for receiving data from category activity. It is a case from the bigger switch in the method "onActivityResult".

With the category's id we now have the ability to fetch its image and use it to draw it both in the list displaying the stations but also on the stations in the game for indicating which category the station is connected to. We also have the color of the category which we use to set the station's color.

When we were about to merge this feature into the master branch and integrate it properly, changes happened to the different applications we depended on which caused problems for our application. Therefore we never got around to merge theses features into the master version, but the code is located on the "TrainCategoryColor" branch which the next group can merge into the master version of the "Kategorispillet" application.

5.1.3 GUI

We have in this sprint wrapped up the remaking of the GUI from section 4.2.2. Therefore we will use this section to also compared the new GUI to the GUI made in the last semester.

5.1.3.1 Analysis

The problem was, when the list containing the station configurations got too long it would squeeze the buttons in the bottom and cause the "Kategorispillet" application to crash. The buttons placement can be seen on figure 5.2 marked with a red circle. This picture is borrowed from the previous "Kategorispillet" group's report [Wortmann et al., 2013] to illustrate its original state. This problem emerged after we changed how the list worked, see section 4.2.2.2. A small problem was the icons we used varied from the provided icons from the GUI group.

Figure 5.2: A picture of the main setting's window where the buttons' placement is indicated with a red circle. Origin: [Wortmann et al., 2013].

The figures 5.3 and 5.4 shows the different layouts. We use the blue and red color to indicate the important layouts in this context. The blue indicates the layout containing the list of stations, while the red indicates the layout which contains the row of buttons. In the old layout the list of stations wrap around its content, which meant the layout would take up as much space as the items contained in it, while an empty layout were used as padding to keep the row of buttons at the bottom of the screen. This meant that when the list of stations grew too big the button layout got squashed which caused the application to crash. This was not a problem before, because the list simply overshadowed the buttons when it got too long.

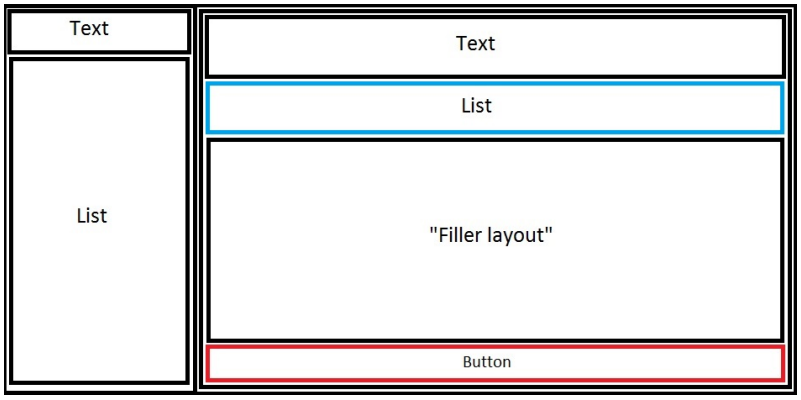


Figure 5.3: The old layout.

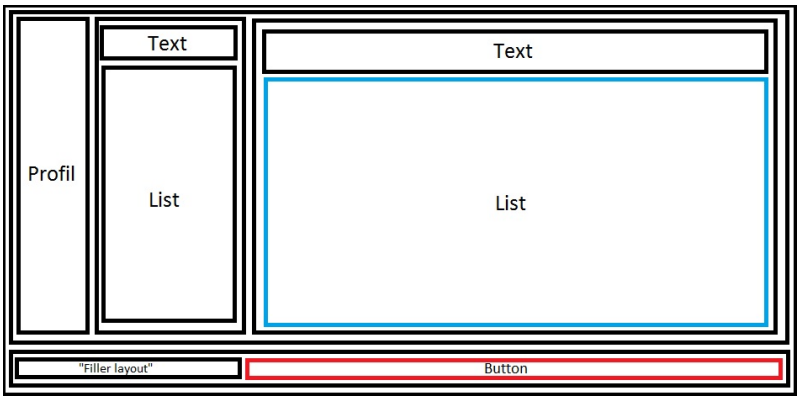


Figure 5.4: The new layout.

The problem with the icons on the buttons being the wrong type was caused by the buttons fetching the wrong resources. This was resolved by making them fetch the icons provided by the GUI group.

