IT2901 INFORMATICS PROJECT II

Group 16 - SINTEF_Energy

Beate Baier Biribakken

Tor-Håkon Bonsaksen

Lars Erik Græsdal-Knutrud

Per Øyvind Kanestrøm

Håvard Holmboe Lian

Pia Karlsen Lindkjølen

Spring 2014

Abstract

With the ever rising consumption of electricity in households and people becoming increasingly conscious about the environment, a market has evolved for systems that monitor power consumption. There are systems on the market that allow monitoring of devices and total usage, but none of them focus on sharing experiences with others. Awareness around, and encouragement to reduce power consumption might be aided greatly if users could share their experiences and progress on social media, like Facebook.

This report describes how a team of six students at NTNU, developed a solution for SINTEF Energy to resolve this problem. It includes both detailed architecture of the system, the development process, and how the team envisions further development of the system. The Android app Wattitude was developed to give users an easy platform to monitor and control their power consumption. The app also gives users the ability to create tips for saving energy, and share both these and their consumption data with their Facebook friends. A server for data synchronization was also developed.

Contents

Figures

Tables

1 Introduction

The objective of this chapter is to give an introduction to the parties involved in the project. First, the assignment will be described, then the customer and the team will be introduced. It also presents an overview of the team organization, as well as the team's available resources, and areas of responsibility.

1.1 About the assignment

The project is a part of an ongoing research program, CoSSMic [1], which focuses on storage of 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 their neighborhood. The research program aims to develop the tools needed for sharing energy.

The project's objective was to create an Android Mobile app, making it possible for the user to monitor his[1] energy usage and energy production.

Similar apps exist on the market, some with complete systems to measure and control users' energy consumption. What they have in common is that they are either expensive, or do not provide all desired functionality. More on this may be found in section 2.2.

The customer had a clear idea on the concepts to include in the app, but no specific requirements. The customer provided a figure illustrating the most important concepts, which was used as a foundation when the requirements were specified, as described in section 4.1.

1.2 About the customer

Our customer is the social inclusion technology research group at SINTEF [2]. SINTEF is one of the largest independent research organizations in Scandinavia, founded at the Norwegian Institute of Technology in 1950. Since then, SINTEF has grown to employ a staff of 2100, whereof more than 50 percent hold doctorate degrees. Their main commodity is research-based technology and auxiliary services. Areas the research group work within includes health care, cultural heritage and smart energy.

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

1.3 Team organization

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

1.3.1 The team members

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

Beate Baier Biribakken
Beate has worked at the IT-companies Student Media AS [3] and Sportradar AS [4] 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.

Tor-Håkon Bonsaksen
Tor-Håkon has a trade certificate in data electronics and much experience with web development through extracurricular activities within the groups dotKom [5] and Casual gaming [6]. This includes knowledge of Python, Django, HTML, CSS and JavaScript. He also has some experience with Android development from personal projects and with Java through school.

Lars Erik Græsdal-Knutrud
Lars Erik has experience with Java, C#, C++ and SQL through his ongoing education. As an internal system developer for Orakeltjenesten Dragvoll he gained experience with PHP and server environments.

Per Øyvind Kanestrøm
Per Øyvind is a GNU/Linux user. He has experience with PHP/Symphony2 and Android development from personal projects. 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 operating system.

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

1.3.2 Main responsibilities

The team divided the project work into several areas of responsibility. All team members had a main responsibility and were in charge of all work within their area was delegated and carried out. The entire team was responsible for completing the project, and contributed with both the app development and writing of the report. The division of responsibilities is shown in table 1.1.

Role	**Description**
Project leader: Pia	Keeping the customer and the team updated and monitor the project's status. Includes typical administrative tasks, such as room booking and communication with supervisor and customer.
Scrum-master: Lars Erik	Make sure that the Scrum-process goes as smooth as possible, that necessary documentation is provided and keep track of the project's progress. Includes administrative tasks, such as generating burn down charts, chair Scrum meetings and add tasks to the backlog.
Development: Tor-Håkon	Keep track of the technological development progress, make sure it is on schedule and take necessary preventive actions.
Report: Beate	Monitor the report's progress, spell check and review content.
Testing: Håvard	Make sure that the code is properly tested in order to detect errors and bugs.

Table 1.1: Description of the main responsibilities and which team member responsible

1.4 Available resources

The team had several available resources throughout the project, including work hours, our institute supervisor and the customer representative. To handle these resources, deadlines were set for important events, enabling the team to create a more satisfying product.

1.4.1 Milestones

The assignment and team was assigned January 20th. The initial customer meeting took place January 24th, where the assignment was presented and discussed. The final presentation of the app was held for the customer May 23rd at SINTEF. Table 1.2 gives an overview of all the important dates the team dealt with throughout the project.

Deadlines	Name	Description
07.02.14	Sprint end 1	First sprint is ended.
09.02.14	Report - preliminary version	Delivery of report outline for IT2901.
21.02.14	Sprint end 2	Second sprint is ended.
07.03.14	Sprint end 3	Third sprint is ended.
16.03.14	Report - mid-semester version	Delivery of report draft for IT2901.
19.03.14	Oral presentation of project	Present project for IT2901 students.
21.03.14	Sprint end 4	Fourth sprint is ended.
21.03.14	Feature freeze	No changes in the specification after this date.
23.03.14	Peer evaluation	Evaluation of other IT2901 group's mid-semester report.
04.04.14	Sprint end 5	Fifth sprint is ended.
02.05.14	Sprint end 6	Sixth sprint is ended.
16.05.14	Sprint end 7	Seventh sprint is ended.
16.05.14	Code freeze	All further development is stopped.
23.05.14	Product delivery to customer	Presentation for SINTEF.
30.05.14	Sprint end 8	Eight sprint is ended.
30.05.14	Report - final version	The finalized report was delivered.

Table 1.2: Milestones and important events

1.4.2 Available hours

Each sprint had a duration of two weeks, excluding the period from April 4th to April 21th, when the entire group left for a school field trip to China. To compensate for the days missed because of the school field trip, it was decided to increase the amount of work hours from 20 to 25 as of sprint 2 up to and including sprint 8. Table 1.3 lists the project's available hours and is based on all six team members spending at least the required hours on the project every week.

Period	**Dates**	**Days**	**Hours**
Sprint 1	January 27 - February 07	10	240
Sprint 2	February 10 - February 21	10	300
Sprint 3	February 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
Sprint 7	May 05 - May 16	10	300
Sprint 8	May 19 - May 30	10	300
Total		**80**	**2340**

Table 1.3: Available hours

1.4.3 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.

1.4.4 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. He may be contacted by e-mail at babak.farshchian@sintef.no.

2 Preliminary work

The objective of this chapter is to clarify the first phase of the project, which involved a lot of planning and research. In this chapter, the domain knowledge the team had to acquire before undertaking the project is discussed. Then, similar and existing solutions already available on the market is explored, and their advantages and drawbacks will be reviewed. Lastly, an overview of the different technologies used to develop the app is given.

2.1 Domain knowledge

The customer's main focus was to raise awareness about energy consumption. To develop an app for this purpose, the team needed a deeper insight in how to measure power consumption in private households and what similar solutions that already existed on the market.

To learn more about power consumption and how to reduce it, the team contacted electrical power providers. Their strategies to raise awareness included campaigns, which was not relevant for our purpose, and providing tips to the consumers. It was concluded that the app should provide users with tips to reduce electricity usage. To further engage the user, the app should include the ability to add tips to a personal check list, and let the user check off tips when they have been performed.

Another field of research was hardware the team could use to measure power consumption in real time. Before the preliminary work was carried out, the customer emphasized features that would distinguish the app from the similar solutions on the market. By allowing the app to control individual devices like light bulbs and heaters, and measure the power consumption automatically, the app could potentially become a unique segment of the market. This functionality was dropped due to time constraints, but the app was developed with future integration of said functionality in mind. More details on this in chapter 8, along with more about how the team envisions further development.

2.2 Existing solutions

To get a full impression of what functionality our product should have, the team did some research on similar solutions that already existed on the market. The research was based on the product requirements the team worked out with the customer, resulting in a list of the most relevant solutions.

This list contains a brief overview of the functionalities and drawbacks of existing solutions, summarized in table 2.1. The team used this study in alternative solutions as an inspiration for functionality and features to implement in the project app.

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 2.1: Comparison of existing solutions

2.2.1 NTE miniSolo energydisplay

The NTE miniSolo energydisplay [7] 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 an overview of how much money the power usage amounts to. However, the miniSolo energydisplay has some drawbacks: It is linked to a proprietary device, and it has no Android app. These drawbacks made the solution incompatible with the team's project.

2.2.2 Theowl

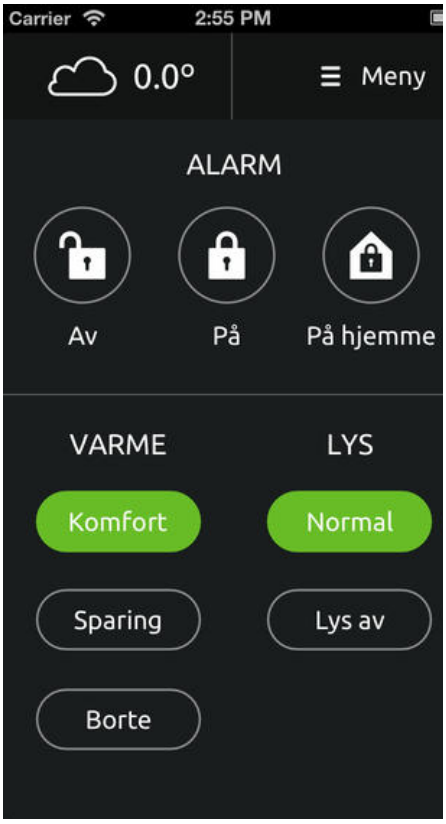
Theowl [8] mainly focuses on temperature control. This was a drawback because the team would like to focus on other properties, such as controlling individual devices. The product does not have any official support in Norway, and it is not within a reasonable price range.

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 project.

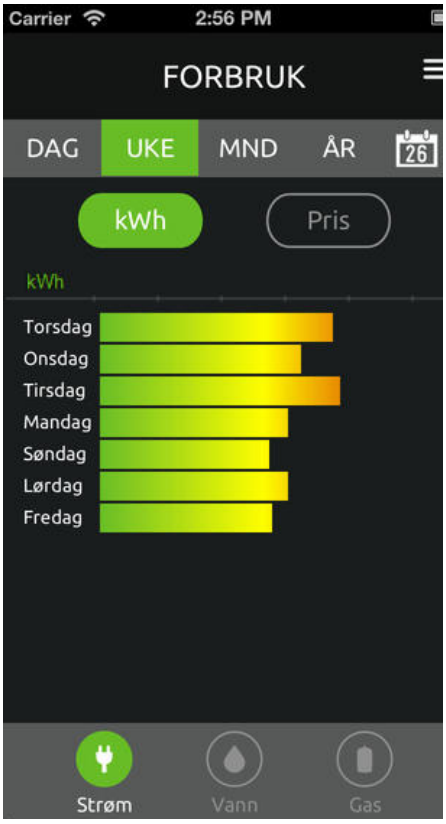
2.2.3 Smartly

Smartly [9] allows the user to monitor and control things like temperature and lighting in the house, as shown in figure 2.1(a). It also has an overview over the total power consumption, as shown in figure 2.1(b).

Smartly has several interesting features and concepts, such as being able to turn on and off certain devices, and functionality for displaying money saved. The graphical user interface is also well designed and user friendly. It does however lack the functionality to monitor the power usage of single devices.



(a) Screenshot of Smartly's possibility to remotely control devices



(b) Screenshot of Smartly's consumption overview

Figure 2.1: Screenshots from the Smartly solution

2.2.4 OpenEnergyMonitor

OpenEnergyMonitor [10] is an open source project that allows data collection from power outages. Some of the architecture for collecting data can, with some improvements, be an interesting option to consider. The architecture, shown in figure 2.2, is somewhat similar to what the team imagined using in the project. This solution also has an application for processing, logging and visualizing energy usage.

However, OpenEnergyMonitor offers no automatic collection of data - the user would have to manually collect it. This solution is somewhat hard to set up, and also not very user friendly for the average consumer.

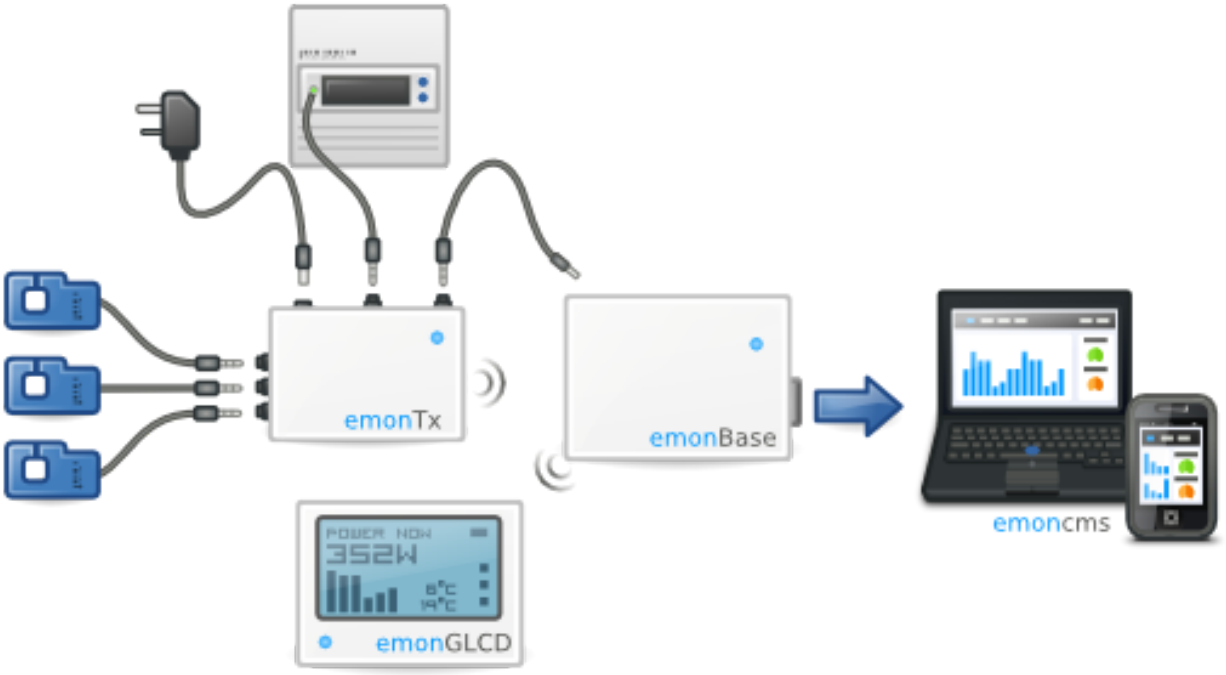


Figure 2.2: OpenEnergyMonitor's system architecture

2.2.5 Efergy

Efergy [11] was considered to be the solution that was closest to what the team wanted to develop. As shown in figure 2.3, it has a nice visual representation of data, and it offers social integration. The architecture is interesting, as it supports measuring the power usage of single devices. There is a local receiver collecting data from the devices and sending it to a server on the Internet. Unfortunately, this solution lacks the ability to monitor and control private power production. It is also beyond a reasonable price range.

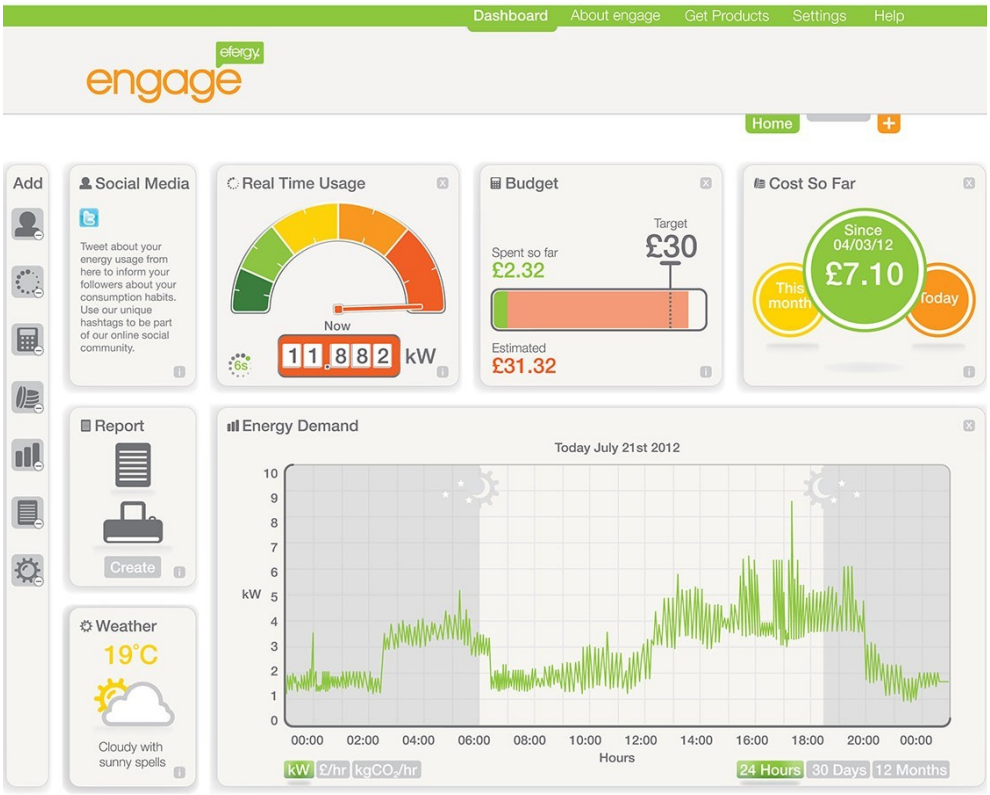


Figure 2.3: Screenshot of the Efergy Engage GUI

2.3 Development environment

The purpose of this section is to give a brief introduction to the technologies and tools used during the development process. Which technologies that was used for the collaboration of code, including which code conventions and design guidelines to follow, was decided in cooperation with the customer.

2.3.1 Code management and versioning

GitHub

The team decided to use Git with GitHub [12] instead of alternative tools like SVN [13] and Mercurial [14], because most of the team had experience with Git from previous projects. It was also requested by the customer and used by the CoSSMic project.

In addition to this, Git offers very good support for team development with advanced functionality like branching and version control to make contributions from several developers easier to manage. GitHub functioned as an external backup for the project as well, making data loss less likely.

Integrated Development Environments (IDEs)

The main IDEs for Java programming are Eclipse [15], NetBeans [16] and IntelliJ [17]. Most of the team members had experience with Eclipse. Despite this, the team chose Android Studio [18] for development. This was because Gradle and Android Studio contained more relevant functionality than its counterparts, such as a faster compiler, and practical auto-completion of code.

The team decided to keep the back-end part optional. All the IDEs have tools for checking code quality and code conventions, and compatibility between them was only a minor issue that could easily be resolved with Git.

Maven

Maven [19] is an intelligent project management, construction and application tool, which was used to build and manage the server parts project. The goal of Maven is to give developers an understanding of the complete state of a development project in the shortest time possible.

Gradle

Gradle [20] is in many ways similar to Maven. It was used to build and manage the Android part of the project. The reason Gradle was used instead of Maven was because it is well integrated with the Android environment. In addition, most of the libraries needed for the Android app was supported through Gradle.

Code convention

This project uses Java [21] and Android code conventions [22], as requested by the customer.

2.3.2 Libraries

Libraries are pieces of code that are integrated into the project. They provide functionality needed and save time by reducing the workload. The libraries used and what they were used for is displayed in table 2.2. All libraries used fall under the Apache 2.0 license [23].

Library	**What it is used for**
Jersey/JAX-RS [24]	Provides the Representational State Transfer(REST) service of our server.
Jackson [25]	Translates data to JSON before it is sent to the client.
Jetty [26]	Is a HTTP server.
JDBI [27]	Exposes relational databases and makes them more modular and flexible.
Volley [28]	Handles network requests on the Android device.
aChartEngine [29]	Displays detailed power usage data through pie charts and line charts.
PagerSlidingTabStrip [30]	Creates tabs for easy swapping between views.
ProgressFragment [31]	Makes it easy to show that a Fragment is loading the view.
android-segmented-control [32]	Segment view for swapping between data resolutions.
android-spinnerwheel[33]	Spinner wheel for date selection.

Table 2.2: Libraries and their application areas

Dropwizard

Dropwizard [34] is a lightweight collection of tools used to set up and host a server. Dropwizard is a relatively new framework, 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 well together. The most notable components of Dropwizard are Jetty, Jackson, Jersey, and JDBI.

Facebook SDK

The Facebook SDK [35] is a collection of functions that allows Facebook's functionality to be integrated into the app. The key functionality for this project, provided by the SDK, is the possibility for the user to log in on their Facebook account.

2.3.3 Documentation

JavaDoc

JavaDoc [36] was used to document the code. The main advantages of using JavaDoc is that it can be integrated with Java code and linked directly with classes and methods, making it easier for new developers to utilize and modify the code.

Yodiz

Yodiz [37] is an agile project management tool with product backlog management, Kanban Scrum board and issue tracking software. The team used this for distributing tasks among team members, monitor project progress and generating Gantt diagram and sprint burn down charts.

Google Drive

Google Drive [38] was used for temporary documentation because it offers an easy way to create documents accessible for editing and collaborating simultaneously. The team used Google Drive for documents like meeting agendas, product concepts, and input from the customer and supervisor.

Additionaly, Google Drive serves as an external backup, making loss of data and documentation highly unlikely.

LaTeX

LaTeX [39] is an advanced typesetting system for document production that is widely used in academic institutions. It allows the author to focus mainly on the content of the document, and less on the design and document layout.

The team chose LaTeX instead of other word processing programs like Microsoft Office, because LaTeX makes it easier to keep track of references and to maintain the appearance of large documents. LaTeX also provides the ability to divide documents into smaller parts. This makes the content and structure of the report easier to manage, and it allows several team members to make changes to the document simultaneously.

Maintaining the quality of the report

Besides producing an app both the team and the customer could be satisfied with, the team aimed to produce a report of high quality that documents the development process. Table 2.3 contains an overview of the tools used to maintain and improve the quality of the report.

Tools	What we used them for
Todonotes [40]	Comment on paragraphs should be rephrased, or whether an illustration was missing. Also provided a list of all the remaining things to do.
Aspell [41]	Spell checking.
BibTex [42]	To keep track of the citations in the bibliography.
Glossaries [43]	Provide a brief overview of singular words, with explanations.

Table 2.3: The tools used to maintain report

2.3.4 Communication

Internal communication

Google groups [44] was used for casual communication. It includes a shared e-mail-list, ensuring that none of the team members got left out unintentionally. The team mainly used this e-mail-list to communicate internally and to share information from external sources.

External communication

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

3 Project Management

The objective of this chapter is to provide information about the project management and give an overview of the project's process.

The first part of the chapter consists of a description of the development methods used. The second part consists of an overview of the project's time allocation. This includes a work breakdown structure, which gives an overview of the tasks the project consists of. It also includes a Gantt diagram, which briefly explains the timeline of the project's progress. The last part consists of a risk analysis.

3.1 Development methods

As suggested by Sommerville [45], agile development methods are usually a good fit for small developer teams when the goal is to produce a small or medium-sized product. These methods are also preferable when the system requirements are likely to be frequently changed.

The intention of such development methods is for the team to quickly deliver working software to the customer. This is beneficial because the customer may propose new features or change the requirements during the development process.

Based on the correlation between the descriptions of agile development methods and the team's needs, it seemed to be the most expedient to follow the guidelines of an agile development method. First and foremost it should be a development framework that was known to all team members, so that a minimum of time would be spent learning a new process. As all team members had previous experience with Scrum, the obvious choice was to follow this approach. Scrum is an agile method with focus on project management, rather than specific technical aspects.

In order to provide a more complete management framework for the project, it is beneficial to use Scrum in a combination with a more technical agile approach, like Extreme Programming(XP).

The following sections will described the concepts behind the Scrum process and the XP practices. The team's project process will be reviewed in detail in chapter 6.

3.1.1 Scrum

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. This process is illustrated in figure 3.1.

The product owner adds user stories to the product backlog. A user story is a short sentence that describe a small piece of functionality the user wants in the system. The user stories are selected from the product backlog and moved to the sprint backlog at the beginning of a sprint. This process is repeated until the limit of available work hours is reached.

The user stories in the sprint backlog are then broken down into tasks and assigned. Each sprint has a duration of one to four weeks and 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.

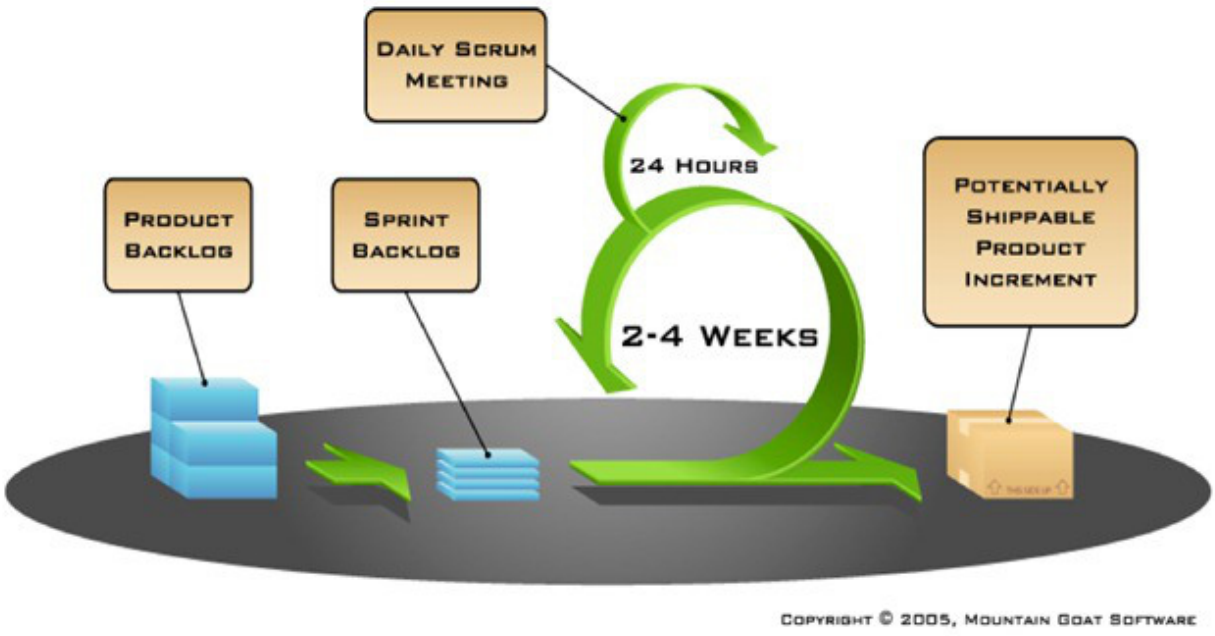


Figure 3.1: Illustration of the process in Scrum

Sprint planning

The objective of 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 tasks, namely the user stories. The estimation of each user story is based on how much time the team think it will take to complete, 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 [46].

