Parental Control - In Smart Home

SW7 Projekt
Group SW701E13
Department of Computer Science
Aalborg University
December 20^{th} 2013

AALBORG UNIVERSITY
STUDENT REPORT

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

Title:

Parental Control - In Smart Home

Theme:

Internet Technology

Project period:
P7, Autumn Term 2013

Project group:
SW701E13

Participants:
Jens M. Lauridsen
Johan Sørensen
Lars Chr. Pedersen
Tommy Knudsen
Lisbeth Nielsen
Jakob Jørgensen

Supervisor:
Hua Lu

Synopsis:

Synopsis goes here...

Circulation: 7

Page count: 122

Appendix count and type: 0

Finished on December 20^{th} 2013

Preface

This report is written by six students from the Department of Computer Science at Aalborg University, and is a project under the subject ”Internet of Things”. The four students are studying as Software Engineers at their 7th semester.

This report serves as a suggestion of how to moderate children's increasing use of media.

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

The group would like to personally thank:

Signatures:

Jens Mikkel Lauridsen

Johan Sørensen

Lars Chr. Pedersen

Tommy Knudsen

Lisbeth Nielsen

Jakob Jørgensen

Contents

Part I

Analysis

Chapter 1

Parental Control - Why we need it

Today children spend more and more time watching TV and playing video games. Research shows that there has been an increase in children's use of electronic media in later years. [10]
Research also shows that this increase in media usage have consequences for the children. It has been linked to sleep deprivation[3] and lack of physical exercise[3].

In this report we give a suggestion towards a solution to control this problem, by extending expanding on the concept of Parental Control over electronic devices.

1.1 Words

In this section keywords are explained, in order to easier understand the report.

Smart Home can refer to multiple forms of improvement on a home. Most commonly is: home automation, environment friendly improvements and power saving improvements. In this report we will only regard Smart Home in relation to home automation.

Parental Control is a concept most commonly found in televisions, computers and internet-browsers. They all serve to limit the access to inappropriate content, either in the form of adult content, or advanced features that should not be touched.

Internet of Things is the subject of this project and is a concept where our society is going towards a greater deal of automation. The concept is based on different entities being able to adjust and act on their own and with each other.

An example could be your coffee machine automatic reacting to you opening the front door and pours a cop of coffee for when you enter the kitchen, but if you do not pick up the coffee, or drink it in time, the coffee machine will start to register this, see patterns and modify its behavior accordingly.[11]
In this project, we focus on the ability for different entities to act with each other.

Media is a term to cover any media, from tablet, phone, television, computer to gaming consoles.

1.2 Increased media usage - It is a problem

As stated in the intro to this chapter, the increase in media usage has been linked to sleep deprivation and lack of physical exercise. Which in turn lead to higher risks of getting type 2 diabetes and concentration issues.
Worse is that, there seems to be an increase in the media usage of each generation. In an article from the Danish Health Department, they point out that over a 5 year period children between the age of 10 and 15, have had an increase in media usage from 1.57 hours a day, to 2.47 hours a day, a rise of more than 50%.[10]

If this tendency is not dealt with, the risk of children getting type 2 diabetes will also rise. Our suggestion to help lower the media usage is to give the parents proper tools to limit their child's us of media in a fair way, which will be discussed in the next section.

1.3 Parental Control - How to extend it

As previously mentioned parental control is commonly found in televisions, computers and internet-browsers. But none of these tools are meant to limit the usage of a device totally, they are able to do so, but it will result in parents having to enter passwords for their children every time they need to use the device. Maybe add a reference? -Jakob
The idea behind this project, is to extend the normal parental control, so that a child has a limited usage of media, without having the constant interaction from parents. But simply restricting a child's media usage is not enough, for the idea to work. We believe it is also necessary to inspire the child to physical activity, e.g. by helping out at the house.

This, coupled with the subject of this report, "Internet of Things" have lead to an idea that consists of the following features:

1. A way to set up permissions for which media and when they can be accessed

2. A way to set up rules to make exceptions to these permissions

3. A way to monitor/limit the usage of media used by a child

4. A way to uniquely identify the child in the physical world

5. A way to inspire children to physical activity

After having figuring out what the key features of the system should be, we started brainstorming how to implement them.
The 1st and 2nd feature is best maintained from a website with a graphical interface. This is due to the complexity of the idea and the realization that simplifying permissions and rules into something a machine would understand is a hard task and an even harder one for non-computer graduate, more information on how we implemented rules can be read in section 5.2 on page 33.
Among other solutions could have been a desktop- or phone application.

For the 3rd feature has a few different solutions. One solution was to write code for a bunch of media, that then would be able to work directly with the administration web-site, but this would force us to write code for every media device from every fabricator.
A better alternative is to create a device that can turn the power on and off for a media instead. We are aware that this solution might influence some media in a bad way, but most media today is setup to handle a sudden disappearance of electrical power.
More about the chosen device can be found in chapter 4 on page 27.

For the 5th feature there are different ways to identify child users. Mobile phones, fingerprints and NFC tags are only a few potential directions. What is important is that we need a way for the device from feature no.3 to uniquely identify each user, such that the rules from the 1st and 2nd feature can be upheld.
More about the chosen identification can be found in chapter 4 on page 27.
We have also taken into account that we want the system to be able to decide if a media device is to be allowed to turn on, without having to involve a parent. This means that the identification of a child will have to be sufficient information to make that decision.

For feature no.5 we have seriously considered how to motivate children in a way that will go along with the idea of the parental control system. We have come up with the idea of implementing chores into the system. We believe that if a sort of point system is used to determine how much time a child will be able to use on electronic media per week. Then a way to obtain extra points could be to do chores. Meaning if the child has a habit of spending a lot of time using electronic media, the only way to obtain more time will be to do chores. Chores that could consist of moving the lawn or other physical exercises.

This idea then boils down to these 3 key aspects:

- A device to toggle the power of an electronic media
- An individual key for each child that can be read by the device
- A website to set up rules, permissions and view/modify allowed usage of electronic media

what follows is a formal description of the problem, in the form of the problem statement followed by a more detailed explanation of how the system is designed and implemented.

Chapter 2

Problem Statement

As the previous chapter explained, children's increase in media usage is becoming a problem. This is a problem we want to solve by the usage of modern technology, internet applications and physical recognition of users.
In this chapter a formal definition of the problem is written together with an emphasis on what challenges this problem rises.

Parents are not able to help their children administer their IT/TV consumption.
This results in their children getting a lessened learning ability, a bad sleeping pattern and a higher risk of type 2 diabetes.
How can hardware identification and webservices give parents the tools to help their children manage their media consumption?

- How do we identify unique users in a subtle and child friendly way.
- How do we enforce restrictions on media devices.
- How do we facilitate concepts as rules, permissions and chores without parents interaction.

With this formal statement of the problem in mind, the next chapter goes into detail of how we intend to solve these challenges.

Part II

Solution

Chapter 3

Overall System Design

In this chapter the challenges from the problem statement are looked upon, and an overview of what will be needed to overcome these challenges, will be formulated.

3.1 Designing a solution

During the design of the system it was decided, what type of hardware that would be needed to accommodate the challenges of the problem.
In order to know what hardware was needed, was a design of the overall solution and designate functionality to components needed.

3.1.1 Identify Unique Users

The first challenge in the problem statement was: ”How do we identify unique users in a subtle and child friendly way.”.
As mentioned in the analysis there are several different ways to do this, including fingerprints and NFC tags. Common for them all is that they require some sort of scanning device to be used. For the choice of hardware see chapter 4.
In order to verify that the user is actually a user on this families system this scanner must have a way to contact some sort of database or an overview of members. Which leads us to the next challenge.

3.1.2 Enforce Restrictions on Media Devices

As mentioned in the analysis, see section 1.3, it is necessary to find a way to physically restrict the usage of medias. We first thought that the best solution would be to implement it directly into all media, but this would result in an endless amount of coding, since a code base would be needed to be implemented for each device it would run on.
Instead we will take the restriction a bit further. Most media today run on

electricity and without batteries, meaning if the power is cut to the media, it will be restricted.
The idea developed into a combination with the solution for the previous challenge and resulted into what we like to call a ”controller”. It is a device equipped with a relay, an internet connection and a scanner.
Using the relay to control the power to media, using the scanner to identify users and the internet connection to look up in a sub-global[1] register of users, which again leads us to the next and final challenge in the problem statement. Here we already have three different devices communicating, on their own, emphasizing the idea of Internet Of Things.

3.1.3 Facilitate Rules, Permissions and Chores

Rules, permissions and chores are complicated concepts to explain which will be explained in further detail in chapter 5. The short version however is:
Rules is a combination of an action and a condition. Where actions could for example be to allow the usage of a device, give points[2] to user or block user. And a condition could for example be a period of time or a repeating event such as every Monday in every week.
Permissions is a subtype of rules. Permissions is only used to tell the system the normal allowed usage of media and rules is there to overwrite these permissions in special cases, or if the user wants to customize the system.
Chores is intended as an idea to get the children to do physical activities. Parents create chores and designate a set of points that can be earned by doing this chore.

By now there already is a lot of information in the system. We have users, rules, permissions, chores, points, timestamps and time periods, and if the system is to be extended to learn anything from the users behavior it should also include logs.
This is more information than should ever be put into a simple hardcoded text sheet. Meaning the system will need a database and considering that we want the devices to independently be able to fetch the information they will need a good solution which would be to store the information on a server accessible by the controller.
Also considering the advanced concepts of rules, permissions and chores it is very likely that a graphical interface will be needed and since there already is the need for a server in our system it would be a good idea to make a web-site for the user to maintain edit or add these rules.

With all these required features we still need to specify how they communicate and this should implement a structure that allows for changes later on. Simply because this is a prototype system and will most likely need improvements if it was to ever be commercialized or extended, which is easiest done if

[1] Global in the scope of each families system.
[2] Points is the currency we intend to use in the system to pay for media usage.

the system is build for it from the start.

Next follows a section that puts the finishing touches on the system design and finally gives it a name.

3.2 Media-Online Management

The Media-Online Management (MOM) is the name of the system which is our solution on the problem statement. To give an overview of the complete system a rich picture [6] have been made, see the figure 3.1. The figure shows a home environment with a TV (media), computer and an internet connection, and it shows a server.

There are two main use patterns. The first is a parent who manage their MOM system using the website from a PC to add, change or delete system settings. The settings will from the server that host the website changed in the database as seen in the right side of the figure 3.1.

The second use pattern is pictured in the left side of the figure. It is a child user who wants to use a media which in this case is the television. But to watch television it need power and its power source is blocked by the controller. So the child needs to scan his tag and then the controller sends a message to the server which then reply whether the television can be turned on or not. When the child is done he must scan again such that points can be withdrawn from his user profile. If the child does not have any points he cannot turn on the television nor can he turn it on if a rule or permission does not allow it. If a parent wants to watch television without being restricted in any way he can make a rule that gives him unlimited access, but he would still need to scan his tag before and after using the television.

On the server there will be a database, files that generate the website, API(a web service), and a Daemon. These elements of the system will be explained further in section 6 on page 39 and after that the system architecture can be presented in section 6.5 on page 41. However, first the hardware choices made for the system will be went through and also the concepts of rules, permissions and chores needs to be explained.
The next chapter is about the hardware choices.

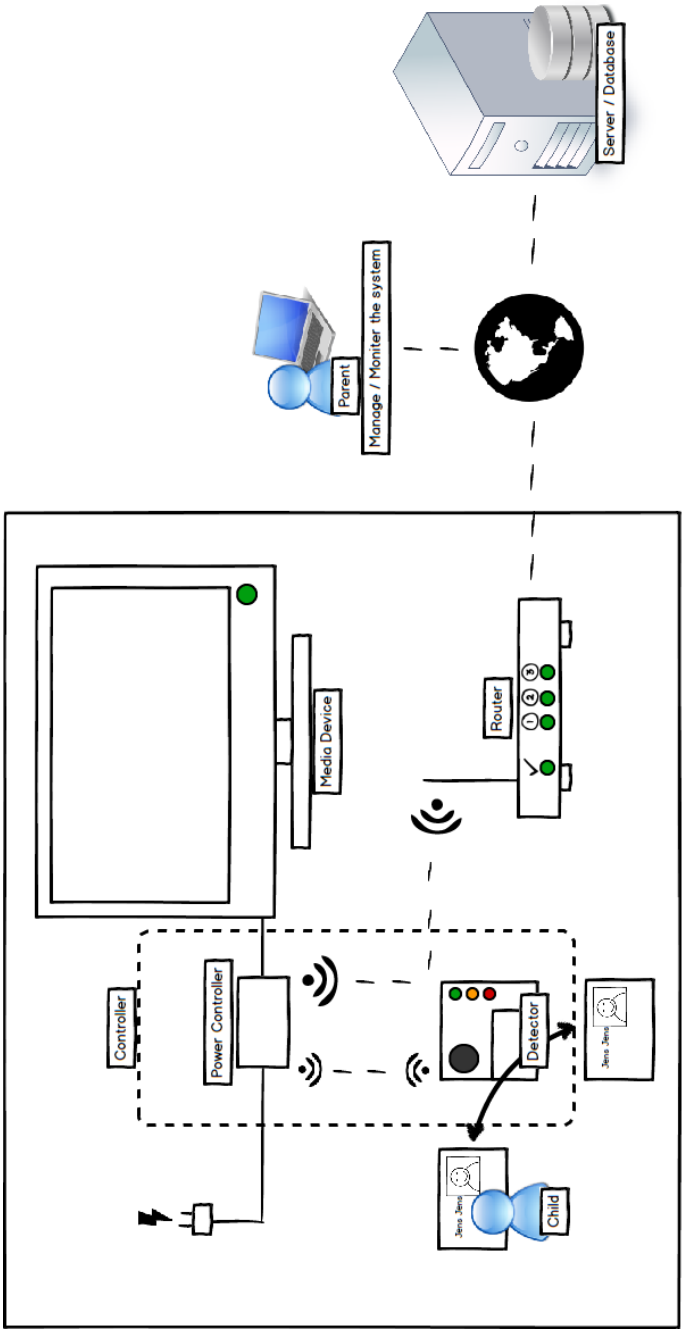

Figure 3.1: system overview

Chapter 4

Hardware

In this section the hardware choices of this project will be talked about and analyzed. First the choice of hardware platform will be discussed, next the choice of hardware identification needed for the system and then the server setup. Followed each by a conclusion stating the hardware chosen.

4.1 Hardware Platform

In order to control the power to a media(i.e. Television or Computer), some analog power control is needed. Typically this is controlled by an embedded microcontroller If this is typical, we need a source , therefore the choice of hardware platform is to pick the best microcontroller for the project. A development board is usually used during development . A development board is a single printed circuit board with a microcontroller and everything needed to get the microcontroller running. Every I/O[1](generally General-purpose I/O) of the microcontroller is also made easily available using group pins. Picking a hardware platform is then condensed into picking a development board. Following is a comparison of a few different development boards.

Arduino

kilde Arduino development boards come in a few variations, but common among all but one of them is that they are based on Atmel microcontrollers. Along with the development boards, Arduino also comes with its own software library `Wiring` which simplifies common I/O tasks. The Arduino development boards also implement a standard called `shields`, which are expansion circuit boards with pin headers that fit onto the development boards, with additional pins so that shields can be stacked. This makes the hardware platform easily extendable and flexible using this stacking system. Examples of shields could be GPS, Ethernet, LCD displays or breadboards.

