



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

---



# **Software Market**

Combining User Experience Design and Data Protection  
in an Agile Software Development Process

Bachelor of Science Thesis in Computer Science and Engineering

DAVID GUSTAFSSON, ARYAN IRANZAMINI, HENRIK MÖLLER,  
ANTON SJÖHOLM AND ANNA SKOGLUND

BACHELOR OF SCIENCE THESIS, CHALMERS UNIVERSITY OF TECHNOLOGY  
SOFTWARE MARKET - USER EXPERIENCE DESIGN AND DATA PROTECTION IN AN AGILE  
SOFTWARE DEVELOPMENT PROCESS

DAVID GUSTAFSSON  
HENRIK MÖLLER  
ARYAN IRANZAMINI  
ANTON SJÖHOLM  
ANNA SKOGLUND

Copyright © David Gustafsson, June 2016

Copyright © Henrik Möller, June 2016

Copyright © Aryan Iranzamani, June 2016

Copyright © Anton Sjöholm, June 2016

Copyright © Anna Skoglund, June 2016

Examiner: Arne Linde

Chalmers University of Technology University of Gothenburg  
Department of Computer Science and Engineering  
SE-412 96 Göteborg  
Sweden  
Telephone + 46 (0)31-772 1000  
Department of Computer Science and Engineering  
Göteborg, Sweden June 2016

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

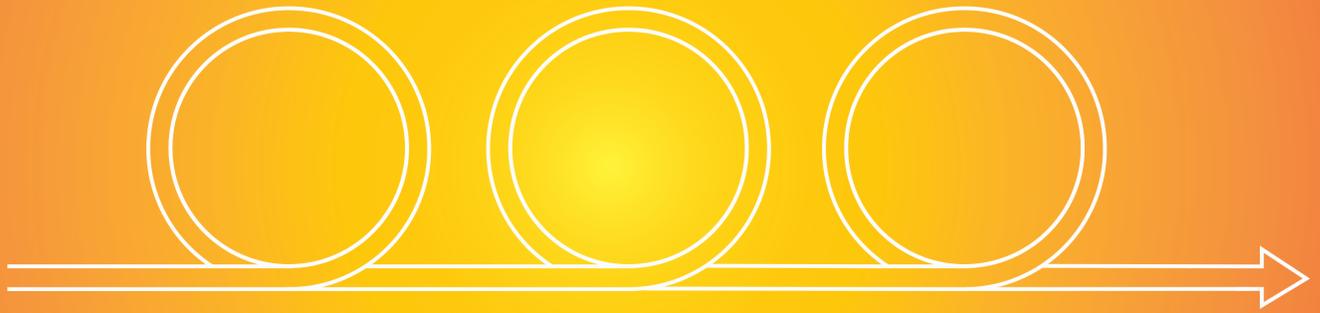
The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Cover:

*Agile Revolutions* created by Anna Skoglund ©

Department of Computer Science and Engineering  
Göteborg 2016

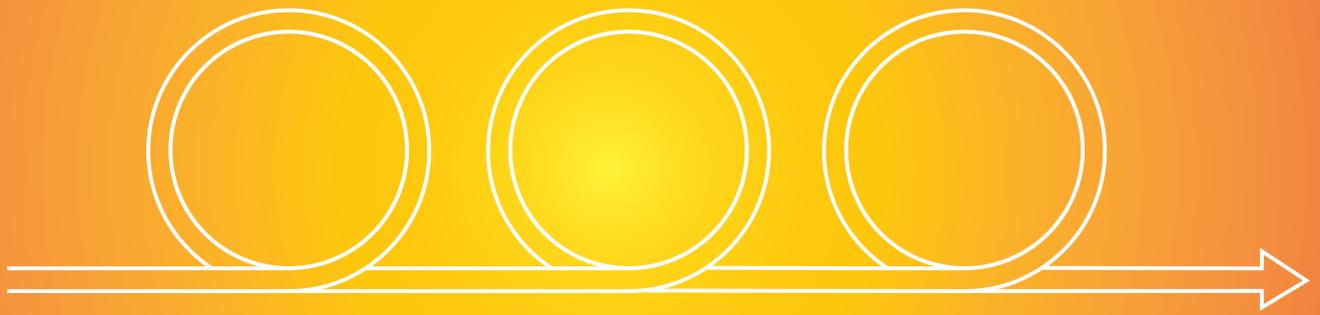
*First release, June 2016*



## Abstract

The purpose of this thesis was to study how an Agile software development methodology would affect user experience design and data protection in a software development process. This was accomplished through the development of a platform called Software Market. The Software Market was created for the Gothenburg based company Software Skills, and its purpose is to connect companies and developers worldwide for shorter IT-projects. The Software Market was developed alongside a bachelor's and a master's thesis project. The process started out organized through the use of Scrum. Even though Scrum is an Agile practice, it was discarded in favor of a more fluid approach to Agile. This was necessitated by the complex and rapidly changing development environment where the other thesis groups were designing and developing related platforms. The user experience design benefited from this fluid approach to Agile due to its ability to rapidly respond to changing requirements. The added stress of evolving design considerations was overshadowed by the ability to quickly reach new design goals. The work with data protection can not be said to have benefited from the fluid Agile approach in the same manner. Since the work on the Software Market was production focused the security considerations tended to become secondary concerns. This was alleviated through the creation of the security design document that enabled the individual developers to implement a secure design in a more independent manner. Still, security would likely have benefited from the use of a more rigidly organized method. Overall we found that the Agile methodology was well suited for this project. Considering the many interactions and connections with other groups, we even deem it a necessity.

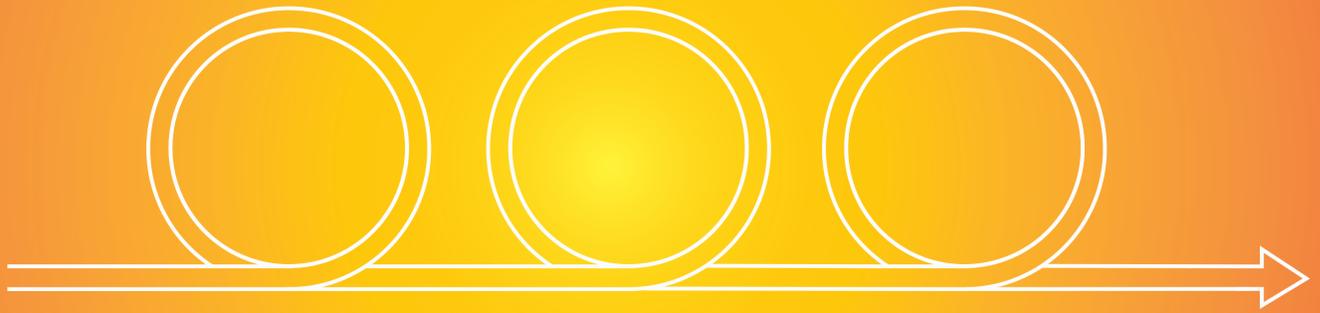
Keywords: Agile, user experience design, computer security



## Sammandrag

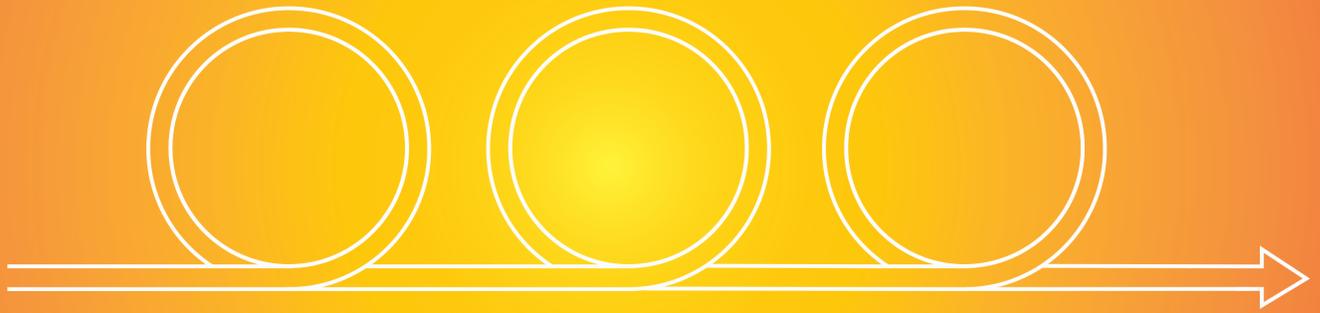
Syftet med detta projekt var att undersöka hur ett Agilt arbetssätt skulle påverka processen kring utvecklandet av design och datasäkerhet i ett mjukvaruprojekt. Detta genomfördes genom att skapa en webbapplikation, Software Market. Software Market skapades åt Software Skills, vilket är ett företag som är baserat i Göteborg. Syftet med utvecklingen av denna webbapplikation var att ge möjligheten åt företag och frilansare att enkelt kollaborera över internet. Utvecklingen av Software Market skedde parallellt med ett annat kandidat och masterarbete. De Agila arbetsmetoder som användes var bland annat Scrum vilket senare övergavs till förmån för en mer anpassningsbar Agil process. Detta nödvändiggjordes av de komplexa och snabba förändringarna i utvecklingsmiljön, där de andra examensgrupperna utvecklade angränsande system. Den Agila metodens förmåga att snabbt anpassa sig till nya krav gynnade processen kring designen. Denna rörliga metod tillförde dock en ökad belastning på designarbetet som ändå överskuggades av förmågan att snabbt kunna uppnå nya designmål. Det är svårt att avgöra om arbetet med informationssäkerhet gynnades på samma sätt av den rörliga metoden. Fokuset kring utvecklingen av Software Market lades mestadels på produktion, vilket innebär att säkerhetsaspekterna kom i andra hand. Det lindrades genom skapandet av ett säkerhetsdokument som möjliggjorde implementationen av säkerhet på en individuell nivå. Dock så skulle arbetet kring informationssäkerheten sannolikt kunna gynnas av en mer organiserad metod. Som helhet fann vi dock att Agila arbetsmetoder passade bra för detta projekt. Med tanke på interaktionerna och samarbetet med de andra grupperna så anser vi det även vara en nödvändighet.

Nyckelord: Agilt, design, datasäkerhet



## Acknowledgements

We would like to give thanks to our supervisor Emil Axelsson and our examiner Arne Linde for their help and support with this Bachelor's thesis. Further we would like to thank Henrik Enström, Luuk van Egeraat, Anna Weiss and Johannes Lundqvist for their guidance in developing the Software Market. We would also like to acknowledge all participants in our tests and interviews for all valuable feedback and insights. We would not have been able to do this without you.



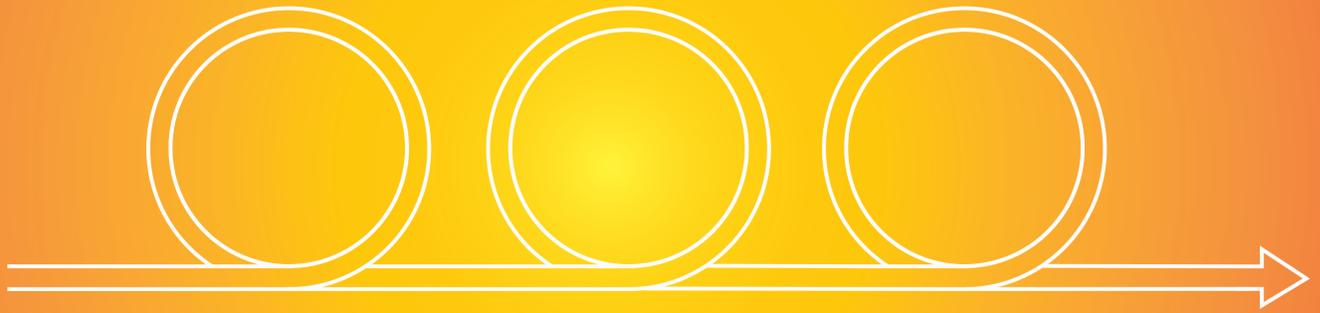
# Contents

<b>1</b>	<b>Introduction</b> .....	<b>9</b>
1.1	Background	
1.2	Software Skills	
1.3	Project Assignment	
1.3.1	Development Specifications .....	10
1.4	Project Limits	
1.5	Purpose	
<b>2</b>	<b>Theory</b> .....	<b>11</b>
2.1	Databases	
2.1.1	NoSQL .....	11
2.1.2	Data Modeling .....	12
2.2	Coding	
2.3	Front End	
2.3.1	AngularJS .....	12
2.3.2	Angular Material .....	12
2.3.3	HTML5 and SCSS .....	12
2.4	Back End	
2.4.1	Node.js .....	13
2.4.2	WebSockets .....	13
2.5	Security	
2.5.1	Importance of Security .....	13

2.5.2	Open Web Application Security Project	13
2.5.3	OWASP Top Ten	14
2.5.4	Application Security Verification Standard	14
<b>2.6</b>	<b>User Experience Design</b>	
2.6.1	Usability	14
2.6.2	Benchmarking	15
2.6.3	Interviews	15
2.6.4	KJ-analysis	16
2.6.5	List of Demands	16
2.6.6	Expression Board	16
2.6.7	Wireframing	16
2.6.8	Task Flow Analysis	16
2.6.9	Usability Tests	16
2.6.10	Theoretical Evaluation Methods	17
2.6.11	Cognitive Walkthrough	17
2.6.12	Predictive Human Task Analysis	17
<b>2.7</b>	<b>Work Method</b>	
2.7.1	Agile	17
2.7.2	Scrum and Sprint	18
<b>3</b>	<b>Method</b>	<b>19</b>
<b>3.1</b>	<b>Work Process</b>	
<b>3.2</b>	<b>Software Skills</b>	
<b>3.3</b>	<b>Coding</b>	
<b>3.4</b>	<b>Front End</b>	
<b>3.5</b>	<b>Back end</b>	
3.5.1	Handlers	20
3.5.2	Database	20
<b>3.6</b>	<b>Security</b>	
3.6.1	Security Design Document	21
3.6.2	Evaluation	21
<b>3.7</b>	<b>Design</b>	
<b>4</b>	<b>Implementation</b>	<b>22</b>
<b>4.1</b>	<b>Work Process</b>	
4.1.1	Scrum	22
<b>4.2</b>	<b>Coding</b>	
<b>4.3</b>	<b>Security</b>	
<b>4.4</b>	<b>Design Methods</b>	
4.4.1	Interviews	23
4.4.2	KJ-analysis	23

