

# BACHELOR'S THESIS

Development of MakerSpace Management System  
Group BO17-G14

Thomas Bergby  
Simon Chen Dybvik  
Nicolai Naglestad  
Espen Ottar Skjeggestad

16/05/2017

Computer Science / Digital Media Production / Information Systems  
Faculty of Computer Science







# HØGSKOLEN I ØSTFOLD

Avdeling for Informasjonsteknologi  
Remmen  
1757 Halden  
Telefon: 69 21 50 00  
URL: [www.hiof.no](http://www.hiof.no)

## BACHELOROPPGAVE

Prosjektkategori: <b>Bacheloroppgave</b>	<input checked="" type="checkbox"/>	Fritt tilgjengelig
Omfang i studiepoeng: <b>20</b>	<input checked="" type="checkbox"/>	Fritt tilgjengelig etter (16/05 2017)
Fagområde: <b>Informasjonsteknologi</b>	<input type="checkbox"/>	Tilgjengelig etter avtale med oppdragsgiver

Tittel: <b>Utvikling av MakerSpace Management System</b>	Dato: <b>May 16, 2017</b>
Forfatterere: <b>Thomas Magelssen Bergby, Simon Chen Dybvik, Nicolai Naglestad, Espen Ottar Skjeggstad</b>	Veileder: <b>Børre Stenseth</b>
Avdeling / Program: <b>Avdeling for Informasjonsteknologi (alle programmer)</b>	Gruppenummer: <b>BO17-G14</b>
Oppdragsgiver: <b>HiØ/IT (MakerSpace)</b>	Kontaktperson hos oppdragsgiver: <b>Espen Teigen</b>

### Ekstrakt:

This report describes the development of an inventory system for the Østfold University College's MakerSpace. This project is to give the MakerSpace a tool to manage its ever growing inventory. The system will be open source and it will be made available for all Makerspaces that wish to use it. The development process focused on using the incremental method to achieve this goal. Throughout the development process the group has used technologies and programs that are unfamiliar to them. This was both to learn about these new technologies and to gain experience using systems that are becoming more and more popular.

3 emneord:

REST API / NOSQL
Inventar
Maker Movement

# **Abstract**

This report describes the development of an inventory system for the Østfold University College's MakerSpace. This project is to give the MakerSpace a tool to manage its ever growing inventory. The system will be open source and it will be made available for all Makerspaces that wish to use it. The development process focused on using the incremental method to achieve this goal.

Throughout the development process the group has used technologies and programs that are unfamiliar to them. This was both to learn about these new technologies and to gain experience using systems that are becoming more and more popular.

# Acknowledgements

First and foremost, we would like to express our gratitude to our mentor, Børre Stenseth, for valuable feedback, advice and guidance throughout the project.

We would also like to express our gratitude to our project owner, Espen Teigen, for ideas during all stages of the project.

Finally, we want to thank Gunnar Misund, Evelien Jacobs and all of our testers.

Last but not least we would also like to thank the open source community.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>List of Figures</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Group . . . . .	2
1.1.1 Thomas Magelssen Bergby . . . . .	2
1.1.2 Simon Chen Dybvik . . . . .	2
1.1.3 Nicolai Naglestad . . . . .	3
1.1.4 Espen Ottar Skjeggstad . . . . .	3
1.2 Employer . . . . .	4
1.3 Task . . . . .	4
1.3.1 Purpose . . . . .	4
1.3.2 Project delivery / Prototype . . . . .	5
1.3.3 Documentation . . . . .	5
1.3.4 Method . . . . .	5
1.4 Report structure . . . . .	6
1.4.1 Introduction . . . . .	6
1.4.2 Glossary . . . . .	6
1.4.3 Analysis . . . . .	6
1.4.4 Design . . . . .	6
1.4.5 Implementation . . . . .	7
1.4.6 Testing . . . . .	7
1.4.7 Discussion and Evaluation . . . . .	7
1.4.8 Conclusion . . . . .	7
1.5 Project Plan . . . . .	8
1.5.1 Main Deliveries . . . . .	8
1.5.2 Milestones . . . . .	8

<b>2</b>	<b>Glossary</b>	<b>9</b>
2.1	MakerSpace . . . . .	10
2.2	Program tools . . . . .	10
2.2.1	MongoDB . . . . .	10
2.2.2	Node.js . . . . .	10
2.2.3	CRUD . . . . .	11
2.2.4	Postman . . . . .	11
2.3	Collaboration tools . . . . .	12
2.3.1	Git and GitHub . . . . .	12
2.3.2	Trello . . . . .	12
2.3.3	L <sup>A</sup> T <sub>E</sub> X . . . . .	13
<b>3</b>	<b>Analysis</b>	<b>14</b>
3.1	The task . . . . .	15
3.2	Open Source . . . . .	15
3.2.1	MIT License . . . . .	15
3.2.2	MakerSpace Management System MIT License . .	15
3.2.3	Security . . . . .	16
3.2.4	Cost . . . . .	16
3.2.5	Support and maintainability . . . . .	17
3.3	The Security Aspect . . . . .	17
3.3.1	Using existing systems (HTTP Auth, OAuth2, Etc.) .	17
3.3.2	Custom system . . . . .	17
3.3.3	Internet vs Intranet . . . . .	18
3.4	The Technology . . . . .	18
3.4.1	Front-end . . . . .	18
3.4.1.1	JavaScript / jQuery . . . . .	18
3.4.1.2	HTML5 / CSS3 . . . . .	19
3.4.1.3	PHP7 . . . . .	19
3.4.2	Back-end . . . . .	19
3.4.2.1	MongoDB . . . . .	19
3.4.2.2	Node.js / Mongoose . . . . .	20
3.4.2.3	REST API . . . . .	20
3.4.2.4	Linux Apache . . . . .	21
3.5	Users . . . . .	22
3.5.1	User Stories . . . . .	22
<b>4</b>	<b>Design</b>	<b>24</b>
4.1	What is already out there? . . . . .	25

4.2	Site layout . . . . .	25
4.3	MongoDB Models . . . . .	28
4.3.1	Items . . . . .	29
4.3.1.1	Node.js Model . . . . .	29
4.3.1.2	JSON Model . . . . .	30
4.3.2	Category . . . . .	31
4.3.2.1	Node.js Model . . . . .	31
4.3.2.2	JSON Model . . . . .	31
4.3.3	Tag . . . . .	32
4.3.3.1	Node.js Model . . . . .	32
4.3.3.2	JSON Model . . . . .	32
4.3.4	Location/Locale . . . . .	33
4.3.4.1	Node.js Model . . . . .	33
4.3.4.2	JSON Model . . . . .	33
<b>5</b>	<b>Implementation</b>	<b>34</b>
5.1	Flowcharts . . . . .	35
5.2	First iteration . . . . .	37
5.2.1	Website (Front-end) . . . . .	37
5.2.1.1	Design . . . . .	37
5.2.1.2	Technology . . . . .	38
5.2.2	Node.js API (Back-end) . . . . .	38
5.3	Second iteration . . . . .	39
5.3.1	Website (Front-end) . . . . .	39
5.3.2	Back-end . . . . .	40
5.4	Third iteration . . . . .	40
5.4.1	Website (Front-end) . . . . .	40
5.4.1.1	General changes . . . . .	40
5.4.1.2	Search field . . . . .	41
5.4.1.3	Item page . . . . .	41
5.4.1.4	Administrator page . . . . .	43
5.4.2	Back-end . . . . .	43
5.5	Fourth iteration . . . . .	44
5.5.1	Website (Front-end) . . . . .	44
5.5.1.1	General changes . . . . .	44
5.5.1.2	Item page . . . . .	45
5.5.1.3	Administrator page . . . . .	46
5.5.2	Back-end . . . . .	47



<b>6</b>	<b>Testing</b>	<b>48</b>
6.1	Why do a user test . . . . .	49
6.2	Test goal . . . . .	49
6.3	Methods . . . . .	49
6.3.1	Usability Testing . . . . .	49
6.4	Target Audience . . . . .	49
6.4.1	User Types . . . . .	50
6.4.1.1	Student . . . . .	50
6.4.1.2	Student Assistant . . . . .	50
6.4.1.3	Employee / Admin of MakerSpace . . . . .	50
6.5	Test Execution . . . . .	51
6.5.1	Roles . . . . .	51
6.5.1.1	User . . . . .	51
6.5.1.2	Test Leader . . . . .	51
6.5.2	Setup . . . . .	51
6.5.3	Tasks . . . . .	51
6.5.4	After interview? . . . . .	52
6.6	Results . . . . .	52
6.6.1	Interview . . . . .	52
<b>7</b>	<b>Discussion and Evaluation</b>	<b>54</b>
7.1	Design . . . . .	55
7.2	Implementation . . . . .	55
7.3	Usability Test . . . . .	56
7.4	The Project Method . . . . .	56
7.5	Future . . . . .	56
7.5.1	Front-end . . . . .	56
7.5.1.1	General Design . . . . .	56
7.5.1.2	Design Item page . . . . .	57
7.5.1.3	Implement new functions . . . . .	57
7.5.2	Back-end . . . . .	57
<b>8</b>	<b>Conclusion</b>	<b>58</b>
<b>9</b>	<b>Bibliography</b>	<b>59</b>
<b>A</b>	<b>Raw Test Results</b>	<b>62</b>
<b>B</b>	<b>API Calls and Output</b>	<b>72</b>
B.1	Getting all items . . . . .	73

---

B.2	Get a single item . . . . .	75
B.3	Updating a item . . . . .	76
B.4	Deleting a item . . . . .	77
B.5	Creating a item . . . . .	78
<b>C</b>	<b>Project Contract</b>	<b>80</b>
<b>D</b>	<b>Group Contract</b>	<b>84</b>
<b>E</b>	<b>Confirmation of Group Change</b>	<b>88</b>
<b>F</b>	<b>Meeting notes</b>	<b>90</b>
F.1	Meeting 24-1-17 . . . . .	91
F.2	Meeting 31-1-17 . . . . .	92
F.3	Meeting 1-2-17 . . . . .	93
F.4	Meeting 7-2-17 . . . . .	94
F.5	Meeting 14-2-17 . . . . .	95
F.6	Meeting 21-2-17 . . . . .	96
F.7	Meeting 28-2-17 . . . . .	97
F.8	Meeting 28-2-17 . . . . .	98
F.9	Meeting 14-3-17 . . . . .	99
F.10	Meeting 21-3-17 . . . . .	100
F.11	Meeting 4-4-17 . . . . .	101
F.12	Meeting 26-4-17 . . . . .	102

# List of Figures

2.1	Screenshot of Postman interface. . . . .	11
2.2	Trello bord . . . . .	12
2.3	Graph about document complexity. . . . .	13
4.1	Wireframe for the landing page . . . . .	26
4.2	Wireframe for the item page . . . . .	27
5.1	Flowchart for adding an item . . . . .	35
5.2	Flowchart for searching for an item . . . . .	36
5.3	Screenshot of the landing page English (first iteration) . .	37
5.4	Screenshot of the landing page Norwegian (first iteration)	38
5.5	Screenshot of the landing page (second iteration) . . . . .	39
5.6	Screenshot of the landing page (third iteration) . . . . .	40
5.7	Screenshot of the landing page with search (third iteration)	41
5.8	Screenshot of the item page of a specific item (third iteration) . . . . .	42
5.9	Screenshot of the administrator page (third iteration) . . .	43
5.10	Screenshot of the landing page with categories (fourth iteration) . . . . .	44
5.11	Screenshot of the item page (fourth iteration) . . . . .	45
5.12	Screenshot of the item page as a user (fourth iteration) . .	46
5.13	Screenshot of the administrator page (fourth iteration) . .	46
D.1	Additional signature from new member . . . . .	87

# **Chapter 1**

## **Introduction**

This chapter gives an introduction of the group, employer, the task and how the group plans to execute the task.

## 1.1 The Group

The group consist of 4 third year students. Initially the group consisted of 3 members, but a 4th member joined later in the project. (See appendix E) The group is divided into 3 different studyfields 1 Informatics, 1 Digital Media Production and 2 Information Systems. Most of the group has worked with each other on multiple occasions, both in student organizations or projects. 3 members of the group also live together in student collectives. All members have a high interest in the project and MakerSpace, where all of them spend a lot of time.

### 1.1.1 Thomas Magelssen Bergby

A student who has been interested in everything regarding IT and technology since he was a kid. Thomas has been a leader for “Lær Kidsa Koding” (A group of students who teach coding for kids) and a leader for student assistants in Web-Development and JavaScript courses.

Thomas studies Informatics, and has gained skills within JavaScript, Java, PHP, CSS, Linux and Python. During his studies, he has taken courses such as Algorithms and Data-Structures, Software Engineering, Object Oriented Programming and Android Programming.

He also enjoys traveling, and hopes to eventually get a job in the United Kingdom or the USA.

### 1.1.2 Simon Chen Dybvik

Simon has been interested in technology his whole life. As a curious child, he often disassembled products to see what’s inside and how it worked. He is broadly interested in Apple and their products. During his studies at Østfold University College he has done an exchange semester abroad to California State University, Monterey Bay, where he focused on web development using CMS, JavaScript, jQuery, HTML and CSS, and graphic design. He is the former vice-president of NITO Studentene Halden which is a union for engineers and technologists.

Simon studies Information Systems with emphasis on web development. He has taken courses like Project Management, Marketing, Business Economics and Graphic Design.

Simon joined the group at a later stage than the rest, see appendix E.

### **1.1.3 Nicolai Naglestad**

Studied International Baccalaureate at Skagerak International School in Sandefjord. Nicolai has a great interest in technology and is always looking for something new to learn. Beside his studies, he works as a student assistant in the subjects Introduction to Programming, Web Development, Object-oriented Programming. He also works at the schools MakerSpace where the latter is a position where he helps students get started on projects and with the use of the 3D printers. Nicolai has a great variety of interests with most aspects the are to be found inside the MakerSpace and you will find him there most of the time.

Nicolai studies Digital Media Production at Østfold University College, but has taken subjects such as OOP, Software Engineering and .NET. He also enjoys learning new systems and languages.

### **1.1.4 Espen Ottar Skjeggstad**

He has a broad field of interest, but the main ones are IT and Biology. He is an active person that likes jogging, training, diving and trips. He is also politically active in the student politics and has roles as elected representative for the class, member of the student counsel and member of the executive committee for the student-democracy. He is currently a student substitute member of the University College Board. He is student assistant for GRIT and is now working at the school library.

Espen studies Information Systems with focus on IT and code, but also includes business leadership and classes on economy.

## 1.2 Employer

The employer for this project is MakerSpace (MS) which is a room located in Østfold University College (HiØ). The MakerSpace is a playroom for creating all types of technology, everything from electronics and robotics, to programming and 3D-printing. The room is currently funded and managed by the IT department.

Students and lecturers can use the rooms equipment to experiment with technology to further educate themselves within topics they find interesting, and that are not necessarily related to any ongoing subjects at the University College. MakerSpace is open to all students and staff of the school, but is mainly used by the IT department.

The employers for this project are Staff Engineer Espen Teigen and University College Teacher Michael Andersen Lundsveen.

## 1.3 Task

The goal of this project is to develop an inventory- and loan-system for Østfold University College's MakerSpace. The purpose of this is to make it easier for employees at MakerSpace to keep track of inventory at all times. A full inventory system will help both students and staff to find equipment when a student assistant or department Engineer is not available. The system should preferably be able to know where equipment is located in MakerSpace at any given time.

### 1.3.1 Purpose

The purpose of this system is to make maintaining MakerSpace easier for all parties, but mainly for the employees of MakerSpace. This means that less time is used to maintain inventory and less time is used to help users find different equipment. This benefits the school by saving time and money, as the student assistants do not need to be consulted as often. They currently help with mundane tasks such as finding equipment, counting inventory, and deciding which items that need to be ordered.