Planning poker

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.

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, the user story is assigned that particular estimate. If not, the team members must defend their choices and come to a conclusion the entire team agrees upon.

Daily meetings

A work day is usually initiated with a daily meeting. The purpose 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 would quickly become aware if something was interrupting the development process. This can be a poorly estimated user story that requires more thought and replanning, or some other risk that may occur. For a full list of risks and their remedial actions, see section 3.3.

End meeting

The end meeting is held at the end of each sprint. This meeting consists of a sprint review and a retrospective discussion of the previous sprint. The purpose of this is to ensure that the project is progressing as planned. It is typical to have meetings with the customer at the end of each sprint to review the product. This gives the customer the opportunity to provide feedback on the product and affect the direction the development.

3.1.2 Extreme Programming

The XP approach has several principles the team practiced throughout the project. A brief summary of these principles are given in table 3.1. Further details on how these principles were used and modified may be found in section 6.2.2.

Principle	**Description**
Pair programming	Developers work in pairs and continuously checks each others work.
Planning game	Involves the whole team in the planning process. The plan is developed incrementally and, if problems should arise, adjusted so the software functionality is reduced instead of delaying the delivery.
Continuous integration and collective ownership	As soon as a task is completed, it is to be integrated into the whole system.
Design improvement and optimization	Continuously refactor code as soon as code improvements are found and leave the optimization process to last.
Small releases	Develop a minimal useful set of functionality first. New releases of the system are frequent, and functionality is incrementally added to the first release.
Simple design	The only design that is to be carried out is the design that meets the current requirements.
Sustainable pace	Avoid large amounts of overtime as it is likely to result in reduction of the code quality and less productivity.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
On-site customer	The customer should be available full time. The customer is usually a member of the development team and is responsible for bringing in system requirements for the implementation.

Table 3.1: Principles in XP

3.2 Allocation of time and resources

This section presents a complete overview of the available time resources and the distribution of effort that was given to the different subsets of the project. These are further explained in the two next sections.

3.2.1 Gantt diagram

A Gantt diagram gives an overview of the overall timeline of a project. The diagram orders items chronologically to easily show the predicted state of the project. These items can represent milestones, sprints, deadlines and team downtime.

The Gantt diagram for the project is shown in figure 3.2. The sprints are shown as blue rectangles. The red rectangle indicates the school trip to China in April that the whole team attended. The blue diamonds indicate milestones, or deadlines for delivery of some major part of the project.

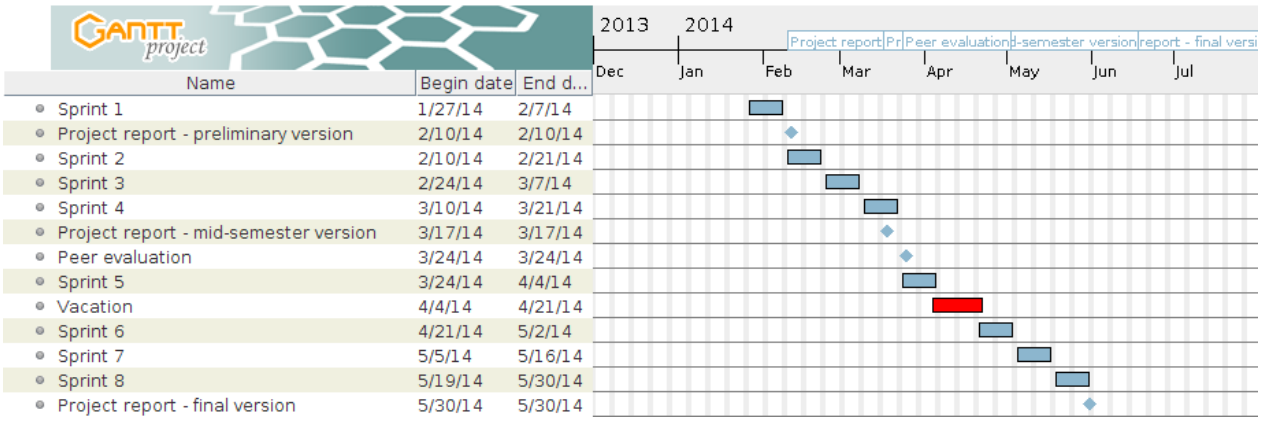


Figure 3.2: The Gantt diagram with sprints and milestones

3.2.2 Work Breakdown Structure

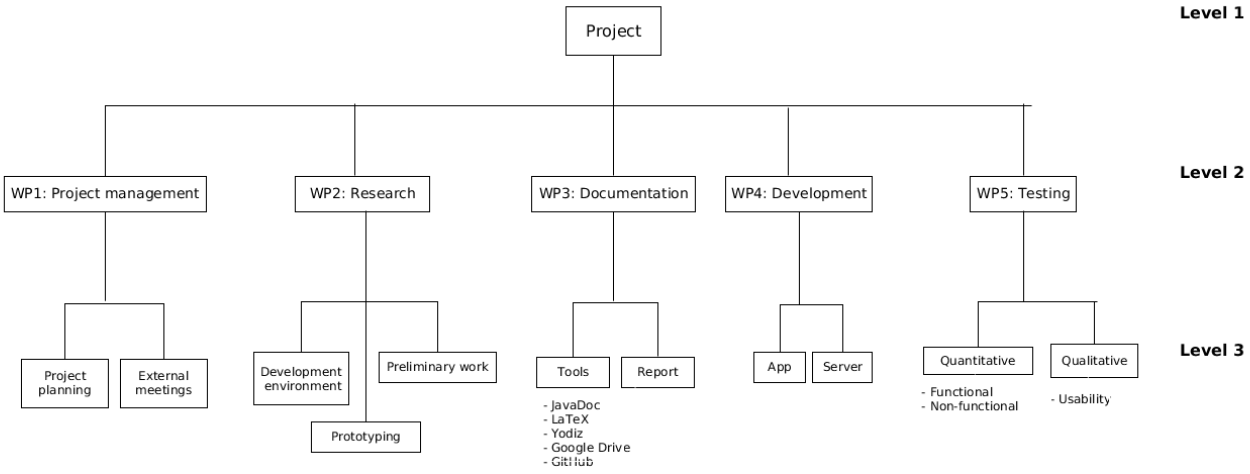


Figure 3.3: Product oriented work breakdown structure for the project app

As shown in figure 3.3, the Work Breakdown Structure (WBS) is divided into five work packages. Each work package represents a subset of the project. Specifically, it specifies *what* will be done, and not how or when the work is to be performed.

The team used WBS in order to get an overview of what kind of work the different parts of the project consisted of, and used this information to estimate how much time to use on each work package.

The estimated time for each work package is given in table 3.2. By comparing the time estimates to the actual time spent, it was possible to evaluate the project's progress to a much greater extent. It was easier to see whether some part of the project was neglected. This comparison is done in section 9.1.

Work Package #	**Name**	**Hours**	**%**
1	Project management	87	5
2	Research	261	15
3	Documentation	696	40
4	Development	522	30
5	Testing	175	10

Table 3.2: Time estimate for work packages

3.3 Risk analysis

As part of the project planning, the team outlined potential risks to the progress of the project. A risk is defined as an unwanted event that may have a negative effect on the process. The team acknowledged the possibility of challenges and issues that might arise during the project, such as technical problems, human errors or personal problems. These risks 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.

In table 3.3 elements considered as risks in the project is evaluated. The probability of an event actually occurring (P) and the effect it would have on the project (E) is measured on a scale from 1 to 9. Importance (I) is the product of probability and effect. The table is sorted from high to low importance.

#	Description	P	E	I	Preventive actions	Remedial actions
1	Underestimation of workload	9	8	72	Continuously revise workload and time left, and prioritize.	Reestimate continuously.
2	Issues with software or tools	9	5	45	Choose software the team members are familiar with.	Hold workshops and view tutorials.
3	Customer does not fulfill obligations.	8	5	40	Keep customer updated. Continuously communicate. Final deadline for feedback.	Contact supervisor.
4	Illness	9	4	36	Multiple team members work on the same task.	Allocate sick member's task to remaining members.
5	Unbalanced workload	5	7	35	Coordinate with team and log hours.	Reallocate tasks.
6	Unproductive hours caused by external disruptions	7	5	35		Find other place to work. Work extra hours to compensate.
7	Team member unavailable	9	3	27	Inform about unavailable dates, progress report and continuous communication.	Keep in touch with unavailable team member or redistribute tasks.
8	Spending more time than estimated on discussions	7	3	21	Include a buffer in our estimations.	Team leader decides whether subject is worth the time or if it should be rescheduled.

Continued on next page

Table 3.3 – continued from previous page

#	Description	P	E	I	Preventive actions	Remedial actions
9	External services unavailable	4	7	21	Project not fully dependent of external services. Server is easy to deploy.	Find new external service to replace unavailable service. Deploy elsewhere.
10	Data loss	2	9	18	Use version control system.	Restore data from previous versions.
11	Customer unavailable	3	6	18	Keep regular contact with customer.	Discuss problem within team, contact supervisor.
12	Communication failure	7	2	14	Make e-mail-list and exchange contact information.	Check e-mail and phones multiple times a day.
13	Customer requirements exceed project time scope	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. Drop low-priority tasks.
14	Supervisor unavailable	3	4	12	Keep regular contact with supervisor.	Discuss problem internally and contact department.
15	Team member drops out of the course	1	7	7	Pair programming and keep everyone motivated	Contact supervisor and redistribute tasks.

Table 3.3: Risk analysis table

4 Requirement specification

The objective of this chapter is to give an overview of the requirements for the app that has been developed in this project. Here, both the functional and non-functional requirements will be discussed, giving a complete overview of the functionality that was included in the app.

4.1 Elicitation of requirements

The customer did not provide a distinct set of specifications when the project first started. Instead, the team was provided with a general idea and a list of concepts that the customer requested. These concepts are shown as a set of prioritized steps in figure 4.1, starting with "Awareness of own consumption". It was up to the team to produce a list of functional requirements that was later be approved by the customer.

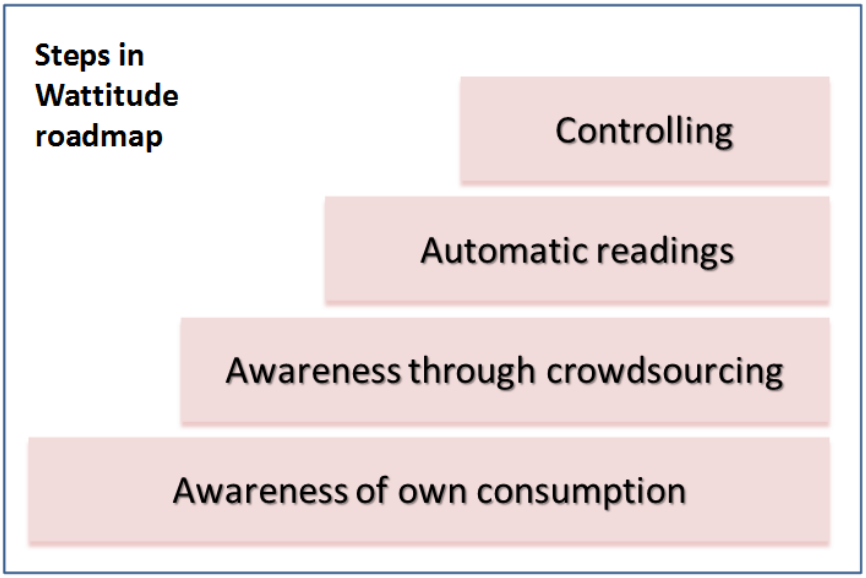


Figure 4.1: Roadmap for Wattitude functionality

The core requirements were:

- The app should be user friendly and simple, so that anyone could measure their energy consumption. This was to be done preferably per device.
- The users should be able to share their energy consumption and production with their friends on Facebook.
- The concept of gamification should be included into the app by allowing the users to compete with each other on saving and/or producing the most energy.
- The finished product should include a user manual and complete technical documentation.

A complete list of all requirements can be found in table 4.1. To measure the user's energy consumption per device, the team needed a hardware device. Optimally, such a device transmits data either via Wifi, Bluetooth, or another form of wireless communication protocol.

Due to time restrictions, the requirement of having hardware monitoring and control over each individual device was deemed optional, and only to be attempted if the team had enough time. After the initial project planning and preliminary work, it became clear that this was too time consuming. The team and customer agreed to remove these requirements.

However, the team did agree to do research on possible hardware solutions provided by the customer. This was done to ease integration in future development. See chapter 8 for more information about further development. In addition to the requirements described in section 4.3 and 4.4, the customer wanted a user manual for the app and technical documentation. These can be found in appendix D and E.

4.2 Use cases

To get an overview of the system functionality from a user's perspective, the team applied use cases. This is a technique to help developers identify functionality that should be implemented, and possible errors that might occur in the system.

The primary actor in this system is the Android app user. An example use case diagram of the final version of the system is shown in figure 4.2. For an overview of all textual use cases, see Appendix F.

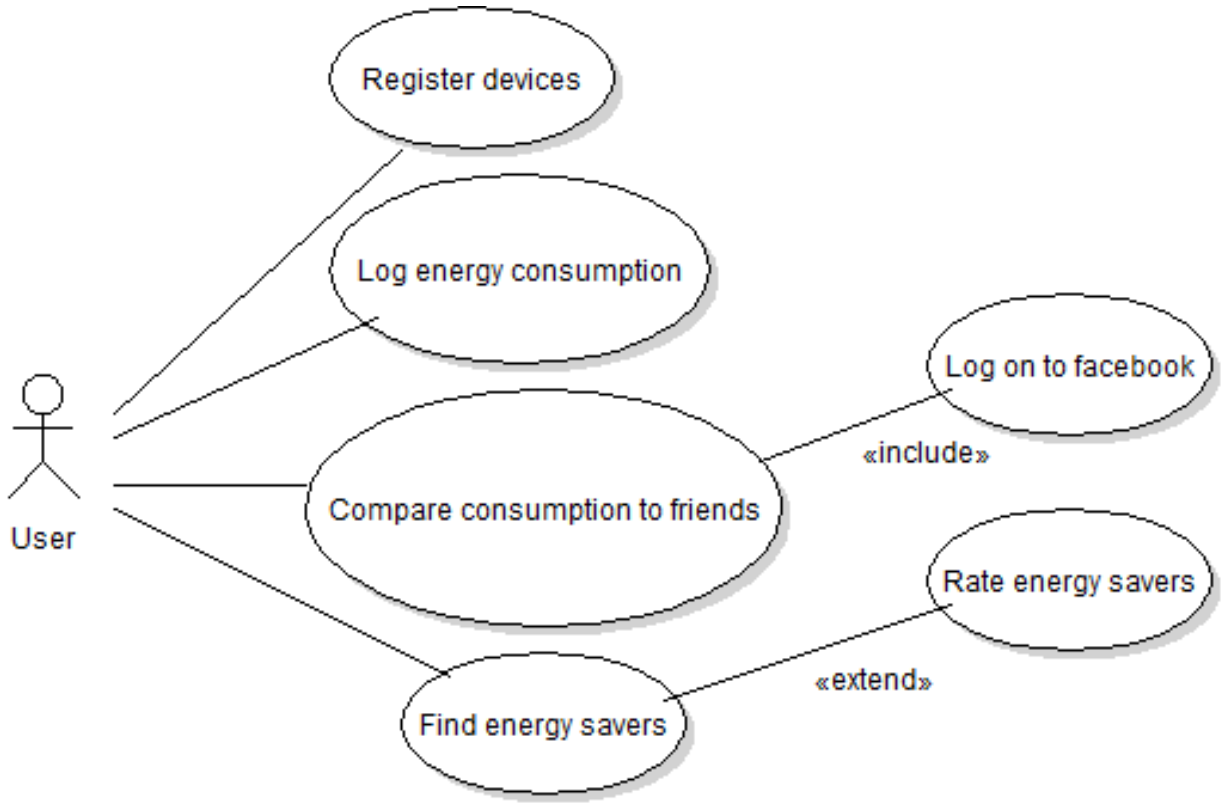


Figure 4.2: Example use case diagram for the system

4.3 Functional requirements

The functional requirements of a system are the specific features that together describe the overall functionality. Table 4.1 represents a complete list of the functional requirements the approved by the customer. An overview of how these requirements were tested is presented in section 7.2.

ID	Description
FR1	Device control
FR1.1.	The user should be able to see how much electricity each device in his house uses on an average basis, on a monthly basis, and on a yearly basis.
FR1.2.	The user should be able to see how much electricity each device in his house produces on an average basis, on a monthly basis, and on a yearly basis.
FR1.3.	The user should be able to add new devices, and to specify whether a device produces or consumes power for each device.
FR1.4.	The user should be able to add the amount of power consumed.
FR1.5.	The user should be able to add the amount of power produced.
FR2.	Power usage
FR2.1.	The app should show the user graphs that show his usage over a predefined set time frames chosen by the user.
FR2.2.	The app should display a graph detailing the selected devices energy consumption.
FR2.3.	The app should display a graph comparing the energy consumption from selected devices.
FR3.	Energy saving tips
FR3.1.	The user should have a list over the most used and best rated energy saving tips.
FR3.2.	The user should be able to rate energy saving tips in a rating bar.
FR3.3.	The user should have a list over what energy saving tips he has done.
FR3.4.	The user will also be able to add energy saving tips he want to do in the future.
FR3.5.	The user should be able to add new energy saving tips to the global list of energy saving tips.
FR4.	Profile
Continued on next page	

Table 4.1 – continued from previous page

ID	Description
FR4.1.	The user should have his own personal profile
FR4.2.	The user should be able to connect his Facebook profile to the app. The app should then retrieve information from Facebook to put in his profile.
FR4.3.	The user should be able to specify whether or not his data should be used for statistics and comparisons.
FR4.4.	The user should be able to choose which criteria he wants to use when comparing his energy usage with other people.
FR5.	Social
FR5.1.	The user should be able to share graphs on Facebook.
FR5.2.	The user should be able to share energy saving tips on Facebook.
FR5.3.	The app should contain a list with Facebook friends that uses the app.
FR5.4.	The user should be able to compare his power usage with friends or users with similar profiles.
FR6.	General
FR6.1.	The app should be written so it is easy to add new languages.
FR6.2.	The app should use the language used on the Android device. The default language, if the Android language is not set, is English.

Table 4.1: Functional requirements

4.4 Non-functional requirements

The non-functional requirements of a system are requirements that can not be directly translated into explicit functionality, but rather a set of criteria that may be used to evaluate the overall system. The testing of these requirements is described in chapter 7.1.

NFR1: The app should be easy to use

The user interface should be easy to use. It should not be necessary with extensive technical understanding or any domain knowledge. The interface should be as plain and intuitive as possible.

NFR2: The app should follow standard Android specifications

To ensure familiarity to users with Android experience, the app should follow Android specifications. This is especially important in the graphical user interface to make sure it is easy for users to start using new apps.

5 System architecture

This chapter provides an overview of the system architecture. The first section presents a superficial explanation of the entire systems architecture. In section two, architectural design decisions the team made is discussed. In the last two sections, a detailed description of the android and the server-side architecture is presented.

5.1 Architecture overview

The system has a client-server architecture with an Android client, and a central server. An overview of the architectural components is given in figure 5.1. This outline shows the primary components of the final solution. The Android app communicates with the central server through the Internet. The server handles functionality for managing and storing data in a database.

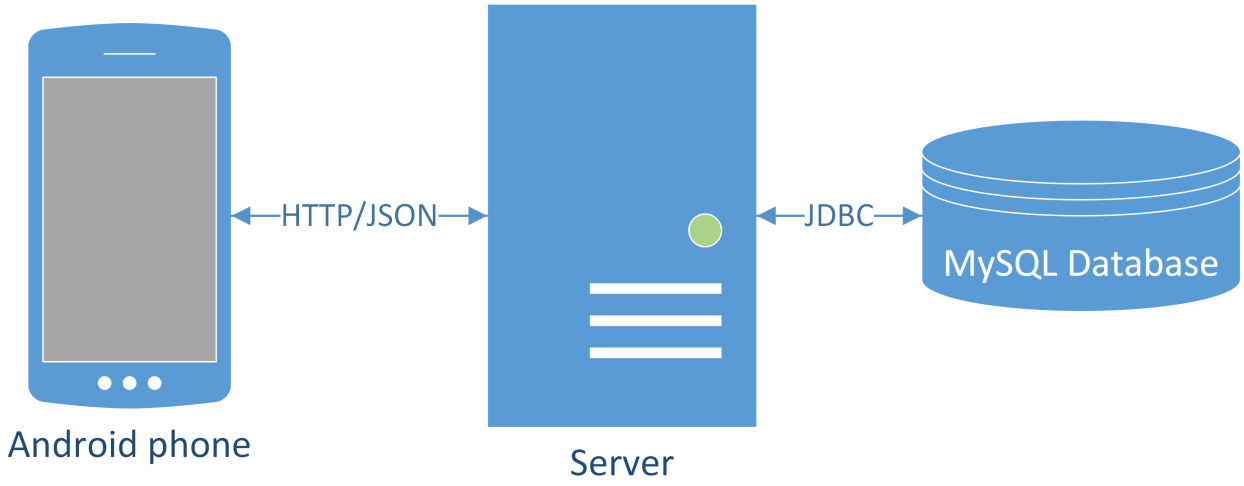


Figure 5.1: Architecture overview

5.2 Architectural design decisions

Selecting the architecture is an important part of any software development project. To create the best possible solution, a preliminary study was performed and a plan on how the different components would interact, was created.

The team members had little to no experience with developing a server according to the chosen design. Development and design was therefore done interchangeably and iteratively in the early stages of the project. Changes in the architectural design were made because many of the requirements in the first draft were discarded due to time constraints.

The customer was initially skeptical to use a dedicated server to store data as this adds a responsibility of up-time and maintenance. The customer suggested alternative solutions, like Dropbox and Google Drive. These were incompatible with several of the functional requirements, including tips (FR3), and comparison functionality (FR5.4). In addition, this would limit further development.

To cover all the functional requirements, storing the data on a central server was necessary. This resulted in a client-server architecture.

5.3 Android app

The app is developed to run on Android versions 4.0 or greater. As of March 2014 this makes up for 79,7% of all Android devices in use [47]. Support for earlier versions would require extra work, and because the app is a proof-of-concept this was not considered a problem.

The best practice guidelines for Android [48] state that one should avoid using a complex architecture. A complex architecture will increase the code base and is unnecessary in most situations. A structured pattern with concise code guidelines is needed to keep the development process as simple, effective, and bug free as possible. This section will give a detailed description of the Android app architecture.

5.3.1 Overall structure

The Activity is the starting point of the app. It is a self contained process with the possibility to display a user interface to the user. To access the different parts of the app it was decided to use a navigation drawer. The sections accessed through the navigation drawer is called a tab. The navigation drawer and all of the tabs are implemented as Fragments. A Fragment is a representation of some behavior or an interface. It is embedded inside an Activity and can be swapped in and out of the Activity. Multiple Fragments can live in the same view. A subset of the Activity life-cycle is implemented in Fragments so it can work as a self-contained module. To read more about the Android life-cycle, see the official Android documentation. [49]

5.3.2 Performance

One of the key issues in mobile app development is keeping the user interface responsive. Android renders the user interface on the main thread. Running long operations on this thread will make the interface unresponsive. Requesting data from external sources, like the server, or Facebook, is time consuming and should not run on the main thread. Handling threading correctly is critical for the overall performance of the app.

5.3.3 Data access

Data access to the underlying database is done using ContentProviders [50]. Data is accessed with URI's that are uniquely defined. An overview of the data layer implementation is shown in figure 5.2. The view box in the figure represent the different Fragments that use data in the app. Fetching data to the view is done through the LoaderManager [51] interface. The LoaderManager loads data from the ContentProvider, and return the result asynchronously on a callback. ContentProviders used with a LoaderManager results in a view that is automatically updated when data changes, without bothering the user interface thread.