---

4.4.3	Wireframes	23
4.4.4	Task Flow Analysis	24
4.4.5	Cognitive Walkthrough	24
4.4.6	Predictive Human Error Task Analysis	24
4.4.7	Usability Tests	24
4.4.8	Boards	24
4.4.9	Final Design	24
<b>5</b>	<b>Results</b>	<b>25</b>
<b>5.1</b>	<b>Did User Experience Design Benefit from Agile Practices?</b>	
<b>5.2</b>	<b>Data Protection with Agile Practices</b>	
<b>5.3</b>	<b>Project Work and Agile Practices</b>	
<b>5.4</b>	<b>Software Market</b>	
5.4.1	Profile	26
5.4.2	Timelog	28
5.4.3	Contract	30
5.4.4	Integration	33
<b>6</b>	<b>Discussion</b>	<b>34</b>
<b>6.1</b>	<b>Legal framework and further studies</b>	
<b>6.2</b>	<b>Front End</b>	
<b>6.3</b>	<b>Back End</b>	
<b>6.4</b>	<b>Security</b>	
<b>6.5</b>	<b>Design Method and Implementation</b>	
6.5.1	Interviews	35
6.5.2	Benchmarking	36
6.5.3	Expression Board	36
6.5.4	Task Flow Analysis	36
6.5.5	Usability Tests	36
6.5.6	Cognitive Walkthrough	37
6.5.7	Predictive Human Task Error Analysis	37
<b>6.6</b>	<b>Final Designs</b>	
6.6.1	Profile	37
6.6.2	Timelog	38
6.6.3	Contract and Contract Details	38
<b>6.7</b>	<b>Agile and Collaboration</b>	
<b>6.8</b>	<b>Scrum</b>	
<b>7</b>	<b>Conclusion</b>	<b>40</b>
<b>A</b>	<b>Security Design Document</b>	<b>44</b>



# 1. Introduction

## 1.1 Background

Efficient, secure and user friendly IT-solutions have become absolute necessities for any company wishing to conduct a business today. This widely affects the global market in terms of an increasing demand for specialists such as programmers, business analysts and UX-designers (User experience designers).

The company Software Skills submitted a bachelor thesis which entailed the development of a platform where talents within the IT-sector and companies will have an opportunity to meet and make great ideas become a reality.

The platform should offer a possibility to establish secure and usable connections between future employers and people with the ability to bring great ideas to life. The platform also had to enable collaborative work where the code and design can be seen by both creators and employer.

## 1.2 Software Skills

Software Skills is a company, situated in Gothenburg, that works with connecting talented software engineers and companies. Their CEO Henrik Enström served as our supervisor at the company. They already had a website containing several tools which can be used to further the platform we created (multiple-choice tests, personality tests, HoneyPot etc.).

## 1.3 Project Assignment

Our assignment was to develop a basic foundation for a software platform that would enable IT companies and freelancers with IT expertise to collaborate. This product had to be well designed, secure and developed through the Agile methodology.

### 1.3.1 Development Specifications

Software Skills specified that the platform had to be built with certain environments and frameworks because it was going to be maintained and further developed by them. The specifications were to use the following:

- MongoDB as the database
- HTML and SCSS to build and style static pages
- JavaScript and AngularJS for dynamic views in pages
- NodeJS for server-side programming
- Websockets for communication between client and server

This gave us the ability to use their current website as a foundation for our software platform since they use the same structure.

### 1.4 Project Limits

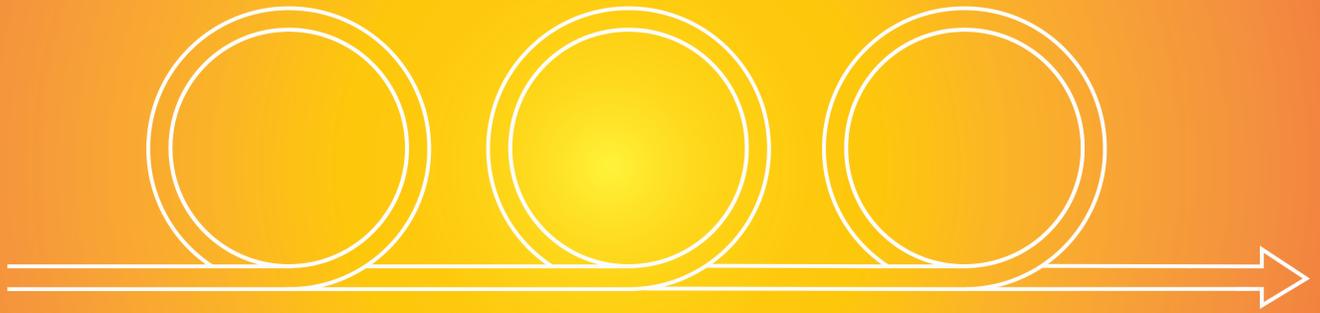
There were two groups collaborating with Software Skills. This group's focus was on the applicant side of the Software Market platform. Therefore the defined scope of our project did not include the Applicant Tracking System (ATS) part of the Software Skills website, nor the parts of the Software Market used to hire developers and administrate payments. Furthermore, the project was not expected to result in a site immediately available to users but in a usable foundation with basic functionality.

Legal aspects concerning trade union rights, job security and other benefits were also beyond the scope of the project. Given the time restrictions we could not focus on the total user experience. Within this area we chose to focus on the usability and utility of the graphical interface on the pages we developed.

### 1.5 Purpose

The purpose of this report is twofold: to describe and discuss the results of the project assignment, and to investigate how to combine user experience design and data protection in an Agile software development process. This investigation is accomplished through answering the following questions, based on our experiences:

- Will user experience design benefit or be hampered by Agile practices?
- Will data protection benefit or be hindered by Agile practices?
- Will our project assignment function well as a case study of Agile practices?



## 2. Theory

The theory section aims to explain the theory behind the different tools and methods that were used to create the Software Market.

### 2.1 Databases

In a large application, large amount of data is often required to be stored. Since the amount of data is usually too large to keep in a folder or a file, a database is used to organize the information and speed up queries. To ensure maximum performance in terms of scalability and reliability, efficient data models and queries are required [1]. The scalability and reliability of the system is important as we expect the system to grow large. This is because the system would be too slow if it was not able to scale up properly and plagued with errors if it was not reliable .

#### 2.1.1 NoSQL

As specified earlier, we used MongoDB which is a document oriented NoSQL database [2]. MongoDB differs from typical SQL databases such as Oracle and MySQL in terms of data structure, possible operations on the data, schema requirements etc. In relational databases a defined schema is required before it can be used. This does not fit along with our Agile software development since we work iteratively with the application, and therefore often need to change the structure of the database. A relational database would thus require us to make a new schema and thereafter migrate the old database to the new schema after every change, making the use of a NoSQL database more beneficial to us.

### 2.1.2 Data Modeling

Data modeling is an important aspect of the back end (back end is everything that is server-side, for example a database) since it determines how fast and scalable the database will be. Since there are no JOIN operations in NoSQL databases, common practice is to denormalize information to increase performance [3]. This will increase the total data volume but significantly decrease the processing complexity. It is also common to nest data in document-oriented databases such as MongoDB. This lets us retrieve information from the database with a single operation thus minimizing the amount of queries needed.

## 2.2 Coding

Since Software Skills already had both front end (front end is everything that is client-side, for example the user interface) and back end code up and running, we had to ensure that our application was constructed to be seamlessly integrable in to the whole system. The web application was therefore written in JavaScript, Node.js, AngularJS, HTML5 and SCSS. We used WebSockets as a mean of bidirectional communication. This combination allowed us to create a fast, responsive and feature-rich application that does not require reloading of the page. The code was combined with the graphics to create an aesthetically pleasing and easy to understand application.

## 2.3 Front End

### 2.3.1 AngularJS

AngularJS is a JavaScript framework with increased functionality [4]. It is able to extend HTML attributes with directives and bind data to HTML. This enabled us to display contents from the database directly in the web application.

### 2.3.2 Angular Material

Angular Material is described as "both a UI component framework and a reference implementation of Google's Material Design Specification" for AngularJS developers [5].

### 2.3.3 HTML5 and SCSS

HTML and CSS are the pillars of web development[6]. HTML defines what is on the page and CSS defines how the page looks. The latest version of HTML, HTML5, offers extensive utility and was therefore the natural choice. SCSS is an extension of CSS that offers better readability and usability. Usability is defined as "the extent to which a product can be used with effectiveness, efficiency and satisfaction by specific users to achieve specific goals in a specific environment" [7].

## 2.4 Back End

### 2.4.1 Node.js

Node.js is a JavaScript library that is an asynchronous, event driven framework which is designed to build scalable network applications [8].

### 2.4.2 WebSockets

WebSockets provide bidirectional communication between client and server [9]. This means is that the client will get immediate feedback from the server. This could be a new message, notification or database update.

## 2.5 Security

Computer security is often described by the acronym CIA. CIA is constructed from the concepts Confidentiality, Integrity and Availability. These concepts can be defined as follows [10]:

- "Confidentiality: Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.
- Integrity: Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.
- Availability: Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system."

### 2.5.1 Importance of Security

The number of cyber attacks targeting corporations and government agencies have risen steadily over the last few years [11] [12]. Therefore web security is increasingly a cause for concern among companies that do business over the Internet. For an application such as the Software Market this is undoubtedly true since it is a platform that enables a multitude of companies to conduct business online. An insecure application could be problematic for a company using the site, but catastrophic for Software Skills since the number of users of the Software Market could plummet.

### 2.5.2 Open Web Application Security Project

The Open Web Application Security Project (more commonly known as OWASP) is a not-for-profit international organization dedicated to "be the thriving global community that drives visibility and evolution in the safety and security of the world's software" [13]. Their members compile large amounts of guidelines and "cheat sheets" detailing best practices in secure web development. OWASP's flagship project is the OWASP Top Ten.

### 2.5.3 OWASP Top Ten

The OWASP top ten is a list compiled by OWASP containing the ten most dangerous vulnerabilities found in web applications [14]. The list is updated every third year, with a new list to be released this year. The current list was released in 2013. The list is used by organizations to identify the largest web based security threats. Examples of organizations using the lists include the NSA [15] and Microsoft [16].

### 2.5.4 Application Security Verification Standard

The Application Security Verification Standard (ASVS) is a list of security requirements and tests used to evaluate and verify the security standard of an application [17]. It is an OWASP project and the current version is version 3.0.

## 2.6 User Experience Design

User Experience (UX) is defined as “a person’s perceptions and responses that result from the use or anticipated use of a product, system or service” [18]. An additional note to this definition is that UX "is a consequence of brand image, presentation, functionality, system performance, interactive behavior and assistive capabilities of the interactive system, the user’s internal and physical state resulting from prior experiences, attitudes, skills and personality, and the context of use".

### 2.6.1 Usability

*For definition of usability, see section 2.2.1*

Patrick Jordan’s five additions to the ISO-standard definition for usability [19]:

- Guessability: "The effectiveness, efficiency and satisfaction with which specified users can complete specified tasks with a particular product for the first time"
- Learnability: "The effectiveness, efficiency and satisfaction with which specified users can achieve a competent level of performance on specified tasks with a product, having already completed those tasks once previously"
- Experienced User Performance: "The effectiveness, efficiency and satisfaction with which specified experienced users can achieve specified tasks with a specific product"
- System Potential: The optimum level of effectiveness, efficiency and satisfaction with which it would be possible to complete specified tasks with a product.
- Re-usability: "The optimum level of effectiveness, efficiency and satisfaction with which it would be possible to complete specified tasks with a product" .

There are many conventions and guidelines in terms of user experience design and usability. In this project we will adhere to Jordan’s five additions to the ISO-definition, Jordan’s ten usability heuristics and Nielsen’s usability heuristics.

Patrick Jordan’s ten usability heuristics [20]:

- Consistency: Solving similar tasks with the product/interface in a similar way.
- Compatibility: Solving tasks in a manner similar to how they are solved with similar products.
- Consideration of user resources: Consideration is taken to how the user is provided with information in order to prevent information overload.
- Feedback: Providing information to the user that the product has registered an action.
- Error prevention and recovery: Design for minimal risks of actually making an error, and present users with a chance to recover from any mistakes.
- User control: Design for maximum experienced control by the user, without unnecessary exaggeration.
- Visual clarity: Ensure that any information provided to the user is easy to understand and read.
- Prioritization of functionality and information: Design the product so that the most relevant information and functionality is easily accessible to the user.
- Appropriate transfer of information: Combine other technology relevant for the product to further improve the product's usability.
- Explicitness: Create clues in the interface/in the product so users can guess what it is used for.