### 1.3.2 Project delivery / Prototype

The group aims to supply the employer with a website (front-end) and server (back-end) that is user tested to the employers and users specification.

The website will support the following features:

- View all items (Name, Location, Description, Amount in stock)
- Create/Update/Delete items (CRUD)
- Register/Modify/Delete/View users
- User registration either by custom system or by OAuth 2.0

Additionally, there will be a REST API based on Node.js and MongoDB to provide a system for storing the information for the website and possible future applications or other systems.

### 1.3.3 Documentation

Each separate prototype/system will also include full documentation on how the system is to be used and in the case of the REST API, how it can be used in other systems. This documentation will be hosted on the same location as where the code is stored (GitHub). As with our main project page, the document will be a web page generated by Jekyll hosted by GitHub Pages.

### 1.3.4 Method

The group will be using the incremental method for the development of the system. This method focuses on development piece by piece, and works well for modular systems. It also works for quantitative and qualitative testing of the parts that are finished. These parts can also be used, and delivered to the employer.

The incremental method works by finishing one piece of the system at a time. E.g The database-system is made first, and finished. Afterwards, the next part can be worked on.

This method has a lower risk of total failure and no delivery, because it is made up by separately working pieces.



## **1.4 Report structure**

The report will be structured in the following way

1. Introduction
2. Glossary
3. Analysis
4. Design
5. Implementation
6. Testing
7. Discussion and Evaluation
8. Conclusion

### **1.4.1 Introduction**

This chapter gives an introduction of the group, employer, the task and how the group plans to execute the task.

### **1.4.2 Glossary**

This chapter describes the tools and terminology that are essential for the project.

### **1.4.3 Analysis**

This chapter show the groups analysis of the task, tools, security, cost, technology and analyses the use of the system.

### **1.4.4 Design**

This is where the design of the system is discussed. The design process, implementations and decisions during the project are addressed here.

### **1.4.5 Implementation**

The group used iterations as a development method, and the different iterations are listed with details below. The back-end is developed all throughout the project, but mostly after the front-end specifications. If it was needed for functions or support for the front-end, it got implemented in the back-end as well. Because of this, there will not be a lot of discussions about how the different iterations changed the back-end.

### **1.4.6 Testing**

This chapter goes through the test process from start to execution. What tasks the test persons had and results from the open interview at the end of the test.

### **1.4.7 Discussion and Evaluation**

This chapter will discuss design, implementation, testing, method and possible future work elements.

### **1.4.8 Conclusion**

In this chapter the group will determine the conclusion on the completion of the project.

## 1.5 Project Plan

The group is required to follow a set of deliveries during the project period.

### 1.5.1 Main Deliveries

**D1 January 18th** - Preliminary Report

**D2 March 9th** - 1st Version of the Main Report

**D3 April 20th** - 2nd Version of the Main Report

**D4 May 16th** - Final Report and Source Code of the System

**D5 May 29th** - Hang up Poster

**D6 June 1st** - Presentation

### 1.5.2 Milestones

**M1** - Alpha of the website, front end

**M2** - Making the website talk to the server, back end

**M3** - Adding items to server through the website

**M4** - Final delivery

**M5** - Presentation

# **Chapter 2**

## **Glossary**

This chapter describes the tools and terminology that are essential for the project.

## 2.1 MakerSpace

MakerSpace is mainly manned by approximately three student assistants. It has a large room with usually a lot of people visiting throughout the day. Many of the students visiting make do by themselves, but occasionally need student assistants to find something or get advice on how to do a project. The student assistants mainly work with making sure the MakerSpace room is in order. They also provide courses in relevant subjects for MakerSpace like 3D printing and drones.

## 2.2 Program tools

### 2.2.1 MongoDB

It was chosen to use MongoDB as a database system because the group mainly wanted to learn how to use a NOSQL database system. MongoDB type databases, have some strengths that make it easier to be used when developing a system. MongoDB themselves have a good overview showing the differences of SQL and NOSQL [1] that can both be helpful to build the database easier and experience can be gained with this type of system as this might become very relevant for business in the future.

### 2.2.2 Node.js

Like with the database system, it is mostly chosen due to the want to learn and experiment with technologies that have not been used yet in subjects at HiØ previously. The group also looked around at what others had been using to make a REST API and why they used that specific architecture.

It was quickly found that most recommended to use Node.js. This was because it is well documented and is very easy to setup and test (requiring only the server itself and a database server to connect to) other architectures require specific software from the server to run. (As an example a REST API based on Spring/Java requires a Gradle or Apache Maven server to run)

### 2.2.3 CRUD

CRUD is a acronym for **C**reate, **R**ead, **U**ppdate, and **D**eleate. The group refers to this as functionality it wants in the REST API when it comes to communication with the database. Every model used in the database will have to follow CRUD. For example it has to be possible to create items in the database as well as update and delete them. CRUD for the most part refers to the communication between a database server and a the database consumer (in this case the API).

### 2.2.4 Postman

Postman is a tool for testing and debugging REST APIs and is used extensively throughout the development of the API. It can emulate and run calls to the API as if it was a client (being a website or a mobile device). The program is free and available on all OS platforms as well as being a Chrome extention.

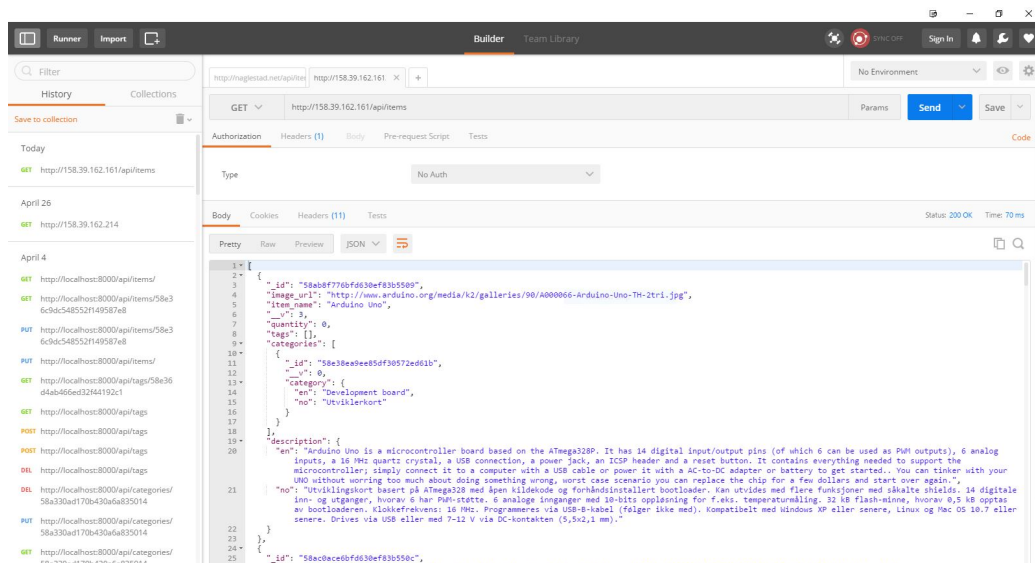


Figure 2.1: Screenshot of Postman interface.

## 2.3 Collaboration tools

### 2.3.1 Git and GitHub

The group uses Git and GitHub for version control, both for the report and the development.

The repositories on GitHub are public for everyone to see and contribute to, since this is a MakerSpace project. All the development is open source, and GitHub is a good platform for this exact purpose.

### 2.3.2 Trello

Trello is a web based Kanban board. Kanban is a method for managing knowledge-work by the use of a post-it like form. This gives a visual rendering of the processes.

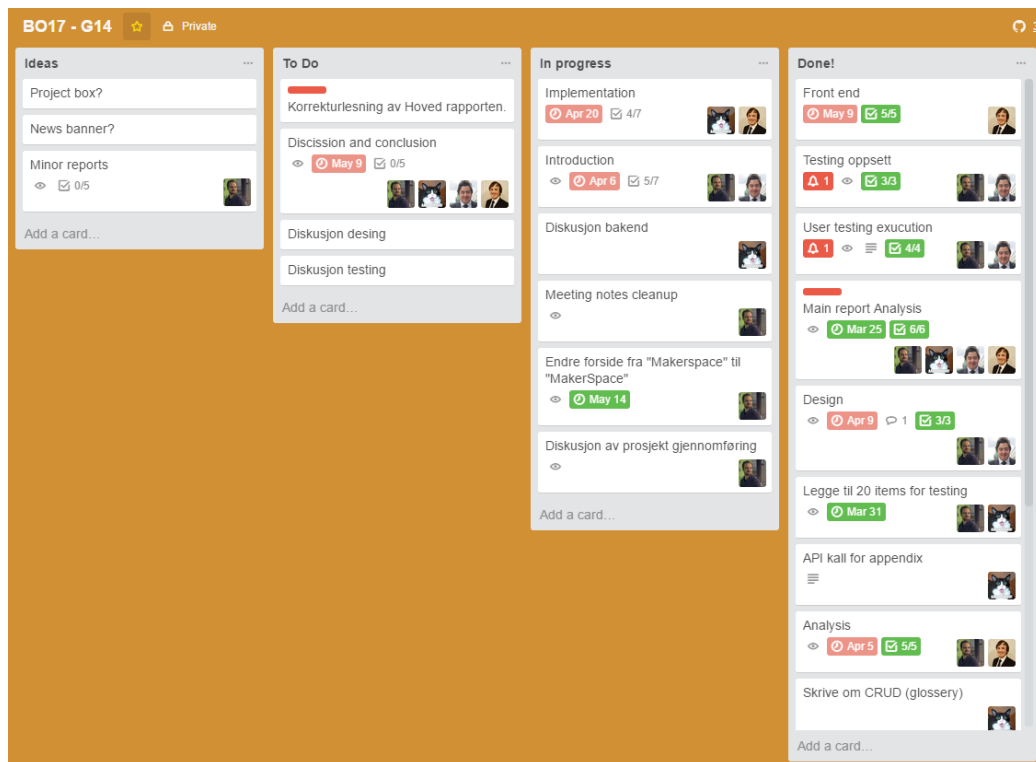


Figure 2.2: Trello bord

### 2.3.3 L<sup>A</sup>T<sub>E</sub>X

The group chose to use L<sup>A</sup>T<sub>E</sub>X for writing the report and the documentation. The Minutes are also written in L<sup>A</sup>T<sub>E</sub>X.

The group chose to use L<sup>A</sup>T<sub>E</sub>X over other text editors, because of the usability. When it comes to large projects and reports, L<sup>A</sup>T<sub>E</sub>X is superior to other editors.

It is proven [2] that L<sup>A</sup>T<sub>E</sub>X is easier to use and more manageable on larger reports and projects, over for example Word. When a document becomes complex, it is a lot easier to use L<sup>A</sup>T<sub>E</sub>X. As can be seen in the graph below (figure 2.3).

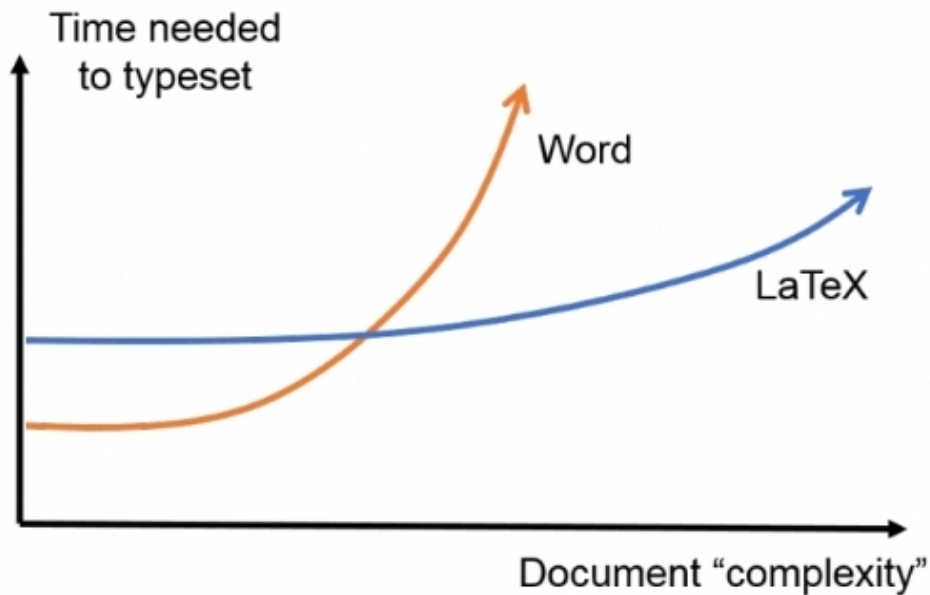


Figure 2.3: Graph about document complexity.



# **Chapter 3**

## **Analysis**

This chapter show the groups analysis of the task, tools, security, cost, technology and analyses the use of the system.

## 3.1 The task

The task comes from a need to get a control over the items that exists in MakerSpace. What they are and approximately how many there are. The system will need the student assistants that work on MakerSpace to add and remove items. The leader of MakerSpace will need to have the same rights and the possibility to add and remove student assistants. There will not be a need to get an exact count on items like small LEDs and screws. However bigger and more expensive equipment like Raspberry Pis or drones will need an exact count.

## 3.2 Open Source

It is more common to develop community software like a MakerSpace inventory system open source, than make everything closed and away from the public. There are many reasons why software should be open source, and what makes open source better. The MakerSpace Management System is licensed under the MIT License.

### 3.2.1 MIT License

The MIT License is a license for free software. It is a permissive license, and puts only a few restrictions on reuse of the software. The license originated from Massachusetts Institute of Technology (MIT).

### 3.2.2 MakerSpace Management System MIT License

The MIT License (MIT)

Copyright (c) 2017- MakerManagement <https://makermanagement.github.io/>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### **3.2.3 Security**

Software developed open source, tends to have a higher level of security than closed software. Since open source software is viewed by everyone who is interested, errors and flaws of the software will be found more easily and fixed quicker than if the software was closed off.

“Security through obscurity” is a popular statement used by companies and people who want their software closed off and restricted from the public. Open source is the opposite from this, and encourages others to help and contribute towards the software. A good example for when open source was a good thing, was when the Linux kernel had a bug which caused people to exploit the software[3]. The bug got fixed quickly, because of the Linux community. This would not be the case in bigger, closed off companies. It could take weeks, even months for a patch in for example Windows.

### **3.2.4 Cost**

The cost of open source software will in most cases be notably lower than closed off software. As software developed open source often uses other open source technologies, it will be practically no cost to run the software.

### 3.2.5 Support and maintainability

To find support and help for most open source software the user can just search for the problem online. There are large groups of people helping each other in online communities, and it is normally free.

## 3.3 The Security Aspect

### 3.3.1 Using existing systems (HTTP Auth, OAuth2, Etc.)

There are already countless authentication and security systems which differ in features and how complex the encryption method is. The benefit of using an existing system is that it is maintained by a larger group of people and is well tested so when it is fully set up, it will work well as described in the documentation, which is also another positive aspect of using an existing system.

A negative aspect of using an existing system is that there is a higher possibility that the system can be hacked/cracked, therefore it will be essential to make sure the system is always up-to-date. Furthermore many of these existing systems allow multiple authentication methods to be used at the same time. (In an example using OAuth2, you can allow login via Google, Facebook and Feide. Here all 3 methods can be attacked and you are trusting these separate systems that they will not grant a unwanted user access.)

### 3.3.2 Custom system

Making a custom system can be both more secure and easier to gain access to. As it is a custom system, it can be better tailored to the specific needs of the project. Whereas an existing system will have loads of functionalities that are not needed, and these extra functions could have vulnerabilities that are unpredictable.

