AALBORG UNIVERSITY
COPENHAGEN

Semester: ICTE4, Master Thesis

Title: Common password

Project Period: Spring 2017

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Semester Coordinator:
Henning Olesen

Secretary: Maiken Keller

Semester Theme:
Master thesis

Supervisor(s):
Henning Olesen
olesen@cmi.aau.dk

Project group no.: 4SER 4.6

Members:

Anton Kivimäki

Harjit Singh Sahota

In recent years, the amount of online services, social medias etc. have increased tremendously. These services require user-profiles and are protected by passwords which are required to be strong and complex to avoid compromising the profiles. However, according to research made in this thesis, since the average internet-user has more than five user-profiles it might be difficult to memorize all the complex passwords hence users tend to either create weaker or duplicated passwords. A survey supported the outcome of the this research and validated that users tend to use the same password for multiple services.

Password managers might be a solution to managing passwords which authenticates users from one master passwords, but the managers are usually difficult to use in enterprises. Enterprises tries to solve the password management-problem by using service providers that supports Single-Sign-On (SSO) but not all service providers support SSO.

The group members developed a proof of concept called ID-Connect with WSO2 Identity Server which worked as a backbone for users' identity.
ID-Connect eased password management in enterprises while supporting service providers with or without SSO. Service providers without SSO was solved with formfill, a feature service implemented in ID-Connect.

Pages: 117
Finished: 7th June 2017

COMMON PASSWORD

One password to rule them all

By:

Anton Kivimäki

Harjit Singh Sahota

Table of content

1 Introduction

The amount of online services, whether they are social media services, web shops or online banking services etc., are growing every day. These services are built around user-profiles in order to deliver personalized and contextual experiences, which mean that these services require registration to the service by entering a username and password. The services often require strong, complex and not-easy-to-guess passwords, where some of the passwords requirements could be minimal length of eight characters, mixture of uppercase and lowercase characters, special characters and numerical values. Since the average user of the internet are using more than five social medias, owns several email addresses, are registered to social services and online bank accounts, the consumers need to remember many passwords. (Mander J., 2015) Remembering all these passwords from the different services can be overwhelming for an individual and according to a study by the british telecommunication company *Ofcom*, "*Four in ten internet users say they tend to use the same passwords for most websites*" (Ofcom, 2016).

If a hacker gains access to one password from a small start-up company, and the owner of this password also uses this password for his bank account, the hacker could gain access to confidentiality-critical and integrity-critical information and cause severe damage. An example of a severe damage was described by Jyllands-Posten with the headline: "*Danske personoplysninger er billigt til salg på nettet*[1]*"* (Ritzau 2016). The hackers, who stole the Dane's personal information, attacked different webshops and gained access to users' passwords were some of these passwords were also used to authenticate the users' NemID and Borgerservice. The victim's identity were then sold for a mere 98 kr on the internet.

1.1 Challenges for Enterprises

Enterprises are also dealing with these problems, since the enterprise itself and its employees often uses online services such as email accounts, services and applications that all require passwords. Duplicates and weak passwords can be a potential threat for the companies which might compromise confidential data. This could also be a contributing factor to the rise of cyber security threats and attacks on small and medium-sized businesses (SME), which FireEye states; "*Over 77% of all cyber crimes target small and midsize enterprises and yet, research shows 42% of small and midsize businesses don't see cyber crime as a risk*" (FireEye, 2017). The popularity of attacks on SMEs are due to their limited budget and unwillingness to invest in cybersecurity compared to large companies, since they do not see cyber crime as a risk. However, SMEs tend to cooperate with larger companies and by compromising the smaller companies the attackers might gain access to the larger companies, thus leaving these companies at risk.

[1] Translated from Danish to English: *Dane's personal information are on sale for a cheap price.*

Enterprises also face problems with former employees who still have access to enterprises resources. A malicious and angry former employee could cause an enterprise great damage by sabotaging the enterprise’s resources which he/she still has access to. An example was the big incident on new year's eve 2016 where a former employee from TDC still had access to the company's resources and cause severe damage resulting into many Danes could not watch television that day (Secher, 2017). This could be avoided by having a system allowing enterprises to easily deauthenticate or unenroll former employees so they cannot access the protected resources which are critical in enterprises.

1.2 Current Solutions for Enterprises

To overcome the problem of weak and duplicate passwords, many services are implementing authentication standards, like OpenID Connect or SAML, to authenticate their users instead of building their own custom authentication system from the ground-up which is time-consuming and complex if done correctly. Using standards reduces the amount of user-accounts, that requires passwords to remember (Kasahara, 2006). The standards also delivers ways to access protected resources from third-parties in a secure manner for instance, a webshop, that sells shoes online, has implemented Facebook-login and wishes to access fine-grain restricted resources from a user on Facebook. This is a great way for the company, that sells shoes, to only access a user’s required scopes from Facebook and not its password credentials, which the webshop might manage insecurely. Protecting resources in an enterprise is critical since compromising and revealing confidential data to i.e. competitors could damage the enterprises businesses. However, if the company has implemented a sharing feature to share their products on Twitter, Google Plus or other social networks, the user needs to authenticate on all the service providers. This can particularly be troublesome for an enterprise, which might have many service providers the employee must use.[2]

A solution to this problem is Single-Sign-On (SSO), which is a common password management solution allowing users to authenticate once and then access all the resources the user is authorized to use without additional authentication. For instance, a user might authenticate itself when using service “*A*” for the first time, but if he/she uses service “*B*”, the user do not need to authenticate again. SSO creates a unique and strong password for each resource and changes this password regularly. The user is not required to have any knowledge or memorize this password but only the SSO-password, which in terms will be described throughout this report as the master password. However, there is a problem of having one passwords which unlocks all the other ressources. If an attacker gains access to the master password they have access to all the associated resources which could put the SME at risk. Implementing multi-factor authentication (see more in section 3.3.7 Password

[2] More details about the whole process of the mentioned authentication standards can be found in section 3.3.3 OAuth.

Theory) when authenticating with a master password could mitigate this issue, since hackers need the additional factors. (Khajuria, 2017)

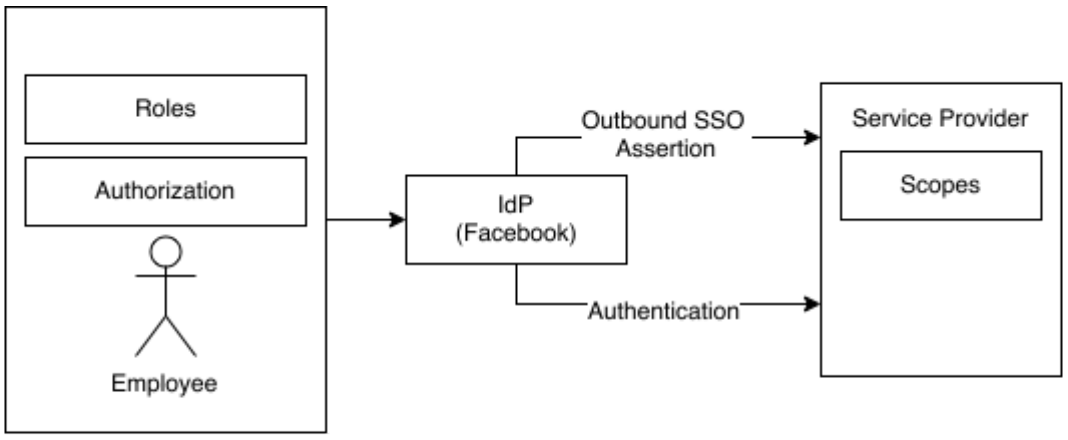


Figure 1 - Illustration of an employee who authenticates with a Service Provider.

Figure 1 illustrates the principle of SSO in an enterprise. An employee has been given roles and authorization to access services to use. To use the services the employee must authenticate via Facebook which sends an *Outbound SSO Assertion* to the *Service Provider* to authenticate the employee. Then the *Service Provider* can authenticate the employee.

A problem with SSO is the lack of interoperability between services. For instance, Google supports SSO with their applications which works seamlessly with Google's products, but it does not work well with Microsoft's products, who also have implemented their own SSO (Kelly, 2002). Fast Identity Online (FIDO) solves the problem of interoperability among authentication devices and the problems users face creating and remembering multiple usernames and passwords. FIDO Alliance removes the need to remember a password but uses tokens in form of biometric, devices, which are never stored on the services, limiting attackers changes to succeed. Although FIDO Alliance provides excellent mechanism to avoid passwords the solution might be too complex for SMEs, who lack technical knowledge to implement. (Khajuria, 2016)

Another way to avoid weak passwords have been addressed by password managers where users authenticate with their master password to access different services. The password manager automatically then creates strong passwords for each services which the user do not need to memorize. However, many of these password managers are complex to implement in non-technical enterprises, and are mainly aimed towards private usage where only one person can authenticate with his master password. Creating user-profiles to manage other users, such as a parent managing its childrens user-profiles or a company managing its employees, is complex for non-technical users (further elaborated in section 3.2 Password Manager). Additionally, the current password managers do not have the ability to quickly deauthenticate or unenroll a former employee, which is a need as explained earlier.

1.3 Idea and Motivation

The group members for this master thesis have also experienced the above difficulties of remembering all their passwords, which resulted into creating weak passwords. One of the group members have been a part of a start-up company and experienced hand-on the trouble of managing his own, his business partners and his employees passwords for services like email accounts, cloud services and applications. Storing the passwords was a problem and often ended up being stored as plain text in documents, physical papers or in email conversations.

Enrolling and registering new employees to services such as email, was troublesome. The group member would create very complex and not-easy-to-guess passwords and deliver them to the employees, however the employees would then change the passwords into much weaker passwords since they were easier to remember. Password managers did not solve this problem since he needed to create a profile account on the password managers for each employee, and there were no clever way to control all these profiles from a central point. The group member also experienced the problem of having a malicious former employee who caused damage to the company by purchasing good on behalf of the enterprise, since he still had access to the company's resources.

1.4 Problem Formulation

From considerations and problems described in the previous sections, the group members found a need for a system which would allow SMEs to manage their employees passwords and formed the following problem formulation:

How can a system be built that would potentially help SMEs securely managing its employees passwords, so the employees can authenticate or deauthenticate multiple services with one master password?

1.5 Conceptual Idea

For this master thesis the group members must develop a proof of concept which would help both answering problem formulation and the problems previously stated in this chapter. Therefore, the group members decided to develop a system that could help SMEs to manage their employees passwords securely.

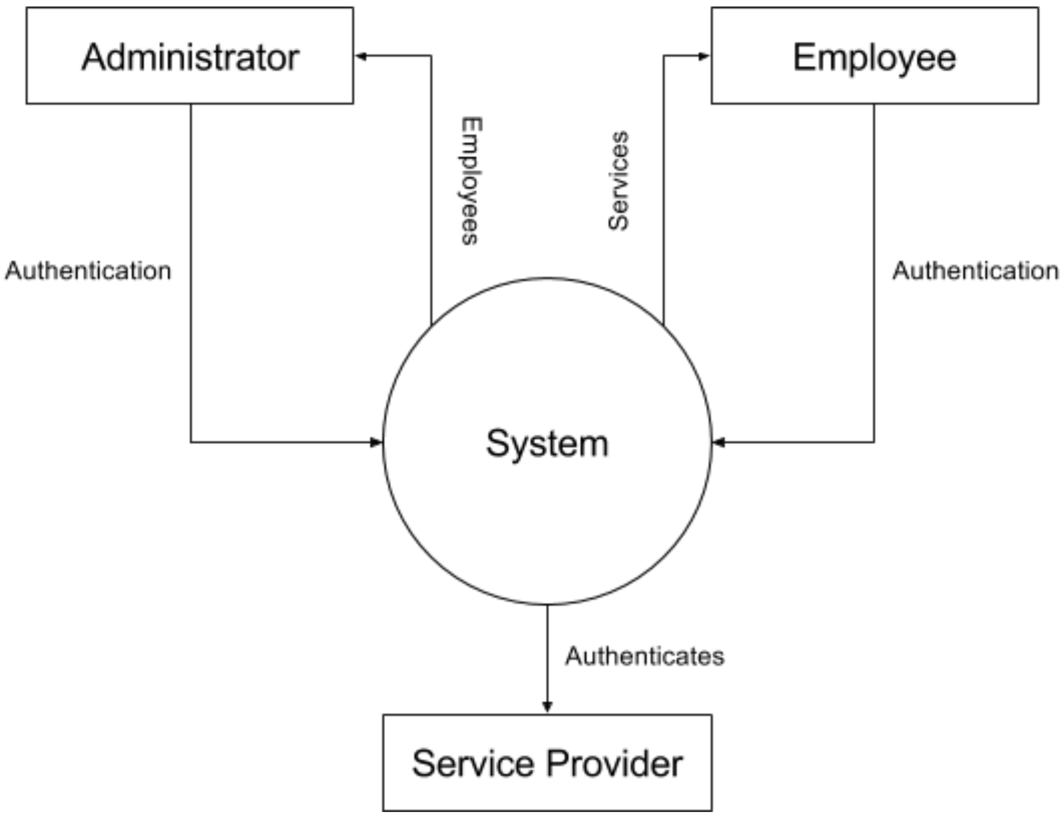


Figure 2 - A context diagram of the conceptual idea for this project.

Figure 2 illustrates a context diagram of the system. It represents an enterprise with an *Administrator*, who is one responsible for the company's security, the company's *Employees* dealing with the enterprise's business, and how the flow of authentication would work in an enterprise.

The *Administrator* would first authenticate itself with its master password. Then the *System* would return a list of all the employees which the *Administrator* is authorized to access, in order to enroll and unenroll an employee from services provided by the *Service Providers*.

The *Employees* authenticates themselves with their own master password, which has a unique relationship with the *Administrator's* master password. The *System* returns all the services from the *Service Providers* an *Employee* is authorized to access, from where the *Employee* can authenticate itself and use the services.

1.6 Delimitations

Due to this project's limited resources parts of the conceptual idea are excluded from the proof of concept. Aspects as the GUI will not be focused on but instead the core structure will be the main focus area for the development. The target group for the system, called ID-Connect, will be reduced to SMEs and not private users, since the main focus is enterprises even though the system could potentially be used by private users.

Additionally the following bullet points are excluded from this project:

- Business aspect of the system
  This project will not address the business aspects and market analytics of the system.

- Math behind algorithms
  The extensive math behind the cryptographic algorithms will not be a focus-point in this project.

- User interface
  The proof of concept will only provide a basic user interface which shows the concept of the system. However, there will be a short introduction of how a low-fi design helped the group members to form the system in section 5.3 Graphical User Interface.

- Legal aspects of privacy
  Although a discussion of General Data Protection Regulation (GDPR) will be presented for future development of the system, the current solution of the system and analytics will not address the legal aspects of user's privacy. However, best-practises learned throughout the courses on this master thesis will be respected to ensure user's privacy by exposure minimal amount of user's data.

- Front-end of WSO2
  For the backbone of password management WSO2 Identity Server will be used but only the core functionalities and not front-end tools such as Management Console, Dashboard etc.

- Testing the system with real users
  The system will be tested with unit testing and not with real users due to time limitations and the lack of a full functioning Graphical User Interface (GUI).