Jakob Nielsen's five usability heuristics:

- Learnability: "How easy is it for users to accomplish basic tasks the first time they encounter the design?" (note: closely connected to Patrick Jordan's theory) [21].
- Efficiency: Resources expended in relation to the accuracy and completeness with which users achieve goals [7].
- Memorability: "When users return to the design after a period of not using it, how easily can they reestablish proficiency?" [21]
- Errors: "How many errors do users make, how severe are these errors, and how easily can they recover from the errors?"
- Satisfaction: "Freedom from discomfort, and positive attitudes towards the use of the product." [7]

### 2.6.2 Benchmarking

Benchmarking is a method used to research the market standard for similar solutions, and to obtain a point of reference [22].

### 2.6.3 Interviews

Semi-structured interviews is a method used to collect information in order to understand user behaviors, identify goals and potential problems [23]. They are often conducted with one interviewer and one interviewee at the time. The interviewer follows a manuscript but is allowed to deviate from the manuscript, letting the interviewee speak freely about subjects and questions. The interviewee is often a potential user of the specific product/interface or has experience within the field. An optimal amount of interviews is usually ten to twenty interviews, which is when you reach a "saturation point" and the marginal level of new information found in each interview is decreasing [24].

### 2.6.4 KJ-analysis

From interviews, relevant quotes are written on post-it notes. Statements and comments are categorized in similar topics and themes. These themes and topics are used as a base for formulating a list of demands [25].

### 2.6.5 List of Demands

A list of demands is a tool for different parties, e.g. client and a company, to form a mutual understanding about the project [26]. This lists the functionality of the product that is required. It is essential for use later on in the project to verify that demands are fulfilled before the product is delivered to the client.

### 2.6.6 Expression Board

An expression board is a tool that is part of the list of demands. Images and words are collected to a board (online or physical) that represent certain words or expressions. It is used to decide, communicate and visualize a specific characteristic. It can also be used as an inspiration throughout the design process [27].

### 2.6.7 Wireframing

Wireframe is "a visual representation of the structure of a web page" [28]. A wireframe shows the "basic size and placement of the screen component" and is used in the early stages of user-experienced design and in software development [29].

### 2.6.8 Task Flow Analysis

A task flow is used to identify if users will perform the correct tasks. Task flow analysis is "a methodology that focuses attention on users and on their tasks and goals" [30]. The analysis is usually represented in a flow chart. This is used to investigate each sequence of the tasks a user performs in order to achieve their goal. It can also be used to identify what user goals that depends on other goals [31].

### 2.6.9 Usability Tests

Usability test is a way to evaluate a product's usability and/or utility. Utility is a measure of a product's functionality, meaning how useful it is for reaching a user's goals. Tests can be used to identify difficulties in the interaction with the interface, compare different design solutions or to compare how a design meets certain requirements[32]. Specific and often representational tasks for usage of a product's interface are selected for users to perform in a test. The test person's behavior is then studied while performing these tasks. The aim is to detect how the user interacts with the interface. Tests can be performed in a lab or in the natural environment where the product is used.

### 2.6.10 Theoretical Evaluation Methods

Theoretical evaluation methods for man-interface interaction analysis consists of four steps [31]:

- Definition of the evaluation process
- Description of system
- Interaction analysis
- Evaluation and summation

In the first phase the purpose of the evaluation is determined. What machine will be used? Who is the user? How is the machine to be used, and in what context?

In second phase user goals, other objectives, tasks/assignments, interfaces and the interactions are identified and described. When all factors are determined, the interaction analysis is split in two parts: analysis on whether users will perform the correct tasks, and analysis to identify potential errors users can face. Finally results are evaluated and summarized.

### 2.6.11 Cognitive Walkthrough

This method investigates the steps taken by the user to reach the specific goal, and where errors occur in the process. This method give answers the two questions: "Will the user perform the correct action?" and "Why is the user not performing the correct tasks?".

### 2.6.12 Predictive Human Task Analysis

Predictive Human Task Analysis is a method for evaluating user errors occurring in interaction with the interface. This method analyzes the ways that users can make mistakes by predicting them, identifying what is causing them and what the consequences will be if the user makes them. How detailed this analysis is depends on the creativity of the designer.

## 2.7 Work Method

### 2.7.1 Agile

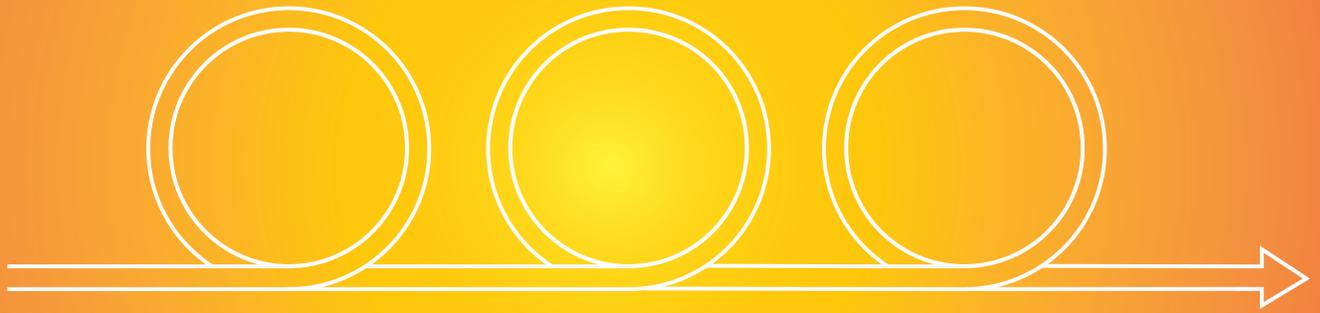
Agile is a methodology for the development of software. It was expressed in the Agile manifesto written in 2001 by a group of software developers [33].

The Agile manifesto called for a software development methodology that valued self-organized teams working with incremental development where the goals and requirements of the software can be changed during development. The Agile methodology values flexibility over having a static plan that dictates the software development. It also values continuous communication between the software development team and the stakeholders of the project over contract negotiation. The Agile method as expressed in the manifesto also states that working software should be the prime measurement of progress in the development progress.

### 2.7.2 Scrum and Sprint

Scrum is an Agile framework that can be used during Agile software development. The Scrum Alliance, an organization that was created to promote the use of Scrum, describes its framework in the following way [34]:

- "A product owner creates a prioritized wish list called a product backlog.
- During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces.
- The team has a certain amount of time — a sprint (usually two to four weeks) — to complete its work, but it meets each day to assess its progress (daily Scrum).
- Along the way, the Scrum Master keeps the team focused on its goal.
- At the end of the sprint, the work should be potentially shippable: ready to hand to a customer, put on a store shelf, or show to a stakeholder.
- The sprint ends with a sprint review and retrospective.
- As the next sprint begins, the team chooses another chunk of the product backlog and begins working again."



## 3. Method

This section contains a general overview of the various methods that were used in the project. How these methods were implemented, and how they were changed or deviated from, is described in section 4.

### 3.1 Work Process

The work was done in two weeks sprints. These sprints were followed by a meeting which evaluated the previous sprint and set up plans for the next sprint. A meeting with the company then followed. In this meeting the plans for the next sprint was explained and feedback was used to guide it in the right direction.

### 3.2 Software Skills

A big part of the project focused on integrating our work into Software Skills' large and complicated system. Therefore Software Skills developers were used as resources to guide us to a better understanding of the system and which tools to use in the project. The main developer who had been assigned to manage us was Luuk van Egeraat who had developed much of the system we worked on.

Our supervisor at Software Skills, Henrik Enström, was also a resource that we used to guide us in constructing the system so that it would suit the company's needs.

### 3.3 Coding

The coding consisted of several areas: front end, back end and security. The amount of knowledge required in each area meant that we had to delegate responsibilities for each field.

The project used a version control system with a repository server. A version control system manages changes done to data. The repository server stores this system so that people can collaborate on the same project. This was an invaluable asset as there were multiple developers in the project. Git was the version control system used, while Bitbucket was used as the repository hosting service.

### 3.4 Front End

The front end work mainly consisted of creating a functional page using HTML, SCSS, JavaScript and the AngularJS framework. These pages were continuously updated using the latest design suggestions and input from Software Skills.

### 3.5 Back end

The back end work consisted of creating server-side handlers that were able to receive requests from clients. The handlers are files that manage the connection between the clients and the database. They also had to model the database in a way such that all necessary information was stored.

#### 3.5.1 Handlers

The handlers were written in JavaScript with Node.js to handle connections concurrently. Each page had different requirements in terms of data, which meant that we created specific handlers for each page.

#### 3.5.2 Database

The initial phase of the database modeling consisted of finding all the necessary entity types for the database, all attributes needed for those entities, and the relationship structure between the entities. The next step was to normalize and remove any redundancy we could find in our database model. This was to prevent inconsistencies in the data associated with update queries, also known as data anomalies.

Finally we tried to optimize the performance by using different NoSQL data modeling techniques. Since there are no JOIN operations in NoSQL databases, common practice is to denormalize information to increase performance. We made sure we only denormalized data that would stay consistent to prevent the possibility of any data anomaly. This increased the total data volume but decreased the processing complexity. We also nested data because it gave us the possibility retrieve information from the database with a single retrieval thus minimizing the amount of queries needed.

## **3.6 Security**

### **3.6.1 Security Design Document**

To aid our group in designing a secure application we created a security design document. This document contains a list of pertinent secure design principles, and more detailed guidelines for critical sections of the application. These critical sections were determined by looking at the OWASP Top Ten and our project as specified at the outset. These are: input validation, authentication and session management. For each of these sections several guides and "cheat sheets" were examined, mainly from OWASP. The information was then reduced to only that which was deemed applicable to our project. Added to this were recommendations on how to implement the suggestion in the guidelines, and links to other resources. The document is appended as appendix A.

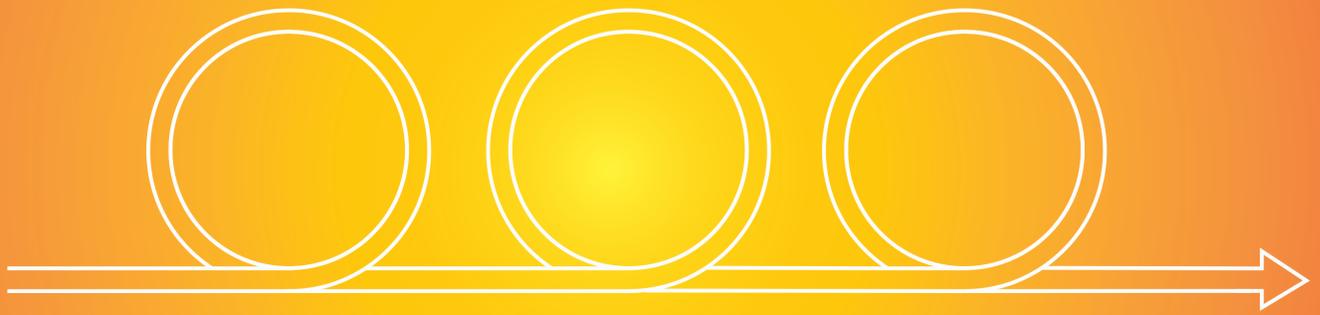
### **3.6.2 Evaluation**

To evaluate our work on hardening the application against attacks we planned to use the Application Security Verification Standard, as well as the expertise of Luuk van Egeraat.

## **3.7 Design**

We applied a number of usability theories and work methods to create the design for the layouts in the interface. Our main focus was to make it serviceable for our users as well as aesthetically pleasing and in accordance with the style of the existing graphical interface.

First we performed the necessary user research to understand our users; interviews and KJ-analysis. Step two was to create a general outline and evaluate these with different methods (see section 4.1). These later turned into final designs. The second and final step were repeated several times to optimize the usability and user experience.



## 4. Implementation

This section describes the implementation of the different methods described in the previous section. It also describes the deviations from these methods that were made.

### 4.1 Work Process

Work with the developers at Software Skills and their input was to be done continuously throughout the project, and recommendations from them were to take precedence over group decisions.

A part of our work process was to get feedback from the other candidate and master groups regarding the look and features of the new system. For the design this meant redrawing to make sure it fit the style of the other groups. For the other parts of the project this meant that links and elements were inserted to access or take advantage of the other groups features in the system.

#### 4.1.1 Scrum

At the start of the project Scrum was used to organize and evaluate the sprints. As the project continued, the Scrum methodology hampered progress. Hence, we abandoned Scrum as a methodology. By doing so, less focus was put on the evaluation of every sprint meeting and the work became more self organized. This put a greater responsibility on the group members and shifted focus to production.

### 4.2 Coding

The work on the HTML and SCSS files mainly supplied the layout and look of the pages, while sometimes the Angular Material framework would supply the layout and design for specific elements

of the page. The JavaScript and AngularJS code was responsible for the functionality of the pages and communication with the back end.

At the start of the project a learning phase was needed to ensure the group members could implement design later on. This learning phase was done through a trial and error method. Group members would try to implement the different ideas that was brought forth during the early sprint meetings, or during consultations with Software Skills developers.

### 4.3 Security

The developers were tasked with reading the security design document and implement the suggestions were appropriate. The general design principles and the guidelines regarding input validation ended up being the most relevant sections for the Software Market. The sections dealing with authentication and session management applied more to the already existing Software Skills website, which the Software Market is intended to become a part of. Upgrading the Software Skills website session management, and indirectly the Software Market, was also done.

### 4.4 Design Methods

When starting the design process we established a mutual understanding of goals and directives beforehand with our supervisor at Software Skills, their developers and the designers in the master thesis group. This was done to avoid misunderstandings and to get clear directives.