However a custom system requires more upkeep as issues and vulnerabilities could be found after it is created and implemented. In this system no sensitive information other than a users password will be stored, so the system would have to make sure that passwords are stored as securely as possible.

### 3.3.3 Internet vs Intranet

The employer has requested that the website should only be accessible while at the university, as the system is a listing of equipment in MakerSpace and could give potential thieves an incentive to steal the equipment. This request adds another layer of security as the website is only accessible through the Intranet.

As the system will be installed at the Østfold University College there are already existing systems that monitor the traffic and log suspect activity as well as all WiFi users being authenticated. In this situation the group would just have to make sure that the server logs all activity so if there is an attempt on hacking the system the logs and the universities systems can be used to find and block the hacker.

The only issue here is that a local custom authentication system has to be used that does not rely on external authenticators. As they all require a connection to the server that a user is trying to authenticate to, in order to make sure that the user is actually trying to authenticate with that specific server.

## 3.4 The Technology

The technology used in developing the MakerSpace Management System is chosen out of a couple of different factors.

Since the system is open source and a part of MakerSpace, it is important to use languages and frameworks that are open source and well documented. Another important factor is that the technologies are well supported and easy to maintain.

### 3.4.1 Front-end

#### 3.4.1.1 JavaScript / jQuery

JavaScript will be used for the main part of the system. Since the front-end of the system is a webpage, it is only natural to use JavaScript. JavaScript is well documented, well known and fits the needs of the system.

jQuery is a JavaScript framework, but supports more GUI and UI functions which will be useful for the user experience.

Since JavaScript is client side, it is less of a load on the server. This means the server can be small and efficient, and can be placed on a Raspberry Pi or similar pocket-size computers.

#### **3.4.1.2 HTML5 / CSS3**

HTML and CSS are the way of showing the information to the user. The whole system is written from scratch, and follows standard conventions from W3C. HTML and CSS are also accessible through all browsers and devices, and will always be shown to the user.

#### **3.4.1.3 PHP7**

PHP is a big part of the system and webdesign in general. The system uses PHP for a couple of different reasons, like language support, user authentication, and page control.

One of the big advantages with PHP is the support against API's. The system will utilize PHP for sending and receiving data from the API, and will handle the different HTML forms. PHP also works well with cURL, which is a command line tool which supports HTTP/HTTPS requests[4]. This means the system can ask the API through PHP and cURL, and support full CRUD.

Like JavaScript, PHP is also widely used and well supported.

### **3.4.2 Back-end**

#### **3.4.2.1 MongoDB**

MongoDB is a database system that is classified as a NOSQL system, NOSQL is where the data in the database is stored and retrieved in another way than the standard table method found in SQL type databases like MySQL. MongoDB stores its data as JSON-like data and therefore is an object-oriented database system. In NOSQL type databases the models for data stores are dynamic therefore during development changes can be made to the database that will not effect already existing data, where in a SQL database the whole table would have to be changed to match the new database model. [5]

These positive factors of MongoDB make the general development of software easier, as changing the model will not affect the JSON objects.

### 3.4.2.2 Node.js / Mongoose

Node.js is very similar to its counterpart JavaScript<sup>1</sup>. As JavaScript is mainly client-side scripting and Node.js is server-side scripting. JavaScript is embedded in a website's HTML code and is run client-side by a JavaScript engine. Node.js oppositely is run on server-side where it can change the dynamics of a website before it is sent to the client. Node.js is very good at handling a lot of asynchronous operations. This makes Node.js a good choice for real-time web applications for example chat systems and online games. However due to the multiple packages you can add to Node.js applications and the simplicity in coding it is widely used as a REST API platform. Furthermore, as it can handle many asynchronous operations it should be able to handle many API calls from different clients. [6]

Mongoose<sup>2</sup> is a package that is installed via the Node.js package manager (npm<sup>3</sup>). This can be used to make communication with the database server easier and assist with error handling. This package helps with connections to the database, model creation and helps with objects that rely on other objects for information (relation based objects) and seems to be the most used package for communication with MongoDB and is also the package recommended by the MongoDB team.

### 3.4.2.3 REST API

REST is short for Representational State Transfer, it may also in some cases be referred to as RESTful. A system being RESTful means that it follows 3 main features:

- Client - Server: Meaning that the client handles the front-end functionality and the server handles back-end operations. Where front and back-end can be replaced independently.
- Stateless: Data from the client is not stored on the server between requests and all session information is saved with the client.
- Cacheable: To improve performance the client can cache the response from the RESTful service, therefore it will not have to re-

---

<sup>1</sup><https://www.javascript.com/>

<sup>2</sup><http://mongoosejs.com/>

<sup>3</sup><https://www.npmjs.com/>

quest the data that it already has. This then improves the performance of both the client and REST service.

There is also a common practice with all REST APIs that they all return data in the form of JSON. As JSON data can be consumed and created by almost every programming language. By building a RESTful service it is possible to create mobile apps and other applications in the future. [7]

#### 3.4.2.4 Linux Apache

Linux is a open source kernel (operating system) clone of UNIX<sup>4</sup>. It is lightweight and therefore requires very little hardware to operate itself. As it is community driven, it is very secure with many supported and well documented programs. Most members of the group have experience using this operating system from classes at the University College, which will result in general setup of the system being quite short as well as any issues that may arise can be handled and fixed fairly easy. Also the University College is able to provide a virtual machine running a flavor of Linux called Ubuntu Server<sup>5</sup>

Apache is a one of the most used web server software and is like most other Linux software, open source and is maintained by the *Apache Software Foundation*<sup>6</sup>. Like with Linux, most members of the group are familiar with the software and require almost no configuration to run a simple website. However there will have to be some edits to the configuration to make it work with the API, but the research seems to point out that this will not be a problem.

---

<sup>4</sup><http://opengroup.org/unix>

<sup>5</sup><https://www.ubuntu.com/server>

<sup>6</sup><https://www.apache.org/>



## 3.5 Users

What can the different user roles do. The three main users of this system will be regular user, student assistant and admin. A regular user is a student or employee at the HiØ without any commitments in MakerSpace. A student assistant is employed by MakerSpace to help with daily activities. The admin has responsibility of the overall system and ordering new items that have low stock.

### 3.5.1 User Stories

A user story is a different scenario of tasks a user, admin or student assistant can do. This will help to understand the functionality of the system, and provide a solution for the different scenarios.

- As a User  
I want to find a LED bulb  
So I can check the availability

*Solution: Use the search field to find the LED bulb*

- As a User  
I want to send an e-mail to MakerSpace  
To check when an item is available

*Solution: Click on "Contact Us" and get redirected to your computers mail client*

- As a Student Assistant  
I want to reply an e-mail  
So I can provide service

*Solution: Log in to the e-mail of MakerSpace to read and reply*

- As a Student Assistant  
I want to add a new Category  
Because that Category does not exist.

*Solution: Log in to the admin panel, click "Add new category" and fill out the form*

- As a Student Assistant  
I want to add a new item Location  
Because that Location does not exist.

*Solution: Log in to the admin panel, click "Add new location" and fill out the form*

- As a Admin  
I want to check the availability of an item  
So I can order more if needed

*Solution: Log in to the admin panel, to get a full overview of availability*

- As a Admin  
I want to add new items  
Because these items do not exist in the database

*Solution: Log in to admin panel and click "Add new item", fill out the form*

# **Chapter 4**

## **Design**

This is where the design of the system is discussed. The design process, implementations and decisions during the project are addressed here.

## 4.1 What is already out there?

The different Makerspaces around the world use different systems for inventory. After researching online, it seems like it is only Dallas MakerSpace that has any information about it online. They have an inventory system, but it is no longer under development since 27 February 2013. This shows how an universal inventory system will be useful, not only for the MakerSpace at HiØ but also Makerspaces around the world. After looking into the system Dallas MakerSpace uses, it was decided to going for a new design. Mostly because their system is old, and not well developed. The new system is versatile and open source, so it is easy for other Makerspaces to implement and run the same system.

Even the different Makerspaces in Norway use standard shelves and paper notes on what they have in inventory, and do not have any system for inventory. Since MakerSpace stands for technology and development, a system to keep track of the inventory should be mandatory.

## 4.2 Site layout

The site layout is focused on easy recognizability through the use of models based on the business model from sites like Komplet.no<sup>1</sup>, Kjell & Company<sup>2</sup> and Thingiverse<sup>3</sup>. This is because the model is well tested and easy to recognize for the user. These kind of designs were chosen because they are product/item based. The basis is a landing page with featured items with a search bar and usually a menu with categories to the left or on the top of the site. Like most sites a language selection is at the top right corner. For each product item/product page there is usually a description, specification, product reviews and if it is in stock and sometimes how many there are. Out of that information the wireframes are made.

---

<sup>1</sup><https://www.komplett.no/>

<sup>2</sup><https://www.kjell.com/no/>

<sup>3</sup><http://www.thingiverse.com/>

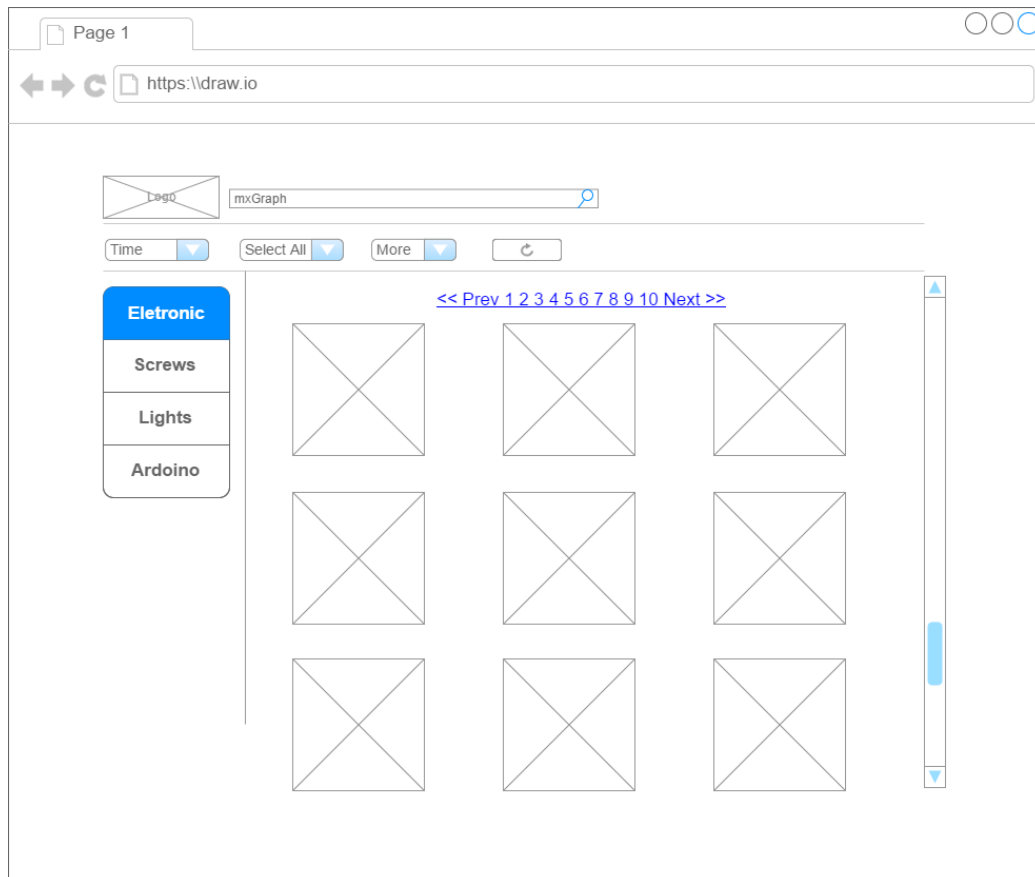


Figure 4.1: Wireframe for the landing page

The start/landing-page makes use of search, categories, and item layout with pictures for each item.

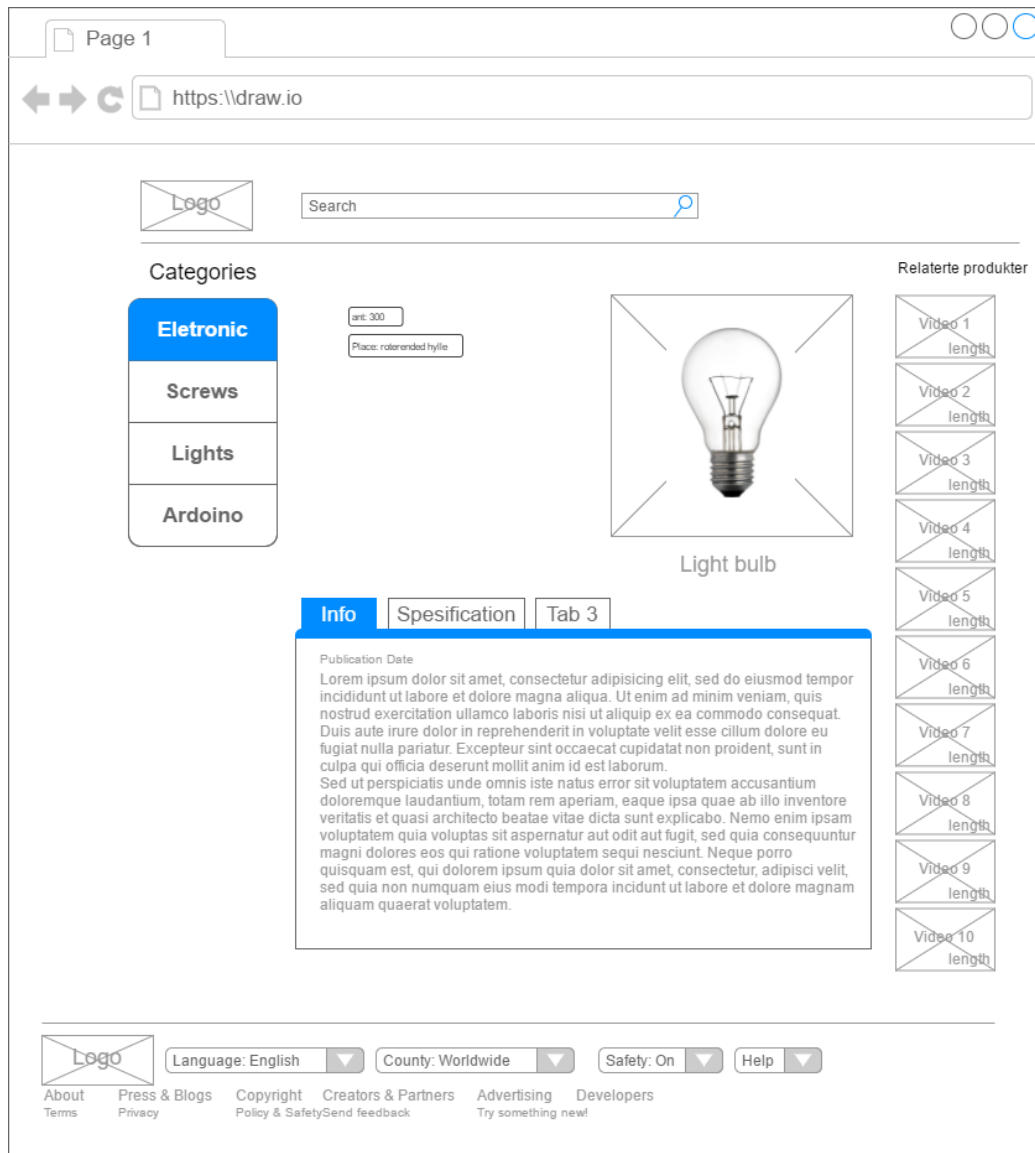


Figure 4.2: Wireframe for the item page

The item page makes use of a picture, amount, specification box, information box and location for the item.