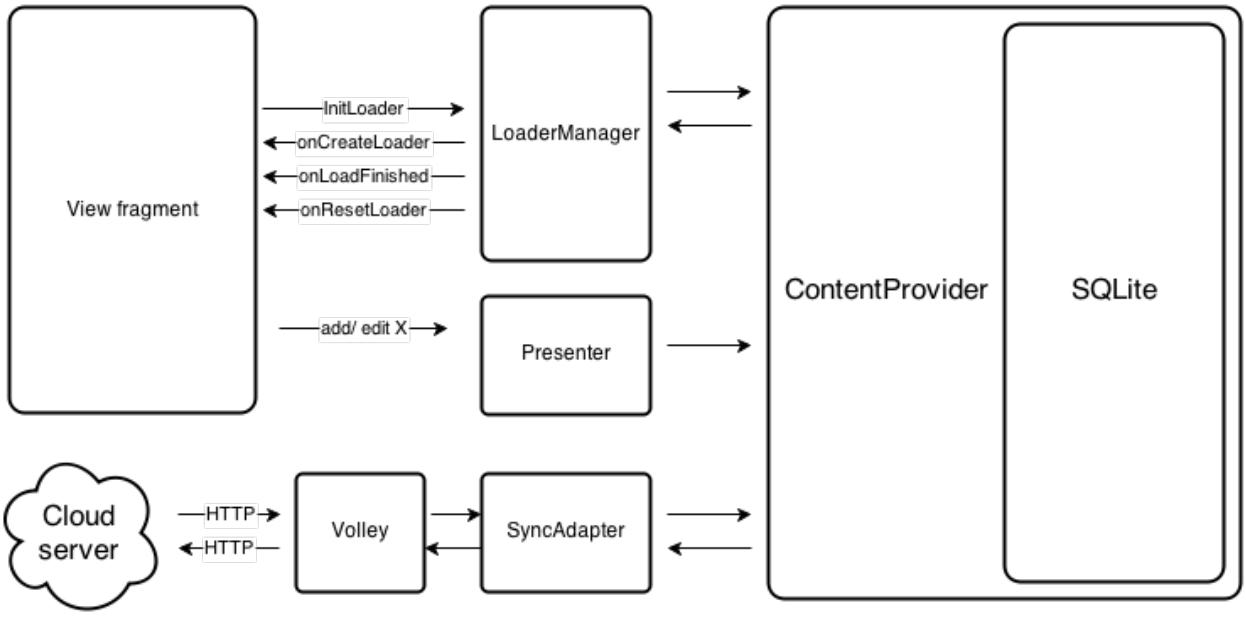


Figure 5.2: Overview of how data is accessed

5.3.4 Data manipulation

Keeping the business logic in one place makes maintaining and updating the code easier. The team decided to use the pattern Model-View-Presenter (MVP). MVP is a Model-View-Controller (MVC) derivative [52]. The MVP pattern separates the models, containing the data, from the presenter, handling the data, and the view, displaying the data. This is illustrated in figure 5.3.

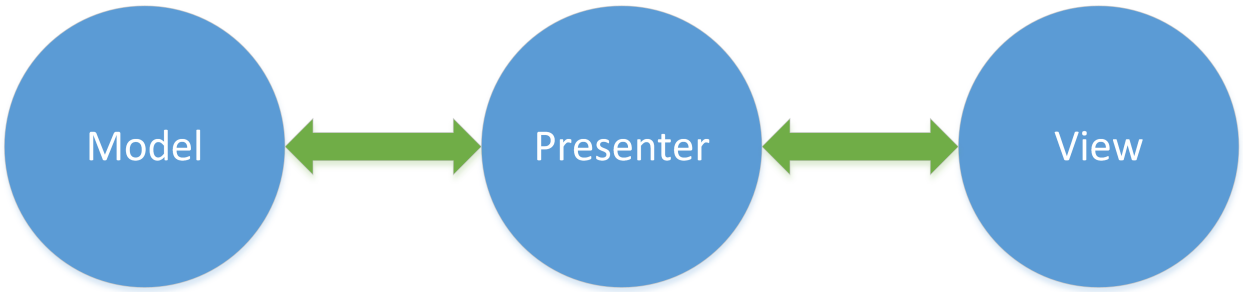


Figure 5.3: An illustration of the Model-View-Presenter pattern

5.3.5 User authentication

The system requires the user to be identified as a unique and authorized user. This authentication is done using the OAuthV2.0 protocol [53]. This needs a third party authenticator, in our case Facebook is used. Other OAuthV2.0 authentication providers can also be used, for instance Twitter or Google. The Android system has its own account architecture. Working with this architecture requires the app to register an AbstractAccountAuthenticator [54] class in the AndroidManifest [55]. When the user logs in, Facebook returns a token, stored in the AbstractAccountAuthenticator object. The token and other related user data (Facebook id) can be accessed, by the application when necessary.

5.3.6 Server communication

The app is designed with the ability to synchronize data to the server. The server, as explained in section 5.4, exposes a restful API were the app can retrieve and store data. Data synchronization is done by using Androids built in SyncAdapter. The SyncAdapter is an external, self-contained process that keeps the data updated. The SyncAdapter is registered to a specific account on the Android system and utilizes the ContentProvider to manipulate data. SyncAdapters can be configured to run when the system has available resources. It can also be forced to run on regular intervals.

5.4 Server architecture

The server is the system back-end and is responsible for permanently storing data and providing it to the user on demand. The team decided that the server should be RESTful in order to make it easy to access. The data-interchange format chosen to transfer data between the app and the server is JSON. This format was chosen for its human readability and it is easily parsed by computer software. In addition, both these properties allow for easy integration with future functionality.

5.4.1 Dropwizard

The server is implemented in Dropwizard which is a Java framework for making RESTful web services. The main class of the server is a service that provides access to the database at run-time. The main structure of how the server uses resources and the database can be found in figure 5.4.

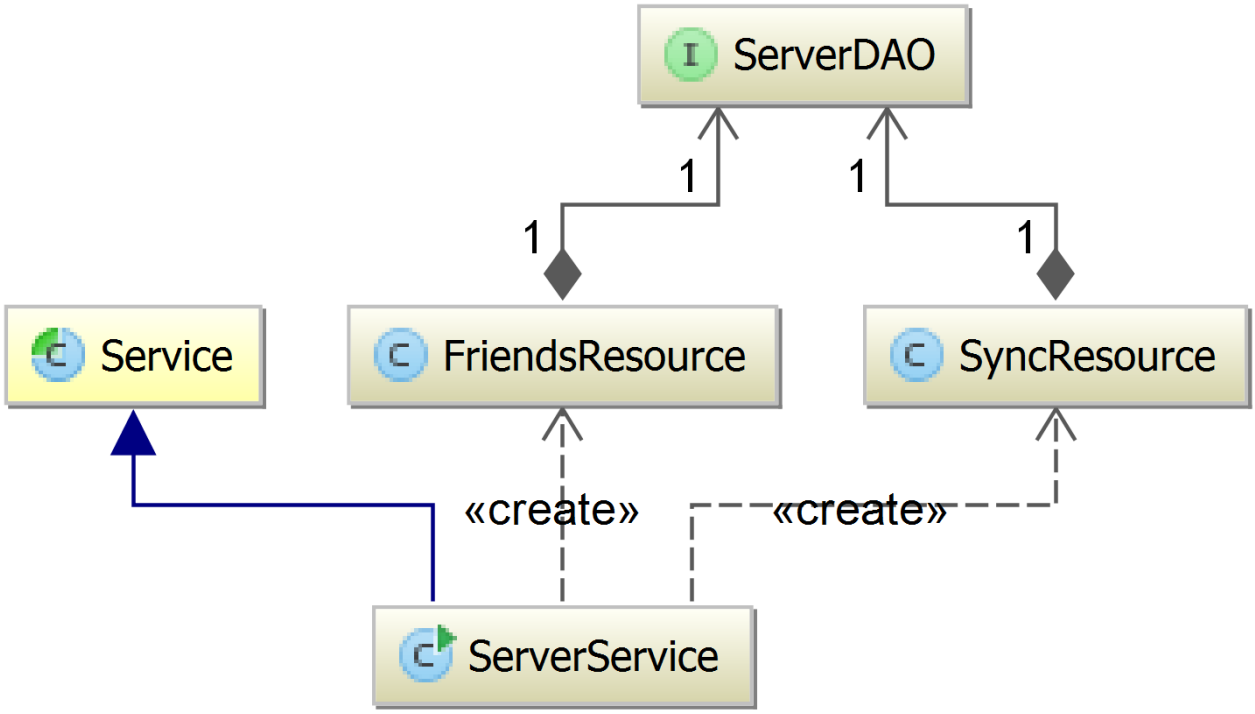


Figure 5.4: UML Class diagram for main server structure

Resources

A Dropwizard implementation has a set of resource classes. These classes contain methods that map to a URI path. When the server receives a HTTP request it will try to match the path to the paths defined by the resource methods. Then it will call the method found or return a

”not found” error. An example resource class is included in code snippet 5.1. The method in this resource will handle HTTP GET calls to ”/user/{user}/sync/usage/{timestamp}” where {user} and {timestamp} matches any text. These variables is then used inside the method to perform actions depending on user and timestamp.

```
1  @Path("user/{user}/sync")
2  @Produces(MediaType.APPLICATION_JSON)
3  public class SyncResource {
4      @GET
5      @Path("/usage/{timestamp}")
6      public List<DeviceUsage> getUpdatedUsage(
7       @PathParam("timestamp") LongParam timestamp,
8       @PathParam("user") LongParam userId) {
9          List<DeviceUsage> usage = db.getUpdatedUsage(
10             userId.get(), timestamp.get());
11         if(usage != null && !usage.isEmpty())
12             return usage;
13         else
14             throw new WebApplicationException(
15                 Response.Status.NO_CONTENT);
16     }
17 }
```

Code 5.1: Dropwizard resource example

Database abstraction

Dropwizard uses the JDBC and JDBI libraries to communicate with the database. The JDBC library is Java’s standard interface to databases. MySQL provides a driver for JDBC in order to make communication possible. JDBI is a convenience layer on top of JDBC which makes it possible to write an interface with SQL queries and use that interface as a data access object.

5.4.2 Database

For permanent storage the server uses MySQL. The database design is illustrated in figure 5.5. This diagram represents the data entities stored in the database and the relation between them.

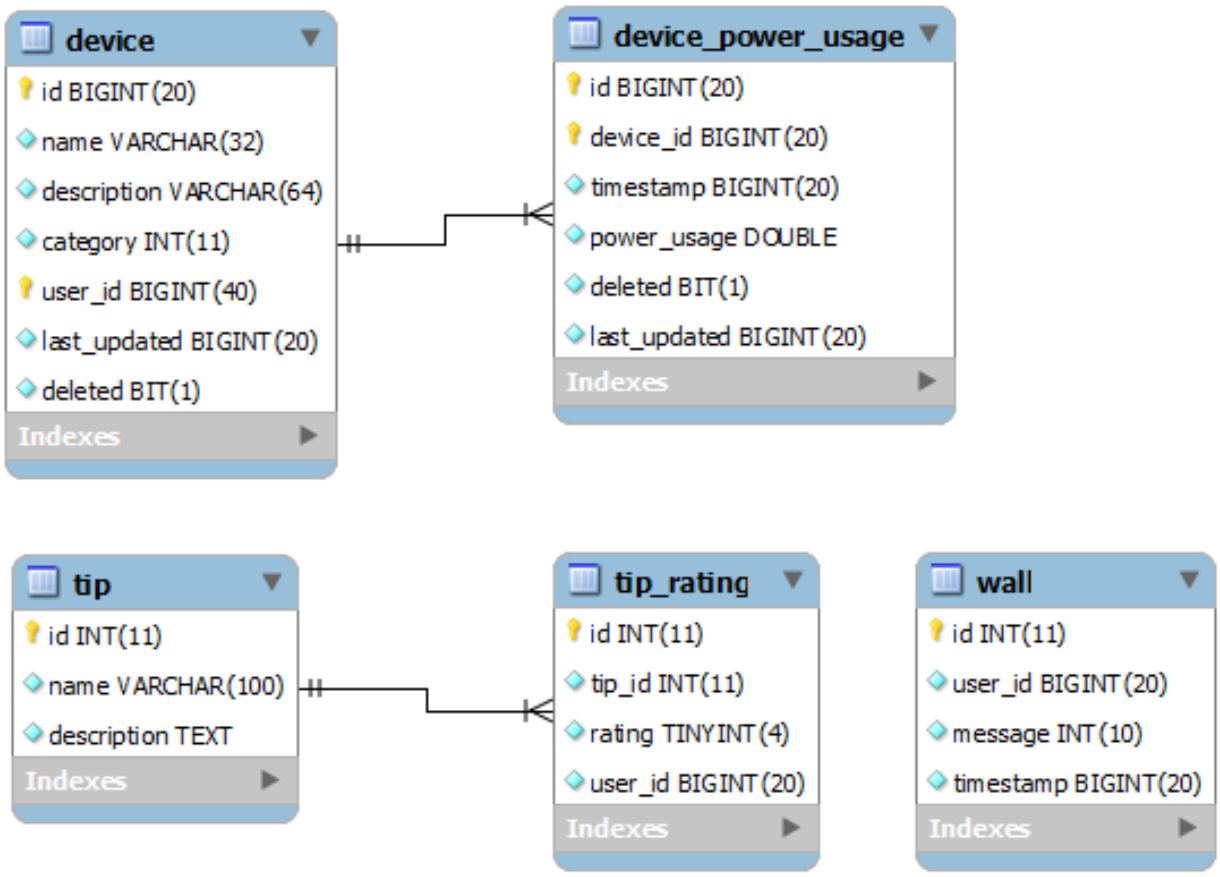


Figure 5.5: ER-Diagram for the database

6 Development process

This chapter presents a brief summary of the design process and the adjustments that were made to Scrum and XP. It also provides an overview of what the team accomplished at different times during the development process.

6.1 Design

To get familiar with the assignment's context, the team did a lot of research in the beginning of the project, as described in chapter 2. A part of that process was to find existing solutions and get some ideas about functionality that might be useful in the app.

This section describes the branding process, and summarizes the team's process of designing the graphical user interface for the app.

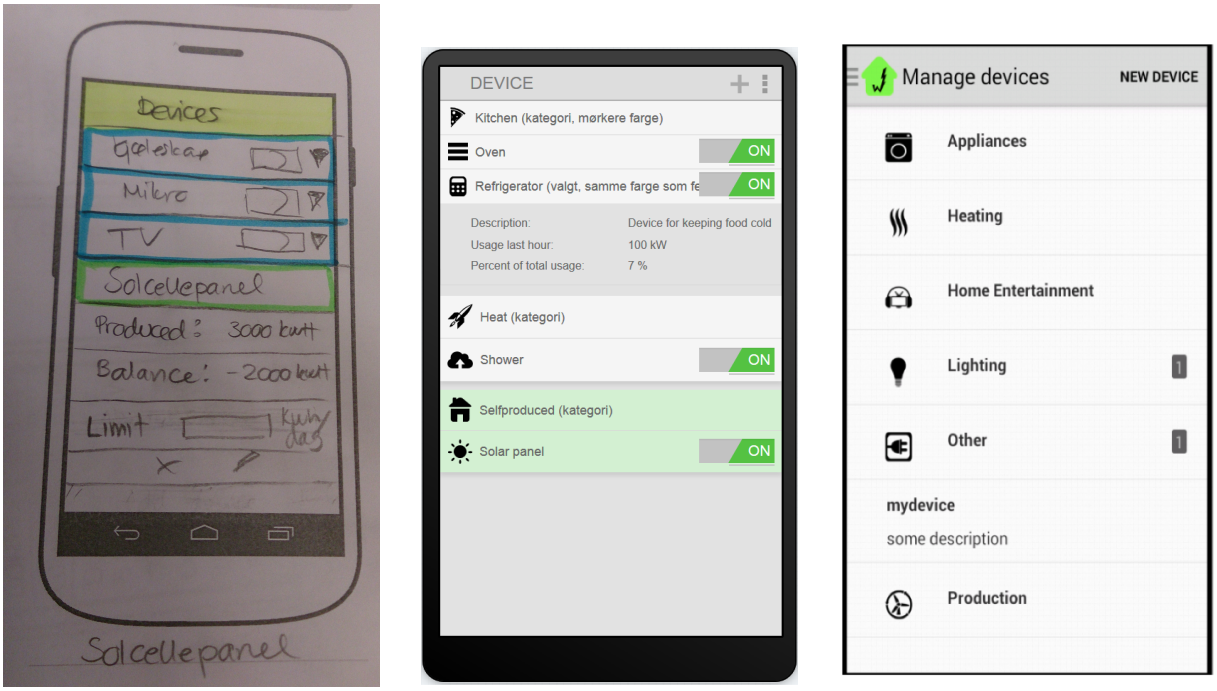
6.1.1 Branding

An important part of the design process was to make sure the app was both user friendly and well customized for its purpose. As the team was unable to find suitable icons, it was decided to create custom icons for the different tabs and devices so that each tabs functionality would become more apparent.

The team decided to change the name of the app, originally named ”UbiSolar”. Considering that the purpose of the app was to increase the awareness of power consumption, the team thought the app should have a name that reflected this purpose. This led to the name ”Wattitude”, a composition of watt and attitude (towards power consumption). This idea was well received by the customer.

6.1.2 Prototyping

In the early stages of the project, the customer requested concepts for the application. To provide this, the team had a brainstorming session, which resulted in paper-prototypes of the app. An example of this early prototype is shown in figure 6.1(a). This prototype laid the foundation for further development and was helpful when specifying both the functional and non-functional requirements.



(a) Paper-prototype of a list of devices where a power producing device is selected

(b) Digitalized prototype of list of devices on proto.io

(c) Screenshot of list of devices in app

Figure 6.1: The evolution: From prototype to product

However, a paper-prototype may have some drawbacks, as suggested by Snyder [56]. For instance, it is hard to incorporate changes without having to re-do work. It is static, and does not give the right impression on what the end product might look like. Therefore, the team moved on to a digital solution, namely the web page proto.io [57]. This tool made it possible to create dynamic prototypes. An example screenshot of this prototype is shown in figure 6.1(b).

After the prototyping was done, the app was implemented in Android. An example screenshot of the finalized app is displayed in figure 6.1(c).

6.2 Project methodology

As explained in section 3.1, Scrum was chosen as project management framework. Adapting Scrum and XP into the project had several challenges that will be described in this section.

6.2.1 The team's adaptation of Scrum

The team made several modifications and adaptations to the Scrum framework in order to make it match the project's needs. These modifications are described in the sections below.

Keeping the project leader

The concept behind the Scrum approach is to empower the entire team to make decisions, so that a project leader 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 the project. The tasks related to this role is described in table 1.1.

Sprints

As having daily meetings was incompatible with the team's schedule, the team agreed to meet twice a week, each meeting starting with the daily meeting routine. In addition, the team met weekly with the customer, and with the supervisor once every second week. In collaboration with the customer, tasks were moved from the product backlog to the sprint backlog. This gave the customer a possibility to affect the team's main focus.

Scrum Tool

The team's main requirements in a Scrum tool was that it should be easy to use, effective for project management, and had GitHub integration. It should support chart generation for documentation and overview purposes as well. With this tool the team would ideally get an overview of the project's progress and the team's performance. After a brief decision process, the team chose Yodiz because it provided free accounts for educational purposes and satisfied the team's requirements. Further details on this process can be found in section 9.2.6.

Planning poker

In the team's planning poker sessions, custom units for the estimations were used. These units were small, medium, large, and extra large, each of them representing one, two, four, and eight hours respectively.

6.2.2 The team's adaptation of Extreme Programming

Several modifications and adaptations were made to the XP method in order to optimize it to fit the project's needs. These modifications are described in the subsequent sections.

Pair programming

The team practiced the principle of pair programming, by dividing the team into small groups or pairs when designing prototypes, deciding on architecture, and in a few cases, collaborated on code sections.

Planning game

The team interpreted the planning game in XP to be a more generic description of planning methods. The team felt that planning poker, which was used for the estimation of the tasks, was a more specific implementation of the planning game in XP. This assumption was made based on one of the most important aspects of planning poker: To avoid the influence of the other participants, no team member should know what the rest of the team estimates. This aspect is not mentioned in the planning game specification in XP.

Continuous integration and collective ownership

The team practiced the principle of continuous integration by developing on separate branches in Git, and then merge the branch back to the master branch when a task was considered to be completed.

The team practiced the collective ownership principle by sharing and collaborating the project's code. It was also agreed with the customer to follow standard code conventions.

Design improvement and optimization

The team continuously followed this practice by refactoring, cleaning up code, and leaving the optimization of the code to the last two sprints. It was also practiced during design improvements in the development of the prototypes.

Small releases

By the end of each sprint the team had a goal of releasing a new version of the application. This new version should consist of new functionality and improvements to previously released functionality.

Simple design

Although the team worked iteratively and in close collaboration with the customer, it was not always easy to *only* implement the basic functionality the customer requested. As it was

up to the team to define the requirements specification, it was hard leaving out requirements the team thought would improve the app, even though they were not directly requested by the customer.

Test-first development

The team did not follow this practice. Although it was a good idea to use a test-driven development approach towards the developing process, the team lacked both the experience and the time to use this practice. The team therefore decided to drop it.

On-site customer

This principle does not apply to the team's project, as our customer comes from an external organization. However, the team had co-located meetings with the customer at a regular basis, usually once a week and hence got continuous feedback and acceptance testing. Further details on the customer's acceptance testing can be found in section 7.3.

6.3 Sprint overview

The following sections summarize the project progress in chronological order. They include the goals the team set for each sprint, and whether these goals were met. The burn down charts and sprint backlog for all sprints are located in appendix G.

6.3.1 Kickoff

An initial meeting was conducted not long after the team was assigned. Regular meeting times, goals and expectations for the project was discussed. All team members agreed to aim for the best grade possible.

6.3.2 Sprint 1

The goals set for this sprint were:
- have a meeting with the customer
- get a better grip of what the tasks ahead were
- research about the assignment
- assign all the team members roles

After choosing the Scrum approach as the project's framework, a project outline was created, and concepts for the app presented to the customer. The customer was very pleased with the team's ideas, and approved the temporary specification. The team also set up the development environment by starting to write documentation in LaTeX, setting up Android studio and chose Yodiz as project management tool for work distribution and tracking.

6.3.3 Sprint 2

The goals set for this sprint were:
- provide the customer with a working software prototype
- set up the server

By making this prototype, the team could use it as basis 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 began to form the foundation to a framework of functions that the Android app could use.

The customer was satisfied with the progress on the prototype, and provided rich and detailed feedback on how the app should be improved for the next iteration.

6.3.4 Sprint 3

The goals set for this sprint were:
- design prototypes to plan basic functionality
- implement Android GUI for basic functionality

The team experienced that it was hard to write sections of code without having a GUI design to follow. It was therefore decided to design prototypes before starting the implementation. This resulted in detailed, digitized prototypes for each part of the app. The team also made some progress on the actual implementation.

During this sprint, some of the team members got sick. This was also a period where other courses at NTNU had big exercise deliveries. The key issue during this sprint was that the effort needed for the other subjects was underestimated.

6.3.5 Sprint 4

The goals set for this sprint were:
- implementing a prototype with functionality from the requirements specification
- complete preliminary report
- peer evaluation

This sprint had two factors that made it a productive, but also exhausting sprint: Because the previous sprint had not met expectations on completed work, the team compensated for this by working overtime and exceeding the expected working hours by 30 percent. Naturally, the progress was uplifting, but everyone got exhausted from all the extra work. It became apparent that the time required for the tasks made so far had been underestimated. The team agreed it would be better to overestimate and add tasks to the sprint rather than underestimate and not finish the goals set in conjunction with the customer.

6.3.6 Sprint 5

The goals set for this sprint were:
- use prototype as basis to create a software prototype
- implement data synchronization
- replace dummy data with server data

This sprint the customer expressed he wanted to become even more involved with the development process by taking part in the sprint start meetings. He also gave us constructive feedback that lead to many major redesigns, particularly in the Android navigation menu. We were made aware that navigation in what was the main menu at the time could be a challenge. The team used a Kanban board to plan what to focus on in the remaining sprints of the project. This planning process is illustrated in figure 6.2.

Figure 6.2: Planning process with customer

This sprint also had some goals that were not met. The data synchronization was not implemented, and thus the dummy data was not replaced by data from the server based on user credentials. However, most of the major app functionality was in place, and the team had a pretty clear image of how the finished app would look like and behave.

6.3.7 Sprint 6

The goal set for this sprint was:

- completing all functionality to allow for usability testing, rigorous testing and bug fixing

Completing the persistence layer was crucial to testing all the app functionality. All tabs were to be completed at the end of this sprint. The only exception would be particularly tricky problems that required more resources.

After reviewing the progress made in sprint six, the team decided it was time to dedicate some extra resources to documentation. Sprint five and six had been devoted almost exclusively to development in order to meet the deadline set in cooperation with the customer. Even though generous estimates were made to ensure delivery, the team divided itself into two groups as an extra precaution.

6.3.8 Sprint 7

The goals set for this sprint were:
- to finish development of the app and the server
- run the test sets in the testing scheme
- optimize code and fix bugs
- acquire the customers approval for the finished product

The testing revealed performance issues and some major stability issues that had to be resolved before customer approval. Half the team was working intensively on bug fixing and stability, while the other half focused on the report. A lot of work, especially in the chapter on further development had to be well under way before the start of sprint eight. After fixing the major issues discovered in testing, the team acquired the customers approval for the finished product on the 16th of May.

6.3.9 Sprint 8

The goals set for this sprint were:
- tweak and fix small bugs in the app
- write technical documentation
- hold a final presentation of the product for the customer
- finalize the report

The team held a presentation of the project at SINTEF on the 23rd of May. The presentation was followed by a Q & A session where the researchers from CoSSMic could ask questions about technical solutions and further development. The last tweaks and fixes were made before the app was presented. This period also had focus on finalizing technical documentation, such as JavaDoc and a wiki on GitHub, as requested by the customer. Lastly, the main focus has been on proofreading, rephrasing and spell checking the report to make it ready for delivery.

7 Testing

This chapter explains what tests were conducted and why the team conducted them. It gives an overview of the overall testing strategy and a detailed description of the testing methods. The team utilized user testing, functional testing and acceptance testing to make sure the system measured up to a set standard.

7.1 Testing non-functional requirements

Non-functional requirements are requirements that represent the overall usability and quality of the app. To test these requirements, different kinds of usability tests were performed.