[1] Input/Output

Teensy

kilde Teensy, like Arduino is based on a Atmel microcontroller, but unlike Arduino is made to be as small as possible. Teensy also supplies a add-on library that makes Arduino programs compatible. Power is also supplied via USB instead of a dedicated power input on the Arduino, which also makes it cheaper. Teensy also does not have an equivalent of shields.

Raspberry Pi

kilde Raspberry Pi, unlike Arduino and Teensy uses a "full-sized" ARM processor, which makes it more like a regular computer than a microcontroller. Despite that it has GPI/O pins just like Arduino and Teensy, which means that it has the capabilities needed for the project. Being ARM based means that the Raspberry Pi can also install computer operating systems such as Linux based operating systems. The ARM processor is also many times more powerful than the Atmel microcontrollers. This however comes with negatives such as more power consumption, unused modules(GPU, sound or SD card storage) resulting in a larger board and a higher cost.

4.1.1 Conclusion - Platform

In conclusion the platform chosen is Arduino. Arduino is chosen because the university already has those development boards, which means that no time is wasted getting another platform approved and bought. Arduino is also chosen because of their shield design, which enables the use of a Ethernet shield as well as a RFID shield. Some of the group members has also already worked with Arduino, which means a lower learning curve.

4.2 Hardware Identification

In order for the person using the system to identify themselves, there has to be some form of hardware identification. This hardware identification has to be unique for that person, so that the system can find out the allowed time of he can use on the media. Several factors have to be considered when choosing what hardware identification method to use, such as cost, user friendliness and security. Following is a comparison between some of the hardware identification methods available.

RFID

kilde RFID, short for Radio-frequency Identification, is a method of using either passive or active tags for transferring data between a reader and a tag. A tag is an embedded chip with an antenna, that when receiving power transmits the data on the chip to any readers within range. Active tags have their own power source and thus can be scaled to emit their data over long distances.

Passive tags on the other hand do not have an internal power source, and instead is powered by the reader via magnetic fields. This makes passive tags work on smaller ranges. A tag can be compared to a bar code, with the difference being that tags scale much better when needing more data storage and the tag does not have to be visible, i.e. it can be embedded into for example key-cards, jewelry and such.

NFC

kilde NFC, short for Near Field Communication, is a set of standards for the same technology that RFID uses. The differences between them is that NFC is a strict standard, uses only passive tags and only works over small distances. NFC is also the technology being embedded in smartphones, which means that the projects system could be made compatible with apps.

Fingerprint

kilde Fingerprint is a method of recognizing whether a persons fingerprint matches another already scanned fingerprint. A number of technologies can be used to scan a persons fingerprint, such as optical and ultrasonic. An algorithm is also needed to compare certain features of the fingerprint in order to be able to compare between fingerprints [4]. One of the advantages of fingerprint is that the only equipment needed is the scanner and something to compare them, this means that the cost could be kept down. Fingerprints also have some disadvantages. One of the disadvantages is gathering the fingerprints of all family members where as other methods can be established by on person will fingerprints require all members are involved in the setup process. Another disadvantage is that it has been shown that children fingerprint change over time as they grow up, which could mean that the fingerprint has to be scanned again after a while. Fingerprints is also characterized as personal information and therefore need extra security measures.

Iris Recognition

Iris Recognition is a method of scanning and comparing the iris of two eyes. The technology is somewhat new and have many good qualities. Our iris is one of the most unique traits as human and therefore is finding one the same iris as you is very on likely. The uniqueness of iris is also very characteristic and therefore easy to distinguish if two irises are a match and is therefore gives at more secure analysts[4]. As a new technology an iris scanner is not cheap and is not simple to implement in household system with the need for many scanners.

4.2.1 Conclusion - Identification

In conclusion the identification chosen is RFID, because it could be provided by the university, however, originally NFC was preferred. The RFID has been chosen because it is easily accessible, and all of the practices of used with RFID

can be translated onto NFC. The reason for choosing NFC over RFID, is that it is a well defined standard, it is compatible with the technology used in smartphones and it is short range so no accidental reads happen. The reason for choosing NFC/RFID over fingerprint is that the security needed for fingerprints do not scale with simplicity and the possibility of a child's fingerprint changing too much. The reason for choosing NFC/RFID over iris is primarily the cost, as the cost of a high quality camera is too high since multiple cameras would be needed.

4.3 Servers

In order to maintain the wast amount of information collected from the users and in order to offer a graphical interface to add rules, permissions and chores to the system a server setup will be needed.
Such server work from a private home or collected global server, maintained by the manufacturer of the system.

We here list the pros and cons of having a decentralized server setup versus centralized server setup. kilde

Decentralized Server Setup
Pros:

- Maintains proper functionality even if the internet connection out of the house is broken.
- Every person can edit the base code
- Could be integrated in a Smart Home setup

Cons:

- Each user will have to setup the system themselves or pay for a professional
- If the server malfunctions the user is the one with the problem
- Need to own a domain name and an external IP address in order to access the system from outside

Centralized Server Setup
Pros:

- The setup of the system is for the user only registering on the website
- The user does not have to pay for extra electricity consumption of the server
- If the server fails, proper security measures can be setup to handle these problems

Cons:

- If the internet connection malfunctions for what ever reason, the system will not work optimally.
- The user is limited to the functionality we offer

As can be seen by the lists, most pros in the central setup is cons in the decentralized setup and vice verse.
This project focuses on developing a user friendly and upgradable system, and therefore a centralized server setup has been chosen. The centralized system offers the easiest way for users to get started and it also makes it easier to deal with problems. The only problem that a user can face, is the lack of internet connection or power, which in our modern time is unlikely for a considerable amount of time.
Also the user would never have to do anything to keep their system up to date, since this would be handled on the server side.

Chapter 5

General Design Concepts

In the Media-Online Management there are a few concepts that will be used through out the report. These are chore, permission and rules. The meaning of each will be explain in following sections.

5.1 Chore

A chore in MOM is a representation of a house chore that is to be done regularly. We have included chore into this system to encourage children to help with the house chores which gives more time for media as a reward. Therefore each chore needs a number of points which will be given when the child has done the house chore. An example on a house chore could be to take out the garbage, then the chore in MOM would have a name: 'take out garbage', possible a bigger description of the chore: 'remember to sort the garbage into the correct trash cans' and when the chore is done some points is given: '10'.

One disadvantage about the chore design is that the parent needs to use the website to award the child with points for doing a chore. We would have liked to automate it further, but to limit the scope of the project this would be future work.

5.2 Rule

In the system a rule can be many things the following is just a few examples of rules:

- Susan has access to use the PC from 15:00 to 17:00 each day.
- The first Monday in a month Peter's points is increased by 100
- Mom and Dad's profiles have unlimited time and unrestricted access to any media

The rule has been included because it gives the parents a more powerful method to control their children's use of medias. But there is one disadvantage with rules, it might be complicated for the user to understand how to use them. This means we must take care to design the concept of rules, so that it is easy to use and understand their effects.

A rule should be connected to one or more user profiles since the rules otherwise would be unnecessary. A rule consist of a name, a set of actions and a restricted set of conditions depending on the actions. A condition is used to decide when a rule is relevant which is when all conditions fits. An action is something that can or should be done if all of the rule's conditions hold. To see a quick overview of the different conditions and their name:

Timeperiod the action can be done in this time period.

Controller on the action can be done if a specific controller is turn on.

Controller off the action can be done if a specific controller is turn off.

True The action can always be taken. This is actually kind of a time period but there is no time bounds.

An overview of the actions and their name is listed below:

Block user it will block the profiles of all profiles connected to the rule.

Add points it will add points to the profiles' points.

Delete points it will delete points to the profiles' points.

Set maximum of point it will set a maximum number of points that a profile can have.

Unlimited time it will give the profile unlimited time to be spend on any media.

Access any controller it will give the profile access to any media in the system.

Cannot access any controller the profile will not have access to any media.

Access controller it will give the profile access to a specific media.

Cannot access controller it will block the user from using a specific media.

Two designs have been made over rule. The first design is explained in the appendix A on page 103. The final design is of the rule is explain later in this section. The main difference is that in the first design is an extra type of condition timestamp. But late into the implementation of MOM we found that it would not be needed and then we have made this design.

The rule's structure is presented in a grammar in listing A.1 expressed in Extended Backus-Naur Form(EBNF) [8]. In a nonterminal is en-captured in

<> and a terminal is just a word or en-captured in "". Also the grouping is used represented in (), the replica symbol is $*$, comments is $(**)$ and alternative is |.

A rule consist of a name and one of five action and condition sets which determine which actions and conditions can be combined. A rule can have several actions and conditions but only from the same set, see line 1-7. The action set is presented from 9-23 where all has a specific name, some have a specific Controller represented as a number and some has points which is a number. The condition set likewise represented from 9-23 and the condition types are presented from 25-30. There are Three types but they each have a specific name. One type has a Controller, another only the name 'True' and the last has a timeperiod. The timeperiod has with two timestamps and if it is repeatable it has a string representation of the weekdays and a representation of then it is repeatable.

Listing 5.1: Grammar of a rule in EBNF

```
1  <Rule>:= <name> (
2        (<ActionsetSet1><ConditionSet1>)*
3      | (<ActionsetSet2><ConditionSet2>)*
4      | (<ActionsetSet3><ConditionSet3>)*
5      | (<ActionsetSet4><ConditionSet4>)*
6      | (<ActionsetSet5><ConditionSet5>)*
7    )
8
9  <ActionsetSet1> := "Block user"
10 <ConditionSet1> := <ConditionTimeperiod> |<ConditionTrue>
11
12 <ActionsetSet2> := ("Add points" | "Delete points") <Points>
13 <ConditionSet2> := <ConditionTimeperiod>
14
15 <ActionsetSet3> := "Set maximum of point" <Points>
16 <ConditionSet3> := <ConditionTrue>
17
18 <ActionsetSet4> := "Unlimited time" | ("Access any ↩
       controller" | "Cannot access any controller") <↩
       Controller>
19 <ConditionSet4> := <ConditionTimeperiod> | <ConditionTrue>
20
21 <ActionsetSet5> := ("Access controller" | "Cannot access ↩
       controller")<Controller>
22 <ConditionSet5> :=  <ConditionTimeperiod> | <ConditionTrue> ↩
       | <ConditionElse>
23
24 <ConditionTimeperiod> :=  "TimePeriod" <Timestamp> <↩
       Timestamp> <Repeatable>
25 <Repeatable> :=   <Weekdays> <Repeat> | Nill
26
```

```
27 <ConditionTrue> := "True"
28 <ConditionElse>:= ("Device on" | "Device off") <Controller>
29
30 <Name> := ALPHA*  (* Upper and lowercase ASCII letters (A-Z,↩
       a-z) *)
31 <Controller> := DIGIT* (* Decimal digits (0-9) *)
32 <Points> := DIGIT*
33 <Timestamp> := 4*DIGIT,"-",2*DIGIT,"-",2*DIGIT, " ",2*DIGIT↩
       ,":"2*DIGIT,":"2*DIGIT  (*YYYY-MM-DD HH:mm:ss*)
34 <Weekdays> := ("monday"| "tuesday"| "wednesday"| "thursday"|↩
        "friday"| "saturday"| "sunday")*
35 <Repeat> := "weekly" | "biweekly" | "triweekly" | "first in ↩
       month" | "last in month"
```

5.2.1 Permission and Rules Precedence

It should be possible to override permissions and rules with another rule, but then some priority rules need to be established. For the overriding of the two to be relevant they need to overlap in time which mean that the condition should use Timeperiod or True. Also if it is a rule then it should have an action with the name 'Cannot access controller', 'Access controller', 'Cannot access any controller' or 'Access any controller' to be relevant. So from the grammar A.1 it is the rule sets $< name >< ActionsetSet4 >< ConditionSet4 >$ and $< name >< ActionsetSet5 >< ConditionSet5 >$ that need the priority.

We came to the conclusion that rules always have precedence over permissions. But if the conflict is among two rules it is a more complex set of precedence rules. First to determine the precedence we look at the action's name. See figure 5.1 where the precedence graph is shown. Notice that the precedence for 'Cannot access controller' and 'Access controller' can be either way depending on the condition. Otherwise both of them has priority over 'Access any controller' which also has priority over 'Cannot access any controller'.

When the condition determines the precedence we have different combinations:

- If one is True and the other is Timeperiod then the rule with the Timeperiod has the higher precedence
- If both is True then Access controller has the precedence.
- If both is Timeperiod:
  - If both is repeatable or non-repeatable then Access controller has the precedence
  - If one is repeatable and the other is nonrepeatable then the rule with the nonrepeatable Timeperiod has the higher precedence.

These precedence rules do not avoid all possible conflict but it limits them.

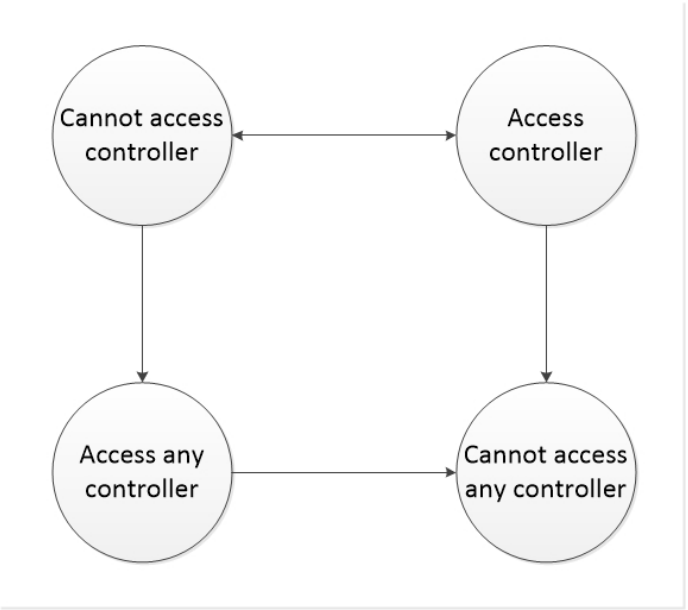

Figure 5.1: Precedence of rules

Examples of Rules

In this section some examples of rules for a profile will be given.

The first example could be that the profile Peter gets point for each football training and the football training is Monday and Thursday from 18:00 to 19:30 each week. Then the action could be add 25 points to Peter when condition holds. The condition is from 19:30 to 19:30 where the day is 'Monday, Thursday' and it is repeated weekly.

The second example could be that Susan is grounded from the 2nd December. In the system there will be a condition with Timestamp which is 2nd December 2013 18:00 and an action that is 'Block user' such that she cannot use any media or gain points in this period.

If the parents should make a rule which give themselves unlimited time and access to any media. The condition would be True and the actions would be 'Unlimited time' and 'Access any controller'.

This conclude the general concepts that will be used through out the remainder of the report. Next further details about the requirements of MOM is described.

5.3 Permission

Permission in this system is whether a user have access to a given media. Permissions can also be expressed as a rule, but it is easier to make and understand a permission. This is also the main reason for differentiating the two for the user.

A permission need a user and a controller. When a user then is connected to a permission it means this user has permission to use the media which is connected to this controller. An example could be that Peter has permission to use the TV.

A permission is different from a rule in two ways according to the data structure. First, a permission does not need a condition, it is always true. Secondly a permission is always of the action type ”Access Controller”.
This means that permissions is only an abstraction making an easier understanding for the user.

Chapter 6

Requirements of MOM

In the analysis we came to the conclusion that the following items is necessary for MOM:

- A device to toggle the power of an electronic media
- An individual key for each child that can be read by the device
- A website to set up rules, permissions and view/modify allowed usage of electronic media

The first item is called controller in this system, the second is the tag and the last is the website on the computer. But also new components of the system is necessary. As explained in the previous section a daemon is needed to check the relevant rules and perform the appropriate actions if the conditions holds. Also during the following sections we find that a web service is needed.