#### 4.4.1 Interviews

We conducted semi-structured interviews with eight people. Seven by phone and one in person. The majority of the interviewees where people who had a lot of experience with headhunting. We had prepared a manuscript beforehand but all interviewees were allowed to speak freely at times. All notes were written during the conversations. These notes were later corrected and transcribed to improve readability.

#### 4.4.2 KJ-analysis

Quotes from interviews were gathered and used in a KJ-analysis. Our goal was to understand our users and grasp what information they find essential. The KJ-analysis was used as a base to formulate a list of demands for each page.

#### 4.4.3 Wireframes

Using the list of demands as a a guide, we sketched wireframes for each different page. These wireframes were later transformed into more detailed mock ups where the majority of the important features from the list of demands were included. These wireframes and mock ups were delivered to the developers and used as a base to build the different interfaces for the pages.

#### 4.4.4 Task Flow Analysis

Along with the wireframes and mock ups we performed a task flow analysis for each page. In order to reach an optimal level of usability it was important to perform this analysis to ensure that users find the information they are looking for without any substantial problems.

#### 4.4.5 Cognitive Walkthrough

To analyze the pages, we did a cognitive walkthrough on two of the pages: timelog and profile. The list of demands and the usability tests were used as a base for this exercise. It helped prioritize important functions based on what the test persons had stated.

#### 4.4.6 Predictive Human Error Task Analysis

For all pages, along with the cognitive walkthrough, a predictive human error task analysis was performed. This was used to predict and identify as many errors as possible.

#### 4.4.7 Usability Tests

Usability tests were conducted with people that were not previously involved in this project. This was done to exclude any bias and to understand what was working with the current interface and where improvements were needed. The tests were performed in a relaxed environment: in an office or more crowded places at Chalmers University of Technology. The test persons were asked to "think out loud" while performing some tasks and to guess what the different elements in each design meant. All tests were performed by following a set of representative tasks for each test, and notes were taken on what the users said as the tasks were performed. After each test, the participants were asked to give general input. Furthermore, the participants were asked what tasks they found easy to perform, and to describe difficulties they might have faced using the interface.

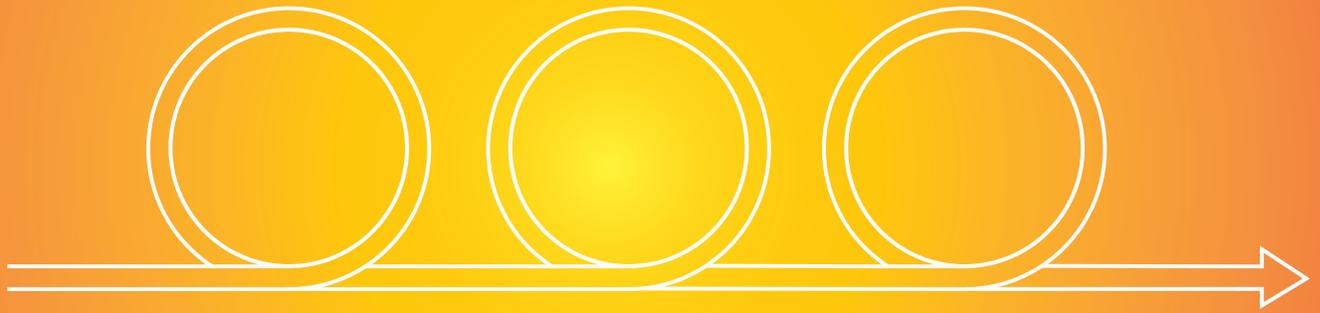
Using this information, we went back to the drawing board and made some improvements and corrections. Afterwards some more tests were performed. They were short, and we tried to confirm if there had been any progress made in terms of usability.

#### 4.4.8 Boards

Before creating the final design we created an expression board. This served as an inspiration for creating the graphical interface for the pages.

#### 4.4.9 Final Design

The last stage of the design process was to create the final design of each page. To reach a coherent design of the pages, the master thesis group was consulted. Their design, Software Skills current home page, and the expression board served as a style guide. Different pixel and vector based image editor programs were used to create the final look for the developers to implement in the front end. The final designs were also tested on different usability test subjects. However, these tests were less extensive than in the initial usability tests.



## 5. Results

This section aims to both describe the final result of our project and to answer the evaluation questions posed in section 1.5.

### 5.1 Did User Experience Design Benefit from Agile Practices?

From a UX design point of view, the Agile methodology proved very useful for our short and dynamic project. The ability to quickly evaluate designs and make changes on the fly rather than adhering to a restricted plan made the design process fluid.

A struggle regarding the design was to adhere to the design of Software Skills' website, which was in flux during development. Agile was the only methodology able to handle these changes in an effective and timely manner.

Some difficulties were faced with the Agile methodology, as it put greater stress upon the creative effort of the design process. These stress factors originated from the fact that the design constantly required changes, rather than being completed after the design was created. This made it more difficult to be sure that effort spent would be useful, rather than being scrapped.

As the gains of the Agile methodology still resulted in completion of the project with a design which fulfilled our goals, we reason that the Agile methodology at large benefited the user experience design.

### 5.2 Data Protection with Agile Practices

The work with data protection was likely the part of the project that benefited the least from the use of the Agile methodology. After we abandoned Scrum and a more organized approach, our focus was more easily shifted towards a tangible and production ready application. Security issues

gradually became a secondary consideration. This was experienced as a very natural process which would require awareness and planning to counteract. The creation of the security design document alleviated the problem, since it provided the developers with a tool that could be accessed and used independently.

From a security standpoint it therefore seems like a more Scrum-oriented approach throughout the project would have been beneficial. This would also have been in keeping with the Agile methodology, but a more formal and organized variant than what we ended up with. In conclusion we cannot readily state that data protection benefited from Agile practices. A more rigid methodology could very well have worked better in this case.

### 5.3 Project Work and Agile Practices

Through the project sprints, incremental progress with rapidly changed goals were present in the software development. Effort was put towards reaching a final product rather than adhering to a plan. Changes were frequent and the form of the product changed many times during the project. Even with the abandonment of Scrum (an Agile methodology), we still worked with methods that were based upon the Agile manifesto.

Because the project involved several other parties with diverging interests to consider, it was similar to multi-team software development. This, and the fact that the interests of Software Skills were present throughout the project, support the argument that our project is a realistic case study.

### 5.4 Software Market

The software project resulted in a system that could handle some of the features we had planned for. The new features lie mostly in three views: profile, contract and timelog. These features were made in regard to the applicant and company interaction with the system.

Much of the work was spent on integrating these different sections both with each other and with Software Skills' existing system. These new additions were based on the final design suggestion for our project, as well as some of the requirements of the security design document.

#### 5.4.1 Profile

The profile page was created as a way for applicants to present and display information about themselves. The applicants' profiles are visible to companies and therefore has relevant information regarding the applicants' qualifications.

The profile page is divided into a central space and a sidebar. The central space of the profile page consists of the applicants' main information. From the interviews we found that employers mostly focus on the candidate's personality, his or her previous experiences, references and education. Hence these features were ranked and placed in that order. The order was also emphasized by design conventions and a benchmarking analysis.

The top part of the main content displays the profile picture, name, title, location and the preferred hourly wage of the applicant. Tags, a type of keyword associated with the applicants' qualifications,

The screenshot shows a user profile for 'Maxime De Gaulle', a Software Engineer. The profile includes a navigation bar with links for 'About', 'Jobs', 'Tests', 'Messages', 'Contracts', and 'Timelog'. The user is logged in as 'Maxime De Gaulle'. The profile features a profile picture, a name, title, location (Sweden, Göteborg), and a rate per hour of \$18.00. Skills listed include Java, Node.js, Angular.js, JavaScript, Hadoop, Google Maps API, and HTML5. The profile is divided into sections: Summary, Experience, and a sidebar with a rating system and language proficiency. The Summary section contains two paragraphs of text. The Experience section lists two roles: Database Administrator at iZettle (2008-2011) and Front-end Developer at Google (1994-2006). The sidebar shows a 'Weighted score (100%)' of 5/5, with breakdowns for General impression (40%), Quality of work (20%), Trust (20%), and Communication (20%). It also includes buttons for 'Dashboard' and 'Settings', and a 'Languages' section showing proficiency in English, Swedish, Spanish, and Available.

Figure 5.1: A screenshot of an example profile.

are also displayed there. This way companies can easily search for specific tags to find qualified freelancers for their projects. Below we have text fields that contain the summary, work experience, education and the tests taken by the applicant.

The sidebar of the profile has complementing information regarding the applicant. This helps employers create a more versatile picture of who the candidate is. At the top of the sidebar is a rating system with an overall weighted score based on a general impression by the employer and by the quality of work, trustworthiness and the communication skills of the applicant. These scores are based on previous employers' ratings of the applicant. This score gives an employer a chance to rate the applicant after a finished job in a more balanced way. The General Impression is 40% of the grade, and the remaining are 20% each. All separate scores are visible on the page, making the grading system more transparent to everyone. If an employer would like to write a comment or a more extended opinion about the applicants' work, this is also possible. The scale is from one to five, where five is the highest rank.

Further down the page we have buttons for different actions. These actions will be explained more thoroughly in the next paragraph. Following the buttons we have additional general information such as languages spoken and total hours worked on the site. Both displays and their features can be seen in Figure 5.1.

The page has two states: you are either looking at your own, or another persons' profile. These two different states have different editing privileges and action buttons. The former state gives you editing privileges while looking at your own profile. We implemented it using hover functionality, meaning that the user can hover over different fields and text to get an edit option. There is also a button that redirects you to your settings where you can change more personal information such as your password. The latter state gives you no editing privileges and the buttons are different. Here you get the option to either message, bookmark or offer a job to the applicant.

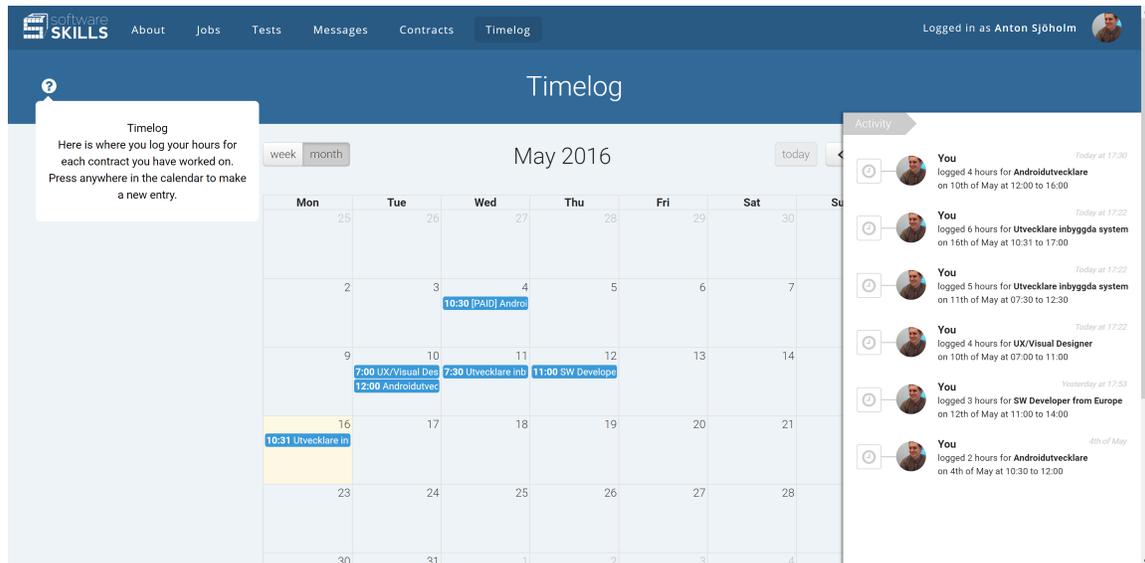


Figure 5.2: A screenshot of the timelog in the monthly view, also showing the question tooltip and activity bar.

## 5.4.2 Timelog

The timelog was created to make the applicant able to log time for their different contracts. The timelog is mainly made up by a calendar, but also has a question and activity button in the header of the page. The focus was on logging hours efficiently and maintaining high compatibility. We found that applicants need strong feedback concerning three things:

- What had been logged?
- What contract and company has the applicant worked for?
- How many hours have the client worked?

Employers are interested in seeing a short description of the applicants' work and how many hours they are billed for. Other information they find redundant.

The question button is activated by hovering over it. When this is done it provides instructions about using the timelog. The activity button opens up a sidebar that shows recent events that have occurred in the timelog, such as new entries or edits.

The calendar provides most of the of the functionality of the page and it is placed in the center of the page. It has two states: the weekly view and monthly view, which will show the timelog entries for the different time spans. The monthly view can be seen in Figure 5.2 and the weekly view in Figure 5.3.

At the top of the calendar view the option to switch state as well as go to the previous or next time span is available. There is also a button that makes the calendar go to the current date.