7.1.1 Usability testing

Usability testing is a technique used to evaluate an app by testing it on actual users, preferably within the application's target audience. This kind of test give the developers a better understanding of how a common user reacts to specific features in the app. If the test subjects struggle with, or are unable to complete certain tasks, some changes in the user interface might be necessary. The target user group for the app is home owners. This is a wide target audience that sets extra requirements on the app usability. To test this usability, two different methods were used, as described in the subsequent sections.

Hallway testing

Hallway testing is a typical method for usability testing [58]. In hallway testing, the subjects are random people ("people who pass by in the hallway," hence the name). During the test, the subjects are given a set of tasks to perform while the developers observe how they perform their given tasks. This allows the developers to make some choices early in the development process without the need for protracted user testing with carefully selected users. The team used this method to some degree, mostly in the early stages of prototyping.

Remote usability testing

By using remote usability testing [59], the team was able to get input and opinions from users closer to the app's target audience. The tests were conducted mainly on friends and relatives. The subjects were given the app and a set of tasks. After the tests, the subject filled out a simple form where they rated the usability of the app. The form was then used to make improvements on the app user-interface.

7.1.2 Performing the tests

The project had two main requirements to cover the overall usability of the app. The following sections describes how these were tested.

NFR1: The app should be easy to use

The target audience of this app does not necessarily have any technical insight. This means that the app should to be easy to use for the average person. This was tested by performing both hallway testing and remote usability testing.

NFR2: The app should follow standard Android specifications

The target platform for the app is Android, which has a set of standards aiming to make Android apps consistent. By running the usability test on users with Android experience, the team got feedback from the test subjects on whether the app looked and behaved like an Android app.

Test plan

When a usability test is performed, the subjects are usually given a set of tasks to perform. During the test, the developers should take special note of whether the subjects are struggling or gets frustrated. An overview of the usability test tasks are presented in table 7.1. For each task, the testers were asked the following questions:

1. What did you find complicating about the tasks given above?
2. Which of the above tasks did you find intuitive and easy to perform?
3. What features do you think should be added to make the tasks above easier?

The test results were used to improve the user interface. An example feedback document can be found in appendix I.

Task id	Task description
1.	**Basics**
1.1	Log in
1.2	View your profile
1.3	Add a new residence
1.4	Log out
2.	**Devices**
2.1	Add a new device
2.2	Add usage to the device
3.	**Usage**
3.1	Check the usage of the device called "Heater"
3.2	Compare the usage of your radio and TV
4.	**Comparison**
4.1	Compare your usage with a friends
4.2	Compare your usage with someone with a similar profile
5.	**Tips**
5.1	Add a tip to "my tips"
5.2	View your list of tips
5.3	Mark a tip as done

Table 7.1: Overview of usability test tasks

7.2 Testing functional requirements

Functional tests are tests that inspect the functionality of the software. This is a type of black-box testing [60], which is a software testing method that examines app functionality without exploring the app's internal structures. This is done by feeding the app data and checking that it responds properly. The given input is either a) correct data, where it is checked that the data is handled correctly, or b) incorrect data, to ensure that the app fails gracefully and does not crash. To test a system the following steps are required:

1. Identify what features or functionality to test
2. Specify input to the system
3. Specify expected output from the system
4. Execute

7.2.1 Test table

Table 7.2 presents a list of functionality that should be tested thoroughly in every iteration of the application if applicable. This table is a subset of table 4.1 with some of the text paraphrased. Because of the lack of automatic unit testing, the functionality had to be tested manually in order to be sure that new functionality and bug fixes did not break any existing functionality.

ID	**Description**	Result
FR1.1.	The user should be able to see how much electricity each device is using.	Ok
FR1.2.	The user should be able to see how much electricity each device is producing.	Ok
FR1.3.	The user should be able to add new devices, and to specify whether a device produces or consumes power for each device.	Ok
FR1.4.	The user should be able to add the amount of power consumed.	Ok
FR1.5.	The user should be able to add the amount of power produced.	Ok
FR2.1.	The app should show the user graphs that show his usage over predefined time frames.	Ok
FR2.2.	The app should display a graph detailing the selected devices energy consumption.	Ok
Continued on next page		

Table 7.2 – continued from previous page

ID	**Description**	Result
FR2.3.	The app should display a graph comparing the energy consumption from selected devices.	Ok
FR3.1.	The user should have a list over the most used and best rated energy saving tips.	Ok
FR3.2.	The user should be able to rate energy saving tips in a rating bar.	Ok
FR3.3.	The user should have a list over what energy saving tips he has done.	Ok
FR3.4.	The user will also be able to add energy saving tips he want to do in the future.	Ok
FR3.5.	The user should be able to add new energy saving tips to the global list of energy saving tips.	Ok
FR4.2.	The user should be able to connect his Facebook profile to the app. Fetch data from Facebook.	Ok
FR4.4.	The user should be able to choose which criteria he wants to use when comparing his energy usage with other people.	Ok
FR5.1.	The user should be able to share graphs on Facebook.	Ok
FR5.2.	The user should be able to share energy saving tips on Facebook.	Ok
FR5.3.	The app should contain a list with Facebook friends that uses the app.	Ok
FR6.1.	The app should be written so it is easy to add new languages.	Ok
FR6.2.	The app should use the language used on the Android device. The default language, if the Android language is not set, is English.	Ok

Table 7.2: Table showing testing of functional requirements

7.3 Acceptance testing

Acceptance testing is a form of testing that checks whether or not the functional requirements are met. Acceptance testing is normally done with a customer representative present. During the development of the app, the acceptance testing with the customer was done regularly, often once or twice for each sprint. This gave the team valuable feedback on the app's status, what parts that could be improved and where to focus the time and resources until the next meeting.

At the end of the development process, the customer met with the team to review the finished product. The customer previewed the app while the team answered questions from the customer. Due to the iterative development process and regular meetings with the customer, there were no major changes in the last version and the product was deemed to be satisfactory. The customer's approval letter can be found in appendix A.

8 Further development

The objective of this chapter is to describe how the team envisions further development of the system. First, software improvements to improve Wattitude will be examined, as well as how these improvements might be implemented. This involves functionality that was left out due to time constraints, and features that need improvements.

Then, all new concepts the team have discussed during the project, but not have implemented in to the app are discussed. This includes how the team envisions functionality for measuring power production and methods for including gamification into the app.

Lastly, some insight into hardware solutions needed to get live data from devices in the users house will be provided.

8.1 Software improvements

Due to time restrictions in this project, not all functionality was fully implemented. In this section all of that functionality and how to implement it is described.

8.1.1 Residences

The way the system is built up right now is that everything in the app is connected to a user. A user has devices and devices have usage. One possible improvement would be to have the user connected to one or several residences and the devices connected to those residences. This change would be relatively simple as there is only one extra layer of models needed. It would also make it possible to compare users more accurately, taking into consideration what type of residence they have, how many people live in that residence and so on. This would give each user the possibility of monitoring the power usage of their summer home, cabin or rented out apartments. It is also possible to connect all tips to residences, so the user can choose which residence he wants to perform specific tips for.

8.1.2 Profile

The profile tab is supposed to show information about the user and let the user manage residences. The team envisions this tab to contain privacy settings, allowing the user to,

for instance, disable data collection for statistics and comparison. In the current version of the app, the user has no option to remain anonymous. On the server side this could be implemented by adding a flag to the user that indicate a wish to remain anonymous. When retrieving data for comparison or statistics the server can then filter out the users based on this flag.

8.1.3 User authentication

The current user authentication is done with the Facebook SDK. This authentication is secure enough in itself. The problem is that user authentication to the server is done by having a user id in the URL. The connection is not secured in any way, and no validation is done on the server side. The token returned by the authentication to Facebook should be included in the header when communicating with the server. This is enough to validate the user and his access rights server-side.

8.1.4 Graph improvements

The current graph creating process is done within an AsyncTask that resides in the class where the line graph is made. The current implementation can create memory leaks, because inner classes should be static. There is a need for an easy way of reusing graphs in different tabs. To solve this, a generic graph factory can be created.

8.2 Unimplemented concepts

During the first few weeks of the project the team came up with several good concepts that can make the users more aware of their power usage, and motivate them to reduce their usage. This section will describe some of the concepts that were ultimately dropped from the project because of time limitations.

8.2.1 Power Production

The app has the possibility of creating a device with the category ”production”. This category is meant to be where the user's energy generating devices can be added. It allows the user to view power production the same way as power usage. To improve functionality for personal power production, a dedicated tab could be created allowing more advanced functionality for displaying power production. It is common for users producing power to store the accumulated power in battery parks. With the hardware support, it would be possible to display, and to some degree, control the battery park in the application.

8.2.2 Power saving comparisons

Most people are interested in saving money. Calculating the money a user saves based on reduction in power usage, and displaying it, can be a major motivational factor. The cost of electricity varies from day to day, but with the new AMS[61] system that is being implemented in Norway, it might be possible to get detailed information about the cost. This would give the user a precise representation of money saved per KWh. It is also possible to use an average price to calculate the money saved, but the results would be less accurate.

Another interesting way to compare power saved would be to compare it to other things or events like for instance ”You have now saved enough energy to send a rocket 1/100 of the way to the moon”. This would give the user a better perspective and it might increase motivation.

8.2.3 Gamification

At the start of the project the customer expressed that he wanted a gamification concept implemented in some way. Gamification is when you turn something into a game or competition to motivate the user. The sections below describes different kinds of gamification concepts that were considered.

Competitive rankings

With enough data collected for a user, it would be possible to create scores based on reduction in the power usage, or improvement in power productions. This score could be used in a ranking system with scoreboards allowing a user to compete with friends and other people.

Unfortunately, this requires a somewhat advanced model to calculate the score representing improvements. The techniques required for creating a ranking system includes an advanced statistical model and some sort of machine learning.

Achievements

Achievements is a common concept in modern applications. The idea behind it is that the user would get some sort of reward for achieving a goal. A goal might be something like ”use 10% less power than last month”, ”Started producing power” or successfully using a tip. One example of a successful achievement system can be found in the app ”Untappd.” [62]

Another achievement system that was discussed was to integrate different ”Wattitude levels” into the app. The idea behind this is that the user gains points for things like reducing power usage, producing power, and sharing tips. The user ”level up” when they get enough points. The user can then compete with friends to gain levels. Based on the level it would be possible to reward the users with things in the app, like emblems or icons, or even real life things like coupon codes.

Wish list

With the energy to money conversion explained in section 8.2.2 implemented, it could be extended with the possibility of creating a personal wish list. The purpose of a wish list is that the user can set up a list of things they want, like new shoes or a vacation. The idea is that the application could keep track of how much money the user has saved, compared to the price of the item. This could be displayed using for instance a progress bar showing how close the user is to the goal.

8.3 Hardware components

In the very beginning of the project, the team looked into the possibility of getting live data directly from devices. This required some technical equipment for collecting and transmitting power usage from a device. This section describes the architecture and the necessary hardware.

8.3.1 Hardware architecture

The envisioned architecture is shown in figure 8.1. This includes both finished functionality and the concepts the team were unable to implement because of technical restrictions or time constraints. The team concluded that the best architecture for a such a system would include a home data aggregator. The home data aggregator receives data from the measuring units connected with the devices. Depending on the functionality of the measuring units, remotely controlling devices, can be possible.

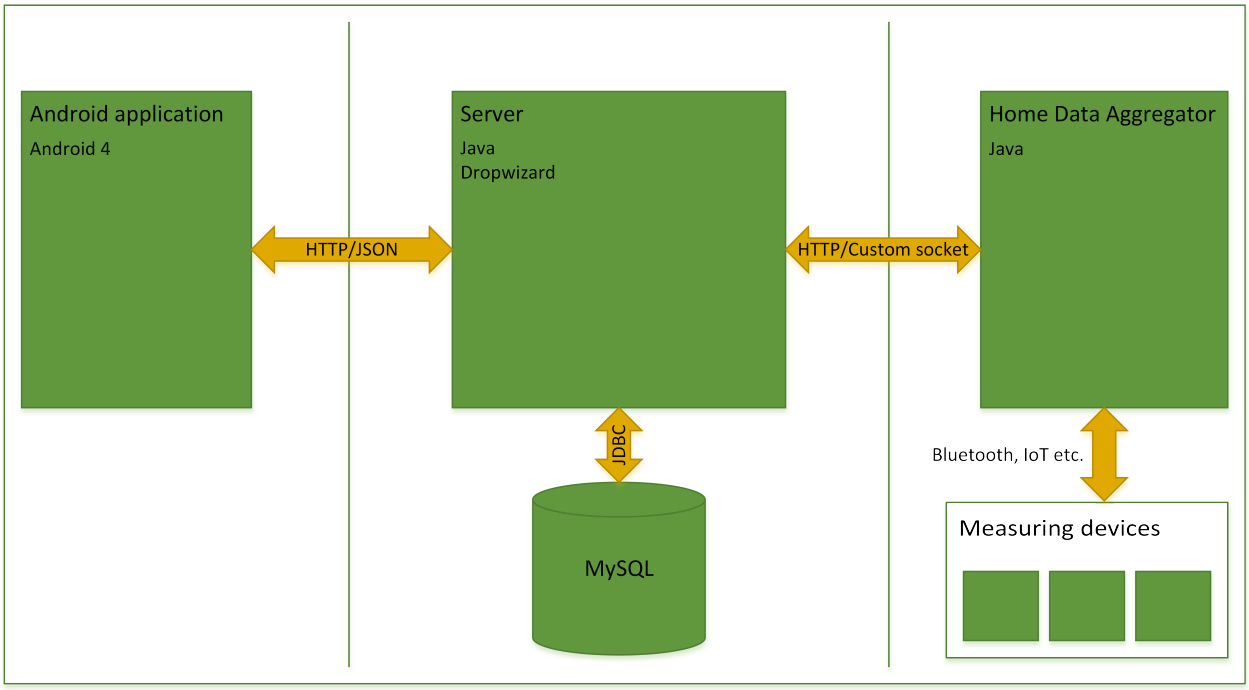


Figure 8.1: Ideal Wattitude architecture

8.3.2 Home data aggregator

To ensure full operability 24 hours a day, a base station is needed in the user's home. This server will serve as an aggregating agent for data from measuring units attached to devices. Given an ideal architecture implementation, this unit would also be responsible for controlling devices. The team envisions this unit as a synchronization service working for the Wattitude

cloud server. The software for such a server could be based on the current server software. The hardware for this base station could be as simple as a Raspberry Pi [63]. These computers are small, cheap and provides all the hardware needed for a simple home aggregator. Some modifications is necessary to communicate with the measuring units. The architecture for this aggregation unit is shown in figure 8.2.

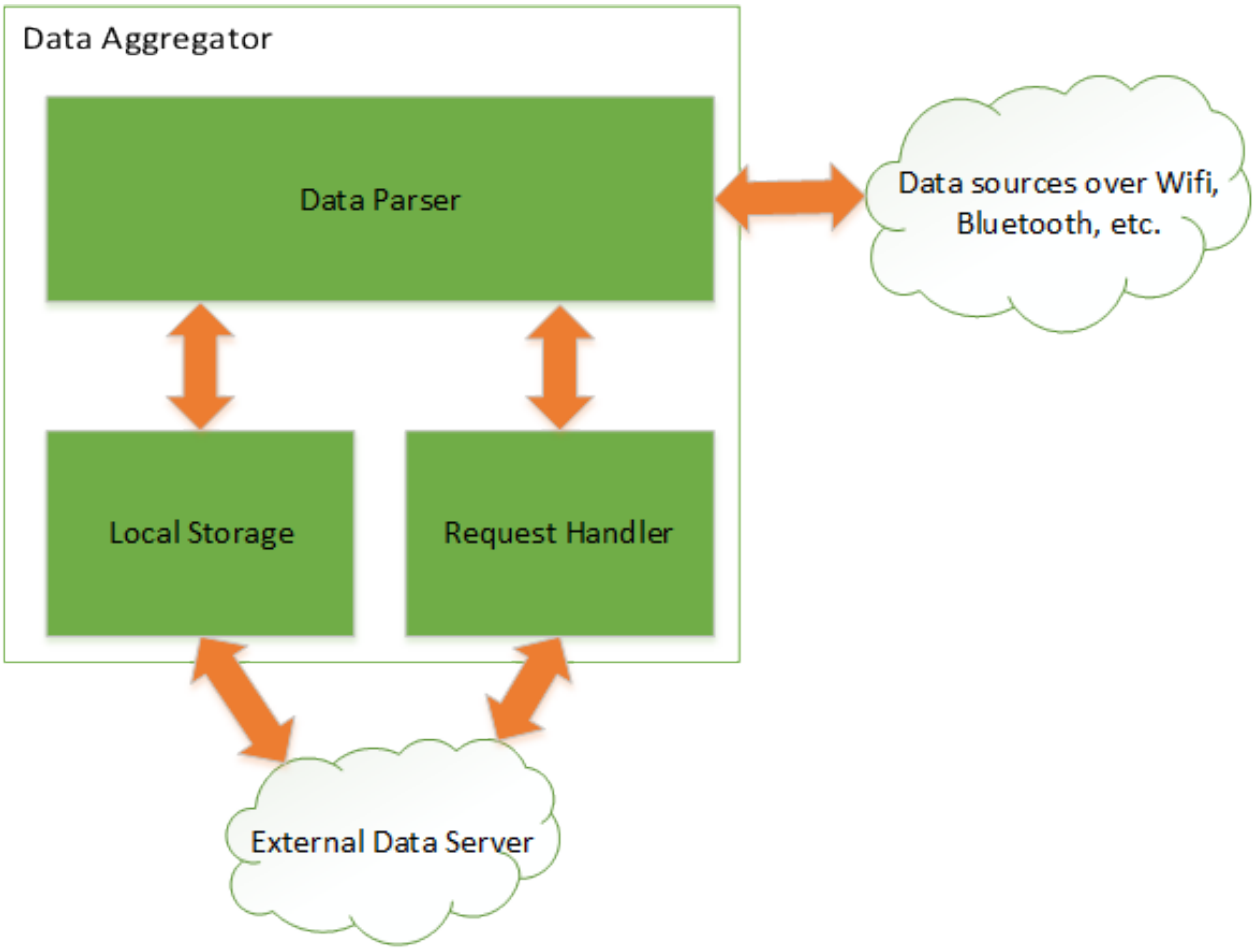


Figure 8.2: Detailed architecture for the home data aggregator

8.3.3 Measuring unit

To make real time monitoring possible, every device the user wants to monitor must be connected to a wireless monitoring unit. The team tried to find an existing solution that was satisfying, but most of the options found was proprietary and/or too expensive. The following paragraphs describes the most viable options.

HomeMatic plug

This is the device that the CoSSMic team at SINTEF has been experimenting with. The drawbacks is that it has a unit price of almost 500 NOK and must be imported. Covering most home devices with such a measuring device would be extremely costly. As hardware devices were not a part of the project scope, the team did not do any research on how to interface with this unit. More details about this can be found in appendix G.8.

Do It Yourself Arduino

A much cheaper, but perhaps limited solution, is one from Open Energy Monitor [64]. These units do not support the ability to control devices, only measurement of consumption. They are much cheaper than the HomeMatic unit, but must be assembled manually. The unit can be good for experimenting and prototyping but it is far from ready for the home market.

9 Retrospect

This chapter presents an evaluation of the project and process. Here, the time and resource allocation, decisions and their impact, and occurring risks will be evaluated. Lastly, the finished product will be reviewed.

9.1 The team's allocation of time and resources

This section presents an overview of actual time spent and evaluates the allocation of resources.

9.1.1 Time spent

At the beginning of the project, the team made an estimation on how much time that would be spent on each section of the project. This is shown in table 3.2. The actual distribution is shown in figure 9.1.

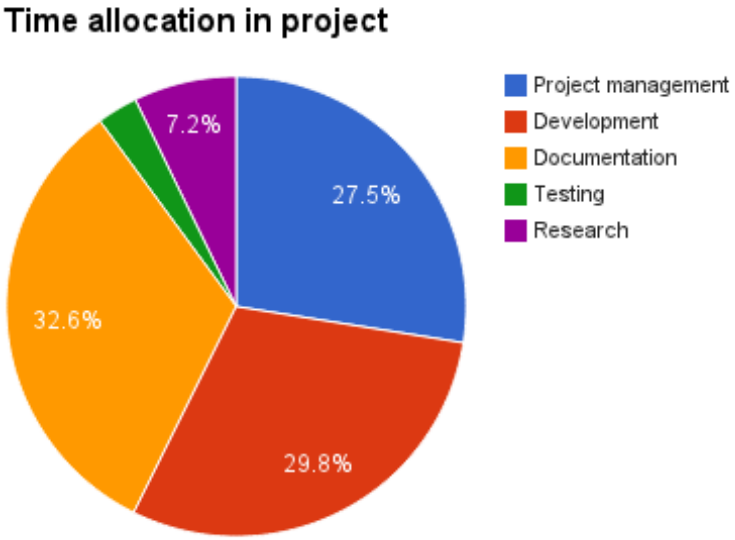


Figure 9.1: Pie chart of time spent on different parts of the project

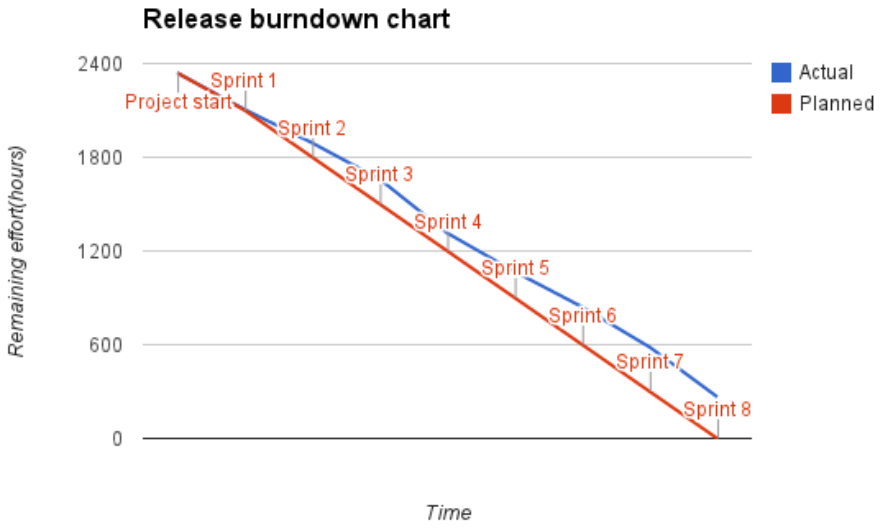


Figure 9.2: Project release burn down chart

When comparing the actual time spent to the estimated time, it is clear that the team's initial estimations does not match the actual time spent. This is illustrated in figure 9.2.

There are several reasons for this, including unplanned absence of team members, underestimation of tasks and too heavy workload due to deliveries in other classes. The subsequent sections will elaborate on some of these deviations.

Deviations in testing

One of the larger deviations is the time devoted to testing. The time logged for testing is more than three times lower than the 10% that was estimated. This is because the time logged for testing only was time that was spent exclusively on testing, mainly including user testing. The acceptance testing with the customer was performed during the customer meetings and is therefore logged as project management. Much of the time spent on running functional tests was also logged as development instead of testing. With this in mind, the actual time spent on testing is closer to the estimated amount.

Deviations in project management

The largest abnormality lies in the project management, logging a total of 27.5% of the spent time, which is five times higher than the original estimate.

There are several factors for this. The most influential factor was poor estimation done in an early stage of the project, due to lack of experience.

The project management consists mainly of hours logged for meetings. Many of the meetings had agendas that could classify them as either development or research. For example, the early prototyping of the user interface was done during prolonged meetings, and was logged as project management.

In retrospect, the project management should have been specified differently, or the meetings should have been dedicated to development or research, and time spent should be logged accordingly.

9.1.2 Resource allocation

As described in section 1.4, the team had several resources available, including a supervisor, and the customer. The team had internal meetings at least two times a week, and an eight hour work session once a week. In addition, weekly meetings with the customer and meetings with the supervisor every other week were held.

Externally

During the meetings with the supervisor, the team were given valuable input and feedback regarding customer relations, and the content and structure of the report. The role of the supervisor was unclear to the team in the early stages of the project. It was quickly learned that he was more a guide, rather than a supervisor. The only real oversight the supervisor had was the activity report, which included a work log and a summary of the progress made since the last meeting.

It was of great value to the team that the customer was easily accessible. This made regular meetings possible. Communicating with the customer on a regular basis reduced the probability of misunderstandings to arise. This increased the likelihood that the team would deliver a product the customer would be satisfied with.

Internally

One of the most valuable decisions that was made during this project was to have fixed meeting times. This assured that the entire team was updated and involved. On these internal meetings, the team planned and distributed tasks among the team members, and discussed occurring issues. Each team member was assigned an area of responsibility, so that no important parts of the project would be neglected. It also turned out to be a way to keep the motivation high and further involve the team members.

9.2 Project management

This section will evaluate the team's project management process.

9.2.1 Preventive measures

In order to ensure that the productivity continuously peaked, preventative measures were taken early on in the project. One of these measures were that the team instated a monetary penalty for arriving late to meetings or not delivering work on time. The money accumulated from this was later spent on a team building. The other measure was that one team member was responsible for bringing snacks and making sure the team took regular breaks during the long work sessions on Wednesdays. This arrangement circulated throughout the project. The team believe that both of these measures had a good effect, and that they increased the productivity and mood during work sessions.

9.2.2 Working with the customer

Initially, there were some problems with the customer not reaching deadlines set in conjunction with the team. This effected the team's research in the early stages of the project. Getting information about the CoSSMic group's research earlier in the project might have changed the course of the product.

9.2.3 Changing roles

In the beginning of the project, the team distributed roles with different areas of responsibility, as described in table 1.3.2. Almost halfway through the project it became clear that the Scrum master at the time, Per Øyvind, also was the team member with the most experience in Android development. This resulted in a lot of extra work on the Scrum master, while the deputy project leader only had a few responsibilities. The team reviewed the project roles and the workload distribution, which resulted in a change of roles. The team came to the conclusion that Lars Erik, the now former deputy project leader, was best fit to take over the role as Scrum master. After this change, the team's best Android resource could focus entirely on the Android development, and make himself available to help others. The team was satisfied with this decision, and efficiency and productivity increased after the change in roles.

