

Portal for test automation

- Developed for the test department at IKEA IT Helsingborg



LUND UNIVERSITY
Campus Helsingborg

LTH School of Engineering at Campus Helsingborg
Department of Computer Science

Bachelor thesis:
Irfan Agovic
Faruk El-Zoubi

© Copyright Irfan Agovic, Faruk El-Zoubi

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Lunds universitet
Lund 2014

Abstract

Environments with a high number of systems that are dependent on each other create complexity in the testing of each system. For a test to perform correctly, every dependency has to be fully functional. If this condition is not met, the test will fail and measures need to be taken into the systems that are causing the problem before it can be repeated again. This is a problem that the test department at IKEA IT in Helsingborg are experiencing.

This bachelor thesis examines the needs of IKEA IT:s existing test environment through a qualitative research and a phenomenological data analysis. Seven interviews were conducted where a wireframe prototype of the portal was presented together with a questionnaire. A small-scale quantitative study based on the questionnaires was also conducted, from which it was decided what test tool used in IKEA IT the portal should be formed for. From the analysis, a system requirement specification and final wireframe prototype were created and used as a guideline for the implementation.

Using a process tool from the Agile family, modified to fit the needs of the developers, the implementation was carried out according to a “Scrumish” development method. In order to work with the process tool in the most effective way, the functional requirements of the product were created as user stories that were broken down into tasks for the developers to work with in parallel during the implementation. The implementation was done in a completely open source based development environment, using the Durandal framework on the client side and Node on the server side. Results of the implementation are presented in a demonstration of the product with descriptions and motivations to each functionality.

The portal serves as a solution to their problem by offering the possibility to monitor all systems in their test environment. In this way, testers can make sure that all dependencies of a system are fully functional before entering a test phase, saving the company both time and resources. In addition to this, it supports test automation with the test results presented on the portal, making it simple for testers to evaluate their work and keep it gathered in one place. The portal is integrated with ALM (Application Lifecycle Management), the most commonly used test tool in IKEA IT, and was implemented using JavaScript, HTML and CSS.

Keywords: test automation, IKEA IT, dependencies, open source, phenomenological, Durandal

Sammanfattning

Miljöer med ett högt antal system som är beroende av varandra skapar komplexitet i testandet av varje enskilt system. För att ett test ska utföras korrekt måste varje beroende vara fullt funktionellt. Om inte detta villkor uppnås kommer testet att misslyckas och åtgärder behöver vidtas i de system som orsakar problemet innan det kan köras om på nytt. Detta är ett problem som testavdelningen på IKEA IT i Helsingborg upplever.

Detta examensarbete utforskar behoven av IKEA IT:s nuvarande testmiljö genom en kvalitativ undersökning och en fenomenologisk data analys. Sju intervjuer genomfördes där en wireframe prototyp av portalen presenterades tillsammans med ett frågeformulär. En småskalig kvantitativ studie baserad på frågeformulären genomfördes även, från vilken det beslutades utifrån vilket testverktyg som används i IKEA IT som portalen skulle utformas. Utifrån analysen skapades en kravspecifikation för systemet samt en slutgiltig wireframe prototyp, vilka användes som en riktlinje för implementationen.

Med hjälp av ett processverktyg från Agile-familjen, modifierat för att passa behoven hos utvecklarna, utfördes implementationen enligt en ”Scrumish” utvecklingsmetodik. För att arbeta med processverktyget på det mest effektiva sättet skapades de funktionella kraven på produkten som user stories, vilka delades upp i tasks för utvecklarna att arbeta med parallellt under implementationen. Implementationen utfördes i en fullständigt öppen källkod-baserad utvecklingsmiljö, genom att använda Durandal frameworket på klientsidan och Node på serversidan. Resultaten av implementationen presenteras i en demonstration av produkten med beskrivningar och motiveringar till varje funktionalitet.

Portalen fungerar som en lösning på deras problem genom att möjliggöra övervakning av alla system i deras testmiljö. På så sätt kan testarna säkerställa att alla beroenden för ett system är fullt funktionella innan en testfas påbörjas, vilket sparar företaget både tid och resurser. Utöver detta stödjer den testautomation där testresultaten presenteras i portalen, vilket gör det enkelt för testare att utvärdera sitt arbete samt hålla det samlat på en plats. Portalen är integrerad med ALM (Application Lifecycle Management), det mest använda testverktyget inom IKEA IT, och implementerades med JavaScript, HTML och CSS.

Nyckelord: testautomation, IKEA IT, beroenden, öppen källkod, fenomenologisk, Durandal

Foreword

First of all we would like to thank IKEA IT for providing us the opportunity to conduct this thesis. Special thanks go out to the owner of this thesis, Anna Gamalielsson - Infrastructure Manager of IKEA IT:s test department of Helsingborg, and our supervisor Olof Ernstsson - Product Specialist of IKEA IT:s test department of Helsingborg. They have both been inspiring, extremely helpful and available during the whole thesis.

A further thanks goes out to the supervisor and examiner from LTH, Campus Helsingborg who have been giving out helpful feedback regarding the academic structure and know-how in during the thesis.

This thesis has been a fun and educational experience at the same time. We have had the chance to work with professionals in the business community and get a feel of how it is to be in a real working environment, and at the same time apply our theoretical knowledge to practical use. What was most educational was to perform an analysis from a research methodology which has been a whole new experience and also to learn a completely new development environment with multiple programming languages we have never worked with earlier.

We hope that our solution to the needs of IKEA IT will be further developed to include even more benefits for the end users and that the user base for the product will grow with time.

Irfan Agovic & Faruk El-Zoubi

List of contents

1 Introduction	1
1.1 Background	1
1.2 Scope	1
1.3 Limitations	2
1.4 More about IKEA IT	2
2 Technical background	4
2.1 JavaScript	4
2.2 Client	4
2.2.1 HTML5/CSS3	4
2.2.2 Durandal	5
2.2.2.1 RequireJS, jQuery & Knockout	5
2.2.2.2 MVVM and Knockout	5
2.3 Server	6
2.3.1 Node	6
2.3.2 MongoDB	6
2.4 Git	6
2.5 ALM	6
2.6 WebEx	7
3 Method	8
3.1 Analysis	8
3.1.1 Research method	8
3.1.2 Approach	9
3.1.3 Selection of stakeholders	10
3.1.4 Data collection and analysis	10
3.1.5 Ethical consideration	11
3.1.6 Discussion of research method	11
3.2 Implementation	12
3.2.1 Kanban versus Scrum	12
3.2.2 Choosing a Scrumish process tool	13
3.2.3 Approach	13
3.2.4 Discussion of process tool	16
3.3 Sources	17
3.3.1 Literature	17
3.3.2 Internet sources	17
3.3.3 Supervisor and test department at IKEA IT, Helsingborg	18
4 Analysis results	19
4.1 Fundamental functionalities	19
4.1.1 Overview	19
4.1.2 Test History	20
4.1.3 Scheduler	21
4.1.4 Dependency	21
4.1.5 Login & Home screen	22
4.2 Report on interviews	22
4.2.1 General information	22
4.2.2 Overview	23
4.2.3 Test History	23
4.2.4 Scheduler	24

4.2.5 Dependency	24
4.2.6 New functionalities	25
4.3 Chronological order of the wireframe	26
4.3.1 New features of version 3.0.....	26
4.3.2 New features of version 4.0.....	31
4.3.3 New features of version 4.1.....	34
4.4 System requirement specification	36
5 Implementation results	37
5.1 What was achieved?	37
5.2 What was not achieved?.....	38
5.3 How is the product implemented?.....	38
5.4 Demonstration of the product.....	42
6 Conclusion.....	52
6.1 Problem description.....	52
6.2 The scope.....	52
7 Further work	53
8 Terminology.....	54
9 References	55
Appendix A – Wireframe 2.0	57
Appendix B – Wireframe 4.1	69
Appendix C – System requirement specification	94
Appendix D – Questionnaire	108

1 Introduction

Testing of computer systems is becoming more complex due to the fact that computer systems are growing more complex with time. In environments consisting of multiple systems, consideration has to be taken into systems that have dependencies. When it comes to testing, this can be critical since the health of dependent systems can affect the testing.

This chapter presents the background, scope and limitations of this thesis and furthermore the company behind the thesis.

1.1 Background

In order to test a system within IKEA IT, a time slot in the test environment is reserved for testing. It is critical that the dependencies of the system to be tested are fully functional during this period of time, since a consequence of a failure in one or more dependencies is that the test becomes invalid. Hence the test has to be ran once again without any dependent systems having any failures. This is a problem that costs the company both time and money. Therefore there is a need for a test environment with an overall monitoring of the systems installed in it, which will serve as a solution to the problem. Additionally, the client wants the test environment to support scheduling of automated tests of which the test results are presented in a simple and comprehensible way.

1.2 Scope

Thus, the scope of this bachelor thesis is to:

1. Identify the needs of the existing test environment through an analysis.
2. Develop a portal which will complement the needs of the existing test environment and also be based on following main requirements.
 - a. Visualize dependencies for each system in the portal.
 - b. Support test automation for each system in the portal.

At first the problem description of the thesis was:

1. What tools are today used in IKEA IT regarding test automation?
2. Where are the test results and test scripts stored and in which way are the test results used for evaluation?
3. Does the portal contain advantages for the end users in comparison to the currently available tools?
4. Which advantages does the portal offer in comparison to the currently available tools?

During the analysis phase of the thesis, multiple questions to the stakeholders were thought of during a number of brainstorming meetings. The first and second problem description together with these new questions was used as a questionnaire. Therefore they were no

longer seen as problem descriptions but rather as survey questions. Due to these circumstances the problem description is decreased to the following:

1. Does the portal contain advantages for the end users in comparison to the currently available tools?
2. Which advantages does the portal offer in comparison to the currently available tools?

1.3 Limitations

In the original scope of this thesis, it was thought by the client that the implementation would consist of three parts; a shell for the portal (graphical interface as well as server calls), a scheduler which would schedule test scripts for automated execution and a machine that would perform the actual testing. Due to a limited time frame for this thesis and more specifically the implementation phase, the scope was narrowed down to only the first part after discussions with the client. Furthermore, the second scheduler was dependent on the machine, which was not implemented. Therefore the other parts could not be implemented unless everything was done, which there was not enough time for.

The implementation was done in a framework and several programming languages which the authors of this thesis had little or no knowledge about. This affected the implementation in the way that there was less time dedicated to implementing the system, since time had to be spent on learning how to work in the development environment. Additionally, information about the framework was limited comparing to a widely used programming language such as Java since the chosen framework is fairly new, which increased the cost in time for adapting to the development environment.

1.4 More about IKEA IT

IKEA is a Swedish furniture company that was founded in 1943 by Ingvar Kamprad and have since then grown into one of the largest home furnishing companies in the global market with 303 stores in 26 countries and 135,000 employees (year 2013). Since 1982, IKEA is owned by a foundation in the Netherlands, which means that profits are reinvested, used for charitable purposes or saved as a financial reserve. In this way, IKEA is a non-profit organization. IKEA is further divided into several different companies, including IKEA Industry, Stichting IKEA Foundation and IKEA IT. (Företagsinformation - IKEA n.d.)

This thesis has been conducted within the IKEA IT company that provides support, maintenance and development in both new and existing applications in the entire IKEA corporation. IKEA does no longer develop their own applications; instead the development has been outsourced to external consultants. These consultants are yet dependent on IKEA IT to deliver a test platform before releasing it to the public. It is in this part where the testing department within IKEA IT comes in to the picture. They maintain and provide the test environment in which the developers can test their product. This thesis has been dedicated to the test department within IKEA IT. (This section has been validated by the supervisor from IKEA IT)

2 Technical background

This chapter is devoted to giving a background of the languages and framework in which the implementation of this thesis' work is done. The product that was developed is a client-server application meaning that it is capable of providing a service and handling a client request. This chapter explains the technical background to the implementation of the portal separately in two parts, client side and server side, as well as other software relevant to thesis.

2.1 JavaScript

JavaScript is an object scripting programming language originally developed by Netscape. The scripting is performed dynamically where some of the abilities include runtime object construction, function variables and source code recovery. Objects in JavaScript are created at run time by the use of empty objects, simply adding functions and properties to them. Therefore they do not have to be predefined as classes. The use of JavaScript is dominated in web browsers; in web pages and server applications. It is also commonly used for developing web servers that handle HTTP request and response objects. In combination with a client side developed in JavaScript, this type of objects can be used for generating web pages dynamically. (Mozilla Developer Network and individual contributors 2013)

That is how JavaScript has been used in this thesis work. All functionality in both the client and server side is fully developed in JavaScript, using the Durandal framework which is explained in chapter 2.2.2.

2.2 Client

The client side of the portal handles user interaction. It has been implemented using several tools for different purposes which will be presented in this sub-chapter.

2.2.1 HTML5/CSS3

HTML is short for HyperText Markup Language and is a standard in creating web pages today (Web Education Community Group 2013). HTML5 was and is still being developed to replace HTML4.01 which dates back to 1999. In HTML5 the main focus was to give *“rich content without the need for additional plugins”* - (HTML5 Introduction n.d.). The group working with the development of HTML5 includes AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera and many others. (HTML5 Introduction n.d.)

This new version gives multiple new features such as New Elements, New Attributes and Full CSS3 Support. New elements include nav, video, audio and others. The purpose of nav is to identify a navigation part of the HTML document, while video and audio have been developed to handle multimedia content on the HTML document. (Pieters 2013)

HTML is used to define parts of a document, for example a title with the title tag (<title>This is a title</title>). It is not meant to be dedicated to formatting the document such as color and margins. The reason is simple; it would mean that a developer must specify the format in every page it creates. The solution was to instead use CSS which is short for Cascading Style Sheets and have been the solution since HTML4. CSS lets the

developer set a format for all of its pages, when needed to change one only needs to change the CSS file instead of every HTML document. (CSS Introduction n.d.). The latest version of CSS is CSS3 which is still under development but also in use. CSS3 has split its attributes into modules and is backwards-compatible. Some modules contain the attributes of the earlier CSS versions while new ones are being added. Some of the new models are Selectors, Box Model, Backgrounds, Borders and others. (CSS3 Introduction n.d.)

2.2.2 Durandal

Durandal is a framework that has been developed by Blue Spire. It combines a collection of different JavaScript libraries such as RequireJS, jQuery and Knockout, creating a tool for developing Single Page Application web pages (Home | Durandal n.d.). Durandal itself is a JavaScript library that has support for a variety of ways to design front ends such as MVP (Model-View-Presenter), MVC (Model-View-Controller) and MVVM (Model-View-ViewModel) (Get Started | Durandal n.d.). In this thesis, the portal is developed according to MVVM.

2.2.2.1 *RequireJS, jQuery & Knockout*

RequireJS serves as a JavaScript module as well as file loader (RequireJS n.d.) and is used on both the client and server side of the portal. *jQuery* is used to simplify several web programming features like Ajax calls, finding in and altering HTML documents and event handling as well as animation in HTML documents (jQuery n.d.). *Knockout* is used when developing a user interface that needs to be updated dynamically. It is built on the MVVM architecture and binds data from the model to the view, making the user interface update as data is modified (Knockout: Introduction n.d.). These are all open source JavaScript libraries.

2.2.2.2 *MVVM and Knockout*

MVVM is a method of constructing user interfaces and is built on three components:

1. Model
2. View model
3. View

Model

A model consists of stored data that is used in the application and is not affiliated with the user interface. This data is kept externally, such as in a database. With Knockout, the model can be reached by server calls to be used in the user interface, which is the usual way of getting stored data.

View model

A view model comprises code of all functionality and data that is used in the user interface. However, it does not contain any graphical components. The view models are entirely written in JavaScript when using Knockout.

View

A view is a graphical user interface which can be interacted with by the user. When in the use of Knockout, the view consists of a HTML document that is linked to the view model through bindings. It is updating the user interface as attributes in the view model are modified, presenting data from the view model and signaling the view model to execute operations. (Knockout: Observables n.d.)

The interaction of these components is done by creating data and functionality for each page in the view model, in which the stored data is retrieved from and saved to a model, creating the graphical interface in the view and connecting them with an attribute in Knockout called "*data-bind*". A demonstration of this is seen in chapter 5.3.

2.3 Server

All the information in the portal is stored in a database called MongoDB. The server side is written in Node, and database calls are made with a plugin called Mongoose.

2.3.1 Node

Node is built on Chrome's JavaScript runtime and is a platform that is designed for creating fast and expandable networks. It rarely performs any I/O operations and does not allocate any memory for threads per user that connects to the server, making it fast and flexible. (node.js n.d.)

2.3.2 MongoDB

MongoDB is an open source based database. Opposed to relation databases such as MySQL, Mongo is document oriented, meaning that the information which the database handles is in the form of objects. These objects are called JSON (JavaScript Object Notation) documents. (MongoDB n.d.)

2.4 Git

Git is an open source project and is a tool used by developers working in teams. Its purpose is to handle versions of systems that are under development. There are tools similar to Git, but what makes Git outstanding is a property called branching. The concept of branching is that a developer creates a copy of the system which is completely independent. No modifications in the branch affect the system until the branch is merged with the rest of the project. (About - Git n.d.)

2.5 ALM

ALM or Application Lifecycle Management is a multifunctional tool that has been created for the purpose of making it easier for a development team to keep control of the lifecycle for an application that is being developed, as well as speeding up the development (Application Lifecycle Management (ALM) n.d.). HP (Hewlett-Packard) is the developer of this software, which is used by testers at IKEA IT for testing application functionalities (among other things).

2.6 WebEx

WebEx is a program that is used for online meetings, developed by Cisco. It is used at IKEA IT for virtual conferences and has been used by the authors of this thesis in some of the interviews that were conducted.

3 Method

This chapter addresses how this thesis was conducted and is split into two main parts, the analysis and the implementation. Furthermore, the chapters are presented in chronological order, with the first part being the analysis and the second part being the implementation.

3.1 Analysis

There was no given system requirement specification, desired architecture nor any other kind of document or knowledge on which functions should be included, how the navigation should work or how the graphical user interface should look like at the beginning of this thesis. Therefore there was a need of conducting an analysis with the aim of gathering information regarding functionality, navigation and the graphical user interface.

This chapter presents the research methodology applied in this study. The chosen method chapter raises the type of research method applied in the analysis part. Furthermore, the chapter highlights the approach to the chosen method, selection of stakeholders, data collection, ethical considerations and methodology discussion of the study.

3.1.1 Research method

In order to choose a method which responds to the needs of a study to be conducted, one must first understand the difference between the available methods. In this thesis, the choice was between a quantitative research method and a qualitative research method.

Qualitative research is not fit for general studies as these examine a limited target group, i.e. it has a strictly limited participant base. The study is done with open questions and provides more space for the participants to express themselves. This means that the study is done with a small number of participants, but each participant is explored deeper and hence the data becomes more abundant. In a qualitative research, it is up to the researcher to interpret the findings. (Justesen & Mik-Meyer 2012) Quantitative research aims to investigate a market or portion of society by providing a study of a broad participatory base. The study's contents are usually closed questions to collect numerical data and quantify these. The results are then to be explained statistically. The goal is to give a description of social conditions or explanation of the conditions in relation to each other. It gives a general idea of the investigated phenomenon. (Justesen & Mik-Meyer 2012)

In order to make a choice of the available methods some fundamentals were laid. The product to be developed was not going to be published for an open market; instead it was exclusively going to be produced for internal usage within the company. Therefore the participants of the study were limited to employees of the company. Furthermore, there was a need of understanding the needs of the stakeholders, which means that experience, feelings and thoughts are the main data to be taken into consideration and studied.

The fundamentals described above are best suited with the qualitative research method. Since the qualitative method gives room for deeper research of each participant, it meets the need of exploring the stakeholders experience, feelings and thoughts. Additionally, the system to be developed was not a market product and thus it would not be feasible to use a quantitative study. Hence the choice of the qualitative research method. It did not necessarily mean that the qualitative research was used exclusively. In one moment of the

analysis there was an instance of quantitative research used, which is explained in chapter 3.1.4.

3.1.2 Approach

An investigation of different types of survey methods was conducted in order to decide which one was the most fitting to meet the needs of the thesis analysis. To do that, the purpose of the study was taken into consideration. As mentioned in chapter 3.1.1, the purpose of the study was to gain greater understanding of the stakeholders' needs and thoughts of the product to be produced. For this reason, interviews with stakeholders were the best choice for the study. (Robson 2011)

The qualitative methodology offers the possibility of several different methods of interviews, including unstructured and semi-structured interviews. In order to give the participants the opportunity to give as much feedback as they wanted and at the same time be able to ask direct questions and follow up questions, it was decided that the most fitting kind of interview was the semi-structured interview. (Robson 2011) Furthermore, it was decided that the interviews would also be of the respondent interview type. Meaning that participants were guided by the interviewers, although room for stakeholders to freely express their thoughts and ideas was given. (Robson 2011) The motivation behind having a respondent interview was that the participant might need guidance since the portal would be an unknown phenomenon requiring some explanation.

It was not possible to interview the participants all at once or in groups due to the fact of participants working in different cities and in one case another country. A group interview was not possible because of this. Instead, every participant was interviewed separately. In total seven interviews were conducted, in which two were performed over a WebEx meeting (see chapter 2.6 regarding WebEx) and the remaining five were performed with both parties physically present in the same room.