2 Methodology

This chapter describes how the methods of gathering the required information helped answering the problem formulation along with the reasons and thoughts behind the methods. The chapter also focuses on the general process of creating this thesis.

2.1 Idea generation

As described in chapter 1 Introduction the idea and motivation came from a group member's experience when he was a part of a startup company, and with personal experience as remembering a lot of passwords. With this idea, the group members used mind-mapping as a technique to further work with the idea and see connections between different aspects of the idea. For the sake of this thesis's limitation a compressed version of the mind-map is illustrated on Figure 3.

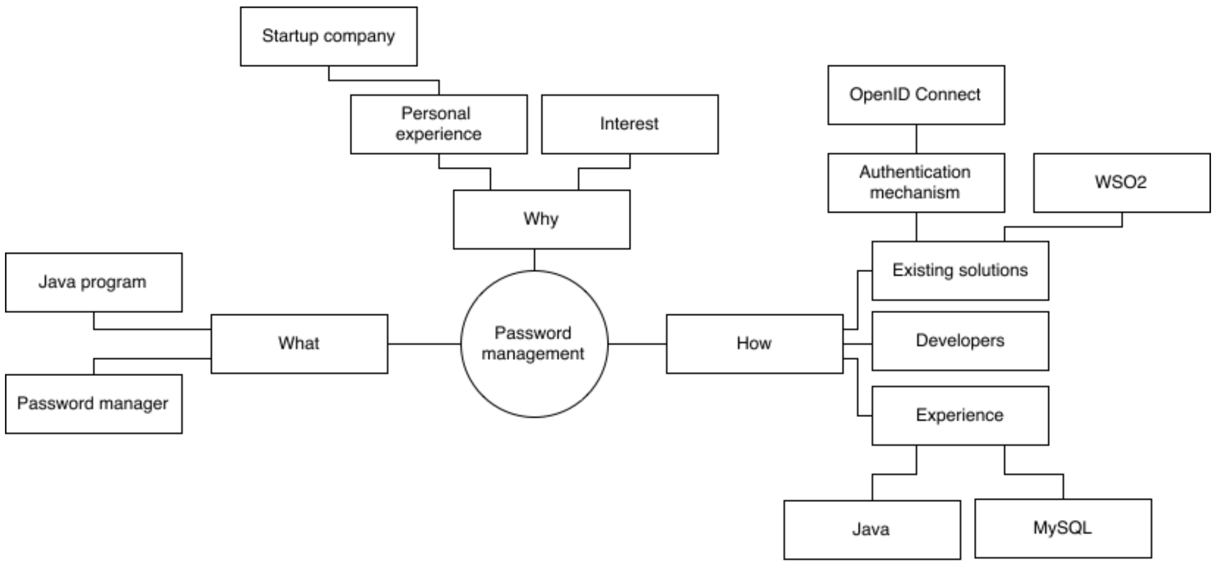


Figure 3 - A compressed version of the mind-map made for this thesis.

The circle in the middle described the idea as *Password management*, and from here the following three questions were asked, why, how and what.

<u>What</u>
The group members thought of developing a password manager, that would solve the issues explained in chapter 1 Introduction. This would be programmed in Java due to their experience in this language.

Why

The group members needed to reflect on why they have chosen this topic. It was clear that the reason was due to the experience with the startup and their personal experience with remembering a lot of passwords.

How

The members elaborated on *How* they could implement such system in this thesis. By using *Existing solutions* and technologies they could use already existing authentication mechanisms such as OpenID Connect instead of reinventing the wheel. The group members also stumbled upon WSO2 which had powerful frameworks which are explained in section 3.4.1 WSO2. The group members *Experience* in programming was both Java and MySQL which they used to develop the system.

2.2 Development Process

An incremental approach was chosen for the development process, since the group members would have the possibility to reevaluate the priorities of the system's features and adjust when necessary.

2.2.1 Scrum

For the development process of the thesis the development-method Scrum was used. Scrum is an agile development method which means that it is fairly dynamic. The basics of Scrum consists of a sprint phase and an evaluation phase. In the sprint phase a set goal is determined for a specific time and during this time there will be various evaluations phases in order to determine if the development is going as planned. The reason for using the Scrum method for the thesis development was its suitability for a small group of two and its flexibility due to the evaluation phases. This gave the project the wanted flexibility to easily and with coordination jump from task to task. (Schwaber, 2002)

2.3 Literature Study

Literature study was conducted through scientific papers, online articles written by experts and course material when the group members investigated theories behind passwords such as cryptographic encryption and hashing. The reason behind the scientific papers and course material is these theories are well-defined and standardized. Therefore a qualitative approach was made over a quantitative for the literature study. The literature study was done in order to better understand the technologies which are already created and how to better user them in the thesis project.

2.4 Questionnaire

Quantity research was used over qualitative as a research-method to verify the problems that were started in chapter 1 Introduction. Especially the problem of users reusing the same password for multiple services was interesting to investigate for employees in SMEs, since there might be a greater responsibility in enterprises compared to private usage. Investigating a large audience of different SMEs would give a better picture of how employees would perform in different enterprises.
Interviews, which is a qualitative research methods were also discussed at the beginning, but it quickly came to mind, that interviewing a large audience would be too complicated for this project.

The questionnaire was distributed to two Facebook groups. One was the largest facebook Facebook-group in Denmark for startups called *Iværksætter Netværk*[3] which has a membership of more than 30.000 entrepreneurs. The second group called *Seedster 2.0* consisted of approximately 200 members, which was chosen since one of the group members attended a startup-course with the 200 members which consisted of both new and experienced entrepreneurs. The online questionnaire allowed continuously collection from the audience and was running while the remaining tasks for this project were executed, which was time-efficient for the project.

However, as with all questionnaires, there were no guarantees of quality answers, which could be achieved through an interview and also people could answer maliciously. This drawback was easily discarded since interviewing 30.000 persons would be overwhelming and questionnaires was a great tool used to identify and graphically visualize relevant tendencies of SMEs password managements.

[3] Translated from Danish to English: *Network of entrepreneurs.*

3 State of the Art

This chapter will first present how enterprises manage their employees identities with existing solutions such as password management tools and password managers. Sub-conclusions for each solution will be presented which later will be used in the requirement specification.

3.1 Enterprise Identity Management System

As briefly explained in chapter 1 Introduction, SMEs way to deal with remembering all the passwords could be done with password management systems built on SSO. The following identity management systems have implemented SSO and are currently being used by enterprises. There will be explanations for each management system along with their pros and cons.

Figure 4 illustrates how the SSO are architectured in an enterprise. This figure will be the foundation of the explanation of identity management systems for enterprises.

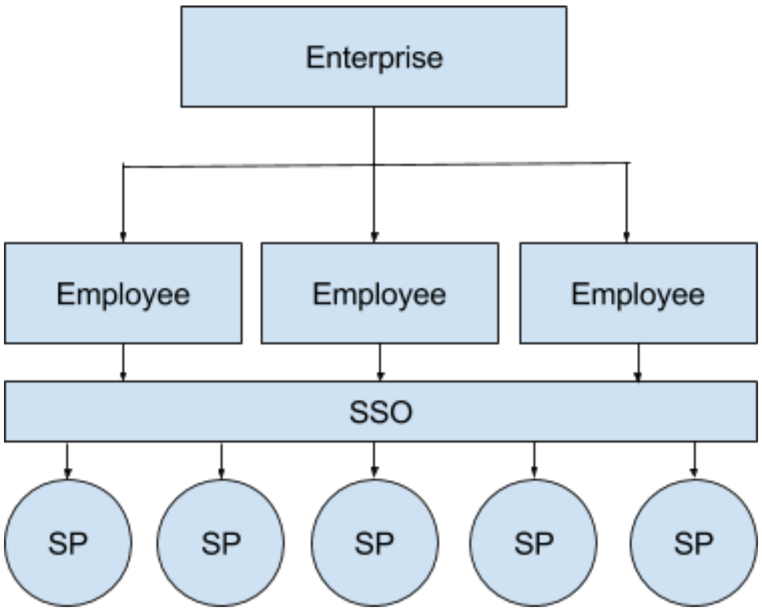


Figure 4: An illustration of SSO are used in enterprises.

3.1.1 Active Directory

Active Directory (AD) was developed by Microsoft and was initially used for Windows-domain networks, that was a part of most Windows Server, Operating Systems (OS) services. Later, when Microsoft focused on cloud computing, they added many functionalities to AD when merging with Azure - their cloud computing solution (Microsoft,

2017) . Microsoft added SSO within Azure, called *OneLogin*, allowing Microsoft's users to authenticate once in their products, and thereby not needing to authenticate again (OneLogin, 2017). This works great with Microsoft's products however, as explained in the chapter 1 Introduction, the problem with SSO lies on the interoperability. If an employee wants to use Google's products instead, they would have to authenticate again. This could be avoid by enterprises implementing policies requiring the employees to only use certain applications with SSO.

3.2 Password Manager

Password managers are another way to deal with the password-management problem. The following sections explains the different existing password managers that are client-based and browser-based. A high-level overview of a password manager can be seen on Figure 5.

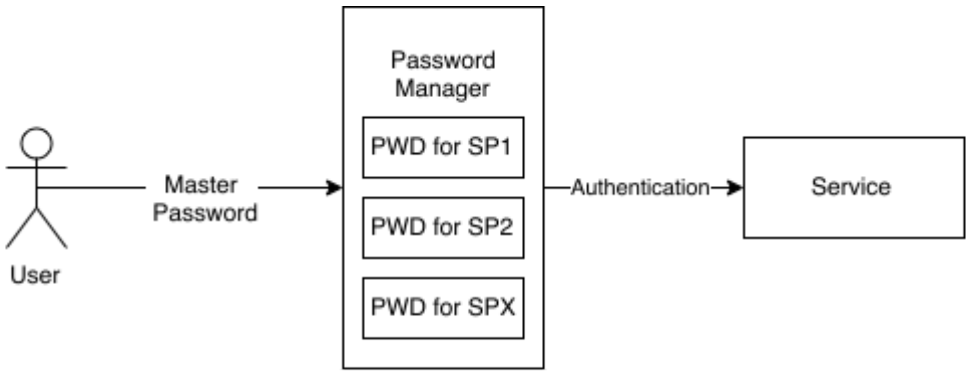


Figure 5: A high-level overview of a password manager.

The *User* authenticates with its master password to the *Password Manager*. When the user registers creates profiles at service providers, the *Password Manager* encrypts the passwords from the service providers with the *Master Password*. In this way, a user must authenticate with the user's *Master Password* to user the services.

Password managers exists for many platforms. The following sections will address the two platforms client-based and browser-based.

3.2.1 Client-based Password Managers

This report defines client-based password managers as software that is installed either as third-party or native application on an operating system, regardless of desktop computers or mobile devices.

3.2.1.1 Dashlane

Dashlane is a client-based password manager available on Windows, Mac OS, iOS, Android as third-party applications, and as plugins for Google Chrome and Firefox. The password manager contains various of features like automatically filling out a form with a user's credentials and information that has been fetched from Dashlane's secure data storage. This means that the user does not have to do it manually, generate and change complex passwords for service providers and all together with having Dashlane securely storing everything online. (Dashlane, 2017a)

<u>Storing passwords</u>
Dashlane stores their users passwords locally if they are using the free version of Dashlane, and is encrypted with Advanced Encryption Standard (AES) 256-bit with the user's master password, hence the users can only decrypt other passwords with the master password. Dashlane also offers premium accounts allowing users to store and synchronize their passwords across multiple devices. However, the master password is not stored anywhere making it difficult for hackers to steal the master password, since it is not stored anywhere other than inside the user's mind assuming he/she has not written it down on paper.
Dashlane also supports multi-factor authentication (email link, SMS, tokens) when a user wants to authenticate with his/her master password. On the other hand, forgetting the master password requires the user to start the setup-process all over again which greatly decreases usability since it might be a hassle for users to enroll again. (Dashlane, 2017b)

Dashlane has a feature called Password Generator which generates strong and complex passwords which can replace user's passwords for their service providers. This means a user frequently can generate strong passwords. (Dashlane, 2017c)

<u>Enterprise version</u>
Dashlane is mostly used for personal usage, but *Dashlane Business* allows enterprises to control their employees authorization-access for different service providers through a web interface. A CSO authenticates itself with its master password, and can then browse all its employees and which service providers they are authenticated with. (Dashlane, 2017d) The pricing for Dashlane Business depends on the number of users but cost between 1,25 - 2,00 dollars for each user with a thirty days of free usage. (Dashlane, 2017e) The group members tried the enrollment-process which seemed intuitive and easy to use, which is also backed up by users who has reviewed the password manager. (Trustpilot, 2017)

<u>Sub conclusion</u>
Dashlane is a mature password manager with many useful features along with the ones presented in this report. Dashlane Business allows SMEs to control their employees authentications on different service providers in an easy manner though for a cost. The choice

of not storing the *master password* anywhere can be troublesome for users who might forget it, which requires them to go through the setup-process again.

Another drawback with Dashlane is its OS-dependency. A user has to install another browser-plugin i.e. Google Chrome if he/she seemingly want to use Dashlane in a browser . Otherwise the user is required to open the application and authenticate through there.

3.2.1.2 LastPass

Functionality-wise LastPass is very similar to Dashlane apart from being able to securely store notes in the same manner as passwords. It also arrives as third-party applications for Windows and Mac OS and plugins for browsers, but provides a standalone web interface if a user does not want to install anything on its device.

Storing passwords
LastPass also uses AES-256 bit encryption for their passwords with the master password. For securing the master password LastPass uses *SHA-256* hashing (further explained in section 3.3.6.2 Hashing) with Password-Based Key Derivation Function (PBKDF2) to create a hash-digest from the master password and an encrypted key, which is a user’s private key, and sends the digest to LastPass servers but never the encrypted key. This key is used for decrypting the user’s passwords and is only stored locally on the user’s machine and never on LastPass’ servers. LastPass also have the same functionality of generating strong complex passwords for their user’s service providers. (LastPass, 2017a)

Enterprise version
LastPass offers a solution to enterprises called LastPass Enterprise which is similar to Dashlane Business. The functionalities are more or less the same providing a client-interface where a user can create groups, so a user can manage other users’ password-management This idea can be applied to SMEs’ employees, so an administrator can control the employees’ identities and passwords.

LastPass is slightly more expensive compared to Dashlane with $2,42 monthly per user for private usage and $4 for enterprise solution. (LastPass, 2017b)
The group members also tried the enrollment-process which was easier than Dashlane if the user uses Google Chrome or Safari since LastPass wants the user to install its plug in for the respective browser.

Sub conclusion
LastPass shares similar functionalities to Dashlane and also offers a business version of its solution. LastPass has openly specified that they locally store a hash-digest of a user’s master password which brings the same drawbacks as Dashlane’s that a user who forgot its master

password has to enroll again. Another similar drawback is the dependency of being client-based forcing users to install software on their browser or computer.

3.2.2 Browser-based Password Managers

Browser-based password managers are integrated directly into browsers and does not require any installation from the users, which is required for client-based password managers. The following sections will go through the browsers Google Chrome and Firefox.