9.2.4 Choice of development methods

As development method, the team used Scrum together with XP, described in section 6.2.

Although Scrum is a well known development method, a lot of the team members had different expectations to how the framework would be implemented in the project. This resulted in a lot of excess discussions and was recognized as a risk (id #8 in table 3.3).

The team resolved this issue by consulting with the supervisor. Besides this, Scrum worked well as a project management framework. Combining this with our adaptation of Extreme Programming led to a very agile and responsive development process.

9.2.5 Underestimation of workload

A prevalent risk to any project is the failure to acknowledge the cost of tasks in the project, and the improper allocation of resources that follows such an error. This can also lead to a false sense of what a team can achieve in a given time frame.

The team experienced this risk (id #1 in table 3.3) to have the biggest negative impact on the work flow. Even though it did not have a detrimental effect on the sprint end result, it was the most occurring risk, which added up to be very time consuming. The main reason for this issue was the lack of experience with many of the tasks, but also that initially the team did not consider who the tasks were assigned to. This turned out to play an important role in the equation. Learning from previous mistakes, the team started looking at previous sprint backlogs to help with the estimation of the tasks.

9.2.6 Choice of Scrum tool

As the team did not have daily meetings, a good Scrum tool was needed to keep track of progress. Having bad experiences with Scrum tools from previous projects, the team decided to spend some time to research the possible candidates. After Yodiz was chosen, the team were confident that this was the best decision, considering all available options. This still might be true, but using Yodiz nonetheless proved to be somewhat frustrating for the team members. An example of an event that caused a lot of frustration in the early stages of the project is given in the subsequent section. Another example is that the burn down charts that Yodiz generated turned out to be useless for documentation. This led to the team having to do the burn down charts manually. As a matter of fact, a digital Scrum tool is very hard to design in an intuitive fashion. Combined with poor support for simultaneous editing and a lot of down-time for the web-page, the result was a system that caused problems.

However, Yodiz was the only tool that covered all the requirements set by the team, and despite its flaws, it did its job. Ideally, the team would have liked to have a meeting room with a Scrum board so the use of such an extensive Scrum tool could be avoided altogether. At the end of the project the team started using a Scrum board in the meeting room that was used during the work sessions. The team could have started using this earlier in the project.

Improper use of Yodiz

Despite the team's efforts to get acquainted with Yodiz, a misunderstanding arose, and was not discovered until the end of the second sprint.

The misunderstanding, displayed in figure 9.3, 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 was assigned as owner of the task.

As a result, it appeared as if only single individuals performed the tasks, even though the entire team in reality was participating. This was also reflected in the burn down charts.

To sort out this issue, the team went through old meeting reports and time sheets to figure out which members of the team that had actually participated on the particular task. With this information, new tasks for each team member was created.

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

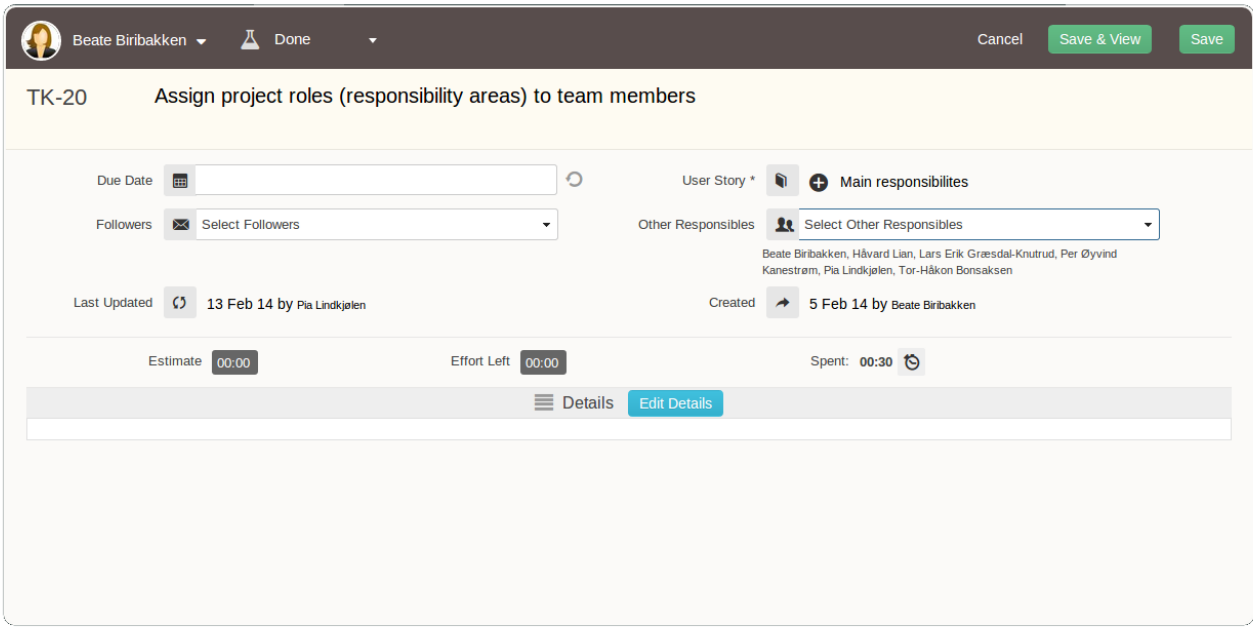


Figure 9.3: Example screenshot to illustrate improper use of Yodiz

9.3 Technical choices

Even though the team aimed to make informed decisions before the development started, some changes had to be made during the project. This section presents an evaluation of these decisions.

9.3.1 Android development

When it comes to the Android development, a great deal of the decisions made turned out to be reasonable. Few of the team members had previous experience with development in Android.

Each team member spent a significant amount of time getting up to speed with the Android platform and the front-end architecture of Android apps. In retrospect, more time should have been spent getting all team members up to date and practicing Android programming together in the beginning of the project. One of the team members had some experience with Android development and ended up spending a lot of time teaching and explaining to the other team members. This could have been organized in a way, for instance by holding a workshop in the beginning. Learning Android development properly from the start could have saved a lot of time.

Development tools

The team agreed on using Android Studio after a brief discussion on whether to use Android Studio or Eclipse with ADT.

At the start of the project, the team experienced some issues with Android Studio - some of the team members had trouble compiling the app. This was caused by an Android Studio version incompatibility, resulting in Gradle not being able to setup the project. Resolving this issue, took more time than estimated. In retrospect, the team has learned that setting up and learning to use a new IDE is more time consuming, than first expected.

9.3.2 Server development

The decision to implement the server in Dropwizard is reasonable as long as the server is used to provide a RESTful interface. There are other choices, but every solution usually includes some, or even all of the libraries used in Dropwizard. The server could be created from scratch and use a stand-alone HTTP server. This was not feasible in the time frame of this project, and would have been a waste of resources considering how well Dropwizard works.

9.3.3 Device monitoring

In the beginning the team designed a complete architecture with a server, Android app, and a monitoring device that communicated with a server. However, device monitoring turned

out to be unrealistic within the project's time frame. The team decided to drop the feature entirely after consulting with the customer. As mentioned in section 4.1, this requirement was optional. After learning more about the CoSSMic project, the team realized that a solution for the complete architecture was already under development. This solution opened the possibility of interfacing to these devices, or at least design the server to be compatible with it. This information was unfortunately made available too late in the project. Because of this, the team was not able to include this functionality into the product. The team's device monitoring concepts is described in section 8.3.

9.4 Miscellaneous issues

The following sections represent the remaining challenges that was encountered, and how these challenges were solved.

9.4.1 Unplanned absence

One of the team members, Lars Erik, had to make a sudden trip abroad due to a death in the family. He was away from the team for a total of twelve days. Prior to departure the team assigned Lars Erik with lighter workload, in accordance with the risk(7) in table 3.3. 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.

9.4.2 Workplace issues

The team had booked meeting rooms three times a week in NTNU's IT building to host the work sessions. Unfortunately, the building was under renovation for the entire duration of the project. Booking other rooms that fit the teams needs so late in the semester proved to be difficult. This issue led to periods where the team were disturbed a lot by construction, which in turn lead to team members growing tired and being a bit more on edge than usual. To counter this, the team instated a cake policy. Every Wednesday a team member would bring something good to eat as well as making sure that the other team members took regular breaks. This helped ease the tension of the work sessions.

9.4.3 Traveling to China

In the beginning of April the entire team left for a school trip to China. This meant that the team would be unavailable for a total of 17 days, including weekends and the Easter holidays. The team decided to not expect that any work would be done during the trip. The team

increased the number of work hours from 20 to 25 per week as described in section 1.4.2. Had the team not traveled to China, the project would have had 9 sprints worth a total of 2160 hours to spend on the project. The team's attempt to compensate for the school trip left the team with 8 sprints, giving a total of 2340 working hours, as shown in table 1.3. This means that the team projected a usage of 180 hours more than the course required.

9.5 Product evaluation

Considering all factors discussed so far in this chapter, the team was content with the final product. The finished product consists of the Android app, the server and documentation for both, and lastly this report.

As with most projects, the team wishes it had accomplished more of its goals. There is some functionality the team would have liked to complete given different circumstances. The team would like to one day see the app fully developed with hardware components that allow automatic measurement and control.

The customer, and the team were pleased with the finished product. The customer wants to integrate the project in the work the SINTEF team has already done for the CoSSMic project. The customer encouraged the team to commercialize the product. A customer approval letter was given by the customer to include in this report. This can be found in appendix A.

The last contact with the customer was done as a presentation to SINTEF Energy and the people working at CoSSMic. After presenting the product, the team had a more in depth discussion with the CoSSMic team on how they had solved the different issues related to creating the entire solution.

Glossary

API is short for application programming interface (API). It specifies how the software components in a system should interact with each other..

burn down charts are graphical representations of the remaining work versus time.

C is a general purpose programming language.

C++ is a general purpose object oriented programming language.

C# is a multi-paradigm programming language.

CRUD is an acronym for create, read, update and delete.

CSS or Cascading Style Sheets is a style sheet language used for describing the look and formatting of a document written in a markup language.

data access object is an object that provides an abstract interface to a database or other persistent storage. It maps application method calls to persistent storage.

Django is a high level Python Web framework.

gamification is the use of game thinking and game mechanics in non-game contexts to engage users in solving problems. Further descriptions about this subject may be found in reference [65].

HTML or HyperText Markup Language is the standard markup language used to create web pages.

HTTP or Hyper Text Transfer Protocol is a common protocol for network communication.

Java is a object oriented programming language.

JavaScript is a dynamic programming language.

JSON is textual representation of the objects and standard way to store and transfer data between different components.

Linux is a open source operating system.

Mobil app is a computer program designed to run on smartphones, tablet computers, and other mobile devices. The term ”app” is a shortening of the term ”application software”.

MySQL is a relational database management system.

navigation drawer is a panel that displays the app's main navigation options.

PHP is a scripting language used for web development.

Python is a general purpose object oriented programming language.

Slug is a part of an URL that identifies a resource and is human readable.

URI is an acronym for uniform resource identifier. Is a string of characters pointing to a specific resource.

Use case is a list of steps, defining interactions between different roles using a software system and a software system. The roles can be users or external systems.

VHDL is a hardware description language.

Bibliography

[1] Collaborating Smart Solar-powered Micro-grids (CoSSMic). www.cossmic.eu.

[2] SINTEF. www.sintef.no/sit.

[3] Student Media AS. mediastud.no and ibok.no.

[4] Sportradar. sportradar.com.

[5] Drift- og utviklingskomiteen (dotKom). https://wiki.online.ntnu.no/projects/18/wiki/DotKom.

[6] Casual Gaming. www.casualgaming.no.

[7] NTE miniSolo energydisplay. www.nte.no/index.php/no/privat/2013-04-10-06-20-41/energidisplay.

[8] Theowl. www.theowl.com.

[9] Smartly. www.smartly.no.

[10] Open energy monitor. openenergymonitor.org.

[11] Efergy. efergy.com.

[12] GitHub. https://github.com.

[13] Subversion (SVN). subversion.tigris.org.

[14] Mercurial. mercurial.selenic.com.

[15] Eclipse IDE. https://www.eclipse.org.

[16] NetBeans. https://netbeans.org.

[17] IntelliJ. www.jetbrains.com/idea.

[18] Android Studio. developer.android.com/sdk/installing/studio.html.

[19] Maven. maven.apache.org.

[20] Gradle. www.gradle.org.

[21] Java code convention.
www.oracle.com/technetwork/java/codeconv-138413.html.

[22] Android code convention.
https://source.android.com/source/code-style.html.

[23] Apache. www.apache.org.

[24] Jersey. https://jersey.java.net.

[25] Jackson. https://github.com/FasterXML/jackson.

[26] Jetty. www.eclipse.org/jetty/.

[27] JDBI. jdbi.org.

[28] Volley. https://android.googlesource.com/platform/frameworks/volley/.

[29] aChartEngine. www.achartengine.org.

[30] PagerSlidingTabStrip. https://github.com/astuetz/PagerSlidingTabStrip.

[31] ProgressFragment. https://github.com/johnkil/Android-ProgressFragment.

[32] android-segmented-control.
https://github.com/hoang8f/android-segmented-control.

[33] android-spinnerwheel. https://github.com/ai212983/android-spinnerwheel.

[34] Dropwizard. dropwizard.codahale.com.

[35] Facebook SDK. https://developers.facebook.com/docs/android/.

[36] JavaDoc.
www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html.

[37] Yodiz. www.yodiz.com.

[38] Google Drive. https://drive.google.com.

[39] LaTeX. www.latex-project.org.

[40] Todonotes. www.ctan.org/tex-archive/macros/latex/contrib/todonotes/.

[41] Aspell. aspell.net.

[42] BibTex. www.bibtex.org/.

[43] Glossaries. www.ctan.org/pkg/glossaries.

[44] Google groups. https://groups.google.com.

[45] I. Sommerville, *Software Engineering*, pp. 58–68 and 72. Addison-Wesley, 9 ed., 2010.

[46] M. Cohn, *Agile estimating and planning*. Prentice Hall, 1 ed., 2005.

[47] Android device fragmentation.
https://developer.android.com/about/dashboards/index.html.
Retrieved March 15, 2014.

[48] Android best practices for performance.
https://developer.android.com/training/best-performance.html.
Retrieved March 25, 2014.

[49] Android documentation.
http://developer.android.com/guide/components/fragments.html.

[50] ContentProvider.
developer.android.com/reference/android/content/ContentProvider.html.
Retrieved March 15, 2014.

[51] LoaderManager.
developer.android.com/reference/android/app/LoaderManager.html.
Retrieved March 15, 2014.

[52] T. Reenskaug, "The model-view-controller (mvc) - its past and present," tech. rep., University of Oslo, 2003.

[53] OAuthV2.0. http://oauth.net/2/.

[54] Account authenticator class.
http://developer.android.com/reference/android/accounts/AbstractAccountAuthenticator.html.
Retrieved May 26, 2014.

[55] Android Manifest. http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[56] C. Snyder, "Paper prototyping," *developerWorks*, November 2001.

[57] PROTOIO Inc. www.proto.io.

[58] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," *Proceedings of ACM INTERCHI'93 Conference (Amsterdam, The Netherlands*, pp. 206–213, April 1993.

[59] S. Dray and D. Siegel, "Remote possibilities? international usability testing at a distance," *Magazine Interactions*, vol. 11, pp. 10–17, March 2004.

[60] R. Patton, *Software Testing*. Sams, 2 ed., 2005.

[61] AMS. http://www.nve.no/no/kraftmarked/sluttbrukermarkedet/ams/.

[62] Untappd. https://untappd.com/.

[63] Raspberry Pi. http://www.raspberrypi.org/.

[64] Open Energy Monitor - Modules. http://openenergymonitor.org/emon/Modules.

[65] G. Zichermann and C. Cunningham, *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*. O'Reilly Media, 1 ed., 2011.

A Customer recommendation letter

I have cooperated with group 16 in IT2901 in the role of customer representing SINTEF. Our task was the development of a mobile application to support awareness of own electricity consumption at home. The task was related to a European project, called CoSSMic, led by SINTEF that is about allowing private houses produce and consume solar energy. Motivational apps have shown to have some effect in changing habits and forming new habits. Our task was to create a prototype that allowed users to become aware of their own consumption patterns at home, and compare their own consumption to that of others in similar situations. The prototype developed by the group will be used as input to the development of an application in CoSSMic that will be used in real-world trials in Germany and Italy later this year.

Group 16 has worked in an organized way and we have had weekly communication in form of Scrum meetings where in almost all of the meetings I have been presented a new version of the application. We have had fruitful discussions where I have learned tremendously and our ideas at SINTEF have matured as a result. The application itself has become an attractive result that can be used by its own and has a potential in the market. I have got the impression that the group dynamics have been good and all of the group members have contributed to the discussions and to the results. It has been a pleasure to work with group 16 and I hope we will have opportunities for cooperation in the near future.

B Meetings with external resources

B.1 Meetings with supervisor

The team had meetings with the supervisor every other week throughout the project, except for the period when the team was absent for the school trip to China. After this period, the semester was ended, and all meetings after this point was initiated exclusively by the team. All these meetings were held in the break room area outside the supervisor's office.

Week #	**Date**	**Summary**
5	31.01	Get familiar with supervisor's tasks and what we were to write in the different project documents requested by the subject administration.
7	14.02	Feedback on report, tips on how we could improve our risk analysis and guidance on how to approach customer if he does not provide us with necessary information.
9	27.02	Feedback and questions on mid-term report.
11	14.03	Feedback on report, especially regarding requirement specification and how we should represent the development process in the report.
13	03.04	General feedback on report and status update before the team left for China.
19	09.05	Last meeting with supervisor. Feedback and questions on the final report.

Table B.1: Overview on meetings with supervisor

B.2 Meetings with customer

These meetings were held in a room at NTNU that was provided by the subject's administration. Initially, they were held every week, until the customer said he did not feel it was necessary to have these meetings that often. This was later withdrawn, and we went back to our original agreement.

Date	**Summary**
24.01	Introductory meeting with customer. Team and customer got to know each other. Assignment presented and discussed. Customer wanted team to brainstorm to figure out what should be in the app.
31.01	Team introduced the concepts they reached after brainstorming. Also discussed existing solutions and possible architectural solutions for the app.
07.02	Discussed back-end solutions. Clarifications regarding documentation and information about the customer's research program.
14.02	Team suggested deadline for changes in requirement specification, approved by customer. Discussed architecture, whether GUI should be vertical and/or horizontal and also if we only should focus on developing the app for mobile phones.
21.02	Demonstration of prototype. Customer approved requirement specification, had some feedback on the prototype that led to changes in the specification. Discussed possible hardware solutions and what to focus on for the next iteration.
14.03	Customer reviewed app. Approved of the name the team came up. Wanted to use the meetings on Fridays as a sprint meeting, review the requirement specification and make a list of what we should prioritize in the following sprint.
28.03	Demonstration and review of the app.
02.04	Last customer meeting before team left for China. Customer wants to do usability testing with his co-workers at SINTEF. Requests that we apply textual descriptions on sites that are not completely finished.
02.05	Demonstration and review of the app. Updated Kanban-board. Customer requested a thorough description of further development in the report and on GitHub-wiki.
16.05	Final demonstration of app. Discuss further development.
23.05	Presentation and demonstration of app on SINTEF. Also met with CoSSMic, and discussed and explaned technical aspects.

Table B.2: Overview on meetings with customer

C Example of project documents

C.1 Meeting report example

Meeting minutes

Team meeting

Date	21.01.14
Present	Beate Baier Biribakken, Tor-Håkon Bonsaksen, Lars Erik Græsdal-Knutrud, Per Øyvind Kanestrøm, Håvard Holmboe Lian, Pia Karlsen Lindkjølen
Absent	
Meeting agenda	1. Meeting times 2. Exchange contact information 3. Development environment 4. Project roles 5. The assignment 6. Other issues 7. Next time

1 Meeting times

After a brief discussion and having each team member review their time schedule, we concluded that the following times for weekly team meetings would suit the entire team's schedule:
- Mondays: 14:15-16:00
- Wednesdays: 12:15-14:00

Should we not get a proper meeting room booked for these times, we will discuss whether we will have a meeting on Sundays as well.

2 Exchanging contact information

All the team members write down their phone number. We decide that the primary communication will go through e-mail. Per Øyvind will create an e-mail-list for the team on Google groups(it2901@googlegroups.com).

Phone numbers:
Per Øyvind: 92295543
Pia: 97421726

Tor-Håkon: 99441554
Håvard: 99426621
Lars Erik: 97654693
Beate: 95444440

3 Development environment

We decide to use Git for both the application development and the report. Håvard will create a private repository and the other team members as users on GitHub.

4 Project roles

Both Pia and Beate have talked to people who previously have taken the course that suggested it would be a good idea to give each team member an area to be responsible of. After a brief discussion, we ended up with these delegations:

Customer relations: Pia
Project leader: Pia
Scrum-master: Per Øyvind
Development chief: Tor-Håkon
Report editor: Beate
Test chief: Håvard
Deputy project leader: Lars Erik
Secretary: Circulates

5 The assignment

5.1 What do we think the assignment is about?

Compare yourself to others and see their usage. The competitive aspect. How to save power. Set up measurements to save power, checkbox. Simulator?

Usage data. Typical usage for different housing types, energy marking, age groups, marital status, how many residents that lives in each residence, in which time period the usage is at its greatest, power included in rent. Typical good power saving measurements. Data represented in charts. How much power does my neighbors consume?

5.2 What should we learn more about before our first meeting with the customer?

Technical solutions:
- Authentication. OAUTH2.0 fb, google
- API against electricity suppliers?

6 Other issues: Questions for the customer

- How will we retrieve information about power measurement?
- Do you have a free room for us to work in?
- Did you have a specific target group in mind?
- Do you have any specific graphical user interface requirements?
- Do you have any usage data we can use in the application?
- What do you want to compare? (kwh/month, how much the users have improved, percentagewise?)

7 For the next time

- Find a fixed meeting time with the customer. Pia will send a mail to find a time for our first meeting.
- Find a tool for Scrum. Suggestion: GitHub
- Development tools - Eclipse or IntelliJ?
- Set up local Git repository
- Choice of rooms, if not assigned one by IDI.
- Think about project roles and what they involve.

C.2 Status report example

SINTEF_ENERGY
STATUS REPORT # 5
Week 19

Agenda	1. Introduction 2. Progress summary 3. Open/closed problems 4. Allocation of time in project 5. Planned work for next period 6. Updated risk analysis

1 Introduction

The team is currently working on sprint #7. What mostly remains is adding some refinements to the product and ensure that we deliver both a satisfactory product and documentation of the creation process. During sprint 6 we focused on assembling and integrate the different parts of the code and replace dummy data with actual feed from Facebook and the server. Some of time was also used to restructure and review the content in the final report.

2 Progress summary

Beate

During this sprint, I've mainly worked with the comparison functionality and layout. A lot of time were spent on solutions that were later restructured or removed, especially when replacing dummy data, such as actual Facebook profile pictures and charts in stead of pictures. I also set up the layout for the home-tab. I addition, I've spent some time reviewing the report and identified tasks that needs to be done to complete a satisfactory report.

Håkon

I have mainly focused on working with the graphs that represent the energy usage of a user. They have been changed to adapt better to irregular user input and give a consistent look. I have also been working on improving the GUI for the graphs to conform with Android design guidelines. Lastly I have done a lot of work on optimilization by shortening the load times for the graphs. This was done by rewriting the queries responsible for building the graphs.

Lars

I have focused on improving the profile tab, with a few tasks in Facebook integration. Adding logic for handling the administration of different residences in all layers has been one of the larger tasks. Another focus has been replacing the static dummy list of friends with an actual list of friends that have the application installed. This functionality includes the feature to create and retrieve Facebook users from our database.

Håvard

Synchronization was my main task throughout this sprint. I implemented the server side support for synchronizing all data that our app collects on the client side. This means that users can have the app on multiple devices and still have persistent data. I have also done some work on solving known bugs in the code, especially in the section of code that handles power saving tips.

Per Øyvind

I have handled the client side synchronization. I wrote code for asynchronous building of the graphs to conserve resources, and also fixed memory leaks and bugs in this area. Using Android Lint, I refactored a lot of code and corrected errors that Lint has pointed out. Some of the models have also been rewritten for optimization.

Pia

I have worked on the administration of devices. The layout for the presentation of devices has been revamped several times. Full support for adding, editing and deleting devices has also been implemented.

3 Open/closed problems

Closed
Synchronization
Home tab

Opened
Complete implementation of residences

4 Updated risk analysis

The team did not think it necessary to update the risk analysis this time.

5 Allocation of time in project period

As shown in the time allocation overview in table 1, there was some divergence between how much time we estimated to use and how much time we actually spent. This sprint the entire team missed out on the two first days of the sprint, as we were travelling from China at the time.

Unfortunately, this resulted in a somewhat incomplete estimation of the sprint. However, we had taken into consideration that we would in fact be affected by the school trip, hence the sprints with durations of 50 hours from sprint 2 and to and including sprint 8.

The table also shows that we planned to focus on usability testing with actual users. Much time was used on planning these tests, but the team was unsatisfied with the application's completeness, and decided to focus on getting it ready in stead of spending time on usability testing we suspected would not be of much help. That is also why we postponed the review of the design consistency.

The amount of time spent on meetings diverges because we missed two meetings, but also because we added an extra hour before the last customer meeting to prepare for the demonstration for the customer.