In order to conduct the interviews some preparations were needed to be made. Some questions had already been formed and could be put in a questionnaire which can be seen in appendix D. Furthermore there was a need to visualize the already existing ideas regarding functions and navigation of the portal. After some research on the internet the prototyping term wireframe was found. This was chosen since it would display the navigation and results of functionalities within the portal and therefore be easier to explain than with just plain words and without visualization.

During every interview, one of the authors had the role as the interviewer and the other as the secretary. These roles were switched between every interview in order to divide the workload fairly. Each interview began with the interviewer introducing himself and the secretary and then explaining that the interview regarded a bachelor thesis and what the thesis was about. After the introduction the participant was asked to fill out a questionnaire (see Appendix D) consisting of seven questions regarding their current profession. When this was completed the interviewer began going through and explaining the wireframe prototype (see Appendix A). The presentation of the wireframe prototype was focused on the structure, navigation and functionality. The participant was free to ask questions and make statements during and after the explanation of the wireframe prototype. The interviewer could then reply with a follow up question or explanation. If the participant did not react to certain vital parts of the wireframe prototype, the interviewer brought this part

to the participant's attention. This is in accordance with the chosen research method, qualitative semi-structured respondent interview.

The vital parts consisted of the following questions:

- Should you be able to be invisible to other systems in the portal?
- Is it interesting to show others that you are dependent on their system?
- Should you be able to see the secondary dependencies of a project?
- Do you want to be able to see which systems / projects that are dependent on your system / project?

3.1.3 Selection of stakeholders

The stakeholders that were chosen for interviews were handpicked by the client who has a deep insight into the test department at IKEA IT. Therefore, he could decide which stakeholders were most appropriate for the analysis. For that reason, the responsibility of selecting the stakeholders was completely on the client. Furthermore, his knowledge and insight in the company made it a natural choice to let him choose the people to be interviewed. The result was a broad selection of stakeholders with different professions at different parts of the test department.

The different stakeholders were following (in chronological order):

1. IT Solution Analyst
2. Global Quality and Test Manager Group HR
3. Test Architect
4. Test Manager
5. Product Specialist
6. Service Owner
7. Oracle Database Administrator
8. Did not respond

One of the stakeholders invited by the client to participate did not respond within the time frame to be a part of the study. Therefore seven out of eight invited people participated in the study.

3.1.4 Data collection and analysis

At the beginning of every interview, the participant was asked for permission to audio record as well as writing down digital notes during the interview. Every stakeholder agreed to this. The audio recording was done with two smart phones since those were the only tools available for audio recording. When the last interview was done, all the audio recordings were transcribed. The transcriptions were then gathered with the notes from the interviews respectively so that all data was kept at the same place.

The goal of the analysis was to gather information about how the stakeholders felt or thought about certain functions of the portal that were shown to them. Also to collect their own thoughts and ideas for how the portal should be implemented based on their own experiences with testing. The data was therefore primarily collected and analyzed according to a phenomenological data analysis and not a quantitative data collection (although that was also performed in a small manner). The data regarding thoughts and feelings about

diverse functions of the portal was collected respectively from every transcription. It was then categorized based on thoughts of the same function or functions very similar to each other, for the reason to keep statistics regarding how many stakeholders mentioned a certain functionality. This led to also having a quantitative data collection, although it was not the main goal of the analysis. The reason why a quantitative data collection was also made was to be able to see the need for certain functionalities in a concrete manner. This made it easier to decide on what was to be implemented and also which parts of the portal were the most important ones.

3.1.5 Ethical consideration

In order to protect the participants' privacy and make them feel more comfortable doing the interviews, no personal information was provided about them except for their professions and workplaces. Instead of representing the different stakeholders by their names, it was done by an identification number. The participants were informed about this at the beginning of every meeting so that they were aware of it.

3.1.6 Discussion of research method

In retrospect, the research method used during the analysis phase was one that gave many needed results which can be seen in more detail in chapter 4. The results were also validated through the client who gave a verbal approval, meaning that the client thought the results were resourceful and relevant.

However, the study is limited to the stakeholders that have been interviewed which are a slim number of seven people. This gives a small representation to rely on and a quantitative research would instead give a much bigger representation. However, the product is (as mentioned in chapter 3.1.1) not focused on a massive user group, but is instead created for internal usage which is why a quantitative research is not as well suited as a qualitative. Also, as mentioned in chapter 3.1.3, the stakeholders were chosen by the client, who did not have the chance to gather more than seven participants. Although more participants would have been beneficial if there was an opportunity for it.

One must consider that neither one of the stakeholders had the same profession, which gives a broad spectrum. But it misses the opportunity to compare the thoughts of multiple stakeholders with the same profession, or in other words the similar experience and knowledge.

During the study there was no consideration to relations of different thoughts or expressions. Meaning that there was no study on which part of the system was most important in relation to another part of the system. This is a lack of information which could or could not change the course of implementation, but it is not possible to know since it was never conducted.

3.2 Implementation

From the analysis phase of the thesis the basic fundamentals for implementation was laid. These being a finalized requirement specification (see Appendix C) and wireframe (see Appendix B) from which an implementation could be based upon.

This chapter presents the development process tool applied in this study. Chapter 3.2.2 raises the type of process tool used during the implementation part of the thesis. Furthermore, the chapter highlights the approach to the chosen process tool and discussion of process tool.

3.2.1 Kanban versus Scrum

Before any implementation was made a development process tool was to be selected. The choices were between Kanban and Scrum, which were derived from the authors' experience of the course EDT 655 - Project grade 3, in LTH, Campus Helsingborg. In order to pick one of the mentioned process tools, a study of them was conducted and later on, a discussion was made resulting in the selection of Scrum.

There are multiple common attributes of Kanban and Scrum since both are Lean and Agile (Kniberg & Skarin 2010). The following list gives a perspective of the common attributes (Kniberg & Skarin 2010).

- Pull scheduling instead of push scheduling.
- Limit of the current work in progress.
- Transparency to drive process improvement.
- Focus on delivering releasable software early and often.
- Based on self-organizing teams.
- Require breaking the work into pieces.
- The release plan is continuously optimized based on empirical data (velocity/lead time).

Both process tools are adaptive, meaning they have few rules or processes compared to other more traditional process tools. Although when comparing Scrum and Kanban, Scrum is more prescriptive, meaning it has more rules than Kanban (Kniberg & Skarin 2010). The main differences are the boards, the sprint in Scrum and the roles (Kniberg & Skarin 2010).

The boards are different in the way that Scrum resets the board after every sprint and Kanban's board is persistent. In this way a board in Scrum belongs to the sprint in which the board has been setup, while in Kanban the same board is used through the whole project (Kniberg & Skarin 2010). Another difference in the boards is that Kanban limits the items that can be in a certain state. For example, if WIP (work in progress) limit is set to 2 in "ongoing", then there can only be 2 items in the ongoing column. The Scrum board does not limit how many items can be in a certain state, instead it limits how many items can be in the board during a period.

"So in Scrum WIP is limited per unit of time. In Kanban WIP is limited per workflow state." (Kniberg & Skarin 2010, s32).

In Kanban there is no fixed time frame where a certain work is to be done. Instead, the team will continuously drag items when it feels it can do so. However, a main part of Scrum is the sprint. A sprint is a fixed time frame containing a certain amount of User stories. The

amount fitted in a sprint is based on the how many items can be done within the time frame, which is estimated by the developing team (Kniberg & Skarin 2010).

Roles are not prescribed within Kanban, although it does not mean it is wrong to use roles. In Scrum, roles are prescribed to include a product owner, team and a scrum master.

The product owner is the client of the product who will decide the goal of the project and the priorities of the backlog.

The team is the whole group of individuals which will implement the product.

The scrum master is kind of a project manager who clears any obstacles and provides process leadership (Kniberg & Skarin 2010).

3.2.2 Choosing a Scrumish process tool

Before going in to the implementation of the project, a request was made by the client. The client wanted to have periodic reports of what was implemented and be able to view the results of the implementation. The client also wanted insight of what was completed and next in line for implementation. This leaned at establishing a prioritized backlog, product owner role and fixed time for release events, which paved the way for choosing Scrum. An important note is that not every attribute within Scrum was used, which means that a “Scrumish” process tool was used and not Scrum. This is further explained in chapter 3.2.3.

Another contributing factor to choosing Scrum as a process tool had to do with the authors’ earlier experience with process tools. Both authors had gained experience within Scrum from the course EDT 655 - Project grade 3, in LTH, Campus Helsingborg. Therefore, both were in agreement that it would be easier and faster to work with Scrum instead of Kanban.

Before deciding to go ahead with Scrum a meeting was set with the client. During the meeting the client was informed in which way the team would work and how the client as a product owner would be able to participate. When the client agreed to the process tool the decision was finalized.

3.2.3 Approach

Before the implementation phase was initiated, a ground was laid for it. In the system requirement specification (see Appendix C) the main part of the functional requirements was made into user stories. This was a deliberate tactic since the choice of the process tool was known in beforehand.

The majority of the guidelines within Scrum were used with some exceptions. The exceptions were story points, burndown charts and cross-functional teams. In Scrum and other Agile process tools it is not necessary to use every prescribed part. One is rather to experiment and use that which benefits the users and project. It goes so far that different process tools can be mixed with each other. But it is important that the term Scrumish replaces Scrum in such a situation (Kniberg & Skarin 2010). This thesis used a Scrumish process tool as a guideline during the development of the product.

The story points were excluded due to the team not finding it beneficial to use. Instead an estimation of how many user stories were to be fitted in one sprint was estimated by its complexity directly. The story points were excluded since the whole team consisted of only two members and a direct dialog could handle the estimation without the story points and it

would be unnecessary to rationalize in form of a numeric value. Since story points were going to be used, burndown charts were directly excluded since they depend on an estimation of time (Kniberg & Skarin 2010).

Meanwhile cross-functional teams were excluded simply because there were only the two authors who were working with the project except from the product owner. It was therefore not possible to have cross-functional teams.

The client was given the role as the product owner while the authors gave themselves the role as the team. There was no official Scrum master since there was nobody in the project available to carry out that role, instead the team also took on the responsibilities of the Scrum master.

Priority	Todo	In progress	Done	Completion Date	Notes
1					
2					
3	1			2014-04-07	User story 1 As a user I want to be able to navigate to home, test history, scheduler, dependency and statistics after i have logged in and chosen a system.
4	2			2014-04-03	User story 4 As a user I want to be able to see a list of all systems I am a part of, and see the status of every system.
5					User story 5 As a user I want to be able to access a specific system in the list of systems. Then be redirected to the overview page of the chosen system.
6					User story 6 As a user I want to be able to see the status of the system.
7					User story 7 As a user I want to be able to see a list of the latest system checks and be able to access the report of all system checks in the list.
8					User story 3 As a user I want to be able to access the portal with a given username and password, and then be redirected to my home page.
9					User story 2 As a user I want to be able to log out at every page after I have logged in.
10					User story 8 As a user I want to be able to create a link that is directed to the overview page of the system I am currently viewing.
11					User story 14 As a user I want to be able to see and adjust my internal dependencies.
12					User story 11 As a user I want to be able to see all my runned tests in a list and to be able to filter the tests by choosing a category.
13					User story 12 As a user I want to be able to schedule a test.
14					User story 13 As a user I want to be able to see and change scheduled tests.
15					User story 18 As a user I want to be able to see statistics of the system's status.
16					User story 15 As a user I want to be able to see my external dependencies.
17					User story 19 As a user I want to be able to receive a report on statistics by e-mail.
18					User story 9 As a user I want to be able to create or edit a public status text of the system I am currently viewing.
19					User story 10 As a user I want to be able to have an e-mail sent to me containing the report of the latest system check every time it is performed, or every time the latest system check has failed.
20					User story 16 As a user I want to be able to see my internal secondary dependencies.
21					User story 17 As a user I want to be able to see a systems state.

Fig 3.1: Screenshot of the backlog in a certain moment of the thesis

The client being the product owner had the privilege and responsibility to prioritize the product backlog. The product backlog seen in figure 3.1 consisted of a Google docs Excel document with the columns “Priority”, “Todo”, “In progress”, “Done”, “Completion date” and “Notes”. The columns were made to easily monitor the state of the user stories and if necessary any extra comments regarding a user story. The rows consisted of the user stories derived from the system requirement specification. The backlog was shared between the product owner and the team.

The start phase of each sprint began with dragging items in the backlog from todo to ongoing and then placing the same user stories in the teams sprint board. The number of user stories taken into a sprint was based on estimation by the team. The team’s estimation was based on how complex a user story was in their experience. When both members of the team were in agreement, the amount was final. In later sprints, experience from the former sprints was also used to assess the amount of user stories that would fit in a sprint.

	A	B	C	D	E	F
1	Sprint 3					
2	2014-04-28 -- 2014-05-02					
3	Userstory	Todo	In progress	To verify	Done	Notes
4	User story 14 As a user I want to be able to see and adjust my internal dependencies.					
5				- present the projects in a list, by fetching project id from database		Hämtar utifrån namn och inte id
6			- Present the status and id of every project in the list			metod för hämtning av status returnerar alltid undefined
7		- set text color of status text to respective color depending on status				
8				-Add a project to internal dep. (server call)		controll för redundans
9				Add a project to internal dep. (GUI)		Projekten visas inte i rullgardinen i IE9.
10					-Remove a project from internal dep. (server call)	

Fig 3.2: Screenshot of the sprint board in sprint 4 under development

The sprint board consisted of the columns “User story”, “Todo”, “In progress”, “To verify” and “Done”. Each user story was broken down to tasks which could be started by either one of the team members. This way of working gave the team the opportunity to work with different tasks simultaneously. Figure 3.2 is an example of how one of the sprint boards looked like during the implementation. The client did not have access to the sprint board and the user stories on the board could not be cancelled, as well as no more user stories could be put into the board during the sprint. This was done according to one of the Scrum principles to protect the team from external disturbances (Kniberg & Skarin 2010).

One abnormality in the sprint was the length of each sprint, which was set to one week. While not being a violation (Kniberg & Skarin 2010) it is a quite small period. This had to do with the period left for implementation which was roughly about a month. Therefore a week was chosen to give both a fast feedback loop to the product owner and at the same time give the opportunity to have maximum amount of sprints.

On every occasion the team met up to begin a work day, a daily Scrum meeting was held in accordance to one of the principles of Scrum(Kniberg & Skarin 2010).The meetings summarized what was done on the previous work day and what the objective for the current day was. This gave clarity to how the sprint was going for each day.

When a user story was considered to be done, its state was updated in the backlog with additional comments if there were any. At the end of every sprint a notification was sent to the product owner, either by mail or by physical contact. This was in accordance to one of the principles of Scrum (Kniberg & Skarin 2010). The notification consisted of a push onto the Git server provided by the client and a message. The message was to give the product owner an idea of what was done, if there were any obstacles and a heads up that the team will drag user stories from the backlog so the prioritization should be complete for the next sprint. The product owner could then review the last push onto the Git server since the product owner was using the same development environment and capable of understanding the changes without the consultancy of the team. The product owner always gave a response to the team telling them if everything was fine or reporting any found abnormalities. After the response from the product owner the team could go into a new sprint and create a new sprint board with the user stories pulled from the backlog.

3.2.4 Discussion of process tool

In retrospect, Scrum was a powerful process tool as it gave both the team and the product owner many benefits such as:

- An overview of how the product was being developed and what would be next in line. (See chapter 3.2.1)
- An effective way of implementing with more than one task being worked with at the same time. (See chapter 3.2.1)
- Protection from disturbance from the team during sprints. (See chapter 3.2.1)
- Included the product owner in the development of the product by showing the progress at the end of each sprint. (See chapter 3.2.1)

Although both the team and client was happy with the results of the thesis (see chapter 6), it does not necessarily mean that Scrum was the only process tool that could be used. The way of working during the implementation phase could be done in a similar way with Kanban as a process tool.

With Kanban the team could have a board in which they would be able to pull tasks and have a backlog which the product owner would be able to prioritize by putting the tasks in the Kanban board's "Todo" column.

However, the team consisted only of two members dragging a task through the whole pipeline of columns. Therefore the Kanban board would not be utilized as it is thought to be, with teams working in different thresholds or columns and helping each other in bottleneck situations (Kniberg & Skarin 2010). Instead, a sprint board is more appropriate. Also, as mentioned in chapter 3.2.2, the product owner wanted periodic reports of the implementation, which is not available in Kanban since it does not use any periods at all.

From this argument, Scrum is still considered to have been a better choice than Kanban. Although more research could have been given to other alternative process tool within the Agile family.

3.3 Sources

3.3.1 Literature

Following literature was used in the thesis (see chapter 9 for more details on the literature):

1. Kanban and scrum making the most of both - Henrik Kniberg & Mattias Skarin
2. Kvalitativa Metoder, Från vetenskapsteori till praktik - Lise Justesen, Nanna Mik-Meyer
3. Real world research - Colin Robson

The first literature in the list is derived from the course EDT 655 - Project grade 3, in LTH, Campus Helsingborg and is considered by the authors to be a trustful source since it is being used in a course included in a higher form of education.

The second literature in the list was found by a search in the LTH, Campus Helsingborg library with the help of a librarian. This literature is focused on the subject of qualitative research methodology and might not be the most appropriate regarding comparison of qualitative against quantitative. However it was rather used for the purpose of using qualitative then comparing it towards quantitative.

The third literature in the list was recommended by Christin Lindholm whose title is “Education Program Leader for the Bachelor programme on Computer Science and Electrical Engineering with Automation” in LTH, Campus Helsingborg, and is by the authors seen as a legitimate and trusted proxy to give such a recommendation.

3.3.2 Internet sources

When developing with open source tools there can be less literature available, especially when using a framework which is relatively new. Therefore all sources regarding the technologies used during this thesis are websites and not articles nor literature.

However almost every website has been a direct official website for each technology and is therefore in a high matter a trusted source. The list includes developer.mozilla.org, durandaljs.com, requirejs.org, jquery.com, knockoutjs.com, nodejs.org, git-scm.com, hp.com, webex.com and w3.org.

The only website which is not an official is w3schools.com. This website is a web developer information website and is not free from external criticism. However this website is only used at chapter 2.2.1 and the information that refers to the [w3schools](http://w3schools.com) page has been verified by the authors, by checking the information towards w3c.org. The reason for using [w3schools](http://w3schools.com) is due to the information being simply and user friendly.

Ikea.com is also an official website but is the only website that is not used in chapter 2 and is a highly trusted source of information regarding IKEA. Furthermore the information used in this thesis from the official webpages have not been able to be bias since the authors only used technical facts and in one case regarding the [IKEA.com](http://Ikea.com) website used general information. The authors did not use information that would consider the tool or company to be a preferable choice against a competitive.

3.3.3 Supervisor and test department at IKEA IT, Helsingborg

The feelings and experience of the supervisor and the rest of the test department team have been used in this thesis to a certain degree. It has been used in the purpose of validating the fulfillment of the goals (see chapter 6) and also in drawing out information about the company(see chapter 1.4). The supervisor and the team are considered by the authors to be the most reliable source of information regarding validation of the goals in this thesis. The supervisor is also the client to receive the product and the team is considered as end users of the system, therefore their feelings can best explain how far the authors have fulfilled their commitment.

4 Analysis results

A wireframe prototype showing the functionalities that had been thought of before the interviews was presented to the participants (see Appendix A). The interviews resulted in a broad perspective of how the portal should be designed because of the input from various stakeholders. The data from the transcriptions and surveys (see chapter 3.1.4) was analyzed and composed into a report that was given to IKEA IT so that they could take part of the interview results, a final version of the wireframe and a system requirement specification.

This chapter presents the fundamental functionalities of the portal, the report on the interview results, the different versions of the wireframe and the system requirement specification for the portal.

4.1 Fundamental functionalities

The wireframe version 2.0 (see Appendix A) that was presented at the interviews contained functionalities that were thought out by the authors and client at several brainstorming occasions. Navigation in the portal was decided between a number of pages (in that version of the prototype) which are explained in this sub-chapter; Overview, Test History, Scheduler, Dependency, Home, Login. All functionality related to each page is presented and explained respectively in the sub-chapter for the page. For the final prototype containing all existing pages and functionality in the portal, see Appendix B.

4.1.1 Overview

List of latest smoke tests

On the overview page for every system there shall be a list containing information about the latest smoke tests ran for the specific system. A smoke test is a functional test, meaning that it tests certain functionalities of a system. What differs a smoke test from any other functional test, is that the smoke test is set to determine the health of the system it's ran within. When referring to a smoke test in this thesis it refers to the test that inflicts the system's status in the way that it is set according to the result of the test. The amount of presented smoke tests are set to three for no specific reason, but could be changed in a future version of the portal. This function is thought for a simple history of the system checks for the selected system.

Redirection to test reports

In every entry in the latest smoke tests list, there is a link to the report for the specific test. The link redirects the user to a report page where the results of the test are presented in detail. This function is thought to simplify the access to test results in comparison to ALM and was one of the main requirements for this thesis.

System state

An alert function that sets the state of a system, depending on the result of the latest smoke test ran for the system. There are three states:

- Down (test failed)
- Attention (test passed with comments or did not run)
- Good (test passed)

The purpose of this function is simple signaling of the well-being of a system. A user can see how a system is doing by just opening the overview page and not having to view any test results.

