Todo list

Smart electricity
Project report: IT2901

Beate Baier Biribakken

Tor-Håkon Bonsaksen

Lars Erik Græsdal-Knutrud

Per Øyvind Kanestrøm

Håvard Holmboe Lian

Pia Karlsen Lindkjølen

Spring 2014

Contents

Code snippets

Figures

Tables

dobbelsjekke om urler er de riktige urlene-> fører til riktig nettsted

dobbelsjekke om referansene fungerer som de skal -> at det ikke står et spmtegn der

sjekke at alle illustrasjoner har en caption

sjekke at captionene gir mening. Forklaring bør gjøres i avsnitt som refererer til figuren/tabellen

sjekke at de riktige illustrasjonene er på plass

divide sections into own files

spellcheck

back end -> back-end

Dobbelsjekke om egennavn er skrevet riktig

check all verbs and formulations for past tense

se over rapporten om vi blir omtalt som noe annet enn the team, our, we; IKKE group, their, they

explain what gamification is in a footnote or in text

1 Introduction

1.1 About the assignment

Current situation

Our project is a part of an ongoing research program, CoSSMic [**?**], focusing on how to store renewable energy in private households. The idea is that entire neighborhoods should be able to benefit from the excess energy produced by other households in the neighborhood. The research program aims to develop the tools needed for this sharing of energy.

The objective of our project is to create an Android application, making it possible for the user to monitor his[1] energy usage and energy production.

There are already a couple of similar applications on the market, and some products with entire system setups to measure and control users' energy consumption. What all of these solutions have in common is that they are either expensive, or they do not provide all the functionality the team want to have in the project application. None of the other solutions provide support for measuring the users' energy production. These solutions are further explained in section 3.5.

What the customer wants

The specifications provided by the customer are open for interpretation. The customer wants the team to come up with an application that is user friendly and simple, so that anyone can measure their energy consumption. This should be done preferably per device. The users should be able to share their energy consumption and -production with their friends on Facebook [**?**]. The customer also wants the concept of gamification [**?**] to be brought into the application by allowing the users to compete with eachother to save and/or produce the most energy. In order to do measure the users' energy consumption per device, the team needs a hardware device. Optimally, the device transmits the data either via Wifi, Bluetooth, or another form of wireless communication protocol.

[1] This report will use the term "his" to refer to "his/her" and "he" to refer to "he/she"

Due to time restrictions, and unless we find a device that is compatible with the team's solution and within a reasonable price range, the customer does not expect the team to make the hardware for the application.

1.2 About the customer

Our customer is the social inclusion technology research group at SINTEF [**?**].

2 Project management

2.1 Team organization

In order to make the best use of the team's resources, all the team members has summed up their background knowledge and which role they wanted to have in the project. The team has organized its structure based on this information.

The team members

The team consisted of six individuals, all in their final year of a degree in computer science. All the members has previous experience on working in teams on educational projects.

Beate Baier Biribakken

Beate has previously worked at the IT-companies Student-Media AS [**?**] and Sportradar AS [**?**] as a web developer. From these experiences, she gained knowledge about Linux, web development, such as PHP, JavaScript, HTML and CSS, and the project framework Scrum. She also has some experience with Java and MySQL from school and has done some Android development in her spare time.

Tor-Håkon Bonsaksen

Tor-Håkon has a trade certificate in data electronics and broad experience with web development through extracurricular activities within the groups dotKom [**?**] and Casual gaming[**?**]. This includes knowledge about Python, Django, HTML, CSS and JavaScript. In addition, he has done some Android development in his spare time. He also has some experience with Java through school.

Lars Erik Græsdal-Knutrud

Lars Erik has experience with Java, C#, C++ and SQL through his ongoing education. As the internal systems developer for Orakeltjenesten Dragvoll [**?**] he also has some experience

with PHP and server environments.

Per Øyvind Kanestrøm

Per Øyvind is a GNU/Linux user. He is currently working on a web project in PHP/Symphony2 and an app project on the Android platform. From school he has experience in Java, Python, Scrum and MySQL.

Håvard Holmboe Lian

Håvard has experience with Java, Python, SQL, C, C#, and VHDL from school. He has also written C code for embedded systems with and without an OS (Linux).

Pia Karlsen Lindkjølen

Pia has some experience with Java, Python, MySQL and Scrum from school projects. She also have had some experience with project management.

Main responsibilities

Role	Description
Project leader: Pia	Keeping the team updated and monitor the project's status
Deputy project leader: Lars Erik	Fill in whenever the project leader is incapable to perform all duties
Scrum-master: Per Øyvind	Make sure that the Scrum-process goes as smooth as possible, that the team provides necessary documentation and keep track of the project process
Customer relations: Pia	All customer communication mainly goes through this person.
Development: Tor-Håkon	Keep track of the technological development progress, make sure it is on schedule and take necessary action if it does not
Report: Beate	Monitor the report's progress, spell check and review content.
Testing: Håvard	Make sure that the code is properly tested during the development process to detect possible errors, deficiencies and bugs and take the necessary action
Secretary: circulates	Take note of important information during meetings within the team and with the customer.

2.2 Available resources

Time schedule

The assignment and team members were assigned January 20. The initial meeting with the customer took place January 24, where the assignment was presented and discussed. Oral presentation of the project was performed March 19 and the final deadline for submission of the project was set to May 30. The team set the deadline for new specifications from the customer that would result in major changes to the software, to March 21., due to time restrictions.

Available hours

Each sprint had a duration of two weeks, excluding the period from April 4. to April 21., when the entire group leaved for a school field trip to China.

Table 2.1 lists the project's available hours and is based on that all team members spends twenty hours on the project every week.

To compensate for the days the team missed because of the school field trip mentioned above, the team decided to increase the amount of work hours from 20 hours to 25 hours as of sprint 2 to and including sprint 6.

Sprint #	**Dates**	**Days**	**Hours**
Sprint 1	January 27. - February 07.	10	240
Sprint 2	February 10. - Februar 21.	10	300
Sprint 3	Februar 24. - March 07.	10	300
Sprint 4	March 10. - March 21.	10	300
Sprint 5	March 24. - April 04.	10	300
Sprint 6	April 21. - May 02.	10	300
Total		**60**	**1740**

Table 2.1: Available hours

Milestones

Milestone #	**Description**	**Deadlines**
Milestone 1	Project report - preliminary version	February 9.
Milestone 2	Project report - mid-semester version	March 16.
Milestone 3	Peer evaluation	March 23.
Milestone 4	Project report - final version	May 30.

Table 2.2: Milestones

Supervisor from the Department of Computer and Information Science

The team's supervisor was Alfredo Perez Fernandez. He is a PhD student at NTNU [**?**] in the Department of Computer and Information Science [**?**]. He may be contacted by e-mail at perezfer@idi.ntnu.no.