ID	**Hours estimated**	**Hours spent**
User testing	18	5
Review requirement specification	4	8
Facebook integration	5	3.75
Development	123.2	150
Design	6	0
Report	14.5	10
Meetings	60	47
Total	230.75	223.75

Table 1: Time allocation overview for sprint 6

6 Planned work for next period

In the following weeks we will finish up the remaining code and run functional tests on the system before presenting it to the customer for acceptance testing. Part of the group will be working full time on the report to make sure it is possible to finish it the week we have from product delivery to project deadline. What remains in implementation is bundling some loose ends in the code and some functionality. The report has a lot of content, but we are adding progress chapters for sprints and other content that we find necessary.

D User Manual

D.1 Starting up the app

After you have installed Wattitude on your phone, as explained in chapter E.2.1, you can start using the app.

Firstly, since this version of the app does not have support for directly connecting the app to power measuring devices in your home, you need to add data. This involves adding all the different devices you want to collect usage for and their usage. The usage of a device is added for each day, and if you do not know your past usage, it is possible to add an average usage. This may be found on a note by the power connector on the device.

D.1.1 Logging on to Facebook

Logging on to Facebook is done by entering the "login" option in the drawer meny. This will show a log in window where you need to enter your username and password. Then press the "Log In" button. This log in window is displayed in figure D.1

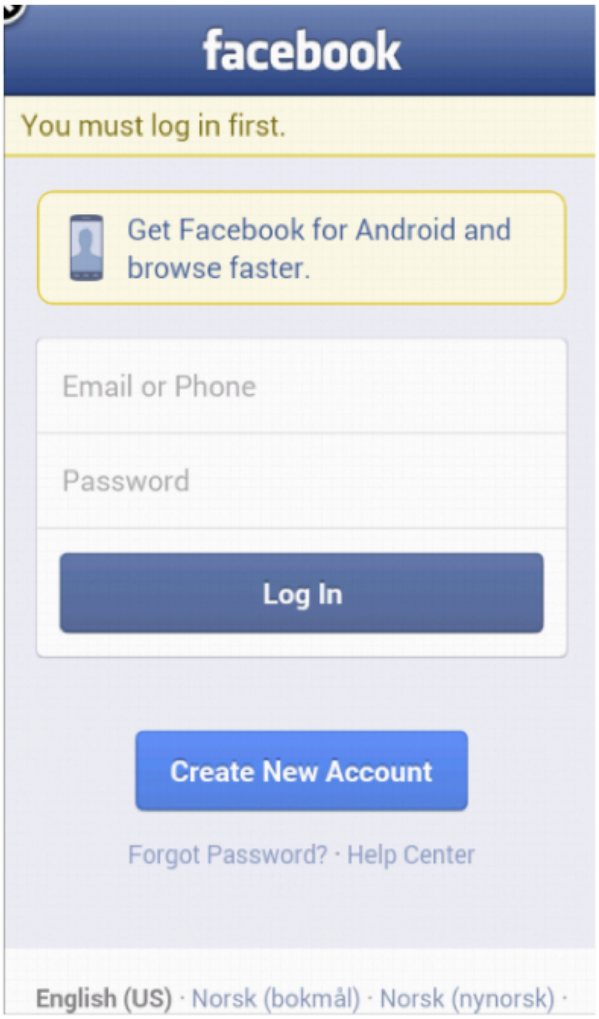


Figure D.1: Screenshot of the Facebook login screen

D.2 Devices

The device tab includes a collection of all your devices, sorted by their category. In this tab you can manage all of your devices. This include adding new devices, editing the devices you have and deleting your devices.

D.2.1 Adding a new device

To add a new device press the ”Add devices” symbol in the option menu. This will make a dialog appear, displayed in figure D.2. Add the name of the device you want to add and what category the device falls under. You may also add a description of the device.

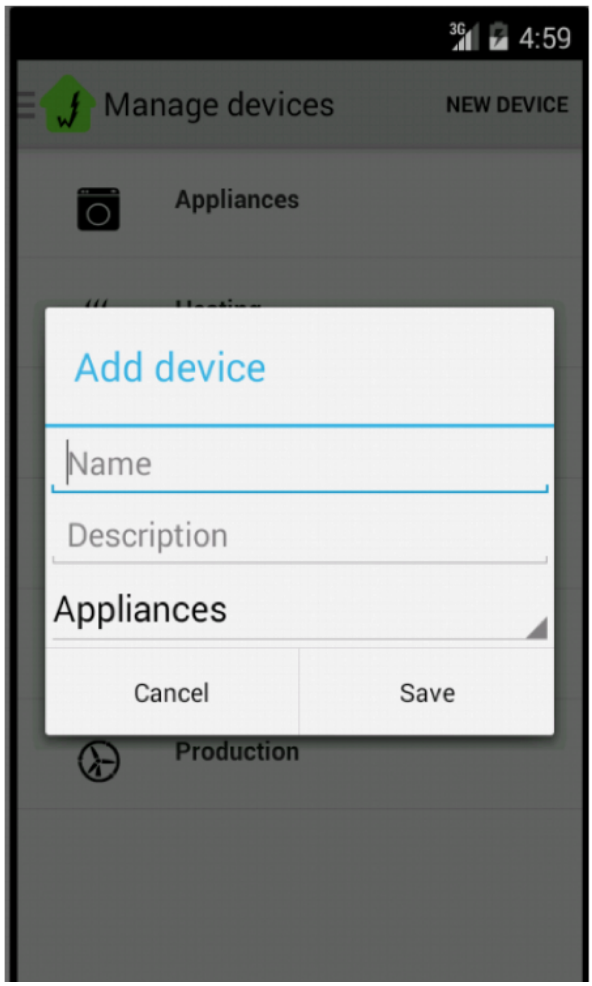


Figure D.2: Screenshot of add device-dialog

D.2.2 Editing a device

To modify a device already added to your list of devices, it is necessary to select the device you want to modify. This is done by holding your finger over it. This results in a dialog that gives you the option to edit or delete the device you selected, displayed in figure D.3. Tap the ”Edit” button and a new dialog will appear. Enter the new values you want to save to the device you chose.

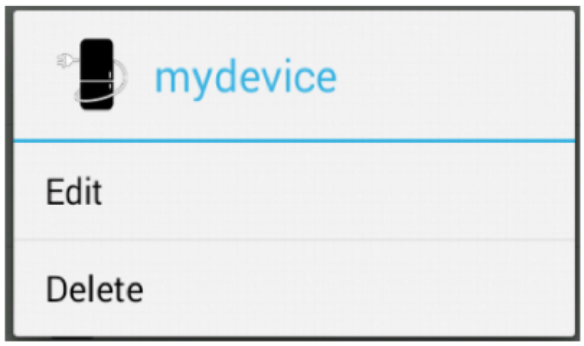


Figure D.3: Screenshot of choose-dialog with edit and delete

D.2.3 Deleting a device

To delete a device in your list of devices and press the device you want to delete. Then select ”Delete” in the dialog that appears. A dialog asking if you really want to delete the device will show, displayed in figure D.4. Press ”OK”, and you will have deleted your device.

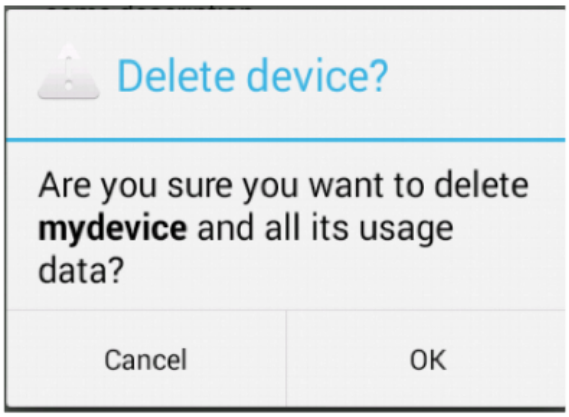


Figure D.4: Screenshot of delete-dialog

D.3 Usage

The usage tab will give you an overview of your usage. You can see the usage of one or several devices, or you can see your total usage. Here, you can also share your usage with your Facebook friends.

D.3.1 Adding usage

To add usage to a device, it is important to first have created a device to add usage to. How to do that is described in section D.2. Add usage to your device by selecting the device from the drop down list, pick a date, the amount of energy consumed and lastly press the ”Add usage” button at the bottom of the screen.

Figure D.5: Screenshot of adding usage

D.3.2 Examine your own usage

You can examine your usage in two different ways. You can look at the usage graph in the usage tab, within the drawer menu user tab. It is also possible to look at how your consumption is distributed between all of your devices. This is done in the distribution tab, within the drawer menu usage tab.

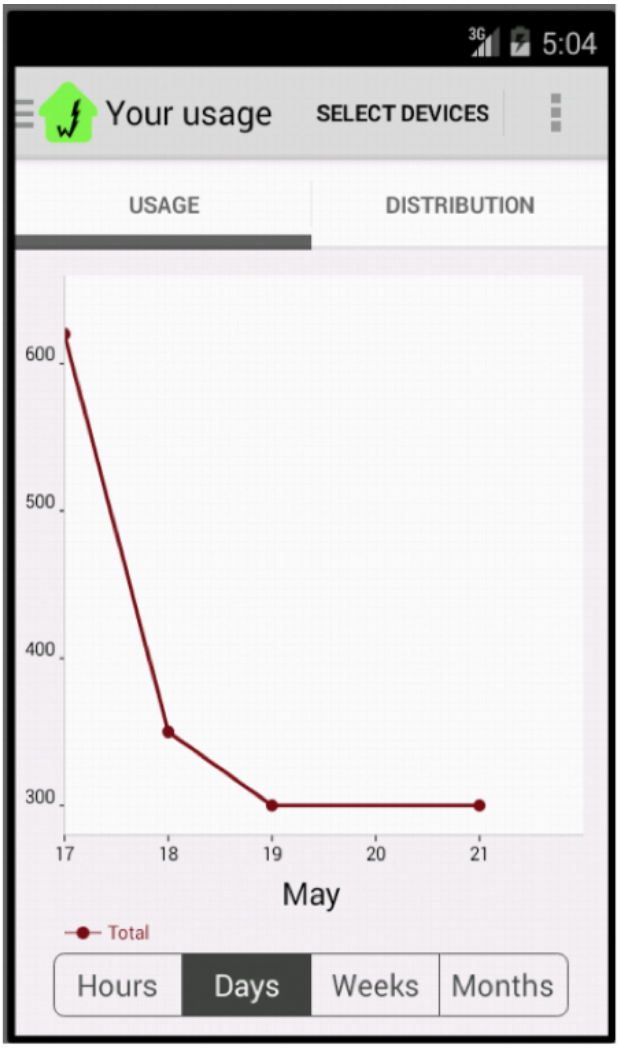


Figure D.6: Screenshot of usage graph

D.4 Exchange tips

Within the exchange tips tab, you can manage all of your tips. Tips is meant to be advise that will help you save energy around your house, when completed.

D.4.1 Creating your own tip

To create your own tip, press the ”Add tips” button in the option bar. This will create a dialog where you can enter a name of your new tip, and a description.

D.4.2 Adding a tip to your list of tips

Your list of tips is a list of the tips you want to perform in you house. To add a tip to your list of tips, you select the tip you want to add, and a dialog will appear. Select the ”Add to my tip” option. The tip you added should now appear in the "My tips" tab.

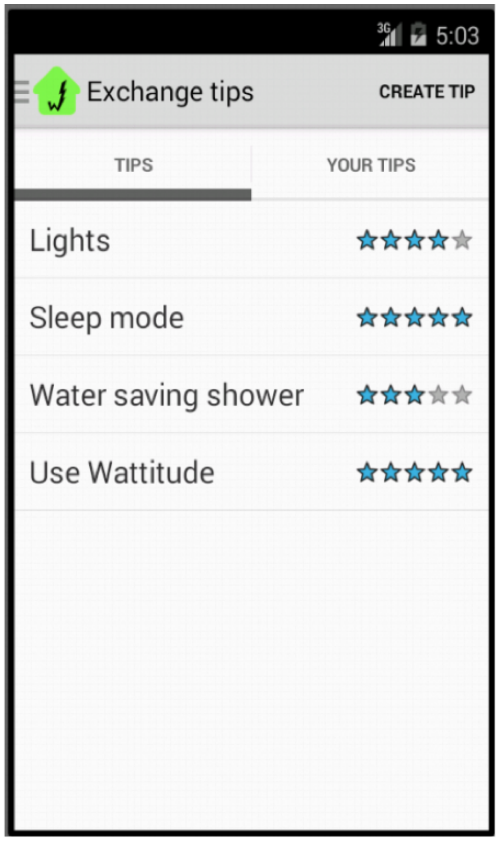


Figure D.7: Screenshot of list of tips with rating bar

D.4.3 Indicate that you have performed a tip

To indicate that you have performed a tip, you check the checkbox beside the tip.

D.4.4 Rate a tip

To change the rating of a tip you hold your finger over the tip you want to change and a dialog will appear.

E Technical documentation

The Wattitude server is a RESTful service written in Java and based on the Dropwizard framework. It acts as a HTTP server that receives and answers requests with JSON objects.

E.1 Installation requirements

The Wattitude server has the following requirements to the environment:

- Java version 7
- Apache Maven
- Git scm tool
- MySQL server
- OS-independent

E.2 Installation procedure

E.2.1 Installing the app on your phone

To install Wattitude on your phone you need the apk-file downloaded on your phone and start it. This will automatically launch the apk-installer and voila, you can start using Wattitude!

The following instructions assume that the previously listed requirements are installed and configured correctly. This is especially important to note on Windows systems as they require modification of PATH.

Clone repository

The systems source code is easily accessible on GitHub. Navigate to desired directory and run "git clone git@github.com:haavardlian/UbiSolar.git"

Build project

The systems server should be build using Apache Maven. Navigate into the "server" directory and build the server by running "mvn package"

Configure server

Configure the included config.json file, found in the root of the server project, to match the settings of your database and http port needs. Run "java -jar target/server-1.0.jar db migrate config.json" from the root directory to create the tables needed for running the server.

E.3 Running the server

Running the server is done, simply by navigating into the newly built "server" directory and run "java -jar server-1.0.jar server config.json".

F Use cases

ID: 1	**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 registered 2. Devices are connected to energy measurement device
Input	Registered energy usage
Post-conditions	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 registered. Average consumption is shown. 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 F.1: Textual use case 1

ID: 2	**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 registered 2. Devices are connected to energy measurement device 3. Devices are active
Input	Registered energy usage
Post-conditions	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 F.2: Textual use case 2

ID: 3	**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.
Input	Description of users Power Saver.
Post-conditions	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.

Table F.3: Textual use case 3

ID: 4	**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.
Input	User's choice.
Post-conditions	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.

Table F.4: Textual use case 4

ID: 5	**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
Input	Users credentials for accounts in question.
Post-conditions	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 app to post

Table F.5: Textual use case 5

ID: 6	**Connecting with friends whom are using the app**
Actor	User
Description	The user should be able to find and connect to his friends through Facebook.
Preconditions	1. User is logged in.
Input	User's Facebook credentials.
Post-conditions	A list of friends currently using the app.
hline **Basic flow**	1. The user taps the social tab
Alternative flow	1.1. The user does not have any friends currently using the app

Table F.6: Textual use case 6

ID: 7	**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
Input	The users energy consumption data or power savers.
Post-conditions	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 app

Table F.7: Textual use case 7

G Sprint backlogs and burn down charts

This appendix contains sprint backlogs and burn down charts for all the sprints.

G.1 Sprint 1

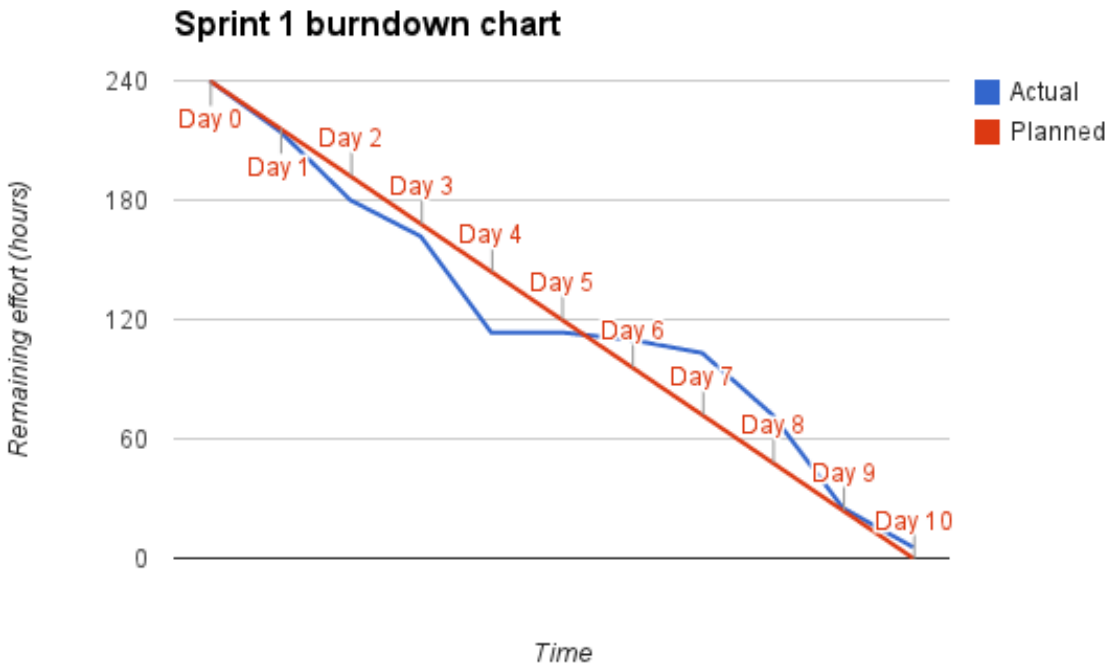


Figure G.1: Burn down chart for sprint 1

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
5	**Risk analysis**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1.25**	**0**	**0.75**
	translate risk analysis	0	0	0	0	0	0	0	0.5	0	0.5
	revise risk analysis	0	0	0	0	0	0	0	0.75	0	0.25
6	**Team organization**	**0**	**0**	**3**	**0**	**0**	**0**	**0**	**1.5**	**0**	**0**
	provide background info	0	0	0	0	0	0	0	1.5	0	0
	delegating roles	0	0	3	0	0	0	0	0	0	0
7	**Main responsibilities**	**6**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	discuss responsibility arease	3	0	0	0	0	0	0	0	0	0
	assign responsibility areas to team members	3	0	0	0	0	0	0	0	0	0
8	**Dev environment**	**0**	**0**	**0.25**	**0**	**0**	**2**	**1**	**1.25**	**0**	**0**
	write about dev environment	0	0	0.25	0	0	2	1	1.25	0	0
9	**Requirements**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1.5**
	revise requirements	0	0	0	0	0	0	0	1	1	0
	digitalize from paper	0	0	0	0	0	0	0	0	0	1.5
10	**Alternative solutions**	**0**	**30**	**14**	**0**	**0**	**0**	**0**	**20**	**36**	**2**
	research alt. solutions	0	30	13	0	0	0	0	18	36	0
	write about alt. solutions	0	0	1	0	0	0	0	2	0	2
11	**Problem specification**	**0**	**0**	**0**	**0**	**0**	**0**	**3**	**0.25**	**0**	**0**
	write about customer	0	0	0	0	0	0	0	0.25	0	0
	write about assignment	0	0	0	0	0	0	3	0	0	0
12	**Architecture**	**0**	**1**	**0**	**0**	**0**	**1.75**	**0**	**0**	**1**	**0**
	research power measurement	0	1	0	0	0	0	0	0	0	0
	draw architecture diagrams	0	0	0	0	0	1	0	0	0.5	0
	write about architecture	0	0	0	0	0	0.75	0	0	0.5	0
13	**Project management**	**0**	**0.5**	**1**	**0**	**0**	**0**	**0**	**4.25**	**3**	**0**
	write about scrum	0	0	1	0	0	0	0	1.5	3	0
	create wbs-diagram	0	0.5	0	0	0	0	0	2.75	0	0
15	**Draw prototypes**	**6**	**0**	**0**	**48**	**0**	**0**	**0**	**0**	**0**	**0**
	discuss what to include in project	6	0	0	48	0	0	0	30	0	0
18	**Set up dev environment**	**7.5**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
	create and cloned git-repo	1.75	0	0	0	0	0	0	0	0	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	set up android environment	2	0	0	0	0	0	0	0	0	0
	set up latex environment	4	0	0	0	0	0	0	0	0	0
	set up SCRUM-environment	0	0	0	0	0	0	1	0	0	0
39	**Report**	**6**	**3**	**0**	**0.25**	**0**	**0**	**1.6**	**2.6**	**5.85**	**15.5**
	proofreading and spell check	0	0	0	0	0	0	0	1.1	1.85	3
	bibliography	0	0	0	0	0	0	0	0.25	0	0
	general structure	2	3	0	0	0	0	0	0.5	4	10.5
	styling	4	0	0	0	0	0	0	0	0	0
	add in latex	0	0	0	0.25	0	0	1.6	0.75	0	1
	Sum	25.5	34.5	18.25	48.25	0	3.75	6.6	31.1	46.85	19.75

Table 1: Sprint 1 backlog

G.2 Sprint 2

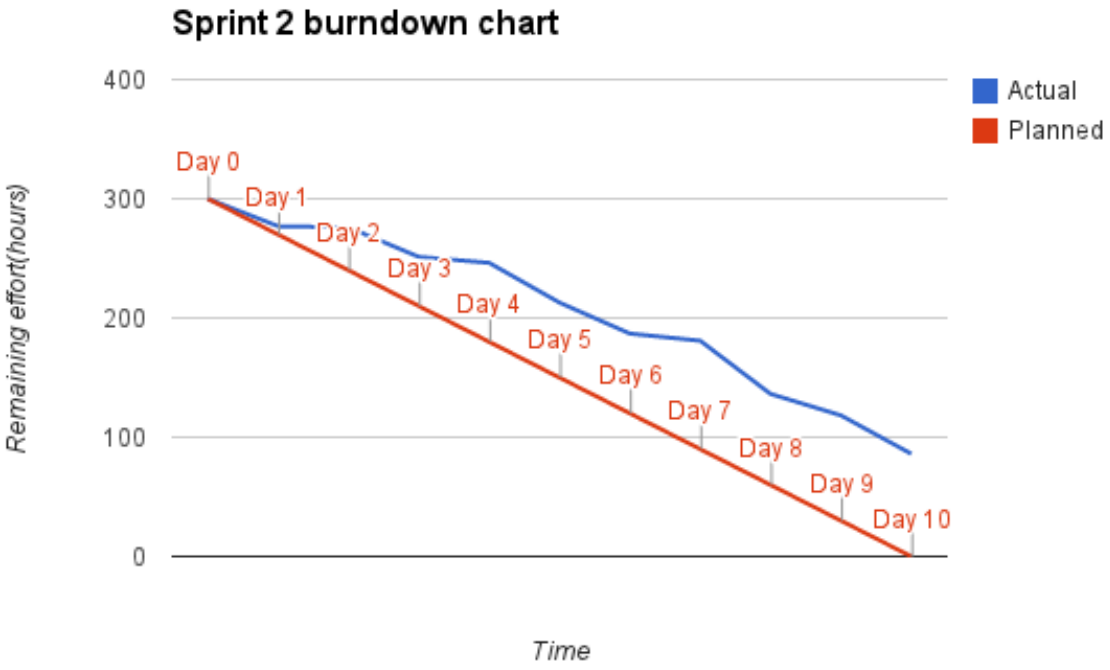


Figure G.2: Burn down chart for sprint 2

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x	Meetings	0	0	0	0	12	0	0	0	0	6
x	Planning	18	0	12	0	0	12	0	12	0	12
3	User can see his personalia	0.25	0	4	0	1	0	0	0	1	0
	Find graph libary	0.25	0	0	0	0	0	0	0	0	0
	Add profile fields	0	0	0	0	1	0	0	0	1	0
	Learn to use graph library and set it up	0	0	4	0	0	0	0	0	0	0
40	User has a log in screen	0.5	0	0	0	1	0	0	0	0	0
	Placeholder log in screen	0.5	0	0	0	1	0	0	0	0	0
41	User has a usage tab	1.5	0	0	0	1	2	0	0.5	4	0
	Graph showing total usage	0	0	0	0	0	0	0	0	4	0
	Buttons for changing views to 'last twelve months'	0	0	0	0	0	0	0	0.5	0	0
	Buttons for changing views to 'show devices' and the fragment	0	0	0	0	1	0	0	0	0	0
	Familiarize with the code and what we should do from here	0	0	0	0	0	2	0	0	0	0
	Implement to all activites	1.5	0	0	0	0	0	0	0	0	0
43	User menu has a search field	0	0	1	0	0	0	0	0	0	0
	Search bar in the drawer	0	0	1	0	0	0	0	0	0	0
44	User has a app on android to start	0.25	0	2	0	10.85	0	0	0	0	1.5
	Create the project files	0.25	0	0	0	0	0	0	0	0	0
	Localization	0	0	0	0	0.1	0	0	0	0	0
	Add all fragments	0	0	0	0	0.75	0	0	0	0	0
	Get development tools up and running	0	0	0	0	6	0	0	0	0	0
	Set up dev. environment for application	0	0	2	0	4	0	0	0	0	1.5
47	Mid-term report	2.5	0.25	0.75	3	0	5	6	7.1	0	5.25
	Add listings in table of contents	0	0	0	0	0	2.5	0	1.5	0	0
	Test plan	0	0	0	0	0	0	2.5	0	0	0
	project management	1.5	0.25	0.75	0	0	0	0	0	0	2.25
	Use case	0	0	0	0	0	2.5	0	2	0	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Update programs and tools used	0	0	0	0	0	1	0	0	0	1.5
	revise non-functional requirements	0	0	0	0	0	0	0	0	0	0.5
	Formalize the functional requirements and add use case text to LaTeX	1	0	0	0	0	0	1.25	1.75	0	0
	Updated risk analysis	0	0	0	3	0	0	0	0	0	0
	Check if tasks in sprint 1 are assigned to correct persons	0	0	0	0	0	0	1	0	0	0
	General layout	0	0	0	0	0	0	0.25	1.1	0	0.5
	Write about how the team is going to compensate for the lost days caused by field trip to China.	0	0	0	0	0	0	1	0	0	0
	Explain more througly about the planning poker	0	0	0	0	0	0	0	0.25	0	0
	Separate general Scrum-material from sprint specific Scrum-material	0	0	0	0	0	0	0	0.5	0	0
	Write about non-functional requirements and how we are going to test them	0	0	0	0	0	0		0	0	0.5
48	Server	0	0	4	2	6	4.5	0	13	0	7.5
	Create server project	0	0	2	0	0	0	0	0	0	0
	Add database functionality to server	0	0	2	0	0	0	0	0	0	0
	Design server database layout	0	0	0	0	0.5	0.5	0	0.5	0	1
	Design internal server structure	0	0	0	0	1.5	0	0	7.5	0	0
	Find server software	0	0	0	2	0	0	0	0	0	0
	Implement client connection	0	0	0	0	0	4	0	0	0	0
	Reading documentation	0	0	0	0	4	0	0	0.5	0	0
	Research Dropwizard	0	0	0	0	0	0	0	0.5	0	0
	Write tests	0	0	0	0	0	0	0	4	0	0
	Add client side methods	0	0	0	0	0	0	0	0	0	4
	Server debugging	0	0	0	0	0	0	0	0	0	2.5