Dependent system failure notification

Also an alert function meant for easier debugging of a system failure. For every smoke test that failed, this function creates a notification by every entry in the list for the latest smoke tests if there was a dependent system whose state was “Down” at the time the smoke test was ran. As there was a dependent system that was not functional, the problem might be there and not only in the system that was checked.

Public URL

The public URL function is meant for simple monitoring of a system. It creates a link that can be sent to anybody connected to IKEA:s network which shows a replica of the “Overview” page for the system from which the link is created. The user of this link does not have to log in to the portal in order to view the page that the link redirects to. This public “Overview” page however is read-only, so it is only the information on the “Overview page” that is presented here, no buttons or links.

Visible/invisible

An option for making a system either visible or invisible to the rest of the portal users. This function is thought for system owners to be able to hide their systems in cases where the system for example is under development or maintenance, and the owner of the system does not want the test results or status of the system to be available for others to see.

4.1.2 Test History

“Test History” is the page where all results of tests that have been ran (through the portal) for a system are available in a list.

Categorization

In order to make it easier for users to evaluate their testing and access their test results, categorization of tests was thought of. This makes it easier to find test results as a user does not have to look through the whole test history. In this version of the wireframe, two categories existed; smoke tests and non-smoke tests.

Redirection to test reports

This function works the same as for the “Overview page” only that it applies to the entries in the list of tests on the “Test History” page. See chapter 4.1.1 for an explanation of this function.

Dependent system failure notification

This function works the same as for the “Overview page” only that it applies to the entries in the list of tests on the “Test History” page. See chapter 4.1.1 for an explanation of this function.

4.1.3 Scheduler

In this version of the wireframe there is no “Scheduler” page presented. Simply for the fact that the authors did not know how it would look or what functionalities would exist for the scheduler at the time the wireframe was made. It was not possible to know as it was not yet decided which test tool(s) the portal would be formed for. There is only a picture of a cloud presented in the wireframe.

4.1.4 Dependency

“Dependency” is the page where everything regarding dependencies is handled.

Categorization

As for the “Test History” page, a categorization function was thought of for the purpose of making it simple to view and evaluate the different types of dependencies of a system. The categories presented in this wireframe are as follows:

- Internal dependencies
- External dependencies
- Secondary internal dependencies

Internal dependencies

Internal dependencies are those systems that a system in question is dependent on. This is one of the main requirements for this thesis and the need for this function can be read about in chapter 1.1.

External dependencies

This type of dependencies is those systems that are dependent on the system in question. The idea of monitoring external dependencies is meant for signaling other system owners in the portal in cases where the system in question is about to go into further development or maintenance, by example. By informing other system owners about this, it could prepare them for the system in question not going to function as it should. External dependencies cannot be set by the user but are automatically set by the portal.

Secondary internal dependencies

Secondary internal dependencies are those systems that the internal dependencies for the system in question depend on. Example:

A is dependent on B.

B is dependent on C.

C is a secondary internal dependency for A.

This function is meant for the user to have a better overview of test environment by seeing an extended view (compared to only internal dependencies) of which systems are dependent on each other. Secondary internal dependencies cannot be set by the user but are automatically set by the portal.

New dependency

This function is meant for users to set systems in the portal as internal dependencies. The user picks a system from a list containing every system installed in the portal and simply adds it as an internal dependency.

Redirection to the report of latest smoke test for a dependent system

This function creates a link to a report of the latest smoke test ran for all internal and external dependencies. It is meant for users to be able to access information regarding the latest system check ran on a specific dependency. It is thought that the service owner for each dependent system should decide whether the report is available for others to view or not, for integrity issues.

4.1.5 Login & Home screen

Login

The login page is simple as the only thing needed is a username and password to login to the page. It was explained to the participants of the interviews that the accounts used for the portal were thought to be connected to the accounts used for the testing tool(s) that would come to serve as guideline(s) for the portal in a further stage of development. For the reason of making all the data in the test tool(s) available for the users in the portal.

Home

Users are redirected to the “Home” page after logging in to the portal. This page presents a list of every system that the user has a role in. Clicking on an entry in this list redirects the user to the “Overview” page of the selected system. Beside every system in the list, there is also a status text, showing the result of the latest smoke test and the date on which it was ran. This function is meant for a quick overview of the systems that are relevant to the user.

4.2 Report on interviews

Through this report, the authors have taken feelings and experiences into account with some exceptions where quantitative data has been used, presented in the form of “X of N participants had a certain feeling or thought”. It is important to keep in mind that the authors have specified how many people verbally expressed a sentence of the same opinion or feeling. However, this does not automatically mean that the remaining participants were in contrast, it only means that they did not utter it in a sentence.

A detailed explanation of all functionalities presented at the interviews can be seen in chapter 4.1.

4.2.1 General information

All stakeholders agreed on the portal being a good and favorable idea in different ways depending on the stakeholders’ perspective. Many new ideas came about during the analysis, of which most regarded the existing functionality in the wireframe, but also some completely new functionalities. One important thing in the analysis was to find out what test tool was mostly used by the employees at IKEA IT. This information was critical because it had to be decided on which tool the portal should be initially formed for. From the interviews, the authors found out that six of seven stakeholders were working with ALM. Therefore it was decided that the portal was to be implemented with support for ALM, meaning that tests should be scheduled to run with that test tool and the test results in the portal should be presented in the same way as in ALM.

4.2.2 Overview

Public URL

An URL to a publicly accessible page of a system's overview was considered a useful feature since the status of a system could be shared without the viewer having to provide any credentials. 3 of 7 users felt that this function would be beneficial. The mutual thoughts were that it allows an easy overview where you can simply see if the system in question is functional or not. A single reflection was if the link was static, i.e. not changing. This gave rise to a thought of being able to utilize this public URL to a single monitor that summarizes multiple systems' health.

Visible/invisible

The possibility of making a system visible or not visible for the rest of the systems in the portal was a function that aroused both positive emotions and hesitant thoughts. What was thought to be positive was that no unnecessary reporting should be made to others in case of the system being considered in a certain state. The hesitant thought was never clarified but still had a positive attitude towards the function. In addition, there were two specific questions about the function:

- "Can I specify a particular group to" - ID 2, *Global Quality and Test Manager*
- "Say you have four users for that button, who is in charge then - ID 4, *Test Manager*

Other comments

3 of 7 participants considered a function that flags a smoke test, indicating that a dependent system was not fully functional at the time of the test being executed, can be beneficial. Mostly because it makes it possible to see that there was not only a failure in one's own system, therefore the failure may be linked to another system. This flag is seen before even looking into the report of the smoke test in question.

Each smoke test shall be represented by the status passed, passed with comments and failed with respective colors green, yellow and red as well as a number (numeric) for amount of subtests passed in the smoke test.

Signaling by a failed smoke test was mentioned by 3 of 7 participants. The idea was that an alarm should be triggered when a smoke test fails. This is because it should be noticed by someone other than just a limited number of people who occasionally look up the status. The signal may be in the form of e-mail, SMS, or iDesk.

"I want an automatic action, when I call and ask how long it will take for you, then they should already be running and working on the problem " - ID3 , Test Architect.

4.2.3 Test History

All comments on the test history can be collected in two specific points: categorization and statistics. 2 of 7 participants believed that the visualization of test results would be favorable.

For categorization, one participant considered "Smoke test" and "Non smoke test" an enough amount of categories, while another one wanted more categories for the purpose of avoiding having to look through all the tests that exist on the test history page when

working with a certain test category. After this, a thought on making the categories adjustable by the users came about. This could also be heard around the scheduler.

Regarding statistics, a few participants expressed the need of additional information presented about test results on the test history page, in comparison to the information presented in the wireframe prototype (see Appendix A). This means that each test listed on the test history page shall be represented by the status passed, passed with comments and failed with respective colors green, yellow and red as well as a number (numeric) for amount of subtests in the smoke test with the respective status.

“I would probably want this page to show a number out of each category, rather than “passed” for smoke tests. How many green, how many yellow, how many red?” - ID 6, Service Owner

This is similar to what was mentioned regarding statistics in the “Other comments” section of chapter 4.1.2.

4.2.4 Scheduler

Feedback about the scheduler has been varied among the participants. There were positive thoughts about having a scheduler in the portal, meaning that it is handy to keep visualization of test results and scheduling of tests at the same place. It simplifies the work as the user avoids having to use multiple tools for executing a test and accessing the results of it.

One question that arose is whether it is possible to look at the results from tests not executed through the scheduler, but performed manually. However, only 1 of 7 participants mentioned that he/she worked mainly with manual testing.

3 of 7 participants thought that the portal should be made universal. This means that it should support various test scripts or test tools. Other feedback has been ideas about the scheduler which have touched different areas:

- The portal should support non-automated tests so that they also result in a report in Test history.
- Smoke tests for a system shall first run smoke tests that belong to its dependent systems, before performing its own smoke test.
- Pause/play function to make it possible for a system to have deviations of recursive runs for a fixed period.
- Categorization of tests to be scheduled.

4.2.5 Dependency

Feelings and thoughts on dependencies were both positive and critical. Some found it could be very beneficial because it gives an insight into the big picture of all systems connected to each other. While some were critical of how beneficial it can be, and how possible it is to implement. Out of the critical there was a common thought that the dependency functionality was less important in comparison to the overview and statistics. At a whole, 3

of 7 participants thought that dependency is favorable while 3 of 7 participants were critical.

Two specific points stood out beyond what is mentioned above:

- Narrow down the levels of secondary dependencies, otherwise there is a risk of it becoming redundant.
- In ALM, there is the ability to link tests to each other. A way to make them dependent on each other. If system B is dependent on system A, system B shall link the smoke test of system A to its own. Hence system A's smoke test is ran first, and then the smoke test of system B. The participant who said this however mentioned that this way of setting smoke tests as dependent on each other was not used as much as it should.

An own thought from authors also came about; how will the dependency part of the portal be affected by a system being set as invisible respectively visible?

4.2.6 New functionalities

Statistics

Several of total participants, 5 of 7, expressed interest in some sort of statistics on systems installed in the portal. Thoughts were to keep statistics on when a system is considered to be down as well as up and running. Also, various versions of statistics were brought up by different participants. For example, how many subtests in a smoke test have performed well or poorly, how many of the total smoke tests have performed well or poorly (per year, month, and week).

The ideas that emerged regarding statistics were as follows:

- Statistics on the number of subtests in a whole test that passed respectively failed.
- Keep statistics on all smoke tests to show an overview of how many passed respectively failed. One axis for time and one for tests. Axis for time could be a week, month or year.
- A report on statistics kept on all smoke tests shall be sent weekly or monthly to the owner of the system.
- Statistics should not be kept when a system is not visible.
- Statistics should not be kept at planned down time.
- Statistics that are kept at maintenance or further development of a system should be noted, for example "Ongoing development".
- The "Make visible/invisible" button should not turn off the keeping of statistics.
- The statistics should be adjusted after less technically proficient personnel. This in order for them to be able to have a perception of a system's well-being over time.
- If a system that the main system in question is dependent on failed its smoke test, statistics should not be kept for the main system.
- Statistics should also be presented in percentage.
- Statistics should be presented in the form of graphs.
- Statistics should always be available even when the system is invisible.

- When a smoke test for system A is ran by system B, the smoke test for system A should be reported into the statistics for system A (B has set A as a dependency).

Monitor

The idea of the monitor is to be able to collect the state out of multiple systems in a single view. In this way you can get an overview on the health of various systems which can be useful in several user perspectives. For example, for a particular department or a particular service owner with the need to easily be able to oversee multiple systems. 3 of 7 participants mentioned that a monitor can be beneficial.

4.3 Chronological order of the wireframe

On the basis of the report (see chapter 4.2) the wireframe version 2.0 was updated to a new version 3.0. There were multiple changes to the newer version which is presented in chapter 4.3.1. Afterwards there was an acceptance meeting held between the authors and the client for the purpose of validation. The updated wireframe was presented to the client together with the report of the interview results. The client gave some valuable feedback which the authors took in consideration and updated version 3.0 to 4.0. The 4.0 version was then again sent for validation to the client and based on the response some smaller updates were made and the final version of the wireframe was version 4.1. This chapter highlights the new features for each version of the wireframe from 2.0 to 4.1.

For the full version of wireframe 2.0 see appendix A and for the full version of wireframe 4.1 see appendix B.

4.3.1 New features of version 3.0

This chapter highlights the new features included in the wireframe version 3.0 when comparing it to the previous wireframe version 2.0. In the comparisons of this chapter, the left-hand side of the image is always taken from wireframe version 2.0, while the right-hand side of the image is always taken from wireframe 3.0.

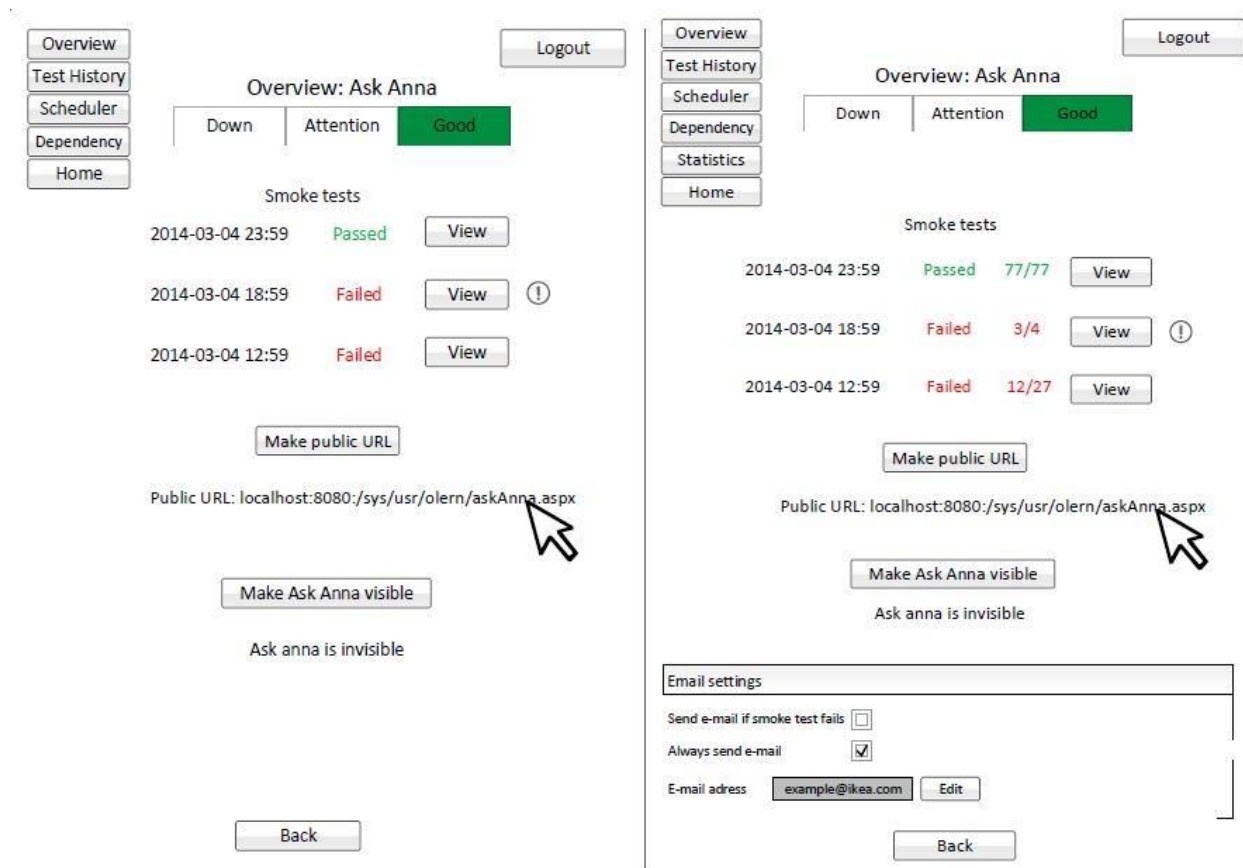


Figure 4.1: Differences in “Overview” page

Figure 4.1 shows the difference made in the “Overview” page. What can be seen here is that an e-mail function has been added at the bottom. This was added to give the end users the ability to get smoke tests reports by e-mail, either if the test failed or always. The reason for the new function is derived from the report in chapter 4.2.2.

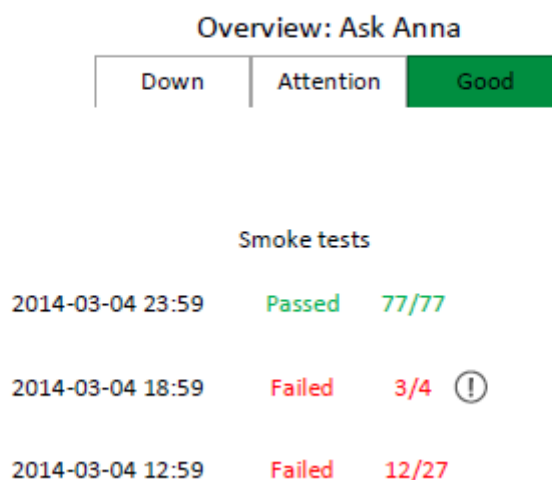


Figure 4.2: “Public Overview” page

Figure 4.2 shows a page which is not a new function, but was not included in version 2.0. It was added to the 3.0 version to give a clearer view of the “Public Overview” function. The function is derived from the report (see chapter 4.2.2).

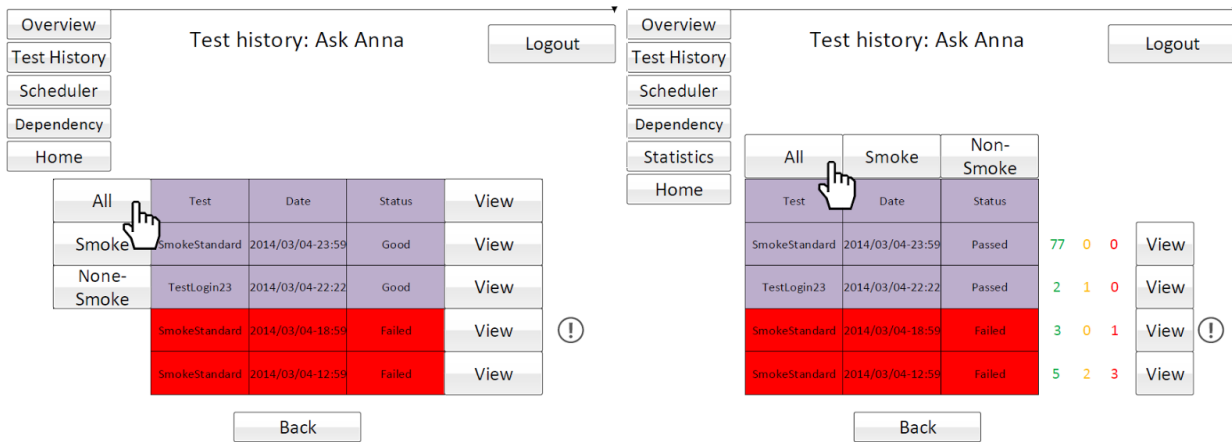


Figure 4.3: Differences in “Test History” page

Figure 4.3 shows the difference made in the “Test History” page. What can be seen here is the numbers at the side of each test set in the test history table. These numbers represent how many tests in the test set that have passed (green number), passed with comments (yellow number) or failed (red number). The reason for the new function is derived from the report in chapter 4.2.3. Additionally, the menu for the different categories of test sets has been moved from the left side to the top side of the test history table. There was no basis for this change other than that the authors thought that it would be more user friendly.