The first plan for the site was to have one item page for each and every item with a description, professional product information, a good picture and the correct amount. However, after discussion with the supervisor about how the workforce of MakerSpace is limited, the idea was changed. There are only 2-3 student assistants working at part time. The work that would go into making sure each item had a lot of

information would take a lot of work. The danger is if the workload is too big to create items in the page then it might end up that it is not used at all. The workload to use the site must be so low and mostly effortless so that it is possible to do on the low work hours that Maker-Space student assistants have. The change was that some items will get grouped together. Like LED lights or filaments for 3D-printers. As well as add color layout and so on in the description. Most items will not display amounts, since this kind of inventory will take too much time to keep up-to-date. Only some items like development boards, drones, printers and other expensive equipment will have amounts. A button to inform that an item is missing or in low supply, is also added.

### **4.3 MongoDB Models**

As stated in chapter 3, the database is a NOSQL type database there is no good way to properly represent the database models. It was therefore chosen to display them in two ways; the `model.js` file in the NodeJS server and a output from the REST API in its default JSON format.

### 4.3.1 Items

#### 4.3.1.1 Node.js Model

```
1 var ItemSchema = new mongoose.Schema(  
2   {  
3     item_name: {  
4       type: String,  
5       unique: true,  
6       required: true  
7     },  
8     description: {  
9       en: String,  
10      no: String  
11    },  
12    short_description: {  
13      en: String,  
14      no: String  
15    },  
16    categories: [  
17      {  
18        type: mongoose.Schema.ObjectId,  
19        ref: 'Category'  
20      }  
21    ],  
22    tags: [  
23      {  
24        type: mongoose.Schema.ObjectId,  
25        ref: 'Tag'  
26      }  
27    ],  
28    locale: {  
29      type: mongoose.Schema.ObjectId,  
30      ref: 'Location'  
31    },  
32    image_url: String,  
33    quantity: Number  
34  }  
35 );
```



### 4.3.1.2 JSON Model

```
1 {
2   "_id": "58de638c5edb8f418b9c22d5",
3   "quantity": 5,
4   "image_url": "http://website.com/image.jpg",
5   "item_name": "A Item",
6   "__v": 0,
7   "tags": [
8     {
9       "_id": "58b445ecbdc857062a247226",
10      "__v": 0,
11      "tag": {
12        "en": "arduino",
13        "no": "arduino"
14      }
15    }
16  ],
17  "categories": [
18    {
19      "_id": "58ac2d03bdc857062a24721a",
20      "__v": 0,
21      "category": {
22        "en": "Arduino",
23        "no": "Arduino"
24      }
25    }
26  ],
27  "description": {
28    "en": "A item.",
29    "no": "En ting."
30  }
31 }
```

## 4.3.2 Category

### 4.3.2.1 Node.js Model

```
1 var CategorySchema = new mongoose.Schema(  
2   {  
3     category: {  
4       en: {type: String, required: true},  
5       no: String  
6     }  
7   }  
8 );
```

### 4.3.2.2 JSON Model

```
1 {  
2   "_id": "58a3309f170b430a6a835012",  
3   "__v": 0,  
4   "category": {  
5     "en": "Test Category 1",  
6     "no": "Test Kategori 1"  
7   }  
8 }
```

### 4.3.3 Tag

#### 4.3.3.1 Node.js Model

```
1 var TagSchema = new mongoose.Schema(  
2   {  
3     tag: {  
4       en: {type: String, required: true},  
5       no: String  
6     }  
7   }  
8 );
```

#### 4.3.3.2 JSON Model

```
1 {  
2   "_id": "58b445ecbdc857062a247226",  
3   "__v": 0,  
4   "tag": {  
5     "en": "arduino",  
6     "no": "arduino"  
7   }  
8 }
```

### 4.3.4 Location/Locale

#### 4.3.4.1 Node.js Model

```
1 var LocationSchema = new mongoose.Schema(  
2   {  
3     locale: {type: String, required: true}  
4   }  
5 );
```

#### 4.3.4.2 JSON Model

```
1 {  
2   "_id": "58e1968555014b6496c614b3",  
3   "locale": "Makerspace",  
4   "__v": 0  
5 }
```

# **Chapter 5**

## **Implementation**

The group used iterations as a development method, and the different iterations are listed with details below. The back-end is developed all throughout the project, but mostly after the front-end specifications. If it was needed for functions or support for the front-end, it got implemented in the back-end as well. Because of this, there will not be a lot of discussions about how the different iterations changed the back-end.

## 5.1 Flowcharts

The group developed the system after some simple flowchart guidelines. These flowcharts were utilized in the implementation of the system.

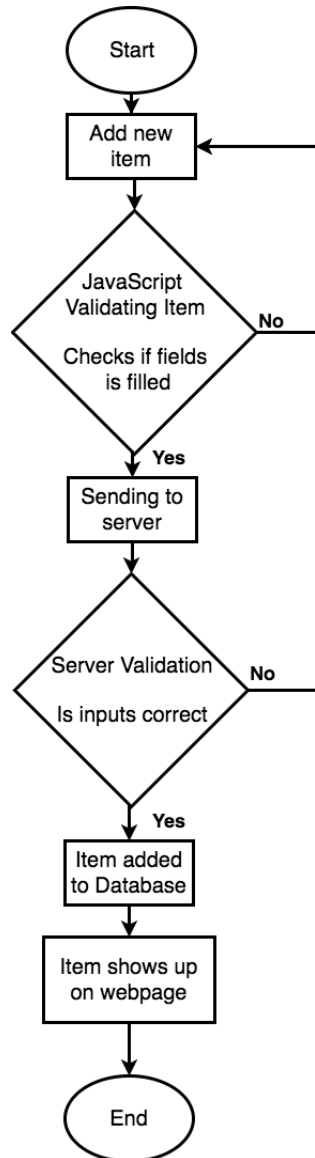


Figure 5.1: Flowchart for adding an item

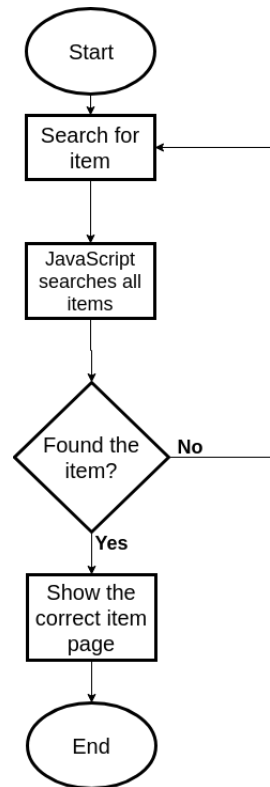


Figure 5.2: Flowchart for searching for an item

## 5.2 First iteration

### 5.2.1 Website (Front-end)

The first website version did not call the API to receive items from the server, but used a local JSON file.

#### 5.2.1.1 Design

As shown in the design site layout chapter 4, the landing page is based on the wireframe from figure 4.1. The page is simple, but shows an early version of the design.

After discussions with the tutor and the employer, the first design needed to be simplified. Since the system will be updated by a small group of people, it must be simple and cannot take too much time to update items.

The system started out as a page that looks much like sites like Komplet.no, mpx.no and other online stores, but had to be shortened down a fair bit.

In this version of the system, it was not possible to click any items and get more information about them. It is also not possible to click the categories.

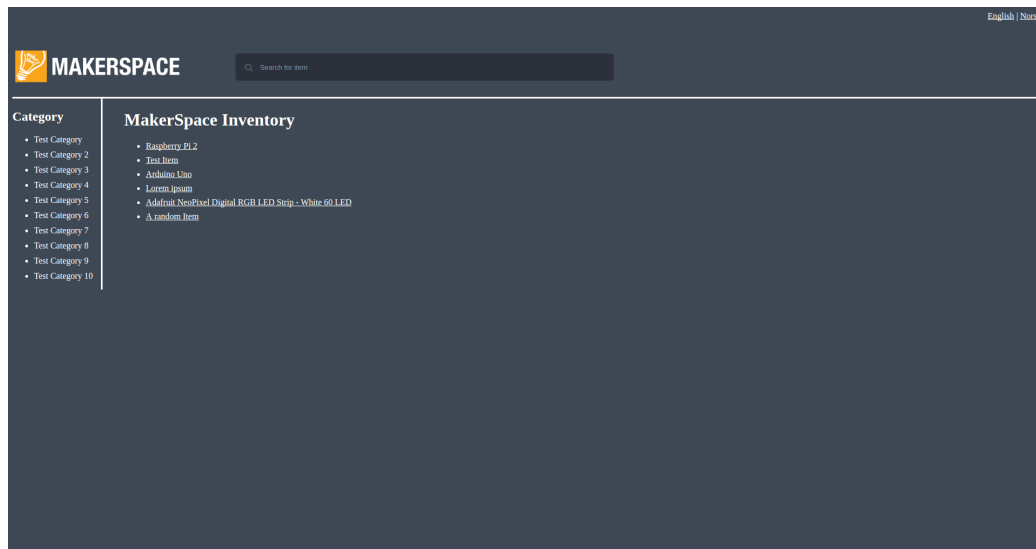


Figure 5.3: Screenshot of the landing page English (first iteration)



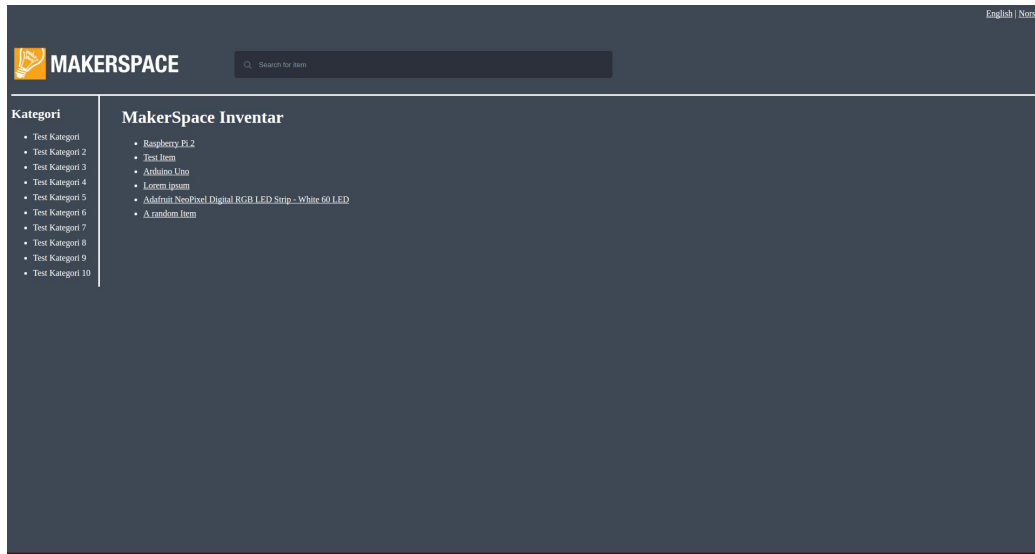


Figure 5.4: Screenshot of the landing page Norwegian (first iteration)

As seen in figure 5.3 and figure 5.4 there are different languages by choosing language in the right corner.

#### 5.2.1.2 Technology

Even though this is an early version of the system, the language support works well. The system uses PHP and JavaScript to achieve the functionality (discussed in chapter 3), and is responsive. The PHP handles language support, and JavaScript for JSON handling.

The language can be changed by clicking the different languages in the right corner. This will change the link accordingly.

For example:

`http://158.39.162.161/?lang=no`

and

`http://158.39.162.161/?lang=en`

#### 5.2.2 Node.js API (Back-end)

The first iterations of the API only had GET and POST calls of items and GET calls for categories. It would either return all objects or a single object based on the URL.

The following endpoints were available in the first iteration of the REST API:

- GET -> `http://api.url/api/items/`
  - Returns all items
- POST -> `http://api.url/api/items/`
  - Adds an item and returns the new item
- GET -> `http://api.url/api/items/[ITEM-ID]`
  - Returns single items that match ITEM-ID
- GET -> `http://api.url/api/categories/`
  - Returns all categories

## 5.3 Second iteration

### 5.3.1 Website (Front-end)

The second iteration of the system contains API calls, so all items listed on the page are collected from the API.

The system utilizes the API calls from the first iteration, and returns a JSON structure. It then lists all items in the JSON and displays each item on the webpage.

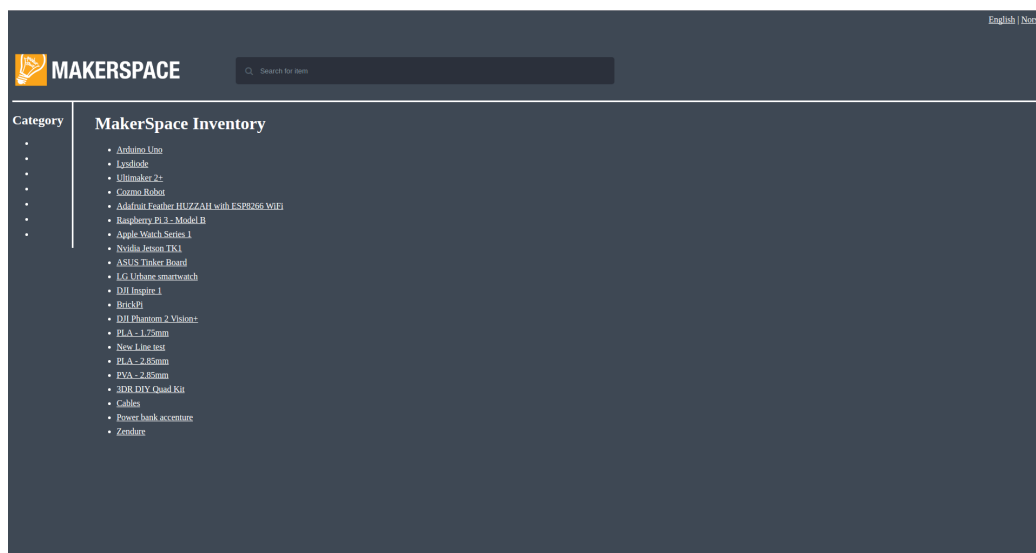


Figure 5.5: Screenshot of the landing page (second iteration)

In figure 5.5 you can see how all the items from the API are listed on the page.

### 5.3.2 Back-end

During this iteration all CRUD operations were made for items and categories. However the front-end system at that time did not use any functions other than read, so most operations were done via Postman. These functions were planned for the next iteration of the front-end. Progress after this iteration is to start making the endpoints for tags and item locations.

## 5.4 Third iteration

### 5.4.1 Website (Front-end)

In this iteration, the system got a major overhaul and loads of functions were added.