3.2.2.1 Google Smart Lock

Google Smart Lock is a password manager within the browser Google Chrome. The feature does not require any installation or setup from the user other than signing into its Google account. The user's Google account's password is used as the master password, so if a user wants to store their passwords from service providers, the user has to authenticate itself to Google and then it can save and retrieve its passwords from the password manager. Auto form filling is also used which fills a user's name, address, age, passwords etc. (Google, 2017)

<u>Storing passwords</u>
Unlike Dashlane and LastPass, Google Chrome securely stores its user's master password on Google servers, since the master password is from the user's Google account, which can be fatal if attackers hacks into Google's databases. However, Google has an optional feature called "sync passphrase" which allows users to encrypt all data that is synchronized across Chrome like history, saved bookmarks etc. The passphrase is used as the master password which is stored locally on the device (PC or mobile), hence all the data is encrypted locally and will not be accessible from other places even from Google. (Google Chrome Help, 2017)

<u>Enterprise version</u>
Google Chrome does not have an enterprise version integrated within the browser, however Google has extended its product to a client-based password manager and called it "Smart Lock at Work". The service is free and works well with "G Suite" which are enterprise-applications from Google (Google Cloud, 2017).

<u>Sub conclusion</u>
Google Smart Lock is an easy-to-use password manager and works well if the user uses Chrome as its internet browser. Using another browser will leave the user without having their passwords, since Chrome only allows the passwords from Chrome. Unlike the client-based password managers the master password is not stored locally but on Google's servers. This can be a problem if hackers gain access to Google's servers and retrieves the master password and thereby gaining access to user's password. However, if the users forgot their master password it is possible for Google to reset it eliminating the need for users to enroll more than one time.

Most importantly Chrome does not support multiple users and cannot be used in an SME to control employee passwords, but this can be achieved with their client-based solution “Smart Lock for Work”.

3.2.2.2 Firefox

Firefox stores their users logins in the “Firefox Password Manager” and in cookies. The password manager allows its users to store the passwords without a master password within itself. This means if a user has setup the password manager on his/her computer without a master password and leaves the device, another user can simply open the manager and see all the passwords in plain text. But using a master password, the user has to authenticate itself before accessing the password, which Firefox recommends its users to do.

Storing passwords

The passwords are stored in a file called “logins.json” which are encrypted with the master password. Like the client-based managers the master password is stored locally on the device in a file called “key3.db” file from which the browser is installed on. Changing the master password will erase all the saved passwords, so the user has to enroll again, which also applies to the presented client-based password managers.

Enterprise version

Firefox does not support the feature of having multiple users to control other users’ identities.

3.3 Relevant Theories

After looking into existing enterprise identity management systems and password managers the focus will shift onto the relevant theories of the thesis. The 3.3 Relevant Theories section will feature Identity Providers, OAuth, OpenID Connect, SAML and Secure Data Storing. These theories will be an essential building block in the future sections of the thesis and are meant to cast light on theories relevant to the proof of concept.

3.3.1 CIA Triad

The NIST Computer Security Handbook defines the term computer-security as: "*The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and*
confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications)" (Guttman B, 1995).

Figure 6 illustrates the CIA triad, consisting of three concepts of the fundamental security objectives for data and computing services. These concepts must be available to define the security objects, meaning if one is missing out, the system can break. (Khajuria, 2014)

- Loss of *confidentiality* is caused by unauthorized disclosure of information.
- Loss of *integrity* are caused by unauthorized altering of information.
- Loss of *availability* is the denial of access to information.

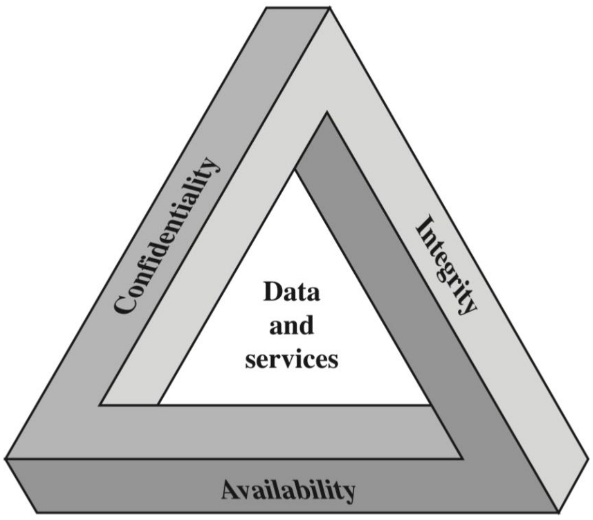


Figure 6: An illustration of CIA triad [source: Khajuria, 2014]

The *C* in *CIA* represents *confidentiality*, which describes the secrecy of an asset. Data-confidentiality deals with private or confidential information that is not accessible to unauthorized users. Related to this project, a secret could be a password, which normally must be kept confidential from unauthorized users. Encryption is often used to ensure confidentiality in a system so only the authorized user with the correct key can access unlock the secret. (Khajuria, 2014)

Data *integrity* (the *I in CIA*) assures that information (bank-details, messages, emails etc.) are not altered or changed but only in a specified and authorized manner. Access-control ensures data integrity to prevent unauthorized users to alter any information. Lastly *availability* ensures a service or asset is not denied to authorized users in when it is expected to be available. (Khajuria, 2014)

3.3.2 Identity Provider

The idea of an identity provider (IdP) is to provide identities for users to interact with a system. For example this could be a case when a user is authenticating to a website via its Google account. In this scenario the website would need to trust Google to provide enough and correct information about the user.

The reasons for the use of an IdP are several. One would be that a website do not need to store a user's credentials on their database, since the IdP would be the only party that would need to have user data. Together with the user data amount being lowered, the user do not need to register to another service which eliminates the need to remember yet another strong password, which would be an improvement to the user experience. However arguably the most important part of using an IdP, must be the trust. This is due to the user being able to use an IdPs stored user data to authenticate towards a website they might not know and/or trust. This in term means that a user would not need to provide user data if an IdP is available and the user is registered under the IdP. An example of this can be seen in Figure 7.

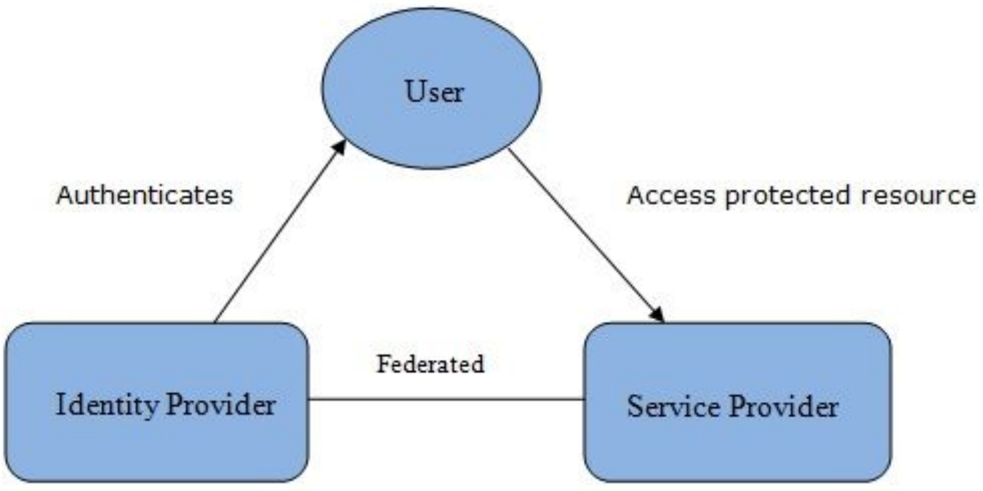


Figure 7 - Example of Identity Provider. [Source: Seshu, 2013]

In broad terms how the use of an IdP works can be explained in steps and by looking at Figure 7. These steps are explained as such below:

- User accesses a Service Provider.
- Service Provider prompts the User to choose an Identity Provider.
- User chooses an Identity Provider.
- Identity Provider prompts the User for verification.
- User provides correct verification for the Identity Provider.
- Identity Provider confirms the User's identity to the Service Provider.
- Service Provider grants access to the User due to the “identity” gained from the Identity Provider.

3.3.3 OAuth

OAuth is an open standard for authorizing access to applications’ resources. Users can grant restricted access to resources, which could be images, documents or audio-files, they own to third-party clients. Unlike the past, where the user was required to share their credentials with the client, which was of high security risk, OAuth allows user to grant limited access to their resources by providing a token. OAuth is important for enterprises allowing enterprises to manage and share their resources with third-parties. The way the sharing occurred was through APIs, where only authorized users (authorized via OAuth) could use the APIs. (CA Technologies, 2014)

3.3.3.1 OAuth 1.0

The first version of OAuth called, OAuth 1.0, was quickly adapted by social medias, such as Facebook and Twitter, since they encouraged integration with their services with third-parties through Representational State Transfer (RESTful) APIs. In the past, if a user wanted to post a tweet on its Facebook profile, the user would store its Facebook credentials in its Twitter profile, and whenever a new tweet was published, the Twitter application would sign in for the user to post the tweet onto Facebook. This would be a security risk since Twitter might have stored the user’s password insecurely, and a hacker could steal the user’s Facebook credentials leading to potentially severe damages. (CA Technologies, 2014)

3.3.3.2 OAuth 2.0

However, there were some problems with OAuth 1.0 and 1.0a. There were many libraries across multiple programming languages which supported OAuth 1.0 but implementing this mechanism in services required from largely do-it-yourself feel, this could appeal developers but not enterprises. An example could be OAuth 1.0 required clients to sign HTTP parameters which could be difficult to do with APIs resulting into frustrated developers and enterprises not wanting to implement this mechanism. (CA Technologies, 2014)

Luckily OAuth 2.0 helped solving some of these problems. OAuth 2.0 attempted to simplify client-development, and improved user experience, which required many changes not backwardly compatible with previous versions of OAuth. OAuth 2.0 explicitly separates the roles of authorization from access control and from the resource server, much like a classical SSO architecture. Figure 8 illustrates the flow of OAuth 2.0. (CA Technologies, 2014)

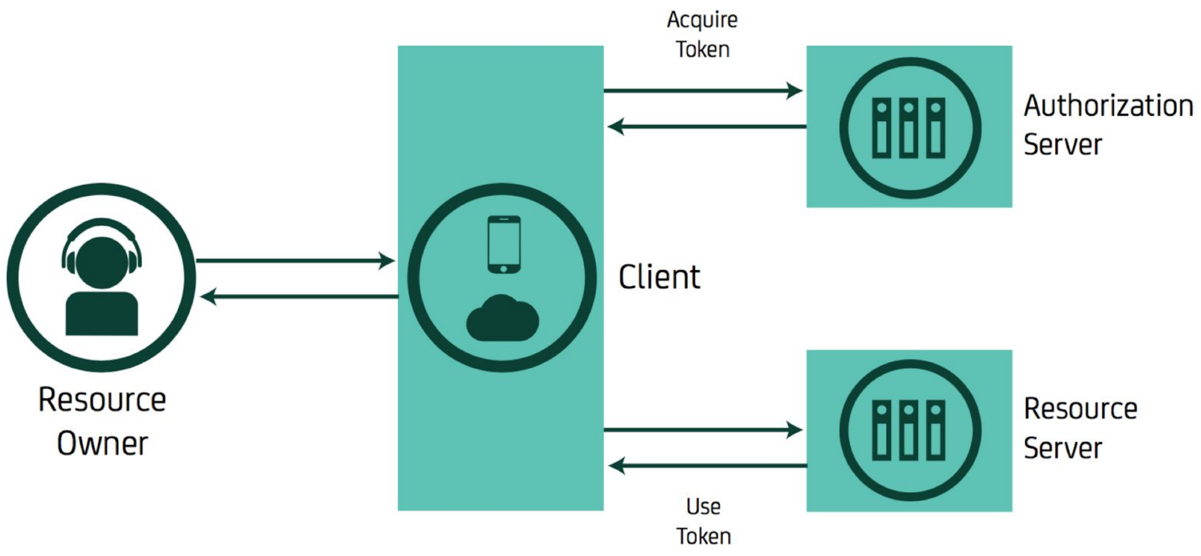


Figure 8 - Illustration of the clear distinction between Authorization Server (AS) and Resource Server (RS) in OAuth 2.0 [Source: CA Technologies, 2014]

The specification in OAuth 2.0 describes several grants, which are as following (CA Technologies, 2014):

- Authorization Code
  This grant delegates authorization from an AS to a client via a resource owner's user agent, without ever sharing the resource owner's credentials and *access token* with the client.

- Implicit
  This grant is simpler compared to *Authorization Code* by allowing the clients to fetch the access token from the AS, but can be a security risk since the RS can fetch the access token.

- Resource Owner Password Credentials
  This grant is only recommended between parties that highly trust each other. The resource owner shares its credentials with the client, from where the client can access obtain the access token.

- Client Credentials
  This grant allows the client, with its own credentials, to access a resource.

Another mechanism introduced in OAuth 2.0 was updates to tokens. Before, access tokens were very long-lived – some unlimited like Twitter, but OAuth 2.0 introduced short-lived tokens and long-lived authorizations, where AS now could issue refresh tokens to clients, which are long-lived tokens a client can use multiple times to acquire short-lived access tokens. In this way resource owners can easily block clients from continually acquiring new access tokens if desired saving resources on the owners servers. (CA Technologies, 2014)

It is important to notice that OAuth is only meant for authorization and not authentication. A method inspired by OAuth used to authenticate is called OpenID Connect, which is explained in the next section.

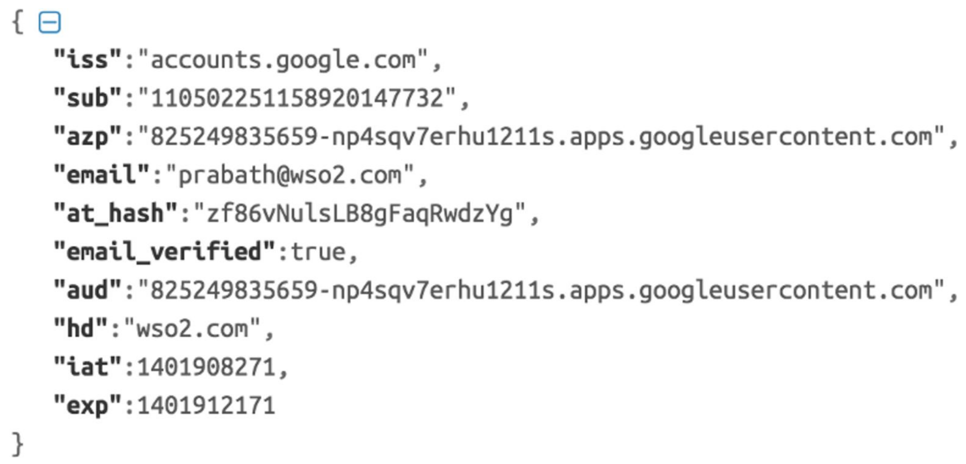
3.3.4 OpenID Connect

OpenID Connect is a standard for SSO and identity provision on the internet. The whole idea of OpenID Connect is to authenticate users via IdPs, so client avoid creating complex authentication systems and manage its users credentials. The users also avoid creating new user-profiles, that requires passwords which potentially could be weak (Olesen, 2017).