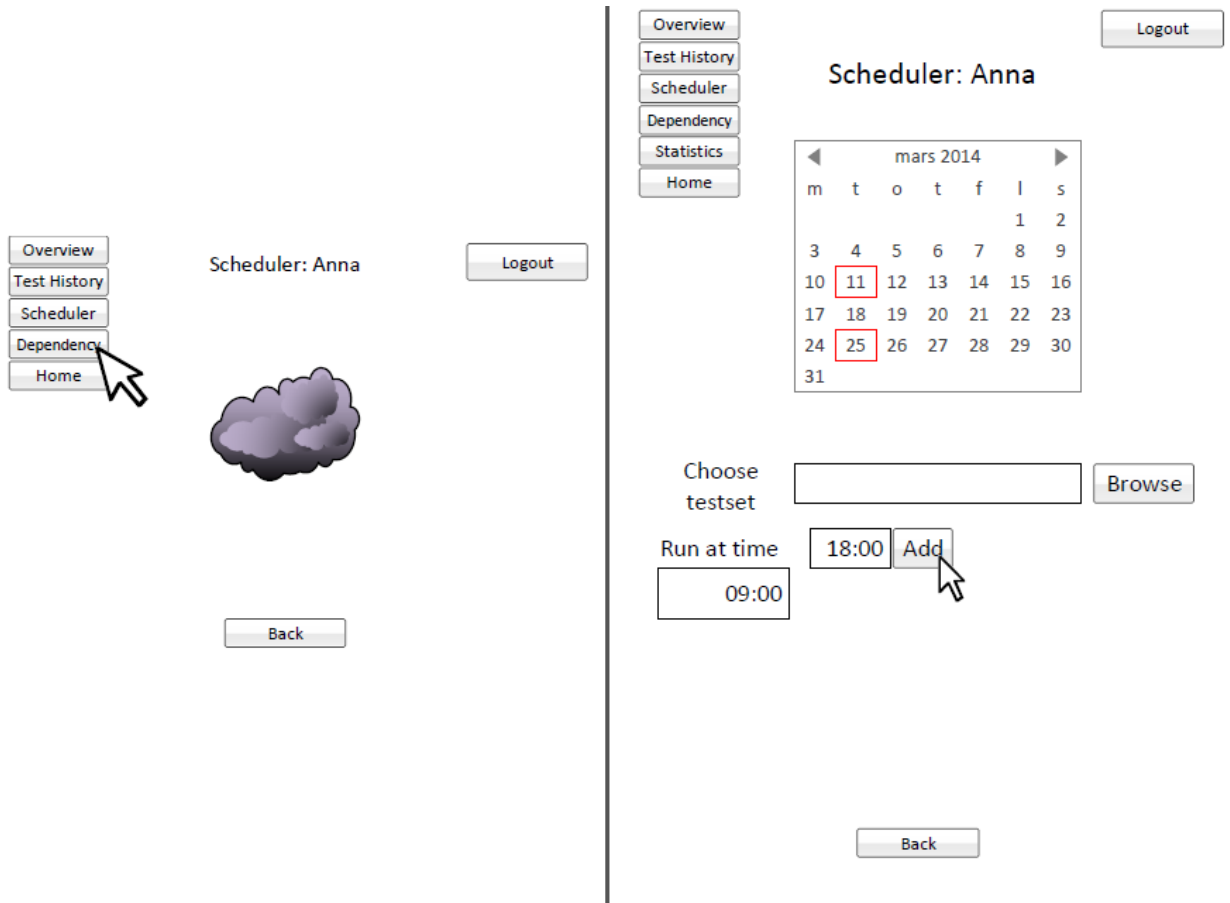


Figure 4.4: Differences in “Scheduler” page

Figure 4.4 shows that the “Scheduler” page went through a total change, since the scheduler in version 2.0 was just a cloud symbol illustrating that there was nothing to display at the moment. In version 3.0 we can see a calendar, test set choice and time function. This is to be able to schedule a certain test set for some specific date(s) and time(s).

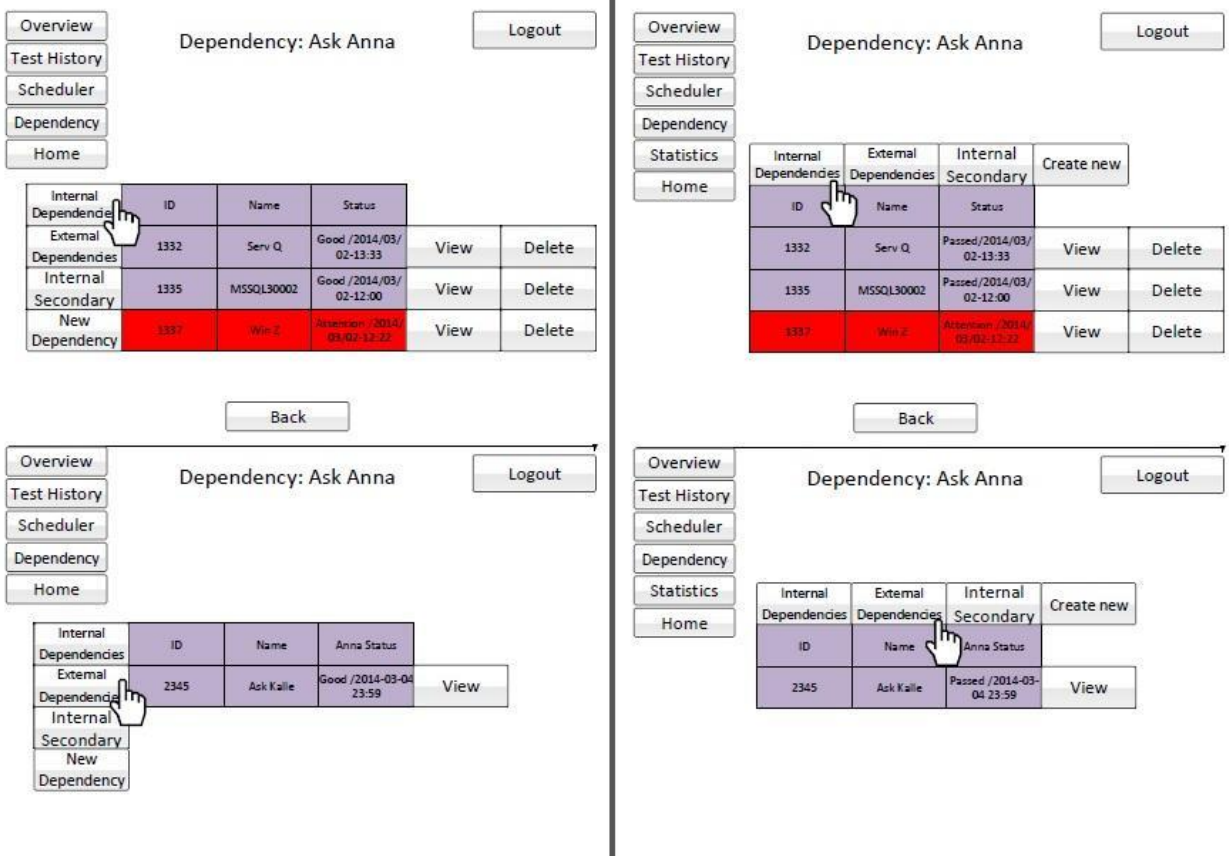


Figure 4.5: Differences in “Dependency” page

Figure 4.5 shows the changes made to the “Dependency” page, which is simply that the menu for the different categories of dependencies has been moved from the left side to the top side of the dependency table. There was no basis for this change rather than that the authors thought that it would be more user-friendly.



Figure 4.6: “Statistics” page

Figure 4.1 shows a new page called “Statistics” which is available through the navigation menu at the top-left corner. The ideas and specifications within this page were derived from

the report in chapter 4.2.6. The principle for the page is to display statistics for the system which the user is viewing. The statistics is based upon the smoke test(s) of the system and not any other test(s). The reason for this is to give a view of the system's health, which in this portal is defined by the smoke tests.

Within the page the user will be able to view the statistics in a graph which will display how many tests have been run and the ratio of how many failed/passed. This should be shown for every unit in the X-axis which is based on the choice, week, month or year. Furthermore, statistics should be kept for the whole time that the project has been active in the portal.

Another function in the "Statistics" page is to send reports by e-mail which can be seen in figure 4.6 bottom. The function is to send reports consisting of the same statistics available in the portal, but for a certain time frame which is dependent on the user's choice. The choices being weekly, monthly or/and year wise. An example is if the user picks weekly and monthly, then a report would be sent at the end of the every week containing only the recent week's statistics. Also it would send a report on the end of the month containing the statistics for the recent month.

4.3.2 New features of version 4.0

This chapter highlights the new features included in the wireframe version 4.0, when comparing it the previous wireframe version 3.0. In the comparisons of this chapter, the left-hand side of the image is always taken from wireframe version 3.0, while the right-hand side of the image is always taken from wireframe 4.0.

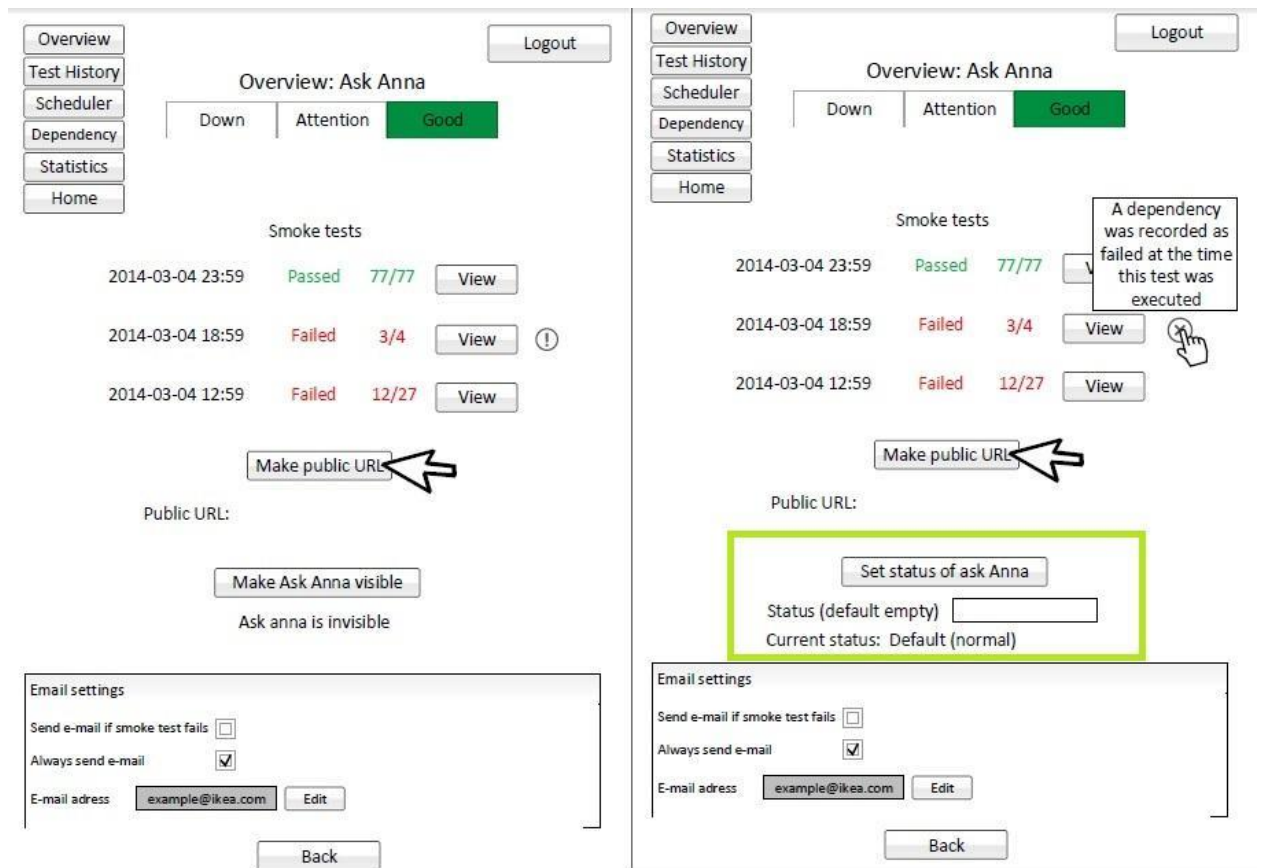


Figure 4.7: Differences in "Overview" page

Figure 4.7 shows the difference made in the “Overview” page. There are two new functions introduced to the newer version. The first one is the hovering text of the “dependency down” function, which can be seen on the right side of figure 4.7, where the “hand” symbol is over the “x within a circle” symbol. This was to give the users an understanding of the symbol’s meaning. The symbol was also changed from an exclamation mark to an X due to the second new function being inserted; so that it would not be misinterpreted which symbol represents a function.

The second new function in figure 4.7 was the extra status field, which has replaced the visible/invisible function. This can be seen within the green square in version 4.0. The text field is writeable and enables the user to change the status of the system when pushing the button. This was thought to be an extra signal so the user can signal others, independently of test scheduling. For example, the user could have a message saying “The system is currently going through maintenance, disregard any failed tests”.

These changes were based on the feedback given to the authors by the client during the acceptance meeting (See chapter 4.3).

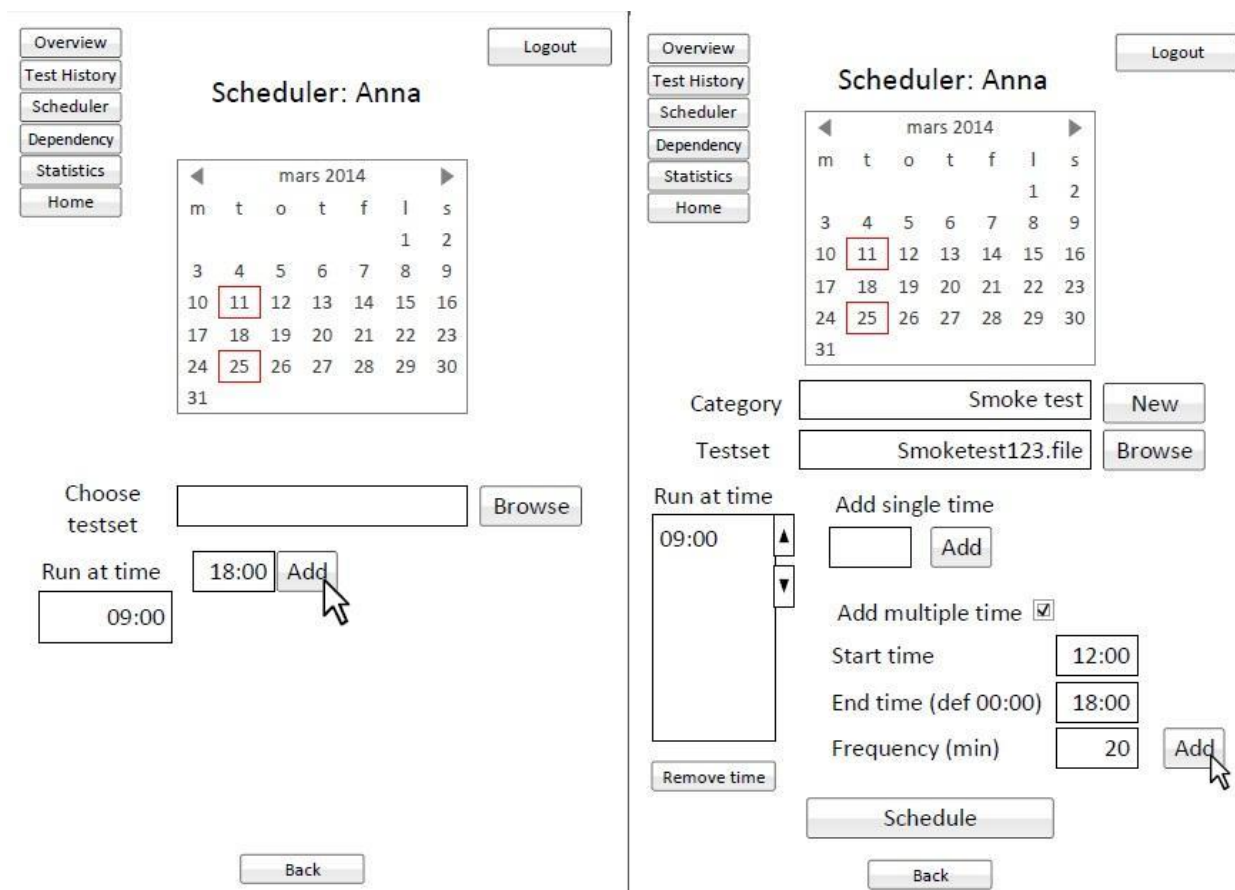


Figure 4.8: Differences in “Scheduler” page

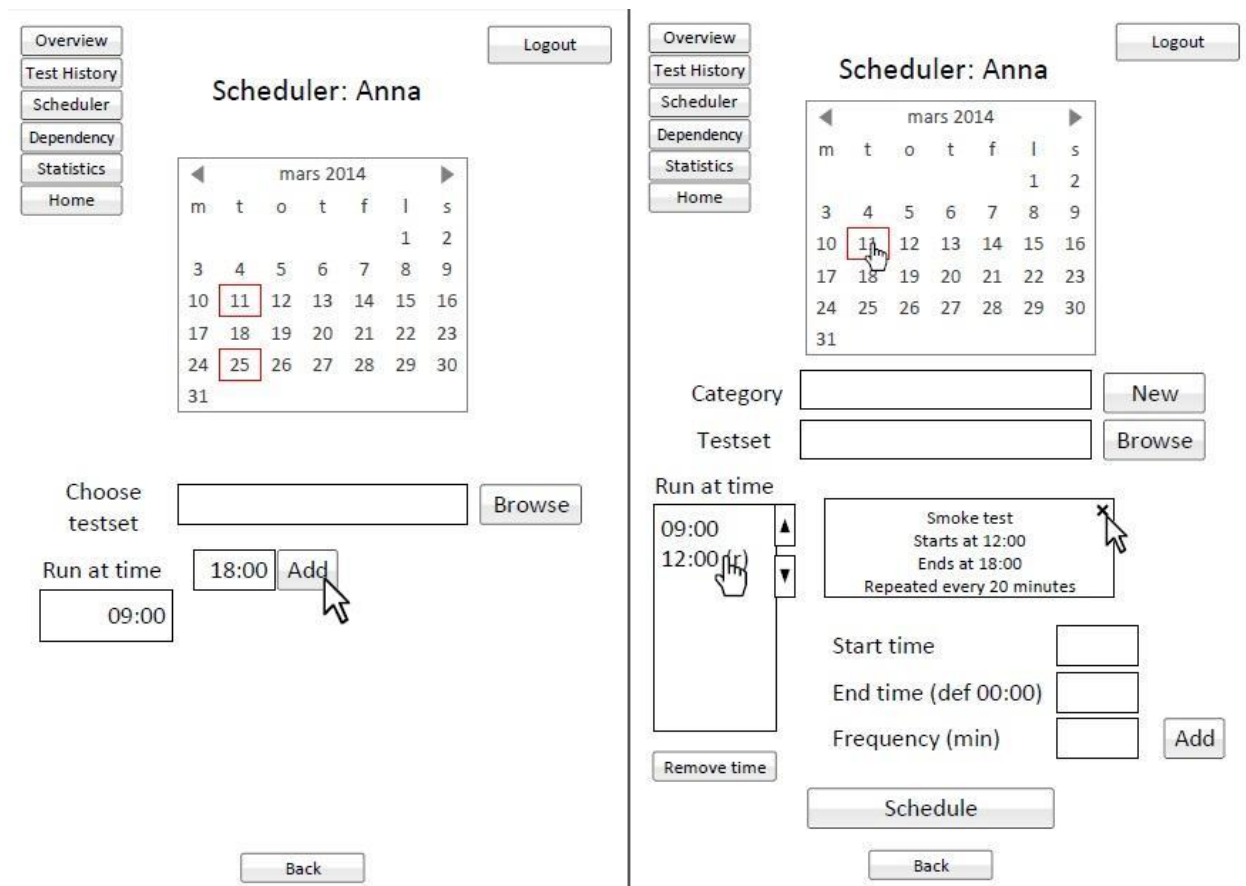


Figure 4.9: Differences in “Scheduler” page

Figure 4.8 and figure 4.9 shows the changes made to the “Scheduler page”. There are multiple changes introduced which can be categorized as following:

- Categories of tests
- Recursive time
- Current scheduled tests

Test categories can be seen in line with the label “Category” in figure 4.8. This function would enable the user to specify which category the test to be scheduled should be placed in. Furthermore, the user would be able create its own categories which will give the user the flexibility to schedule any type of test. The idea was derived from the report (see chapter 4.2.3) and on the feedback given to the authors by the client during the acceptance meeting (see chapter 4.3).

Recursive time can be fully seen in figure 4.8 from the checkbox “Add multiple time” down to the label “Frequency (min)”. This function would give the user the possibility to add multiple times more efficiently. For example, if the user would like to put multiple times with 30 minute intervals, in the 2.0 version the user would have to do many inputs. With the new function the user would instead check the checkbox “Add multiple time”, set the “Start time” to 00:00, leave the “End time” as blank (as it is default 00:00), set the “Frequency (min)” to 30 and finally push the add button. These changes were based on the feedback given to the authors by the client during the acceptance meeting (see chapter 4.3).

Current scheduled tests can be seen in figure 4.9 in the message box above the label “Start time”. The message box appears when a user has selected a date in the calendar and a time from the “Run at time” list box. The message is only shown if there is a time in the list box, which will only be filled with objects if there is a scheduled test for that date. The reason for the new function was to give the users the possibility to view already scheduled times for different tests. The example given in figure 4.9 is one with the recursive times (indicated with the “(r)”) and would only include the time and category.

These changes were based on the feedback given to the authors by the client during the acceptance meeting (See chapter 4.3).

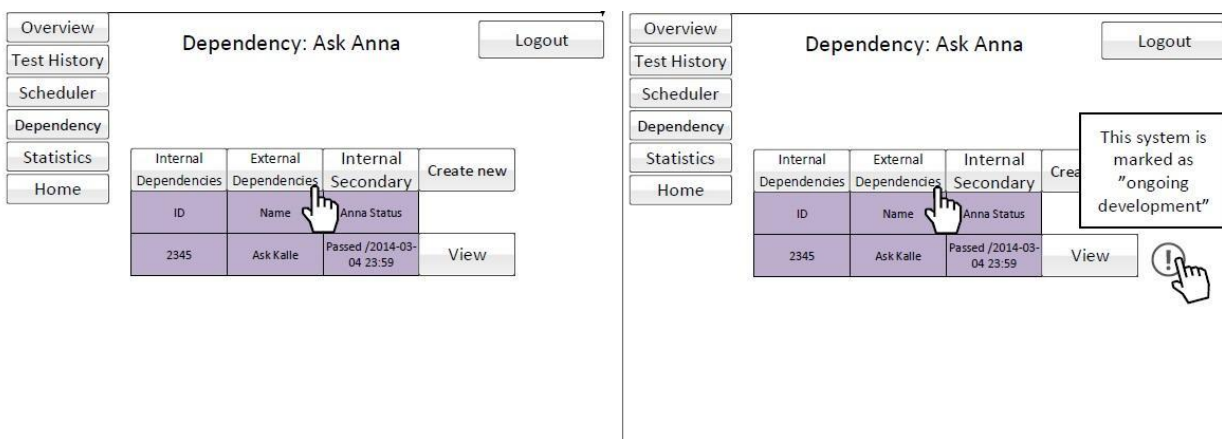


Figure 4.10: Differences in “Dependency” page

The difference presented in figure X is a notification symbol beside an entry in the external dependencies list. This mark indicates that the system in question has set a custom status. If a user hovers over the mark, an explaining text is seen, saying that the system is marked as the custom status that has been set. The purpose of this function is to inform users of the portal about the system’s status in order to give more insight into how the system is doing.