Creating an entry in the timelog is done by simply pressing a section of the timelog, which causes a

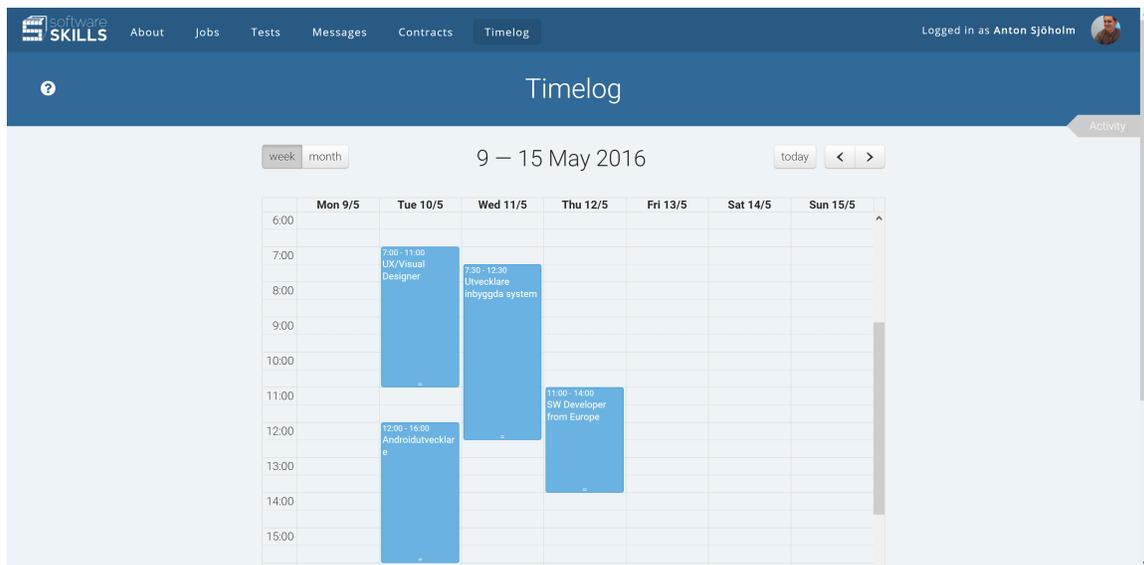


Figure 5.3: A screenshot of the timelog in the weekly view.

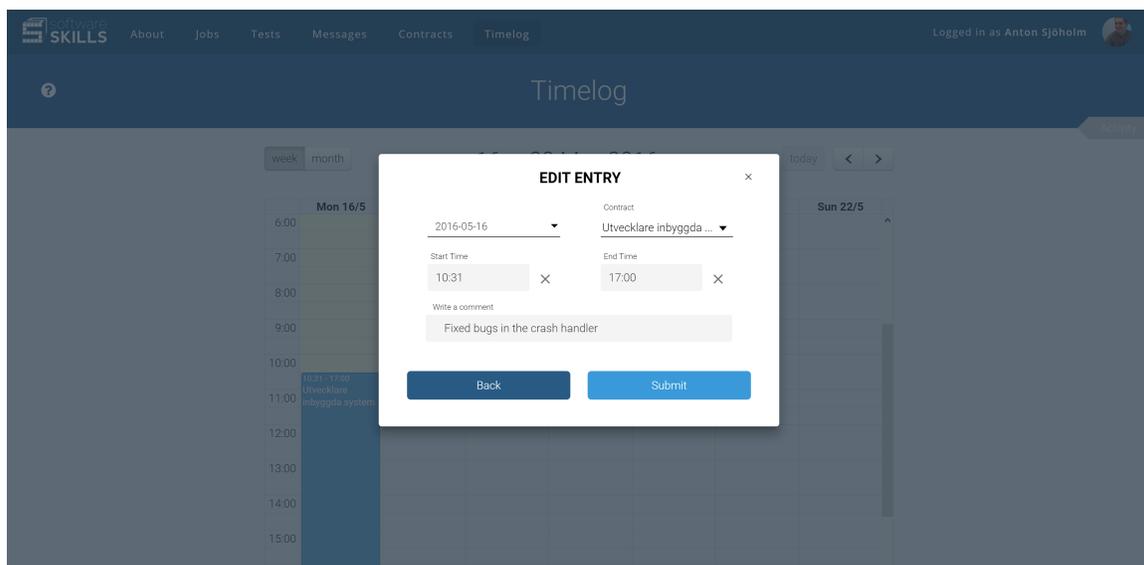


Figure 5.4: A screenshot of the timelog dialog when editing a timelog event.

dialog that enables the applicant to specify the details of the entry to be presented. In this dialog the applicant supplies the date, duration and contract of the timelog entry. It is also possible for the applicant to add a comment on the entry, which can be used by the applicant to describe what was done during the entry's duration. If the date, duration and contract for the entry have been supplied the dialog closes and an entry is stored in the system. From here the entry can be edited in several ways: either by pressing it and changing the information in the dialog, which can be seen in 5.4,

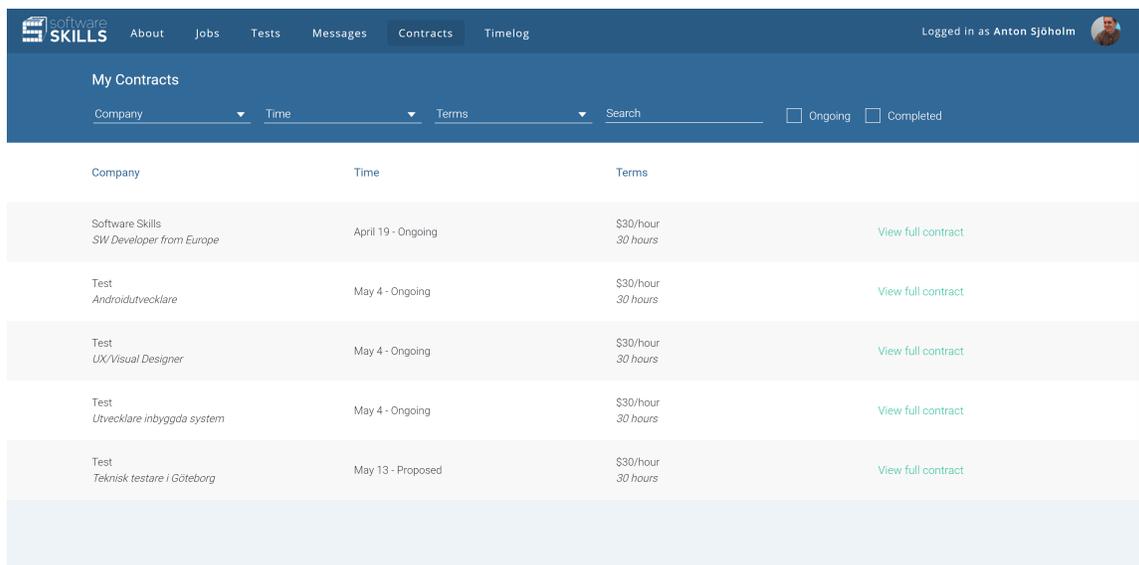
grabbing and pulling it in either start or end to change duration, simply dragging and dropping it on another time or date, or even deleting it. An entry that have been changed will at the moment of writing create a copy of the previous version which will be seen as edited. When an entry has received a payment id in the database, it can no longer to be edited or deleted. However, an applicant can edit all entries until they are paid.

### 5.4.3 Contract

The contract feature is present in the pages responsible for showing all the applicants' contracts, and the details of a specific contract. Focus is to maintain visual clarity and consistency within the interface. The first view, shown in Figure 5.5, shows all the contracts that are connected to the applicant. The search option enables the candidate to search and sort their contracts. To maintain consistency, the page was designed so it has similar attributes to other pages in the Software Skills interface that display similar types of methods for organizing information.

We listed the relevant information that the applicant requires to get a clear overview of his or her contracts. The search option filters the contracts based on the following:

- The company the contract belongs to.
- The month the company was started in.
- The rate of the contract.
- The description of the contract.
- The status of the contract.



Company	Time	Terms	
Software Skills <i>SW Developer from Europe</i>	April 19 - Ongoing	\$30/hour 30 hours	<a href="#">View full contract</a>
Test <i>Androidutvecklare</i>	May 4 - Ongoing	\$30/hour 30 hours	<a href="#">View full contract</a>
Test <i>UX/Visual Designer</i>	May 4 - Ongoing	\$30/hour 30 hours	<a href="#">View full contract</a>
Test <i>Utvecklare inbyggda system</i>	May 4 - Ongoing	\$30/hour 30 hours	<a href="#">View full contract</a>
Test <i>Teknisk testare i Göteborg</i>	May 13 - Proposed	\$30/hour 30 hours	<a href="#">View full contract</a>

Figure 5.5: A screenshot of available contracts in the contracts view.

The page that shows a specific contract is split into several parts that either show information about the contract or lets the applicant interact with the contract.

The topmost section of the contract displays the contract title and provides navigation back to the "All Contracts" page. It also contains a button that makes a sidebar slide out. In this sidebar the recent activity related to the contract is displayed. The top section can be viewed in Figure 5.6.

Following that the top section of the contract is split into four sections responsible for showing most of the information about the contract and the time spent on it. These sections are:

- A chart
- Time and money overview
- Company overview
- Contract overview

The chart is linked to the database and displays the time spent during the last twelve weeks on the contract in a line chart. There are a total of twelve points in the line chart that each represent the amount of hours from paid timelog entries that week.

Under the chart is an overview of the time spent and the money earned by the applicant on the specific contract. In this overview the applicant can see the time spent this week on the contract as well as the total time spent on the contract. The applicant can also view money earned this week and the total money earned so far. The information in the display is gathered from the paid timelog entries. Timelog entries that have not been paid will not count towards the hours spent in either the week or in total.

The company overview shows the logotype of the company that is part of the contract as well as information related to the company's contact person for the contract. It also has a link to the messaging page between the applicant and the company's contact person. The messaging page was created by another bachelor thesis group. The contracts overview displays the contract's id, time frame, rate and work limit. Its time frame is shown by displaying the date where the contract started and the deadline set on the contract. The work limit of a contract is either an amount of hours per week or a fixed amount of hours depending on what is agreed upon in the contract.

The bottom section of the contract is split into three parts that are placed on top of each other. These parts are the task display, the terms display and the buttons that changes the status of the contract.

The task display consists of several cards that are meant to represent the different assignments that should be completed to fulfill the contract. These cards are sorted into three lists that display the state of progress that a task can have. The task themselves have a short description of what is supposed to be done to complete them, as well as buttons that allow the applicant to move them between lists. The task display can be viewed in Figure 5.7.

The "Terms and Conditions" display, seen in Figure 5.8, is a simple text display that shows the terms of the contract as agreed beforehand by the applicant and the company. The terms themselves are stored in a HTML variable which allows flexibility in choosing what to display. The terms also contain a link that is meant to redirect the user to the Software Skills' terms and conditions. However, at the moment of writing this feature is not yet incorporated.

The buttons at the bottom of the contract allows the applicant to change the status of the contract. What buttons that are available to the applicant depends on the status of the contract. For a contract that has been proposed to the applicant, the options of accepting or rejecting will be possible. When

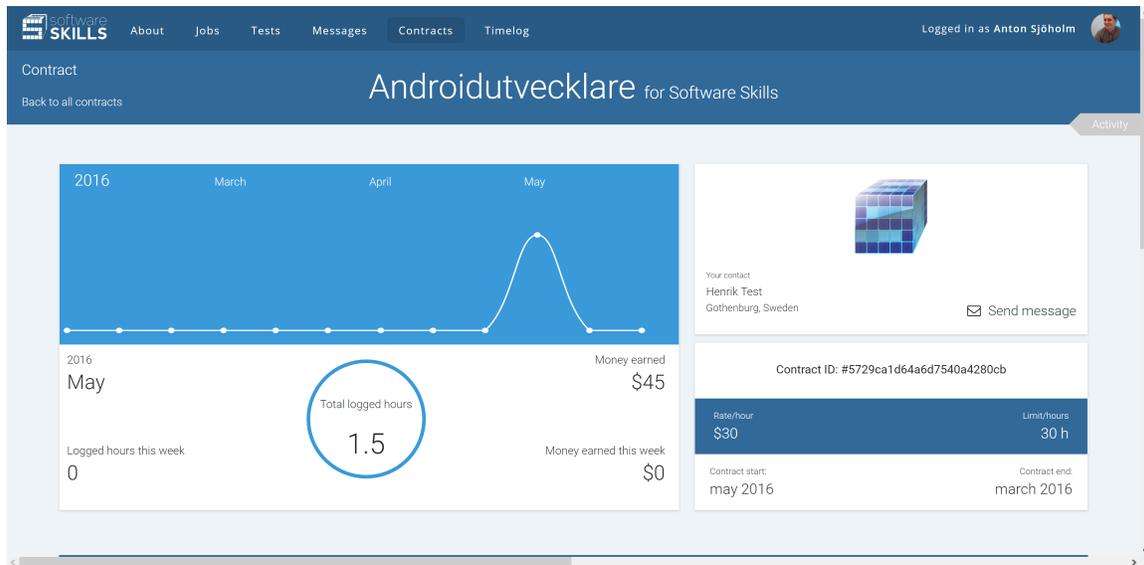


Figure 5.6: A screenshot of the top section of a specific contract view.

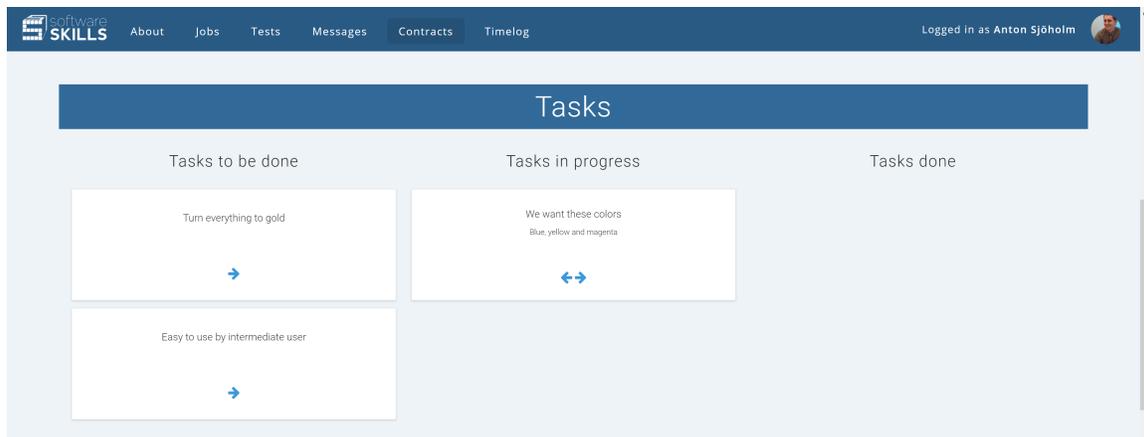


Figure 5.7: A screenshot of the task display of a specific contract.