In the following sections the specification of the website, controller and tag, web service and daemon will be explained, before the design and implementation will be presented for each of them in the following chapters.

6.1 MOM's Website

A Media-Online Management (MOM) owner needs a way to manage the system and for this a website will be made. There are several requirements for this website:

- Register a new MOM together with a user profile which is a manager of this system
- Make it possible to add, edit and delete user profiles in an existing MOM
- Make it possible to add, edit and delete Controllers from an existing MOM
- There should be a way to add, edit and delete Tags to the system. Also a tag should be connected to a user profile

- There need to be an option to add, edit and delete rules, permissions and chores from a system
- The user should be able to connect rules and permission with one or more user profiles. The connection should also be able to be removed without removing the rule or permission
- When a chore is done in the real world by a child profile then by connecting this profile to a chore, points should be added to his profile

There are also some requirements that is nice for the parents to use, but they are not essential for MOM system:

- present the media consumption data as graphs or logs such that the parents easy can get an overview of their children's media consumption
- present data from MOM in a calendar that shows rule and permission with profiles for whom this is relevant. The calendar should also show when a chore have been done and by who.

In chapter 11 on page 75 the design and implementation of the website will be presented in more detail.

6.2 Tag and Controller

The tag used in this system is using Radio-frequency Identification (RFID). The tag need to be uniquely identified in MOM and it must uniquely identify its user. The tag is used in combination with the controller.

The controller is an Arduino which is connected to a tag reader. Like the tag it need to be uniquely identified in MOM. The controller must be able to do the following.

- Read the data from the tag
- Send and receive messages from the server
- Control the power source of the media
- Temporarily store the tag id that activated this controller
- Keep track of the time spent between activation of the media to its end

The design and implementation of the controller will be explained further in chapter 10 on page 61.

The controller must communicate with the server and this is done via a web service.

6.3 API

The API is used to parse the relevant data to the controller. Its requirements are:

- Receive and send messages to the controller
- From a tag and controller it should be able to determine whether a user may use the media which is connected to the controller
- It must be able to subtract points from a user after he has been using a media
- It should able to calculate when a controller should be turned off because of a rule, permission or point.

The web service's design and implementation is explained further in chapter 8 on page 53.

6.4 Daemon

The daemon is used to update data in the database from a rule which the user have made at some point. The daemon does not have any direct contact to a client. The daemon should handle any rule that is time determined so the condition is a Timeperiod. Then the daemon should do the appropriate action which could be add points to user or block a user.

The design and implementation of the daemon is presented in detail in chapter 9 on page 57.

6.5 System Architecture

Now that all subsystem of the Media-Online Management have been presented the overall system architecture can be explained. MOM is using the client server architecture which is shown in the figure 6.1 [6]. This system has two types of clients the first is a controller and the second is a computer where the user manages the system via the web site. The controller is depended on the web service on the server and the computer is depended on the web site. On the server there is the web site, web service and daemon which all are depended on the functions component, MySQLHelper class and data component. In the function component there are several functions to work on the data in the database and this component is depended on MySQLHelper because it should establish the database connection and parse the quires on to the database.

In the following chapters design and implementation of the web site, web service, deamon and database will be presented.

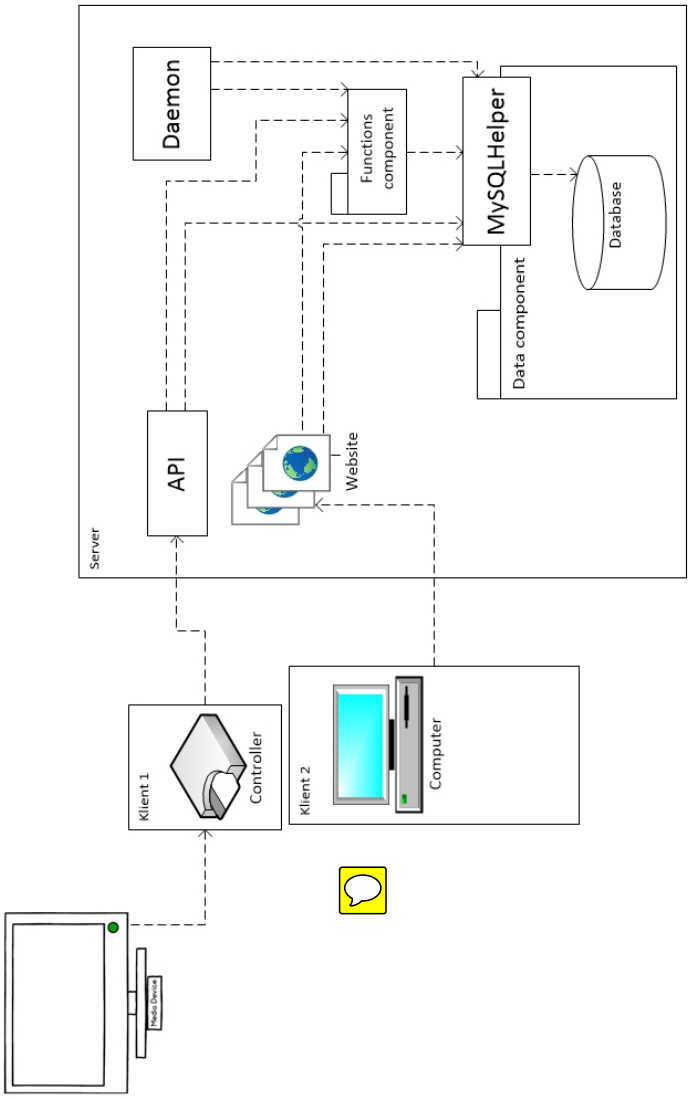

Figure 6.1: system architecture

Chapter 7

The Database

In this chapter the database design and implementation are explained. We will presents the ER-shema with our design and explain some of the design decision. Then we will shown the database scheme created from the implementation and explain some of the implementation decisions there have been made. We will shown code samples from the function that update the simpler objects and finally there will be code from updating a rule.

7.1 Design

In the parent control system there are seven essential objects; control system, profile, tag, controller, chore, rule and permission.

Control system is identifying the individual system and all of the other objects are connected to a control system.

Profile is a representation of a specific person in the control system. The person can both be a child or the parent, but this should be distinguishable.

Tag identifies the physical tag that a profile uses to activate the controllers. A tag is identified by the serial number of the physical tag.

Controller is a object representation of the physical controller and like the tag it is identified by the serial number of the physical controller.

Chore is a representation of a house chore which can be carried out by a child (profile) which then gets point as a reward.

Permission is representing time restrictions on the controllers which some of the profiles need to abide by.

Rule is representing other restriction or rules that cannot be expressed by the permission. A rule consist of some conditions that need to meet before some actions should be taken as explained previously in 5.2 on page 33.

From the former description the database design is made. It is represented in a ER-diagram in figure 7.1 where the cardinality ratio is expressed in Chen notation[5]. It should be mentioned that the design of a rule in the database is based on the first design of the rules, which can be found in appendix A on page 103. The reason for not using the final design is that it was changed fare into the implementation phase and the first database design does still support the final design of rule.

In the ER-diagram it should be noticed that Permission is absent. The reason is for every permission that could be made then there is an equivalent rule. Therefore the data representation of Permission is the same as Rule. However, on the website the two should be represented differently and so the attribute `isPermission` have been added to Rule which will differentiate the two.

Another observation from the diagram is the condition of a rule. The condition can be one of three representations. The first is the simplest because it is just the Condition. The second is Condition and a timestamp which is the `cond_timestamp` in the design in 7.1. The last representation is Condition, a time period, and a representation of when it is valid, which in the ER-diagram is `cond_timeperiod`. The type of condition that should be used, can be distinguished by the condition name. If the name is `Timestamp` it should be cond_timestamp, if it is `Timeperiode` then it is cond_timeperiod and any other name it is only the Condition.

Notice if the database design had been design from the final design then the table `cond_timestamp` would not have been in the design. After the database design has been completed it was implemented into a MySQL database.

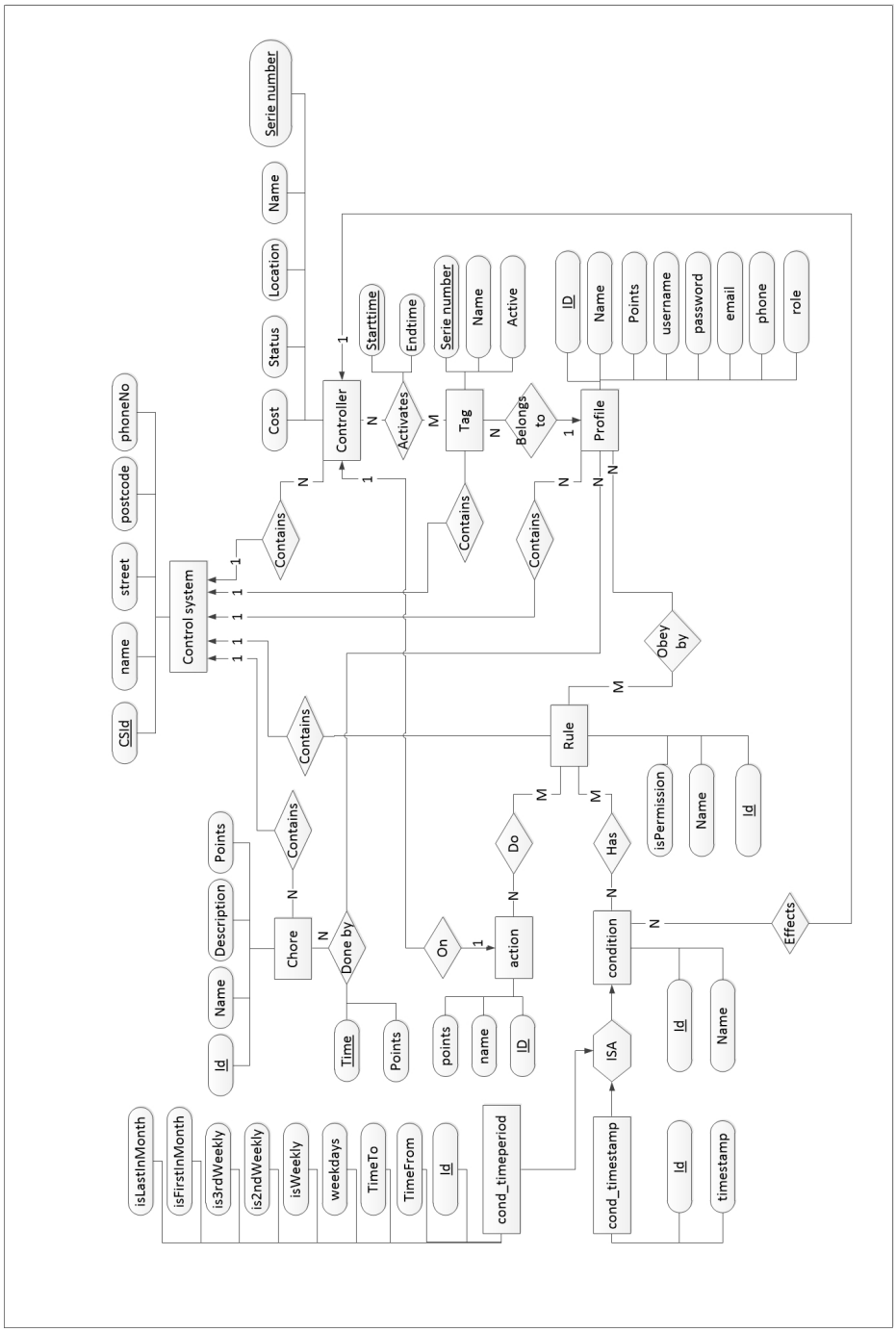

Figure 7.1: ER schema of the database design

7.2 Implementation

The database design has been implemented in a MySQL database and the result can be found in figure 7.2. In the implementation of the database redundancy and anomalies is to be avoided and null values reduced. The mapping from the ER-diagram to the relation representation in the database is a basic mapping according to the relations $1:1$, $n:1$, $n:m$ except for Condition which is clarified in the following section. [5]

7.2.1 Mapping of Condition

The relational modeling of Condition is more advanced because it uses generalization and as such there are a implementation choice that should be made. There are four typical method of how to deal with generalization and a solution of using each method is described below.[5]

Using main classes, then Condition is split up into three tables rcondition, cond_timestamp and cond_timeperiod where all attributes in Condition are repeated in each table. An disadvantage of using this method is in querying because it is possible to go though all three before finding the wanted data.

Using partitioning, there are three tables rcondition, cond_timestamp and cond_timeperiod. The common attributes are in Condition and via an id the possible additional data can be found in either cond_timestamp or cond_timeperiod.

Using full redundancy, it is much like the first, but all conditions from cond_timestamp and cond_timeperiod are also represented in Condition, without the additional data. As the name suggest this causes redundancy which is best to be avoided.

Using a single relation, such that there will be a single table with all attributes from Condition, cond_timestamp and cond_timeperiod in the ER-diagram. It is easy to use but it will increase the number of null values.

As can be see in the figure 7.2 the partitioning option is used in our implementation since it does not causes any redundancy or null values and it is easier to search for conditions compared to using the main classes. However, using partitioning there will be some challenges in making the functions to insert, update and get data about the rules, as will be explained in section 7.2.4 on page 49.

There have also been made an important decision about how to deal with deletion of data from the essential tables and this is explained in the following section.

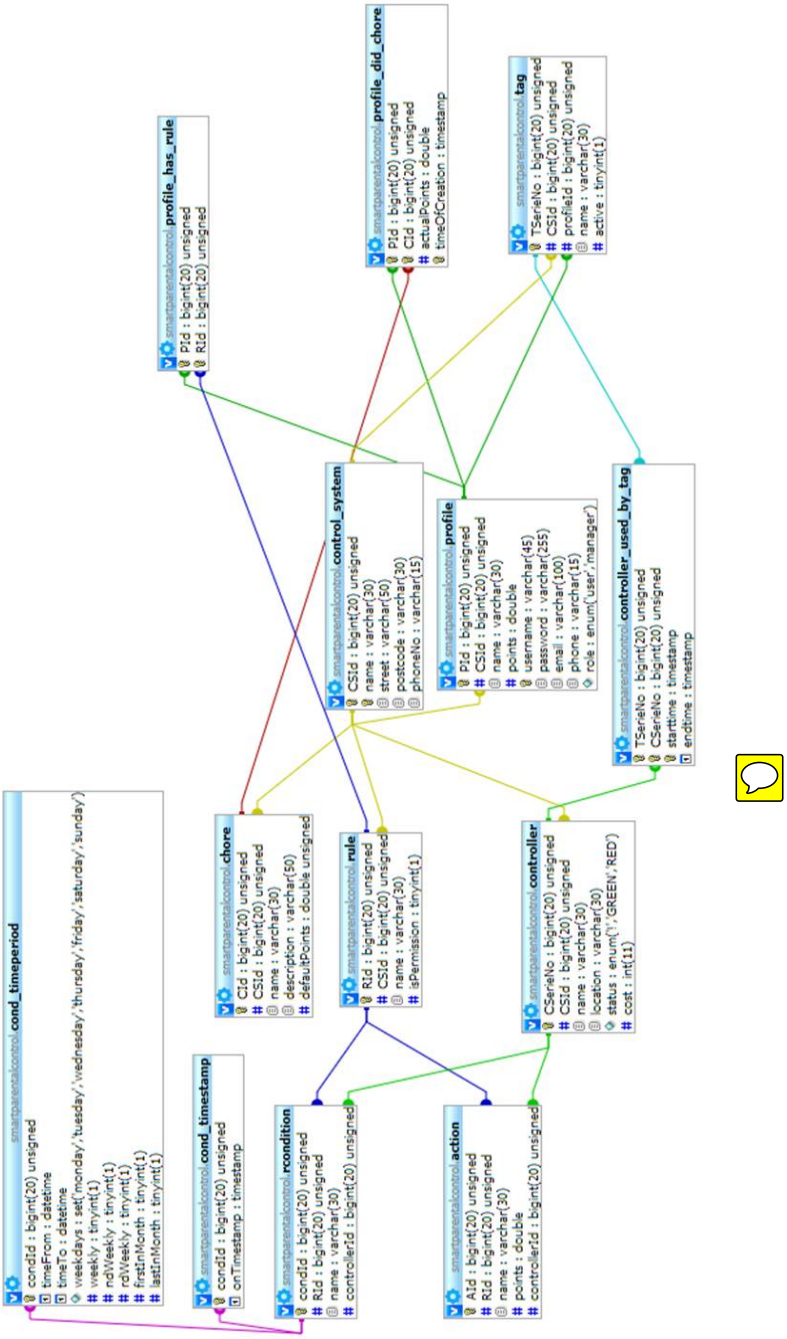

Figure 7.2: The Database implementation

7.2.2 Deleting in the database

When a tuple in one of the tables control_system, profile, tag, controller, chore, and rule should be deleted it will likely affect data from another table because of the foreign keys. We have made an intentionally decision to use `ON DELETE CASCASED` with all foreign keys, because in any case when deleting a tuple from the essential tables its history is irrelevant and should be deleted.