This change was based on the feedback given to the authors by the client during the acceptance meeting (See chapter 4.3).

4.3.3 New features of version 4.1

This chapter highlights the new features the wireframe version 4.1 includes when comparing it the previous wireframe version 4.0. In the comparisons of this chapter, the left-hand side of the image is always taken from wireframe version 4.0, while the right-hand side of the image is always taken from wireframe 4.1.

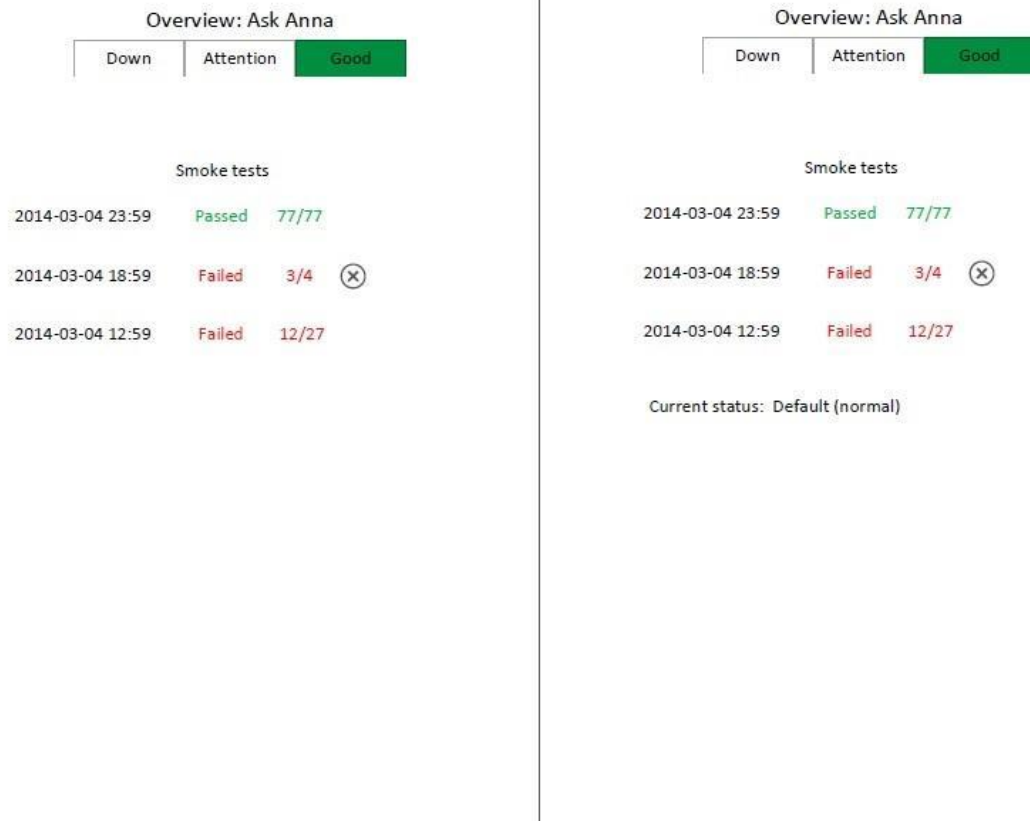


Figure 4.11: Differences in “Public Overview” page

Figure 4.11 displays the changes made to the “Public Overview” page. The new feature is to show the status of the system in the public overview, which is further explained in chapter 4.3.2. This is to give the user viewing the “Public Overview” page, the information of the current status of the system. This change was made on the basis of the feedback from the client, after version 4.0 was sent to the client for validation.

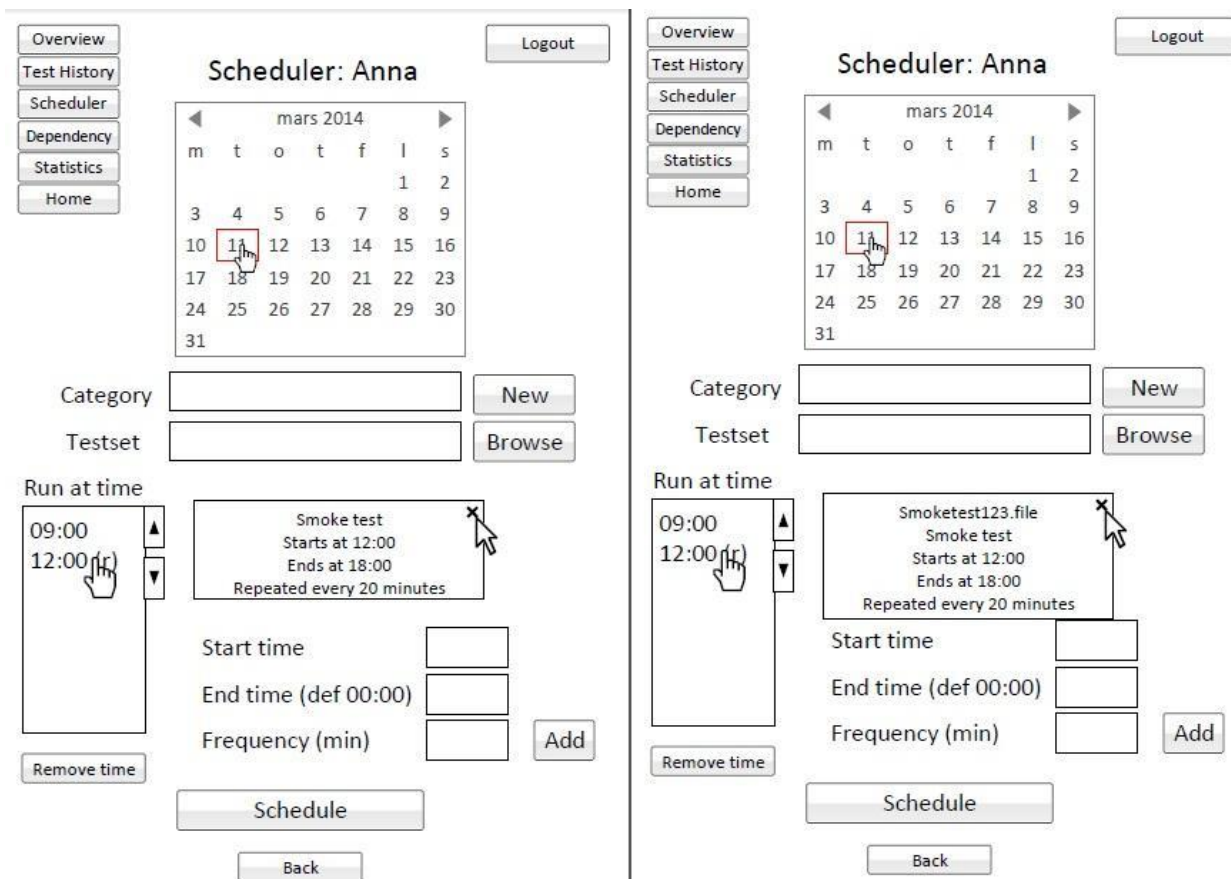


Figure 4.12: Differences in “Scheduler” page

In figure 4.12, a difference can be seen in the box in the middle of the screen containing information about a test that has been selected from the box on the left side of the screen, labeled “Run at time”. A detailed explanation of this function can be seen in chapter 4.3.2. What is different in this version of the wireframe is that additional information is presented in the box, consisting of the name of the test that has been scheduled. In this example, the test is named “Smoketest123.file”. This was added to make it easier for users to keep track of the scheduled tests, by knowing exactly which test has been scheduled at a certain date and time.

4.4 System requirement specification

A requirement specification (see Appendix C) for the portal was made for several reasons. Firstly, it was needed for a guideline of how the portal should be implemented for the authors to follow. Secondly, to make validation possible and finally for a guideline for developers who will further develop the portal.

The specification came about after an acceptance meeting was held between the client and the authors (see chapter 4.3) and is based on the report (see chapter 4.2) as well as the acceptance meeting. It is also based on the wireframe (version 4.1, see Appendix B), although some ideas and functions have been modified from the wireframe. The changes were put directly in the specification and have been validated by the client.

The most specific document and main guideline for the implementation in this thesis has been the system requirement specification.

5 Implementation results

This chapter is meant to give a view of what was and what was not implemented during this thesis. As mentioned in chapter 3.2 the implementation was based upon the results of the analysis being the system requirement specification (see Appendix C) and the wireframe version 4.1 (see Appendix B). Furthermore, this chapter displays a demonstration of the product.

5.1 What was achieved?

Following user stories from chapter 4.3 in the SRS were implemented.

- User story 1 As a user I want to be able to navigate to home, test history, scheduler, dependency and statistics after I have logged in and chosen a system.
- User story 4 As a user I want to be able to see a list of all systems I am a part of, and see the status of every system.
- User story 5 As a user I want to be able to access a specific system in the list of systems. Then be redirected to the overview page of the chosen system.
- User story 6 As a user I want to be able to see the status of the system
- User story 7 As a user I want to be able to see a list of the latest system checks and be able to access the report of all system checks in the list.
- User story 3 As a user I want to be able to access the portal with a given username and password, and then be redirected to my home page.
- User story 2 As a user I want to be able to log out at every page after I have logged in.
- User story 8 As a user I want to be able to create a link that is directed to the overview page of the system I am currently viewing. User story 14 As a user I want to be able to see and adjust my internal dependencies.
- User story 14 As a user I want to be able to see and adjust my internal dependencies.
- User story 11 As a user I want to be able to see all my runned tests in a list and to be able to filter the tests by choosing a category.
- User story 12 As a user I want to be able to schedule a test.

In total 11 out of 19 user stories were completed. The reasons for not completing every user story can be read at chapter 1.3.

Besides the user stories there were multiple other requirements in the SRS that were fulfilled: 4.1.1, 4.1.2, 4.2.2, 4.2.3, 4.2.4, 4.2.5, 4.2.6, 4.4.1, 4.4.2, 4.4.3, 5.1.1, 5.2.1, 5.2.2, 5.2.3, 5.2.4, 5.2.5, 5.2.6, 5.2.7, 5.2.8 and 5.2.10. In total 20 out of 22 requirements, which are not within chapter 4.3 and 6.1 in the SRS, were fulfilled.

The requirements 6.1.1, 6.1.2 and 6.1.3 of the SRS is neither fulfilled nor unfulfilled, due to these being requirements on the machine that will use the product. Therefore not the responsibility of the authors.

5.2 What was not achieved?

At the implementation phase of this thesis not every requirement was fulfilled in the system requirement specification, SRS (see appendix C). Following user stories from chapter 4.3 in the SRS was not implemented.

- User story 13 As a user I want to be able to see and change scheduled tests.
- User story 18 As a user I want to be able to see statistics of the system's status.
- User story 15 As a user I want to be able to see my external dependencies.
- User story 19 As a user I want to be able to receive a report on statistics by e-mail.
- User story 9 As a user I want to be able to create or edit a public status text of the system I am currently viewing.
- User story 10 As a user I want to be able to have an e-mail sent to me containing the report of the latest system check every time it is performed, or every time the latest system check has failed.
- User story 16 As a user I want to be able to see my internal secondary dependencies.
- User story 17 As a user I want to be able to see a systems state.

In total 8 out of 19 user stories was not completed. The reasons for not completing every user story can be read at chapter 1.3.

Besides the user stories there some other requirements in the SRS that was not fulfilled: 4.2.1 and 5.2.9. In total 2 out of 22 requirements, which are not within chapter 4.3 and 6.1 in the SRS, was not fulfilled. Requirement 4.2.1 was fulfilled in some of the functions of the portal, but was missing in the “scheduler” page. Requirement 5.2.9 was not fulfilled due to the time limit in the thesis, since the completion of sprints was of higher priority.

5.3 How is the product implemented?

The development environment (see chapter 2) was requested by the supervisor (who was also the client) since he had experience from working in it and is the one who will further develop the product after this thesis. Therefore he wanted the authors to develop in an environment that he was familiar with already.

Testing of code was done in Google Chrome, although the portal was required to be compatible with Internet Explorer 9. The reason for testing in Chrome is explained later in this sub-chapter. Durandal is a JavaScript library and the framework that was used for the implementation, so most of the code is written in JavaScript. The graphical interface of the portal is written in HTML/CSS.

By the use of Durandal and the plugins that are explained in chapter 2.2.2.1, the portal is implemented according to the MVVM architecture (see chapter 2.2.2.2). All functionality is made in JavaScript files. The attributes and functions in the JavaScript files are then used in the HTML files for the graphical interface as well as binding functions to objects such as buttons through the Knockout plugin, by a property called *data-bind*. An example is shown in figure 5.1 and 5.2.

```

define(['knockout', 'jquery', 'socketio', 'underscore', 'plugins/router', 'durandal/app', 'services/projectService', 'services/dateService'], function (ko, $, io, _, router, app, ps, us, projectDb, config,
var ps = require('services/projectService');
var us = require('services/userService');
var vm = {};
vm.username = ko.observable(us.getCurrentUser());
currentUser = ko.observable(config.getAuthKey());

// stores the id of the project that user views in the cache. Id is used in all views. Navigates to overview
vm.gotoProject = function (proj) {
  ps.setCurrentProject(proj.name);
  config.setProjId(proj._id);
  app.trigger("showNavbar:true", 1);
  router.navigate("#overview", true);
  return true;
};

// user logs out, clears all locally stored data
vm.logout = function () {
  us.logout();
};

```

Figure 5.1: Snippet of the view model for the “Home” page

```

<div>
  <div class ="container">
    <div class="col-md-6">
      <div class="page-header">
        <h1>Welcome back, <span data-bind="text: username"></span>!</h1>
      </div>
      <table class="table table-bordered ">
        <tbody class="list-group" data-bind="foreach: projects">
          <tr>
            <td style="border-top: none;">
              <a class="list-group-item" data-bind="click: $root.gotoProject">
                <span data-bind="text: name"></span>
              </a>
            </td>
            <td style="border-top: none;">
              <!-- ko if: status === "Failed" -->
              <div class="text-center" data-bind="style: { color: 'red' }"><span data-bind="text: status"></span></div>
              <div class="text-center" data-bind="style: { color: 'red' }"><span data-bind="text: date"></span></div>
              <!-- /ko -->
              <!-- ko if: status === "Passed" -->
              <div class="text-center" data-bind="style: { color: 'green' }"><span data-bind="text: status"></span></div>
              <div class="text-center" data-bind="style: { color: 'green' }"><span data-bind="text: date"></span></div>
              <!-- /ko -->
              <!-- ko ifnot: ((status == "Failed") || (status == "Passed")) -->
              <div class="text-center" data-bind="style: { color: 'orange' }"><span data-bind="text: status"></span></div>
              <div class="text-center" data-bind="style: { color: 'orange' }"><span data-bind="text: date"></span></div>
              <!-- /ko -->
            </td>
          </tr>
        </tbody>
      </table>
      <button class="btn btn-large btn-primary" data-bind="click: $root.logout">Log out</button>
    </div>
  </div>
</div>

```

Figure 5.2: View for the “Home” page

It can be seen at the top of figure 5.1 that there is a declaration of the object *vm*. This is the view model, and all attributes and functions that will be used on the “Home” page are added as attributes to the view model, in other words the object *vm*. In the line of code next to the declaration of *vm*, there is a declaration of an attribute *username*. This is added to the view model, therefore it is called *vm.username*.

vm.username is declared as an *observable*. Observables are JavaScript objects that can be watched, and alerts the watchers whenever it is modified (Knockout: Observables n.d.) In this way, a HTML component can keep track of an observable and update its state as the observable is changed. This is done through data-binding. It is critical that the observables

are added in the view model, otherwise they cannot be bound to a view (Knockout: Observables n.d.)

As seen in the div class in figure 5.2 called “*page-header*”, there is a span tag with a data-bind on text for username. This means that the data of an object username in the view model is bound to this span, by the text attribute. The HTML page will generate a span with the text of the object vm.username, which in this case will be the username of the user who is logged in to the portal. Anytime the object is modified, the page will generate the change.

At the bottom of figure 5.2, there is a button class called “*btn btn-large btn-primary*” with a data-bind on *\$root.logout* by the click attribute. Logout is a function in the view model which can be seen in figure 5.1, called *vm.logout*. When a user clicks on the button generated on the HTML page, the function vm.logout() will be called, since the button is binded to that function.

```
/**
 * @api {post} /project Create project
 * @apiName PostProject
 * @apiGroup Project
 *
 * @apiSuccess {String} _id Project unique identifier.
 */
app.post('/api/project', function (req, res) {
  projectController.post(req, res);
});

/**
 * @api {get} /internal_dep/:name Request all internal dependencies of a project
 * @apiName GetInternalDependencies
 * @apiGroup Dependency
 *
 * @apiParam {String} name Project name.
 *
 * @apiSuccess {Object} project All internal dependencies for a project.
 */
app.get('/api/internal_dep/:name', function (req, res) {
  projectController.getInternalDep(req, res);
});
```

Figure 5.3: Snippet of REST API for the portal

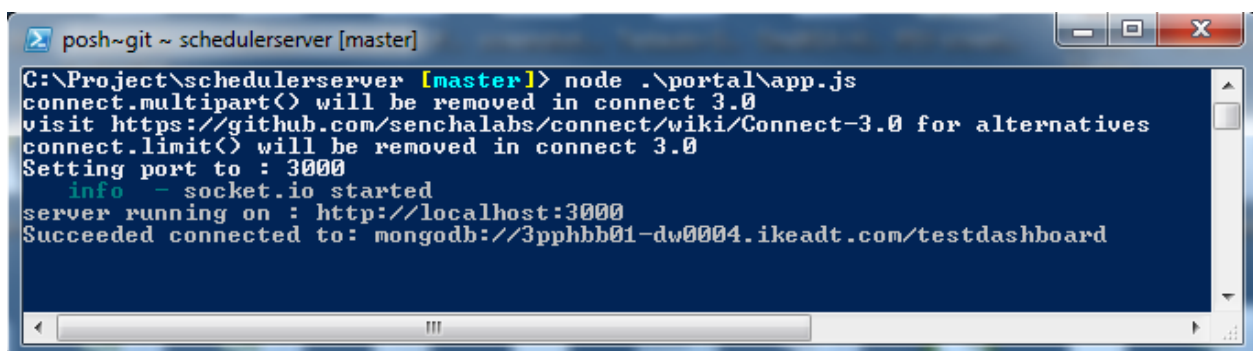


Figure 5.4: Startup of portal through Node

The product is implemented as a client-server oriented application, in which the client makes requests to the server through a REST API. Figure 5.3 shows a snippet of the REST API. A documented REST API is one of the requirements in the SRS. The server is handled with Node, a demonstration of the portal startup is seen in figure 5.4.

In order to troubleshoot the code, a plugin for Google Chrome called “KnockoutJS Context Debugger” was used. This plugin was needed since the development tool did not offer any debugging. Internet Explorer also offers a debugging environment but the authors felt that Knockout’s debugger was a better choice, since it is easier to work with. Internet Explorer by example closes the debugger every single time a page has been loaded, while Chrome keeps the debugger open all the time.