5.1.3.2 The Development

The problem with the buttons were solved by changing how the layouts were structured, to illustrate this we use two figures. These figures can be seen on figure 5.3 and 5.4.

In the new layout, seen on figure 5.4 we separated the buttons and the list into separate layouts, so the list of stations now fill the layout it is contained within, while the buttons are uninfluenced by it. This solved the problem with the list of stations squashing the buttons. With the new layout we also changed the buttons to expand using the unused space underneath the list containing the stations.

The issue with the wrong icons was solved by choosing the right image resources provided by the GUI group.

5.1.3.3 Conclusion of the GUI Development

To get an overview of all the changes we will compare it to a picture of how "Kategorispillet" looked before we started developing on it and compare it to the result we have achieved.

Figure 5.5: This picture illustrates the original layout of the "Kategorispillet" highlighting the buttons placement in particular. Origins: [Wortmann et al., 2013].

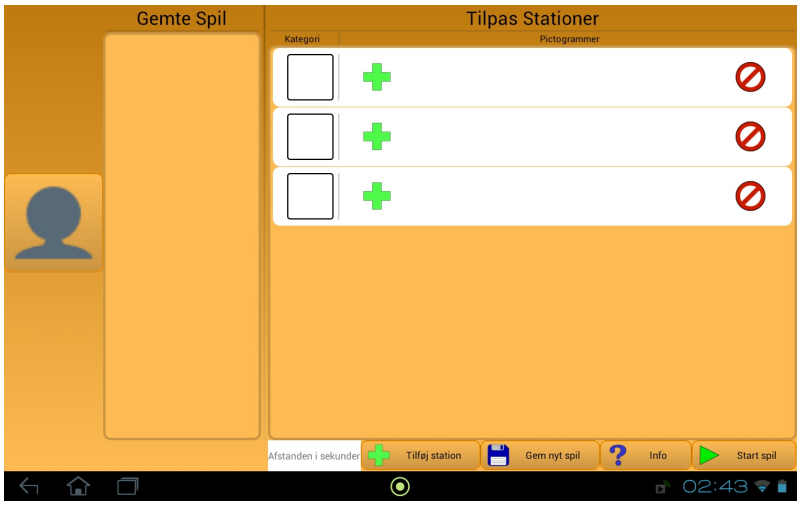


Figure 5.6: The result of the development.

The figures 5.5 and 5.6 shows the start and end state of the GUI. The picture, see figure 5.5, is borrowed from the previous "Kategorispillet" group's report [Wortmann et al., 2013] to show how it looked like before we started developing on it. As seen on the figure we have changed the profile selector list to the profile button, changed the icons, added components for interacting with the new features, changed the structure of the layout and refactored the way the lists were created. With the development we have removed the errors caused by the lists. The new lists also allows for scrolling while the old design extended the list till they were outside of the screen. Experiences from the usability tests showed us that the customers were familiar with the layout of the GUI and could navigate through it, see section 5.2 for the tests we performed.

5.2 Testing

In this sprint we did a usability test of "Kategorispillet". Since it was late in the sprint we knew we would not be able to make any corrections, so we used our results as guidance

to write a section about what the next people taking over "Kategorispillet" should focus on which can be found in section 5.5.

5.2.1 Instant Data Analysis

Instant Data Analysis (IDA) is a technique for making usability testing and evaluation [Kjeldskov et al.]. For conduction this technique you need a test monitor, a data logger and a facilitator. The use of the IDA technique follows immediately after the conduction of 4 to 6 think-aloud usability tests. Think-aloud usability tests centers around the testers explaining the monitors all their thoughts while performing the tests. The tests are performed by some testers that tries to use the product that should be tested. An example of test persons could be stakeholders, customers or intended users. The tests are monitored and documented so the tests later can be evaluate and used to document the results. For performing these test a set of roles are needed which are fulfilled by different persons. The roles includes a facilitator, data logger and a test monitor. During the tests the data logger should record problems. After the tests the data logger and the test monitor conduct an hour long analysis session, finding usability problems and ranking them on the scale: "Critical", "Serious" or "Cosmetic". This analysis session is assisted by the facilitator. The facilitator's role is to manage the session, ask questions for clarification, note all the problems and categorize them. After the session the facilitator spend an hour to an hour and a half on writing down the ranked list of the problems with short descriptions. At last the test monitor, data logger and facilitator run through the list to ensure all agree on them [Kjeldskov et al.].