As an consequence of this decision if a control system is deleted then every profile, tag, controller, chore, and rule that is connected to control system will be deleted. If a profile is deleted then this profile's tags will be deleted including all of its history, such as when the tag have been used to activate the controllers, or when the user have done a chore. So it is very important that when data are to be deleted then it would not be regretted later.

However, for anything to be added, updated or deleted by the user the website need a connection to the database which is done through the class MySQLHelper.

7.2.3 MySQLHelper class

The website and API for the controllers connect to the database by using a PHP class MySQLHelper, which establish a connection and send queries to it. The class have a construct and destruct that respectively establish the connection and close the connection to the database. The queries goes also through this class in at least one of these method:

insertInto this make a SQL string to insert into the database.

update this make a SQL string to update data in the database.

delete this make a SQL string to delete data in the database.

query this make a SQL query string.

executeSQL this send the query to the database and return the result. All of the above uses this method.

It is also possible to control the transactions through this class by disabling the auto-commit of transactions and then manually commit when it is necessary.

This class is used by several functions that are connected to the website, API and Daemon in the following section we present the more interesting of them how to update rules.

7.2.4 EditRule from the Function Library

This system also have a function library which get, insert, update or delete data from the database. Among these function are e.g. `simpleInsertIntoDB` and `simpleUpdateDB` that is used to insert and update the essential tables with the exception of Rule. Insert and updating a rule is similar but the latter is the more interesting so it will be this sections focus.

As stated previously in mapping of a rule in 7.2.1 on page 46, there would be some challenges to make the insert and update functions. This is partly due to the fact that the data of a condition is one of two alternative: condition, condition plus timestamp or condition plus timeperiod as was explained in 7.1 on page 43. However, in the newer design of a rule the condition plus timestamp is removed which makes it easier but the problem still exist. In update it is also a challenge that each condition and action can be removed, add or updated which all should be handled together and this will be explained further in this section.

The function to update a rule is called editRule and it also used some other function as described below:

addAction from a rule id and a object of type Action it adds an action to the rule.

editAction from a object of type Action it updates the attributes in the database.

addCondition from a rule id and a object of type Condition adds a condition to the rule.

editcondition from a object of type Condition it updates the attributes in the database.

The editRule has three inputs a Rule object, an array with Condition objects, and an array with Action objects. First the rule data will be update and then each condition and action must also be updated.

However, when updating an action and especially a condition the changes can have different characteristics as listed below. This is also in the decision diagram in figure 7.3.

- a new condition or action is add.
- a condition or action need to be deleted.
- an existing condition or action is to be updated with no structural change in the database.
- an existing condition or action is to be updated but with a structural change in the database.

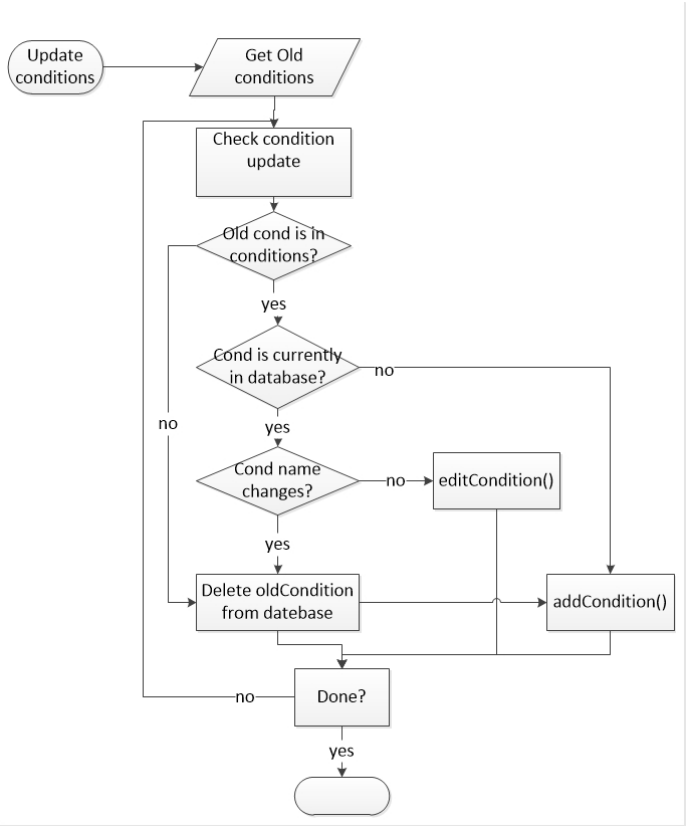

Figure 7.3: updateRule

With a structural change it is meant that the name of the condition or action has been changes and this change can influence e.g. whether cond_timeperiod or cond_timestamp should be used. As a result it have been decided that in this case the old condition should be deleted and the new be added anew. A structural change should not happen for an action, but the function are still able to handle it in the same way as a condition.

Status on Database and Function Library

The database and the functions mentioned have all been made prior to when the decision of the rule's design changed. The functions which uses condition can now be further improved but this will not be done because it is not a priority since these functions still meet the requirements of the system.

Up till now the rules could have several condition and action but late in the implementation of the Website we have decides to only allow one action and condition per rule to get a more user-friendly website, more about this decision

can be found in section 11.2.2 on page 79. Most functions are still implemented on the first design so they can handle several conditions and actions.

However, a few function can only work with one condition and action per rules. An example is the function that checks whether a user can use a media, called *db_rules_user_can_turn_device_on*. If the implementation of rules on the website is change a new version of these functions need to be made.

Chapter 8

The API

This chapter will explain the design and implementation of the API for this project.

8.1 Design

The API started with one clear goal, to be the middleman between the controllers and our database. The API's main purpose is to abstract away the need for the controllers to work directly with the database, along with it meaning more security for the database. With this in mind the API should also be as simple as possible, which lead to the API having only three actions. The three actions are `status`, `turn on` and `turn off`.
`Status`' job is a way for the controller to check up on if there have been any changes in time remaining for the user. This means that `status` has to check if there are any rules that will cut the user off before they run out of points. If there are no rules that interferes then it should calculate the time remaining again for the user and return that.
`Turn on`'s main function is for the controller to request for it to turn on. This means that `turn on` should check to see if the controller can be turned on, calculate the time remaining, turn on the controller inside the database and finally return that. When checking to see if the controller can be turned on, `turn on` should check both that the user has enough points for at least a minute of on time and that there are no rules that are preventing the controller from turning on.
`Turn off`'s main function is for the controller to tell the API that it wants to turn off. This means that `turn off` should check if it is the same user who is turning the controller off and if it is not the same user report an error. `turn off` should then calculate the time spent on the media and calculate how many points to remove from the user and then remove it. Finally it should return the time spent and the points removed.
Figure 8.1 shows the flow of how the API checks if a controller can turn on.

First it checks if the controller id and tag id is valid. It then checks if any rules or permissions prevent the controller from turning on or user from turning on the controller. Then it checks if the device is already off and if the user has enough points to turn the controller on. If any of the checks are false an error is returned, otherwise it turns on the controller in the database and returns the result to the controller.

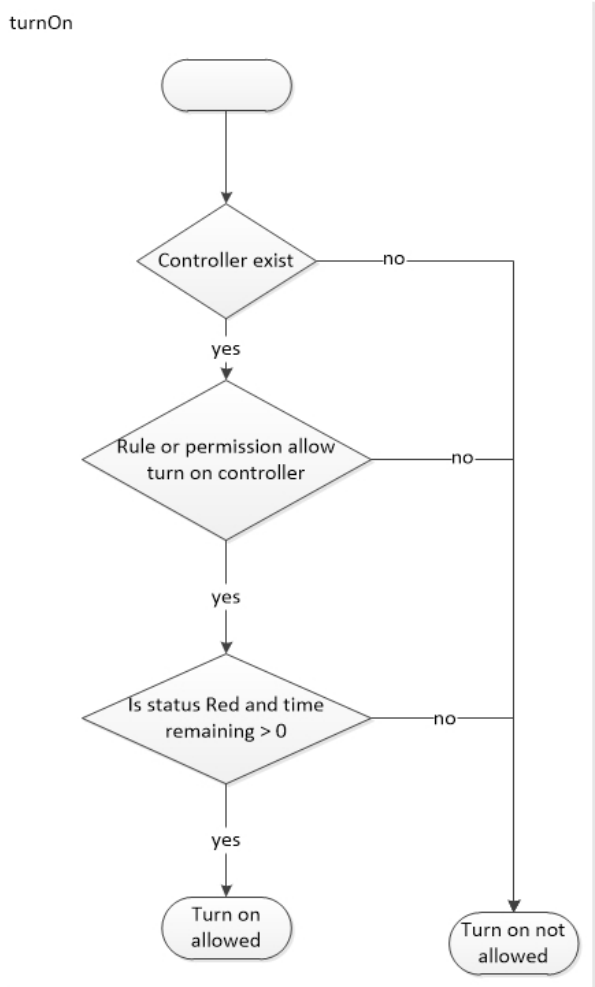

Figure 8.1: Flowchart showing how turn on works

8.2 Implementation

The API has been implemented on top of the Slim routing framework and PHP. The reason for this framework is that it makes it quicker and easier to make an API, while also making it easy to add additional endpoints. The communication between controllers and the API are done using HTTP and JSON. JSON has been chosen because of its almost universally support and its small overhead. This means that the communication is easy to implement, and easy to port to additional platforms if necessary. Calling the API is as simple as navigating to a website. Parameters to the API is done using custom GET messages. An example of a call to get status could be `http://example.com/api/api.php/turnOn/123/234` where `example.com` is the domain where the API is hosted, `123` is the controller id and `234` is the users tag id. An overview of the available calls and their parameters can be seen in Table 8.1.

Call	Parameters	Return values
/status/	cId	status, action, timeRemaining
/turnOn/	cId, tId	status, timeRemaining, error
/turnOff/	cId, tId	status, timeSpent, pointsRemoved, error

Table 8.1: A table containing the available calls of the API. A legend of the keywords can be found in Table 8.2

Keyword	Description
cId	The id of a controller
tId	The id of a users tag
status	Status of the API call, `OK` if successful, `ERROR` if failure
action	The action the controller should take, `none` if no action should be taken
error	A description of the error that has occurred if any has, error is omitted from the JSON if there was no errors
timeRemaining	The amount of time before the controller should turn itself off in minutes
timeSpent	The amount of time the controller was one in minutes
pointsRemoved	The amount of points removed from the user

Table 8.2: A table containing a legend of the keywords used in Table 8.1

Chapter 9

The Daemon

This chapter will explain the design and implementation of the daemon of this project. A daemon is a computer program that runs in the background continuously without any input from a user.

9.1 Design

The daemon has a one main job, to activate rules that require action at a specific or recurring time. Right now there is only one such action, `Add points`, which when activated has to add points to a given users profile. This cannot be done using a normal website because of the time requirement that the points have to be added at the right time. Further things that the daemon should handle could be such things as `Activate tag` and `Block tag` if these were added as actions to the rule system. Figure 9.1 shows how the flow of the daemon should work. It should check rules every minute if they should activate, and activate those who should, doing the action specified.

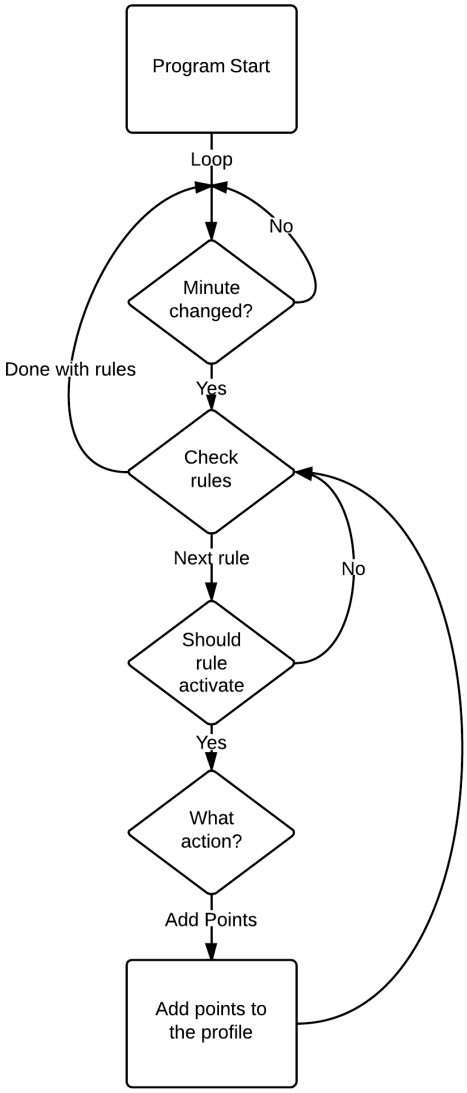

Figure 9.1: Flowchart showing how the daemon should work

9.2 Implementation

Because daemons usually run all of the time, it is not appropriate to be implemented in PHP, as PHP is not meant to run continuously. The choice has then fallen on Python as the programming language. The reason for this choice is that Python is a dynamic language like PHP that is quick to build prototypes

in. Unlike PHP, Python is meant to run continuously while PHP is meant to run and exit.
The daemon like the website communicates with the database directly. The daemon is set up to check rules that have actions, conditions and profiles attached. It checks if these rules should activate based on their `time from` and `time to` timestamps as well as their repeatability options. There are four ways a time period condition attached to a rule can activate based on weekday, week, last/first in month and time stamp. Weekday is activated when the time period has weekdays selected but not any repeatability options enabled, i.e. the current weekday should match one of the selected ones and the current date and time should fall between `time from` and `time to`. Week activates based on the difference between the `time from` date and time, and the current date and time. This means that the absolute week is the one used to calculate weekly, biweekly and triweekly. Last/first in month is based on weekday, if the weekday selected is the current day then the condition activates if the current day is the first/last monday/tuesday/.. of the month. Lastly the condition activates if the current time and date is between `time from` and `time to` and the condition has no repeatability enabled.
The daemon checks rules every minute, by keeping check on the last minute the rules was checked at, and if it has changed, then it checks the rules. This method was favored instead of a simple sleep for 60 seconds as there were possibilities where the daemon would not check a minute, e.g. if the daemon started checking rules 40 milliseconds before a new minute, the daemon would take 50 milliseconds to check rules, then sleep for 60 seconds, this would make the daemon skip a check on a minute which could make conditions not activate when they should.