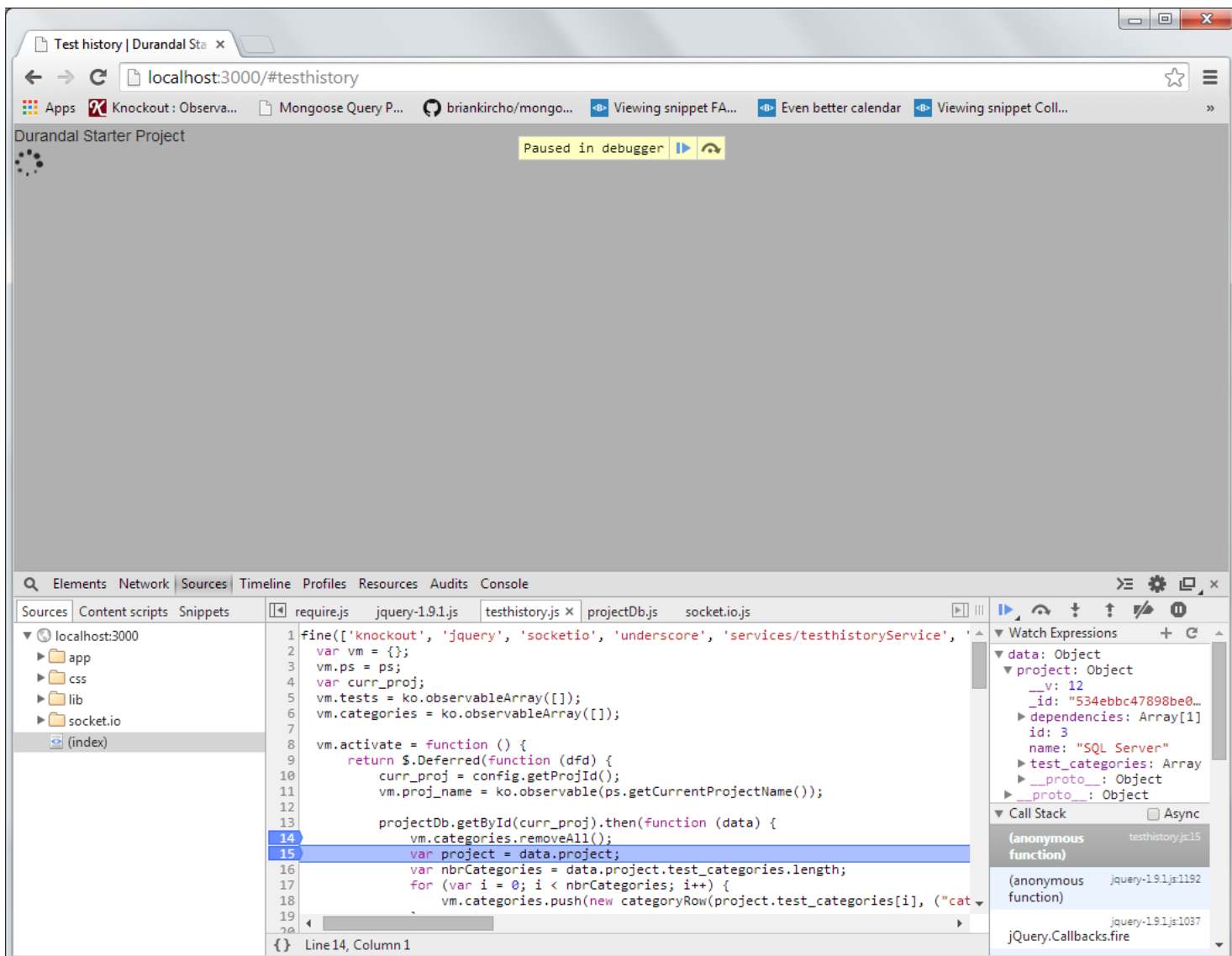


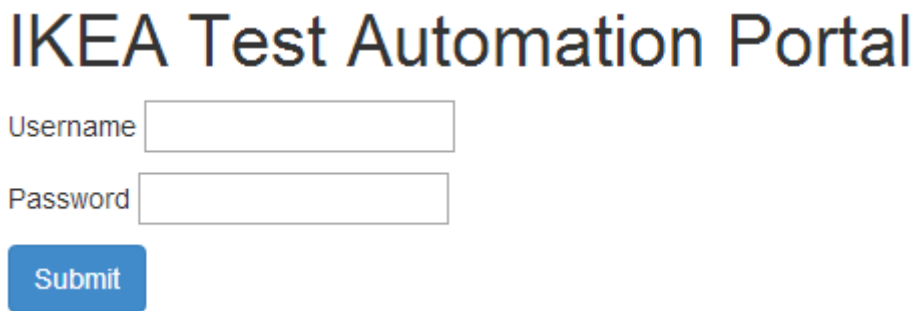
Figure 5.5: Debugging with KnockoutJS Context Debugger

Figure 5.5 displays a screenshot taken during the implementation and shows the KnockoutJS Context Debugger in action. The user (one of the authors) has navigated to the “Test History” page and is stopped due to a break point which can be seen in line 14 and 15 (blue highlight). Furthermore the tool allows the user to examine JSON (JavaScript Object

Notation) objects which can be seen within the “watch expressions” box in the bottom-right of the figure. This is useful and needed since during the development there was a need of following the flow of execution when navigating in the portal. Also, it was important to investigate the JSON objects to verify that the server calls were returning correct data in a given situation. The plugin offers more abilities to be used but the main usage during this thesis was as described above.

5.4 Demonstration of the product

This sub-chapter presents how the product looks and how it works. Everything is not implemented, such as the “Statistics” page. For details about what was implemented, see chapter 5.1, and for what was not implemented, see chapter 5.2. Note that the test sets presented in this demonstration are not real test sets. They are examples with test results that have been modified by the authors when implementing the functionality, for testing purposes (of the portal’s functionality). Therefore some test sets have inconsequent results, such as a “Failed” status when there is no subtest that has failed.



IKEA Test Automation Portal

Username

Password

Submit

Figure 5.6: “Login” page

Figure 5.6 shows the layout of the “Login” page. The portal requires the user to enter a correct username and password in order for the user to log in to the page.



Welcome back, faruk!

Test	Passed with comments 2013-07-18 11:14:49
SQL Server	Passed 2013-07-18 11:14:49

Log out

Figure 5.7: “Home” page

When a user has logged in to the portal, the user is redirected to the “Home” page, which is presented in figure 5.7. A list of all the systems that the user *faruk* is a part of is presented

on this page. Beside the systems there is a status text, showing the result of the latest smoke test and when it was ran. The status text is colored based on the test result:

- Green color: Test passed
- Yellow color: Test passed with comments or test did not run
- Red color: Test failed

The user can then navigate further into the “Overview” page of any system by clicking on one of the bars that represent the systems by their names respectively. The user can also choose to log out by clicking the blue button labeled as “Log out”. It will then be redirected back to the login screen.

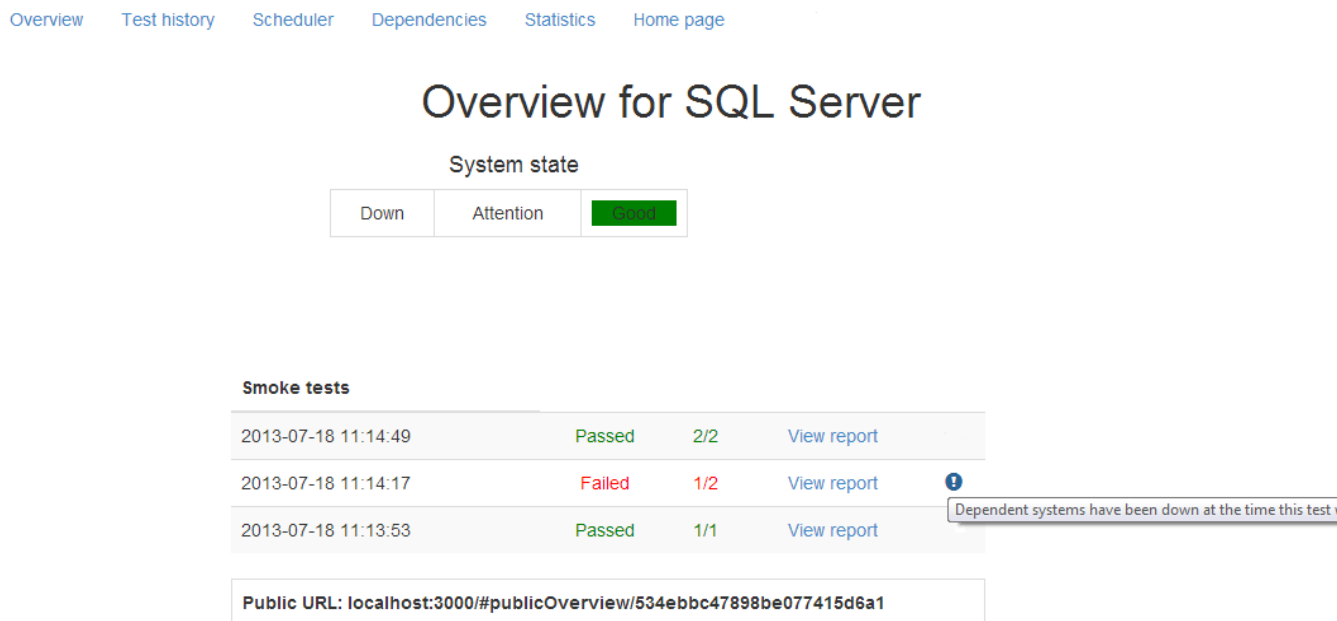


Figure 5.8: “Login” page

Figure 5.8 shows the “Overview” page for the selected system *SQL Server*. At the top of this page a navigation bar is seen. This is used for navigating through the portal. The navigation bar is available on all pages except for “Home”.

This page contains general information about the selected system. At the label “System state”, the status of the system is presented. This status is determined based on the result of the latest smoke test ran for the system. It is also colored depending on the status:

- Green color: “Good” - latest smoke test passed
- Yellow color: “Attention” - latest smoke test passed with comments or did not run
- Red color: “Down” - latest smoke test failed

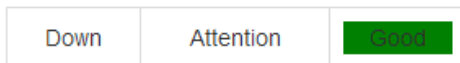
This function serves as an easy monitor of the system’s health and does not require the user to spend any time looking into the result of the latest smoke test. The user can simply see how the system is doing by looking at its current state.

In the middle of the page, there is a list of smoke tests ran for the system. The list contains three of the latest smoke tests, with information regarding when it was ran, the result of the test set, how many subtests passed and a link to a detailed report of the test (figure 5.10). The status and number of passed subtests text is colored under the same conditions as the system's state function. The list of smoke tests is made for users to have easy access to the results of relevant smoke tests, since it is the latest ones that were ran.

A notification symbol can be seen beside the second entry in the list which is the test that failed. The symbol indicates that there has been at least one dependent system that has not been functional at the time the smoke test in question was ran. Hovering the symbol displays an informative text as can be seen in figure 5.8. The purpose of this function is to simplify debugging since it lets the user know that there might be a problem in a dependent system that is causing the test to fail, and not only in the system that has been tested.

Overview for SQL Server

System state



Smoke tests


2013-07-18 11:14:49	Passed	2/2	
2013-07-18 11:14:17	Failed	1/2	 <div style="border: 1px solid gray; padding: 2px; width: fit-content;"> Dependent systems have been down at the time this test was executed </div>
2013-07-18 11:13:53	Passed	1/1	

Figure 5.9: “Overview” page

Under the list of smoke tests, there is a label “Public URL” and a link. The link redirects the user to a copy of the “Overview” page for the system in question, but with read-only rights. That meaning there are no links or buttons on the “Public Overview” page. An example of this can be seen in figure 5.9, which presents the “Public Overview” page for the system *SQL Server*.

Additionally, viewing this page does not require the user to be logged in to the portal. Therefore it can be shared to anybody connected to IKEA:s intranet. There are several purposes to this function, the first being easy monitoring of a system since the “Public Overview” page presents the same information as the “Overview” page and requires no credentials. The second one is that it can be shared.

Report: TestSeverallInstances

Test set	Date	Category	Execution time	Status
TestSeverallInstances	2013-07-18T11:14:49.182246Z	Smoke test	28 seconds	Passed
Test	Id	Instance	Status	Report
[1]TestSameName	7122	1	Passed	View report
[2]TestSameName	7123	1	Passed	View report

Figure 5.10: “Overview” page

Figure 5.10 shows how a report of any test in the portal is presented. In this example, the test *TestSeverallInstances* is viewed, and is one of the smoke tests for the system *SQL Server*. The report is presented in two tables, one for the test set and one for the subtests. The test set is presented in green columns, and the subtests in the blue columns. These are functional tests as can be seen in the name “TestSameName”, SameName being a function in this case. One thing to notice is the “View report” links in the table for tests. This is a report that is generated by ALM and has nothing to do with the implementation of this thesis. The link simply redirects the user to a HTML page containing detailed information about the specific subtest. The data in the report is taken directly from ALM, which can be seen on the date format by example.

Test history for Test

All Smoke test Non-smoke test Functional test Regression test					
Name	Date	Status	Result	Report	
TestSeverallInstances	2013-07-18 11:14:49	Passed with comments	1 0 1	View report	
TestSameNameinTS	2013-07-18 11:14:17	Passed	1 0 1	View report	
TestForMismatch	2013-07-18 11:13:53	Failed	1 0 0	View report	

Figure 5.11: “Test History” page

Figure 5.11 shows the “Test History” page, where all results from tests that have been ran through the portal are stored. What can be seen is a list of different test sets and information about them. The name of the test set, date of execution, status of the test, the result of the subtests and a link to the report on each test set is presented in each entry. The status text and results of subtests respectively are colored by the following conditions:

- Green color: Test set passed and number of subtests that passed
- Yellow color: Test set passed with comments or did not run and number of subtests that either passed with comments or did not run
- Red color: Test set failed and number of subtests that failed

The result column in the list shows the number of subtests in the test set that passed, passed with comments/did not run, respectively failed. This function is meant for showing some statistics of the test sets without the user having to view the report on the test set.

The list of test sets is contained in different tabs, each representing a certain category of the test sets. Figure 5.11 shows the tab “All”, which presents the results of all test sets available. Categorization is made to simplify the evaluation of a user’s testing since the user does not have to look through all test sets in order to find the result of a test set that the user is searching for. If the user only wants to look at functional tests, it can do so by clicking into the “Functional test” tab. One important thing to note is that there are only two default test categories:

- Smoke test
- Non-smoke test

Other test categories are added manually in the “Scheduler” page, which is presented later in this sub-chapter.

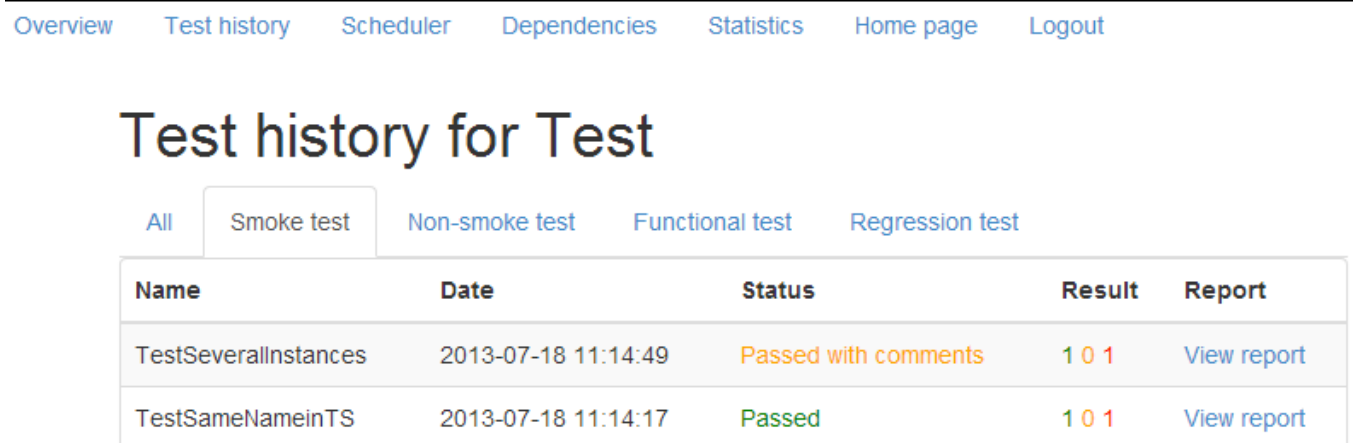
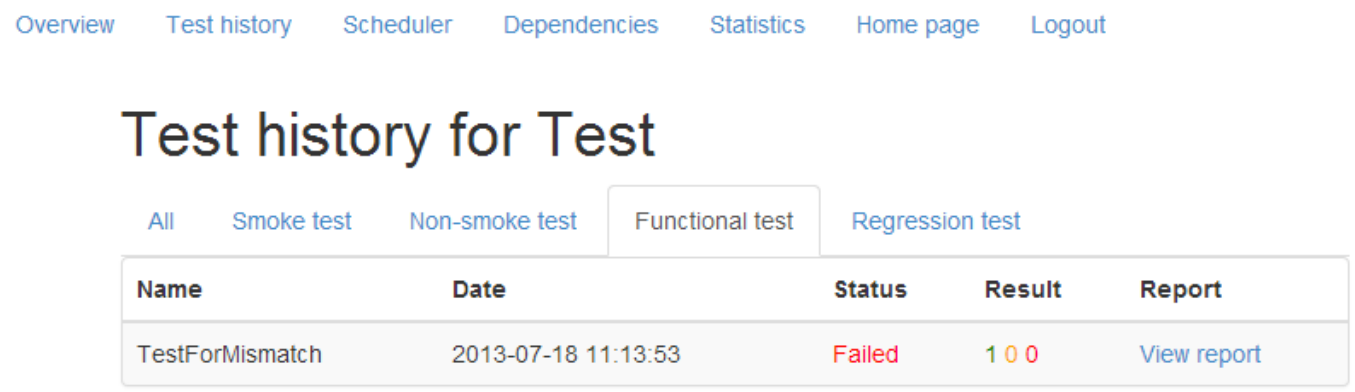


Figure 5.12: “Test History” page, showing two different tabs

As seen in figure 5.11, all test results that are available for the system *Test* are presented in that list. Figure 5.12 shows the difference between the tabs. On the upper half of figure 5.12, the user is viewing the “Functional test” tab. There is only one test set that has been categorized as a functional test, which can be seen in the list. On the bottom half of figure 5.12, the user is viewing all smoke tests for the system.

Scheduler

Schedule
Scheduled tests

Category Functional test + 🗑️

Testset

Host

Email

Time Repetitive

Timeout

Date 2014 6 19

Run every day

Email on:

Fail Warning Passed

Include HTML report

Date

2014-06-18 ✖

2014-06-19 ✖

Start time

10:00 ✖

12:00 (rep. every 15 min) ✖

Figure 5.13: Scheduling a test set

The “Scheduler” page is presented in figure 5.13. This is where the test automation is being performed from the portal. The “Scheduler page” consists of two tabs, “Scheduler” and “Scheduled tests”. The latter was not implemented due to lack of time, and will therefore not be presented in this demonstration. One important thing to keep in mind is that the scheduler is completely based on ALM. The client, who has insight in how ALM works, specified the needs of the scheduler and how it was to be implemented.

What can be seen in figure 5.13 is that there are many attributes that are required to be specified, such as what category the test to be scheduled is of, which test set to be scheduled, on what host the test is to be scheduled for execution and which e-mail should be used for alerting.

Selecting a category for the test set is done by clicking the dropdown list by the label “Category”. This list contains test categories that have been added by the testers working with the specific system. There are two default categories, “Smoke test” and “Non-smoke test”. This function is based on results of the analysis, which can be read about in chapter 4.2.4.

The user has the option to add and remove test categories. This is done by clicking the green plus symbol respectively the red trash symbol (at the side of the category dropdown list), which opens up a small window on the screen.

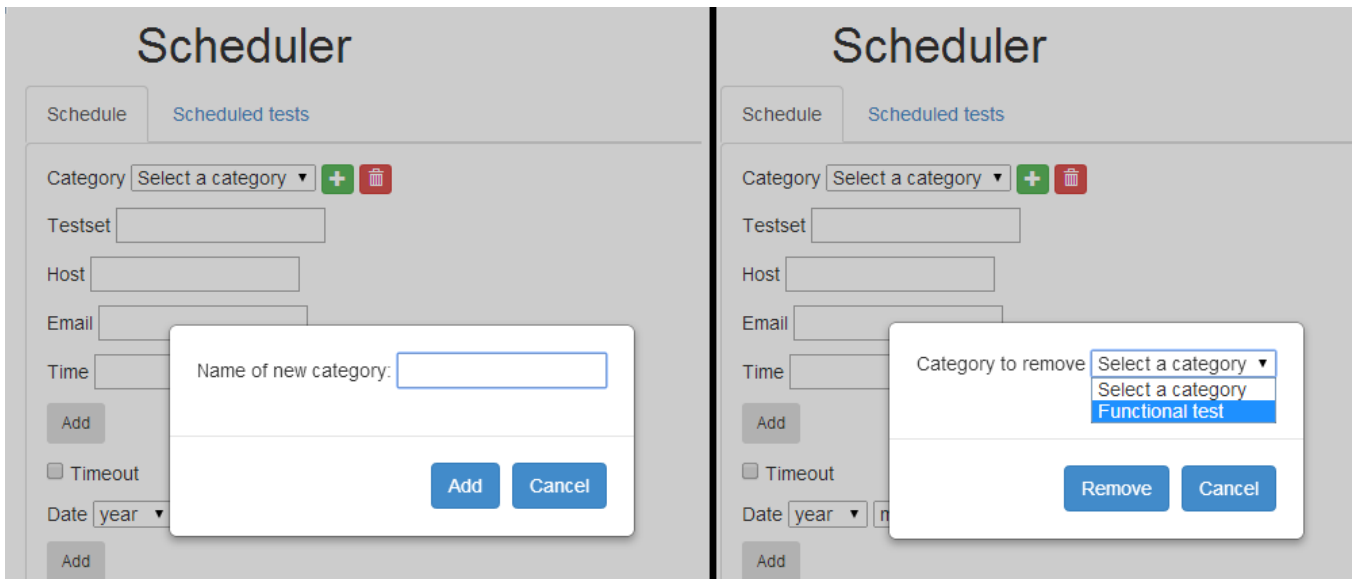


Figure 5.14: Adding and removing test categories

When adding a new category, the user simply specifies the name of it and presses the button “Add”, which can be seen on the left-side of figure 5.14. When removing a category, the user selects one of the existing categories in a dropdown list, as can be seen on the right-side of figure 5.14. The user then presses the “Remove” button to remove the category. One thing to notice here is that the categories “Smoke test” and “Non-smoke test” are not possible to remove as they are default categories and are therefore not available in the dropdown list.