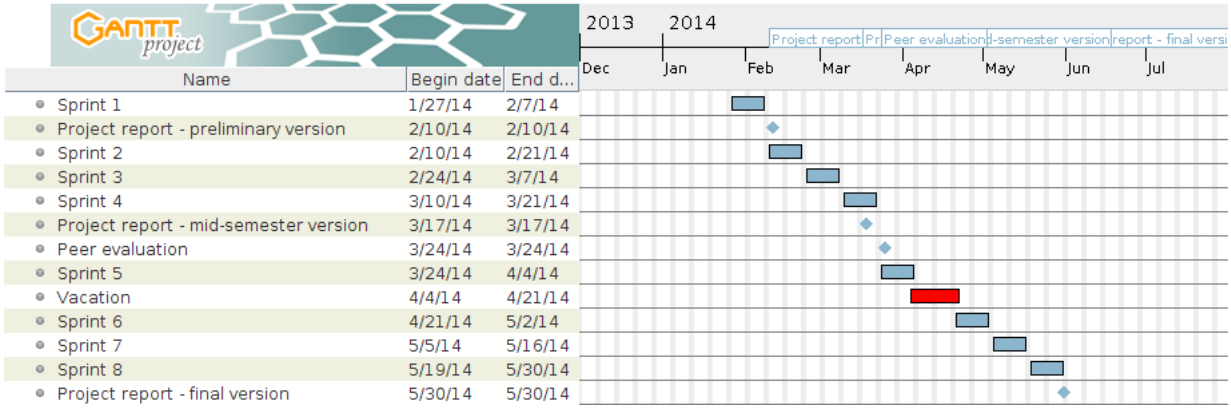
Customer representative at SINTEF

The team's customer representative at SINTEF was Babak Farshchian. He is an adjunct associate professor at NTNU and a researcher at SINTEF ICT [**?**]. He may be contacted by e-mail at babak.farshchian@sintef.no.

3 Planning

Gantt diagram

A Gantt diagram is a representation of all the work hours, milestones and deadlines that is involved in a project. The Gantt diagram for our project is given in figure 3.1.



Name	Begin date	End d...
Sprint 1	1/27/14	2/7/14
Project report - preliminary version	2/10/14	2/10/14
Sprint 2	2/10/14	2/21/14
Sprint 3	2/24/14	3/7/14
Sprint 4	3/10/14	3/21/14
Project report - mid-semester version	3/17/14	3/17/14
Peer evaluation	3/24/14	3/24/14
Sprint 5	3/24/14	4/4/14
Vacation	4/4/14	4/21/14
Sprint 6	4/21/14	5/2/14
Sprint 7	5/5/14	5/16/14
Sprint 8	5/19/14	5/30/14
Project report - final version	5/30/14	5/30/14

Figure 3.1: The Gantt diagram with sprints and milestones.

3.1 Work Breakdown Structure

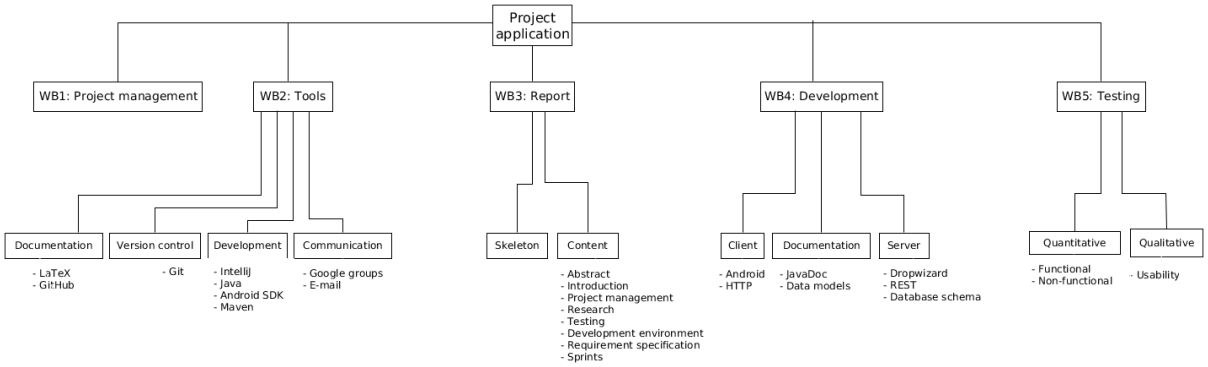


Figure 3.2: The work breakdown structure. Each work package (WP) is a product of the project work.

@peri: is wbs-packet correct? should it be wp (as in work package?)

The work breakdown structure 3.2 is divided into five parts. Each part is a work package that represents a part of the work that needs to be done. The team decided to use WBS is so that we may analyze how much time that is actually spent on the different parts of the project, compared to how it originally was planned. The effect of doing this is that it is easy to see whether some part of the project is forgotten or
gets too much time.

WB #	Name	Hours
1	Project management	.5/10 = 87
2	Tools	1.5/10 = 261
3	Report	4/10 = 696
4	Development	3/10 = 522
5	Testing	1/10 = 174

From the total amount of hours for project, the team has divided each WBS packet into a smaller part, based on how much time the team should use one the given part.

For each sprint the team can see how much time that has been spent on each packet, and if time is beeing distributed according to the planned amount

3.2 Agile software development

Agile development methods [?] are known to be successful for small developer teams where the goal is to produce a small or medium-sized product, and is best suited for application development where the system requirements are likely to be frequently changed.

The intention of such development methods is that the team is to deliver working software quickly to the customer so that he may propose new features or change the requirements for future iterations of the application.

Based on the correlation between the descriptions of agile development methods and the team's needs, it would be safe to assume that the team would benefit from following the guidelines of an agile development framework.

First and foremost it should also be a development framework that is known to the team, so that a minimum of time will be spent trying to learn a new process.

As all of the team members had previous experience with the Scrum approach from the course IT1901, a mandatory course in the computer science bachelor at NTNU, the obvious choice was to follow this approach, which is a general agile method in which the main focus is the project management itself, rather than specific technical aspects.

In the following section, the concept behind the Scrum process will be thoroughly explained. The team's project process will be reviewed in detail in chapter 7.

Scrum

The concept behind the Scrum approach is to empower the entire team to make decisions, so that a project manager would become redundant. However, in order to manage and make the best of the team's resources, the team found it necessary to keep this position in this project.

The Scrum approach is an iterative and incremental agile software development framework that consists of three phases: The outline planning phase, which is followed by a series of sprint cycles, and lastly a project closure phase.

These sprint cycles each consist of a single sprint, in which has a duration of one to four weeks. Each sprint has three important parts: The first part is the planning meeting, then the daily meetings, and finally, the process is concluded with an end meeting.

Sprint planning

The objective of the sprint planning is to find out what work that needs to be completed within the duration of the sprint. This is done by preparing a sprint backlog that consists of the tasks to be done, called user stories. Each user story is estimated based on how much time the team thinks it will take to complete, based on previous experiences or lack of it, and difficulty level. In the Scrum approach, planning poker [?] is a common strategy to use for time planning.

In planning poker, each team member individually decide on how many units of time they think a task and/or user story will require to complete. This is repeated for each task and/or user story. The units of time can be in hours, work days, or whatever unit the team sees fit.

In the team's planning poker sessions, it was decided to use the units small (S), medium (M), large (L) and extra large (XL), each of them respectively representing one hour (S), two hours (M), four hours (L) and eight hours(XL).

When a user story is presented, every team member presents the unit they believe the user story will require. If the entire team agrees on one estimate per user story, then the user story is assigned that particular estimate. If not, the team must discuss why they chose the particular unit and come to a conclusion the entire team agrees upon. After this meeting the team should have a well prepared strategy for the sprint.