OpenID Connect is built on top of OAuth 2.0 and authenticates users through identity tokens (ID token). ID tokens are constructed in JSON format and which are called JSON Web Token (JWT). The tokens can be signed by the AS and are called JSON Web Signature (JWS). The ID token gets returned from an AS that supports OpenID Connect along with the access token. (Olesen, 2016a)

ID token

ID tokens delivers the authenticated user's information from the AS to the client. The token is *base64 encoded* so it can be sent via the URL. JWT follows a structure defined by the OpenID Connect specification and can be seen in Figure 9:



```
{
   "iss":"accounts.google.com",
   "sub":"110502251158920147732",
   "azp":"825249835659-np4sqv7erhu1211s.apps.googleusercontent.com",
   "email":"prabath@wso2.com",
   "at_hash":"zf86vNulsLB8gFaqRwdzYg",
   "email_verified":true,
   "aud":"825249835659-np4sqv7erhu1211s.apps.googleusercontent.com",
   "hd":"wso2.com",
   "iat":1401908271,
   "exp":1401912171
}
```

Figure 9 - JWT with values inside the JSON object. [Source: Malalgoda, 2016]

The following bullet-points describes the tags that are relevant for this project in the ID token.

- *iss*: The issuer of the ID token formatted in HTTP URL.
- *nonce*: A unique random value that helps mitigate replay attacks. The AS must reject an ID token if multiple tokens carries identical nonce.
- *exp*: Expiration time of the token
- *auth_time*: Defines the time when a user was authenticated. If the user is already authenticated the AS will not require the user to authenticate again.

<u>Architecture</u>
Figure 10 illustrates a simplified flow of OpenID Connect.

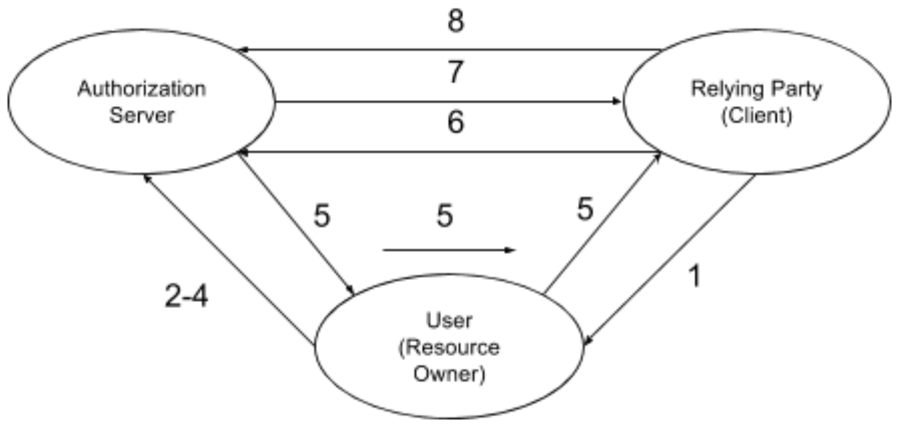


Figure 10 - A simplified illustration of the flow for OpenID Connect. [Source: Olesen, 2016a]

The following bullet-points are the steps for OpenID Connect:

1. The Client initiates an Authentication Request (AR) with the requested parameters.
2. This request is sent to the AS.
3. The User gets authenticated by the AS.
4. User provides its consent to the AS.
5. AS redirects the User to the client.
6. Client requests for an access token and an *ID token*.
7. Client retrieves the end-user’s identifier.
8. Client can request user’s information from the UserInfo (UI) endpoint of AS.

3.3.5 SAML

The SAML theory works with the same goal as OpenID Connect. SAML is quite similar to an extent, but also has a unique way of getting to the goal. This is done by using an identity provider in order to authenticate a user for an service provider. The perks of implementing a system in this fashion is that the service provider itself does not need to store that much user data about the user. This is all taken care of by the identity provider.
The way the SAML flow works is by having three different components, the requestor, service provider and the identity provider. The service provider is the actual place a requestor is login into and the identity provider is the third party which authenticates the requestor itself. The flow of this can be seen in Figure 11 below. (Olesen, 2016b)

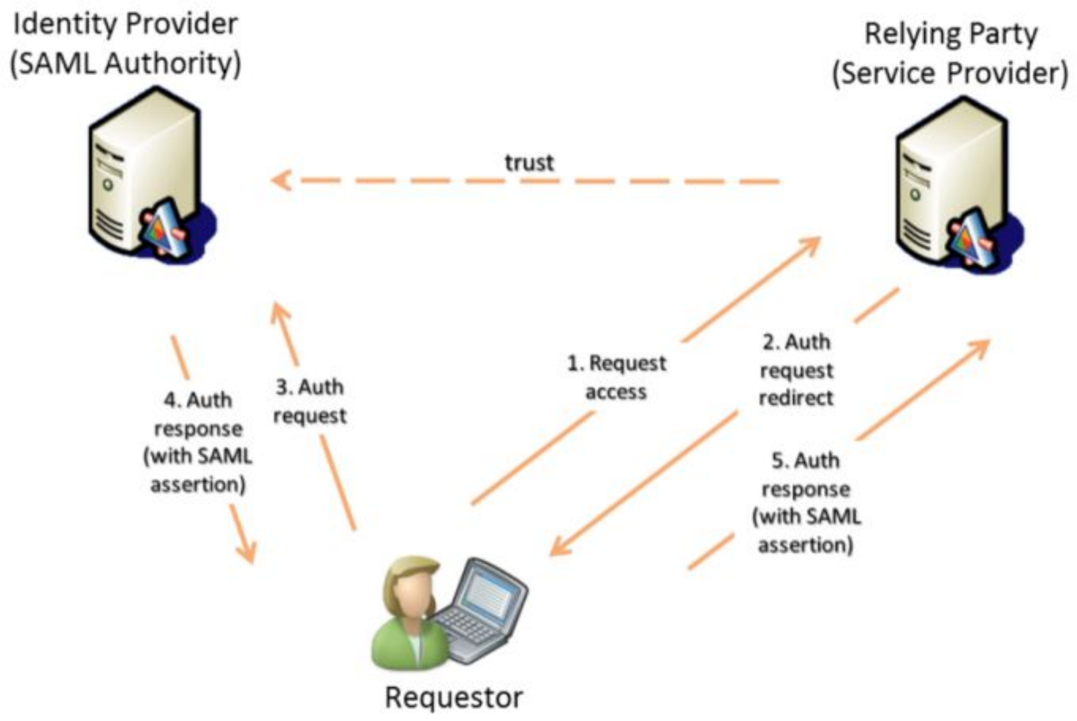


Figure 11 - SAML Workflow Example [Source: Olesen, 2016b]

The SAML workflow seen in Figure 11 can be boiled down to five point:

1. The Requestor requesting access.
2. The Service Provider provides an authorization redirection.
3. The Requestor contacts the identity provider with the authorization request.
4. The Identity Provider provides a SAML assertion in the form of authorization.
5. The Requestor relays that authorization to the service provider.

3.3.6 Secure Data Storing

For the secure storing of data, encryption and hashing are key. These theories are a used for the distortion of data in order to make it harder decypher by unwanted entities. The two theories have the same basic goal but are achieving it in different ways. The main difference between the two is that using a hashing function is a one way function and can not be deciphered like an encryption algorithm.

3.3.6.1 Encryption

As briefly mentioned, the encryption method compared to the hashing method can be reversed. The point of encryption in storing data is to be able to scramble data in a way where it can only be accessed by a party which is meant to be able to access it. This can be done with the help of an encryption key, which is the unique part of the algorithm from usage to usage. An encryption key can be used in two different ways, as symmetric key and an asymmetric key. Even though encryption can be used in various ways the main goal stays the same, protecting the confidentiality of the plaintext.

Symmetric Key Encryption

The first type of encryption is the symmetric key encryption, which is commonly found in encryptions as AES and 3DES. This encryption method uses the same encryption key for both the encryption as the decryption. The way this works is by having the plaintext run through an encryption algorithm which has the encryption key. When this has been done, a ciphertext is created. This ciphertext is unique to the plaintext and has to be run through the reverse algorithm to be decrypted. Since the encryption type is symmetric the decryption algorithm will use the same encryption key to decrypt the ciphertext. This flow can be seen in Figure 12. (Ayushi, 2010)

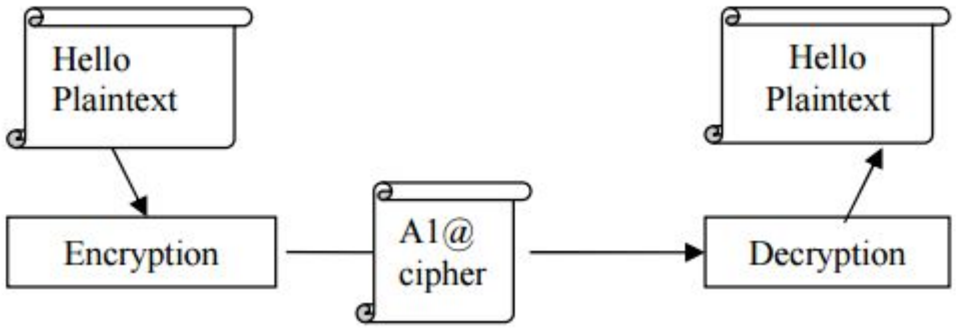


Figure 12 - Symmetric Key Encryption [Source: Ayushi, 2010]

Asymmetric Key Encryption

The other type of encryption is the asymmetric key encryption, which can be found in protocols such as SSL and OpenPGP. This encryption method uses two different keys, one for the encryption and one for the decryption. This is done by having a private- and public key, where one is used for encryption and the other for decryption. The private key is a secret which is only known by the holder of it and the public key is generally not kept as a secret. These two can be interchangeable depending on the purpose of the system the encryption is implemented in. One of these examples could be a client with the task of encrypting data which should only be available for one other client to access. This is done by encrypting the sent plaintext with the recipient's public key and at a later point decrypted back to plaintext by the recipient with their private key. This usage of the asymmetric encryption is done to keep the confidentiality and integrity of the sent data or message. An example of this can be seen in Figure 13. (Bellare, 2000)

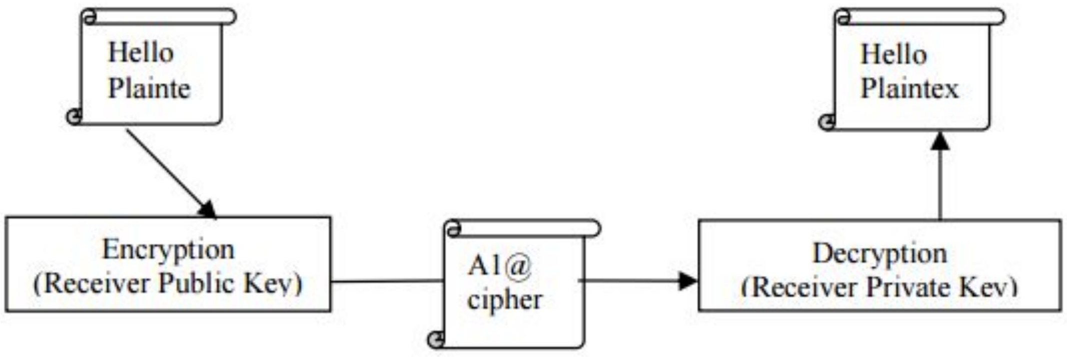


Figure 13 - Asymmetric Key Encryption [Source: Ayushi, 2010]

The asymmetric encryption can also be used in order to authenticate that the encrypted message or data came from the legitimate client. This is done by using the encryption in the reverse order than previously explained. This is done by a client encrypting using their private key, and everybody with the corresponding public key will be able to decrypt the message and also be able to confirm that the plaintext was encrypted with the private key of the client encrypting. (Bellare, 2000)

3.3.6.2 Hashing

The second way to store data is by hashing with functions as SHA and MD5. As previously explained, hashing is a one way function and can not be undone. This means that when a hash function is used it creates an irreversible unique hash digest. Since the process is irreversible the uses of hashing are quite different than a regular cryptographic function. One of the uses are in password storing and comparing. Since the hash is a one way function and therefore cannot be used in cases where the plaintext needs to be decrypted, but it can be used in cases where the plaintext needs to be compared. This works due to the hash digest always being the same for the same plaintext. Therefore a hash function can be used and the digest saved for comparison of next time when a system would need to check if the original plaintext was the same. There are multiple reasons to use hashing instead of encryption in the case of password storing would be a good idea, but the main reason would be security and retaining the integrity. This is due to a system not needing to store a password in their system, but merely a hash digest which cannot be decrypted. One of the small drawbacks of using a hash functions is that the storage needed could be larger, due to the hash digest always being the same size. This means that a password with only 3 characters would fill just as much as a 32 character one. The size of the hash digest is determined by the hashing function used and is generally increased with the complexity and security of the function. An example of hash functions and their change from digest to digest can be seen in Figure 14. (Silva, 2003)

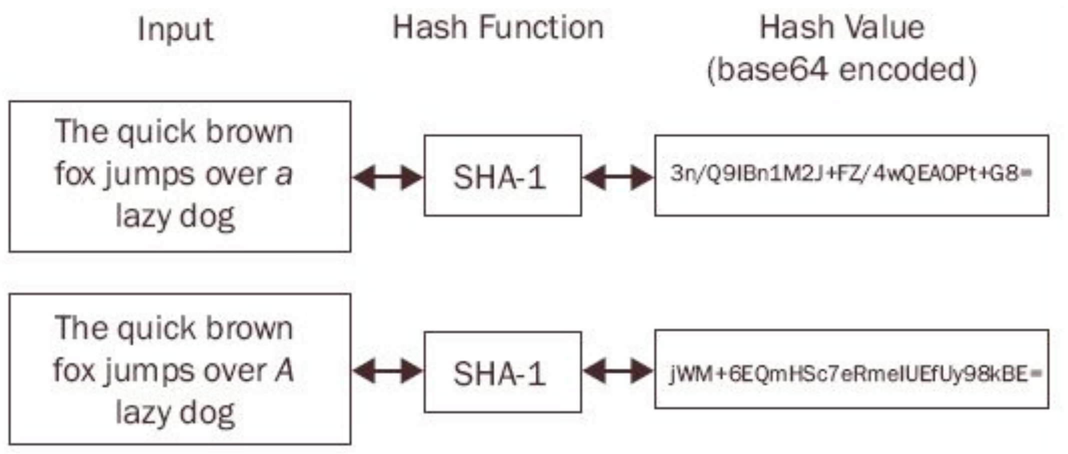

Input
Hash Function
Hash Value (base64 encoded)
The quick brown fox jumps over a lazy dog
SHA-1
3n/Q9IBn1M2J+FZ/4wQEAOPt+G8=
The quick brown fox jumps over A lazy dog
SHA-1
jWM+6EQmHSc7eRmeIUEfUy98kBE=

Figure 14 - Hash Function Example [Source: Paradigm 2017]

3.3.7 Password Theory

Managing passwords deals with defining, implementing, and maintaining password policies within a company, and if made effectively they can reduces the risk of compromising authentication system. Enterprises need to protect the CIA triad of passwords so any authorized users, and no unauthorized users, can authenticate the service providers for an enterprise. Especially confidentiality of passwords is difficult due to the security controls and the password's requirements themselves. A requirement could be the password's minimal length and it must be complex which makes it less likely for hackers to guess them, but as explained in the chapter 1 Introduction, the more complex passwords are, the more difficult it is to remember which might lead to be stored insecurely such as posters. (Scarfone, 2009)

However, it is important to understand the core principles of how passwords works and even better, how to create strong and complex passwords.