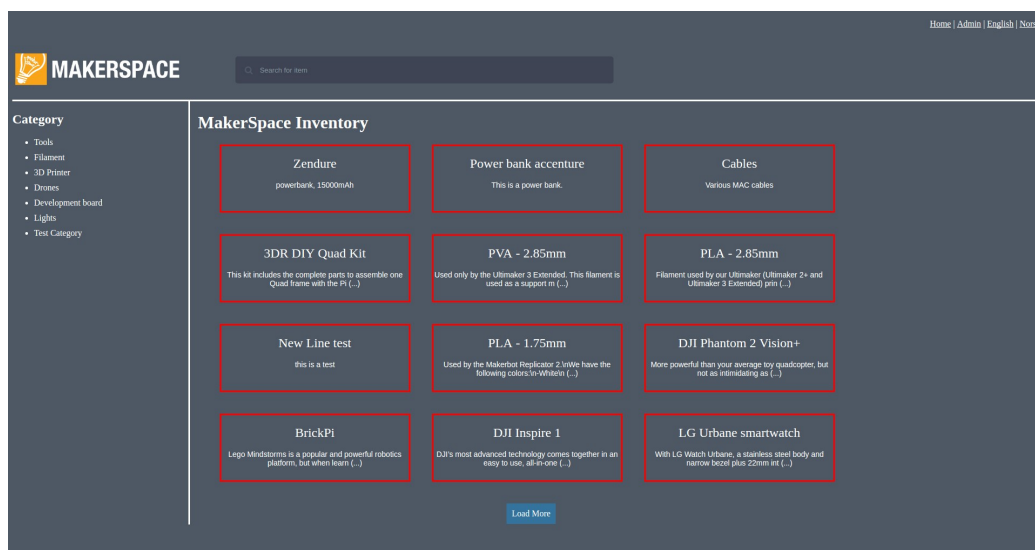


Figure 5.6: Screenshot of the landing page (third iteration)

#### 5.4.1.1 General changes

The items are easier to see, and they contain a title and a description. The categories are now listed on the side, from the API. There is also a button for loading more items on the page. There are only 12 items loaded when entering the website, but you can click the Load More button to get 12 more items. This makes the site tidier and look cleaner.

### 5.4.1.2 Search field

The search field also works in this iteration. When the user starts to write in the search field, the system shows all items that contains the letter or word. See figure 5.7

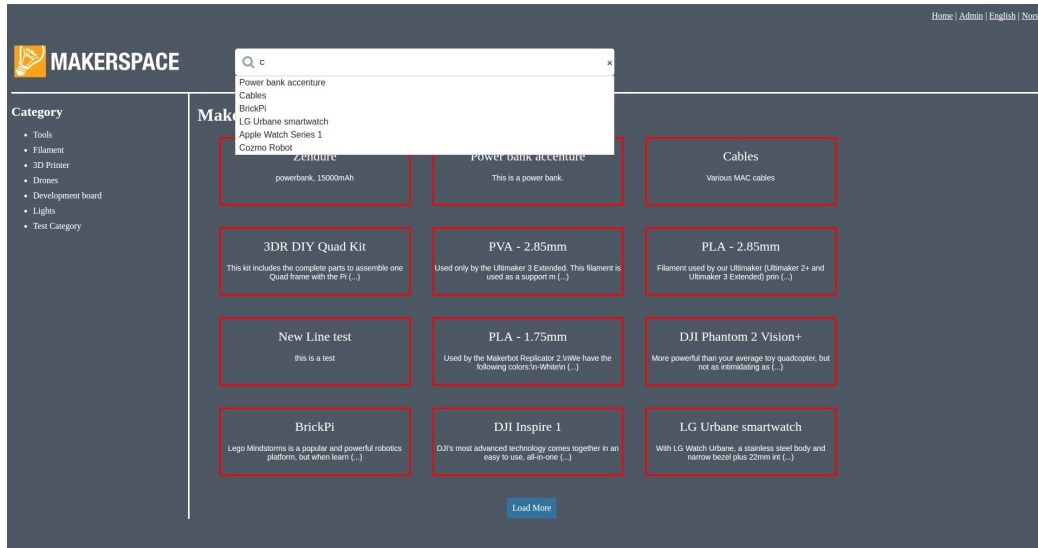


Figure 5.7: Screenshot of the landing page with search (third iteration)

### 5.4.1.3 Item page

The item page is a big part of the system, and is the main area where the user will get information about an item. In this iteration, the user can find a description and a picture of the specific item.

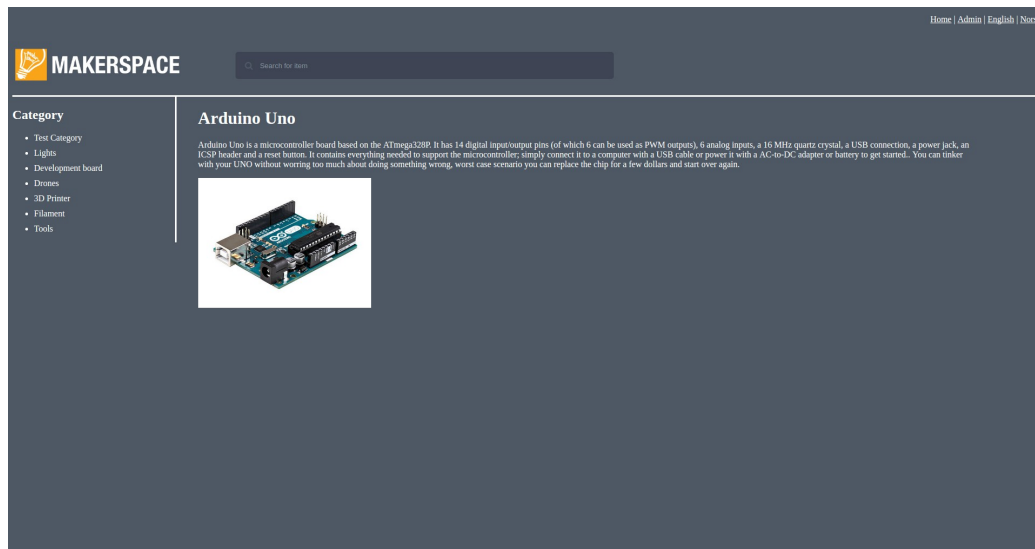


Figure 5.8: Screenshot of the item page of a specific item (third iteration)

The specific item gets chosen with PHP, through the ID of the item.

For example:

`http://158.39.162.161/itempage.php?item=58ab8f776bfd630ef83b5509`

or

`http://158.39.162.161/itempage.php?item=58ac0ace6bfd630ef83b550c`

#### 5.4.1.4 Administrator page

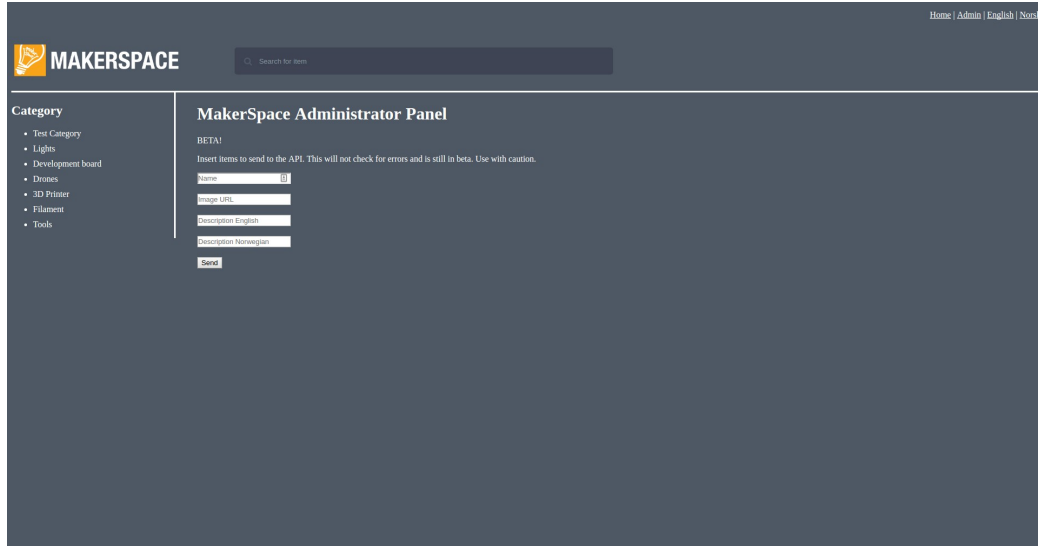


Figure 5.9: Screenshot of the administrator page (third iteration)

An administrator page is implemented where the system administrator and those who works at MakerSpace, can add new items to the API. These will automatically appear on the landing page when the admin clicks send. The simple design of the administrator panel can be seen in figure 5.9.

#### 5.4.2 Back-end

At this stage all endpoints (item, categories, location, tags) have CRUD functionality. These functions are to be added to the front-end on its next iteration. Work now will consist of research and testing of authentication for the API.

## 5.5 Fourth iteration

### 5.5.1 Website (Front-end)

This is the final iteration of the system, where most functionalities are implemented and work well.

Most of the design remains the same, with some changes to the item page design.

#### 5.5.1.1 General changes

The landing page looks much like the other iterations. The biggest difference here is that the categories now work.

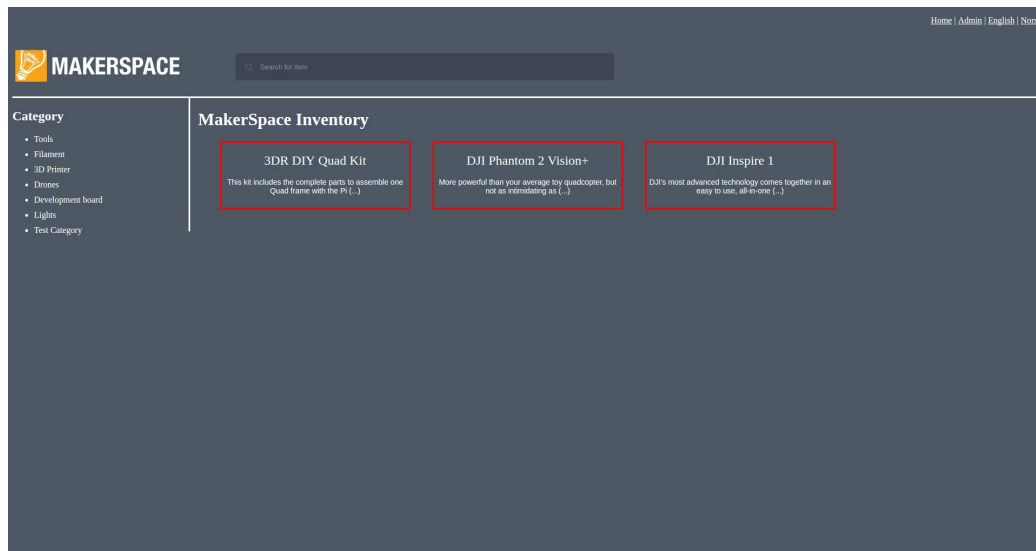


Figure 5.10: Screenshot of the landing page with categories (fourth iteration)

The categories get chosen with PHP, through the ID of the category. You can see this in the URL field of the browser, For example:  
`http://158.39.162.161/index.php?category=58e38f5fee85df30572ed61c`  
or  
`http://158.39.162.161/index.php?category=58e38ea9ee85df30572ed61b`

### 5.5.1.2 Item page

The item page got some changes regarding how the admins will see it, plus some additional information about quantity and location.

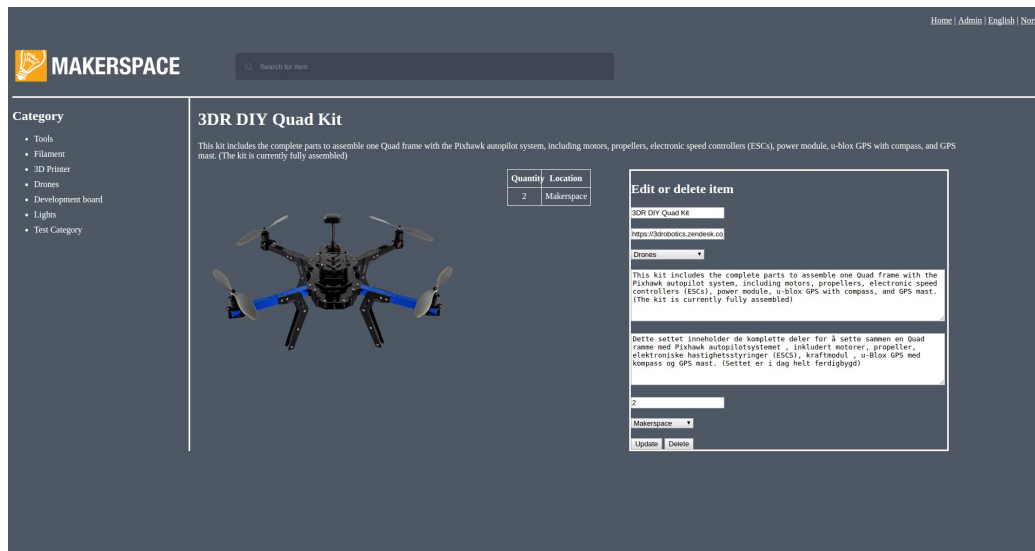


Figure 5.11: Screenshot of the item page (fourth iteration)

As seen in figure 5.11, there is now an edit field on the right side of the item. This field is only visible to admins and employees at Maker-Space, and it is the only way of editing an item.

When the admin clicks update or delete on this form, it sends the information to the API and either updates (PUT request) or deletes (DELETE request) the item.

The user would just see the item page without the edit item option. They would still see quantity and location of the item, since this is valuable information. See figure 5.12 for an example of an item as a user.



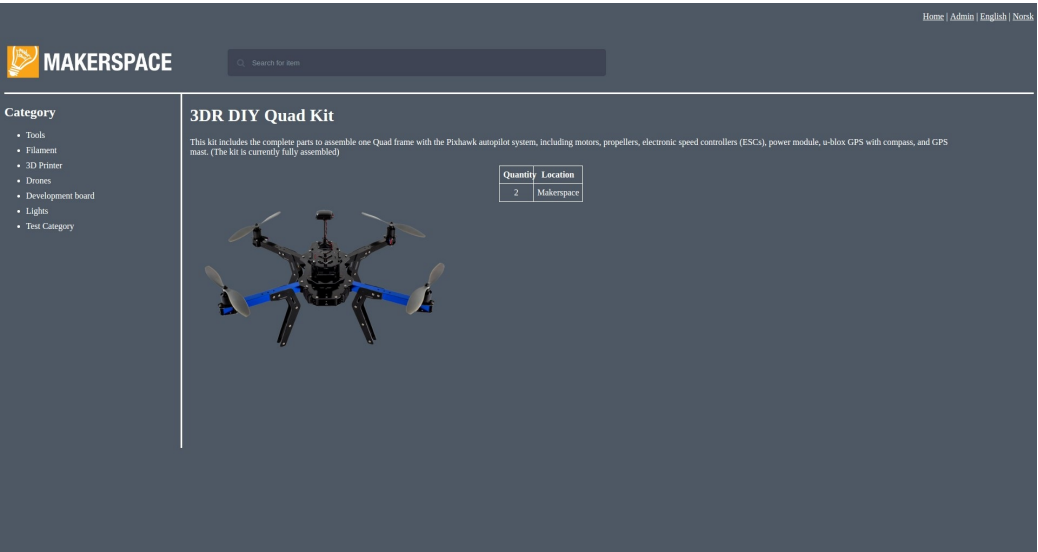


Figure 5.12: Screenshot of the item page as a user (fourth iteration)

5.5.1.3 Administrator page

The administrator page now supports adding new items, new categories and new locations. The administrator page is dynamic, and it is only a small part of the content that needs to be changed out when the admin interacts.