Daily meetings

A work day is usually initiated with a daily meeting. The point of these meetings is to give an overview to the entire team of what all the other members are working on. For about fifteen minutes, all the team members briefly summarize which tasks they have performed and whether something went wrong.

By having this meeting, the team will quickly become aware if something has not gone as planned, such as a poorly estimated user story that requires more thought and replanning, or some other risk may occur, as discussed in section 3.3. The result of having this meeting is that the problems that arise may be dealt with quickly.

End meeting

The end meeting is held at the end of a sprint. The meeting consists of a sprint review and a retrospective discussion for the last sprint. Thus the project progress is reviewed, and accumulated lessons from the sprint can be taken in account for the next sprint. It's also typical to have customer meetings to show what has been done so the customer can give feedback on whether or not he is satisfied with the product being developed.

Our process

The team has chosen to have sprints with a duration of two weeks. Unfortunately, having daily meetings is not possible with the team's schedule. The team has come to an agreement to meet twice a week, each beginning with a daily meeting. In addition, the team has a fixed time to meet with the customer once a week, unless the team or the customer does not see the need for it.

Scrum tool

The team has also decided to use a Scrum tool to make the process easier. To find this tool, the team discussed what kind of functionality that would be useful to the project.

Specifically, the team wanted a tool that provided automatic sprint backlogs, time measurements for each user story, graphs, and an interactive Scrum board. See section 7.1 for further details.

Add something about XP

3.3 Risk analysis

As part of our project planning, we outlined potential risks to the progress of our project. A risk is defined as an unwanted event that has a negative affect on the process. We acknowledged the possibility of challenges and problems that might arise during the project, such as technical problems, human errors or personal problems. These are risks that might apply to both individuals and the entire team. Effects of these problems include delays, conflicts and anything that might slow down the progress, and ultimately lead to failure to meet deadlines.

The risk elements are sorted by their importance. Importance is calculated with two factors in mind; the calculated probability of the event actually occurring and the effect the event will have on the project.

In the table given below, we analyze the elements we consider risks to our project. Likelihood (L) and effect (E) is measured on a scale from 1 to 9. For likelihood, a 9 is very likely and a 1 is very unlikely. For effect, a 9 is devastating and a 1 is very little effect. The table is sorted from high to low importance (I). Importance is the product of probability and effect.

Description	L	E	I	Preventive actions	Remedial actions
Underestimation of workload	9	8	72	Continuously revise workload and how much time is left, and prioritize	Reestimate continuously
Issues with software or tools	9	5	45	Choose software the team members are familiar with	Hold workshops and view tutorials
Customer has not fulfilled his obligations	8	5	40	Keep customer updated and maintain continuous communication. Set final deadlines for feedback.	Contact supervisor.
Illness	9	4	36	Multiple team members work on the same task	Allocate sick member's task to remaining members
Unbalanced workload	5	7	35	Coordinate with entire team and log hours	Reallocate tasks
Team member unavailable	9	3	27	Inform each other of dates we know we will be unavailable. Good infrastructure for communication and progress reports	Keep in touch with unavailable team member or redistribute tasks
Eclipse and IntelliJ is incompatible	7	3	21	Evaluate whether use of different IDEs have any negative impact on project	Decide to choose only one of them.
External services unavailable	4	7	21	Project not fully dependent of external services. Server is easy to deploy.	Must find new external service to replace the one(s) becoming unavailable. Deploy elsewhere.
Data loss	2	9	18	Use version control system	Restore data from previous versions

Continued on next page

Table 3.1 – continued from previous page					
Description	**L**	**E**	**I**	**Preventive actions**	**Remedial actions**
Customer unavailable	3	6	18	Keep regular contact with customer	Discuss problem within team and contact supervisor
Communication failure	7	2	14	Make e-mail-list and exchange contact information. Be explicit.	Check e-mail and phones multiple times a day.
Customer requirements exceeds predicted amount of time	4	3	12	Continuously revise the workload and prioritize. Set a date for last major changes.	Politely explain that there is not enough time for the changes he suggests.
Supervisor unavailable	3	4	12	Keep regular contact with supervisor	Discuss problem internally and contact department.
Conflict in group	5	2	10	Have in mind that people often misunderstand each other and have an open dialog to solve problem.	Contact supervisor for help.
Team member has not completed given assignments	1	7	7	Keep track of the effort and the hour's log public	Contact supervisor and redistribute tasks
External disruptions in the work environment	x	x	x		Find another place to work

Table 3.1: Risk analysis table

3.4 Architecture

The implementation will consist of three parts. The server, The device in the users home used to collect and send usage data, and the Android application. The android application is the main focus for this project and most of the teams resources will be used on this.

Server

The server will consist of two parts. One for handling requests from the Android application and one for storing data that is collected in the users home. The part that handles the android application will be running Dropwizard [**?**] which is a collection of Java libraries that together make a REST service.

These libraries include Jetty [**?**], Jersey [**?**], JDBC [**?**], and Jackson [**?**]. The server will expose a set of URL's that will provide the android application with possibilities to get and store data from the database.The other part that stores data from the users home is under development. Info will come here.

Android Device

The application running on the users Android device will be designed to run on Android 4.0 or greater. The team will also use the Model-View-Presenter (MVP) [**?**] this is a Model-View-Controller (MVC) [**?**] derivative that pushes almost all logic into the presenter. The application will follow standard Android design guidelines [**?**] when it comes to user interface design.

Home Data Aggregator

This device will collect data from sources in the user's home. The reason for this external box as opposed to using the user's phone to collect data, is that with this solution the system can fetch data at regular intervals throughout the whole day. This would result in that the user would not need to be home in order for the application to collect usage data. The device will pass data along to the external server at request from the server.