Chapter 10

Design of the Controller

This chapter is focused on the controller system of the Media-Online Management system. The controller system is the hand and mouth of the MOM system. Handling the every day use of MOM system as activating and restricting users from media devices.
This chapter is focused on the controller of the Media-Online Management system. The controller consist of a tag reader and power control device which is disgusted in this chapter but when mentioning controller both is implied.
The design and functionality of the prototype controller will be presented in parallel to the designed end product.

10.1 Product Design

The product design section is the vision of the system as a finised product. Figure 10.1 is the illustation of the embedded system called the controller system. The controller system is connected to the server though a wifi connection to insure that a ethernet outlet don't need to be a every media devise. The controller is two devices, detection device and power control device.

Detection device reads user tags when swiped over the detector device. This device is separated from the power control device to increase user friendliness. The power control device will be placed on the power cable to the media device will that not always be as accessible for the use if one want to scan a tag. Therefor shall the detection device often be placed infront of the media device to give the user friendliness.
The other device is a powercontrol device which is located on the powerline to the media device and turns on and off the electricity when needed.
The powercontrol device(PCD) is the brain of the two devices and is therefore to monitor the time usage of the media. It should also communicate with the server and detection device.
The detection device(DD) should also inform the user who is trying to activate

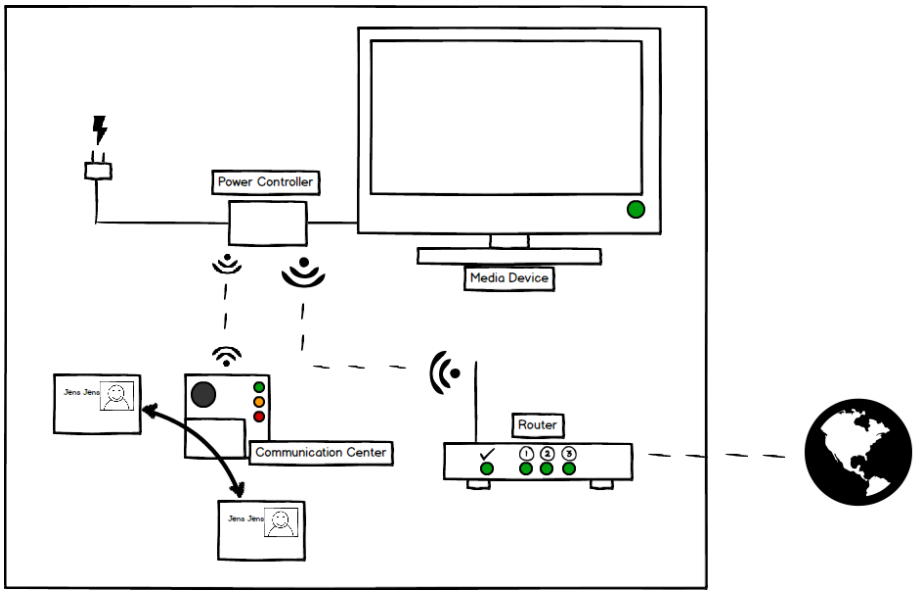

Figure 10.1: Rich picture for Powercontrol and Tag reader

a media of whether the activation has been approved or declined by the server.

10.1.1 Senario Design

The system has been designed on multiple scenarios and the criteria of communication to the server and user. The flow chart **??** is an overview of the action and communication between server, the controller system and user. The flow chart illustrat the controller system:

- Register user tags and get user rights from server.
- Adminstrat user rigths according to server information.
- Moniter the connection wifi to server and handele the exseption op disconnected.
- Regulate the power to a device recording to senario.
- Manage the time usaged and user restigtions.
- Communicat system status to user.

indsæt Flowcart af systemet

The development was done by asking and answer questions to the system in different senarios.
The flow chart is a combined answer here of.

Setting up and turning on the system.
In the final design the PCD will receive the information about the WiFi connection from USB drive. If the information on the USB is wrong or not received then a restart button is included to redo the procedure.
The PCD will then connect to the WiFi network and add the controller to the database in his MOM system. If the connection is not found a message is send to the DD.
DD indicates that there is no connection by showing a constant red light. The communication protocol can be found in the appendix B on page 107.
When the connection is established then the DD is ready to read user tags.

However, this design will not be implemented in this project. Instead there is a wired connection to the internet and the controller need to be manually added into the database.

Detection of connection issues and status changes.
The connection to the sever is tested by a routine call every five minutes from the PCD to the server. In this call the PCD will receive information on when the media should shutdown based on a rule or the user's remaining time. This insures that if any changes in the remaining time or the rules which will influences when the media should shut down then it will still be handled. In between those calls the PCD will check for a tag from the DD. If the PCD is disconnected from the server that is informed to the DD and to the user.

Logon/logoff media divice.
When a tag is read by the DD the PCD will receive the tag id which is then formed into a log-in message and send to the web servers API. however this is only if no one ready is logged on the media device in the MOM system. The PCD will then get a message about whether the user has permission to use the media from the API. If the user has permission then the PCD should direct power to the media, then save the tag id in local memory and receive the remaining time from the API. The DD should indicate whether the user is approved to use the media.
To log-off the user will swipe ones user tag again and the PCD receive the same tag id that is stored in the local memory. The PCD will then switch off the power to the media and send a log-off message to the API.
If another user wants to take over the usage of the media then the PCD should try to log-in with the new tag. If this is declined then the DD should then signal this and keep the old user logged on. If it is accepted then the server should overwrite the previous user and PCD will receive the information for the new user with out turning the power off and then on again.
The functionality of switch users is not implementet in the current prototype.

Server disconnection under use.
A disconnection from the server is either found under the regular status call after each five minutes or at a log-on/log-off call. A disconnection will lock the device so only the current user can use the media until the PCD is reconnected

with the server. The server discovers the lack of connection when there have not been a status call from the device in five minutes. The disconnection is translated to a log-off the media at the server side. The user will not be logged off by the controller but will be able to use his time according to the last status. If the user want to log-off in a period where there is a disconnection a time stamp is saved and will be sent to the server when the connection is reestablished. The media can also first be used again when the connection is reestablished. This procedure have the advantage that the user will not have to cut off the media in use if the connection is unstable. The user will also be able to use a another media with connection to the server. The disadvantage is that the user has the possible to exceed the time restriction if there has been a change in the rules or the user's remaining time.
The implemtation of exseption of disconnection in prototype is not as the descriped above.

10.2 Prototype Implemtation

The prototype is designed to prove the concept of the requirements to direct the power to and away from the media, control a tag reader, communicate with the server and it need to be a Embedded system.
To prove the requirements the system the to be able to:

- Read and accept/decline tags in communication with a server though a online connection.
- Have a good system structure monitor the server connection.
- Regulate the power recording to the scenario.
- Manage the time usage and user restrictions.
- Communicate system status to user though LED light indication.

The Arduino hardware platform is made the base on the prototype as analyzed in section 4.1. To make a functional prototype the Arduino Uno became equipped with a Ethernet port. To the Arduino Uno is a RFID shield and breadboard connected. First a deeper look into the structure of the controller is shown. Then some code to shown the communication with the RFID shield and Ethernet port is documented to give better knowledge how they work.

10.2.1 ArduinoDesign.

Indsæt flowchart

Flowchart is the current level of the prototype as is very similar to figure 10.1.

Therefore the Ardurino hardware should be seen as replaceable by another controller device in this system.

10.2.2 Programming Basics for Arduino

An Arduino code file is referred to as a Sketch. At is very core the Arduino sketch has two components, `setup()` and `loop()`, which are filled by the programmer to apply functionality. The `setup()` method is called on start up only to initialize the Single-board micro-controller, while the `loop()` method runs repeatedly, making the Arduino perform its designated task.

Implementation of Setup() And Loop()

In our `setup()` we initialize The Serial Communication, some global variables, connect the Ethernet shield to the internet along with configuring 3 pins needed for RFID and LED communication.
The Arduino has 3 states which is used to dictate what happens in `loop()`. These states are:

- State 0: "No User Logged in". The arduino is looking for a user to log in.
- State 1: "User Logged in". The Arduino is waiting for a user to log out.
- State 2: "Connection Error trying to `turnOff()`". A temporary solution to avoid the complication of a user trying to log into a device before the previous had been logged out properly.

Depending on what state the Arduino, there are 3 important method calls that is either invoked by the user or called automatically as the `loop()` runs, these are:

- `getStatus()` retrieves the server's view of the device on whether its supposed to be in use or off, it updates the users remaining time if its the prior and removes access by changing from state 1 to 0 when it is the latter.
- `turnOn()` checks if the user is allowed to turn on the device. If he is it turns the device from state 0 to state 1. However, if he is not or the Arduino could not connect to the server, it removes the user trying to log in and remains in state 0.
- `turnOff()` turns off the device by first informing the server of this and then actively turns off the device by removing the user ID, turning off the LED and changing from state 1 to 0. `turnOff()`, unlike the other two, does not receive feedback from the server. However, if there is an issue connecting to the server to log off, the Arduino will change to state 2, repeatedly trying to log off until it successfully gets a connection.
- The way all 3 methods connect to the server and parse the response is similar and will be described in **??**.

However, before the Arduino checks for any user interaction with the `loop()`, it performs two checks. The first check is named `checkConnection()`, this method evaluates the native method `Ethernet.Maintain()`. If needed, `Ethernet.Maintain()` tries to renew the DHCP lease and reports back the result.
The second check is named `CheckTimeStatus`. Because the servers require a frequent `getStatus()` to contact the status API, this check assures that one is performed approximately every 5 minutes unless the arduino is in state 2. `CheckTimeStatus` will be explained detail at **??**.

lisbeth - I think this listing can be removed without losing anything

Listing 10.1: The Arduino loop.

```
1  void loop()
2  {
3    checkConnection();
4    delay(10);
5    if(checkTimeStatus())
6    {
7      switch(state)
8      {
9        case 0:
10                 getStatus();
11                 break;
12       case 1:
13                 getStatus();
14                 break;
15       case 2:
16                 break;
17     }
18   }
19   switch (state)
20   {
21     case 0: Serial.println(F("State 0"));
22             stateZero();
23             break;
24     case 1: Serial.println(F("State 1"));
25             if(checkNoTimeLeft())
26             {
27               turnOff();
28               break;
29             }
30             else
31             {
32               stateOne();
33               break;
34             }
35     case 2: Serial.println(F("State 2"));
36             turnOff();
```

```
37              if(state == 0)
38              {
39                getStatus();
40              }
41              break;
42    }
43  }
```

As seen in Listing 10.1 there are two methods that depends on the state. `stateOne()` and `stateZero()`. They are extremely similar, differing only in the need to compare the tag ID in use with the one being swiped which occurs in `stateOne()` which can be seen in Listing 10.2. The Code block shows a series of method calls related to the RFID functionality which is described in **??**.
To a point, the code finds, authenticates and reads an RFID tag. If the tag has the same ID as the one already used to log in, the Arduino will log him out. However, if it is a new tag, the old tag will be logged out, and the new tag logged in.

Listing 10.2: The Arduino state one code.

```
1   seek();
2   delay(10);
3   parse_response(rf_block_response, seek_length);
4   delay(10);
5   if(rf_block_response[2] == 6)
6   {
7     Serial.println(F("Authenticating."));
8     authenticate();
9     parse_response(rf_block_response, authenticate_length);
10    delay(10);
11    if(rf_block_response[4] == 0x4C)
12    {
13      Serial.println(F("Reading."));
14      read_block_RFID();
15      delay(10);
16      parse_response(rf_block_response, block_length);
17      delay(10);
18      if(rf_block_response[2] == 0x12)
19      {
20        char tempID[4];
21        Serial.println(F("Read Successfull:  "));
22        for(int i = 5; i < 8; i++) //TODO: Length of ID.
23        {
24          tempID[i-5] = rf_block_response[i];
25        }
26
27        if(tempID == useID)
28        {
29          turnOff();
```

```
30            strcpy(useID,"");
31          }
32          else
33          {
34            turnOff();
35            strcpy(useID,tempID);
36            turnOn();
37          }
38          delay(10);
39          Serial.println(F("Stop"));
40        }
41        else{
42          Serial.println(F("Read Failed"));
43        }
44      }
45      else
46      {
47        Serial.println(F("Authentication failed"));
48      }
49    }
50    else
51    {
52      for(int i=1;i<11;i++)
53      {
54        Serial.println(rf_block_response[i], HEX);
55      }
56
57      Serial.println(F("Wait for it"));
58      delay(2500);
59    }
```

10.2.3 Using the Ethernet Shield.

Mangler kort kommentar om hvorfor Ethernat er brugt istedet for en Wifi connection
The Ethernet shield is the primary component in accessing the API as it allows for the Arduino to access the web as a client. We use the Ethernet shield along with a JSON parser in order to make HTML requests on the web site that contains the system API and decode its responce.

The Ethernet Shield comes with an MAC address printed on the side which is used to automatically obtain an IP Address. This along with the target server is established as global variables as seen in Listing 10.3 along with the Method call in `Setup()` that establishes a connection with the LAN. All supplied from the Arduino Library `Ethernet.h`. kunne være korte statement på sær funktioner for eksemple F står for n
linje 22, Hvad sker der så, er det bare en død Audrino?

Listing 10.3: The Method Calls used to establish and maintain a connection.

```
1  //The Mac Address declared.
2  byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xC5, 0x94 };
3
4  //Our target Server that the API is called on
5  char server[] = "spcadmin.tk";
6
7  //When this statement is evaluated the Arduino connects to
       the LAN.
8  //This happens during Setup();
9  if (Ethernet.begin(mac) == 0)
10 {
11   //If the Evaluation returns 0 it fails and stops doing
         anything.
12   Serial.println(F("Failed to configure Ethernet using DHCP
         "));
13   while(true);
14 }
15
16 //The following code us run in every loop through
       checkConnection() call to assure
17 //that the DHCP lease is renewed when needed.
18 Ethernet.maintain();
```

Whether the user tries to log in or out from the device, the Arduino will have to access the web server via the API. The process is almost identical in either case and can be divided into two parts:

- Connecting to the API.
- Parsing the servers Reply.

As seen in Listing 10.4 the Device ID and the Tag ID is connected in a char array to form the url which complies with the action that is being performed. connect this to the API implmentation Then in return the Ethernet Shield receives the HTML Page, and in Listing 10.5 the Arduino reads this information one byte at the time where all but the JSON string is stripped and stored in a char pointer. When all the bytes have been through this progress the connection is closed and the JSON string is parsed the result of the action. This can be seen in Listing 10.6.