Scheduler

Schedule
Scheduled tests

Category Functional test + 🗑️

Testset

Host

Email

Time Repetitive

Frequency (minutes)

Timeout

Date 2014 6 19

Run every day

Email on:

Fail Warning Passed

Include HTML report

Date

2014-06-18	✖
2014-06-19	✖

Start time

10:00	✖
12:00 (rep. every 15 min)	✖

Figure 5.15: Scheduling a test set for repetition

There are different ways to schedule a test. The scheduler is implemented in a way that offers flexibility, since a test can be scheduled for multiple times at a day. The user can also choose to use the same schedule for multiple dates. Additionally, a test can be scheduled to run every day at the selected time(s), starting from the date it is scheduled for. This can be seen in figure 5.15 at the right side of the page.


The test *TestLogin* is set to be scheduled at the 18th and 19th of June 2014, at 10:00 and 12:00. What can be seen at the times is that at 12:00, the test is to be repeated every 15 minutes. This is a function that was requested by the client. The option to schedule a test for repetition is selected by checking the “Repetitive” checkbox. The user then has to specify how often the test should be repeated, in minutes as can be seen in figure 5.15.

The conditions for the alerting function can be seen at the bottom of the page, right above the “Schedule” button. An e-mail can be sent to the user upon a failed test, a test passed with comments or a passed test, with the option of having the report included in the mail.


There is also a time-out function, which is activated by checking the checkbox “Timeout” that can be seen in figure 5.13 and 5.15. This function sets a fix for how long a test is allowed to run. By example, time-out can be set to one minute for a test set. For each subtest in the test set, the subtest is cancelled when one minute of execution has passed and then the next subtest in the sequence starts to execute. This function exists for breaking infinite execution of tests in cases where by example a pop-up window interferes with the testing and does not let the execution proceed until the window is closed.

[Overview](#) [Test history](#) [Scheduler](#) [Dependencies](#) [Statistics](#) [Home page](#) [Logout](#)

Dependency view for SQL Server

Name	Status	Report	
Ask Anna	Passed	View report	

Create a new dependency

Choose project 

- Choose a system
- Test
- Ask Kalle
- Ask Anna
- Test123

Figure 5.16: Adding an internal dependency

Figure 5.16 presents the “Dependency” page for the system *SQL Server*. Internal dependencies (see chapter 4.1.4) for the system are presented in a list on this page. As figure 5.16 shows, information about each dependency can be seen in the list. An entry in the list consists of the name of the dependent system, its current status and a link to the report on the latest smoke test ran for the dependent system. For every system that is presented in the list, it is the manager of the system that decides whether the report should be available for other users or not. The purpose of that function is to protect the integrity of the system in question.

Removing an internal dependency is done by clicking the red trash symbol that is seen on the right-side of figure 5.16. This symbol exists in every entry in the list so that the user can easily handle dependencies, avoiding having to search through a list of internal dependencies in order to remove one.

Setting a system as an internal dependency is done by selecting a system from the dropdown list that can be seen on the down-left corner of figure 5.16, by the label “Create a new dependency”. The user then simply clicks the green plus symbol next to the dropdown list and the chosen system is set as an internal dependency. All systems that are installed in the portal are available in this dropdown list in order to make it possible for managers of newly installed systems to setup the dependencies of the system, but also for other managers to handle dependencies in a flexible way.

The “Dependency” page makes it possible for testers and managers to see an overview of the test environment, the health of it and more specifically the well-being of critical systems, which are the internal dependencies of the system in question. Having an insight into this makes it easier for testers to plan a test period with no failures due to external factors.

6 Conclusion

6.1 Problem description

Both problem descriptions of the thesis (see chapter 1.2) were answered at the end of the thesis during the final presentation for the test department of IKEA IT, Helsingborg. The presentation was held at the IKEA test centers in Älmhult, Helsingborg and gave insight into how the project was implemented and the last results, the portal. The participants consisted of the full team from the test department of IKEA IT, Helsingborg and totaled in 17 participants. Included in the participants were also the owner of project Anna Gamalielson and the supervisor Olof Ernstsson. These were given the room to ask questions and give feedback to the authors.

Through the meeting the authors could confirm that the end users verbally expressed that the portal would be beneficial, thus answering the first problem description.

Regarding the second problem description the participants including the supervisor gave some concrete feedback which answered the problem description. The feedback was the following:

“Previously there was no good way for a project to know before their test phase if any depending project was running as expected. With the test dashboard they now will have a tool that allows them to setup test automation of depending systems to early get an indication if something is not working as expected in systems they are depending on. The dashboard will also give a better overview of test status for test automation. Before users had to open several test sets and look at their status individually, now instead they can login to one place and see the overall status of all test sets.” - Olof Ernstsson, Product Specialist.

6.2 The scope

The scope of the thesis (see chapter 1.2) consisted of two goals which is partially fulfilled.

The first goal “Identify the needs of the existing test environment through an analysis” is considered by the authors and both the owner and supervisor of the project to be achieved. This is based on the fact that an analysis to identify the needs was conducted (see chapter 3.1) and the results of the analysis (see chapter 4) which reflects a broad spectrum of stakeholders (see chapter 3.1.3). There is still however some room for discussion of how the analysis was performed (see chapter 3.1.6)

While the second goal is partially achieved since the system requirement specification (see appendix C) was not entirely fulfilled (see chapter 5). However the two subsections of the goal was achieved since there is visualization for dependencies and support for test automation in the portal. The reason for not fulfilling is due to the limits the authors were confined to (see chapter 1.3).

7 Further work

There are multiple areas within the portal that needs further development.

As mentioned in chapter 1.3 the machine that would perform the actual testing has not been within the scope of this thesis. This should be the main target for further work since the portal itself is dependent on this machine to run its test automation.

It is also mentioned in chapter 5.2 that not all user stories from the system requirement specification (See appendix C) were implemented. Therefore they are a secondary recommendation to be included in further implementation.

Furthermore it is possible that the analysis which was performed during this thesis might have excluded some stakeholders or missed some ideas. Therefore a further evaluation based on a live version is recommended.

The client has already received a further development plan specialized for him regarding code improvements and non-existing functionality. This is according to requirement 5.2.3 in the system requirement specification.

8 Terminology

User story	Describes a user's need of performing something in the system in one or multiple sentences. Rather than having to specify each part of a need or function in traditional requirements.
Portal	Also called the product and the system, the portal is the concept and result of the implementation of the thesis. A web platform that allows users to get information from multiple sources at one location.
Client	The person(s) which the thesis is conducted for. The client consists of two people from the test department of IKEA IT, Helsingborg. Anna Gamalielsson - Infrastructure Manager Olof Ernstsson - Product Specialist
Supervisor	The person(s) who has watched over the thesis and have been giving feedback about different subjects, such as report writing and programming code validation. The supervisor from IKEA IT was Olof Ernstsson - Product Specialist The supervisor from the Lund Institute of Technology was Christian Nyberg - Docent in Communication Systems
Test set	Contains several different sub tests. Is used to include multiple sub tests rather than to select one by one.
Subtest	One of the tests contained within a test set
Smoke test	Standard functional test for a certain system that can determines if the system is functional or not.
REST API	Describes the functions GET, POST, PUT and DELETE of methods contained in a class.
Process tool	A tool which provides recommended guidelines for how a project can be processed.
Test automation	When testing is done in an automated matter, meaning that it only needs a start and will then continue to perform the tasks without any human interference.
SRS	Is short for system requirement specification. This declares the requirements of the product to be developed for the client.
Phenomenological data analysis	When studying data, the focus is to draw out the experience regarding a certain phenomenon, in order to understand the feelings, thoughts and point of view of the subject of the research.

9 References

- About - Git. (n.d.). *About*. Retrieved 29 05, 2014 from <http://git-scm.com/about>
- Application Lifecycle Management (ALM). (n.d.) *Application Lifecycle Management (ALM)*. Retrieved 29 05, 2014, from <http://www8.hp.com/us/en/software-solutions/application-lifecycle-management.html>
- CSS Introduction. (n.d.). *CSS Introduction*. Retrieved 29 05, 2014, from http://www.w3schools.com/css/css_intro.asp
- CSS3 Introduction. (n.d.). *CSS3 Introduction*. Retrieved 29 05, 2014, from http://www.w3schools.com/css/css3_intro.asp
- Företagsinformation - IKEA. (n.d.). *IKEA Gruppens struktur*. Retrieved 04 06, 2014 from http://www.ikea.com/ms/sv_SE/about-the-ikea-group/company-information/
- Get Started | Durandal. (n.d.). *Getting Started*. Retrieved 26 05, 2014, from <http://durandaljs.com/get-started.html>
- HTML5 Introduction. (n.d.) *HTML5 Introduction*. Retrieved 29 05, 2014, from http://www.w3schools.com/html/html5_intro.asp
- Home | Durandal. (n.d.). *Meet Durandal*. Retrieved 26 05, 2014, from <http://durandaljs.com/>
- Justesen, L. & Mik-Meyer, N. (2012). *Kvalitativa Metoder, Från vetenskapsteori till praktik*. 1st ed., Studentlitteratur AB
- jQuery (n.d.). *What is jQuery?* Retrieved 26 05, 2014, from <http://jquery.com/>
- Kniberg, H. & Skarin, M. (2010). *Kanban and scrum making the most of both*. 1st ed., C4Media
- Knockout: Observables. (n.d.). *Observables*. Retrieved 05 06, 2014, from <http://knockoutjs.com/documentation/observables.html>
- Knockout: Introduction. (n.d.). *Introduction*. Retrieved 26 05, 2014, from <http://knockoutjs.com/documentation/introduction.html>
- MongoDB. (n.d.) *Agile and Scalable*. Retrieved 27 05, 2014, from <http://www.mongodb.org/>
- Mozilla Developer Network and individual contributors. (2013, 09 17). *About JavaScript*. Retrieved 26 05, 2014, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

node.js. (n.d.). *Node's goal is to provide an easy way to build scalable networks*. Retrieved 26 05, 2014, from <http://nodejs.org/about/>

Pieters, S. (2013, 05 28). *Differences from HTML4*. Retrieved 29 05, 2014, from <http://www.w3.org/TR/html5-diff/#new-elements>

RequireJS. (n.d.). Retrieved 26 05, 2014, from <http://requirejs.org/>

Robson, C. (2011). *Real world research*. 3rd ed., John Wiley Sons Ltd

Web Education Community Group. (2013, 03 02). *HTML/Training/What is HTML*. Retrieved 29 05, 2014 from http://www.w3.org/community/webed/wiki/HTML/Training/What_is_HTML%3F

Appendix A – Wireframe 2.0

Username

Password



Home

Good 2014/03/02-15:05

Good 2014/03/02-15:00

Fail 2014/03/02-10:05

- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Overview: Ask Anna

Down	Attention	Good
------	-----------	------

Smoke tests

2014-03-04 23:59	Passed	View	
2014-03-04 18:59	Failed	View	!
2014-03-04 12:59	Failed	View	

[Make public URL](#) 

Public URL:

[Make Ask Anna visible](#)

Ask anna is invisible

[Back](#)

- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Overview: Ask Anna


Down	Attention	Good
------	-----------	------

Smoke tests

2014-03-04 23:59	Passed	View	
2014-03-04 18:59	Failed	View	!
2014-03-04 12:59	Failed	View	

Make public URL

Public URL: localhost:8080:/sys/usr/olern/askAnna.aspx

Make Ask Anna visible 

Ask anna is invisible

Back



- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Overview: Ask Anna

Down | Attention | **Good**

Smoke tests

2014-03-04 23:59	Passed	View
2014-03-04 18:59	Failed	View 
2014-03-04 12:59	Failed	View 

[Make public URL](#)

Public URL: localhost:8080:/sys/usr/olern/askAnna.aspx

[Make Ask Anna invisible](#)

Ask anna is visible

[Back](#)

- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Smoke test for Ask Anna 2014-03-04 18:59

Dependency 1335 was down at 2014/03/04-17:59

Test			Status
Initialize	A initialize: OK	B initialize: OK	Good
Messaging	A MSG: OK	B: MSG OK	Good
Save Log	Save Log to DB: Fail		Failed
Exit	Disconnect A: OK	Timer close	Good

Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Test history: Ask Anna

All	Test	Date	Status	View
Smoke	SmokeStandard	2014/03/04-23:59	Good	View
None-Smoke	TestLogin23	2014/03/04-22:22	Good	View
	SmokeStandard	2014/03/04-18:59	Failed	View
	SmokeStandard	2014/03/04-12:59	Failed	View




Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Test history: Ask Anna

Logout

All	Test	Date	Status	View
Smoke	TestLogin23	2014/03/04-22:22	Good	View
None-Smoke				





Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Test history: Ask Anna

Logout

All	Test	Date	Status	View
Smoke	SmokeStandard	2014/03/04-23:59	Good	View
None-Smoke	SmokeStandard	2014/03/04-18:59	Failed	View
	SmokeStandard	2014/03/04-12:59	Good	View

Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Smoke test for Ask Anna 2014-03-04 18:59



Dependency 1335 was down at 2014/03/04-17:59

Test			Status
Initialize	A initialize: OK	B initialize: OK	Good
Messaging	A MSG: OK	B: MSG OK	Good
Save Log	Save Log to DB: Failed		Failed
Exit	Disconnect A: OK	Timer close	Good

Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Logout

Scheduler: Anna



Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	ID	Name	Status		
External Dependencies	1332	Serv Q	Good /2014/03/02-13:33	View	Delete
Internal Secondary	1335	MSSQL30002	Good /2014/03/02-12:00	View	Delete
New Dependency	1337	Win Z	Attention /2014/03/02-12:22	View	Delete

Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	ID	Name	Anna Status	
External Dependencies	2345	Ask Kalle	Good /2014-03-04 23:59	View
Internal Secondary				
New Dependency				

- Overview
- Test History
- Scheduler
- Dependency
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	ID	Name	Status	Dependent	
External Dependencies	1332	Serv Q	Good /2014/03/02-13:33	None	View
Internal Secondary	1335	MSSQL30002	Good /2014/03/02-12:00	MSSQL SERV1	View
New Dependency	1337	Win Z	Attention /2014/03/02-12:22	None	View



Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Dependency: Ask Anna

Logout

Choose Object

MSSQL

MySQL

LINUX



Choose Object

MSSQL30003

MSSQL30004

MSSQL30005



Accept Cancel



- Overview
- Test History
- Scheduler
- Dependency
- Home

Dependency: Ask Anna

Logout

Internal Dependency	ID	Name	Status		
External Dependencies	1332	Serv Q	Good /2014/03/ 02-13:33	View	Delete
Internal Secondary	1335	MSSQL30002	Good /2014/03/ 02-12:00	View	Delete
New Dependency	1337	Win Z	Attention /2014/ 03/02-12:22	View	Delete
	1335	MSSQL30005	Good /2014/03/ 02-12:00	View	Delete

Back

- Overview
- Test History
- Scheduler
- Dependency
- Home

Test set 1337 2014/03/02-12:22


Logout

Test			Status
Ping1	0 % loss: OK	32 x 4 KBITS	Good
Ping2	0 % loss: OK	32 x 4 KBITS	Good
Respons time	10ms: HIGH		Attention

Back

Home

3D kitchen	Good 2014/03/02-15:05
Ask Anna	Good 2014/03/02-15:00
IKEA Family	Fail 2014/03/02-10:05
Logout	




Username

Password

Appendix B – Wireframe 4.1

Username

Password



Home

Passed 2014/03/02-15:05

Passed 2014/03/02-15:00

Failed 2014/03/02-10:05

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Overview: Ask Anna

Down | Attention | **Good**

Smoke tests

2014-03-04 23:59	Passed	77/77	
2014-03-04 18:59	Failed	3/4	
2014-03-04 12:59	Failed	12/27	

A dependency was recorded as failed at the time this test was executed



Make public URL

Public URL:

Set status of ask Anna

Status (default empty)

Current status: Default (normal)

Email settings

Send e-mail if smoke test fails

Always send e-mail

E-mail adress

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Overview: Ask Anna

Down | Attention | **Good**

Smoke tests

2014-03-04 23:59	Passed	77/77	View	
2014-03-04 18:59	Failed	3/4	View	
2014-03-04 12:59	Failed	12/27	View	

[Make public URL](#)

Public URL: localhost:8080:/sys/usr/olern/askAnna.aspx



[Set status of ask Anna](#)

Status (default empty)

Current status: Default (normal)

Email settings

Send e-mail if smoke test fails

Always send e-mail

E-mail adress [Edit](#)

[Back](#)

Overview: Ask Anna

Down	Attention	Good
------	-----------	------

Smoke tests

2014-03-04 23:59 **Passed** 77/77

2014-03-04 18:59 **Failed** 3/4 

2014-03-04 12:59 **Failed** 12/27

Current status: Default (normal)

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Overview: Ask Anna

Down | Attention | **Good**

Smoke tests

2014-03-04 23:59	Passed	77/77	View	
2014-03-04 18:59	Failed	3/4	View	
2014-03-04 12:59	Failed	12/27	View	

[Make public URL](#)

Public URL: localhost:8080:/sys/usr/olern/askAnna.aspx

[Set status of ask Anna](#)



Status (default empty)

Current status: Default (normal)

Email settings

Send e-mail if smoke test fails

Always send e-mail

E-mail adress [Edit](#)

[Back](#)



- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Overview: Ask Anna

Down | Attention | **Good**

Smoke tests

2014-03-04 23:59	Passed	77/77	View
2014-03-04 18:59	Failed	3/4	View 
2014-03-04 12:59	Failed	12/27	View 

Make public URL

Public URL: localhost:8080:/sys/usr/olern/askAnna.aspx

Set status of ask Anna

Status (default empty)

Current status: Maintenance W.1-4

Email settings

Send e-mail if smoke test fails

Always send e-mail

E-mail adress

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Smoke test for Ask Anna 2014-03-04 18:59

Dependency 1335 was down at 2014/03/04-17:59

Test			Status
Initialize	A initialize: OK	B initialize: OK	Passed
Messaging	A MSG: OK	B: MSG OK	Passed
Save Log	Save Log to DB: Fail		Failed
Exit	Disconnect A: OK	Timer close	Passed

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Test history: Ask Anna

All	Smoke	Non-Smoke
Test	Date	Status
SmokeStandard	2014/03/04-23:59	Passed
TestLogin23	2014/03/04-22:22	Passed
SmokeStandard	2014/03/04-18:59	Failed
SmokeStandard	2014/03/04-12:59	Failed

77 0 0

2 1 0

3 0 1

5 2 3

- View
- View
- View
- View



Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Test history: Ask Anna

Logout

All	Smoke	None-Smoke			
Test	Date	Status			
TestLogin23	2014/03/04-22:22	Passed	2	1	0

View

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Test history: Ask Anna

Logout

All	Smoke	None-Smoke			
Test	Date	Status			
SmokeStandard	2014/03/04-23:59	Passed	77	0	0
SmokeStandard	2014/03/04-18:59	Failed	3	0	1
SmokeStandard	2014/03/04-12:59	Failed	5	2	3

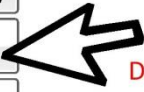
View
View
View

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Smoke test for Ask Anna
2014-03-04 18:59



Dependency 1335 was down at 2014/03/04-17:59

Test			Status
Initialize	A initialize: OK	B initialize: OK	Passed
Messaging	A MSG: OK	B: MSG OK	Passed
Save Log	Save Log to DB: Failed		Failed
Exit	Disconnect A: OK	Timer close	Passed

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

New

Testset

Browse

Run at time

Category name:

Add

Start time

End time (def 00:00)

Frequency (min)

Add

Remove time

Schedule

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

New

Testset

Smoke test
Non-smoke test
Regression test

Browse

Run at time

▲

▼

Add multiple time

Start time

End time (def 00:00)

Frequency (min)

Remove time

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

Smoke test

New

Testset

Smoketest123.file

Browse

Run at time

▲
▼

Remove time

Add single time

09:00 Add

Add multiple time

Start time

End time (def 00:00)

Frequency (min)

Add

Schedule

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

Testset

Run at time

09:00 ▲

▼

Add single time

Add multiple time

Start time

End time (def 00:00)

Frequency (min)

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

Testset

Run at time

- 09:00
- 12:00 (r)

Add single time

Add multiple time

Start time

End time (def 00:00)

Frequency (min)

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

New

Testset

Browse

Run at time

09:00 ▲

12:00 (r) ▼

Remove time

Smoketest123.file ✕

Smoke test

Starts at 12:00

Ends at 18:00

Repeated every 20 minutes

Start time

End time (def 00:00)

Frequency (min)

Add

Schedule

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

New

Testset

Browse

Run at time

09:00 ▲

12:00 (r) ▼

Remove time



Add single time

Add multiple time

Start time

End time (def 00:00)

Frequency (min)

Add

Schedule

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Logout

Scheduler: Anna

mars 2014						
m	t	o	t	f	l	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Category

New

Testset

Browse

Run at time

09:00

▲
▼

Remove time

Add single time

Add multiple time

Start time

End time (def 00:00)

Frequency (min)

Schedule

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	External Dependencies	Internal Secondary	Create new	
ID	Name	Status		
1332	Serv Q	Passed/2014/03/02-13:33	View	Delete
1335	MSSQL30002	Passed/2014/03/02-12:00	View	Delete
1337	Win Z	Attention /2014/03/02-12:22	View	Delete