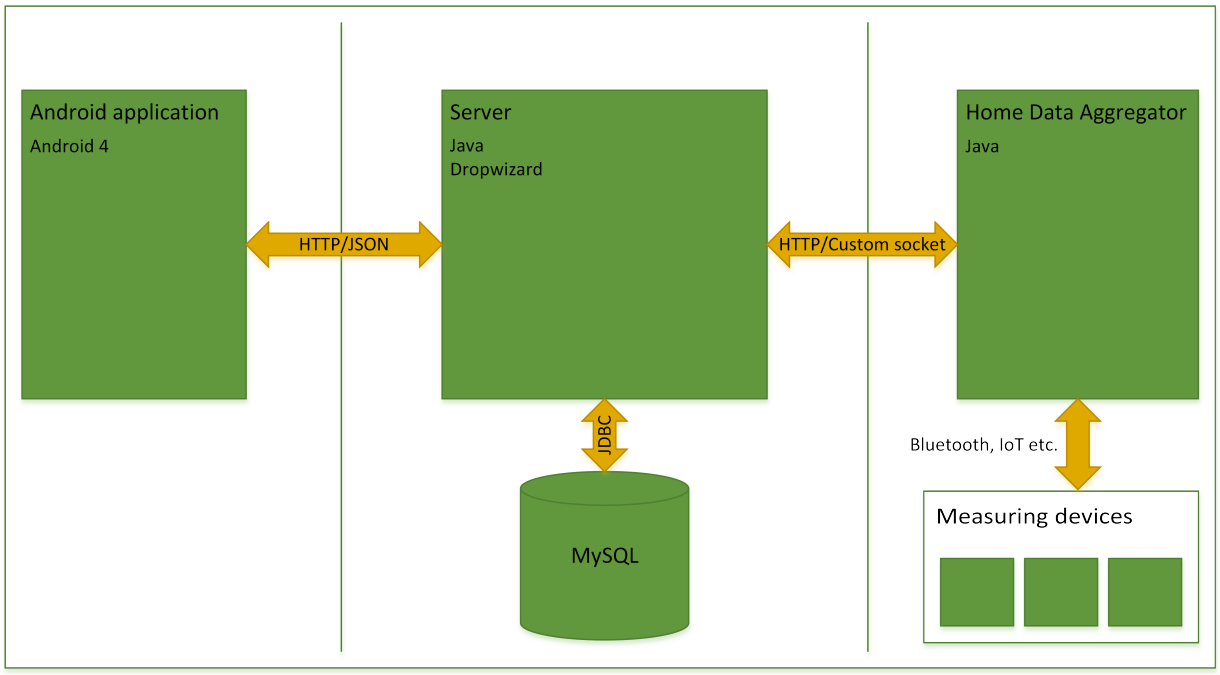


Figure 3.3: Architecture

3.5 Existing solutions

Based on the product requirements the team has worked out, the team has done some research on existing solutions in the same market. This has resulted in a list of the most relevant solutions, possibly competing with the team's own application.

This list contains a brief overview of the functionalities and drawbacks of the existing solutions. The team will use this study in alternative solutions as an inspiration for functionality and features to implement in the project application.

Smartly

Smartly [**?**] allows the user to monitor and control things like temperature and lighting in the house. It also has an overview over total power consumption. Even though Smartly has some of the features the team would like to have in the application, such as being able to turn on/off devices, a good design and a display of money saved caused by use of the application, it does not offer the ability to measure the power usage in each device, which is a high-priority feature in the team's application.

OpenEnergyMonitor

OpenEnergyMonitor [**?**] is an open source project that allows data collection from power outages. Some of the architecture for collecting data can be an interesting option to consider if improved. The architecture is somewhat similar to what the team itself imagines to use in the application.

However, OpenEnergyMonitor offers no automatic collection of data - the user would have to manually go around his house to collect the data. It is also somewhat hard to set up, which would make the solution not very user friendly for the average consumer. This solution also has an application for processing, logging and visualizing energy usage. These are features the team would like to have in the application.

NTE miniSolo energydisplay

The NTE miniSolo energydisplay [**?**] measures the total power usage real time and allows the user to power off devices to see the effect the device has on the total power consumption. It offers some interesting features like detailed power consumption and a relation to how much money the power usage amounts to, which the team also want to have in the application. However, the miniSolo energydisplay has some drawbacks: It is linked to a proprietary device and has no Android application. These drawbacks makes the solution incompatible with the team's application.

Theowl

Theowl [**?**] mainly focuses on temperature control. As that is not the only property the team would like to focus on, the solution lacks many of the features the team would like to have in the application. Other drawbacks is that the product does not have any official support in Norway and also is beyond a reasonable price range. However, Theowl has an architecture with remote sensors that sends precise data and allow the user to control certain devices, which the team would like to consider for the application.

Efergy

Efergy [**?**] is considered to be the closest solution to what the team want to develop. It has nice visual representation of data, and it offers social integration. The architecture is very interesting as it has support for measuring the power usage of single devices over wireless radio, communicating with a local receiver. The receiver is connected to a server through the users Internet. The disadvantages of Efergy is that it lacks the ability to monitor, and control private power production. Monetary it is also beyond a reasonable price range.

Comparison of existing solutions

Add column on whether the solutions can measure the energy consumption of single devices

Product	Device control	Social media	Measure production	Measure individual devices
Smartly	Limited	No	No	No
OpenEnergyMonitor	Yes	No	Limited	No
Minisolo	Yes	No	No	Limited
TheOwl	Yes	No	No	Yes
Efergy	Yes	Yes	No	Yes

Table 3.2: Existing solutions

4 Testing

In order to deliver a complete and functional product to the customer, it is paramount to make sure the system is regularly tested during development. By running tests we increase system stability and robustness, which in turn increases the usability of the product.

There are several approaches when it comes to testing of software systems. The team looked into unit testing and functional testing. Unit testing is a form of white box testing and when unit testing Java based applications it is very common to use JUnit [**?**]. With JUnit you can easily test every function on the program. The team decided that functional testing made the most sense for this project, this approach is described in its own section.

At the end of each sprint tests will be run to see if the applications meets the specifications. Tests will also be run at the end of the development process. In addition to this the customer will also participate in the testing, and deliver the application to third-party users so the application will be tested by users as well.

4.1 Test methods

Testing is done with several approaches, this helps with structuring the work involved with testing. The team has decided to use functional testing and acceptance testing in order to test the server and the android application during development.

Functional testing

Functional testing is tests that control the functionality of the software. Internal classes and methods is tested indirectly. This is a type of Black Box Testing [**?**]. The team will pick several classes and methods that the application should complete and some that it should not, but handle in a satisfying way. This is done by asking the application to do something and compare the result with what was expected.

The following steps are required in order to test the system

1. Identify what features are to be tested

2. Specify input to the system
3. Specify expected output from the system
4. Execute. Feed input data into the system and compare the output with what is expected. If the output is not what was expected there is a bug that needs to be investigated and fixed before the test is run again.

Test table

	Description	**Result**
Server	*Functional tests of server*	
	Server should deliver data in JSON format	
	REST calls that should have a reply	
	Fetch usage data for user	
	Fetch usage data for users friends?	
	Store usage data for user	
Android application	*Functional tests of application*	
	Add device	
	Remove device	
	Add power used for device	
	Add power produced for device	

Table 4.1: Test table

Planned tests

Test 1	Get usage data in JSON for user 1
Test 2	Store usage data in database for user 1
Test 3	User can read the amount of power a device has produced
Test 4	User can add a device
Test 5	User can delete a device
Test 6	User can read the amount of power a device has consumed
Test 7	User can type in the amount of energy produced by a device
Test 8	user can type in the amount of energy consumed by a device

Table 4.2: Planned tests

Acceptance testing

In addition to the functional testing the customer will also participate in the testing process. This is done because the customer has insight in what he wants and can provide a fresh set of eyes to spot problems. The customer will also deliver the application to third party users so the application can be tested in real life circumstances.

Acceptance testing will be done throughout the development process, but is also a part of the final testing.

5 Development environment

The team has in conjunction with the customer decided which technologies to use. These technologies are used as an aid in the development. To solve the task and completing the project, the team have adopted the

5.1 Code management and versioning

GitHub

The team have decided to use Git with GitHub [?] instead of alternative tools like SVN [?] and Mercurial [?]. The team chose GitHub because most of the team members have extensive experience with Git from previous projects. In addition, it was requested by the customer.

By also taking the customer in this matter into consideration, we would be more able to give the customer a better overview of the project and also to easily integrate our application into the customer's existing project. As a consequence, both the team and the customer would be updated on the last version of the project and the team would be more likely to receive relevant feedback.

In addition, GitHub offers very good support for team development with advanced functionality like branching and version control to make contributions from several developers easy to manage. GitHub also works as an external backup for the project and some of the documentation making data loss less likely.

Development in Android

For the development in Android, the team has decided to use Android Studio [?], which is an integrated development environment based on IntelliJ [?], that is specifically designed for development on the Android platform.