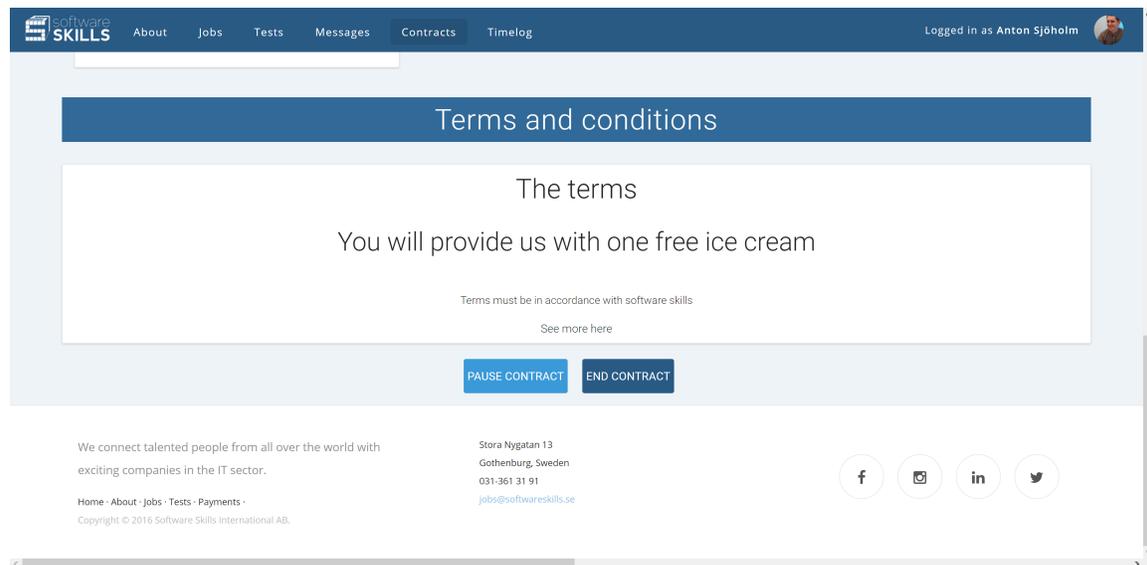


Figure 5.8: A screenshot of the terms section of a specific contract.

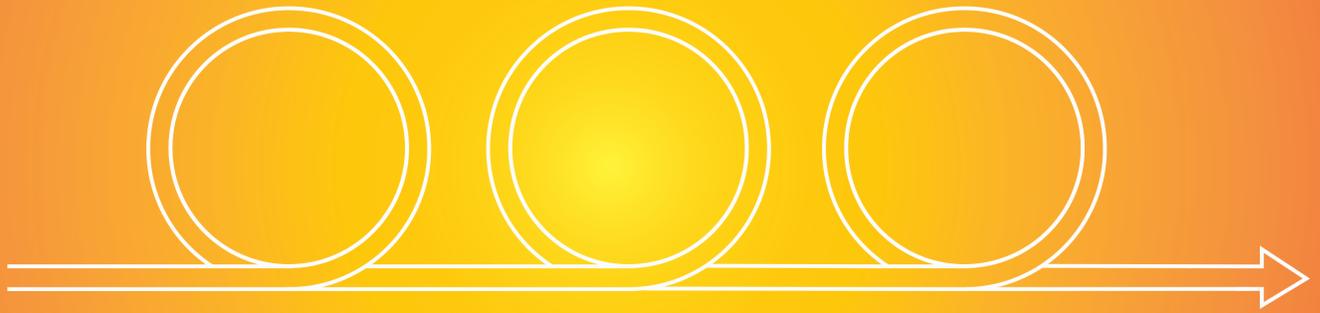
a contract has been accepted, the option changes to "Pause" or "End". When paused the option to resume will replace the pause option. The pause and end contract buttons bring forth a dialog when pressed. This provides feedback to the applicant, and makes sure it was the users intention to perform the action.

#### 5.4.4 Integration

The integration between the pages, database and Software Skills' system was done by reusing and expanding their old code base. Their website was built upon views, which essentially means that when a client is navigating the website, the user is changing between different templates. The ability to navigate the website and get specific views was done by assigning a state to each page and associate a HTML, SCSS, and JavaScript file with each state. This is called routing.

For communication between the client and the server they had handlers that read, updated and sent requested information. All security measures they had taken such as checking user privileges were also done in the handlers. This due to the fact that those files are on the server, and cannot be edited by the client. For each route they had one handler that provided the information necessary for the initial state of the page. Many pages had additional handlers that provided the ability to update information as well.

To keep the website consistent in terms of code, we had to adhere to their code architecture. This was done by associating an HTML template, a JavaScript file, a handler and a SCSS file to each of our pages.



## 6. Discussion

### 6.1 Legal framework and further studies

With the given time frame for the project analysis, the legal framework of the software was not analyzed. This area is suitable for further research and gives a unique perspective on how software can be affected by society.

In particular the different restrictions on the platform from Swedish and international labor laws would be an interesting area of study. Further on it could also be interesting to study if software platforms, such as the one developed in this project, will cause the need for a new international framework. This framework needs to detail which labor laws should apply for work performed in global online services.

### 6.2 Front End

The work in the front end was done in a similar manner to that in section 3.4. HTML and SCSS were used for design and layout, while AngularJS and Angular Material were used mostly for functionality and usability.

There is an interesting difference in the amount of Angular Material used in the pages. This entirely depended on which group member developed the page. Angular Material provides a lot of services in terms of creating standard elements on the pages. However, these elements can also be hard to modify. Different approaches for creating elements were present in the group. Some preferred to use Angular Material to a minimum and some used it whenever possible to save time. Situations when it was necessary to use it appeared in the project, but would often require modifications to make them resemble the final designs.

The front end still requires slightly more work to reach a level that Software Skills can use for a working platform. This was noted by the Software Skills' developer Luuk van Egeraat.

## 6.3 Back End

The back end work continued in a similar way to what we imagined when the project was planned. There were additions in handlers and the communication with the back end for the contract and timelog pages. The main challenge laid in understanding the existing system and its specific quirks. This was a somewhat difficult task for the group, but the developers at Software Skills were a big help in teaching us how the system worked.

## 6.4 Security

Since the project ended up very different from what was expected from the beginning, the work on security also ended up very different. The application was reduced to the addition of three main views which could not easily be evaluated using the ASVS, since it was designed for a more complete application. Furthermore, some parts of what was recommended in the Security Design Document were not applicable to the parts of the application we ended up developing. We mainly used the secure design principles and the guidelines regarding input validation in our project. However, Software Skills wanted access to this document in their continued development of the application. They are also working, in collaboration with our group, on implementing some of our suggestions related to session management in their existing candidate testing and recruitment system.

## 6.5 Design Method and Implementation

### 6.5.1 Interviews

The interviews served as the main base for the design/user study. An improvement in how they were performed could be the amount of people conducting the interviews. One person was responsible for directing and transcribing the interviews and due to the difficulties of multitasking, it would have been an advantage to have one person writing and one person focusing on asking questions. Another solution could have been to record all interviews and later on transcribe them to make sure no information was lost. There is a possibility that information/comments that were deemed less important at the time of the interview, could have been valuable later in the process.

The interview questions were revised after a couple of interviews because some of the questions were not applicable. This means that not all interviewed subjects were asked the exact same questions. On the other hand, since the interviews were semi-structured the interviewees were also asked to express further thoughts when giving their answers. This can be argued to compensate for some shortcomings of the wording of the questions.

Our candidates for interviews are experienced headhunters, and provided much information about the interviewing process of recruiting. A negative aspect was that they were mostly experienced as recruiters for long term employment rather than hiring staff for shorter periods of time. This could have some effect on the outcome of the material in terms of the social aspect. For the recruiters we interviewed it appeared that personal chemistry and well developed social skills are desirable qualities in a candidate. These qualities were not considered a priority for applicants in this project. However, these qualities were included in the design process as an important aspect when developing the page. The profile page in particular.

### 6.5.2 Benchmarking

One aspect of benchmarking is that it provides the designer with an idea of what the users might use as references. This aids usability in terms of Patrick Jordan's third principle: Compatibility. By applying similar ways of solving problems on our pages, the user can quickly grasp how problems can be solved. However, benchmarking beforehand can limit the designer by establishing a mental model on how specific problems should be solved beforehand. This might restrict the creative process. If the designer is influenced by existing graphical interfaces, this could mean that they are influenced by the study and thus restricted from creating groundbreaking designs.

Another aspect was that we only benchmarked English speaking websites. This was due to limited knowledge in other languages apart from Swedish and English. There is a possibility that we might have missed cultural aspects on what other conventions look like, and if the benchmarked websites conflict with other international practices. Nevertheless, the master thesis group decided to only benchmark similar websites in English. Ergo, it was likely not very harmful to this study not to have benchmarked a wider selection of pages.

### 6.5.3 Expression Board

The expression board captured the feeling of the designs we wanted to establish which contributed nicely to the graphical style of the current web page. We followed style guidelines from the current homepage, softwareskills.com, since it was necessary to maintain consistency with the current style of the pages. The expression board served as a guidance for the general user experience. However, it was probably considered somewhat inane to participants in this project not involved in the more creative phase.

### 6.5.4 Task Flow Analysis

The task flow analysis helped us predict the user behavior and target the graphical interfaces' weak spots. Nevertheless, because there was only one designer within the team there would have been a greater input if there had been more people involved in the analysis. This could have helped minimize bias, and give a more nuanced, versatile perspective on how users solve tasks.

### 6.5.5 Usability Tests

The main advantage of performing usability tests are that they provide input from users. Both in what the participants express verbally and by their actions in interaction with the graphical interface. This can shed light on difficulties users face in the interaction that designers and front end developers assume are no problems. To further improve our analysis we could have used a video camera or audio recorder to observe behaviors the users were unaware of. Our results could also have improved if these tests had been performed in a usability lab with multiple cameras and without external disturbing factors influencing. It would help us see patterns, look at the material again from different angles. External disturbance could be users feeling observed by friends and acquaintances, noise or being interrupted by other people.

### 6.5.6 Cognitive Walkthrough

Due to time restrictions there was only time to do a cognitive walkthrough on the timelog and profile. It was valuable to get an overview, but not an extensive contributor to the project. Although, it could depend on our own evaluation methods. We believed that the usability tests provided the majority of the material required for this study.

### 6.5.7 Predictive Human Task Error Analysis

The PHEA was, like the cognitive walkthrough, not very extensive. This was a consequence of them being carried out in final parts of the project. Therefore, this analysis was not prioritized but the theory was used when creating the designs. The PHEA theory was created to help designers include interactions in the interface that would prevent users from committing errors due to bad design. We hope to have avoided most damaging mistakes, due to working with consistency and error recovery in the design.

## 6.6 Final Designs

### 6.6.1 Profile

The score was a debated subject when creating the profile page, and was one of the features that were helped by the Agile process. It took many trials and improvements before finally finding a system that worked. Many web pages that provide some kind of service between humans contain a feature where people can rate each other. Here all users can give each other grades on a single scale between two extremes, depending on how satisfied they are with the exchange. Often these grades are invisible to both parties until both have given each other a review. From our interviews, many of the interviewees were not positive to the idea of a grading system. One interviewee said it was dehumanizing and could have fatal consequences if one applicant received a bad grade, possibly due to unfair circumstances. Perhaps this individual will face huge difficulties finding a job again through the platform. This made us question whether or not a grade system was an acceptable and moral thing to do. Another problem with grading each other is deciding grades beforehand. This is a common problem today on other sites. Instead of giving a fair and just grade, the parties agree beforehand to give each other the highest grade. Nevertheless, grades are also reassuring for any employer that the applicant is a trustworthy individual, and vice versa. No one wishes to hire someone that does a terrible job or is very difficult to work with. Therefore we decided to find a compromise and devoted a lot of time and investigation to find the right solution.

The reason why an uneven scale was chosen was that it forces an individual to make a decision if they are positive or negative to the applicant or company. There was also a debate on whether a profile picture was necessary. This creates a conflict between personal bias and compatibility. According to the benchmarking analysis, all profile pages had place for the applicant to upload a photograph of him or herself. The negative aspects of including a photo of the applicant is that bias might affect the employers opinion of the applicant based on looks. Photographs can contribute to creating a preconception of how an applicant will perform their work, and lead to discrimination. This is something that we have no desire to contribute to. However we decided to follow compatibility and use a photo, having these other aspects in mind. By not including a photograph, we believe

that the positive, reassuring experience of seeing what a person looks like weigh up the negative aspects.

### 6.6.2 Timelog

To improve user control, error prevention and recovery, and feedback, we use pop-up dialogs. If the user wants to edit a logged event, there is an opportunity to do so. However, since this is a billing base we wish to eliminate any chances of fraud. If any changes are made, feedback is provided to both user and company by publishing changes in the activity bar and logging them in the database. In further development of the actual billing page, these changes will also be visible in the invoice sent to the company. Making any changes transparent will make the applicant careful of how he or she logs their time, but also provides them with the possibility to correct errors in a way transparent for the company. User errors are easy to commit, and for a good user experience they should be easily corrected.