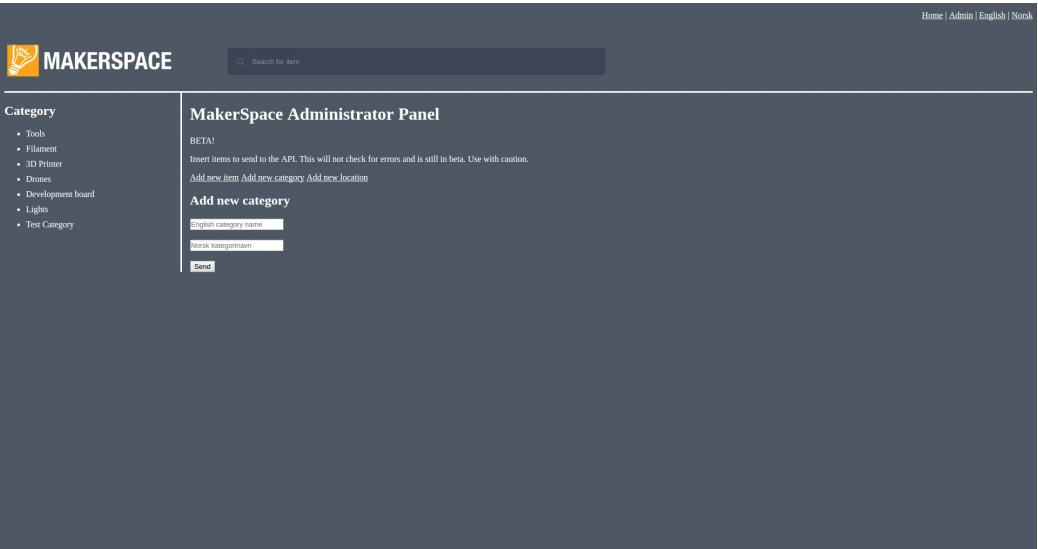


Figure 5.13: Screenshot of the administrator page (fourth iteration)

As seen in figure 5.13 the administrator can now add new categories and locations.

To add new locations or items, the administrator only needs to click the different links. This will change the URL, and PHP will change the content. For example:

`http://158.39.162.161/admin.php?type=item`

or

`http://158.39.162.161/admin.php?type=category`

### **5.5.2 Back-end**

Here continuous work on authentication has been done. The possibility to use the national authentication platform Feide has been discussed and will be looked into.

# **Chapter 6**

## **Testing**

This chapter goes through the test process from start to execution. What tasks the test persons had and results from the open interview at the end of the test.

## 6.1 Why do a user test

A user test is to get the insight needed to make a site user friendly. [8]

- Simulates a close to real situation.
- Has concrete tasks for the user.
- Is a test to observe the user.
- Evaluates the user-friendliness of the site.

## 6.2 Test goal

The goal is to test the site's usability. This include finding items, using categories and adding new items.

To ensure the system does what it is supposed to do, the group needs to verify this through a usability test. The group needs to ensure that the new functions work as intended after implementation, as already existing functions can get affected after rolling out the new implementation. [9]

Testing verifies that the system meets requirements and verifications to ensure that the system is built correctly. Testing helps to validate that the system is being developed for what the user needs and expects. [10]

## 6.3 Methods

### 6.3.1 Usability Testing

A qualitative method for testing the usability of a product is used, which means giving the user a set of tasks to do on the product and evaluating the results after the test. It is important to note that it is the product that is tested, not the user. [8]

## 6.4 Target Audience

The target audience will be students and employees at HiØ, that do not have as much knowledge about MakerSpace and its items. This is to

make sure the system is optimal to understand for new users.

The test group will consist of 4 students, 2 male and 2 female, and 2 employees, 1 representative from each sex.

### **6.4.1 User Types**

#### **6.4.1.1 Student**

Students are the main users in the user group. They will be using the system mainly to search for items, look up information on specific items and notifying when an item is low or out of stock.

#### **6.4.1.2 Student Assistant**

Student Assistants will be maintaining the inventory. This by adding, editing and deleting - items, categories and locations.

#### **6.4.1.3 Employee / Admin of MakerSpace**

The admin of MakerSpace will mainly have an overview of the inventory. The admin can also add, edit and delete users. This can be admins and student assistants.

## 6.5 Test Execution

### 6.5.1 Roles

#### 6.5.1.1 User

The user will go through different instructions given by the test leader for the site. The reactions and problems the user will encounter will be valuable test data to improve the site.

#### 6.5.1.2 Test Leader

The test leader will give instructions to the user and note all activity.

### 6.5.2 Setup

The user will be in the same room as the test leader. This is because there is no good test environment available to have the user and the test leader in separate rooms, like what would be normal for user tests.

The user will sit in an environment with as few distractions as possible. The test leader will sit behind the user to not distract and guide the user with body language or other non-verbal communication.

### 6.5.3 Tasks

1. Find any kind of a LED light bulb that is the color red.

*The goal of this test is to see if the user will prefer to find the LED light bulb by using category or use the search bar*

2. Find Cozmo and tell us what it can be used for.

*The goal of this test is to see how the user will find an item they do not know how to spell and if the search bar will help autocorrect.*

3. Change language of the site from English to Norwegian.

*The goal of this test is to see if the user easily can find where to change the language of the site, or if this should be emphasized more.*

4. Add a random item (that the test leader provides) to the site.

*The goal of this test is to see if it's easy to understand for a user how to add a new item.*

5. Find Arduino Uno without using the search box.

*The goal of this test is to check what the user prefers to use. Category list or the inventory list.*

#### 6.5.4 After interview?

After testing the different tasks the test person got some follow-up questions regarding the MakerSpace site.

1. Did you get an overview over what kind of items there are in MakerSpace?
2. Did you feel the site, interface and design was easy to understand and figure out?
3. Do you have any suggestions to features that were missing or any improvements to the site?

## 6.6 Results

In this section we present the results from the interview after we had the test. A full set of the result will be found in Appendix A

### 6.6.1 Interview

**Did you get an overview over what kind of items there are in MakerSpace?**

Three of the six people who were in the test group said it was easy to get an overview over the site and the products, and easy to read information from the product boxes. The rest said it was not visually represented well, too much text in the product boxes and that it took a lot of time to understand what MakerSpace inventory was.

**Did you feel the site, interface and design was easy to understand and figure out?**

Five said it was easy to understand the site, and easy to get an overview. One person complained that the categories were not clear enough.

**Do you have any suggestions to features that were missing or any improvements to the site?**

Need more focus on a clean inventory site and more items. Need a site with overview over employees at MakerSpace. The search bar and "Load More"-button need to be more highlighted. The "Load More"-button blended too much with the background. Change name to picture in the product boxes, or have picture and name. There was a problem with the red border on the dark background, and a dark background generally does not work in a website. One said that the category bar was too small and category name were bad choices. The category names need to have the same type of clarification. E.g. "Tools" is a vague category, where "Drones" is more specific.



# **Chapter 7**

## **Discussion and Evaluation**

This chapter will discuss design, implementation, testing, method and possible future work elements.

## 7.1 Design

During the start of the project the group had ambitions to make a website with design that was equal to Komplet.no and Kjell&Company, a webshop theme.

After discussions with the mentor and the employer of the project, the design got simplified quite a bit. This was to make it more friendly to those who need to update and add new information to the system.

## 7.2 Implementation

The group discussed the use of technologies and how the project should be developed. Since it is a MakerSpace development system, the group decided to use technologies they teach at Østfold University College and make everything without the use of frameworks. The development of the system will continue by students, and employees at MakerSpace.

The group checked out different frameworks to see if it would make development of the website easier, and concluded that AngularJS<sup>1</sup> would make development easier. AngularJS is not part of any classes or subjects at Østfold University College, which makes it difficult for students to update and keep developing on the system. So even though frameworks would make the development process easier, the group decided to go for barebone JavaScript and PHP.

For the back-end, the group discovered and discussed to use a Node.js framework called LoopBack<sup>2</sup> that generates most of the code needed to make a REST API. However after testing it was found out that the resulting code is almost identical to what the system already had. It was then decided that there was no need to use a framework to achieve the same result.

It was easy to debug and resolve errors on the system, since it was all developed by the group, and not by a framework. Like the front-end, it is easier for any student or employee at MakerSpace to keep developing the system when it is handwritten from scratch instead of generated code.

---

<sup>1</sup><https://angular.io/>

<sup>2</sup><https://loopback.io/>

## **7.3 Usability Test**

One issue of having a usability test with few testers is how reliable the result is. With more testers, there would be more data that would give better indications on changes that need to be done.

As said in test setup 6.5.2 the optimal setup for a usability test is when the test leader and tester are in separate rooms, and the test leader can still observe what the tester is doing. This is to prevent the tester to get distracted by the test leader. This is something the group could have used a little more time to get in order.

A new iteration was never made based on the results from the user tests, as there was a shortage in time.

## **7.4 The Project Method**

The project followed the use of the incremental method 1.3.4. The project tool used to follow this was Trello 2.3.2. The groups early focus was to start on coding project as fast as possible. This was an advisement from our guidance counselor. In that early phase, with the use of extreme programming, pair programming could have helped to give more of the group better insight to the code. The other challenge was that the incremental method should have each increment clearly defined. This was not always easy to define as the group was inexperienced with the development of layer systems made from scratch, and not always knew what all necessary components were in each part.

## **7.5 Future**

### **7.5.1 Front-end**

#### **7.5.1.1 General Design**

The design of the front-end could need an update, to make it more user friendly. This can be achieved by using different frameworks or CMS systems, instead of hardcoding everything.

### 7.5.1.2 Design Item page

The item page needs to be updated to show the description, location and quantity better, since the design is currently not optimized. This can be solved by the same methods as section ??esign

### 7.5.1.3 Implement new functions

One of the functions the team has discussed, but not implemented, is a way for users to contact employers of MakerSpace. This could be an e-mail system, where the user fills out a form which sends an e-mail to the employees. This e-mail could be information about items missing, new items, wrong quantity or general questions.

## 7.5.2 Back-end

The REST API could be rebuilt to work with different frameworks, this will make the general code base smaller as many functions that are written by the group will be handled by the different frameworks. This change will make it more difficult for new students at HIØ, as they will be able to understand the basic JavaScript/Node.js code but might struggle with understanding the generated code from the framework as well as how it organizes files. However the benefit with using the framework is that future development of the API could become easier as these frameworks are usually well documented.

As this system is mainly designed to be used at universities and other university colleges it should for future releases and before it is used at other schools implement the same authentication methods that are used at schools (mainly Feide<sup>3</sup>). This will simplify the use of the system as the system as a whole will not have to store all user info, it would only have to keep a database of the authenticated users that are admins; here only the user's id would be stored not any other information like passwords as that is handled by the external authenticator.

---

<sup>3</sup><https://www.feide.no/introducing-feide>

# **Chapter 8**

## **Conclusion**

In this chapter the group will determine the conclusion on the completion of the project.

The main goal of the project was to develop and implement an inventory system for MakerSpace at Østfold University College. This is to make the management of its inventory easier. The group achieved the goal by using the incremental method to develop a well-functioning inventory system based on feedback from testers, the employer and the mentor of the project.

The system is made up by two parts, one front-end and one back-end. Users can see the website by going to <http://158.39.162.161/> on the Østfold University College network, as the employers specified.

The group can conclude that the system now allows the users to search for and find items, and employees can add items to the system.

# Bibliography

- [1] 2017. [Online]. Available: <https://www.mongodb.com/nosql-explained>.
- [2] J. Blanco, *Word or latex typesetting: Which one is more productive? finally, scientifically assessed | computer science | mapping ignorance*, 2015. [Online]. Available: <http://mappingignorance.org/2015/04/06/word-or-latex-typesetting-which-one-is-more-productive-finally-scientifically-assessed/>.
- [3] E. Teo, *Bug 634457 - cve-2010-3081 kernel: 64-bit compatibility mode stack pointer underflow*, 2010. [Online]. Available: [https://bugzilla.redhat.com/show\\_bug.cgi?id=CVE-2010-3081](https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2010-3081).
- [4] S. Pillai, *Curl command tutorial in linux with example usage*, 2014. [Online]. Available: <http://www.slashroot.in/curl-command-tutorial-linux-example-usage>.
- [5] 2017. [Online]. Available: <https://www.mongodb.com/what-is-mongodb>.
- [6] N. Foundation, *About | node.js*, 2017. [Online]. Available: <https://nodejs.org/en/about/>.
- [7] T. Fredrich, *What is rest?* 2017. [Online]. Available: <http://www.restapitutorial.com/lessons/whatisrest.html>.
- [8] E. T. Andersen and J. G. World, *Praktisk bukertesting*, 1st ed. Cap-pelen damm as, 2011.
- [9] C. Thomson, *5 reasons we need software testing - test talk*, 2014. [Online]. Available: <http://www.te52.com/testtalk/2014/08/07/5-reasons-we-need-software-testing/>.

- [10] A. Ghahrai, *Why do we test ? what is the purpose of software testing ?* 2010. [Online]. Available: <http://www.testingexcellence.com/why-do-we-test-what-is-the-purpose-of-software-testing/>.



## **Appendix A**

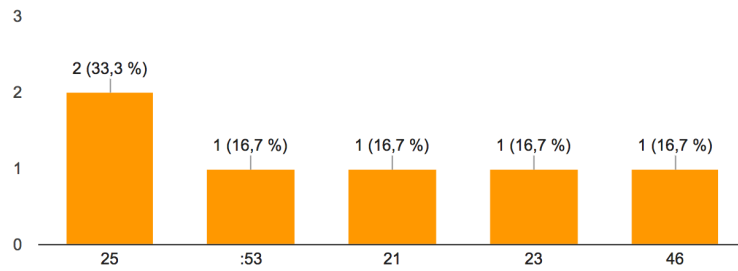
### **Raw Test Results**

## APPENDIX A. RAW TEST RESULTS

---

### Age

6 svar



### Sex

6 svar

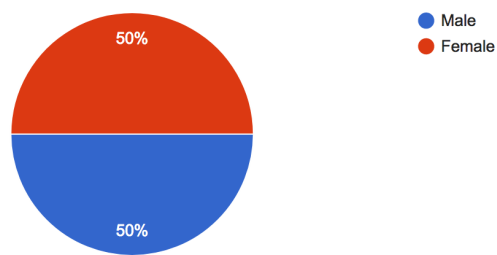
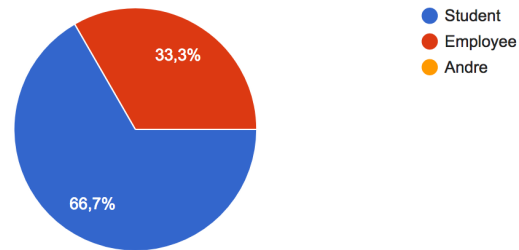


Figure A.1: Age and sex of the test persons

What role

6 svar



Do you know what MakerSpace is?

6 svar

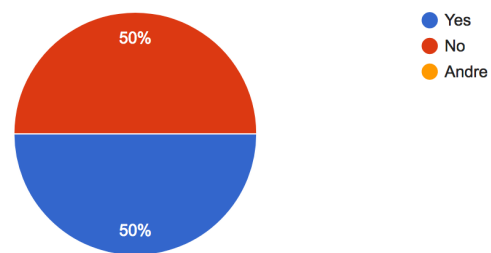


Figure A.2: If the test persons is a student or employee.

### Task Questions

1. Find any kind of a led light that a red color. (time)

6 svar

14 sek

7sek

20 sec

15 sec

2 min og 27 sek

29 sek

1. Find any kind of a led light that a red color. (What happend)

6 svar

lights , no items

Gikk rett på light kategori som ikke fungerte

Skroller nedover på siden for å finne lys

Går på kategori lys

Ble for distraheret av alle testproduktene. letet kun under inventar, de røde boksene. klikket seg inn på alle produktene, før testpersonen fant "load more" knappen.

Gikk først på søkebaren, kom ikke opp noe på søket(15sek). Etterpå gikk testperson på kategori, ingenting kom opp, men jeg velger å godkjenne .