Despite the fact that some the team members already had experience with Android Studio's counterpart Eclipse ADT [?] and few of us had any experience with Android Studio, the team wanted to give it a try, as it appeared to have more features, such as a faster compiler, better search function and auto completion of code, than Eclipse ADT.

As the project has progressed, it has become clear that the learning curve has not been as steep as expected. Neither have we encountered any problems that we did not think also would have been present if we had chosen Eclipse ADT.

Integrated Development Environments (IDEs)

The main IDEs for Java programming are Eclipse [**?**], NetBeans [**?**] and IntelliJ. Most of the team members has previous experience with Eclipse for both Java and Android development, but IntelliJ has the best support for Android development, which is further explained in the section above.

The team has decided to keep the back-end part optional. Both IDEs have tools for checking the code quality and code conventions, and compatibility between them will presumably be a minor issue that may be resolved with Git.

Maven

Maven [**?**] is an intelligent project management, construction and application tool. Maven is a tool that can be used to build and manage Java-based projects. The goal of Maven is that developers should understand the complete state of a development project in the shortest amount of time possible.

Gradle

Gradle [**?**] is in many ways similar to Maven. It is used to build and manage the Android part of the project. The reason we use gradle instead of Maven for the entire project is because gradle is well integrated in the Android community and most of the libraries needed for the android application is supported through gradle.

Code convention

The customer has requested that the team should use Java [**?**] and Android's code conventions [**?**].

5.2 Libraries

Dropwizard

Dropwizard is a lightweight collection of tools used to set up and host a server. Dropwizard itself is new and not much used yet, but the tools in the collection are popular and well maintained. The advantages of using Dropwizard instead of using the tools by themselves

is that everything is configured to work together and it synergizes well. Dropwizard and all its parts is under the Apache 2.0 licence [**?**].

Jersey/JAX-RS

Jersey provides the REST(Representational State Transfer) service of our server. REST is the API(Application Programming Interface) that provides programmers with an interface to communicate with the server through basic HTTP commandos like GET, POST, PUT and UPDATE. For the application communication will be done using the JSON format described under jackson.

Jackson

Jackson is the tooled used to translate data to JSON before it's sent to the client. JSON is used as a textual representation of the objects and is a standard way to store and transfer data between different components.

Jetty

Jetty is an integrated HTTP server that allows the server application to run by itself without the need for any complicated and bloated web service like apache or tomcat. This makes setting up and running the server application more user friendly and faster.

JDBI

JDBI is a library that exposes relational databases and makes them more modular and flexible. This library removes many of the problems related to database usage and communication.

Volley

Volley is a library that will handle network requests on the android device. It is a easy-to-use, lightweight library that makes communicating with the server simpler and more error proof. Volley is made and maintained by Google and is licensed under apache 2.0.

aChartEngine

aChartEngine [**?**] is a popular android library for creating graphs. In the project this library will mainly be used for line graphs, for displaying detailed power usage data, and pie charts for comparing power consumption between devices. This library is licensed under apache 2.0 [**?**]

Facebook SDK

TBA (Lars)

5.3 Documentation

Google Drive

The team has decided use Google Drive [?] for most of the temporary documentation because it offers an easy way to create documents that is accessible for editing and collaboration simultaneously for the entire team. Here, the team will mainly keep documents like meeting agendas, information gathering, input from the customer and the supervisor, product concepts and other documents waiting to get put into the final report.

In addition, Google Drive serves as an external backup, making it highly unlikely that important documentation will be lost.

LaTeX

LaTeX [?] is an advanced typesetting system for document production, widely used in academic institutions. The system focuses on allowing the author to focus mainly on the content of the document, and less on the design and document layout.

The team has chosen LaTeX for the final report instead of other word processing programs like Microsoft Office or Google Drive, because LaTeX makes it easier to keep track of references and to maintain the appearance of large documents.

In addition, LaTeX provides the ability to break documents into smaller parts and for several team members to simultaneously make changes to the report.

JavaDoc

The team will use JavaDoc [?] to document the code. The main advantages of using JavaDoc is that it can be integrated with the Java code and linked directly with classes and methods making it easier for others to use the code.

Yodiz

The team's main requirements in a Scrum tool is that it easy to use and effective for project management, has Git integration, and that can create charts for documentation purposes. With this tool the team should be able to get an overview over the projects progress and the team's performance.

The team chose Yodiz [?] over other Scrum tools because it provides free accounts for educational purposes and it suited the team's requirements.

5.4 Communication

Google groups

The team has chosen Google groups [**?**] as an arena for casual communication. Google groups includes a shared e-mail-list, ensuring that none of the team members unintentionally gets left out of the information stream. E-mail has the advantage of being an asynchronous communication, well fitted for the team's purposes. The team mainly use this e-mail-list to communicate between the meetings and to share information that is urgent or from external sources.

E-mail

The team has primarily used e-mail as communication protocol when in dialogue with the customer and our supervisor.

6 Requirement specification

6.1 Functional requirements

FR1. Device control/overview

FR1.1. The user should be able to monitor his energy usage

- FR1.1.1. The user should be able to see which devices in his house that actually use electricity, real time.
- FR1.1.2. The user should be able to see how much electricity each device in his house use on an average basis, on a monthly basis and on a yearly basis.

FR1.2. The user should be able to turn devices on and off from within the application

- FR1.2.1. If a device's power consumption becomes too high, the user should be able to turn it off.
- FR1.2.2. The user should be able to turn on devices, for instance the heat in his house, from the application. An example of such a use can be if the user wants the heat to turn on before he comes home.
- FR1.2.3. The user should have the ability to schedule when he want a device to consume power. If he want the heat to be off while he is at work, for instance.
- FR1.2.4. The user should have the ability to specify a maximum of power consumption per day for a given device and be notified if the device's power consumption reaches that limit. This functionality will not be available for critical devices such as refrigerators.

FR1.3. The user should be able to monitor and control his energy production

In the same way the user can see his energy usage. Be able to add devices producing power and control how much they produce.

FR2. Power usage

FR2.1. The application should show the user graphs over his total usage real time, the last week and/or the last month and/or year.

Here, the user will get the alternatives to how he want to display the data, and will get the data accordingly.

FR2.2. The user should be able to view his power consumption from one device over a given time

The user will get alternatives in the same way as when showing total usage.

FR2.3. The user should get information about what he can do to lower his power consumption from a ”best power saver of the month”

Which power saver that is the best is decided by all the users through rating.

FR3. Power savers

FR3.1. The user will get a list over the most used and/or best rated actions towards saving power.

The rating will be by all of the users.

FR3.2. The user will get a list over what actions he has taken.

The user will also be able to ”checkout” actions he want to perform in the future. The application should part what actions the user has already done, and what he wants to do at a later stage.

FR3.3. The user should be able to add actions

If an action is not present in the list, the user should be able to add the action to the list.

FR4. Profile

FR4.1. The user should have its own profile, with all the information about him

The user should be able to choose to remain anonymous.

FR4.2. The user should be able to add a ”wish list” with things he wants to save money for

This list will be connected to how much money the power the user has saved represents.

FR4.3. The user should be able to choose what currency to use when converting from power saved to money.

FR4.4. The user should also be able to choose what language he wants the application to be in.