5.2.2 The Usability Test

Five different groups were testing, and since the group member Mette Thomsen Pedersen was responsible for arranging the tests, because of her role as contact person to the costumers, she was taking summary for all the groups. Because of this, it was decided she should take the role as data logger in the test while another from our group had the role as test monitor. First the customer was presented with a short questionnaire asking about their experience with tablets as a whole as well as Android tablets specifically. The test person stated they were experienced with using an iPad but did not have experience using tablets running Android. After this they got a short briefing of what the test was about and that they should think-aloud. They then had to solve two tasks using all of the different functions in the game. While doing this they got no help unless they were completely stuck, and they were asked to tell everything they did and thought about the game while the data logger wrote everything down. When all the tasks were solved the test person was asked about their opinion of the game and asked if they had any suggestions for improvements to the system.

After our group and the other four groups from the multi-project were done testing, the test monitor and the data logger met again to talk about the observed problems and rank their severity. Because of the time pressure there were not enough manpower for another from our group to fill the role as the facilitator, so the test monitor and data logger made the list of problems on their own.

Due to lack of time in both our and the costumers schedule it was not possible to test with more than two costumers on separate days.

5.2.2.1 Evaluation

The result from our usability test yielded two critical, one serious and four cosmetic problems, see table 5.1.

Problems	Rating
The pictograms was not shown correctly	Critical
The application could not handle "æ", "ø" and "å"	Critical
The pictograms are not scrambled in the beginning	Serious
Missing response when a game is loaded	Cosmetic
Missing introduction in the start of the game	Cosmetic
The pictograms should appear on the train instead on the start station	Cosmetic
The reward for winning is not big enough	Cosmetic

Table 5.1: The results of the usability tests.

The first critical problem was that when some pictograms had been chosen they would not appear in the station. The problem's essence was that upon returning from "PictoSearch" to our application, the list used to display the stations was not updated probably. This problem was solved by calling the "notifyDataSetChanged" method on the adapter object, which updated the list so the pictograms would be displayed correctly. Since this test was made the day before the last sprint review we did not have the time to correct the rest of the problems, so they must be handled by the group taking over "Kategorispillet" next year. The second critical problem we found during the test were that the program could not handle "æ", "ø" and "å" in the name of a saved game configuration. A serious problem was the pictograms were not randomized before being lined up on the starting station making the game a bit too easy. The other cosmetic problems were there was missing a response when a saved game was loaded, no introduction in the start of the game confused the customer and might be solved by the pictograms appearing on the train from the start instead of needing to be dragged down from the start station, and the reward in the end should be more like the one in "Stemmespillet". "Stemmespillet" had a reward which was a golden trophy filling most of the screen with a sound indicating that the citizen had won.

All in all they thought the idea and design of the game was smart and nice to use, and they thought the game will be very useful, if the bugs gets corrected.

5.2.3 Integration Testing and Jenkins

Continuous integration is a software development practice where integrations are verified be an automated build to detect integration errors as soon as possible. Each member of a team is expected to integrate their work often, usually at least daily. This practice should lead to reduced integration problems and more time for developing software [Fowler].

In the multi-project we used Jenkins to make continuous integration. Jenkins is an extendable open source continuous integration server [Jenkins]. Jenkins takes a project and tries to build it, if it fails it reports that a build have failed with the console output available for debugging the problem. Jenkins in this multi-project is setup to fetch and build all projects from their master branches. Jenkins acted as a tool for getting an overview of the entire project before each sprint end to see whether each project could build. Jenkins would try to build a specific project when asked for it. We have experienced that Jenkins provided a good tool for checking that your project builds together with other

projects. An example of this could be days before a deadline where it is useful to know if the project is in a state where it can be built.

5.3 Demonstration

Before we transitioned from sprint III to sprint IV, each group made a demonstration of their product to show the customers. This demonstration was done to get the costumers' impression of the product and to catch any shortcomings the costumers might find. We also installed our applications on each of the costumers tablets so they could test the quality of our work. After each group had demonstrated their application the costumers as well as the other groups would provide feedback and suggestions.