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
49	User can plot in data	0	0.75	0	0	0	1.5	0	9	1.5	0
	User can add energy data	0	0	0	0	0	0	0	9	0	0
	SQLlite database schema for energy data	0	0	0	0	0	1.5	0	0	1.5	0
	Provider wrapped around energy DB	0	0.75	0	0	0	0	0	0	0	0
50	Design prototype for sprint 2 functionality	0	0	1	0	1.5	0	0	0	0	0
	User has a menu	0	0	0.25	0	0	0	0	0	0	0
	User har usage graph	0	0	0.75	0	0	0	0	0	0	0
	Transferred prototype on paper onto computer (took pictures)	0	0	0	0	1.5	0	0	0	0	0
52	User has device tab	0	0	0	0	0	0	0	3	11.5	0
	Create device list	0	0	0	0	0	0	0	0	3	0
	Making it possible to add devices to the list	0	0	0	0	0	0	0	3	0	0
	Making it possible to add usage per device	0	0	0	0	0	0	0	0	1	0
	Doing research about android components	0	0	0	0	0	0	0	0	4	0
	Making it possible to see detailes on each device from the list	0	0	0	0	0	0	0	0	3.5	0
	SUM	23	1	24.75	5	33.35	26	6	44.6	18	32.25

Table 1: Sprint 2 backlog

G.3 Sprint 3

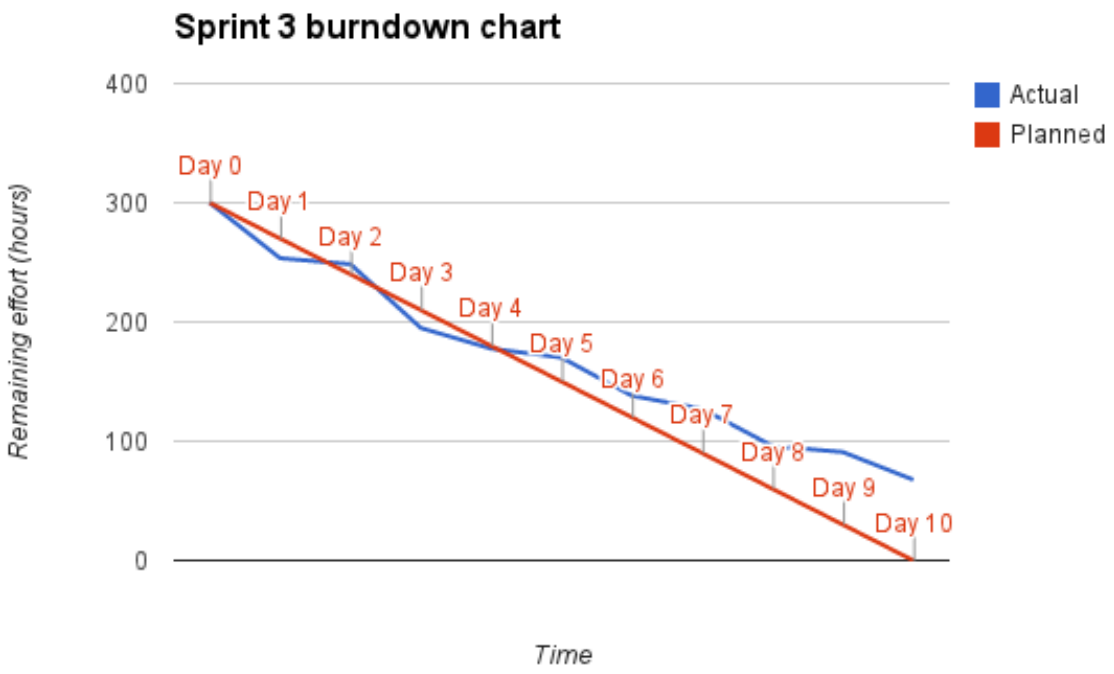


Figure G.3: Burn down chart for sprint 3

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
51	**Planning**	**12**	**0**	**6**	**0**	**0**	**12**	**0**	**12**	**0**	**0**
52	**Meetings**	**0**	**0**	**0**	**6**	**0**	**0**	**0**	**0**	**0**	**0**
53	**Administration**	**0**	**0**	**2**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	mail correspondance	0	0	0.5	0	0	0	0	0	0	0
	status report	0	0	1.5	0	0	0	0	0	0	
55	**Branding**	**0**	**0**	**12**	**0**	**0**	**0**	**0**	**8.25**	**0**	**0**
	find new app name	0	0	12	0	0	0	0	0	0	0
	make icon packs	0	0	0	0	0	0	0	8.25	0	0
56	**Mid-term report work**	**4**	**2**	**2**	**1.85**	**2.5**	**7.5**	**2**	**0**	**5**	**5**
57	**Prototyping**	**30.5**	**3**	**31.5**	**9.5**	**5**	**10**	**0**	**0**	**0**	**5**
	find online site for prototyping	0.5	2	0	0	0	0	0	0	0	0
	research general ui guidelines	12	1	2	1	3	0	0	0	0	0
	decide on general ui guidelines	18	0	0	0	0	0	0	0	0	0
	decide on functionality we want	0	0	8	0	0	0	0	0	0	0
	prototype device tab	0	0	3	0	2	10	0	0	0	5
	prototype social tab	0	0	5	1	0	0	0	0	0	0
	prototype graph functionality	0	0	5	3	0	0	0	0	0	0
	prototype usage tab	0	0	6	2	0	0	0	0	0	0
	revise profile tab	0	0	2.5	2.5	0	0	0	0	0	0
58	**Concept usage tab**	**0**	**0**	**0**	**0**	**0**	**2.5**	**7**	**0**	**0**	**1**
	change graph library	0	0	0	0	0	0.5	0	0	0	0
	make usage tab use new graph library	0	0	0	0	0	2	2	0	0	0
	zooming changes data correctly	0	0	0	0	0	0	5	0	0	1
59	**Concept profile tab**	**0**	**0**	**0**	**6**	**0**	**0**	**0**	**2**	**1.5**	**4.25**
	research	0	0	0	0	0	0	0	2	1.5	0.75
	implement profile GUI	0	0	0	0	0	0	0	0	0	1.5
	implement facebook API	0	0	0	0	0	0	0	0	0	2
61	**Concept device tab**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**5**	**0**	**0**
	research	0	0	0	0	0	0	0	5	0	0
62	**The server site**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**4.5**	**0**	**6**

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	update database schema	0	0	0	0	0	0	0	1	0	1
	database/server fixes	0	0	0	0	0	0	0	1	0	0
	add tips to server	0	0	0	0	0	0	0	0	0	2
	add ratings to server	0	0	0	0	0	0	0	0	0	3
	research	0	0	0	0	0	0	0	2.5	0	0
69	**Concept social tab**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**2**
	research	0	0	0	0	0	0	0	0	0	2
	Sum	46.5	5	53.5	17.35	7.5	32	11	40.75	5	23.25

G.4 Sprint 4

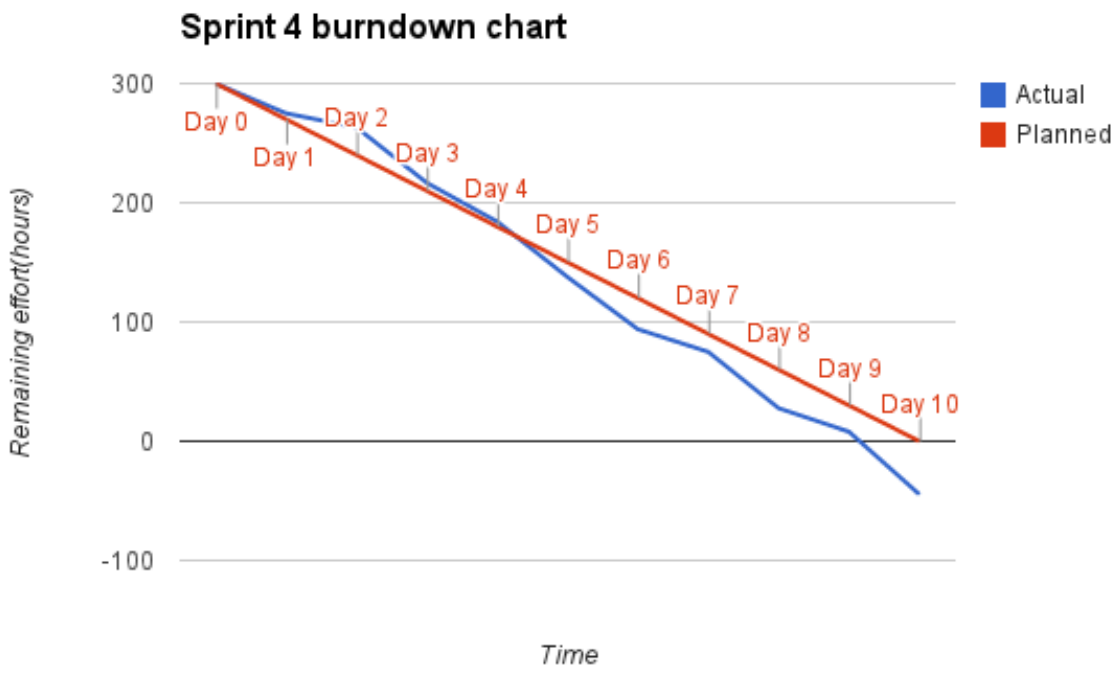


Figure G.4: Burn down chart for sprint 4

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x	**Planning**	**12**	**0**	**0**	**0**	**0**	**12**	**0**	**0**	**0**	**0**
x	**Meetings**	**0**	**0**	**0**	**0**	**12**	**0**	**0**	**0**	**0**	**0**
53	**Administration**	**0**	**0**	**0**	**2**	**0**	**0.25**	**0**	**0**	**0**	**0**
	mail correspondence	0	0	0	0	0	0.25	0	0	0	0
	status report	0	0	0	2	0	0	0	0	0	0
64	**Concept usage tab**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	work on the design	0	0	0	2.5	0	0	0	0	0	0
	zooming changes data correctly	2	0	0	0	0	0	0	0	0	0
	list to show devices	0	0	0	3	0	0	0	0	0	0
	filter data on choosen devices	5	0	4	0	0	0	0	0	0	0
	add usage view	0	0	4.5	0	0	0	0	0	0	0
	rewrite to use ContentProvider	0	3.5	0	0	0	0	0	0	0	0
	implement database data into multiple device filtering	0	0	0	5	0	0	0	0	0	0
	bug fixes	0	0	0	2	2	0	0	1	0	0.5
	implement database data into pie chart	0	0	0	0	0	0	4	0	0	0
	dont reset the chart when new devices are added	0	0	0	0	0	0	0	3	0	0
	optimalize data simulating	0	0	0	0	0	0	0	6	0.75	0
	save state on activity swap	0	0	0	0	0	0	0	0	0	1
	save state on application exit	0	0	0	0	0	0	0	2	0	0
	tabs	0	0	0	0	0	0	0	0	0	6
	hook up the graphs to use the compressed data from the database	0	0	0	0	0	0	0	0	0	2
	separate device selection for pie and line graph	0	0	0	0	0	0	0	0	0	2
67	**Concept profile tab**	**0**	**0**	**9.75**	**0**	**0**	**0**	**1**	**0**	**0**	**11.5**
	research Android	0	0	0	0	0	0	1	0	0	3.5
	implement the profile GUI	0	0	2.75	0	1	0	0	0	0	3.5
	implement Facebook authorization	0	0	7	0	0	0	0	0	0	4.5
68	**The user wants icons**	**0**	**0**	**0**	**0**	**0**	**2**	**0**	**0**	**0**	**0**

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	change icons to fit in app	0	0	0	0	0	2	0	0	0	0
72	**Concept social tab**	**0**	**0**	**0**	**0.25**	**0**	**0**	**0**	**0**	**7.5**	**5**
	Implement sosial GUI	0	0	0	0	0	0	0	0	7.5	5
	add scrollbar	0	0	0	0.25	0	0	0	0	0	0
73	**Report**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	General layout fixes	0	0	1.8	0	0.25	0	0	0	0	0
	Revise architecture	0	0	0.5	0	0	0	0	0	0	3
	Add description of MVP to report in LaTeX	0	0	0	0	0	2	0	0	0	0
	Make class diagrams	0	0	1	0	1.25	0	0	0	0	0
	Make gantt diagram	0	0	1	0	0	0	0	0	0	0
	proofreading	0	5.5	11.75	0	2.5	13.5	1.25	0	0	6
	todonotes	0	0.25	0	0	0.5	0	0	0	0	0
	Revise and restructure sprints section	0	0	2.75	0	3	0	0	0	0	0
	Switch bibliography	0	0	0	2.5	0	0	0	0	0	0
	burndown charts	6	2	1	4	3	0	0	2	2	0
	Add activity plan	0	0	0	0	2	0	0	0	0	0
	Refrase section wbs-section	0	0	0	0	2.75	0	0	0	0	0
	Add section about extreme programming	0	0	0	0	0	4	0	0	0	0
	Add testplan	0	0	0	0	0.5	0	0	0	0	0
	glossary	0	0	0	0	3.25	0	0	0	0	0
	Complete the library section	0	0	0	0	1	0	0	0	0	0
	Take screen shots of proto.io	0	0	0	0	0	0.25	0	0	0	0
	Add description of each risk in risk table (beneath the table)	0	0	0	0	0	2.5	0	0	0	0
	Reorganize the structure of the report	0	0	0	0	4	4	0	0	0	0
	Write about what the different requirements is	0	0	0	0	0	1	0	0	0	0
	Convert extreme programming practices to table	0	0	0	0	0	0	0.75	0	0	0
	Write about sprints	0	0	0	0	7	0	0	0	0	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Report structure cleanup	0	0	0	0	0	2.25	0	0	0	0
76	**Peer-evaluation**	**0**	**0**	**1.5**	**0**	**0**	**0**	**0**	**20.25**	**0**	**0**
	Evaluate other group's report	0	0	0	0	0	0	0	11.25	0	0
	Watch the presentation	0	0	0	0	0	0	0	9	0	0
	Prepare presentation	0	0	1.5	0	0	0	0	0	0	0
77	**Git fixing**	**0**	**0.5**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	rebase and FF master into develop	0	0.5	0	0	0	0	0	0	0	0
78	**Concept device tab**	**0**	**0**	**4**	**6**	**0**	**0**	**9**	**11**	**0**	**0**
	Add category to devices	0	0	0	0	0	0	1	0	0	0
	Put the "add device-" button to the top toolbar	0	0	0	4	0	0	0	0	0	0
	Research MVP and make the code follow this pattern	0	0	0	2	0	0	0	0	0	0
	Research the content provider and change it into giving categories to devices	0	0	0	0	0	0	0	5	0	0
	Research the provider and the changes we need to do	0	0	2	0	0	0	0	0	0	0
	Change the structure of the code	0	0	2	0	0	0	0	0	0	0
	Change the devicelist to use expandablelist	0	0	0	0	0	0	8	6	0	0
80	**Server**	**0**	**0**	**3**	**0**	**0**	**0**	**0**	**0**	**0**	**3**
	Add JavaDoc to server	0	0	1	0	0	0	0	0	0	1.5
	Restructure server and database	0	0	2	0	0	0	0	0	0	1.5
83	**User testing**	**0**	**0**	**0**	**0**	**0**	**0**	**0.25**	**0**	**0**	**0**
	Create test cases	0	0	0	0	0	0	0.25	0	0	0
84	**Overall app code**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**4**	**0**
	Change drawer fragment placements	0	0	0	0	0	0	0	0	0.5	0
	Changing tabs in drawer uses backstack	0	0	0	0	0	0	0	0	2	0
	Drawer uses icons	0	0	0	0	0	0	0	0	1.5	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
87	**User can get and create tips**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**1**	**12.5**
	Create tips in database/server with volley	0	0	0	0	0	0	0	0	0	1.5
	Get tips from database with volley	0	0	0	0	0	0	1	2	1	2
	Add user interface for tips	0	0	0	0	0	0	0	0	0	4
	Add tips to database/server	0	0	0	0	0	0	0	0	0	5
	SUM	25	11.75	46.55	32.5	46.5	43.75	19	47.25	19.75	52.5

G.5 Sprint 5

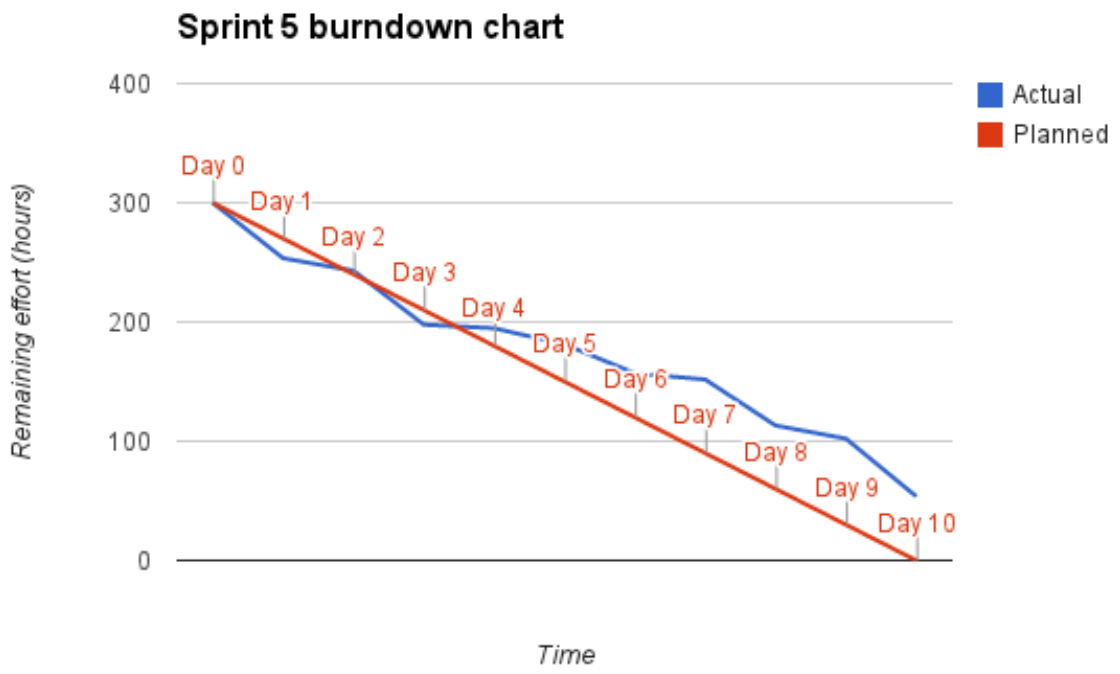


Figure G.5: Burn down chart for sprint 5

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x	**Planning**	**12**	**0**	**0**	**0**	**6**	**12**	**0**	**0**	**0**	**0**
x	**Meetings**	**0**	**0**	**0**	**6**	**6**	**0**	**0**	**0**	**0**	**0**
88	**The user wants icons**	**0**	**0**	**0**	**0**	**0**	**5**	**0**	**1**	**0**	**0**
	create icons	0	0	0	0	0	1	0	1	0	0
	invert and fix size	0	0	0	0	0	4	0	0	0	0
89	**Concept profile tab**	**0**	**0**	**5**	**0**	**0**	**2**	**0**	**3**	**4**	**0**
	Implement the profile GUI	0	0	0	0	0	2	0	3	4	0
	Facebook auth look better	0	0	5	0	0	0	0	0	0	0
90	**Concept device tab**	**1.5**	**0**	**11**	**0.5**	**1**	**3**	**0.75**	**8**	**0**	**0**
	bug fixes	0	0	0	0	0	0	0	5	0	0
	research	0	0	0	0	0	0	0	2	0	0
	add the data we want to show in the dropdown datafield	0	0	1	0	0	0	0	0	0	0
	change the devicelist to use expandablelist	0	0	2	0.5	1	0	0	0	0	0
	add hamburger with the right values	1.5	0	5	0	0	0	0	0	0	0
	make the deviceModel have categoryfield	0	0	3	0	0	0	0	0	0	0
	make the list be sorted by categories	0	0	0	0	0	3	0.75	1	0	0
92	**Concept social tab**	**5**	**2.5**	**6.5**	**0**	**0**	**0**	**2**	**0.5**	**0**	**0**
	implement social GUI	5	2.5	6.5	0	0	0	2	0.5	0	0
93	**Concept usage tab**	**4**	**5**	**9.5**	**1**	**0**	**2**	**2**	**13**	**2**	**7**
	Save state on activity swap	0	5	0	0	0	0	0	0	0	0
	Make the graph bether with manually added points	0	0	2	0	0	0	0	4	0	0
	tabs	0	0	1	0	0	0	0	0	0	0
	Database date query needs to filter on selected devices	2	0	0.5	0	0	0	0	0	0	0
	Fragment state behavior	2	0	6	1	0	0	0	0	0	0
	Implement time selection in the pie graph	0	0	0	0	0	2	0	4	0	0
	Implement zoom in the pie chart	0	0	0	0	0	0	0	3	0	1
	Get a picture of the graph view	0	0	0	0	0	0	0	0	0	3

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Fix manual usage adding	0	0	0	0	0	0	0	0	2	1
	Resolution handling rewrite	0	0	0	0	0	0	0	2	0	0
	Graph edge case fixing	0	0	0	0	0	0	2	0	0	1
	Pie chart cleanup	0	0	0	0	0	0	0	0	0	1
94	**Administration**	**0**	**0**	**1.75**	**0**	**0**	**0**	**0**	**1.1**	**0**	**1**
	mail correspondance	0	0	0.25	0	0	0	0	1.1	0	0
	scrum-master tasks	0	0	0	0	0	0	0	0	0	0
	status report	0	0	1.5	0	0	0	0	0	0	1
95	**User testing**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0.5**
	write user stories for user tests	0	0	0	0	0	0	0	0	0	0.5
97	**The user can get tips**	**0**	**0**	**5**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	ratings are shown and updated throughout session	0	0	5	0	0	0	0	0	0	0
98	**The user wants to be social**	**0**	**0**	**4.5**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Look into facebook api for posts and photos	0	0	4.5	0	0	0	0	0	0	0
100	**Restructure the menu**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**5.5**	**0**	**0**
	update the menu with new look and items	0	0	0	0	0	0	0	5.5	0	0
117	**Final version of report**	**24**	**3**	**2**	**1**	**6.5**	**0**	**0**	**0**	**0**	**40.5**
	add review of main responsibility	0	0	0	0	0	0	0	0	0	0.5
	look through previous reports	0	0	0	0	0	0	0	0	0	40
	review requirements	8	0	0	0	6	0	0	0	0	0
	fix burndown charts	0	0	0	0	0.5	0	0	0	0	0
	read example reports	3.5	3	2	1	0	0	0	0	0	0
	discuss report changes	8	0	0	0	0	0	0	0	0	0
	restructure report based on other reports	4.5	0	0	0	0	0	0	0	0	0
	SUM	46.5	10.5	45.25	3	13.5	24.25	5.25	38.6	12	48.5