FR5. Social

FR5.1. The user should be able to share his power savings on Facebook.

FR5.2. The user should be able to connect to other friends, also using this application

FR5.3. The user should be able to compare his power usage and savings with friends, other similar people, or people in his area

FR6. Other (Not specified yet)

FR6.1. The user should be able to choose a environmental profile that he wants to follow

FR6.2. The application should involve gamification in some way

- FR6.2.1. Ideas so far is that the user gets point as a result of how much he saves.
- FR6.2.2. The user can get achievements when he has reached his goals, or when he has taken an action.
- FR6.2.3. The user can get points/achievements when he has shared his savings with other people through social media.

Non-functional requirements

NFR1: The application should be easy to use.

To ensure familiarity for Android users, the user interface should follow standard Android specifications for graphical user interfaces.

Furthermore, the application should be able to function without the purchase of new hardware. The application will be designed for use with wireless power consumption readers and a small server to be positioned in the home of the user, but the basic operation should

be possible to perform even without these tools. The most basic mode operation is logging of data that the user inputs manually after reading standalone power consumption meters.

More generally, the user interface should be easy to use, also for people whom are not accustomed to Android devices. In addition to following Android specifications, this will be achieved by keeping the interface as plain and intuitive as possible.

NFR2: Installation guide and documentation

The system should come with a comprehensive guide to using the system. This should include documentation for operation of the application along with a guide to what the rest of the system operates. As the team will not be developing hardware for metering and aggregating data from homes, the main focus will be on the application and the team's central server. The documentation will contain an outline of the intended architecture for the rest of the system.

Testing non-functional requirements

Testing the non-functional requirements is a process quite different from testing that functionality has been implemented correctly.

The only non-functional requirements the team will be testing is the usability of the Android application. These test will primarily be tested by individuals that are not a part of the development team. The team will have a user test, or a multiple of such tests, to observe how people use and experience the application without interference from the team.

After testing the users will be able to rate different aspects of the application and comment on usability. Detailed test plans for testing usability can be found in chapter 3.x Research: Testing.

- The user interface should follow standard Android specifications.
- The applification should not need a lot of extra material to get started (Power devices, own server).

7 Sprints

check whether all necessary sections are there and add those (if any) missing

7.1 Sprint 1

Goals for the sprint

The team's first sprint had several diverse goals, which is common for the start up phase of any project. The goals can be divided roughly into four different categories: Project management, product specification, early product design and documentation.

Project management

The first sprint was mainly about organizing how the team wanted to work, what the team wanted work with, and discussing how to reach these goals.

The first sprint contained a lot of management and planning. The team mapped its resources and tried to get an overview of the task at hand. The team did a clarification of the expectations to the project. All team members agreed to aiming for a

a good grade

and at the same time learn more about both group work and android development. Suitable working hours for all group member was also discussed and found. Additionally, tools and technologies had to be chosen for the initial superficial system architecture. The rest of the work in management consisted of allocating responsibilities, drawing up a rough project plan and coming up with some possible concepts for the system.

Product specification

The functions and design system to be developed turned out to be very open for interpretation. The customer was a bit vague in describing the product at first, but he did have some

concrete requirements: The product should include an Android app, and the app should raise awareness on power consumption in private homes. The customer also communicated that the app should be integrated with social media somehow. We turned our ideas and the customer's wishes into a draft of the requirements specification that the customer later approved.

Product design

With a specification and some tools chosen beforehand, the team started looking for open source technologies that could be utilized in the development of the product. After sketching the architecture of the entire system, including the app and supporting systems, the team decided on a set of libraries and tools as aid in our future development.

Documentation

To make sure that documentation was not neglected early on, the team started working on it immediately. A report template was created using LaTeX so that the team easily could update the documentation continuously.

Add subsection about the execution of this sprint, how we want to reach our goals

Add this under subsection of Administration

Selection of tools

As part of the teams project management, the team needed to agree on what tool sets to use during the project.

Election process of Scrum tools

The team started by researching what tools that satisfied our Scrum criterias 3.2. Some of the tools that was suggested was iceScrum [**?**], Yodiz, ScrumDo [**?**] and Jira [**?**].

The team has previous experience with iceScrum. After a discussion, the team came to the conclusion that iceScrum was not to be used again and that an alternative was needed. The reason why this was decided is that iceScrum must be deployed on a server,

and it is restrictive in how to use it.

Some of the tools was professional but optimized for enterprise use. Others lacked professionalism. After having reviewed the different cloud-based Scrum tools, the team ended up with Yodiz as the best alternative. The winning arguments was a good sprint board, the ease of use, many positive reviews and the pricing, which was free for educational purposes.

The project was divided into epics, user stories and tasks. Epics are user stories that are more general, and that demand at least twelve hours of work to be completed. Each

epic consists of all the user stories that explain the usage of the epic in more detail. The user stories will have time estimates discussed during sprint planning meetings. The sprint backlog will consist of the user stories that the team has decided to handle.

A user story is given smaller tasks that are the different technical aspects of the story that needs to be done.

During a sprint the progress is handled in a Scrum board. The Scrum board has four columns. One for the user stories, new tasks, tasks in progress, and completed tasks. All the tasks for the user story are in the new column and on the same row as it's user story, then dragged appropriately to the correct column during the sprint, and updated with time usage.

Choice of IDE

For android development Android Studio from Google was chosen. I has very good integration with GUI development i Android, and also it builds on the IntelliJ IDE of which several of the team members had prior knowledge. As Gradle was being used in building the project, it was left open for team members to chose IDE when working on the server implementation of the system.

Results

The team made several key decisions during sprint 1 regarding project methodology, tools and architecture. The customer was very pleased with the teams ideas and concepts for the app, and approved of the temporary specification. The team members deemed sprint 1 very successful.

7.2 Sprint 2

Goals for the sprint

In the second sprint the team focused on providing the customer with a working prototype to further develop the requirements specification. Furthermore, a working prototype might help the customer see how the team envisioned the app and its functions. Lastly, work on the server application began to form the foundation a framework of functions that the android app could use.

Design

Before implementing the specification that was approved by the customer, the team built prototypes of the design in sketching programs to ensure a consistent and clean design throughout the app.

Development

As the GUI design neared an end, the team started implementing the features the customer specifically asked to have included in the first iteration of the app.

Server

The groundwork for the apps server was put down in sprint 2. Using Dropwizard the development team started the implementation of the functions available to the app.