The costumers were really pleased with our application in its current state, as described in sprint III. Though one of the other groups suggested that instead of saving specific games, it might be a better idea to be able to save stations as templates for future use. We thought about the suggestions, but since we focused on finishing up the loose ends in sprint IV, we used the suggestions to write the "Further Development" section, see section 5.5.

5.4 Reflection

This was the last sprint and therefore we used to the time to finish up any loose ends.

5.4.1 Collaboration

On the last day of this sprint the whole system came under a lot of pressure and problems occurred everywhere because of the late integration of the remote database into the system. The problems were caused by SQL-statements which caused the launcher to crash which meant other applications could not be launched. Therefore there were a lot of panic the morning we should deliver the applications to the customers and we spent most of our time trying to get our application working with the changes made to the launcher. This unfortunately made the application unusable in one of the two planned usability tests, as we did not manage to fix it in time, which meant we could not complete the test. From this we learned that changes to a system should not happen so close to the deadline where there is not enough time to correct potential errors.

Before the integration that caused a lot of panic, we also had a problem with implementing "GScrollView" which we contacted the GUI group about. After some debugging and discussion they said that the cause to the problem was the way we used the component and that they would have to change the way they had implemented the feature for it to work and would not have the time for it. Therefore we went back to using the normal Android component, the "ScrollView", in this case because it provided the necessary functionally that we needed at that moment.

As we focused on finishing the "Kategorispillet" application we experienced a problem while implementing the code to find a category's color. If a category's icon were made in the application "Piktotegner" the "Kategorispillet" application would crash. We noticed through the use of the wrapping property of items in a list, that pictograms created in "Piktotegner" appeared to be bigger than the pictograms from the standard package that had been uploaded to database and this might likely was the problem. The problem only occurred when choosing these self-made pictograms so we believe the size was the

issue. We contacted the "Piktotegner" group and asked if they could find a solution to the problem. Within an hour they came back with a possible solution we could try. This was done on the last day of development, doing the panic caused by the problem from integrating the remote database into the system and therefore we sadly did not get to test if it worked.

5.4.2 The Multi-Project

In this sprint, the time was used on implementing the function to remove a saved game in "Kategorispillet" and making more of the GUI components into "GComponents" - to be more precise the alert messages that are used to notify that a game configuration is invalid and to give a warning if "PictoSearch" or "Kategoriværktøjet" is not installed on the tablet. The list used to display the stations in a game and the image on the buttons that remove a pictogram from a station and the button that allowed the user to choose a color for a station, did not get done by the end of developing phase.

We tried to make the list that was showing the pictograms belonging to a station into a scroll view. "GComponents" had a scroll view but it could not handle being in a list as this made "Kategorispillet" crash when we tried to scroll in the list. If we had started working on the list in a previous sprint we could have had some more time to work on the problem. As the problem was caused by the way the list had been constructed when we took over the code. We decided to discard the changes and return to a fixed size list instead, since we wanted to focus on some of the other problems instead.

As in the last sprint there were some problems with a dependency on PictoSearch's functions we used, but the group responsible for it was difficult to reach and the problem again seemed to persist. The resulted in for the third time that some of the other groups had to take over the PictoSearch's work and fixing the problems so we actually could use the function. In the 48 hour before the sprint ended a problem occurred when the local database could synchronization with the remote database. The problem was that "PictoSearch" tried to load the approximately 22,000 pictograms, that had been uploaded to the remote database, into the memory of the tablet all at once. Another problem was that "PictoSearch" did not release used memory when it was closed, which caused it to crash when used repeatedly due to running out of memory. These errors was fixed with a hot-fix by one group member each from "remoteDB", "GIRAF", "Piktooplæser" and "Livshistorier / Skema" groups. After the hot-fix of the two first problems, a new hot-fix was made from one group member from "Piktooplæser" and from "Livshistorier / Skema" that allow to search for a string in the local database and only fetching 100 pitrograms at a time in "PictoSearch" at any time making sure it does not use up all memory and does not crash. All these problems resulted in a lot of stress and frustration up to the last sprint end review.

5.5 Further Development

As this was the last sprint for us, we made a list of the feedback we got from the customers as well as suggestions for the the next semester group to either fix or develop.

- Because the code has been under development by a lot of different groups, we suggests to simply look at the game mechanics and then build them from scratch, to make sure the code follow a shared convention and design pattern.