National Institute of Standards and Technologies (NIST) has defined requirements for passwords and passwords managers. Their definition of a password is a secret a claimant uses to authenticate its identity. The secret is usually a range of characters in form of a *string* containing both numbers and special characters, but can also be something the claimant physically possesses such as an identification-card, or something the claimant *is* like fingerprint or iris-pattern. Using the password together with a user-identifier like an email address or user name, is a form of identification and authentication and is often used to protect confidential information. (Scarfone, 2009)

Using multiple forms of authentication can increase the security of the system. Single-factor authentication only uses one factor such as, a character-string or a fingerprint, while multi-factor authentication uses multiple factors. Increasing the factors could make it more difficult for hackers, since the hacker is required to steal for instance one's identity-card and a password. However, increasing the factors might make the system more complex and can be overwhelming for a user since it needs to remember the ID-card and the password etc., hence it is not always beneficial for a system to provide multi-factor authentication (Scarfone, 2009)

Passwords arrive in different forms such as the Personal Identification Number (PIN). The length of a PIN is usually four to six digits, thus consuming less time than other passwords to enter meaning the asset, which the password is protecting, is usually not critical, and are rarely used to authenticate users.
Passphrases are another form of passwords and are, opposite of PINs, relatively long, which contain characters with letters, digits or special characters and a phrase. An example of a passphrase is *hellowordIam#1*. Passphrases might be easier to remember compared to remembering a series of arbitrary characters such as: *h91h1f@FH#q*, but they might also be too easy-to-guess and predictable which eliminates its advantage. An example can be *iliketrains* which attackers might have easier to guess compared to a passphrase like *1liK3Tr@ins*. Thus, the strength of a password does not depend on its length alone. (Scarfone, 2009)

To sum up the above text, the following bullet points are the best-practices from NIST for passwords, which were used later in 4.5 Requirement Specification:

- The length of the passwords does not directly determine its strength.
- Passphrases are an excellent way to create more complex passwords, but can be easy to guess despise their potentially long length.
- Multi-factor authentication is recommended if the assets is of confidential-critical.

3.3.7.1 Mitigating Threats Against Passwords

Along with NIST's definitions of passwords they also provided ways to mitigate the threats against passwords so they are kept confidential (Scarfone, 2009). These mitigations were helpful when defining the requirement specifications for this project, and are listed below:

- Preventing password capturing
- Preventing password attacks

- Effectively improving strength of passwords
- Effectively generating passwords

3.3.7.1.1 Preventing Password Capturing

Password capturing is when a hacker gain access to passwords from storage (remote or local database etc.), transmission when the password is transmitted to and from an endpoint, or from a user's knowledge. The following points are ways to mitigate password capturing:

Storage
Normally when passwords are used for authentication, OS and applications store their passwords on hosts. The passwords should be stored securely otherwise an attacker could physically gain access to the host, where the passwords are stored and steal them. An example would be a malicious employee who gained unauthorized access to its colleague and steal the confidential passwords. Therefore passwords should not be stored without additional security controls to protect them. The following security controls are examples to protect stored passwords: (Scarfone, 2009)

- Encrypt files containing passwords. This could be done by the OS, application or a password manager.
- Restrict employees' access to files containing confidential passwords.
- Store the hash-digest of the password rather than the password itself (further explained in section 3.3.8.2 Hashing)

However, just using encryption or hashing is not satisfactory. In US, federal agencies must store passwords using Federal Information Processing Standards (FIPS)-approved cryptographic algorithms standards. However, many systems requiring authentication support cryptographic algorithms that are either no longer approved by FIPS, for instance Data Encryption Standard (DES), or were never accepted (e.g., MD5, RC2, RC4). Therefore, enterprises should be aware of how well their passwords and hash-digits are protected.

Transmission
Authentication-systems that requires access to the network often transmit passwords or hash digests over the an internal or external network. This method is great for distributed systems where applications are either stored or needs to connect to a server, however, attackers could sniff the passwords by listening to the network, either passively via eavesdropping or actively with man-in-the-middle attack. The attackers sniff usernames and passwords that are transmitted unencrypted by protocols such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP).

The following are ways to mitigate these attacks:

- Encrypt communication channel using e.g. Transport Layer Security (TLS) to enable for instance Hypertext Transfer Protocol Secure (HTTPS).

- Create a tunnel for the communication via Virtual Private Network (VPN)
- Prevent transmitting of passwords in plaintext.

Knowledge of a user
Attackers can gain access to user's passwords via social engineering. For example, an IT criminal could pretend he/she is a help-desk employee, contact a user and ask for its password to assist the IT criminal to troubleshoot a problem. Social engineering can be in form of phishing emails, which redirects a user to malicious websites, and therefore steal the user's credentials. Mitigation such attacks requires user's awareness and signs of threats and how users should handle them. Another problem could be employees revealing their passwords to a malicious insider that could be a former employee now working at the competitor, which is interested in sharing the credentials to unauthorized parties.

3.3.7.1.2 Preventing Password Attacks

Attackers can determine weak passwords from password hashes through two techniques: guessing and cracking. For guessing, the attacker repeatedly attempts to authenticate on using common passwords such as 1234, or most popular passwords that people are using. Keeper Security, a password manager, revealed the most common passwords from 2016 from more than ten million users, which attackers could use to do dictionary-attacks where the attacker tries to authenticate with a list of passwords (Guccione, 2017). More advanced attacker would use brute-force attacks which tries any possible combinations of passwords in order to authenticate, but this technique takes longer time compared to dictionary-attacks, due to all the possible combinations of passwords.

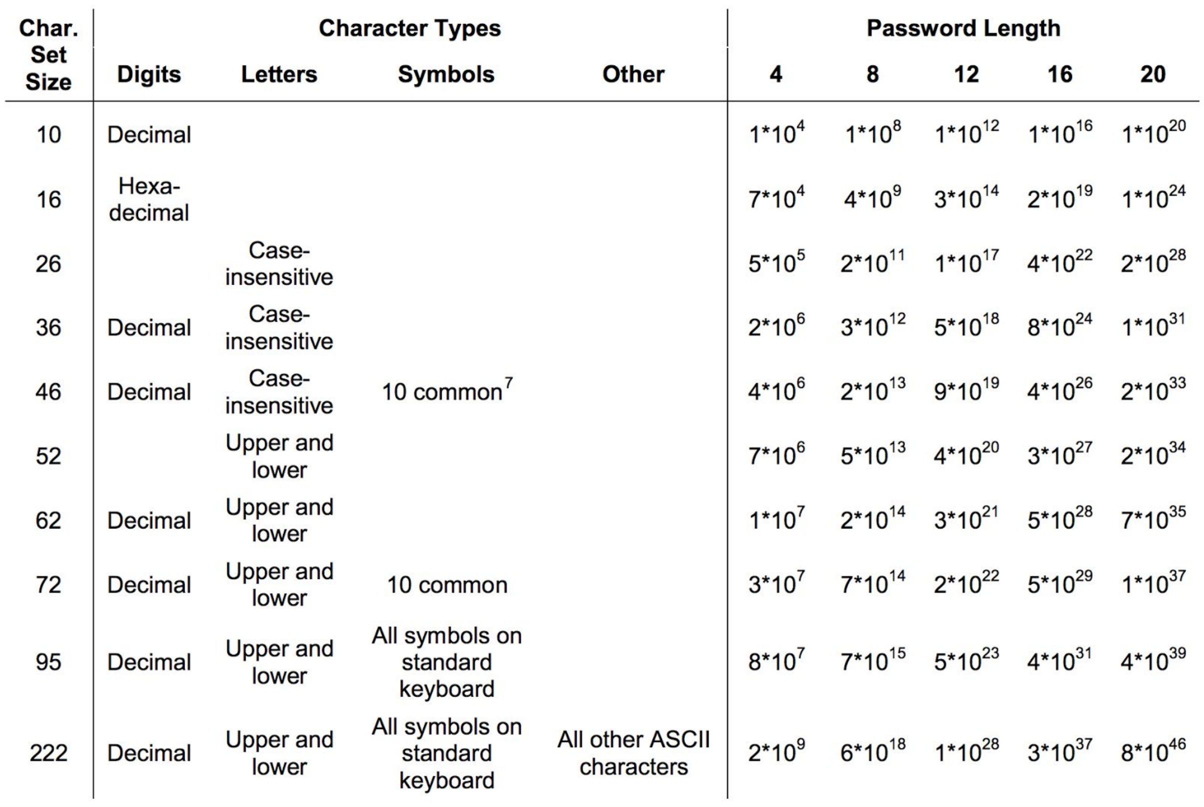
The following are security controls to prevent password attacks:
- Limit the amount of password-attempts.
- Delay the user if the password-attempts have exceeded the limit. This can be done with a delay that is fixed or exponential.
- Block and lock the account if too many failed attempts and delays has occurred.
- Notify the user that failed attempts has been occurred by email etc.

Cracking means that an attacker tries to recover cryptographic hash-digits by using various analytical methods to identify a plaintext string that might produce the digest. Weak hash-functions might produce the same hash-digest from different plaintext strings, which is not secure. In order to prevent this the hash-functions must implement salting into its algorithm. Salting is a technique to add a pseudorandom variable to the password and then generate the digest, which decreases the likelihood of generating a unique digest. Therefore, a way to mitigate cracking is:
- Use hash-functions that are FIPS-approved with salting.
- Ensure the user is using a strong password to create a strong hash-digest.

3.3.7.1.3 Effectively Improving Strength of Passwords

Strong passwords helps mitigate against guessing and cracking passwords. The strength of passwords is determined by the length and its complexity of passwords, where the complexity is determined by the unpredictability of its characters. These requirements are often part of password-policies which could be used by enforcing them in an enterprise. An example of a password-policy would contain, that passwords must have at least one uppercase-letter, at least three numeric digits and at least one special character.

Figure 15 illustrates how the effect of a password's length affects the number of possible combinations. For a simple PIN-code of four digits there exist $1x10^4$ (10.000) possible combinations, and doubling its length the possible combinations also doubles ($1x10^8$). Adding passwords with character-specific characters along with digits and with special characters, then possible combinations increases dramatically, which can be seen with a password with a length of four having $2x10^9$ possible combinations.



Char. Set Size	Character Types				Password Length				
	Digits	Letters	Symbols	Other	4	8	12	16	20
10	Decimal				$1*10^4$	$1*10^8$	$1*10^{12}$	$1*10^{16}$	$1*10^{20}$
16	Hexa-decimal				$7*10^4$	$4*10^9$	$3*10^{14}$	$2*10^{19}$	$1*10^{24}$
26		Case-insensitive			$5*10^5$	$2*10^{11}$	$1*10^{17}$	$4*10^{22}$	$2*10^{28}$
36	Decimal	Case-insensitive			$2*10^6$	$3*10^{12}$	$5*10^{18}$	$8*10^{24}$	$1*10^{31}$
46	Decimal	Case-insensitive	10 common[7]		$4*10^6$	$2*10^{13}$	$9*10^{19}$	$4*10^{26}$	$2*10^{33}$
52		Upper and lower			$7*10^6$	$5*10^{13}$	$4*10^{20}$	$3*10^{27}$	$2*10^{34}$
62	Decimal	Upper and lower			$1*10^7$	$2*10^{14}$	$3*10^{21}$	$5*10^{28}$	$7*10^{35}$
72	Decimal	Upper and lower	10 common		$3*10^7$	$7*10^{14}$	$2*10^{22}$	$5*10^{29}$	$1*10^{37}$
95	Decimal	Upper and lower	All symbols on standard keyboard		$8*10^7$	$7*10^{15}$	$5*10^{23}$	$4*10^{31}$	$4*10^{39}$
222	Decimal	Upper and lower	All symbols on standard keyboard	All other ASCII characters	$2*10^9$	$6*10^{18}$	$1*10^{28}$	$3*10^{37}$	$8*10^{46}$

Figure 15 - Table from NIST illustrating the relation between length of a password and possible combinations. [Source: Scarfone, 2009a]

However, as mentioned earlier, it is important to keep in mind that Figure 15 only illustrates the total number of possible combinations of a password, but that alone does not define the strength of a password. Thus, it is not enough alone to have a long password.

Ways of mitigating weak passwords are:

- Avoid easy-to-guess passphrases like *iliketrainsverymuch*, even though its length satisfies the minimal length. Try replacing some of the letters with special characters so the passphrase ends up being: *1LiK3Tr@in$_v3rYMuCh.*
- Define password policies in enterprises, enforce them and educate staff to create immensely strong passwords.
- Counter authentication-mechanisms that could truncate the all characters in a secret key and passwords before hashing it.

3.3.7.1.4 Effectively Generating Passwords

The following are the two ways to generate passwords, pseudo random-generated passwords or user-selected passwords. Automatically generated passwords are often very strong since the administrator can define the requirement of the passwords. However, due to their complexity they can be difficult to remember for a regular user. Conversely, user-generated passwords are easier to remember but tend to lack in strength. For passwords where memorizing the passwords are not required, a password-generator which automatically generates passwords, could be feasible. These generated passwords should potentially be as strong as possible, using very complex combinations of upper- and lowercase characters, numeric values and special characters since they are not intended to be memorized. In this way, an application can create very strong passwords.

As explained earlier, the second way to generate passwords are user-generated. When users generates a new password they should be made provided with password-requirements, including restrictions on password combinations. An application might require passwords to be between eight and twenty characters long, require a combination of upper- and lowercase letters and digits. Providing a clear list of restrictions aid users to create strong passwords which meets the requirements, thus avoiding having the passwords rejected, which would be insecure.

The following are sum up from the above section:

- Use pseudo random-generated passwords for passwords not intended to be memorized.
- Create clear criteria for passwords to avoid user's passwords not being approved.

3.4 Candidate Technologies and Solutions

In the Candidate Technologies and Solution section the existing relevant technologies and solutions will be described in detail. The goal for this section is to gather information and elaborate on the technologies and solutions, in order for them to be used in the creation of the ID-Connect system.

3.4.1 WSO2

WSO2 is a framework created for the purpose of identity management, specifically for enterprise use. The way the identity management is fulfilled is by having a platform connected to various applications in each respective website which is chosen as an authentication-point. The WSO2 framework can be split into two key sections, the Identity Server (IdS) and App Manager.