Results

Improper use of Scrum tool

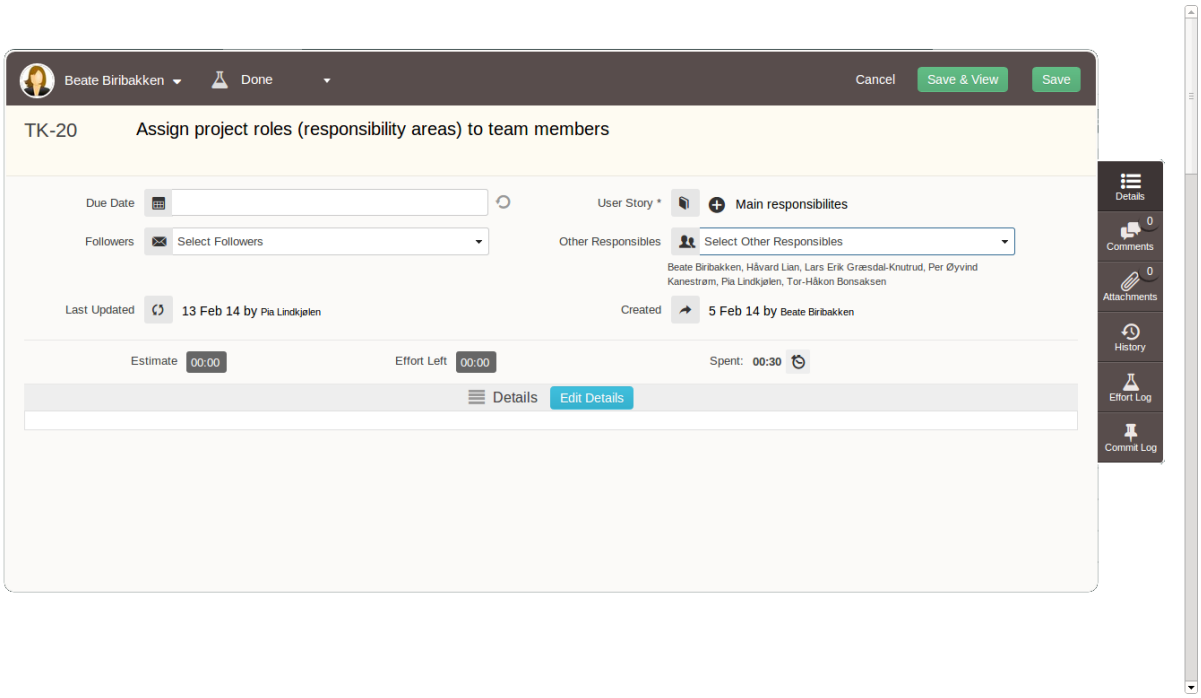
As mentioned in section 7.1, the team used a lot of time on deciding on which Scrum tool to use for the project management. Although our choice fell on Yodiz, the team was in lack of any previous experience with the tool, and despite the team's efforts to get acquainted with the tool, a misunderstanding arose and was not discovered until the end of the second sprint.

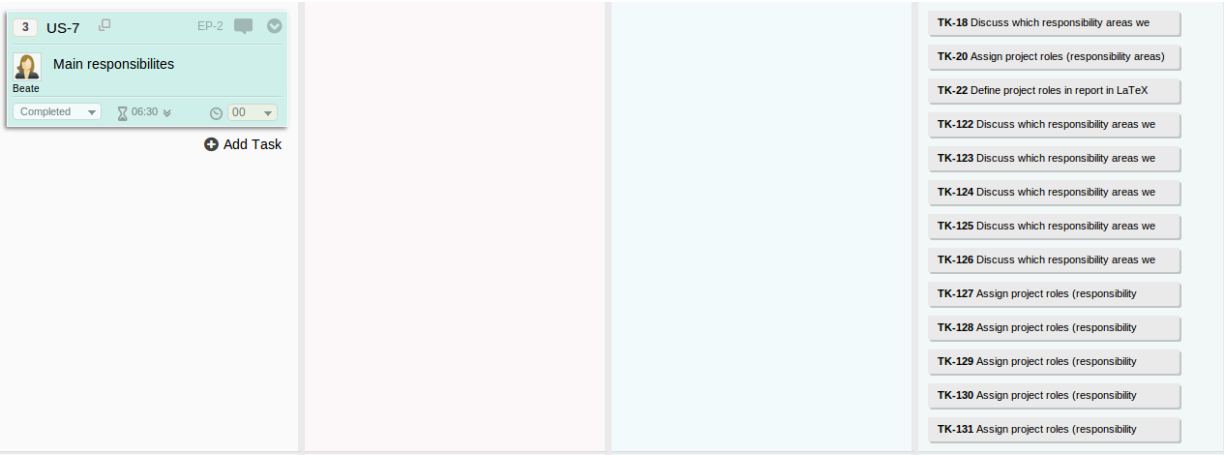
The misunderstanding, displayed in figure **??**, was that the team assumed one could add multiple individuals as responsible on a particular task, while Yodiz' functionality only assigned the time spent to the individual that either created the task or was assigned as owner of the task.

As a result, it appeared as if only singular individuals performed the tasks, even though the entire team in reality was participating, which was also reflected in the burndown charts and the generated gantt diagram.

To sort out this issue, the team went through old meeting reports and timesheets to figure out which members of the team that had actually participated on the particular task, and added new tasks and the time spent to the members that at the time had not recorded this information, as shown in figure **??**.

This issue was unfortunate, but not insurmountable, and also not a critical issue for the overall progress of the project.





Working away from the team

One of the team members, Lars Erik, had to make a sudden trip abroad due to death in the family. He stayed away from the team for a total for 12 days. Prior to departure the team divided the tasks with a lighter workload assigned to Lars Erik, as described in the

risk analysis. Even with the redistributed workload the team and Lars Erik in particular found the distance and time difference to be a bigger problem than expected. Some work was left undone and had to be picked up at the end of the sprint or pushed to the next sprint. The team learned that the impact of such leaves of absence should probably be overestimated rather than underestimated in the eventuality of another event.

General results from sprint 2

The customer was happy with progress on the prototype, and provided rich and detailed feedback on how the app should be improved for the next iteration. The sprint was deemed successful by the team members.

7.3 Sprint 3

Goals of the sprint

Execution of the sprint

Development

Testing

Design

Project Administration

Results

A Use cases

Devices

ID: 1	**Overview of devices**
Actor	User
Description	The user should be able to see which devices in his house that actually use or produce electricity, real time.
Preconditions	1. Devices are registrered 2. Devices are connected to energy measurement device 3. Devices are active Depends on x.
Input	Registrered energy usage
Postconditions	App displays list of all active devices
Basic flow	1. User taps ”Devices”-tab in menu 2. All active devices are displayed in list
Alternative flow	1.1 User taps wrong menu tab. Return to basic flow 1.

Table A.1: Textual use case 1

ID: 2	**Display energy usage per device**
Actor	User
Description	The user should be able to see how much electricity each device in his house use on an average basis, on a monthly basis and on a yearly basis.
Preconditions	1. Devices are registred 2. Devices are connected to energy measurement device Depends on x.
Input	Registered energy usage
Postconditions	App displays sector graph where each area represents a device
Basic flow	1. User taps ”Usage”-tab in menu 2. User taps ”Show devices” 3. User wants to display graphs for specific time period 3.a User choose ”Last month” in drop down menu (default view) 3.b User choose ”Last year” in drop down menu
Alternative flow	3.1 No electricity usage is registred. Average consumption is displayed. 3.1.a User choose ”Last month” in drop down menu (default view) 3.1.b User choose ”Last year” in drop down menu

Table A.2: Textual use case 2

ID: 3	**Turn off device**
Actor	User
Description	User should be able to turn off devices from the app.
Preconditions	1. Device is connected to app. 2. Device is not a critical device. 3. The device must be functional. 4. The device must be turned on. Depends on x.
Input	User presses ”Turn off”-button
Postconditions	Device is turned off.
Basic flow	1. User taps ”Devices” 2. User taps the specific device he wants to turn off 3. User turns off device by pressing ”Turn off”-button
Alternative flow	2.1 User taps wrong device. Return to basic flow 2. 2.2 User turns off wrong device. 2.2.a User turns on the wrongly selected device. See use case scenario 4. 2.2.b User turns off the correct device. Return to basic flow 3.