Table 1: Sprint 5 backlog

G.6 Sprint 6

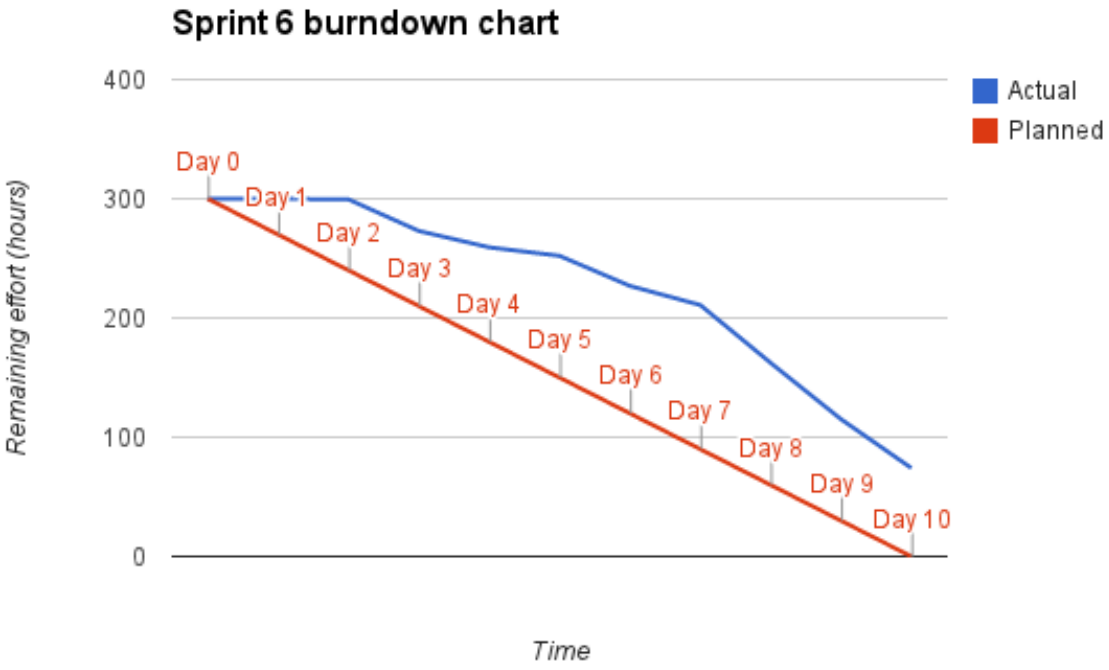


Figure G.6: Burn down chart for sprint 6

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x	Planning	0	0	12	0	0	12	0	12	0	0
x	Meetings	0	0	0	0	0	0	0	0	0	6
101	Replace dummy data with data from server	0	0	4	0	0	5.25	5.25	8.5	0	10
	Backend sync logic	0	0	0	0	0	5.25	0	0	0	5
	Syncprovider perform sync	0	0	4	0	0	0	5.25	8.5	0	0
	Make tips work properly	0	0	0	0	0	0	0	0	0	5
103	User testing	0	0	0	0	0	0	0	0	4.5	0.5
	perform tests	0	0	0	0	0	0	0	0	0	0.5
	prepare tests	0	0	0	0	0	0	0	0	4.5	0
123	Share usage on Facebook	0	0	0	0	0	0	0	4.25	0	0
	Research facebook methods	0	0	0	0	0	0	0	2.5	0	0
	Post on facebook wall	0	0	0	0	0	0	0	1.75	0	0
124	Language support	0	0	0	0	0	0	0	2	0	1.25
	Cleanup the strings.xml	0	0	0	0	0	0	0	1	0	0
	Translate the Usage tab	0	0	0	0	0	0	0	1	0	0
	translate all strings in social context to norwegian	0	0	0	0	0	0	0	0	0	1.25
125	Review of requirements	0	0	0	0	0	0	0	0	8	0
118	Usage tab	0	0	0	1.25	0	3	5	11	20	2
	Work on design	0	0	0	1.25	0	3	2	2	7	0
	Fix addUsageTab to add manual points correctly	0	0	0	0	0	0	3	0	0	0
	Usage cleanup	0	0	0	0	0	0	0	0	2	0
	Rewrite the usage tab to use a set of points instead of all	0	0	0	0	0	0	0	9	6	2
	Implement segments	0	0	0	0	0	0	0	0	5	0
114	Profile tab	0	0	0	8.75	0	2.25	0	0	1	4.5
	Retrieve facebook info when logged in	0	0	0	3.25	0	1.5	0	0	0	0
	Profile tab backend	0	0	0	5.5	0	0	0	0	1	0
	Finish profile tab design	0	0	0	0	0	0.75	0	0	0	0
	Add support for editing residences	0	0	0	0	0	0	0	0	0	4.5
120	Social tab	0	0	4.5	2.5	3	1	2.5	4.75	10.5	11.5

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	add "back" button in compare to others	0	0	0	0	0	0	0	1	0	0
	Add actual facebook data. Picture and friends list.	0	0	0	0	0	0	0	0	0	10.5
	comparison settings	0	0	4.5	2.5	1.5	0	0	0	0	1
	add comparison settings to hamburger menu(filter)	0	0	0	0	0	0	0	0	7.5	0
	fix of compare to friends	0	0	0	0	0	0	0.75	0.75	3	0
	move comparison settings to be above graph in comparison to others	0	0	0	0	0	0	0	3	0	0
	Change the similar profiles tab to contain the comparison settings and a link to chainge them	0	0	0	0	1.5	1	1.75	0	0	0
121	Final version of report	0	0.5	0	1.25	4.1	1.25	1	1	1.25	2.5
	add examples pf meeting reports + agenda	0	0	0	0.5	1.5	0.75	0	0	1.25	0
	set up outline for statusreport	0	0	0	0	0	0	0	1	0	2.5
122	Home tab	0	0	0	0	0	0.5	0.25	0	0	2.75
	Fix events on the serverside	0	0	0	0	0	0	0	0	0	2
	set up xml	0	0	0	0	0	0.5	0.25	0	0	0
	setup hard-coded newsfeed	0	0	0	0	0	0	0	0	0	0.75
110	Device tab	0	0	6	0	0	0	2	5	2	0
	work on design	0	0	1.25	0	0	0	0	0	0	0
	Split producing devices from other devices	0	0	1	0	0	0	0	0	0	0
	Make each category contain all the devices in that category	0	0	2.5	0	0	0	0	0	0	0
	Update Android studio	0	0	0.5	0	0	0	0	0	0	0
	Insert the correct categories	0	0	0.75	0	0	0	0	0	0	0
	Make the devices editable	0	0	0	0	0	0	2	5	0	0
	Add icon to the categories	0	0	0	0	0	0	0	0	2	0
	SUM	0	0.5	26.5	13.75	7.1	25.25	16	48.5	47.25	41

Table 1: Sprint 6 backlog

G.7 Sprint 7

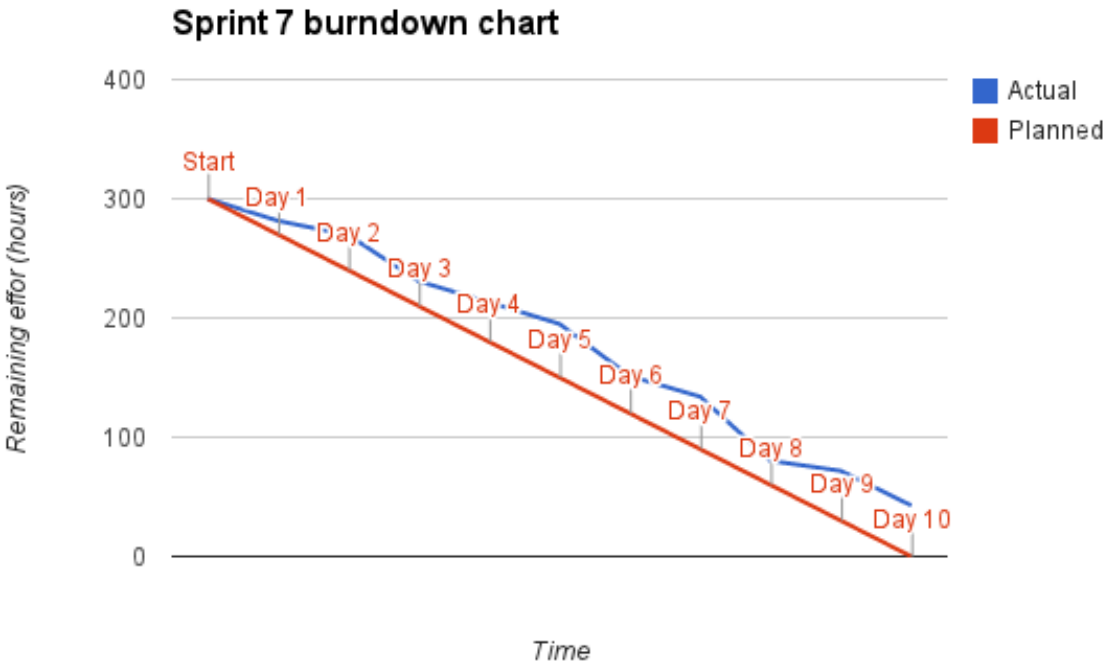


Figure G.7: Burn down chart for sprint 7

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x	planlegging	12	0	12	0	0	12	0	0	0	0
x	møter	0	0	0	0	6	0	0	0	0	0
104	Refactor and optimizing	0	0	0	0	0	0	0	0	0	0
106	Bug fixing and testing	0	0	5	0	4.5	0	0	8	0	0
114	Profile tab	0	0	0	1.25		3	0	0	0	0
	hide share function when not logged on to FB	0	0	0	1.25	0	0	0	0	0	0
	Weeks bugged in the line graph	0	0	0	0	0	2	0	0	0	0
	Line graph start at wrong place	0	0	0	0	0	1	0	0	0	0
129	Final version of report	0.75	3.75	11.75	7.75	2.25	24.5	1.5	12.75	7	25.25
	Technical documentation	0	0	0	0	0	0.75	0	0	0	0
	revise report	0	0	0	0	0	0	0	0.5	1	4.5
	User Manual - write overview of the usagetab	0	0	0.5	0	0	0	0	0	0	0
	rewrite use cases	0	0	0	0	1	0	0	0	0	0
	burndown charts	0	2.25	0	0	1	0	0	0.5	0	0
	User manual outline	0	0	1	0	0	0	0	0	1.5	0
	Sprint - summary	0	0	0.75	0	0	0	0	0	0	0
	Sprint - issues and solutions	0	0	0.5	0	0	0	0	0	0	1.5
	Sprint - work done	0	0	0.75	0	0	0.5	0	0	3	0
	Sprint - review	0	0	0.75	0	0	0.5	0	0	0	0
	add screenshot of existing solutions	0.75	1.5	0	0	0	0	0	0	0	2.75
	Fix sharelatex	0	0	0.5	0	0	0	0	0	0	0
	add further development outline	0	0	0	0	0	2	0	0	0	2.25
	Read through the report and add/fix todos	0	0	0	0	0	3	0	0	0	0
	Write status report for supervisor	0	0	1	0	0	0	0	0	0	0
	Restructure sprints	0	0	0	0	0	0	0	4	0	0
	backlog reformatting	0	0	6	6.5	0	10.75	0	0	0.5	5.25
	Discuss restructure of report	0	0	0	1.25	0.25	3.5	1.5	0	0	0
	design in development process	0	0	0	0	0	0	0	2.75	0	4.75

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Write domain knowledge	0	0	0	0	1	0	0	0	0	0
	Report cleanup	0	0	0	0	0	0.5	0	0	1	0
	Fix small tasks after meeting	0	0	0	0	1	1	0	0	0	0
	Move most influential risks to retrospect	0	0	0	0	0	2	0	0	0	0
	report restructure	0	0	0	0	0	0	0	4.25	0	0
	Update about customer	0	0	0	0	0	0	0	0.25	0	0.25
	Update milestones	0	0	0	0	0	0	0	0.25	0	0
	Development envrionment intro + dropwizard contents	0	0	0	0	0	0	0	0.25	0	0
	add scrum illustration	0	0	0	0	0	0	0	0.25	0	0.75
	Add intro to 3.2	0	0	0	0	0	0	0	0	0	0.5
	Write sections in further development	0	0	0	0	0	0	0	0	0	2.25
	Read through the report conventions on github	0	0	0	0	0	0	0	0	0	0.5
130	Concept social tab	2.25	0	4	1.75	0	1.5	1	6.75	0	0
	add "back" button in compare to others	0	0	0	0	0	1	1	3.75	0	0
	Add actual FB data. Picture and friends list.	0	0	4	1.75	0	0	0	0	0	0
	add comparison settings to hamburger menu(filter)	2.25	0	0	0	0	0	0	0	0	0
	"Filter" menu-item multiplies	0	0	0	0	0	0.5	0	3	0	0
131	Branding	0	0	0	0	2.25	0	0	1.25	1.5	0
	Make logo	0	0	0	0	1	0	0	0	1.5	0
	Create icons	0	0	0	0	1.25	0	0	1.25	0	0
133	Concept device tab	3	6.25	4	5	0	3	5	9	0	0
	Work on the design	3	4	1	1	0	1	4	2	0	0
	Category spinner selects first category when editing	0	0	1	0	0	0	0	0	0	0
	Add delete option to devices	0	0	2	0	0	0	0	0	0	0
	Fix android device	0	1.25	0	0	0	1	1	3	0	0
	Change name of edit dialog	0	1	0	0	0	0	0	0	0	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Implement number of devices indicator	0	0	0	2	0	0	0	0	0	0
	Update device tab to support permament total usage instance	0	0	0	1	0	0	0	0	0	0
	Add toasts for events	0	0	0	1	0	0	0	0	0	0
	Long clicking category craches the app	0	0	0	0	0	1	0	4	0	0
141	Complete design of tabs	0	0	0	0	1.75	0	0	0	0	0
	Revise design consistency	0	0	0	0	1.75	0	0	0	0	0
142	Language support	0	0	0	1	0	0	2	1	0	0
	Translate usage changes	0	0	0	0	0	0	1	0	0	0
	Translate device tab changes	0	0	0	1	0	0	0	0	0	0
	Fix missing translations	0	0	0	0	0	0	1	1	0	0
145	General improvements	0	2	2	0	0	0.75	4	7.75	0	3.25
	Change the spinner text/ style	0	0	0	0	0	0.5	0	0	0	0
	Update device user id when user logs on to FB 1. time	0	0	0	0	0	0.25	0	0	0	0
	input field for kWh cant even hold 2 digits.	0	0	0	0	0	0	0	1	0	0
	when keyboard opens add usage button looks bad	0	0	0	0	0	0	0	0	0	1
	Update look on the homepage	0	0	0	0	0	0	0	4	0	2.25
	Server configuration	0	0	0	0	0	0	0	2.75	0	0
	General optimizations	0	2	2	0	0	0	4	0	0	0
146	Administration	0	0	0	0	0	0.75	0	0	0	0
	Add tasks from friday to yodiz	0	0	0	0	0	0.75	0	0	0	0
148	Functional home page	0	0	0	0	0	0	2.25	7	0	0
	Add support for wall posts on server	0	0	0	0	0	0	0	4	0	0
	Add support for wall posts on client	0	0	0	0	0	0	2.25	3	0	0
	Sum	18.5	12	38.75	17.5	18.25	43.75	17.25	53.75	8.5	29

Table 1: Sprint 7 backlog

G.8 Sprint 8

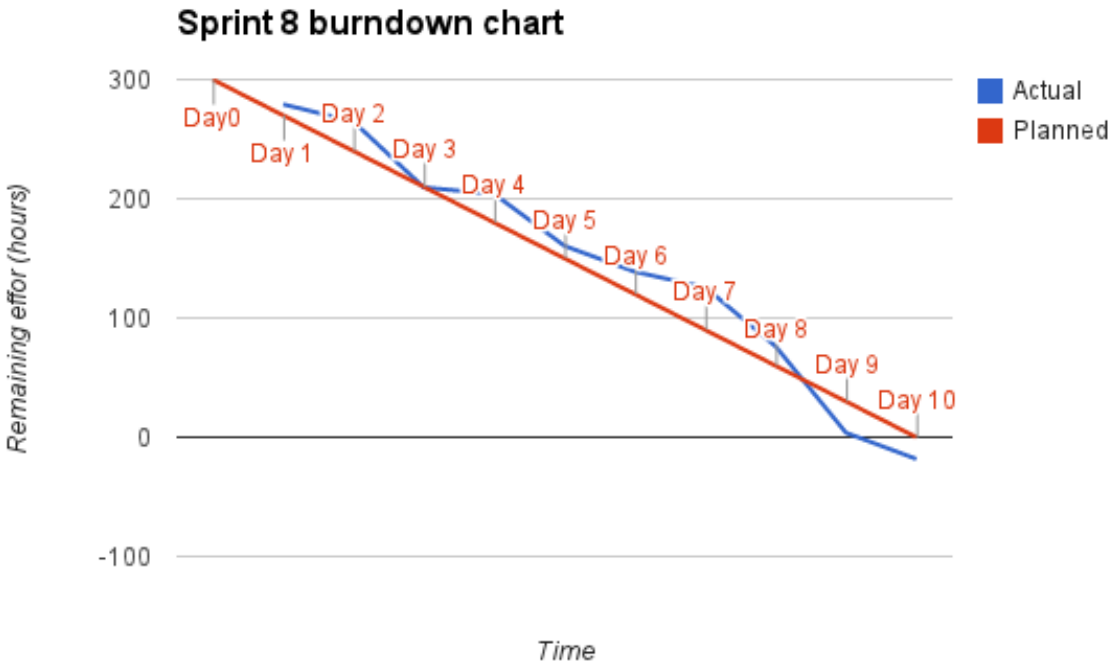


Figure G.8: Burn down chart for sprint 8

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x	planning	12	0	12	0	0	12	0	0	0	0
x	meetings	0	0	0	0	6	0	0	0	0	0
154	customer presentation	0	0	0	0	0	0	0	0	0	0
	present	0	0	0	0	9	0	0	0	0	0
152	Code quality and cleaning	0	0	0	0	0	0	0	0	0	0
	Cleanup Tips	0	0	2	0	0	0	0	0	0	4
	Cleanup server	0	0	2	0	4	0	0	0	0	0
	Going over large parts of the code to make usure its consistent.	0	0	0	0	0	0	0	6	0	0
	Cleanup in app (code conventions)	4	0	3	0	0	0	0	0	4	0
147	Branding	0	0	0	0	0	0	0	0	0	0
	Branding	0.75	0	0	0	0	0	0	0	0	0
153	Last minute functionality	0	0	0	0	0	0	0	0	0	0
149	Final version of report	0	0	0	0	0	0	0	0	0	0
	Technical documentation (installation guide)should be written	0	2	0	0	0	0	0	0	0	0
	rewrite the use cases	0	0	0	0.5	0	0	0	0	0	0
	Add usecase diagram and write about it	0	0	0	0	0.75	0	0	0	0	0
	Write User Manual	0	2	0	0	0.75	0	0	0	0	0
	Ch. 5: Fix design of class diagrams	0	0	0	0	0	1.75	1.75	0	0	0
	Correct the referance from retrospectchapter to the risktable	0	0	0.25	0	0	0	0	0	0	0
	Modify architecture chapter. Ref. mail 13.04.2014	0	0	2.75	0	0	0	0	0	0	0
	backlog sprint 6	2	0	0	0	0	0	0	0	0	0
	Write abstract	0	0	0.75	0	0	0	0	0	0	0
	update meeting overview with supervisor and customer to include latest dates	0	0	0	0	0.5	0	0	0	0	0
	backlog sprint 7	0	0	2	0	0	0	0	0	0	0
	revise wbsa	0	0.25	0	0	0	0	0	0	0	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Ch.5: Reformulate and revise system architecture chapter	0	0	0	0	2.25	0	0	0	0	0
	6.3 : Write our adaption to scrum	0	0	0	1	0	0	0	0	0	0
	7.1.6 : Revise testing non functional requirements	0	0	0	0	0	0	0	4	0	0
	7.2.1 : Write hallwaytesting section	0	0	0	0	0	0	0	1	0	0
	Prestudy : Write about libraries not written about	0	0	0	0	0	0	0	2	0	0
	Rewrite non functonal requirements	0	0	1.25	0	0	0	0	0	0	0
	Write about acceptance testing	0	0	0.25	0	0	0	0	0	0	0
	fix todo and title page	0	0	1.75	0	0	0	0	0	0	0
	update wbs	0	0	1	0	0	0	0	0	0	0
	Write our adaption to scrum	0	0	3	0	0	0	0	0	0	0
	Write about the patterns we use and revise chapter 5	0	0	0	0	0	2	2	0	0	0
	6.3 : Add list of goals in each sprint in process development sprint overview	0	0	0	0	1	0	0	0	0	0
	Write about sprint 7 in development process	0	0	0.5	0	0	0	0	0	0	0
	revise domain knowledge	0	0	0	1	0	0	0	0	0	0
	Insert backlog 7 into report	0	0.25	0	0	0	0	0	0	0	0
	restructure report and add chapters/sections according to discussion	0	0	0	0	1	0	0	0	0	0
	Create pictures to user manual and research how to place pictures in latex	0	0	0	0	2	0	0	0	0	0
	Write about how to actually install the application on your phone	0	0	0	0	0.25	0	0	0	0	0
	write resource allocation	0	0	0	0	1	0	0	0	0	0
	Write further development	0	0	0	0	1.5	0	0	0	0	0
	8.5 - product evaluation	0	0	0	0	1	0	0	0	0	0

Continued on next page

Table 1 – continued from previous page

ID	Description	Remaining effort in sprint (days)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Revise further development	0	0	0	0	0	2.5	0	1	0	0
	Change elicitation of requirements to include documentation and picture of steps	0	0	0	0	0	0	0	0.75	0	0
	Fix usermanual	0	0	0	0	0	0	0	1	0	0
	Revise chapter 3	0	0	0	0	0	0	0	3	0	0
	Revise chapter 1	0	0	0	0	0	0	0	0	2.25	0
	Revise chapter 2	0	0	0	0	0	0	0	0	2.25	0
	Revise chapter 3	0	0	0	0	0	0	0	0	2.5	0
	Revise report	0	2	0	0.5	2.75	0	2.75	6.5	13	2
	Write documentation	0	0	0	2.5	0	0	0	0	0	0
	Revise report	0	2	0	0	1	0	2	6	14.5	0
	Revise report	0	2	0	0	0	0	0	10	12	10
	Finalize retrospect	0	0	0	0	0	0	0	0	0	0
	Revise report	2	2	3.25	0	5	3.5	4.5	1.75	12	6

Table 1: Sprint 8 backlog

HomeMatic

The information about HomeMatic plug that was provided by the customer is found on the next page.

Schaltaktoren

Funk-Schaltaktor 1-fach mit Leistungsmessung, Zwischenstecker

Artikel-Nr.: 130248

PRODUKTEIGENSCHAFTEN

- Der HomeMatic Funk-Schaltaktor mit Leistungsmessung verbindet zwei Funktionsbereiche in einem Gerät:
    1. Schalten von angeschlossenen Verbrauchern (Schaltkanal)
    2. Messen von Spannung, Strom, Wirkleistung, Frequenz und Energieverbrauch (Messkanal)
- Über den Schaltkanal können angeschlossene Verbraucher oder angelernte HomeMatic Aktoren ein- bzw. ausgeschaltet werden
- Der Messkanal verfügt über eine Messfunktion und Empfangs- sowie Übertragungsmöglichkeit von Messdaten (z. B. Spannung, Strom, Wirkleistung, Frequenz und Energieverbrauch bis 3680 Watt/16 A) an die HomeMatic Zentrale
- Zyklische Übertragung der Messwerte an die HomeMatic Zentrale
- Grafische Anzeige der Messdaten in der Bedienoberfläche der CCU2
- Über die Funktion „Bedingtes Schalten“ können angeschlossene Verbraucher oder andere HomeMatic Geräte autark geschaltet werden, wenn ein individuell definierter Messwert erreicht wird (z.B. das Über- oder Unterschreiten von Grenzwerten)
- Robuste Ausführung: 50.000 Schaltzyklen bei cosφ=1

TECHNISCHE DATEN

Versorgungsspannung:	230 V/50 Hz
Stromaufnahme:	16 A max.
Leistungsaufnahme Ruhebetrieb:	< 0,6 W
Typ. Funk-Freifeldreichweite:	> 150 m
Funkfrequenz:	868,3 MHz
Empfängerklasse:	SRD Class 2
Schutzart:	IP20
Messkategorie:	CAT II
Max. Schaltleistung:	3680 W (ohmsche Last)
Relais:	Schließer
Abmessungen (B x H x T):	59 x 123 x 40 mm (ohne Netzstecker)
Gewicht:	165 g

LOGISTISCHE DATEN

Artikelnummer:	130248
EAN-Code:	4047976302482
Kurzbezeichnung:	HM-ES-PMSw1-Pl
Verpackungseinheit:	16
Maße Verpackung:	150 x 75 x 90 mm
Gesamtgewicht:	244,5 g

LIEFERUMFANG

Funk-Schaltaktor
Bedienungsanleitung in D und GB

Technische Änderungen vorbehalten.

I User test example

Remote usability test

Test 1 : The basics

Tasks	1. Log in to facebook 2. View your profile 3. Add a new residence 4. Log out
Question 1	What did you find complicating about the tasks given above?
Answer	Have not used android before, and I found that complicating.
Question 2	Which of the above tasks did you find intuitive and easy to perform?
Answer	It was easy to log in. There should be a link to login from profile.
Question 3	What features do you think should be added to make the tasks above easier?
Answer	There should be a login link in the profile tab.

Test 2 : Devices

Tasks	1. Add a new device 2. Add usage to that new device ← did theese two in the last one
Question 1	What did you find complicating about the tasks given above?
Answer	The device changed categories, and that was confusing. It should be possible to delete devices.
Question 2	Which of the above tasks did you find intuitive and easy to perform?
Answer	No comment
Question 3	What features do you think should be added to make the tasks above easier?
Answer	No comment

Test 3: Usage

Tasks	1. Check the usage of the device named “Heater” 2. Compare the usage of the Radio and the TV in the Pie chart
Question 1	What did you find complicating about the tasks given above?
Answer	Went to add usage instead of add device. Should show an error message when adding usage without any device. Should have more shortcuts between the tabs. It’s hard to tell if you’re changing an earlie entry or creating a new one.The pie chart should show info without the user having to click them first.
Question 2	Which of the above tasks did you find intuitive and easy to perform?
Answer	No comment
Question 3	What features do you think should be added to make the tasks above easier?
Answer	No comment

Test 4 : Comparison

Tasks	1. Compare yourself to a friend 2. Compare yourself to other similar profiles
Question 1	What did you find complicating about the tasks given above?
Answer	I didn't really understand anything of what was going on. I didn't really understand what similar profiles was supposed to do. I makes sense that this should compare you to everyone else.
Question 2	Which of the above tasks did you find intuitive and easy to perform?
Answer	No comment
Question 3	What features do you think should be added to make the tasks above easier?
Answer	There should be a name under the pictures, in case two people don't have a profile picture.

Test 5: Tips

Tasks	1. Add a tip to your list of tips 2. View your list of tips 3. Mark a tip
Question 1	What did you find complicating about the tasks given above?
Answer	This is very confusing and unintuitive. It should be possible to directly share tips with other users.
Question 2	Which of the above tasks did you find intuitive and easy to perform?
Answer	No comment
Question 3	What features do you think should be added to make the tasks above easier?
Answer	Call the tab "Your tip collection"

Overall:
The app should give feedback whenever a user does somthing. Give notifications every time the app does something. My overall impression is that work is not being done towards a common goal.