A side effect of using a JavaScript calendar, and a calendar in general, is that the calendar is not notably groundbreaking in its ways of logging hours. Meaning that there will most likely be many associations to e.g. Google by reasons of how users interact with the timelog. This however makes room for improvement in the future.

### 6.6.3 Contract and Contract Details

The contract details page was the hardest page to create in terms of the graphical interface. Contract details view deviates a lot from a regular job contract. Due to legal aspects, the contract page cannot be considered an official contract. The contract page should display a great variety of information but none that could be as easily categorized in to blocks like in profile. The ability to benchmark was limited since most contracts demanded actually applying for, and accepting, a job on similar pages. This page resulted in different sections that prioritizes and displays the most relevant information to the applicant first. This page has been redesigned many times during the process, meaning that this is possibly the best example of the Agile working process. The most relevant information is there, but the graphical interface requires refinement.

## 6.7 Agile and Collaboration

We started this project trying to incorporate both the Agile and the Scrum methodology in our work. This was partly successful. We started out adhering closely to the Scrum principles, while toward the end of our project we abandoned Scrum as our main methodology.

This change was done as we felt that the ever changing requirements from the other groups and Software skills made it difficult to effectively work with Scrum. The backlog would have become unmanageable by changes back and forth, and effort would have been spent organizing the project rather than working effectively with the code. The primary aspects of Agile that continued were the sprint duration and meetings, and the iterative process we worked through. In place of Scrum, group members assigned themselves work areas in which they sought to improve the platform and the report.

This self organization made it easier to produce the required views and progress in our work. The drawback was that group members became less aware of the different parts of the project they did not work with. It became harder to discern a clear and overarching picture of the platform and its functionality for the individual group members. Even with these drawbacks we felt that this change was necessary if we wanted to achieve our goals in the project. We also felt that these changes still adhered to Agile theories valuing production over time spent on devising a systematic approach.

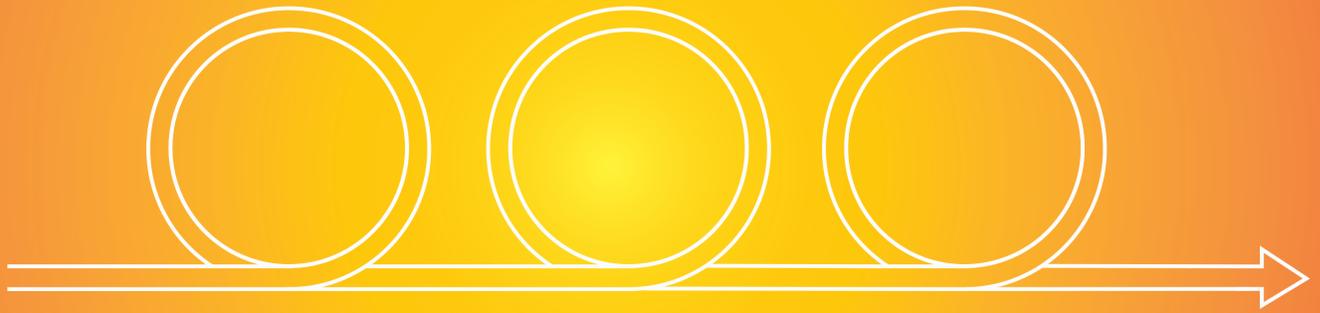
## 6.8 Scrum

Scrum is an easy way to organize a project and provide guidance to project members on how to structure their work. In the later parts for our project we felt that Scrum was an obstruction and we abandoned it for a more self organizing work method. This raises the question: was the Scrum practices a worthwhile investment for this project?

Scrum itself proved beneficial in giving an overarching picture of the platform. Its loss could be said to leave the security aspects of the project in a more worrisome situation. Scrum proved itself a hindrance in the later part of the project. When the ever changing needs for the project changed the backlog, it made Scrum hard to manage. The time and effort that would have been put into the method would have affected the final product and jeopardized the outcome of the project. It can even be argued that the project overall got a slow start by focusing time and effort into the practices instead of focusing on the product itself.

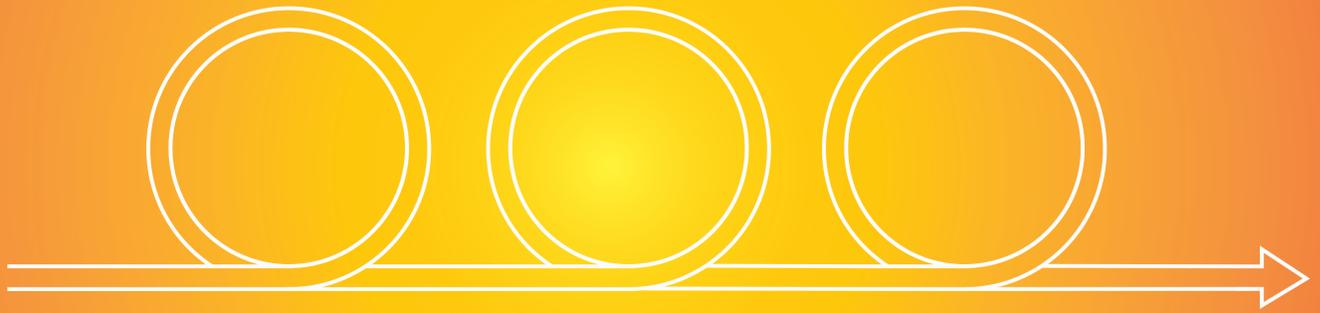
It is hard to evaluate how much of the benefits an overarching picture and somewhat clear guidelines benefited us in the beginning of the project. This is due to the learning phase of the project, which was a big part of the first half of the project. The learning was heavily self organized and incorporating its steps in Scrum became a hassle. On the other side, if more effort would have been spent incorporating the company, its developers and the other groups in the Scrum process, the result could have changed. If the backlog had included the other groups' plans, the changes that we felt were ever changing could have been more predictable.

There are ways in which a working Scrum process could have been accomplished. One way would have been to have more meetings where the backlog itself could have been discussed. A collective Scrum project backlog or spending more time visualizing the final product could have worked out as well. The counter argument to this solution would be that this would require even more time and effort to be spent on planning rather than work on the product. It would also delay the other groups from creating a plan that would best suit their project.



## 7. Conclusion

Agile worked well as a development method. Working with a lot of input from the different candidate and master thesis groups with differing goals necessitated a method able to handle rapidly changing requirements. The design goal of consistency with the other groups was greatly supported by a fluid design process. Even though the methodology could be said to increase stress on the design, since it could be altered at a moment's notice, we believe the design goals were accomplished in a satisfying manner through the use of Agile. The development rate of the application itself also increased by using the Agile methodology. By using this method we most likely spent more time developing the application rather than spending excessive time on planning. Even Scrum, which is an Agile methodology, proved too cumbersome for our rapidly changing requirements and our result driven focus. The largest drawback was a loss of focus on the security aspect. This felt like a natural process that would require a more rigid methodology to counteract. The problem was made less severe by the security design document which was designed to guide the developers in individual implementation of secure practices. In the end Agile proved not only worthwhile but necessary for the development of the application.

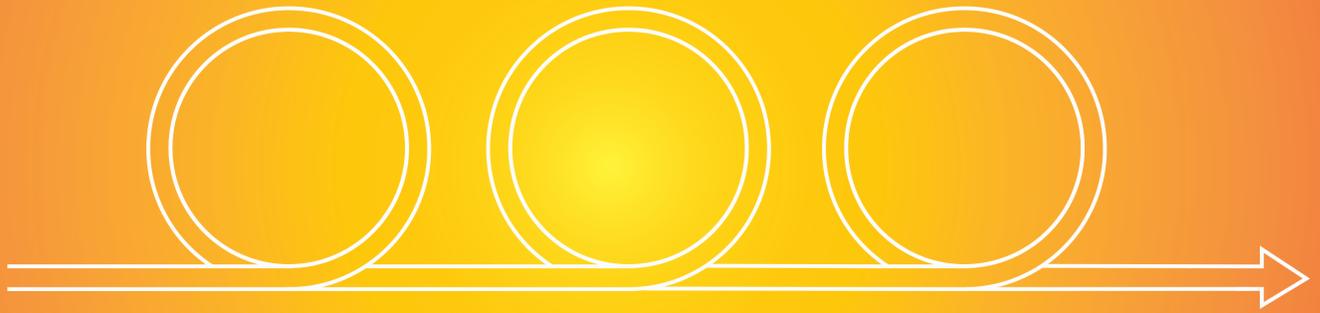


## Bibliography

- [1] R. Schumacher. (2016). Why you want to be good at data modeling, [Online]. Available: <https://dev.mysql.com/tech-resources/articles/why-data-modeling.html> (visited on 05/16/2016) (cited on page 11).
- [2] MongoDB. (2016). Nosql databases explained, [Online]. Available: <https://www.mongodb.com/nosql-explained> (visited on 05/10/2016) (cited on page 11).
- [3] I. Katsov. (2012). Nosql data modeling techniques, [Online]. Available: <https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/> (visited on 05/16/2016) (cited on page 12).
- [4] Google. (2016). Why angularjs?, [Online]. Available: <https://angularjs.org> (visited on 05/10/2016) (cited on page 12).
- [5] —, (2016). What is angular material?, [Online]. Available: <https://material.angularjs.org/latest/> (visited on 05/10/2016) (cited on page 12).
- [6] Mozilla. (2016). Html, [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML> (visited on 05/10/2016) (cited on page 12).
- [7] T. C. I. 159, *Usability*, ISO, 1998. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-1:v1:en> (visited on 03/22/2016) (cited on pages 12, 15).
- [8] N. Foundation. (2016). About node js, [Online]. Available: <https://nodejs.org/en/about/> (visited on 03/28/2016) (cited on page 13).
- [9] Mozilla. (2016). Websockets, [Online]. Available: [https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API) (visited on 05/10/2016) (cited on page 13).
- [10] W. Stallings and L. Brown, *Computer Security Principles and Practice*, 2nd edition. Pearson, 2012, pages 34–35 (cited on page 13).
- [11] Symantec. (2016). Internet security threat report 2016, [Online]. Available: <https://www.symantec.com/security-center/threat-report> (visited on 05/08/2016) (cited on page 13).

- [12] —, (2014). Internet security threat report 2014, [Online]. Available: [http://www.symantec.com/content/en/us/enterprise/other\\_resources/b-istr\\_main\\_report\\_v19\\_21291018.en-us.pdf](http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf) (visited on 05/08/2016) (cited on page 13).
- [13] O. Foundation. (2016). About the open web application security project, [Online]. Available: [https://www.owasp.org/index.php/About\\_OWASP](https://www.owasp.org/index.php/About_OWASP) (visited on 05/06/2016) (cited on page 13).
- [14] —, (2015). Owasp top 10 project, [Online]. Available: [https://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project) (visited on 04/22/2016) (cited on page 14).
- [15] NSA. (2013). Hardening deployed web applications, [Online]. Available: [https://www.nsa.gov/ia/\\_files/factsheets/Hardening\\_Deployed\\_WebApplications11182013.pdf](https://www.nsa.gov/ia/_files/factsheets/Hardening_Deployed_WebApplications11182013.pdf) (visited on 04/22/2016) (cited on page 14).
- [16] G. Holliday. (2008). Team foundation server (tfs) and the open web application security project (owasp) top ten, [Online]. Available: <https://msdn.microsoft.com/en-us/library/dd129898.aspx> (visited on 04/22/2016) (cited on page 14).
- [17] J. Manico *et al.* (Oct. 2015). Application security verification standard 3.0, [Online]. Available: <https://www.owasp.org/images/6/67/OWASPApplcationSecurityVerificationStandard3.0.pdf> (visited on 03/25/2016) (cited on page 14).
- [18] I. 1. Technical Committee, *User experience*, ISO, 1998. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en> (visited on 03/23/2016) (cited on page 14).
- [19] O. Rexfelt, *Introduktion till usability*, University Lecture, 2015 (cited on page 14).
- [20] —, *Att utforma gränssnitt*, University Lecture, 2015 (cited on page 14).
- [21] J. Nielsen, *Usability 101: Introduction to usability*, 2012. [Online]. Available: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/> (visited on 03/20/2016) (cited on page 15).
- [22] “Online etymology dictionary”, May 2016. [Online]. Available: <http://www.dictionary.com/browse/benchmarking> (visited on 05/04/2016) (cited on page 15).
- [23] P. Wallgren, *Frågebaserade datainsamlingsmetoder*, University Lecture, 2014 (cited on page 15).
- [24] —, *Introduktion behov o krav 2014*, University Lecture, 2014 (cited on page 15).
- [25] —, *Analys*, University Lecture, 2014 (cited on page 16).
- [26] Anonymous. (2016). Vad är en kravspecifikation?, [Online]. Available: <http://www.kravspecifikation.se/> (visited on 03/25/2016) (cited on page 16).
- [27] L. Wikström, *Produktsemiotik - expression board*, University Lecture, 2014 (cited on page 16).
- [28] Anonymous, May 2016. [Online]. Available: <http://www.dictionary.com/browse/wireframe> (visited on 05/05/2016) (cited on page 16).
- [29] J. Patton, “Hitting the target: Adding interaction design to agile software development”, volume OOPSLA 2002 Practitioners Reports, number 1, 1–ff, Nov. 2002 (cited on page 16).
- [30] A. F. Rose *et al.*, “Using qualitative studies to improve the usability of an emr”, volume 38, number 1, pages 51–60, Feb. 2005 (cited on page 16).