Table A.3: Textual use case 3

ID: 4	**Turn on device**
Actor	User
Description	User should be able to turn on devices from the app.
Preconditions	1. Device is connected to app. 2. The device must be functional. 3. The device must be turned off. Depends on x.
Input	User presses ”Turn on”-button
Postconditions	Device is turned on.
Basic flow	1. User taps ”Devices” 2. User taps the specific device he wants to turn on 3. User turns on device by pressing ”Turn on”-button
Alternative flow	2.1 User taps wrong device. Return to basic flow 2. 2.2 User turns on wrong device. 2.2.a User turns off the wrongly selected device. See use case scenario 3 2.2.b User turns on the correct device. Return to basic flow 3

Table A.4: Textual use case 4

ID: 5	**Specify a maximum of power consumption**
Actor	User
Description	The user should have the ability to specify a maximum of power consumption per day for a given device.
Preconditions	1. Device is registered 2. Device is not a critical device 3. Device is connected to app 4. Device is turned on Depends on x.
Input	User registers maximum kwh/day
Postconditions	Device is turned off when maximum power consumption limit is reached.
Basic flow	1. User taps ”Devices”-tab in menu 2. User taps specific, correct device 3. User sets maximum power consumption limit
Alternative flow	2.1 User taps the wrong device. Return to basic flow 2. 3.1 User gives invalid input 3.1.a Error message appears 3.2.a User inputs valid input
Comments	Not available for critical devices

Table A.5: Textual use case 5

ID: 6	**Schedule device's consumption**
Actor	User
Description	The user should have the ability to schedule when he want a device to consume power. If he want the heat to be off while he is at work, for instance.
Preconditions	1. Device is connected to app 2. Device is registered 3. Device is turned on 4. User has scheduled a specific device 5. Device is not a critical device Depends on x.
Input	Deviceid, time period
Postconditions	The specific device is turned off in the chosen time period.
Basic flow	1. User taps "Devices"-tab in menu 2. User taps specific, correct device 3. User choose a time period
Alternative flow	2.1 User taps wrong device. Return to basic flow 2. 3.1 User chooses an invalid time period. 3.1.a User receives error message 3.1.b User inputs valid time period.
Comments	Not available for critical devices

Table A.6: Textual use case 6

ID: 7	**Notification if the device's power consumption reaches maximum limit**
Actor	System
Description	The user should have the ability to specify a maximum of power consumption per day for a given device and the system should notify the user if the device's power consumption reaches that limit.
Preconditions	1. Device's maximum limit is specified 2. Device is not a critical device 3. Device is connected to app 4. Device is turned on 5. Device is registered 6. The limit has been reached Depends on x.
Input	Boolean value that checks whether the usage is higher than the specified maximum kwh/day.
Postconditions	System sends notification to user about the device in question. User receives notification.
Basic flow	1. System sends notification to user about the device in question. 2. User receives notification.
Alternative flow	
Comments	Not available for critical devices

Table A.7: Textual use case 7

ID: 8	**Adding Power Savers**
Actor	User
Description	The user should be able to add his own Power Savers, steps that have been taken in order to decrease power consumption.
Preconditions	1. User is logged in. Depends on x.
Input	Description of users Power Saver.
Postconditions	The Power Saver is added to the public database and is visible for other users.
Basic flow	1. System sends notification to user about the device in question. 2. User receives notification.
Alternative flow	

Table A.8: Textual use case 8

ID: 9	**Choosing anonymity**
Actor	User
Description	The user has a profile connected to his Facebook-account, but he should be able to choose whether the data he aggregates is to be used for comparison against others.
Preconditions	1. User is logged in. Depends on x.
Input	User's choice.
Postconditions	Anonymity settings based on user's choice.
Basic flow	1. User taps profile tab. 2. User taps anonymity button. 3. User is prompted with the choice to stay anonymous.
Alternative flow	

Table A.9: Textual use case 9

ID: 10	**Show a wishlist of items**
Actor	User
Description	The user should be able to add items with a name and a given price. He should later be able to track his savings up against the values of the items he has entered in the list.
Preconditions	1. User is logged in. Depends on x.
Input	Name, price, and an url (optional) for an item the user's wishes for
Postconditions	A list of items with their cost, followed by how much
Basic flow	1. The user taps the profile tab 2. The user taps the wish list button 3. The user adds an item with name and cost 4. The user views the list with a progress bar representing his energy savings
Alternative flow	4.1 The user does not have any consumption data, and therefore no progress is shown on the item
Comments	

Table A.10: Textual use case 10

ID: 11	**Showing power savings in desired currency**
Actor	User
Description	The application should project savings in consumption in a currency chosen by the user.
Preconditions	1. User is logged in. Depends on x.
Input	Desired currency
Postconditions	All savings in the application are projected in the chosen currency
Basic flow	1. User taps profile tab 2. User taps currency button 3. User chooses desired currency from lists
Alternative flow	3.1 User cannot find desired currency in list
Comments	

Table A.11: Textual use case 11

ID: 12	**Changing the display language of the application.**
Actor	User
Description	The user should be able to change the language of the application.
Preconditions	1. User is logged in. Depends on x.
Input	Desired language of operation.
Postconditions	The application uses the chosen language.
Basic flow	1. User taps the profile tab 2. User taps the language button 3. User selects desired language from list
Alternative flow	3.1 User cannot find desired language in list
Comments	Only a limited amount of languages will be available upon release.

Table A.12: Textual use case 12

ID: 13	**Sharing power savings of social media like Facebook and Twitter.**
Actor	User
Description	The user should be able to share any savings in monthly consumption with his friends on social media.
Preconditions	1. User is logged in 2. User has connected social media accounts in question Depends on x.
Input	Users credentials for accounts in question.
Postconditions	The data is shared on the selected accounts.
Basic flow	1. The user taps the social tab 2. The user taps share 3. The user selected the data he wants to share, and what accounts to share it on 4. Preformatted messages with the given data is posted to the social media accounts
Alternative flow	4.1. The social accounts do not allow the application to post
Comments	

Table A.13: Textual use case 13

ID: 14	**Connecting with friends whom are using the application**
Actor	User
Description	The user should be able to find and connect to his friends through Facebook.
Preconditions	1. User is logged in. Depends on x.
Input	User's facebook credentials.
Postconditions	A list of friends currently using the application.
hline **Basic flow**	1. The user taps the social tab
Alternative flow	1.1. The user does not have any friends currently using the application
Comments	

Table A.14: Textual use case 14

ID: 15	**Comparison of data and power savers**
Actor	User
Description	The user should be able to compare his power usage and power savers with friends or other people in the area
Preconditions	1. User is logged in 2. User has logged energy consumption data Depends on x.
Input	The users energy consumption data or power savers.
Postconditions	Application displays a list of friends and their respective energy consumption. A list of energy savers implemented by friends is also available.
Basic flow	1. The user taps the social tab 2. The user taps the compare button 3. The system retrieves the users friends and graphs their consumption against his own logged consumption 4. Power savers from friends and others is also available
Alternative flow	3.1. The does not have any friends currently using the application
Comments	

Table A.15: Textual use case 15