Listing 10.4: Connecting to the Server and creating an HTML request.

```
1  void turnOn(void)
2  {
3    if(client.connect(server, 80))
4    {
5      Serial.println(F("Connected"));
6
7      client.print(F("GET /api/api.php/turnOn/"));
```

```
8     client.print(devID);
9     client.print(F("/"));
10    client.print(useID);
11    client.println(F(" HTTP/1.1"));
12    client.println(F("Host: spcadmin.tk"));
13    client.println(F("Connection: close"));
14    client.println();
15
16    Serial.println(F("Message Sent"));
```

Listing 10.5: Removing all but the important information from the website.

```
1  while (client.connected())
2  {
3    //Builds the JSON string from the data passed by the website.
4    while(client.available())
5    {
6      buff = client.read();   //Bytes are passed through the Ethernet Shield with client.Read();
7      if(buff == '{')         //The JSON string starts with '{' and stops with '}'.
8      {
9        toggle = !toggle;
10       *(on+i) = buff;
11       i++;
12     }
13     else if(buff == '}')
14     {
15       toggle = !toggle;
16       *(on+i) = buff;
17       i++;
18     }
19     else if(toggle)
20     {
21       *(on+i) = buff;
22       i++;
23     }
24   }
25 }
26 client.stop();
```

Listing 10.6: The JSON Code Getting a value with a Token.

```
1 aJsonObject* json = aJson.parse(on);
2 aJsonObject* statJson = aJson.getObjectItem(json,"status");
3 char * stat = statJson->valuestring;
```

10.2.4 Using RFID With The Arduino

Mangler lidt mere start som der var i Using the ethernet shield. The Arduino board utilizes the SM130 by sending it commands and receiving replies through Digital Pins 7 and 8. These pins are initialized using the library `SoftwareSerial.h`.

```
1 SoftwareSerial rfid(7, 8); //Sets up the Digital Pins 7 and ↩
    8 '
2                            //Which the RFID reader ↩
                                Communicates through.
3
4 rfid.begin(19200); //This Method is called in the Setup() ↩
    phase
5                    //of the Arduino Startup.
6                    //It starts the Communication on the ↩
                        standard baud
7                    //rate of 19200 bps, N, 8, 1.
```

enten skal vi forklare hvad "'baud"' og hvad 19200 bps, N, 8, 1 står for.

These messages are a series of bytes sent one at the time in the form of Hexadecimals and in the context of an UART framed message.

Header:	1 Byte	Must always be 0xFF.
Reserved:	1 Byte	Must always be 0x00.
Message Length:	1 Byte	The amount of bytes used on Command and Data.
Command:	1 Byte	The Hexadecimal value of the command in question.
Data:	n Bytes	The bytes containing the data needed for the command.
Checksum:	1 Byte	The combined value of all above hexadecimals.

The command messages needed we need to allow logging in with a tag are:

- Seek: The Module activates 'anti-collision' and 'Select Tag' so that it can catch a tag when it comes in range.
  - Message Length:Seek has a message length of 1 byte as there is no accompanying data.
  - Command: 0x82 is the Seek Command.
- Authenticate: When a tag has been found this command authenticates a single block of data on the RFID Tag, which must be done before it is read.
  - Message Length: The authenticate command message is 3 bytes long as it needs 2 databytes.
  - Command: 0x85 is the Authenticate Command.
  - Data Byte 1 (Block Number): Tells the SM130 which data block on the tag is to be authenticated.

  - Data Byte 2 (Key Type): Informs which type of key should be used to authenticate with. The SM130 has 2 Key types to authenticate with (A and B), and 15 slots for each to store individual keys. We use the build in 0xFF which authenticates with Type A and the transport key "FF FF FF FF FF FF".

- Read Block: Reads a previously Authenticated block of data on the RFID Tag, The data contains the Tag ID.
  - Message Length: The Read Block Command message is 2 bytes long as it needs information on what block is to be read.
  - Command: 0x86 is the Read Block Command.
  - Data Byte (Block Number): Tells the SM130 which data block on the tag is to be read.

As an example Listing 10.7 is the code run when sending the Read Block Command to the SM130.

Listing 10.7: Example of the Read Block Command from the code point of view.

```
1  void read_block_RFID(void)
2  {
3    //Read Block Command in UART, sent to the SM130. The Block↩
         needs to be Authenticated beforehand.
4    rfid.write((uint8_t)0xFF); //Header: 1 byte, must always ↩
       be 0xFF.
5    rfid.write((uint8_t)0x00); //Reserved: 1 byte, must always↩
         be 0x00.
6    rfid.write((uint8_t)0x02); //Message Length: 1 byte, both ↩
       for Command and Data (Here: 2 bytes).
7    rfid.write((uint8_t)0x86); //Command: 1 byte, 0x86 is the ↩
       Read Block Command
8    rfid.write((uint8_t)0x01); //Data(Block Number): 1 byte, ↩
       read block no 0x01.
9    rfid.write((uint8_t)0x89); //The Message Checksum.
10   delay(10);
11 }
```

On being given a command the SM130 will send a response from which you can tell if the command has been successfully performed or not. In the code this information is saved on a char array by the parse method seen in Listing 10.8. On 'Seek' and 'Read Block' we determine the success by reading the slot containing the length of the Message, while on the Authenticate command the Data slot is read to determine success.

Listing 10.8: The method used to read the SM130's response.

```
1  void read_block_RFID(void)
```

```
2  {
3  void parse_response(char PH[], int length)
4  {
5    for(int i=1;i<length;i++)
6    {
7      PH[i] = 0;
8    }
9
10   while(rfid.available()) //This whileloop runs so long as ↩
         there is still bytes to be read.
11   {
12     if(rfid.read() == 255) //Checks for the Message Header.
13     {
14       for(int i=1;i<length;i++)
15       {
16         PH[i]= rfid.read(); //For the Length of the expected↩
               UART message, Add bytes to the Array.
17       }
18     }
19   }
20 }
21 }
```

the table with byte code is moved to appendix ref is append:bytecode

Chapter 11

Admin Web Interface

This chapter goes into details about the design and implementation of the website used to administrate Media-Online Management (MOM).

11.1 Design

As mentioned in the design overview, see subsection 6.1 on page 39, we know what the website is required to do for the system. These requirements made the basis for deciding what pages should be created and what each page's functionality should be. During the designing of the website we aimed to fit all relevant information into each page.
All pages have been designed on the blackboards, where inconsistencies was discovered, new navigation paths and learn about how concepts such as rules should be implemented.
Next is a short overview of what pages were designed and their purposes.

Dashboard is used to get a quick overview of relevant data. The overview is customizable so that each parent can decide what information is most relevant for them.

Devices is used to get an overview of which controllers and tags is registered within their MOM system. It also is the only way to navigate to the Add Tag/-Controller and Details Tag/Controller pages.
From this page it is also possible to deactivate a tag, meaning the owner of that tag, can no longer log in with it.

Devices - Details Controller is used to display, delete and change information about a controller. A user can only access controllers that is registered to his system.

Devices - Details Tag is used to display, delete and change information about

a tag. A user can only access tags that is registered to his system.

Devices - Add Controller is used to add controllers to the user's system. In order to add a controller the user must add a unique key found on the controller, this is then checked towards a database of available keys.
However it has not currently implemented this way of validation, but it would be the right way to go in order to commercialize the system and ensure security for our users.

Devices - Add Tag is like Add Controller, except with tags. Also in order to register a tag, a user must be dedicated to this tag. The one registering a tag, will be meet by a list of current users in the system. This means that in order to apply a tag, one must first create a user, then register a tag.

Users is used to get an overview of users in one's system. It is also used to navigate to Create User and User Details.

Users - User Details is for viewing details about a user. It is also used to change information about a user, however it is only possible to change information, if your user is of a manager-role, or if you are the owner of that profile.

Users - Create User is as the name suggests, used for creating a user. The system has been made ready for children to login to the system already now. We do not have any feature that requires the children to log in to the system as of yet, but a feature might utilize this later on.

Users - Chores is used to get an overview of available chores, and for easy rewarding of already created chores. When a chore is done, the parent should go to this site, and simply select a child and press a button to give them the assigned reward of the chore.
Chores will also have a subsite for creating chores.

Users - Rules is where parents can get an overview of rules and create them for their system. We have taken great care to help users in creating rules, by making a step by step procedure. Starting with making the user choose what sort of action they want to achieve and thereafter choose the sort of condition they want to set, since actions only match certain conditions. This should limit some of the confusion about the rules. For more information about rules, actions and conditions see subsection 5.2 on page 33.

Users - Permissions is designed for an easier way to think of rules, in relation to what devices which child is allowed to use in which time slots of the days.
We decided to create this permissions page, because permissions is an essential part of even the smallest of systems. This tool gives the parents an easy way to limit their child's usage, by deciding devices can not be turned on after a

certain time.

Graf page is intended for viewing statistics, such as how much have a controller been active, how is a users usage of media divided and so forth.

Calendar has two purposes. It serves as a graphical presentation of rules, in the sense that it shows when a rule will apply, and to what user.
It also serves to provide a shortcut to create rules, from other calendars. For example, if the user has a Google calendar, that have been registered to the system, they can simply click that event and the system will start creating a rule with a condition corresponding to the calendar events configuration.

Every page have been explained and in the next section the navigation is explained.

11.1.1 Navigation

When the user has logged in, they are navigated to the dashboard. From here and every other page they can navigate to the main pages: Dashboard, Devices, Users, Permissions, Rules, Chores, Graf and Calendar, see figure 11.1.
We decided to keep our navigation as simple as possible. In the design there is 5 head menus, Dashboard, Devices, Users, Graf and Calendar. Meaning Permissions, Rules and Chores are sub-menus and these are all listed under Users.

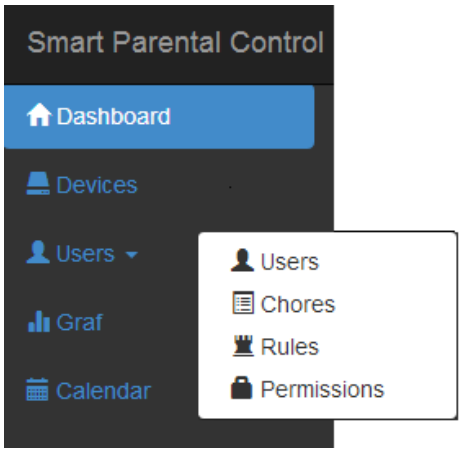

Figure 11.1: Website Navigation

We also designed the system such that no other menu or functionality is any more than 2 clicks away, every add or detail function can be accessed directly from one of the head menus.

11.1.2 Color Scheme and Layout Design

Our color scheme and layout design was inspired by both bootstrap and a website which utilizes bootstrap, called SB Admin[9]. A standard bootstrap color scheme have been used, and added in a little blue, we wanted to keep the colors simple but no further thought was put into it.
The most of the design is based on standard tools from bootstrap. With exception of the table sorter functionality. Which is a JQuery tool that allows for sorting of tables.
Bootstrap has been chosen since it gives an okay design and it is easy to implement prototypes in bootstrap.

11.2 Implementation

The previous section explained what each page should do, and in some detail how from the users viewpoint.

The implementation itself of the web-pages is not very interesting since they are based on a simple format of inputting the required information and submitting it to the server.
A few of the functionalists are however a bit more advanced. Two of these are toggling activity of a user and the page for users.

11.2.1 Toggling Activity

Under the page Devices, the overview of tags and controllers can be seen. But this is also were the activity of a user can be toggled.

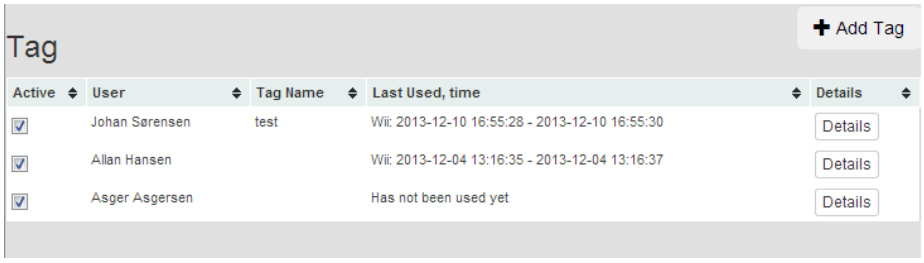

Figure 11.2: Tag Overview

This is done by pressing the checkbox in the left side of the tags overview, see figure 11.2. When this is pressed it activates an AJAX call that makes a http request to the PHP page `setActiveTag.php`. That PHP page then makes a query to the database in order to set the `active` status of a tag in the system, see figure 11.3. Resulting in the user no longer being able to use said tag. If it is pressed again the opposite happens and the user can again use said tag.

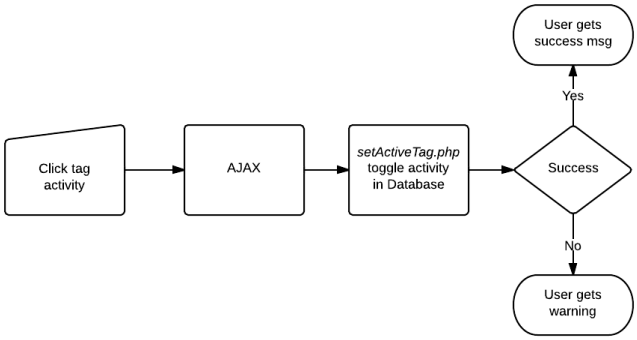

Figure 11.3: Flowchart for Toggling Activity

11.2.2 The Rules Page

The most advanced feature of the web interface is the rule page, therefor it is explained here.

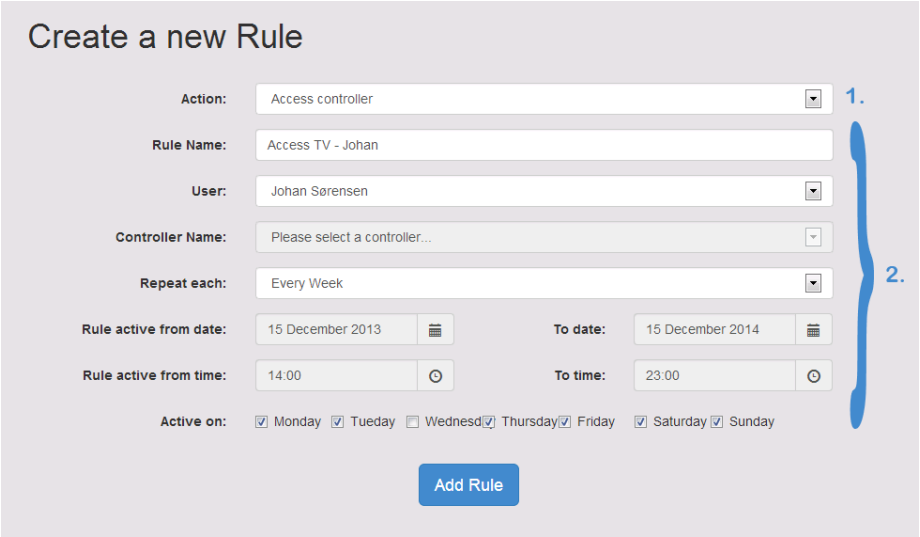

Figure 11.4: Create New Rule

As mentioned in the Database chapter, see 7 on page 43, rules underwent some changes during development of the MOM system. In the design of rule we decided that a rule could have several actions and conditions (1:many relationship), but during the development of the web page we found the that it would be too advanced for the user. Therefore it was decided that in the website a rule can have up to one condition and action (1:1 relationship). lisbeth: dette er min version, jeg synes det er bere

But that is not the only change made. The first layout was created with names for rules that fit into a computer science way of thinking. Names such as: ”permissionNone”, ”permissionAll”, ”true” and so forth. These names have been rethought into ”Cannot access any controller”, ”Can access any controller” and ”Infinite” Check om Infinite stadig er ordet., to fit into a pure English language. Yet another change was that rules could not be set to start or end at a given time, they would simply be active constantly after creation and follow the conditional pattern give by the creator.
In the final version a rule no longer consists of an action and a condition, at least not from the users viewpoint. Instead they choose an action, see 11.4 mark 1., and give that action a time constraint, see 11.4 mark 2..
This means that conditions such as ”Controller is On” and ”Controller is Off” is now incorporated into special action names. Evt. få navnet på disse actions. This is done in order to reduce the amounts of steps in a rule and also to reduce the number of decisions for a user to take during rule creation.