3.4.1.1 Identity Server

Overall WSO2 IdS is a framework to manage and keep track of identities through one platform. The IdS section is used for tasks such as Identity Federation and SSO, Identity Provisioning, Access Control and Analytics (Example in Figure 16). In relation to a system which would be in term creating a platform to use as the entry point to multiple other sites, these tasks would be a great help in insuring Access Control and Identity Securing. (WSO2, 2017a)

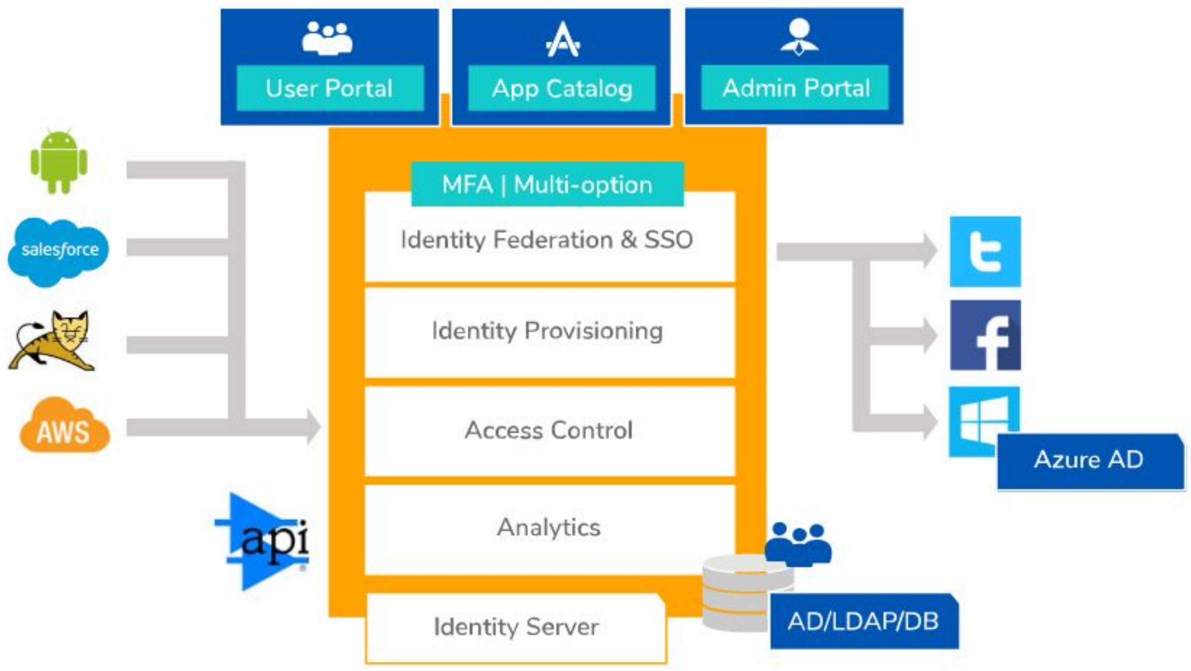


Figure 16 - IdS Framework [Source: WSO2, 2017b]

The way the WSO2 IdS framework works is by having an API connection from a platform such as an Android device or a web server to the framework. This connection is then managed in terms of Access Control and Identity, before being connected to the end point app on the accessed site or service. This is also the reasoning behind the Identity Federation, SSO, Identity Provisioning and Access Control are the most key elements of the framework.

- **Identity Federation**
  Linking a user's identity across multiple identity management systems.
- **SSO** is a technology which allows a user's identity to be used as a login method to multiple platforms. This in term would make the WSO2 framework an IdP or IdP manager.
- **Identity Provisioning**
  The ability to add or remove users authorization to application.
- **Access Control** is the part of the WSO2 framework which helps determine which users are allowed to do what. This means that the framework can easily use Role-Based or Identity- Based Access Control.

3.4.1.2 App Manager

The App Manager is second part of the WSO2 framework and is in charge of both the App Publication and Store, which are used for the connection from the framework to the specific service that the user wants to access. The App Manager can be split into three parts; The End User Apps, Runtime and The End Point (Seen in Figure 17). (WSO2, 2017c)

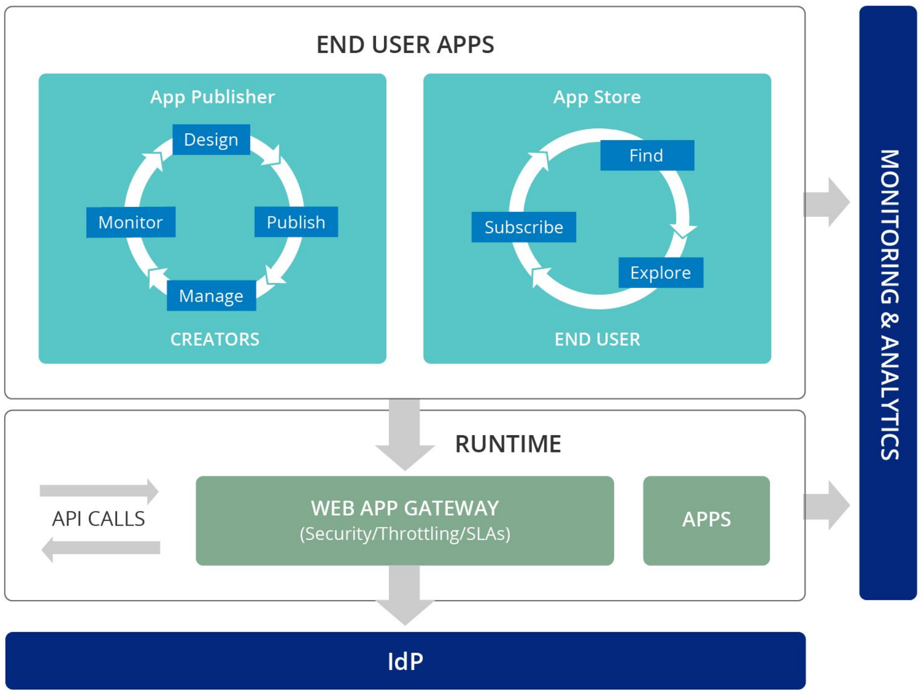


Figure 17 - App Manager Framework [Source: WSO2, 2017d]

These three parts can be as such:

- **End User Apps** is the first part of the App Manager and is in charge of the Publishing and the Store of Apps. The Publishing is created to easily allow developers to create a connection to individual sites or services from WSO2. These apps are then stored into the App Store which the user can explore and subscribe to, in order to connect to individual sites or services.
- **Runtime** is the actual connection between the given app to the app specific site or service. This happens through a series of API calls.
- **The End Point** is the actual site or service where the app is connected to.

3.4.2 Social Identity Providers

A lot of the existing IdPs are social media companies, such as, Google, Twitter, Facebook, etc. These IdPs can also be a great asset for the individual company, in example for promotional use.
These IdPs work in basically the same way as the standard IdP structure which was previously explained in section 3.3.2 Identity Provider, and are mainly using OAuth, OpenID Connect or a combination. The way these IdPs are set up is by users having registered some of their user data with the service that the social media service provides. Part of this data is thereafter relayed to the party using the social media IdP. The data which is relayed, depends on the individual IdP. This means for example that the data Facebook provides is not always the same as what Google. This is determined by the party using the IdP and limited by the limitations of what data the IdP has stored and allows access to.

3.4.2.1 Google

The first Social IdP is the Google IdP. The Google IdP is a very popular IdP platform which can be used on most SSO login friendly sites. This IdP uses OAuth 2.0 as its framework, which was previously explained in section 3.3.3.2 OAuth 2.0. Google like many other IdP can work with quite minimal information, since the user having an account on an IdP site does not need to fill in much information. For the OAuth 2.0 flow the Google IdP only gives out four pieces of data. These are the ID, Name, Image URL and Email of the user. This gives the third party using the IdP some freedom given the parameters provided by Google. Even though these parameters are given to the third party for usage, there is gathered much more data by Google which are not disclosed. (Google, 2017b) (Google, 2017c)

3.4.2.2 Twitter

Another very popular IdP that is used is Twitter. Twitter in an interesting IdP platform in a way where the platform gathers a vast amount of user data but that data does not need to be personal data. This is due to the user being able to hide behind an alias on the Twitter platform. This provides some privacy protection but a the cost of some personalization perks. Another interesting fact about Twitter is that they are actually only using OAuth 1.0 instead of the newer and better OAuth 2.0.
The way Twitter works is by utilizing the OAuth 1.0 workflow. This flow can be split up into three parts. (Twitter, 2017)

Part One: Request token
The first part of the flow is the fetching of a request token from the IdP, which in this case is Twitter. What happens is that the client calls the Twitter IdP which returns a token and a token secret. This can be seen in Figure 18.

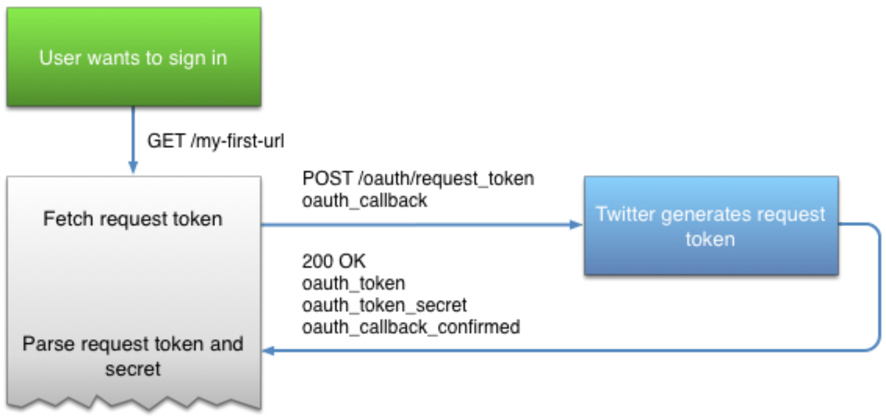


Figure 18 - Twitter OAuth 1.0 Request Token [Source: Twitter, 2017]

Part Two: Redirection
After the system can be introduced to the token which is in possession of the client, the IdP checks it and redirects the client with an OAuth verifier. The verifiers job is to check if the client is authenticated and approved by the IdP. The token and verifier is relayed by the client to the service provider. This flow can be seen in Figure 19.

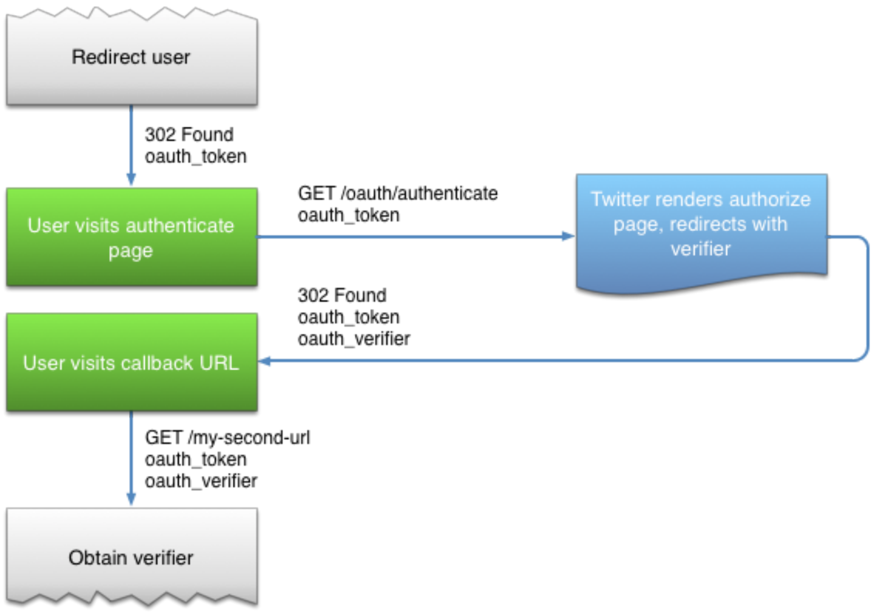


Figure 19 - Twitter Redirection [Source: Twitter, 2017]

Part Three: Access Token

The last part of the workflow is actually creating the access token used for authorization. This is done by the client sending the verifier to the IdP which then upgrades the request token into an access token. This token is the final token which is used for the actual access to the service provider. The token in Twitter's case consists of a token and a secret, which is mandatory and then the user's "user id" and "screen name". These are the chosen user data parameters which are given to the service provider by the Twitter IdP. This flow can be seen in Figure 20.

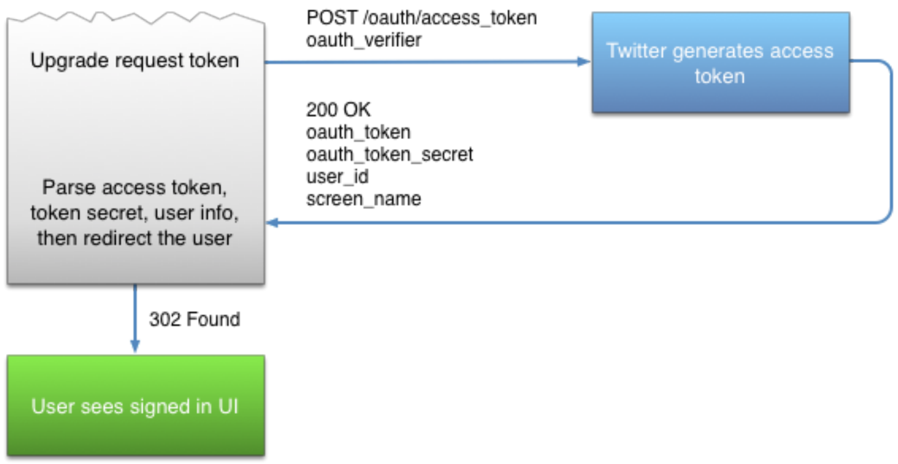


Figure 20 - Twitter Redirection [Source: Twitter, 2017]

3.4.2.3 Facebook

The Facebook IdP is likely to be the most flexible IdP in terms of how much data can be given by the user. This is both due to the availability of the shared parameters and the vast amount of data which Facebook gatherers about the user. The gathered user data therefore give Facebook the option to share this data with third party platforms to create a more user specific experience.
Some of the most common and useful shared data provided by the Facebook IdP is the general user information such as email, name, age, gender, location and profile picture. These informations can also be very useful in managing and identifying the user. Besides these general user informations there are a lot of other data that has been gathered and are available to the third party using the Facebook IdP. Some of these options are posted pictures/videos, friends, posts, geo locations and even Facebook inbox. These extended options of user data shared does provide the third party application with a vast amount of options.
The Facebook IdP is also built upon the OAuth 2.0 platform which means that is has the same workflow as the Google IdP, with the exception of providing different user parameters to the service provider. This flow was previously covered in section 3.3.3.2 OAuth 2.0
(Facebook, 2017)

3.4.3 Additional tools

The following tools were used to develop the proof of concept. Most of the tools were chosen based on the requirements specifications and previous knowledge of the group members.

3.4.3.1 JUnit

JUnit was used to unit test the Java-part of the ID-Connect system. This tool is a unit testing framework for developed for Java, and that allows frameworks for testing the UI in a Java program. It provides APIs which can simulate user interactions. The framework is deeply integrated with the thread-pooling system Java so the developers do not need to worry about life-cycles of the application. (JUnit, 2017)

4 Analysis

In the previous chapters, several areas have been investigated such as the currently existing password managers and how they can be used SMEs, relevant theories and solution, such as encryption and hashing mechanisms, to investigate the mechanisms of how secret elements should be protected. Different manners of how to provide identity for users has been explained, such as IdP and how service providers uses them.

Based on the obtained knowledge from these topics a deep analysis of what the best-practises are when implementing authentication mechanisms in a system. A stakeholder analysis will be presented which explains the different stakeholders interests for this project's solution. Furthermore, the results from a questionnaire will be presented, and further elaborated on. At the end of this chapter the requirement specifications from the research and analysis will be defined.

4.1 Stakeholder Analysis

The Stakeholder Analysis section will focus on the stakeholder which are connected to the ID-Connect system. These stakeholders will have each of their key interests presented together with the drawbacks.