Figure A.3: Test task 1: Find a red LED light bulb

2.Find Cosmo and tell us what it can be used for. (time)

6 svar

44
1min
10 sec
53 sec
20 sek
1 min 40 sek

2.Find Cosmo and tell us what it can be used for. (What happend)

6 svar

Kan ikke finne cosmo da søket ikke helper med skrive feil
Prøver å søke men ikke noe opp. Søk sier ikke at den ikke finner noe. Hjelper deg ikke med skrive feil. Selv om du skriver riktig MÅ du velge fra nedtrekks menyen. Skjer ingenting hvis du trykker enter.
Scroller ned og finner items
Prøver å finne det via kategorier.
letet i inventarboksene. Klikket på "load more" nesten med en gang, så fant testpersonen cozmo, og fortalte hva han kunne brukes til
Prøvde først å søke uten lykke. Gikk så gjennom alle kategorier uten å finne Cozmo, prøvde så å søke igjen, før testperson fant "Load more"-knappen på bunnen av Home-page.

Figure A.4: Test task 2: Find Cozmo the robot

3.Change language of the site from English to Norwegian. (time)

6 svar

3
7 sek
5 sek
5 sec
24 sek
3 sek

3.Change language of the site from English to Norwegian. (What happend)

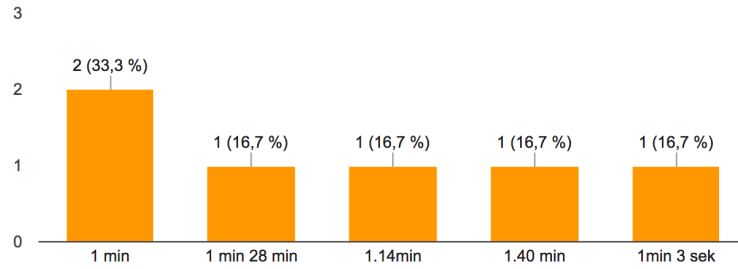
6 svar

Went straight up to stanard oppsett
Finner den lett opp til høyre
Gikk til toppen i høyre og fant byttet med en gang
Gikk rett opp til høyre.
Begynte å se i "Category" først, så begynte blikket å gå opp i høyre hjørne til testpersonen så språkalternativ i øverste hjørne.
Gikk rett til høyre hjørnet.

Figure A.5: Test task 3: Change language of the page from English to Norwegian

4.Add this item to the site "random little thing". (time)

6 svar



4.Add this item to the site "random little thing". (What happend)

6 svar

Uklare kategorier. Trenger flere kategorier

Finner siden og forstår boksene.

Mistorstod linken til å være en stor link. item cat osv. Viste ikke hva som skulle gjøres med image. Var usikker på om den var opprettet etter send.

Finner navn og deskripsjon lett. Tror at hun må laste opp et bilde i stedet for link. Prøvde å bruke komma på antall som ikke fungerte. Lurte litt på hva som var krav for å legge inn.

Gikk først til "kategori", før testpersonen fant Admin-panelet, fant fort hvordan man kunne legge til item. Tok bare en random kategori, for han fant ikke en som passet. etter å ha lagt til, gikk han inn tilbake til homepage for å se at itemen ble lagt til.

Undersøkte siden først. Fant Admin-panelet, leste seg så gjennom alt som sto der. kom over "Legg til Item", item ble addet.

Figure A.6: Test task 4: Add a product provided by the test leader

5.Find Arduino Uno without using search box. (time)

6 svar

n/a
15 sek
13 sec
52 sec
5 sek
10 sek

5.Find Arduino Uno without using search box. (what happend)

6 svar

finner ikke load more er ikke tydlig nok. Fant det vell å hoppe i kategorier
Klikker gjennom kategorier til den blir funnet.
Scrollet ned på items.
Går gjennom kategorie og finner det under development bords
"Load more" så fant testpersonen Arduinoen uten problem.
Gikk rett til "Load more" for å finne produktet.

Figure A.7: Test task 5: Find Arduino Uno without using search field



### Post interview

Did you feel you got an insight in what items MakerSpace got to offer?

6 svar

Det måtte læres. Er ikke helt intuitivt. Kategori bør henspille i menybiten. Tok lenger tid å forstod makerspace inventar. Hva skiller verktøy fra printer. Må gjøre kategorier klarere.

JA var mulig å få en insikt i tingene i makerspace.

Ja. Var lett å lese informasjon i boksene.

Nei, kun mye tekst og en må gå inn på ting for å forstå mer hva det er.

ja, hadde lite innsyn fra før, og viste ikke hva MakerSpace var. veldig praktisk å få testet hva et makerspace har.

nei. det var dårlig visuell representasjon av produktene. Dårlig indexer på produkter.

Did you feel the interface/site/design was easy to understand and figure out?

6 svar

Var ikke klart nok at kategorier var koblet til inventaret pga begrepet som brukes.

Finner siden lett å bruke og greit oppdelt.

Ja lett forstådd. Ryddig og enkelt.

Godt og oversiktlig desing

ja, måtte bare bruke litt tid til å bli vandt til siden først. det var en knapp for alt.

Middels, kan gjøres bedre, ikke værste han har vært borti.

Figure A.8: Post interview question 1 and 2: Insight on what items MakerSpace have to offer, And what the testers thought about the design

Do you have any suggestion to features that was missing or would improve the site?

6 svar

Må fokusere på at det er en ren inventar side. Hvis det skal være en større side for makerspacec må det innholde mer. eks prosjekter som er brukt. Personer som jobber der osv.

Gi søkerfunksjonen funksjonen muligheten at ved enter at itemen som var i forslag kommer i inventar bildet.

Virker greit slik det er nå.

Søkefelt tydeligere.

Tydeliggjør "Load more"-knappen. Eller ha alle produktboksene til å load med en gang. "Load more"-knappen forsvart da han letet etter det han skulle finne. den forsvant litt i bakgrunnen.

Bytte navn i boksene til bilder, eller bilder og navn. Ikke sikker om beskrivelse trengs på første visualisering. Rød på svart bakgrunn funker dårlig. Mørk bakgrunn funker dårlig. "Kategori"-baren var for liten. Dårlig valg på kategorinavn. samme grad presisjon på kategoriene. Feks Verktøy er vag kategori, hvor "Droner" er spesifikk.

Figure A.9: Post interview question 3: Do the testers have any suggestions to improvements

## Appendix B

### API Calls and Output

**Note:** Some text has been shortened down or changed to conserve space.

## B.1 Getting all items

### Client Message

GET -> <http://158.39.162.161/api/items/>  
No other information is needed from the client

### Reply from API

Showing first 2 JSON objects

**HTTP Status code:** 200 OK

```
1  [
2    {
3      "_id": "58ac0ace6bfd630ef83b550c",
4      "image_url": "https://www.website.com/image.jpg",
5      "item_name": "Lysdiode",
6      "__v": 1,
7      "quantity": 0,
8      "tags": [],
9      "categories": [
10     {
11       "_id": "58ac2d2abdc857062a24721c",
12       "__v": 0,
13       "category": {
14         "en": "Lights",
15         "no": "Lys"
16       }
17     }
18   ],
19   "description": {
20     "en": "LED-lights Colors: white, yellow, ...",
21     "no": "LED-lys Farger: hvit, gul, ..."
22   }
23 },
24 {
25   "_id": "58b4afe58e7a3974d0e6eacb",
26   "image_url": "https://www.website.com/image.jpg",
27   "item_name": "Ultimaker 2+",
28   "__v": 1,
```

```
29   "locale": {
30     "_id": "58e1968555014b6496c614b3",
31     "locale": "Makerspace",
32     "__v": 0
33   },
34   "quantity": 1,
35   "tags": [],
36   "categories": [
37     {
38       "_id": "58e38fdfee85df30572ed61d",
39       "__v": 0,
40       "category": {
41         "en": "3D Printer",
42         "no": "3D Printer"
43       }
44     }
45   ],
46   "description": {
47     "en": "The Ultimaker 3D printer, a ...",
48     "no": "Ultimaker 3D printer, en ..."
49   }
50 },
51 .....
```

## B.2 Get a single item

### Client Message

GET -> `http://158.39.162.161/api/items/<ITEM_ID>`

This call requires the items ID, if no ID is provided all items are returned. We want the item Ultimaker 2+ with ID: 58b4afe58e7a3974d0e6each, therefore our URL will look like this:

`http://158.39.162.161/api/items/58b4afe58e7a3974d0e6each`

### Reply from API

**HTTP Status code:** 200 OK

```
1 {
2   "_id": "58b4afe58e7a3974d0e6each",
3   "image_url": "https://www.website.com/image.jpg",
4   "item_name": "Ultimaker 2+",
5   "__v": 1,
6   "locale": {
7     "_id": "58e1968555014b6496c614b3",
8     "locale": "Makerspace",
9     "__v": 0
10  },
11  "quantity": 1,
12  "tags": [],
13  "categories": [
14    {
15      "_id": "58e38fdfee85df30572ed61d",
16      "__v": 0,
17      "category": {
18        "en": "3D Printer",
19        "no": "3D Printer"
20      }
21    }
22  ],
23  "description": {
24    "en": "The Ultimaker 3D printer, a ...",
25    "no": "Ultimaker 3D printer, en ..."
26  }
```

```
27 }
```

## B.3 Updating a item

### Client Message

To update a item the item ID *must* be provided in the URL, otherwise the API will return a error. As well as the ID being provided the whole object that is to be updated is to be provided with old and new information, as any missing details are assumed that they are to be deleted. Note that for the locale and categories sections only the ID is given. The API will accept both versions of this; only the ID or the whole object. It is only interested in the ID and will ignore the rest of the data.

PUT -> <http://158.39.162.161/api/items/58b4afe58e7a3974d0e6eacb>

```
1 {
2   "image_url": "https://www.website.com/image.jpg",
3   "item_name": "Ultimaker 2+",
4   "locale": {"58e1968555014b6496c614b3"},
5   "quantity": 1,
6   "categories": ["58e38fdfee85df30572ed61d"],
7   "description": {
8     "en": "The Ultimaker 3D printer, a ...",
9     "no": "Ultimaker 3D printer, en ..."
10  }
11 }
```

### Reply from API

**HTTP Status code:** 200 OK

As well as returning status code 200 the API will return the updated object. We choose not to include it here as it would look the same as above.

## B.4 Deleting a item

### Client Message

To delete a item the item ID must be provided in the URL, otherwise the API will return a error.

DEL -> `http://158.39.162.161/api/items/58b4afe58e7a3974d0e6each`

### Reply from API

**HTTP Status code:** 200 OK

```
1 {  
2   "message": "Item removed!"  
3 }
```



## B.5 Creating a item

### Client Message

To add a item to the API you have to at least provide a unique item name all other options are optional

POST -> <http://158.39.162.161/api/items/>

```
1 {  
2   "image_url": "https://www.website.com/image.jpg",  
3   "item_name": "Ultimaker 2+",  
4   "locale": ["58e1968555014b6496c614b3"],  
5   "quantity": 1,  
6   "categories": ["58e38fdfee85df30572ed61d"],  
7   "description": {  
8     "en": "The Ultimaker 3D printer, a ...",  
9     "no": "Ultimaker 3D printer, en ..."  
10  }  
11 }
```

### Reply from API

If the API found no other item with the same name it will return status code 201 and return the newly created item with its ID so the client can use it immediately without having to ask the API for all items to update itself.

**HTTP Status code: 201 Created**

```
1 {
2   "message": "item added!",
3   "data": {
4     "__v": 0,
5     "quantity": 1,
6     "image_url": "https://www.website.com/image.jpg",
7     "locale": "58e1968555014b6496c614b3",
8     "item_name": "Ultimaker 2+",
9     "_id": "5917607bee85df30572ed62e",
10    "categories": [
11      "58e38fdfee85df30572ed61d"
12    ],
13    "description": {
14      "en": "The Ultimaker 3D printer, a ...",
15      "no": "Ultimaker 3D printer, en ..."
16    }
17  }
18 }
```

## **Appendix C**

### **Project Contract**

## Prosjektkontrakt

Bacheloroppgaven  
Avdeling for Informasjonsteknologi  
Høgskolen i Østfold

Gruppe BO17-G14:  
Thomas Magelssen Bergby  
Nicolai Naglestad  
Espen Ottar Skjeggstad

Oppdragsgiver:  
Høgskolen i Østfold: Avdeling for Informasjonsteknologi  
Michael Andersen Lundsveen  
Espen Teigen

Veileder:  
Børre Stenseth

15. desember 2016

### Prosjektbeskrivelse

Studentgruppen skal utvikle et inventar- og utlånssystem for høyskolens MakerSpace. Formålet med dette er å gjøre det enklere for de ansatte på MakerSpace å holde styr på inventar til en hver tid, samtidig som et komplett inventarsystem vil hjelpe både studenter og ansatte å finne utstyr når ikke en studentassistent eller avdelingsingeniør er tilgjengelig. Systemet skal helst kunne vite hvor utstyret er på MakerSpace til en hver tid. Samtidig ønsker arbeidsgiver å ha et utlånssystem for utstyret som er tilknyttet MakerSpace.

Ved sluttet prosjekt skal det minst leveres en fungerende prototype av en webside med dummyinnhold hvor alt av MakerSpace inventar kan legges inn. Det er også ønskelig med et utlånssystem. Det skal gjøres et forsøk på å lage et fysisk system for å gjøre det enklere å finne utstyr på MakerSpace.

Prosjektet løses ved å se på andre systemer ved å analysere disse og se hva som fungerer og ikke fungerer. Det vil også gjøres intervjuer med studenter, ansatte og arbeidsgiver til å få forståelse på hva de forskjellige brukergruppene ønsker i dette systemet.

### Studentgruppen

Studentgruppen har selv det fulle ansvar når det gjelder organisering og gjennomføring av prosjektet. Ikke minst innebærer dette å følge opp tidsfrister. Kontakt med oppdragsgiver og veileder er også studentenes ansvar. De må også holde seg oppdatert på beskjeder og meldinger fra fagansvarlig (som regel pr. epost). Studentene skal selv fordele arbeidet i gruppa, og eventuelt velge prosjektleder (kan byttes på underveis hvis ønskelig).

Gruppen har rett til veiledning fra en eller flere av HiØ/IT sine fagansatte tilsvarende ca. en halv time pr. uke gjennom hele semesteret. Studentene skal også ha fri tilgang til program- og maskinvare og annet nødvendig utstyr. Er det behov for reiser og liknende, skal dette dekkes etter avtale med oppdragsgiver og/eller HiØ/IT.

### Oppdragsgiver

Det er viktig at oppdragsgiver er innforstått med rammene rundt bacheloroppgaven. Oppdragsgiver må ikke definere en oppgave som ellers ville bli utført av eksterne konsulenter o.l. Det vil alltid være en risiko for at studentene ikke kommer helt i mål, eller leverer noe som ikke nødvendigvis er ferdige, operative løsninger.

Oppdragsgiver skal gi rammene for prosjektet, dvs. hjelpe til med å definere formål, leveranser, og evt. metode. Det er svært viktig å gi studentene nok informasjon, og evt. tilgang til utstyr, programvare o.l.

## APPENDIX C. PROJECT CONTRACT

---

Oppdragsgiver må være forberedt på å svare på spørsmål på kort varsel, og sørge for å være oppdatert på prosjektets framdrift. Det kan også være aktuelt å hjelpe til med å revidere og endre prosjektplanen underveis.

Det er vanlig å ha regelmessige møter med gruppa, med en frekvens som passer prosjektets egenart.

### Veileder

Veileder skal hjelpe til med: 1) Gjennomføring og dokumentasjon, og 2) Faglige spørsmål (i de tilfellene der det faglige blir for smalt og spesielt, må studentene enten skaffe faglig hjelp fra oppdragsgiver, eller fra annet hold).

### Rettigheter

Deltagere med rettigheter i prosjektarbeidet er ideforfatter, utøvende medarbeidere og oppdragsgiver. Ideforfatter kan være veileder, en eller flere studenter, oppdragsgiver eller en kombinasjon av disse. Utøvende medarbeidere er studentene og i noen grad veileder. Oppdragsgiver kan enten være en ekstern institusjon/bedrift eller en avdeling ved Høgskolen i Østfold.