How it works

The changes to rules have been made clear and next is an explanation of how the page is used.

The flow can be seen from figure 11.5. First the user selects an action, then by use of JQuery the user is meet with the right information boxes. They are hidden until they need to be filled out, this helps the user stay focused on the right info and also does not distract the user with information that is not needed.

Next the user gives the rule a name, the name is only used in order for the user to recognize the rule at a later point, if it needs editing or removal.
Next the user should select the user and/or controller for the rule to affect. Whether the user has to select a user and/or controller is decided by the action they first choose.
Next the repeat pattern is selected. The repeat pattern changes the time constraint information boxes, such that it is possible for a user to create rules that only last for a single period of time, different sets of repeating periods of time or if the rule is to be always evaluated to true.

Lastly the user enters the time constraint corresponding to the select repeat pattern, this is done by the use of graphical selectors, such that the user does not enter a wrong format of day, month, year, hour and minute.
When the user then submits the rule, it is formatted to fit the database.

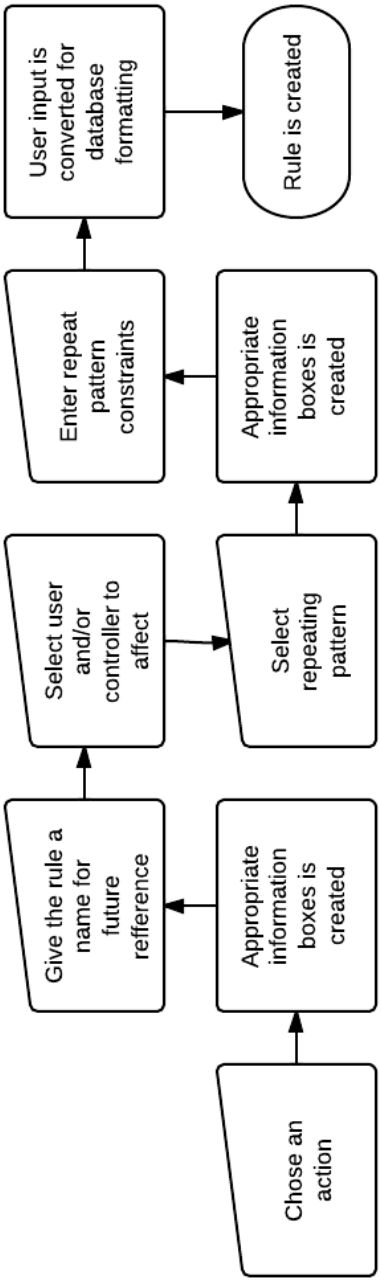

Figure 11.5: Create Rule Flowchart

Chapter 12

Overview of the Implementation

In the previous chapters we looked at the design and implementation of the different system part there are in the Media-Online Management (MOM). In this chapter we are going to show how it all of the parts are working together.

In figure 12.1 is a deployment and component diagram of the Media-Online Management system. There are a tag-reader and an Arduino node this is the controller. The controller is working independently of the website but it is dependent on the API. In the future if the Arduino should be replaced or another type of controller also should be used with MOM then it is only the code on the controller which should be adapted. On the centralized server there are five components: the Website, API, Daemon, Function lib, and the Database. All dependencies of among the components is going downwards from a system and user interfaces to the functions lib which manipulate the data in the database which is independent of all the components. The components Website, API and Daemon can be replaced without changing anything further in the system. However, if the component is API it could affect the controller if the system interface does not remain the same.

Before going explaining the testing of MOM, we are going to summarize some of the conclusion that the previous chapters came to.

The Database and the function library

The database is able to store all relevant data that are to be used by different system parts. In the database rule's and permission's data is considered to be the same. The tables that are storing a conditions data uses partitioning to avoid redundancy and reduce null values. In the database we use DELETE CASCADE with all foreign keys which mean that when a object is deleted then all of the references to this object is deleted.

The function library consist of several functions that are able to manipulate

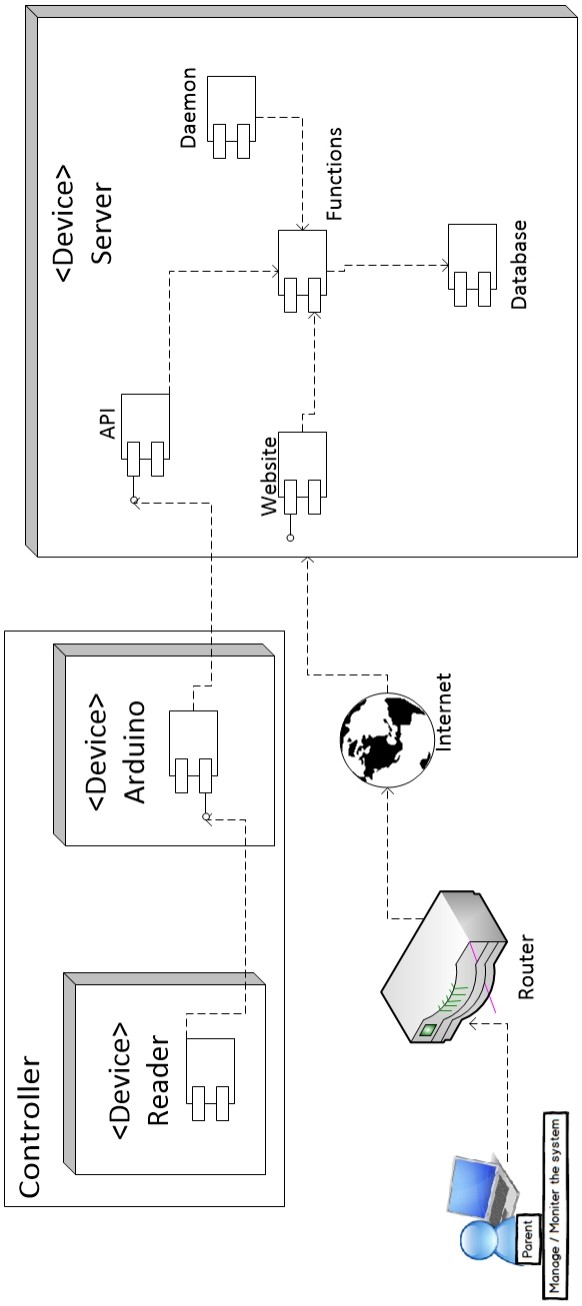

Figure 12.1: Deployment and component diagram

the data in the database. The database and many of the functions support that a rule can have several actions and conditions, but some of the functions only support one condition and action.

The API

In the API there are three functions turn on, turn off and status. Turn on checks whether a user is allowed to turn on a media, if it is then it calculate the time remaining, update the controller status in the database, and send a turn on message to the controller. Turn off will update the controller status in the database and withdraw the points used. Status checks up on if there have been any changes in time remaining for the user.

The Daemon

The daemon is a background process on the server which add points to the profiles when the time period is valid.

The Website

Through the website it is possible to add, edit and delete profiles, tags, and controllers. It is also possible to add and delete rule, permission and chores but edit has not been implemented.

The Arduino

The Arduino is the implemented solution for controlling interaction with media devices and it has been issued with both the software and extension in hardware to function in this capacity. To turn on a device the user must swipe his tag over the RFID antenna. Upon the swipe, the Arduino will read the Tag ID and call the API to check for permission to turn on the device, and if the permission is granted, the Arduino will do so. The Media device will continue to run until one of three events occur: The user chooses to log out from the device, which is done with another swipe and another call to the API. The user could also run out of time, at which point the Arduino shuts off the device. Finally, status updates, which the Arduino gets from calling the API regularly, could tell the Arduino to shut off.

Chapter 13

Testing of the Media-Online Management

There are different ways to evaluate the web interface e.g. an expert or user-based evaluation could be made. We would have preferred to make a user-based evaluation because this give a valuable input from a person who might use the system. However, to make such a sufficient user evaluation then we should find at least people with children which is between 5-14 years old and that would take too much time. A user-based evaluation of the entire MOM system would be optimal if the user and his/hers family could try it for a week or two before we interview them about the system. Also a log for our benefit could be made during this time to see how they use it of course with the test persons permission. Therefore to only evaluate the web interface we use an expert evaluation called Heuristic evaluation.

13.1 Heuristic evaluation in Theory

In a Heuristic evaluation the test persons look for errors and potential usability errors which can be categorized in 12 item and divided into 3 groups[1], which is shortly described below:

Learnability make it ease for the user to learn and remembering the system:

- Visibility - make it clear for the user which functions are available in the system and show what the system is doing.
- Consistency - be consistent in the design features and let it be similar to other comparable systems
- Familiarity - the language and symbols in the system should be familiar to the user.
- Affordance - E.g. if it looks like a button, it is a button.

Effectiveness is who ease it is for the user to use it and how safe it is to use:

- Navigation - provide support to enable people to move around. Make clear navigation instructions.
- Control - make it clear who and what is in control, the user should be able to take control. There should be a connection between what happens in the system and what will happen outside of the system.
- Feedback - give feedback on what have happened after user do an action in the system.
- Recovery - how good it is to recover if the user makes an error.
- Constraints - prevent people from doing inappropriate things in the system. This should be able to prevent serious error.

Accommodation accommodating differences between and respecting those differences:

- Flexibility - let there be multiple ways of doing things.
- Style - is the design attractive to the user.
- Conviviality - be polite and friendly. Do not use aggressive language.

Typically the test is done by 1-3 test persons[7]. The test person should be a usability experts or a software developer with the expertise in a relevant system. To make the test they should have relevant information of the user and a set of task to follow but they are also allowed to make their own.

13.2 Heuristic Evaluation of Media-Online Management

Before performing a Heuristic evaluation we need to determine what is needed before the evaluation is done and how the results should be presented[2].

13.2.1 Planning the Evaluation

When planing the evaluation the test persons need to get test persons. They need to get a small description of the user and some test cases. They should also get a list of the heuristics from which the system need to be evaluated from. This should all be made prior to the evaluation.

The evaluation of Media-online Management system is being done by test persons in the group which have been highly involved in the development of system. This is not optimal in this test where the test persons should be an usability expert from outside the development group[1].

The test persons also need to know the user. So the test persons need a general description of the user and their technology capabilities. The user of

MOM is parents and children. The children is only in contact with the controller and the parents are both using the controller and the website. The parents has a general experience with the internet, but they are not experts. However we expect them to be able to at least set up an e-mail account with Google or Hotmail. The children has few technology skill but they are capable to turn on a normal television.

Before the test is conducted some test cases should be made the development team. This should cover several cases in the system such that the test persons is presented to every aspect of the system. The test cases of MOM can be found in section 13.3.

We want to primarily be validated on.... (- choose the Heuristics which it should be evaluated against,(Visibility, Consistency, Control, Feedback, Familiarity is lisbeths suggestion)) TODO

13.2.2 Presenting the Results

to do THIS IS IN PROGRESS

13.3 Testcases

In table **??** is one of the test cases that we made for testing the entire Media-Online Management. In the test case both apart of the Website and a controller are tested. The change been done in the web site should affect the controller. All test cases can be found in appendix D on page 111.

Table 13.1: Example of a test cases

Name:	WS001
Description:	Setup a complete system with a managing user, a regular user, a 'TV Controller' and the accompanying rights to use it.
Requirements:	• A computer with Internet access. • The MOM website. • Two Tags prepared with a Tag ID. • An Arduino to function as the TV controller.
Expected Results:	A managing user capable of logging into the TV Controller without loosing points. A regular User able to log into the TV controller while loosing points.
Steps:	1. Log into the MOM website. 2. Create a profile with appropriate personal information to act as a manager. 3. Attach the first Tag to the new Manager profile. 4. Add the permissions that enables the use of all devices without expending points. 5. Create a profile with the appropriate person information to act as a user. 6. Attach the second tag to the new user profile. 7. Add the permissions to log into the TV controller. 8. Perform Test AT001A on both profiles with addendum: Wait 3 minutes for both users and note if either expends points.
Result of Test:	

13.4 Collected Results of the Heuristic Evaluation

THIS IS IN PROGRESS

Part III

Perspective

Chapter 14

Conclusion

THIS IS IN PROGRESS

First we need to take a look at the problem statement again.

Parents are not able to help their children administer their IT/TV consumption.
This results in their children getting a lessened learning ability, a bad sleeping pattern and a higher risk of type 2 diabetes.
How can hardware identification and webservices give parents the tools to help their children manage their media consumption?

- How do we identify unique users in a subtle and child friendly way.
- How do we enforce restrictions on media devices.
- How do we facilitate concepts as rules, permissions and chores without parents interaction.

14.1 How do we identify unique users in a subtle and child friendly way

A user is identified by a tag.

14.2 How do we facilitate concepts as rules, permissions and chores without parents interaction

The parent interaction has been limited to only the adding, editing and deleting of rules and permission on the website. They do not have to keep count on the time the children have been using the media, since this is automatically calculated. The rules and permission is enforced by the controller.

The chores, however, require that the parents know that the child has done a chore and the manually give points to this child's profile.

rules and permission, website, daemon.

14.3 How do we enforce restrictions on media devices

controller, api

Chapter 15

Future Work

In this chapter we discuss some improvement that could be made to the Media-Online Management system.

Rules
The rules have been changed during the implementation. Now the web site can only add one rule to one profile, but it should be possible to add a rule or permission to several profiles. On the website a rule can only have one condition and action it would be nice if a web interface at least could support several conditions. Such that a rule like the following to be created: Peter may not watch television from 8:00 to 14:00 and 19:00 to 23:59. However if this change should not happen then the functions working with rules can be simplified.

Chores
It would be nice if the points for doing a chore would be more automatized such that the parents should be less involved. This could be done by a special device where the child could choose a chore and scan his tag, which then register the chore has been made. However, a parent should still approve that the chore has been done by the profile such that the child do not misuse this.

Learn habits
It would be nice to implement a learning algorithm that could detect a pattern in the users behavior an example if a user regularly add points to the same user then the learning algorithm can detect a pattern and make a rule that will do this.

The data collected from this system could also be valuable statistics for studies with the users permission of course. So a data warehouse could be made to analysis the data. This data could also be used to marketing purposes.

Setup of controller
In section 10.1.1 on page 62 we discussed how the system should be setup. The controller's internet connection is currently a wired connection. To make it

more user friendly the controller should have WiFi connection. Then the setup of a controller will be more difficult because it is done manually. However this should be automatized such that it would be more user friendly. It could be that the user has a USB connector which can setup the wireless internet connection and add the controller to the database.

Another possible improvement of the controller would be a wireless connection between the tag reader and the power controller device. Such that the user can scan his tag from the couch. This require some other hardware than what we had access to.

Error handling
The error handling for when connection between the controller and server is lost has not been fully design and implemented. –More someone–

Calendar
Implement the calendar functionality fully. Where all the time bound rules are presented and the user should be able to make a rule in the calendar. It should be possible to show a calendar for one, some or all users. Other possible functionality could be import and export this calendar.

Mobile application
Another improvement of MOM could be to make a mobile application.

The Daemon The daemon should detect when a controller has not contacted the service in a while and do some disciplinary actions against the user. There could also be more real time rules added which would need to be implemented in the daemon. Also other real time things such as daily/weekly reports sent via email or SMS could be implemented to give parents automatic statistics about their childs media usage.

Bibliography

[1] David Benyon. *Designing Interactive Systems*. Peason, 2nd edition edition, 2010.