4.1.1 Enterprise (SME)

The SMEs are the primarily stakeholder for this project, since the project's solutions is aimed for SMEs. Their interest for the system is to easily and securely manage their employees passwords, and being able to effectively unenroll an employee who is leaving the company. This would be possible from one endpoint where the person, who is responsible for IT security within the company, can authenticate with his/her master password, and enroll or unenroll employees. With this approach the SMEs could better focus on their businesses rather than leaving the responsibility of managing employee's own passwords.

The following are the key interests for SMEs for this project:

- Remembering only one master password
- Better and easier management of passwords of employees
- One place to manage authentication of employees
- Minimize the risk of employees creating weak passwords for service providers
- Improving security controls of the enterprise
- Easier to do business with bigger cooperative companies due to increased security.

The drawbacks of implementing such solution are listed below:

- Cost of migration to the new system
- Cloud infrastructure for data storage.
- Relying on third-party to securely store the passwords.
- The master password might not be strong or complex after all.
- Hackers only needs to gain access to the master password.

4.1.2 Enterprise Employees

The SMEs employees are the ones using the services provided or supported by the SMEs, and naturally they must register to the services if they use the services for the first time. As stated in chapter 1 Introduction remembering multiple passwords can be a difficult, hence many tend to use the same password for multiple services. By using this project's system, the employees do not have to remember all the different passwords but only one master password, used to authenticate to the other services.

Thus, the key interests for enterprise employees are:

- Eliminating the need of remembering many strong and complex passwords.
- Easier registration and authentication to service providers through the system.
- Eliminating the risk of getting personal elements hacked.

The following are the drawbacks of the system:

- The employees most go through the system in order to authenticate to the service provides, which can take time to learn.

4.1.3 Identity Providers

The second last stakeholder is the identity providers. This stakeholder can easily be seen as an asset to the system, since the system heavily relies on the use of service providers for the login process. Even though the identity providers are a big stakeholder in the system they do not have to many key interests in the system. This also means that the drawbacks for the identity providers needs to be kept to a minimum. What the system means for the identity providers is that they will increase in usage and get more users, but these users that are gained from the ID-Connect system might not have that much interesting data for the identity providers.

This leads to the following key interests for the identity provider:

- Increase in usage of the identity provider.
- Increase in users in the identity provider.

The following drawbacks of the system for the identity provider:

- The users registered in the identity provider might not have any usable data for the identity provider.

4.1.4 Service Providers

The last stakeholder is the service provider, which is the actual end point where a user logs into. Just as the identity providers the service providers are also an important part of the system. Therefore it is important that the key interests of the system exceeds the drawbacks. The service provider is selv gets an easier and more secure way of login in which will both potentially increase usage and limit misuse of the service. Though just as with the identity providers, the users which are created might not have as much data on them as if they would with out using the ID-Connect system. Also as mentioned in the previous stakeholders, there is only a single point of entry which means that if a misuse would occur it would likely be on a larger scale by impacting multiple service providers.

The key interests of the service provider can therefore be boiled down to:

- Increased usage of the service.
- More security for logging in and therefore less change or misuse.
- More confident to do business with Enterprise do to extra security

The following drawback of the system for the identity provider.

- In the case of misuse the problem would likely be on a larger scale due to the single point of entry.

4.2 Survey

In order to investigate and validate the problem of managing passwords in SMEs a questionnaire was sent to a specified audience (see section 2.4 Questionnaire). Their answers would help better defining the requirement specifications for ID-Connect. In the original questionnaire, which can be found in chapter 10 Appendix under *Appendix A* along with the result in *Appendix B*, the questions were written in Danish, but are translated into English in this thesis.

4.2.1 Questions

The questions were divided into three categories, introduction, passwords and password managers in order to ease the analysis and not overwhelming the respondent with many questions at once.

Part 1 - Introduction
Here the questions were aimed to get a better understanding of who the respondent was and which type of enterprise the respondent owned or was a part of.

<u>Primary goal:</u>

Understanding who the respondent was and the its enterprise he/she is working in.

Secondary goal:
The defined audience did not include age and gender but it would be interesting to investigate how the young and elder generation responded to the questions, to investigate any tendencies.

Questions:

1. What is your gender?
2. What is your age?
3. What type of company do you have (IVS, IS, ApS or A/S)?
4. What is your company's occupation?
5. How many employees are there in the company?

Part 2 - Passwords

In this part the core purpose of this questionnaire came into place by asking the respondents about their management of passwords.

Goal:
Validating if users uses the same passwords for multiple services, and how and who manages the users passwords in an enterprise along with the reason behind these decisions.
Questions:

6. Generally how complex do you think your password is?
7. Do you use the same password to different services?
8. What is the purpose of using the same password?
9. Have your company made a budget for IT security?
10. Who is responsible for your IT security?
11. How do you manage yours and your employees passwords?
12. If your enterprise wanted to create an Office 365 account for your employee, how would the process be?
13. If an employee leaves, how do you remove the employee from the services?

Part 3 - Password managers

The final part were questions about password managers and the respondents knowledge about them, since a way to deal with memorizing strong passwords could be dealt with password managers as stated in chapter 1 Introduction.

Goal:
Investigating the respondents knowledge about the password managers.

Questions:

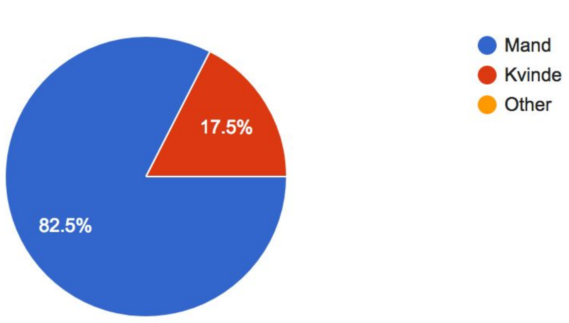
14. Do you know what a password manager is?
15. Which password manager are you using?

4.2.2 Analysis of Results

As mentioned in chapter 2 Methodology, the questionnaire were sent to two different Facebook groups for entrepreneurs. Despise the two Facebook groups having a large and active community of more than 30,000 members, the total amount of respondents were only 40, where the expected answers was between 200-300 people. Not meeting the expected responses could be due to the very dynamic community especially in the group "Iværksætter Netværk", resulting into the group members Facebook post being pushed down.

Part 1 - Introduction

The charts on Figure 21 clearly shows that males are the majorities who answered the questionnaire (82.5%), which is not surprising due to the high number of males in both Facebook groups. However, the question about age represented a much more diverse result. 18-25 year-olds where the largest group having 30% of the score, 36-45 year-olds counted for 25% and 26-25 and 46-55 year-olds had 22.5% each. The diversity in age range was good for the survey since age was not defined in the target group, and it would be interesting to see how the responders responded based on their ages.



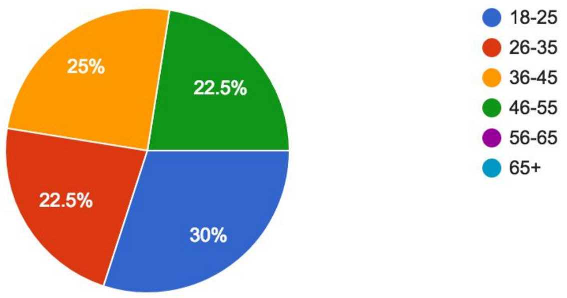


Figure 21 - Result for gender (left) and age (right).

The next questions were about which type of company the respondent belonged to. IVS and IS are normally considered small companies since IVS are called "enterprises for entrepreneurs" and start-up companies usually begin with an IVS and later become an ApS, hence in this report IVS and IS are considered small enterprises where ApS and A/S are considered medium and large companies (Virk, 2017). The charts on Figure 22 shows that ApS together with A/S stands for 62.5% (medium-sized companies) of the result, and the rest are small companies. Even though the medium-sized enterprises are majorities it is still balanced, which this rapport's group members interprets to be good due to the diversity of the survey.

Most of the companies in the survey are working with "software " (45%), where the second place is "other" (27.5%) and the third is "sales" (15%). The majorities of companies working with software is not surprising, since most Facebook posts from both groups are software-related.

Regarding the number of employees the result showed that 75% of all companies had 1-5 employees employed, which is not surprising since 85% of all companies in Denmark have less than five employees (Eurostar, 2017).

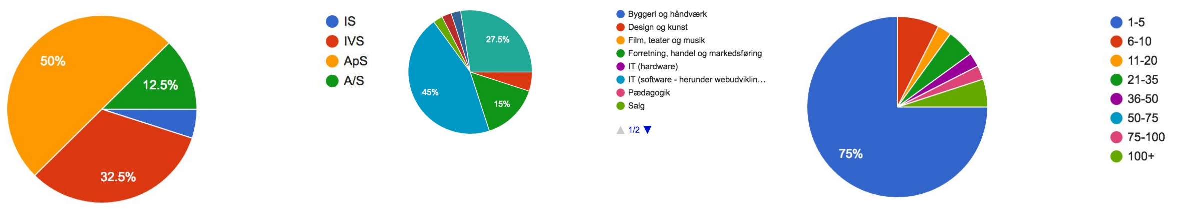


Figure 22 - Result for Type of company (left), Company's occupation (middle), and number of employees (right).

Part 2 - Passwords

Figure 23 illustrates how strong and complex people in general believe their passwords are, with a score from one to five, where five is "very strong". 45.5% answered "4", which represented "strong", where 22.5% answered three and 22.5% five, hence generally people believe their passwords are "strong". Even though the group members thought people would have weaker passwords before the results from the survey, they still believe the result is true, since the majorities of the companies are working with software. However, it is difficult to validate this truth, since asking the respondents to type their password in the survey might scare them off.

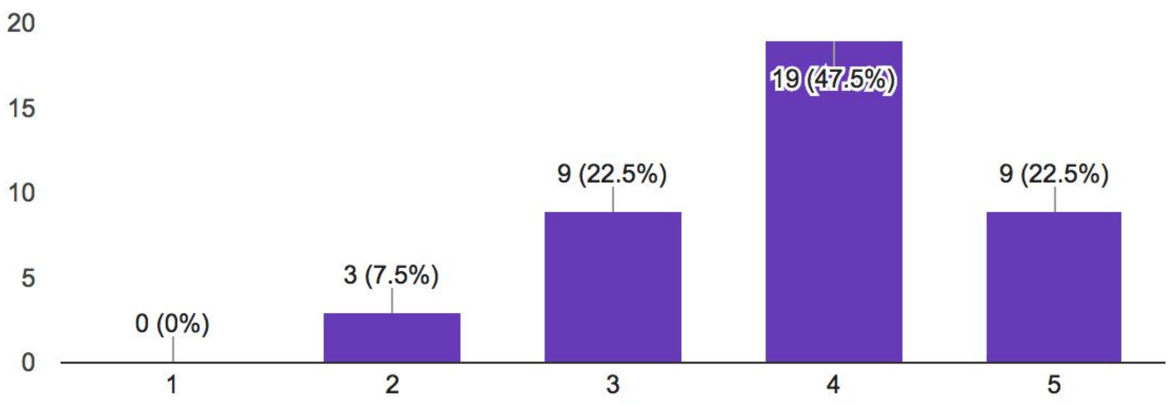


Figure 23 - Result that shows how strong people think their passwords are.

Surprisingly almost half of the respondents answered “yes” of using the same password for different services. This clearly validates the theory in the *Introduction* section that many people uses the same password for different services, but the group members did not expect 45% which was higher than expected. 47,5 % answered that the reason of using the same password was, that it is difficult to remember all the passwords. 35% answered that they never use the same password which makes sense, since 55% answered “No” to never using the same password twice. This can be seen in Figure 24 below.

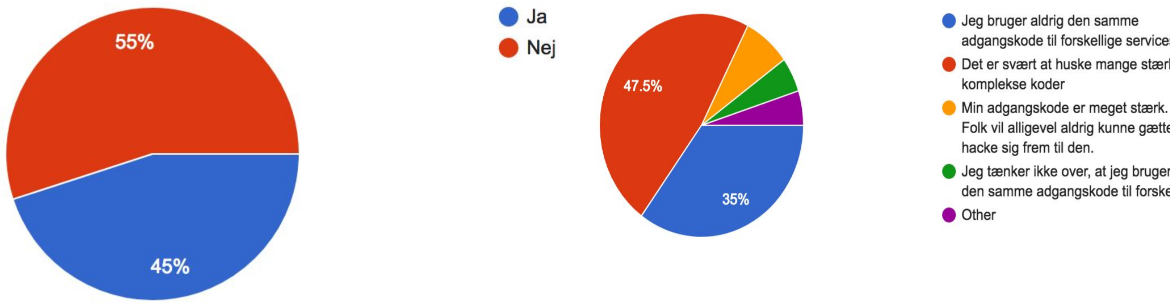


Figure 24 - Results from “Have you used the same password for different services?” (left) and “What is the reason of using the same password. (right)”.

90% of the respondent’s enterprises have not made a budget for IT security, which is not surprising at all since SMEs’ budgets are relatively low compared to larger companies. More than half of the respondents (57,5%) answered that head-managers of the companies are responsible for the company’s IT-security, and only 10 % used their IT-section for this task. Considering that most of the companies have 1-5 employees the SMEs might not have enough employees for an IT job.

The answers to where the employee's credentials are stored a different. The majority (32,5) answered mail conversations, where 27,5 % answered that it is the employee’s responsibility to store their own credentials. This can be troublesome for an enterprise since a former employee would still have access to their service providers and could potentially harm the company. Another vulnerability could be hackers attacking the employees who could potentially store their credentials irresponsibly and insecurely since they are in control in their credentials. This can be seen in Figure 25.

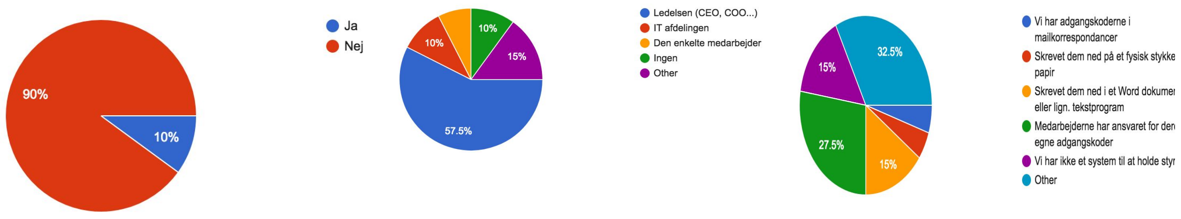


Figure 25 - Charts from question 9 - 11.

There were many different answers for question twelve. 30% answered that the person responsible for the IT-security enrolls the new employee and sends a link to the employee's email from where the employee registers itself. On second place the responsible person hands over the credentials to the employee on a physical piece of paper, which can be read by third-parties.

The next question was about how the employees unenroll themselves from the service providers. 56,5% answered that the IT-security responsible person unsubscribes the former employee manually, which means the person has to keep track of which services providers the former employee was registered to.
Surprisingly 28,2% of the respondents did not know how how the enterprise unsubscribed the employees. This can be a problem since former employees could still have access to the services like emails, documents and other assets. 15,4% answered that they used a software-tool, which would be great to know, but the question was only a bullet point. These questions are illustrated in Figure 26.