Følgende rettigheter er knyttet til prosjektdeltagelse:

**Opphavsrett** Rett til benevnelse ved publisasjon og eiendomsrett til originalt åndsverk (som sikres ved publisasjon). Opphavsrett innehas av ideforfatter og utøvende medarbeidere.

**Disposisjonsrett** Rett til videreutvikling av et avsluttet arbeid. Denne innehas av oppdragsgiver.

**Eiendomsrett** Rett til oppbevaring og omsetning av et avsluttet arbeid og innehas av oppdragsgiver.

Avvik fra disse prinsippene kan avtales før godkjenning av prosjekter etter ønske fra oppdragsgiver. Det skal da lages en avtale som inngås av alle deltagerne i prosjektet. Tvister om rettigheter avgjøres etter deltagerens samtykke av avdelingsstyret, subsidiært ved forfølgelse i det alminnelige rettsvesen.

Hvis oppdragsgiver ikke ønsker at resultatene eller deler av disse skal offentliggjøres, må dette avklares ved oppstart av prosjektet.

Underskrifter

Thomas Berg  
Nicola Masteghini  
Eugen

# **Appendix D**

## **Group Contract**

## Gruppekontrakt

Bacheloroppgaven  
Avdeling for Informasjonsteknologi  
Høgskolen i Østfold

Gruppe BO17-G14  
Thomas Magelssen Bergby  
Nicolai Naglestad  
Espen Ottar Skjeggstad

7. desember 2016



## APPENDIX D. GROUP CONTRACT

---

- § 1 **Demokrati** Alle avgjørelser skal baseres på flertallet i gruppa. Medlemmene bør likevel prøve å komme fram til fullstendig enighet.
- § 2 **Plikter** Det er møte- og forberedelsesplikt for alle. Ved sykdom eller annen gyldig fraværsgrunn skal ett eller flere av de andre medlemmene varsles så tidlig som mulig. Medlemmene forplikter seg også til å levere avtalt arbeid til rett tid, eller si fra i god tid hvis forsinkelser oppstår.
- § 3 **Møtetider** Møtetider bør avtales slik at ingen i gruppen hindres i forberedelser eller oppmøte.
- § 4 **Utgifter** Alle utgifter i prosjektet (reising, kopiering, inngangspenger, tidsskrifter etc.) skal deles likt mellom medlemmene. Utgiftene skal kunne dokumenteres.
- § 5 **Twister** Uenigheter og problemer bør bli behandlet og løst i plenum, internt i gruppen. Hvis dette slår feil, skal veileder kontaktes.
- § 6 **Arbeidsdeling** Alt arbeid skal deles så likt og rettferdig som mulig. Alle har rett til å påpeke forhold de mener er urettferdige. Totalt må hvert medlem bruke ca. 400 timer på prosjektet. Dette inkludert alt relatert arbeid, slik som f.eks. møter med oppdragsgiver og veileder. Det skal føres individuelle timelister som dokumenterer tidsbruken.
- § 7 **Samarbeid** I en gruppe gjelder en for alle, alle for en. Det er viktig at alle forsøker å dele tid, arbeid, erfaringer og kunnskap. Det er en selvfølge å hjelpe til hvis noen står fast.
- § 8 **Skjevfordeling** Ved en åpenbar skjevfordeling kan det være aktuelt å gi ulike karakterer i gruppen. Dette vil i så fall være etter gruppens eget ønske. Gruppen må fange opp slike problemer så raskt som mulig, og prøve å løse de umiddelbart, gjerne i samråd med veileder og evt. fagansvarlig.
- § 9 **Kontraktsbrudd** Alvorlige og/eller gjentatte regelbrudd kan føre til eksklusjon fra gruppen. Dette må i så fall skje i samråd med veileder og fagansvarlig.

Underskrifter:

Thomas Beggen  
Nicola Nysted  
Egon

## APPENDIX D. GROUP CONTRACT

---

- § 1 **Demokrati** Alle avgjørelser skal baseres på flertallet i gruppa. Medlemmene bør likevel prøve å komme fram til fullstendig enighet.
- § 2 **Plikter** Det er møte- og forberedelsesplikt for alle. Ved sykdom eller annen gyldig fraværsgrunn skal ett eller flere av de andre medlemmene varsles så tidlig som mulig. Medlemmene forplikter seg også til å levere avtalt arbeid til rett tid, eller si fra i god tid hvis forsinkelser oppstår.
- § 3 **Møtetider** Møtetider bør avtales slik at ingen i gruppen hindres i forberedelser eller oppmøte.
- § 4 **Utgifter** Alle utgifter i prosjektet (reising, kopiering, inngangspenger, tidsskrifter etc.) skal deles likt mellom medlemmene. Utgiftene skal kunne dokumenteres.
- § 5 **Twister** Uenigheter og problemer bør bli behandlet og løst i plenum, internt i gruppen. Hvis dette slår feil, skal veileder kontaktes.
- § 6 **Arbeidsdeling** Alt arbeid skal deles så likt og rettferdig som mulig. Alle har rett til å påpeke forhold de mener er urettferdige. Totalt må hvert medlem bruke ca. 400 timer på prosjektet. Dette inkludert alt relatert arbeid, slik som f.eks. møter med oppdragsgiver og veileder. Det skal føres individuelle timelister som dokumenterer tidsbruken.
- § 7 **Samarbeid** I en gruppe gjelder en for alle, alle for en. Det er viktig at alle forsøker å dele tid, arbeid, erfaringer og kunnskap. Det er en selvfølge å hjelpe til hvis noen står fast.
- § 8 **Skjevfordeling** Ved en åpenbar skjevfordeling kan det være aktuelt å gi ulike karakterer i gruppen. Dette vil i så fall være etter gruppens eget ønske. Gruppen må fange opp slike problemer så raskt som mulig, og prøve å løse de umiddelbart, gjerne i samråd med veileder og evt. fagansvarlig.
- § 9 **Kontraktsbrudd** Alvorlige og/eller gjentatte regelbrudd kan føre til eksklusjon fra gruppen. Dette må i så fall skje i samråd med veileder og fagansvarlig.

Underskrifter:

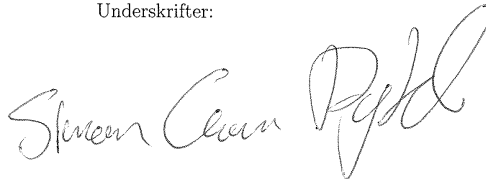
A handwritten signature in black ink, appearing to read 'Sivert C. Ryd', is written over a horizontal line.

Figure D.1: Additional signature from new member

## **Appendix E**

### **Confirmation of Group Change**

## APPENDIX E. CONFIRMATION OF GROUP CHANGE

### Bekreftelse på deltagende gruppearbeid

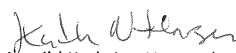
Dato: 10. Mai 2017

Sted: Halden

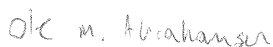
Simon Chen Dybvik byttet bachelorprosjekt i uke 10 2017. Vedkommende byttet fra gruppe **BO17-G22** Studieplan og kursplan for "Datasikkerhet" til **BO17-G14** MakerSpace Management System. Dette skjedde i samråd med veileder og deltagere i de to gruppene. I begge gruppeprosjektene var vedkommende aktiv, deltok på alle gruppeaktiviteter (gruppemøter, intervju, utflukter m.m.) og gjorde alle oppgaver han ble tildelt. Dette bekreftes av underskrifter av gruppedeltagere av begge grupper og muntlig godkjenning av veiledere.

Underskrifter:

#### BO17-G22

  
Ingvild Kathrine Nygaard-Hansen

  
Karina Rabben Jacobsen

  
Ole Martin Abrahamsen

#### BO17-G14

  
Nicolai Naglestad

  
Thomas Bergby

  
Espen Ottar Skjeggstad

  
Simon Chen Dybvik

## **Appendix F**

### **Meeting notes**

## **F.1 Meeting 24-1-17**

### **BO17-G14 Guidance meeting**

Minutes for January 24, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas

**Absent:**

#### **Reports**

The pre-report is delivered and approved

#### **Last meeting points**

1. Create pre-report
2. Create project contract

The minutes of the previous meeting were approved.

#### **New Business**

1. Define tools we are going to use
2. Have perimeter meeting with employer.

**Next Meeting:** Thursday, January 31, at 10:30

## **F.2 Meeting 31-1-17**

### **BO17-G14 Guidance meeting**

Minutes for January 31, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas

**Absent:** B. Thomas (*Travelling*)

#### **Reports**

Nothing notably to report.

#### **Last meeting points**

1. Create high fidelity wireframes

The minutes of the previous meeting were approved.

#### **New Business**

1. Create wireframes
2. Define work roles
3. Meeting with employer to discuss wireframes

**Next Meeting:** Thursday, February 07, at 10:30

## **F.3 Meeting 1-2-17**

### **BO17-G14 Guidance meeting**

Minutes for February 1, 2017

**Present:** T. Espen (Chair), S. Espen, N. Nicolai B. Thomas

#### **Reports**

1. Discussed simple wireframes
2. Not all items need a the amount displayed or counted.

#### **New Business**

1. Start on front and back-end design

**Next Meeting:** Tuesday, February 7, at 10:30



## F.4 Meeting 7-2-17

### BO17-G14 Guidance meeting

Minutes for February 07, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas

**Absent:**

#### Reports

##### *Webpage*

*Database* — We now have a server. MongoDB is now created and with some dummy-data.

*Webpage* — The HTML landing page is now created and shows dummy data and has a standard navigation menu that follows on all pages.

#### Last meeting points

1. check the webpage
2. Defining terms

The minutes of the previous meeting were approved.

#### New Business

1. ????
2. ????

**Next Meeting:** Thursday, February 14, at 10:30

## F.5 Meeting 14-2-17

### BO17-G14 Guidance meeting

Minutes for February 14, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas

**Absent:**

#### Reports

##### *Webpage*

*Database* — We now have a server. MongoDB is now created and with some dummy-data. It used Json files.

*HTML page* — The HTML landing page is now created and show dummy data and has a standard navigation menu that follow on all pages.

#### Last meeting points

1. Check the webpage
2. Defining terms

#### New Business

1. Start to fill out main report.
2. continue a prototype webpage so we can start user-testing.
3. Start to create a user-test.

**Next Meeting:** Thursday, February 21, at 10:30

## F.6 Meeting 21-2-17

### BO17-G14 Guidance meeting

Minutes for February 21, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas

**Absent:**

#### Reports

##### *Site*

*HTML page* —

- The page now speaks with the API
- Created item page and admin page

*Database* — Continud working on the API

- Fix database form by empty items.
- Fix models
- tarted working on authentication for API

##### *Main report*

Added some parts form pre-report to main-report. Gone through the main structure for the template. Checked existing hall of fame reports to se best practise for the main-report.

#### Last meeting points

1. Start to fill out main report.
2. continue a prototype webpage so we can start user-testing.
3. Start to create a user-test.

#### New Business

1. Make a design on the webpage to make it ready for user testing.
2. Create the structure and fill out what we can on the main report. Add discussion of why we chose to focus on a easy to update site rather then a heavy administrated site.

**Next Meeting:** Thursday, February 28, at 10:30

## F.7 Meeting 28-2-17

### BO17-G14 Guidance meeting

Minutes for February 28, 2017

**Present:** S. Børre (Chair), N. Nicolai, B. Thomas

**Absent:** S. Espen (*Travelling*)

#### Reports

##### *Website*

- We now have a detailed item view (not all info included)
- Search implemented (still testing)

#### Meeting discussions

##### *New group member*

We have been asked by a member of another bachelor group if he can join our group. We discuss this matter during the meeting, where our supervisor states that this decision is up to us te members of the group. Between now and next meeting we will make a decision if he will join our group or not.

##### *Website / System*

We discussed different aspects of the website, what it still needs and how we will solve different issues.

We discussed the following points that we need to implement on the website:

- Items
  - View (done)
  - Item out of stock / messaging system
  - New items (semi done) / Edit items
  - Tags on item page and in search
- Messaging system
  - Item out of stock
  - Loaned item
  - General messages
  - Need assistance

- Box location

## F.8 Meeting 28-2-17

- Users
  - Unauthenticated
    - \* View items
    - \* Send messages
  - Authenticated
    - \* Same as Unauthenticated
    - \* CRUD items
  - Admin
    - \* Same as Authenticated
    - \* CRUD news
    - \* CRUD users
- Login System

Here it was discussed if we need a complicated login system as the majority of users on the system are unauthenticated users.

### *Report*

The deadline for the report is March 9, but our supervisor states that this date is not that important as we can review the report every meeting.

### Until next meeting

1. Continue work on website, to prepare it for user testing
2. Continued work on the report.

**Next Meeting:** Tuesday, March 7, 10:30

## **F.9 Meeting 14-3-17**

### **BO17-G14 Guidance meeting**

Minutes for March 14, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas D. Simon

#### **Reports**

New member is now in the group and active in all the tools we are using.

#### **Last meeting points**

1. Continue work on website, to prepare it for user testing
2. Continued work on the report. The minutes of the previous meeting were approved.

#### **New Business**

1. Work more on report. Especially Analysis and implementation
2. Remove most of template fill that is not needed in the main report.
3. Add part about why we changed the direction of the system from high bureaucracy to low bureaucracy.

**Next Meeting:** Thursday, March 21, at 10:30

## **F.10 Meeting 21-3-17**

### **BO17-G14 Guidance meeting**

Minutes for March 21, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas D. Simon

#### **Last meeting points**

1. Work more on report. Especially Analysis and implementation
2. Make sure that the new report always is ready for Mondays.
3. Make sure that all task has defined owner and time limit.

#### **Reports**

First version of main report i delivered.

#### ***User-testing***

User testing is done and noted.

#### **New Business**

1. Summarize testing result.
2. continue to work on report.

**Next Meeting:** Tuesday, April 04, at 10:30

## **F.11 Meeting 4-4-17**

### **BO17-G14 Guidance meeting**

Minutes for April 4, 2017

**Present:** S. Børre (Chair), S. Espen, N. Nicolai B. Thomas D. Simon

#### **Last meeting points**

1. Summarize testing result.
2. continue to work on report.

#### **Reports**

##### **New Business**

1. Add refferens to MongoDB.
2. Consider removing Latex figure.
3. Move wireframe figures to right spot.
4. change chapter 2 too glossary.
5. add implementations possess.
6. explain the difference from site layout one to two. Why we changed it.
7. add test evaluation under test chapter.
8. make sure that the reason to change the direction to a simpler design is clear.

**Next Meeting:** Wednesday, April 26, at 10:30



## **F.12 Meeting 26-4-17**

### **BO17-G14 Guidance meeting**

Minutes for Wednesday, April 26, at 10:30

**Present:** S. Børre (Chair), S. Espen, B. Thomas, D. Simon

**Absent:** N. Nicolai

#### **Last meeting points**

1. Add references to MongoDB.
2. Consider removing Latex figure.
3. Move wireframe figures to right spot.
4. change chapter 2 too glossary.
5. add implementations possess.
6. explain the difference from site layout one to two. Why we changed it.
7. add test evaluation under test chapter.
8. make sure that the reason to change the direction to a simpler design is clear.

#### **Reports**

1. Added implementations and incremental versions.
2. added more to design chapter.
3. changed chapter 2 to glossary.

#### **New Business**

1. Discussion about why and how the system got simplified for the use in MakerSpace.
2. Make sure that in the text that it's clearly state that Simon came in later in the project.
3. Add and expand user-story for all roles that the system will have.
4. Describe the functionality we have in the system.
5. Remove 5 second test since it was not used.