[2] Nicky Danino. *Heuristic Evaluation – a Step By Step Guide Article*. URL: http://www.sitepoint.com/heuristic-evaluation-guide/, 2001. Looked up: 15-12-2013.

[3] Mads F Hjorth. Seasonal variation in objectively measured physical activity, sedentary time, cardio-respiratory fitness and sleep duration among 8–11 year-old danish children: a repeated-measures study. Website, January 2013. http://www.biomedcentral.com/1471-2458/13/808.

[4] irisid. Iris recognition technology. Website, 2013. http://www.irisid.com/irisrecognitiontechnology.

[5] Abraham Silberschatz m.f. *Database System Concepts*. McGraw-Hill, sixth edition edition, 2011.

[6] Lars Mathiassen m.f. *Objekt Orienteret Analyse & Design*. Marko ApS, 3rd edition edition, 2001.

[7] Jakob Nielsen. *Technology Transfer of Heuristic Evaluation and Usability Inspection*. URL: http://www.nngroup.com/articles/technology-transfer-of-heuristic-evaluation/, 1995. Looked up: 15-12-2013.

[8] Robert W. Sebesta. *Concepts of Programming Language*. Pearson, 9th edition edition, 2010.

[9] Iron Summit Media Strategies. Sb admin. Website, December 2013. http://startbootstrap.com/sb-admin.

[10] Sundhedsstyrelsen. Fakta - bØrn og bevÆgelse. Website, 2005. http://sundhedsstyrelsen.dk/~/media/C3428A331C024E508754301DAE7A0399.ashx.

[11] Wikipedia. Internet of things. Website, December 2013. http://en.wikipedia.org/wiki/Internet_of_Things.

Part IV

Appendix

Appendix A

Rules first design

Before the final design had been made we used the design in this chapter.
To see a quick overview of the different conditions and their name:

Timeperiode the action can be done in this time period.

Timestamp the action may only happen at a given point in time.

Controller on the action can be done if a specific controller is turn on.

Controller off the action can be done if a specific controller is turn off.

True The action can always be taken.

An overview of the actions and their name is listed below:

Block user it will block the profiles of all profiles connected to the rule.

Activate user it will activate the profiles connected to it.

Add points it will add points to the profiles' points.

Delete points it will delete points to the profiles' points.

Set maximum of point it will set a maximum number of points that a profile can have.

Unlimited time it will give the profile unlimited time to be spend on any media.

Access any controller it will give the profile access to any media in the system.

Cannot access any controller the profile will not have access to any media.

Access controller it will give the profile access to a specific media.

Cannot access controller it will block the user from using a specific media.

The rule's structure is presented in a grammar in listing A.1 expressed in Extended Backus-Naur Form(EBNF) [8]. In a nonterminal is en-captured in <> and a terminal is just a word or en-captured in "". Also the grouping is used represented in (), the replica symbol is $*$, comments is $(**)$ and alternative is |.

A rule consist of a name and one of five action and condition sets which determine which actions and conditions can be combined. A rule can have several actions and conditions but only from the same set, see line 1-7. The action set is presented from 9-23 where all has a specific name, some have a specific Controller represented as a number and some has points which is a number. The condition set likewise represented from 9-23 and the condition types are presented from 25-30. There are four types but they each have a specific name. One type has Timestamp, another a Controller, the third only the name and the last is a timeperiod. The timeperiod has with two timestamp and if it is repeatable it has a string representation of the weekdays and a representation of then it is repeatable.

Listing A.1: Grammar of a rule in EBNF

```
1  <Rule>:= <name> (
2        (<ActionsetSet1><ConditionSet1>)*
3      | (<ActionsetSet2><ConditionSet2>)*
4      | (<ActionsetSet3><ConditionSet3>)*
5      | (<ActionsetSet4><ConditionSet4>)*
6      | (<ActionsetSet5><ConditionSet5>)*
7    )
8
9  <ActionsetSet1> := "Block user" | "Activate user"
10 <ConditionSet1> := <ConditionTimestamp>
11
12 <ActionsetSet2> := ("Add points" | "Delete points") <Points>
13 <ConditionSet2> := <ConditionTimeperiod> | <↩
       ConditionTimestamp>
14
15 <ActionsetSet3> := "Set maximum of point" <Points>
16 <ConditionSet3> := <ConditionTrue>
17
18 <ActionsetSet4> := "Unlimited time" | ("Access any ↩
       controller" | "Cannot access any controller") <↩
       Controller>
19 <ConditionSet4> := <ConditionTimeperiod> | <↩
       ConditionTimestamp>
20
21 <ActionsetSet5> := ("Access controller" | "Cannot access ↩
       controller")<Controller>
22 <ConditionSet5> := <ConditionTimestamp> | <↩
       ConditionTimeperiod>
23           |   <ConditionTrue> | <ConditionElse>
```

```
24
25 <ConditionTimestamp> :=  "Timestamp" <Timestamp>
26 <ConditionTimeperiod> :=  "TimePeriod" <Timestamp> <↩
       Timestamp> <Repeatable>
27 <Repeatable> :=   <Weekdays> <Repeat> | Nill
28
29 <ConditionTrue> := "True"
30 <ConditionElse>:= ("Device on" | "Device off") <Controller>
31
32 <Name> := ALPHA*  (* Upper and lowercase ASCII letters (A-Z,↩
       a-z) *)
33 <Controller> := DIGIT* (* Decimal digits (0-9) *)
34 <Points> := DIGIT*
35 <Timestamp> := 4*DIGIT,"-",2*DIGIT,"-",2*DIGIT, " ",2*DIGIT↩
       ,":"2*DIGIT,":"2*DIGIT  (*YYYY-MM-DD HH:mm:ss*)
36 <Weekdays> := ("monday"| "tuesday"| "wednesday"| "thursday"|↩
        "friday"| "saturday"| "sunday")*
37 <Repeat> := "weekly" | "biweekly" | "triweekly" | "first in ↩
       month" | "last in month"
```

Appendix B

Light Table

Description	**Light Action**
Looking for connection to router	All lights blink
Looking for connection to internet	Constant red light
Ready for tag	Constant green light
Decline user	Blinking red light in 3 seconds
Accept user	Blinking greenand orange light 3 seconds
User logged in with internetconnection	Constant green and orange
Accept new user with a current user	Blinking greenand orange light 3 seconds
Decline new user with a current user	Blinking red light 3 seconds
Disconneted with user	Constant red and orange light

Appendix C

bytecode???????

	The RFID output in UART for seeking for tag: On 'Tag Found'. (The length would be 0x06.)
Slot 0-3	Contains the message "Header", "Reserved", "Length" and "Command".
Slot 4	Contains the tag type.
Slot 5-8	Contains the data stored in the block.
Slot 9	Contains the Checksum.
	On 'no tag found'. (The Length is 0x02.)
Slot 0-3	Contains the message "Header", "Reserved", "Length" and "Command".
Slot 4	Contains the Error Code. - 0x4C: 'L' Command in progress. - 0x55: 'U' Command in progress but RF field is off.
Slot 5	Contains the Checksum.
	The RFID output in UART for Authenticating a Data block.
Slot 0-3	Contains the message "Header", "Reserved", "Length" and "Command".
Slot 4	Contains the Status/Error Code. - 0x4C: 'L' - Login Successfull. - 0x4E: 'N' - No Tag Present or Login Failed. - 0x55: 'U' - Login Failed. - 0x45: 'E' - Invalid Key format in E2PROM.
Slot 5	Contains the Checksum.
	The RFID output in UART for reading a block: On a success. (The length would be 0x12.)
Slot 0-3	Contains the message "Header", "Reserved", "Length" and "Command".
Slot 4	Contains the number of the block read.
Slot 5-20	contains the data stored in the block.
Slot 21	Contains the Checksum.
	On a Fail (The length is 0x02.)
Slot 0-3	Contains the message "Header", "Reserved", "Length" and "Command".
Slot 4	Contains the error code: - 0x4E: 'N' No tag present. - 0x46: 'F' Failed to read.
Slot 5	Contains the Checksum.

Appendix D

Test Suite

This chapter contain all the testcases which has been tested in the Media-Online Management system.

Name:	AT001A
Description:	Log in and out with the Arduino and a valid Tag.
Requirements:	• A LED light connected to the Arduino which functions as the "Device". • Tag connected to a user with enough points and permission to use the Device. • The Arduino running the final software version. • Serial Connection to Arduino. • Web Browser with link to Status API for the device being used.
Expected Results:	When the RFID antenna detects the tag, the LED light will turn on, The Serial will note that it is now running in State 1 and the web browser will report that the Status is green. When swiping the tag a second time, the LED light will turn off, The Serial will report that the Arduino is running at State 0 and the webbrowser will confirm that the status is RED for not running. After either swipe the Arduino will be ready for a new tag swipe.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. On the Serial Output, note if the State changes from 0 to 1. 4. Confirm on the web browser that the device is marked status:GREEN running. 5. Swipe tag over the RFID antenna again and observe if the LED turns off. 6. On the Serial output, note if the State changes from 1 to 0. 7. Confirm on the web browser.
Result of Test:	

Name:	AT001B
Description:	Log in with the Arduino and a Tag that does not have the proper permissions.
Requirements:	• A LED light connected to the Arduino which functions as the "Device". • Tag connected to a user without the permission to use the Device. • The Arduino running the final software version. • Serial Connection To the Arduino. • Web Browser with link to Status API for the device being used.
Expected Results:	When the RFID antenna detects the tag, the LED light will remain off, the state will remain 0 and the web browser will report that the Status is RED for not running. The Arduino will return to waiting for a new tag swipe.
Steps: Result of Test:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. On the Serial Output, note if the Arduino changes state. 4. Confirm on the web browser that the device is marked status:RED for not running.

Name:	AT001C
Description:	Log in with the Arduino and a Tag not supplied with enough points to run.
Requirements:	• A LED light connected to the Arduino which functions as the "Device". • Tag connected to a user without enough points to use the Device. • A serial connection to the Arduino • The Arduino running the final software version. • Web Browser with link to Status API for the device being used.
Expected Results:	When the RFID antenna detects the tag, the LED light will remain off, the state of the arduino will not change and the web browser will report that the Status is RED. The Arduino will return to waiting for a new tag swipe.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. Observe on the Serial Output if the state changes. 4. Confirm on the web browser that the device is marked status:RED for not running.
Result of Test:	

Name:	AT001D
Description:	Log in with the Arduino and a Tag not recognized.
Requirements:	• A LED light connected to the Arduino which functions as the "Device". • A tag that has not been introduced to the system yet. • A Serial Connection to Arduino. • The Arduino running the final software version. • Web Browser with link to Status API for the device being used.
Expected Results:	When the RFID antenna detects the tag, the LED light will remain off, the state of the arduino will not change and the web browser will report that the Status is RED. The Arduino will return to waiting for a new tag swipe.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. Observe on the Serial Output if the state changes. 4. Confirm on the web browser that the device is marked status:RED for not running.
Result of Test:	

Name:	AT002A
Description:	Swipe a valid Tag that has the right permissions and points, while another user is logged in with the Arduino.
Requirements:	• A LED light connected to the Arduino which functions as the "Device". • Tag connected to a user with enough points and permission to use the Device. • A second Tag connected to a user with enough points and permission to use the Device. • The Arduino running the final software version. • Web Browser with link to Status API for the device being used.
Expected Results:	When the second tag is swiped while the first user is still active the LED should briefly flicker off and then on again as the new user logs back in.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. Confirm on the web browser that the device is marked status:GREEEN for running. 4. Swipe tag over RFID antenna and observe if the LED Turns off and then on again. 5. Confirm on the web browser that the device is still marked status:GREEEN for running.
Result of Test:	

Name:	AT003A
Description:	Let Arduino run until getStatus is called without logged in User.
Requirements:	• The Arduino running the final software version. • Web Browser with link to Status API for the device being used. • Serial Connection to Arduino.
Expected Results:	Run smoothly, remain in State 0, Remain turned off.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Wait and confirm that the status has run with the Serial Watch and note if its Status:RED. 3. Confirm on the web browser that the device is still marked status:RED for running.
Result of Test:	

Name:	AT003B
Description:	Let Arduino run until getStatus is called with logged in User.
Requirements:	• The Arduino running the final software version. • Tag connected to a user with enough points and permission to use the Device. • A serial connection to the Arduino. • Web Browser with link to Status API for the device being used.
Expected Results:	The User will Remain logged in and the Arduino will not change from state 1.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. Wait and confirm that the status has run with the Serial Watch and note if its Status:GREEN. 4. Note if the Arduino changes state. 5. Confirm that the LED remains on. 6. Confirm on the web browser that the device is still marked status:GREEN for running.
Result of Test:	

Name:	AT003C
Description:	Let Arduino run until getStatus is called with logged in User no longer permitted to use the Arduino.
Requirements:	• The Arduino running the final software version. • Tag connected to a user with enough points and permission to use the Device. • Access to the stuff that controls permissions. • A serial connection to the Arduino. • Web Browser with link to Status API for the device being used.
Expected Results:	The Arduino will run with user logged in state 1 and with the LED turned on until the timer is reached. Then the user will be logged out, the LED will turn off and the Arduino will move to state 0.
Steps:	1. Turn on the Arduino. (Wait for Serial to confirm that the device is running.) 2. Swipe tag over RFID antenna and observe if the LED turns on. 3. Confirm on the web browser that the device is still marked status:GREEN for running. 4. Use the Web Service to rescind permission to the device. 5. Use browser to confirm that the device is marked as status:RED. 6. Wait and confirm that the status has run with the Serial Watch and note if its Status:RED. 7. Confirm that the LED turns off.
Result of Test:	

Name:	WS001
Description:	Setup a complete system with a managing user, a regular user, a 'TV Controller' and the accompanying rights to use it.
Requirements:	• A computer with Internet access. • The MOM website. • Two Tags prepared with a Tag ID. • An Arduino to function as the TV controller.
Expected Results:	A managing user capable of logging into the TV Controller without loosing points. A regular User able to log into the TV controller while loosing points.
Steps:	1. Log into the MOM website. 2. Create a profile with appropriate personal information to act as a manager. 3. Attach the first Tag to the new Manager profile. 4. Add the permissions that enables the use of all devices without expending points. 5. Create a profile with the appropriate person information to act as a user. 6. Attach the second tag to the new user profile. 7. Add the permissions to log into the TV controller. 8. Perform Test AT001A on both profiles with addendum: Wait 3 minutes for both users and note if either expends points.
Result of Test:	

Name:	WS002
Description:	Adding a rule to ensure that one device is turned on in order for another to be turned on.
Requirements:	• A computer with Internet access. • The MOM website. • Web Browser with links to the API to simulate 'TV' and 'Playstation' devices.
Expected Results:	The simulated devices one will have to be turned on in order to turn on the Simulated controller 2.
Steps:	1. If a 'TV' device has not been established from earlier Test, create this. 2. Create a 'Playstation' device. 3. Establish the Rule that the 'Playstation' device cannot be turned on unless the 'TV' device is. 4. Use the browser to call turnOn for the 'Playstation' and note if it turns on. 5. If the the 'Playstation' did not turn on in step 4, turn on the 'TV' and then try again to turn on the 'Playstation'.
Result of Test:	

Name:	WS003
Description:	Adding a rule to block a profile from using the 'TV' Device.
Requirements:	• MOM Website. • TV Device. • Test Profile with Tag. • Web Browser with links to the API to simulate 'TV' device.
Expected Results:	The user attatched to the profile will be unable to log into the 'TV' device in accordance to the established Rule.
Steps:	1. Log into Mom Website. 2. Add Rule to block the profile. 3. Use the Web browser API to test if you can activate the Device.
Result of Test:	