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	External Dependencies	Internal Secondary	Cre
ID	Name	Anna Status	
2345	Ask Kalle	Passed /2014-03-04 23:59	View

This system is marked as "ongoing development"



- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	External Dependencies	Internal Secondary	Create new	
ID	Name	Status	Dependent	
1332	Serv Q	Passed /2014/03/02-13:33	None	View
1335	MSSQL30002	Passed /2014/03/02-12:00	MSSQL SERV1	View
1337	Win Z	Attention /2014/03/02-12:22	None	View

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Dependency: Ask Anna

Logout

Choose Category

MSSQL

MySQL

LINUX

Choose Object

MSSQL30003

MSSQL30004

MSSQL30005

Accept
Cancel

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Dependency: Ask Anna

Logout

Internal Dependencies	External Dependencies	Internal Secondary	Create	
ID	Name	Status		
1332	Serv Q	Passed/2014/03/02-13:33	View	Delete
1335	MSSQL30002	Passed/2014/03/02-12:00	View	Delete
1337	Win Z	Attention /2014/03/02-12:22	View	Delete
1339	MSSQL30005	Passed/2014/03/02-12:00	View	Delete

Back

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Test set 1337 2014/03/02-12:22

Logout

Test			Status
Ping1	0 % loss: OK	32 x 4 KBITS	Passed
Ping2	0 % loss: OK	32 x 4 KBITS	Passed
Respons time	10ms: HIGH		Attention

Back

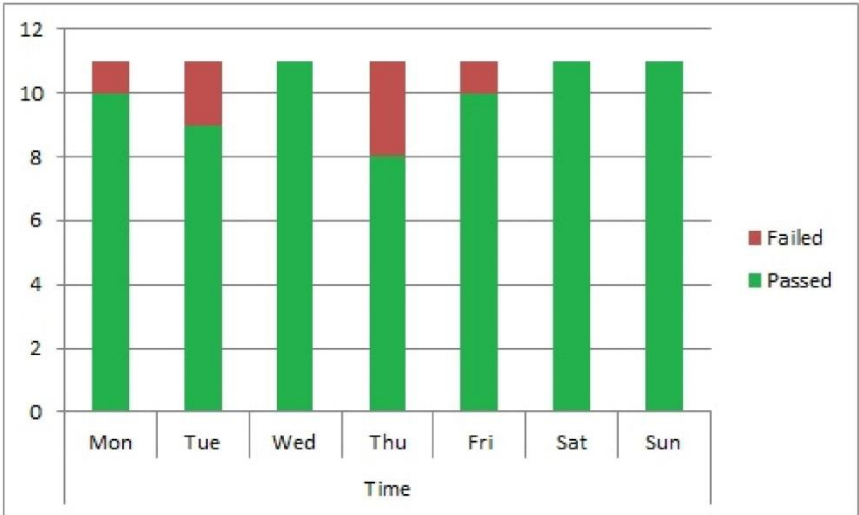
- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Statistics: Ask Anna

Logout

-  Week
- Month
- Year

<- W. 17 ->



Email settings

Send reports by email

Send report last day of every Week Month Year

E-mail adress

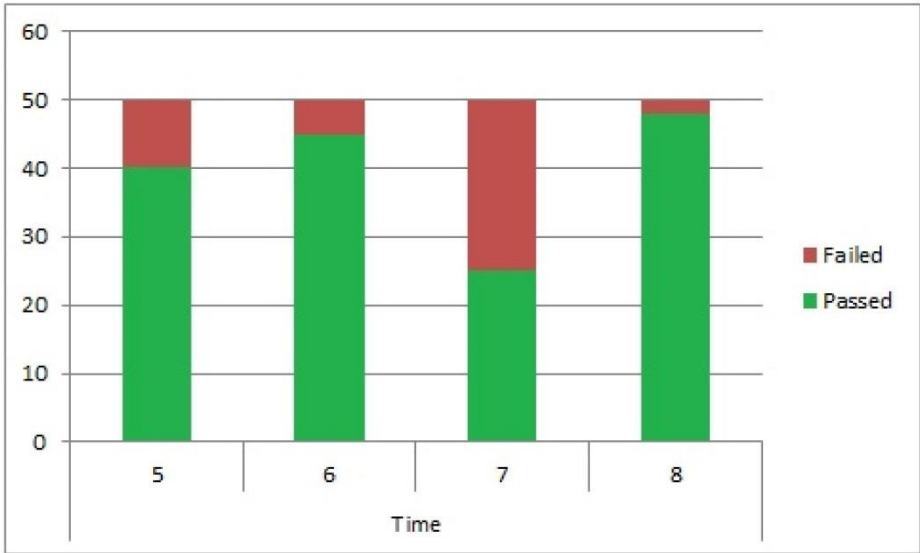
- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Statistics: Ask Anna

Logout

- Week
- Month
- Year

<- Feb ->



Email settings

Send reports by email

Send report last day of every Week Month Year

E-mail adress

- Overview
- Test History
- Scheduler
- Dependency
- Statistics
- Home

Statistics: Ask Anna

Logout

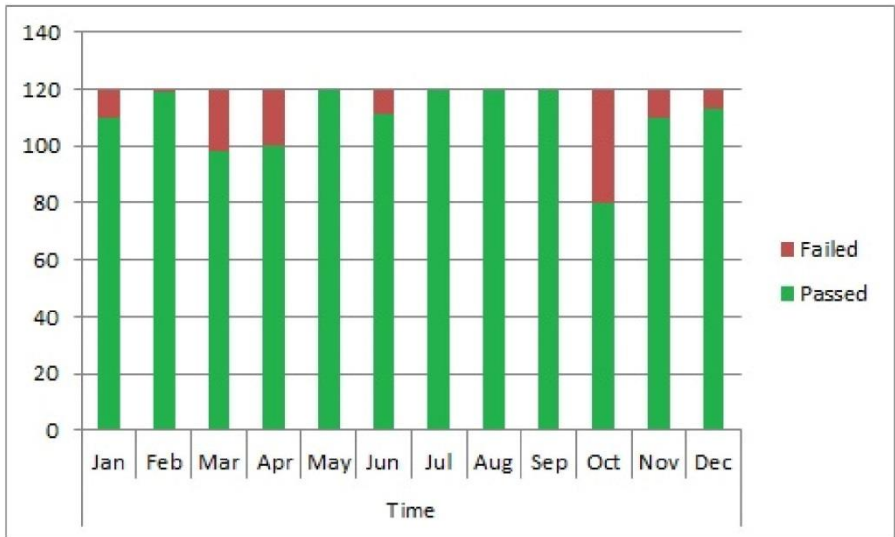
Week

Month

Year



<- 2013 ->



Email settings

Send reports by email

Send report last day of every Week Month Year

E-mail adress

Home

3D kitchen

Passed 2014/03/02-15:05

Ask Anna

Passed 2014/03/02-15:00

IKEA Family

Failed 2014/03/02-10:05

Logout



Username

Enter Text

Password

Enter Text

Login

Appendix C – System requirement specification

Software Requirement Specification

Portal for test environment in IKEA IT

Faruk El-zoubi & Irfan Agovic

<i>Version:</i>	2
<i>Date of change:</i>	2014-03-31
<i>Date of creation:</i>	2014-03-25

Table of content

1. Introduction	3
2. Background and goal	3
3. Terminology	5
4. Functional requirements	
4.1. General requirements	6
4.2. Data requirements	6
4.3. User stories	7
4.4. User interface	12
5. Quality requirements	
5.1. General quality requirements	12
5.2. Delivery requirements	12
6. Conditions and restrictions	
6.1. Requirements on the target machine	13
6.2. Restrictions of the system	13
7. Agreement	14

History of changes

<i>Vers</i>	<i>Date</i>	<i>Time</i>	<i>Responsible</i>	<i>Change</i>	<i>Reason</i>
1	2014-03-25	14:00	F.E & I.A	Document created	
2	2014-03-31	18:00	F.E & I.A	Requirements and user stories.	Validation with customer lead to new requirements, change of user stories and new user stories.

1 Introduction

This document gathers and specifies the requirements for a portal that is to be developed under a bachelor thesis at Lunds Faculty of Engineering, Campus Helsingborg. The owners of the project is IKEA IT that together with the contractors set the goal of the portal.

2 Background and goal

At the time of this document being written, the client already has existing software and organization available for executing automated tests for the products within their occupation. In contrast, they lack the possibility of monitoring the test environments at an application level. The client wishes to extend the function of the test environments in a way that the health of the environments can be monitored through a portal. Several ideas regarding additional functionality have blossomed from this basic concept. These ideas have been brought up and processed throughout the analysis that has been made during the project.

The analysis has consisted of multiple brainstorming meetings between the client and the contractors. The thoughts around the functionality were then gathered into a wireframe prototype. This was later presented in seven qualitative interviews to both confirm the targeted goal and gather additional thoughts from the stakeholders. The stakeholders were handpicked by the client. Since the portal is not a open market product, the client holds the

3

best insight in appropriate candidates. When the interviews were completed, all voice recordings were transcribed and the data was processed based on a phenomenographical (qualitative) data analysis. Based on this data the wireframe was adjusted to a new version which was presented to the client. The client had some feedback regarding the new wireframe which lead to further adjustments, resulting in a final version.

It is based on the final version of the wireframe prototype of which this requirement specification is made. Additional specification has been applied to give a clearer picture of the several functions.

The client and the contractor have under a verbal understanding reached an agreement that the primary goal of both the wireframe and this requirement specification is to give a broad recommended solution with roots to the stakeholders wishes and needs. The secondary goal is to implement the reached recommended solution. It is also understood that most likely every functionality will not be implemented during this project, since the project has a fixed timeframe.

3 Terminology

Portal	The portal is the web based application that holds the user interface of the assignment.
Client	The client is the person assigned by IKEA IT to be the supervisor of the project.
Owner	The owner of the project is the IKEA IT organisation.
Contractors	The contractors are those who have been assigned to carry out the mission of developing the portal, including analyses.
Wireframe prototype	A wireframe is a method used to display an idea visually. In this thesis the focus of the wireframe is to show functionality in combination of navigation.
System	A system is an application that IKEA owns and is launched or are preparing to be launched. Applications can vary from web to smartphone application.
User	A user is one that has an user account and can access the portal. A user can have permission to handle multiple systems.
Testset	Testset is a set of multiple tests called subtests. A testset can thereby contain multiple tests at one run.
System check	A system check is an executed testset that runs specific functional testing on the test environment to determine the systems health.
Dependency	A dependency is when an object A is dependent on object B, or contrariwise.
Primary dependency	A dependency is when an object A is dependent on object B, or contrariwise. The dependency is primary which means it is the first in level of dependencies.
Secondary dependency	A dependency is when an object A is dependent on object B, or contrariwise. The dependency is secondary which means it is the second in level of dependencies.
Human errors	Human errors are those errors that have occurred due to human mistakes or misunderstanding and not faulty source code.
User account	Username and password which specifies an user.

4 Functional requirements

4.1 General requirements

Requirement 4.1.1 Every word in the portal shall be written in English.

Requirement 4.1.2 The portal shall be compatible with Microsoft Internet Explorer 9.

4.2 Data requirements

Requirement 4.2.1 The portal shall give a message incase of human errors.

Requirement 4.2.2 The client user interface shall be shall be written in HTML.

Requirement 4.2.3 The client user interface shall be shall be written in CSS.

Requirement 4.2.4 The client source code shall shall be written in the Javascript language.

Requirement 4.2.5 The server source code shall be written in Nodejs.

Requirement 4.2.6 The server shall use the MongoDB database.

4.3 User stories

General

User story 1 As a user I want to be able to navigate to home, test history, scheduler, dependency and statistics after i have logged in and chosen a system.

User story 2 As a user I want to be able to log out at every page after I have logged in.

Login

User story 3 As a user I want to be able to access the portal with a given username and password, and then be redirected to my home page.

Home

User story 4 As a user I want to be able to see a list of all systems I am a part of, and see the status of every system.

- A. A status text shall be presented at the side of every entry in the list with the date of the last status check runned on the system.
- B. The status text shall be presented in green colour if it has the status of "Passed".
- C. The status tet shall be presented in red colour if it has the status of "Failed".

User story 5 As a user I want to be able to access a specific system in the list of systems. Then be redirected to the overview page of the chosen system.

Overview

User story 6 As a user I want to be able to see the status of the system.

- A. The status of the system shall be displayed with text as "Good" and have a green column background if the latest system check was passed.
- B. The status of the system shall be displayed with text as "Attention" and have a yellow column background if the latest system check was passed with warning.
- C. The status of the system shall be displayed with text as "Down" and have a red column background if the latest system check failed.

User story 7 As a user I want to be able to see a list of the latest system checks and be able to access the report of all system checks in the list.

- A. Date and time shall be presented in every entry in the list.
- B. The result of the system check shall be displayed with numbers of passed subtests divided by the number of total subtests, and the result of the system check shall be displayed with text as "Passed". The text and quota shall be green if the latest system check was passed.

- C. The result of the system check shall be displayed with numbers of passed subtests divided by the number of total subtests, and the result of the system check shall be displayed with text as "Passed with warning". The text and quota shall be yellow if the latest system check was passed with warning.
- D. The result of the system check shall be displayed with numbers of passed subtests divided by the number of total subtests, and the result of the system check shall be displayed with text as "Failed". The text and quota shall be red if the latest system check failed.
- E. There shall be a link at the side of every entry in the list that redirects the user to the system check report.
- F. If the system had a dependency in which the latest system check was failed at the time that the user system check was performed, there shall be a notification symbol at the side of the list entry.
- G. When the user hovers its mouse over the notification symbol there shall be a description of the symbol displayed.

User story 8 As a user I want to be able to create a link that is directed to the overview page of the system I am currently viewing.

- A. The link shall redirect anybody who clicks it to a page with the same information that the overview page of the system shows, but with no write rights. There shall not be any buttons on this new page.

User story 9 As a user I want to be able to create or edit a public status text of the system I am currently viewing.

- A. The public status text shall represent the state of which the system is in to the rest of the systems in the portal.
- B. The public status text shall be left blank to represent the normal state of the system.

User story 10 As a user I want to be able to have an e-mail sent to me containing the report of the latest system check every time it is performed, or every time the latest system check has failed.

- A. The user shall be able to put in multiple email addresses that shall receive the report.
- B. In the sent mail including the report there shall also be an option to remove oneself from the list.

Test History

User story 11 As a user I want to be able to see all my runned tests in a list and to be able to filter the tests by choosing a category.

- A. The title of the test, date and time shall be presented in every entry in the list.
- B. The result of the test shall be displayed with numbers of passed subtests passed with warning subtest and failed subtests. In the same order the numbers shall have green, yellow and red text colour.
- C. The result of the system check shall be displayed with text as "Passed"and the background of the column shall be green if the test was passed.
- D. The result of the system check shall be displayed with text as "Passed with warning"and the background of the column shall be yellow if the test was passed with warning.
- E. The result of the system check shall be displayed with text as "Failed"and the background of the column shall be red if the test was failed.
- F. There shall be a link at the side of every entry in the list that redirects the user to the test report.
- G. If the system had a dependency in which the latest system check was failed at the time that the user system check was performed, there shall be a notification symbol at the side of the list entry.
- H. When the user hovers its mouse over the notification symbol there shall be a description of the symbol displayed.
- I. If the user is viewing a report of a system check with a notification symbol, there shall be a text presented in red color with information about the dependency. The information shall contain the dependency ID and the date and time of which its latest system check was performed.

Scheduler

User story 12 As a user I want to be able to schedule a test.

- A. The user shall be able to browse for a test set.
- B. The user shall be able to choose a category for the scheduling and make a new category if needed. There should be always be default categories, "smoke test" and "none smoke test"
- C. The user shall be able to choose either one or several dates from a calendar on which the selected test set shall be executed.
- D. The user shall be able to specify either one or several times of the specified date/dates at which the selected test set shall be executed.

- E. The user shall be able to add repetitive times by specifying start time and frequency (in minutes). The user shall be able to schedule a test set to run every day.
- F. At the time the user is giving input for a new scheduling, the user shall be able to remove a time from the time list.

User story 13 As a user I want to be able to see and change scheduled tests.

- A. The user shall be able to see list of all scheduled test sets. In every entry there shall be information regarding the scheduled test set, consisting of dates, times, name and category of the test set.
- B. The list shall only contain test sets that are scheduled to run or are already running. No test set which last date of execution has passed the current date shall be included in the list.
- C. The user shall be able to pause, resume and remove a test set that has already been scheduled.
- D. The user shall be able to filter scheduled tests in the list.

Dependency

User story 14 As a user I want to be able to see and adjust my internal dependencies.

- A. The internal dependencies shall be presented in a list.
- B. Every entry shall contain information regarding the dependency such as id, name and status. The status is based upon the latest system check for the dependency.
- C. The status of the dependency shall be displayed with text as "Passed" and have a green column background if the latest system check was passed.
- D. The status of the dependency shall be displayed with text as "Attention" and have a yellow column background if the latest system check was passed with warning.
- E. The status of the dependency shall be displayed with text as "Failed" and have a red column background if the latest system check failed.
- F. The user shall be able to create a new dependency by choosing a category and an object from that category. The new dependency shall then be added to the list of dependencies.
- G. The user shall be able to remove an entry from the list.
- H. The user shall be able to view the report of the dependencies latest system check.

User story 15 As a user I want to be able to see my external dependencies.

- A. The external dependencies shall be presented in a list.

- B. The list of external dependencies shall not be manually applied but automatically. The list shall show external systems that have assigned the user system as an internal dependency.
- C. Every entry shall contain information regarding the dependency consisting of id, name.

User story 16 As a user I want to be able to see my internal secondary dependencies.

- A. The internal secondary dependencies shall be presented in a list.
- B. Every entry shall contain information regarding the primary dependency such as id, name and status. The status is based upon the latest system check for the dependency.
- C. The status of the primary dependency shall be displayed with text as “Passed” and have a green column background if the latest system check was passed.
- D. The status of the primary dependency shall be displayed with text as “Attention” and have a yellow column background if the latest system check was passed with warning.
- E. The status of the primary dependency shall be displayed with text as “Failed” and have a red column background if the latest system check failed.
- F. The user shall be able to view the report of the primary dependencies latest system check.
- G. For every primary dependency the user shall be able to see a secondary dependency.

User story 17 As a user I want to be able to see a systems state.

- A. For every dependency, either internal or external. There shall be a symbol at the side of the entry in the list, if the system has a specific state. If the system does not have a specific state (default mode) then there shall be no symbol.
- B. When the user hovers its mouse over the symbol there shall be a description of the state displayed.

Statistics

User story 18 As a user I want to be able to see statistics of the system’s status.

- A. Statistics shall be made based on system checks.
- B. Statistics shall be presented in bar diagrams.
- C. Statistics shall show passed and failed system checks.
- D. The user shall be able to view statistics week-wise, month-wise and year-wise.

User story 19 As a user I want to be able to receive a report on statistics by e-mail.

- A. The user shall be able to put in multiple email addresses that shall receive the report.
- B. In the sent mail including the report there shall also be an option to remove oneself from the list.
- C. The user shall be able to decide how often a report shall be sent to the user, weekly, monthly or yearly.

4.4 User interface

Requirement 4.4.1 The user interface shall be based upon the navigation of the version 4.1 of the wireframe (IKEA Wireframe UI V4.1).

Requirement 4.4.2 The user interface shall be based upon the functionality of the version 4.1 of the wireframe (IKEA Wireframe UI V4.1).

Requirement 4.4.3 The user interface shall not consider the graphics of the version 4.1 of the wireframe (IKEA Wireframe UI V4.1).

5 Quality requirements

5.1 General quality requirements

Requirement 5.1.1 There shall be at least 3 successful test cases. By successful meaning that all features that are implemented are proven to be fully functional. The test cases are considered fully functional when the expected results and actual results are equivalent.

5.2 Delivery requirements

Requirement 5.2.1 All source code shall be delivered to the client.

Requirement 5.2.2 The latest version of the wireframe shall be delivered to the client.

Requirement 5.2.3 A document containing further development ideas shall be delivered to the client, incase there are further developing ideas.

Requirement 5.2.4 The final report shall be delivered to the client.

Requirement 5.2.5 The final report shall be written in English.

Requirement 5.2.6 The final report shall be a PDF document.

Requirement 5.2.7 A public REST API shall be delivered to the client.

Requirement 5.2.8 The REST API shall be documented with comments.

Requirement 5.2.9 A video user manual shall be delivered to the client.

Requirement 5.2.10 The requirement specification document shall be delivered to the client.

6 *Conditions and restrictions*

6.1 *Requirements on the target machine*

Requirement 6.1.1 To enter the portal one must have a user account.

Requirement 6.1.2 To enter the portal one must be connected to the IKEA network.

Requirement 6.1.3 To enter the portal one must use the internet browser Microsoft Internet Explorer 8.

6.2 *Restrictions of the system*

None

Appendix D – Questionnaire

1. Which department are you currently working in?

2. What is your profession in your current workplace?

3. What software do you mostly use to test a product with, as of today?

3.1. Where are the test scripts that you use for testing a product stored, in the software that you use the most as of today?

3.2. What does testing of a product result into? E.g. a html page, data in a text document etc:

3.3. Where are the results of a test execution stored? E.g. e-mail account, database, repository etc:

3.4. How do you or another partner analyze the test results from testing of a product, and what is interesting in the test result?