- Collaborate with the "Stemme Spillet" project to make sure you use the same icons and generally do similar things the same way. For example when user finishes the game they could get the same type of reward.
- Make the stations able to change color. Our group tried and almost had it completed, the code we worked on have been saved on a separate branch on git, so the next group might benefit from it. The stations are already set up to receive a color, but we did not have the time to implement the code in the "MainActivity" class completely that should fetch the category and then parse the data to the "GameActivity" class.
- Make the stations even more flexible by redesigning the templates, so for example the current box is only framed by a black line and then make doors, roofs and windows into individual templates.
- Like stations should change color, make it so the train itself also can change its color.
- The customers would like to able to name each cart, so they can sort the pictograms as they place them in the carts. Another idea could be to simply shuffle the pictograms onto the carts from the start, instead of having the user drag them from the starting station.
- In the current design if too many pictograms have been selected the picograms will be shown on one line and will be reduced in size. So change the way pictograms are displayed, this could be done by insetting more carts or how the pictograms are placed on each cart.
- It was suggested that instead of saving whole game configurations, it should be possible to save a station as a template making it easier to mix up the games.
- The game currently have a help button in the game section which just a wall of text and this might be changed to simple illustrative instructions when the help button is pressed, or make it so the guardian can set whether or not they should be shown when the game starts.
- Develop an easier way to share saved games/stations with profiles. Right now it is possible to share stations by selecting a saved game, so the stations are shown on the edit list, then change to a different profile and the stations will still be there and can saved to the selected profile.
- With the new features, a new main menu interface design would also be necessary as there would be new options that would need a good placement.
- Assure that the profile picture is shown as the driver of the train.
- Assure the application can handle "æ", "ø" and "å" in names of saved games.
- We recommend that the specification requirements from this sprint are reevaluated with the customers and used as foundation for a new version of the specification requirements.

Reflections and Experiences 6

In the first sprint we chose to work on the specification requirements for the whole multi-project this turned out to be both a good and a bad experience. Good because it was very educational to try to talk to real customers/stakeholders, making interview and from these interview analyzing our way to a list of requirements. Another very good and educational experience was seeing how the citizens with autism and their guardians lived their daily life as well as the tools they used.

A bad experience from sprint I was that our expectations on how the level of technical details for the specification requirements was not the same as the rest of the multi-project groups. This led to a lot of frustration because we had not made sure our expectations were the same and resulted in taking time out of sprint II to finish the requirements documents. We learned when your work affects the entire semester it is important to talk about what is expected at the start, and upholding deadlines instead of just complaining when a nearly finished product is presented and the deadline for changes was way overdue.

Through the rest of the sprints we worked on the application "Kategorispillet", which also brought both good, bad and interesting experiences with it. When we first got our hands on the game it could not run on the tablet, because of some changes to the local database. The code was badly structured and hard to navigate through when we got it from the group, who had it in sprint I so we started on re-factoring the code.

We have throughout the sprints worked with the code and learned a lot about how an activity works, its life cycle and how to program on an Android system. We have also learned how layouts work and how they should be handled in the code. The project also provided us with the experience to grasp code developed by another group. The project has also provided a lot of opportunities to work with problems and fix them, sometimes in collaboration with other groups. We have learned about GIT as a tool for version control and how to use it. The multi-project also gave us experience in programming within a bigger context, which problems that can occur when depending on others code and how to communicate with those responsible for the code we are depending on.

It was also a new and educational experience working in a multi-project which could be stressful at times. A very good example is the end of sprint IV where all application had to be ready for release to the customers/stakeholders. All hell breaks loose when a late change to the database makes problems for all applications and nothing was working. Here the need for strong leaders and hard working people was really at display, when other groups steps up and spends hours of overwork to make sure the problem is fixed.

We also learned through time how to structure meeting between the different groups, and the importance of having different people being responsible for different areas, so the responsibility are not on only one person's shoulders.

All in all it was a great and very educational though very hard and at times stressful experience to try to work together with so many groups. We would suggest that next year they split up in two multi-projects since they, like we did not, probably do not have much experience in working in such big work environment.

Bibliography

Android, **a**. Android. *Adapter*.
`http://developer.android.com/reference/android/widget/Adapter.html` [Last seen: 2014/05/09].