- [31] L.-O. Bligård, *Teoretiska metoder för utvärdering av användarvänlighet*, University Lecture, 2015 (cited on pages 16, 17).
- [32] O. Rexfelt, *Empiriska metoder: Användbarhetstest*, University Lecture, 2015 (cited on page 16).
- [33] M. Beedle *et al.* (2001). The agile manifesto, [Online]. Available: <http://agilemanifesto.org/iso/en/> (visited on 03/28/2016) (cited on page 17).
- [34] Scrumalliance.org. (2016). Learn about scrum, [Online]. Available: <https://www.scrumalliance.org/why-scrum> (visited on 03/28/2016) (cited on page 18).
- [35] C. Punga. (Oct. 2015). Principles of security engineering, [Online]. Available: <https://github.com/OWASP/DevGuide/blob/master/02-Design/01-Principles%20of%20Security%20Engineering.md> (visited on 02/27/2016) (cited on page 44).
- [36] V. Maniar *et al.* (Dec. 2015). Input validation, [Online]. Available: <https://github.com/OWASP/DevGuide/blob/master/03-Build/0x06-InputValidation.md> (visited on 02/27/2016) (cited on page 46).
- [37] D. Wichers. (Feb. 2016). Input validation cheat sheet, [Online]. Available: [https://www.owasp.org/index.php/Input\\_Validation\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet) (visited on 02/29/2016) (cited on pages 46, 47).
- [38] J. Williams *et al.* (Mar. 2016). Xss (cross site scripting) prevention cheat sheet, [Online]. Available: [https://www.owasp.org/index.php/XSS\\_\(Cross\\_Site\\_Scripting\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet) (visited on 03/02/2016) (cited on page 47).
- [39] E. Keary *et al.* (Mar. 2016). Authentication cheat sheet, [Online]. Available: [https://www.owasp.org/index.php/Authentication\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Authentication_Cheat_Sheet) (visited on 03/09/2016) (cited on pages 48, 49).
- [40] K. Kenan *et al.* (Jan. 2016). Session management cheat sheet, [Online]. Available: [https://www.owasp.org/index.php/Cryptographic\\_Storage\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet) (visited on 03/07/2016) (cited on page 49).
- [41] B. Chesney *et al.* (Dec. 2015). Authentication, [Online]. Available: <https://github.com/OWASP/DevGuide/blob/master/03-Build/0x03-Authentication.md> (visited on 03/07/2016) (cited on page 49).
- [42] R. Siles. (Jan. 2016). Cryptographic storage cheat sheet, [Online]. Available: [https://www.owasp.org/index.php/Session\\_Management\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Session_Management_Cheat_Sheet) (visited on 03/11/2016) (cited on page 50).



## A. Security Design Document

### **Purpose**

In order to build a secure web application security must be considered at all levels of development, from design to final testing. This document aims to give the necessary security awareness to the developers and supply them with tools and guidelines to both simplify and enhance the creation of a secure application.

### **Structure**

The security design document will consist of two main parts. The first of these, Secure Design, will describe a set of principles for secure design that should be applicable to the security design of the Software Market application. The second part, titled Guidelines for Critical Sections, will be a list of guidelines for secure design in specific crucial parts of the application.

### **Secure design**

The following set of design principles should, where applicable, be adhered to [35].

#### **Deny by default**

When access control fails it should fail securely. That is to say, the default state should be to deny access.

### **Layered defence**

This principle, also known as Defense in Depth, suggests eliminating single points of compromise by creating a series of security safeguards. If the nature of the safeguards is diverse increased security is achieved.

### **Least privilege**

The principle of least privilege states that users or processes should be given the very minimum set of privileges needed to achieve an assigned task. This includes restricting access time to the time necessary to complete the task. For this to work a proper level of privilege granularity is needed.

### **Economy of Mechanism**

This principle is a derivation of the more general principle of Keep it Simple, Stupid. Simply put, the simpler and more easily understood a piece of software is, the less attack surface it has and the easier it is to protect.

### **Complete Mediation**

Here we seek to prevent security breaches that abuse access given once to a certain object to access that object again, perhaps without the proper privilege this second time. Access privilege should, in other words, be checked every time the object is accessed, and never depend on previous access being granted. This can affect performance as accesses often are cached, and these caches need to be flushed or invalidated in order to adhere to this principle.

### **Open Design**

A security design that depends on keeping the particulars of its implementation secret is a vulnerable design. Attackers can often quite easily get access to information about e.g. the hashing function used for passwords or the settings of a firewall. The Open Design principle concludes that the overall security design should be robust enough to prevent intrusions even if details about its implementation is revealed to an attacker.

### **Least Common Mechanism**

This principle states that the same mechanisms should not be shared by users or processes that operate at different levels of privilege, i.e. the same function should not be used to retrieve data about a user's contact information and their billing information.

### **Psychological Acceptability**

Often implementations of security principles can be cumbersome for the user, who might then decide to circumvent them. The principle of psychological acceptability takes this into account and aims

to maximize adoption of security functionality by ensuring it is easy to use and transparent to the user. In the end, security mechanisms should not make a resource more difficult to access for the user.

### **Weakest Link**

The ease with which a system can be penetrated usually depends on the most vulnerable part of it, e.g. interfaces, code base, services etc.

### **Leveraging Existing Components**

This principle seeks to enable adherence to the Economy of Mechanism principle by promoting reuse of existing software components and functionality in order to avoid increasing the attack surface of the software.

### **Guidelines for Critical Sections**

The following guidelines for critical sections of the application should be followed to ensure a secure end product.

### **Input Validation**

Many web applications are insecure due to naive faith in user input. In the case of the Software Market application the following vulnerabilities are a minimal list of threats [36]:

- HTML injection
- NoSQL injection
- Script injection
- Social engineering

This is an application wide strategy for input validation and exceptions must be argued for. The following considerations will be taken for all user input:

- We will assume all input fields are attack vectors and all users are potential attackers.
- Client side code can be viewed and modified, therefore input validation in client side code will never be trusted. Server side validation is required.
- HTML headers are user input. Thus we will never trust HTML headers or base security decisions on their content as they are easily manipulated.
- Where applicable we will validate all user input for type, length, format and range.

### **White List Validation and Regular Expressions**

The most common way to validate input is to use a “black list” containing forbidden characters or sequences of characters that are not allowed [37]. This is a flawed way of validating input since it is relatively easy for an attacker to avoid being caught by such a filter. A better method is a white list

---

which instead defines the input that is allowed. This will in most cases be done by comparing the input to a regular expression designed for that input.

For tips on how to write regular expressions see: <http://www.regular-expressions.info/>

The following page contains a lot of ready-to-use regular expressions and examples: [https://www.owasp.org/index.php/OWASP\\_Validation\\_Regex\\_Repository](https://www.owasp.org/index.php/OWASP_Validation_Regex_Repository)

### **Preventing Cross Site Scripting (XSS)**

To avoid execution of malicious code we must encode all data before it is returned to the HTML page[38]. For example `<script>` would be encoded as `&lt;script&gt;`. For free text input inserted into HTML elements we should use the sanitizer in the angular module `ngSanitize`. Returned data not placed in the HTML body will need to be encoded specifically for the environment into which it is inserted. That is to say, if we insert user input into javascript data values we must escape javascript syntax from that input.

For more specific rules consult the OWASP XSS Prevention Cheat Sheet:

[https://www.owasp.org/index.php/XSS\\_\(Cross\\_Site\\_Scripting\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

### **File Uploads**

Since we will most likely allow users to at least upload profile pictures we need to validate these files [37].

- Upload verification: we need to match file name to expected file extension and ensure that the file is no larger than a reasonable maximum size.
- File storage: we will use a system generated temporary file name when storing user files on the server. All uploaded files will also be scanned for malicious code.

### **Email Address Validation**

Validating email addresses according to RFC 5321 should be done in the following way:

- Check for at least one @ symbol in the address.
- Local portion no longer than 64 octets Domain portion is no longer than 255 octets
- Ensure address is deliverable

Performing the last step can only be done by sending an email and having the user reply. The identifier used to verify the address should expire when used by the user or after eight hours if not used.

### **Authentication**

Authentication is the process of ascertaining the identity of a user.

**User ID**

User ID will be the user's email address. Though the local portion of the address is case sensitive as specified in RFC 5321 which defines the SMTP protocol, most email providers and Internet Service Providers do not enforce case sensitive addresses. Indeed, most users would be surprised and confused if case sensitive addresses were allowed. For this reason we will not allow case sensitive addresses. The email must be validated and verified as per the Email Address Validation section of the Input Validation guideline.

**Password Strength**

The rules in this section should be displayed on the password creation/change page. All rules that are broken should be displayed when a user fails to create a password [39].

**Password Length**

Lengthwise, we should not permit passwords that are shorter than 8 characters. If a user chooses a password shorter than 10 characters the application should inform the user that this password is considered weak. Max length should be 128 characters to allow for passphrases. A passphrase shorter than 20 characters is considered weak if it only contains lowercase Latin characters.

**Password Complexity**

In addition to being case sensitive we will require a certain minimum complexity for all passwords. The following rules should be enforced:

1. No more than two identical characters in a row.
2. They must meet at least three out of the following complexity rules:
  - At least one uppercase character.
  - At least one lowercase character.
  - At least one special character, space included.
  - At least one number (0-9).

**Password Recovery**

It is a good idea to require users to add a second non public email address to their account, known as a side-channel. As soon as the "reset password" button or link is clicked the user's account must be locked. We can then send a 10 character random generated string, a token, to the side-channel address along with a one time link to a simple HTML form containing input for the token, a requested new password and a field verifying correct entry of this password. The link and token should have a validity time of 20 minutes. Be sure to apply the password strength rules here.

Preferably this should all be logged. Data such as IP address, time and browser information can be used to detect suspicious behavior.

**Password Storage**

User authentication data must be protected against a variety of attacks, e.g. rainbow tables, brute force and so on. To achieve good protection each piece of authentication data, together with a cryptographically strong fixed-size random value called a salt, should be run through some form

of protective function. This should be a strong one way transform called a hash function, or a strong encryption algorithm. The use of a salt will ensure uniqueness when stored in the database and increase the information entropy of the stored authentication data without the need of great complexity of the user's original input. The resulting protected data is in protected form. Whenever a user inputs authentication data this will be transformed into protected form, which is then compared to the protected form stored in the database. The salt needs to be appended to the protected form since access to the salt is needed for comparison.

### Implementation

Since we are working with Node.js we have access to the crypto module. This gives us some strong tools that should ensure secure storage of authentication data.

- For generating the random value salt we shall use the randomBytes method. Preferred length is 64 bits, but 32 bits should be ok if storage space is a factor.
- MD5 is NOT considered a strong hash function[40]. SHA1 is the better choice, though that is also considered lacking. Preferably we will use PBKDF2[41]. This method needs four inputs: password, salt value, n iterations and the length of the resulting hash. The method then hashes the password input n times (10 000 times or more if possible). This should result in a very secure protected form that also takes a sufficiently large work factor. A work factor is the amount of work it takes to turn the authentication data into protected form and it should be large enough so that an attacker cannot realistically crack the protected for with a brute force attack, but small enough that the user isn't inconvenienced. Adjusting the iterations in step with advancements in technology so that it takes close to one second for the average user to perform the protect function is advised.

### General protected form pseudo code:

```
[protectedform]=[salt]+protect([protectionfunc],[salt]+[credential]);
```

Which in our case becomes:

```
[salt]=randomBytes(64bits)
```

```
[protectedform]=[salt]+[PBKDF2([credential],[salt],10000,256)]
```

For more information on crypto in Node.js read: <https://masteringmean.com/lessons/46-Encryption-and-pass>

Note: the Software Skills site doesn't use PBKDF2, instead it uses the bcryptjs module that is not a part of the crypto module. bcrypt also has a large work factor. See: <https://www.npmjs.com/package/bcryptjs>

### Failed login

#### Error Messages

The error message given when a user fails the authentication process should be generic and give no information on whether the username or the password was incorrect [39]. A good error message is: "Login attempt failed - invalid username or password".

### **Account Lockout**

To further strengthen the application against brute force attacks it is beneficial to lock an account, and its password check, for 20 minutes if five consecutive failed attempts to log in are made. This means we will have to store number of consecutive failed attempts together with date/time data. Also be sure to display the number attempts left before lockout occurs.

### **Logging**

It is recommended that all password failures are logged, as well as information about all account lockouts.

### **Re-Authentication for Sensitive Features**

To Combat Cross Site Request Forgery (CSRF) and session hijacking it is a good idea to have users re-authenticate when performing sensitive actions such as accepting or canceling contracts, approving pay-outs etc.

### **Session Management**

Session management and authentication are closely linked. Here are some guidelines for keeping the now authenticated session secure.

### **Secure Transmission (TLS)**

For all transmission of sensitive information we must ensure end-to-end encryption. This is done through Transport Layer Security (TLS) which implements Public Key Infrastructure (PKI) over the TCP protocol[42]. The server is validated to the client by a certificate issued by a trusted root authority. If configured for it the client can also be validated to the server through a similar process, but that is not likely to be the case in the Software Market. An older version of this, not considered secure today, is Secure Socket Layer (SSL).

The protocol to use will of course be the one Software Skills use for their existing site. The same goes for the encryption algorithm used in the PKI.

- For the application to achieve secure transmission we must transmit over TLS not only when logging in, but for all authenticated pages. This includes images, scripts and css.
- Never switch from HTTPS to HTTP or vice versa within a session.
- Never have a HTTP version of a page that is also on HTTPS.
- Set the “Secure” flag in all user cookies. If set the cookie must be sent over HTTPS.