Android, **b**. Android. *Activity*.
`http://developer.android.com/reference/android/app/Activity.html` [Last seen: 2014/05/12].

Android, **c**. Android. *ListView*.
`http://developer.android.com/reference/android/widget/ListView.html` [Last seen: 2014/05/09].

Android, **d**. Android. *View*.
`http://developer.android.com/reference/android/view/View.html` [Last seen: 2014/05/09].

Android, **e**. Android. *Parcel*.
`http://developer.android.com/reference/android/os/Parcel.html` [Last seen: 2014/04/22].

Benyon, **2010**. David Benyon. *Designing Interactive Systems*. Pearson Education Limited, 2010.

Fowler. Martin Fowler. *Continuous Integration*.
`http://martinfowler.com/articles/continuousIntegration.html` [Last seen: 2014/05/22].

Geagea et al., **2010**. Sarah Geagea, Sheng Zhang, Niclas Sahlin, Faegheh Hasibi, Farhan Hameed, Elmira Rafiyan and Magnus Ekberg, 2010. *Software Requirements Specification*. `http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf` [Last seen: 2014/04/23].

Grandleware. Grandleware. *Gradle: The New Android Build System*.
`http://www.gradleware.com/resources/tech/android` [Last seen: 2014/05/09].

Jenkins. Jenkins. *Jenkins*. `http://jenkins-ci.org/` [Last seen: 2014/05/22].

Jensen et al., **2013**. Anders Jensen, Danial Bøcker Sørensen, Tom Petersen and Jakob Albertsen. *Cars A Voice Controlled Game*, Aalborg University, 2013.

Kjeldskov et al. Jesper Kjeldskov, Mikael B. Skov and Jann Stage. *Instant Data Analysis: Conducting Usability Evaluations in a Day*. URL `http://cmapspublic.ihmc.us/rid=1H8MBVM94-1PBWLN7-3VVH/instantdataanalysis-kjeldskov.pdf`.

Knudsen et al., 2013. Søren Knudsen, Christian Lykke, Søren Enevoldsen and Sam Olesen. *GIRAF: Zebra*, Aalborg University, 2013.

LLC. JABstone LLC. URL `https://play.google.com/store/apps/details?id=com.jabstone.jabtalk.basic`. *JABtalk*. Application.

Mortensen et al., 2013. Christian Mortensen, Brian Holbech and Nedim Cokljat. *Tortoise An Application for Managing Autistic Children's Life Stories in GIRAF*, Aalborg University, 2013.

Wortmann et al., 2013. Jacob Karstensen Wortmann, Jesper Riemer Andersen, Nicklas Andersen and Simon Reedtz Olesen. *Interactive Learning Exercise for Children With Autism*, Aalborg University, 2013.

Interview Questions A

- Which tools do you use in your daily routine?
  - Which functionality in the tools do you like?
  - Which functionality do you find irrelevant in a digital solution?
- Which previous apps have you worked with, if any?
  - How did the app perform?
  - Is there anything you would improve?
  - Any functionality that you find less important compared to other?
    - Example of apps: Timer, Sekvenser(Zebra), Train, Life Stores(Tortoise), Pictocreator and Categorizer(Cat).
- How important is it to isolate the solutions individual functionality?
- How important do you find the flexibility of the solution?
  - Color/text/pictures
  - Some kids do not like a specific color, should it be possible to change the background color or is it better to keep it more consistent?
- Is it important that the apps follows the same design scheme/theme
- What information is important to store in the system about the children?
  - Possible relevant information: Name, group, guardian and picture.
- What is most important to you, new functionality or that previous functionality finishes development?
  - Such as a day/week schedule.
  - Barrel(Surprise game).
  - Train, expansion of the game.
- Regardless of us, what should the product look like?
- Anything else?
  - Any specific wishes for the 55" screen?
  - What wishes do you have for a solution and how would you priorities them?

Specification Requirements B

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

B.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.

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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)

B.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.

User Stories C

We have used the user stories from report [Jensen et al., 2013], [Knudsen et al., 2013] and [Mortensen et al., 2013] 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.

Project DVD D

The DVD found on this page contains the following:

- The source code for this semester project.
- An apk of the game for this semester project.
- The report in the format PDF.