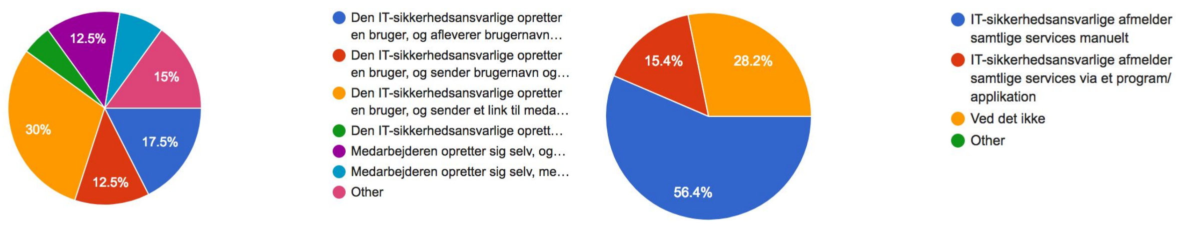


Figure 26 - Charts from question 12 - 13.

Part 3 - Password managers

As seen in Figure 27, 55% of the respondents answered "Yes", if they had heard of password managers. This was fewer than expected, since there exists many services requiring passwords, which needs to be managed.

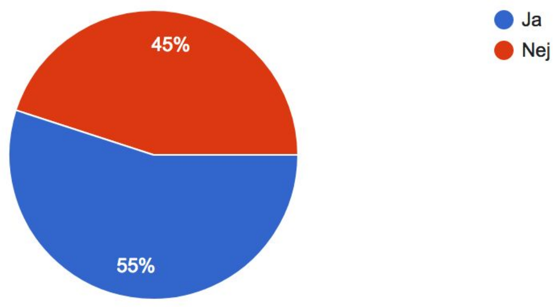


Figure 27 - Chart for the question "Have you heard of password managers?".

Figure 28 illustrates the currently existing password managers that the respondents are using. While 57,5% are not using a password manager, the rest are spread out in the different managers. "Safari Keychain" is the most used with 10% where 7,5% uses LastPass, 7,5% Dashlane and 7,5% Google Smart Lock. The 25% who answered "Other" have not provided with the name of the password manager, which would be interesting to have.

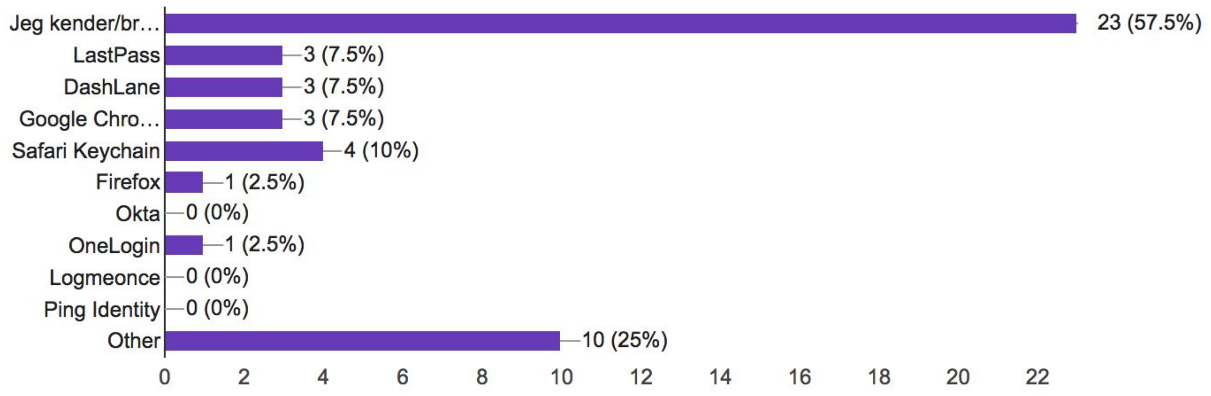


Figure 28 - Chart for answers to question 14.

4.2.3 Sub conclusion of survey

Summing up the results of this survey, it appears that many people uses the same password for multiple services, which was mentioned in chapter 1 Introduction, and the reason is the difficulty of remembering all the passwords. Password managers could solve this problem but surprisingly almost half of the respondents did not have any knowledge about password managers, and did not use them in their enterprises. Also unenrolling a former employee was a problem for the enterprises since the majority manually removed the employee from the system, which could be time-consuming if the former employee was registered in many services, and the registered services were not on a list before. These considerations will later be used for the requirements of the ID-Connect system.

4.3 Scenarios

The scenario section will create and elaborate on scenarios for the user interaction with the system. These scenarios will be used for the creation of sequence diagrams and are derived from the possible interactions with the system. The scenario section will include three different scenarios, User Creation, User Management and User Login.

4.3.1 Scenario 1 - User Creation

Alice is a new user of the system but has not created an user profile yet. Alice is also the first user who is going to be registered for her company and therefore her user will become the administrator.
Alice starts by opening up the system and clicks on the register button. Thereafter she is redirected to a page where she needs to fill in her credentials. These credentials are saved into the system database and here user is tied to her company. Alice can now enjoy her administrator user.

4.3.2 Scenario 2 - User Management

Bob is the administrator for a small IT firm and is registered as the administrator in the system. Bob's task for today includes removing Carl, a previous employee and giving Dylan access to the service provider Silvan.
Bob starts by login into the admin management system. The system can recognize Bob as an administrator and which company he works in. Bob is therefore greeted with a list of employees for his company. He thereafter starts by choosing the previous employee Carl, and revokes access for his account. The next step for Bob is to choose the user Dylan, and see all his permissions. Bob thereafter grants Dylan access to the service provider Silvan. Bob logs off and continues his work.

4.3.3 Scenario 3 - User Login

Dylan is an employee at Bob's small IT firm and needs to login to the service provider Silvan. Dylan can now do this since Bob has granted him access to do this. Dylan first starts by launching the system. He thereafter inputs his username and password. The username and password is checked and Dylan is guided into the system. The next screen prompts Dylan with the choice of service provider from the list of available once for him. Dylan finds Silvan and clicks on the login button for Slivan. He then is logged into Silvan. Dylan can now proceed with his work.

4.4 Sequence Diagram

The Sequence Diagram section of the Analysis chapter is derived from the previously mentioned scenarios and are created to further define the structure of the system. Together with defining the structure of the system, the sequence diagrams are used to help with the creation of the Requirement Specifications and the Traceability Matrix which validates these. The sequence diagrams will feature three sequence diagrams, User Creation, User Management and User Login. Each of these sequence diagrams will include a description, the preconditions, steps for the sequence diagram and the postconditions of the system.

4.4.1 Sequence Diagram 1 - User Creation

Sequence Diagram 1 consists of the user interaction of the system, where the goal is to create a user in the system. This can be seen in Figure 29 below.

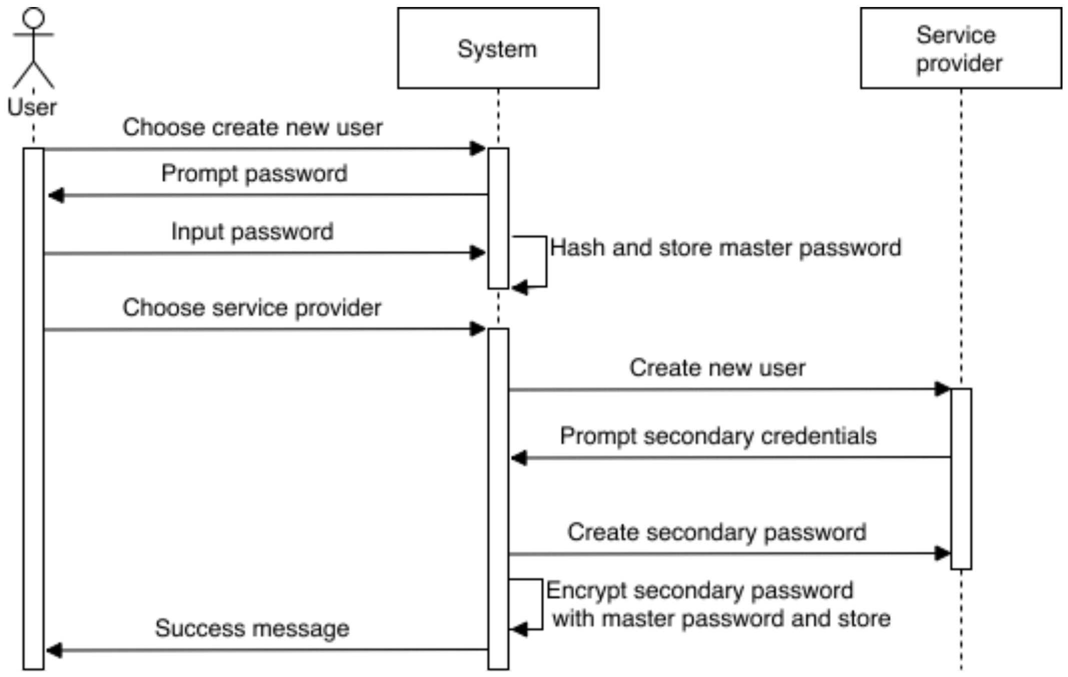

User
System
Service provider
Choose create new user
Prompt password
Input password
Hash and store master password
Choose service provider
Create new user
Prompt secondary credentials
Create secondary password
Encrypt secondary password with master password and store
Success message

Figure 29 - Sequence Diagram 1, User Creation

Preconditions:

- Actor must be authorized to use the system.
- Actor must have a reliable internet-connection.

Steps:

1. Actor chooses to create new user.
2. System prompts Actor to provide a new username and master password.
3. Actor inputs the requested credentials.

4. Provided master password is hashed and stored.
5. Actor chooses which service provider they want to login to.
6. System sends a request to the Service Provider to create a new user.
7. Service Provider prompts for a new secondary password.
8. System creates a random secondary password.
9. System encrypt and store the secondary password with user's master password.
10. System returns with an appropriate message.

Postconditions:

- The system has the user data stored with a hashed master password and an encrypted secondary password.
- The new user can now authenticate to the system and access its authorized service providers.

4.4.2 Sequence Diagram 2 - User Management

Sequence Diagram 2 consists of the user-interaction of the system, where the goal is to authorize a new user to use a service provider. This can be seen in Figure 30 below.

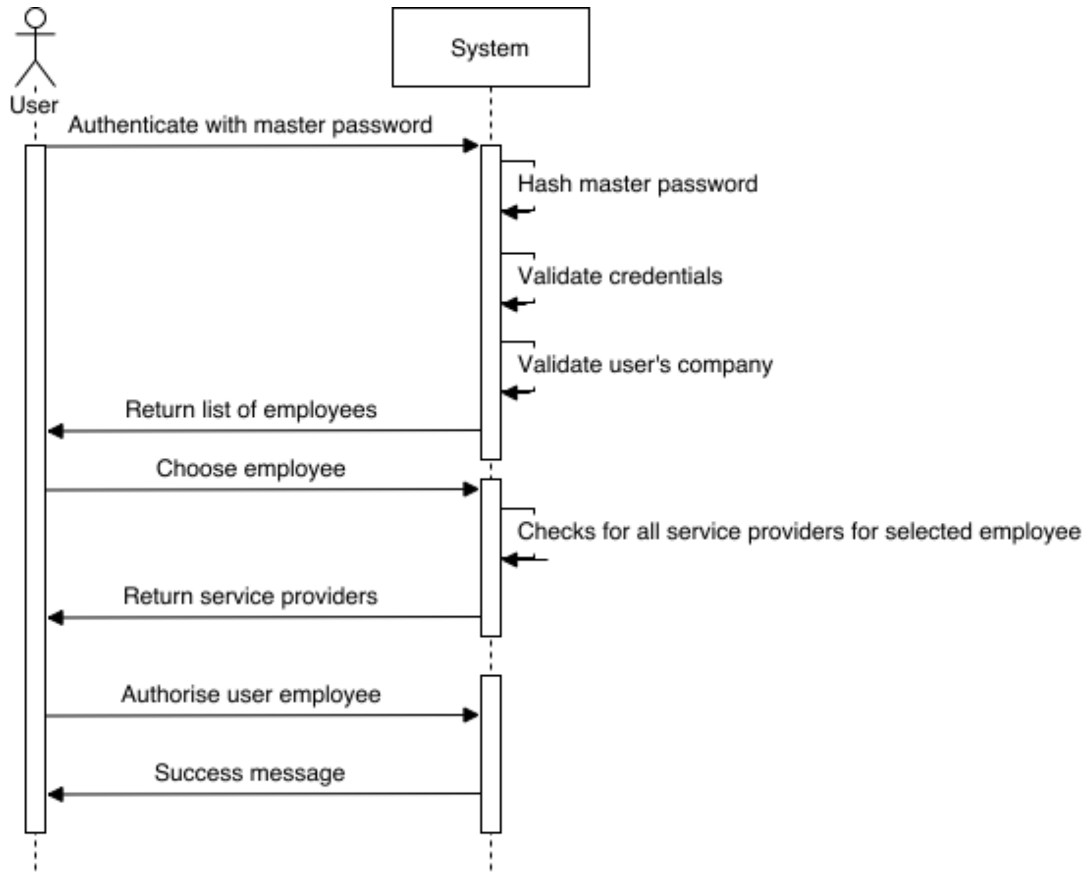


Figure 30 - Sequence Diagram 2, User Management.

Preconditions:

- Actor must have administrator permission to the system.
- Actor must be connected to the internet.

Steps:

1. Actor authenticates with username and master password.
2. System hashes the master password.
3. System checks validates credentials.
4. System checks which company the Actor works for and returns a list of employees.
5. The Actor selects an employee.
6. System returns a list of services providers for the selected employee.
7. Actor authorizes the employee for services providers.
8. System updates the database with the new service provider permissions.

Postconditions:

- System has authorized the employee to more service providers for the chosen employee.
- The chosen employee is authorized to use the service providers.

4.4.3 Sequence Diagram 3 - User Authentication

As illustrated in Figure 31, Sequence Diagram 3 describes how a user authenticates to the system consists of the user-interaction of the system, which the goal of logging into the system.

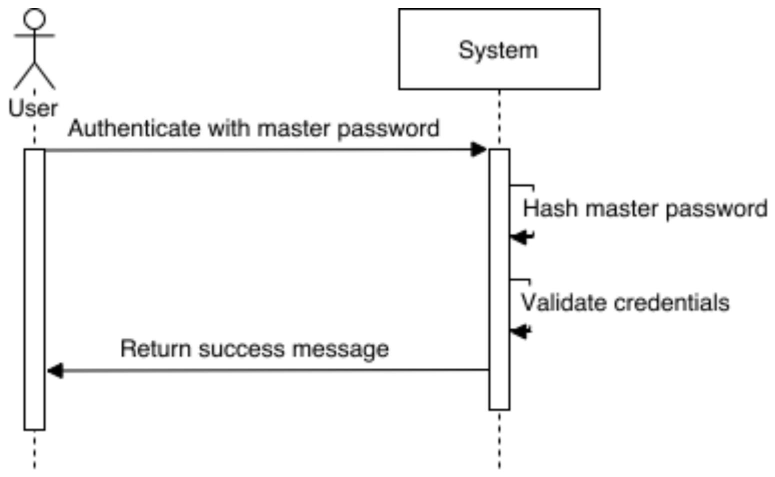


Figure 31 - Sequence Diagram 3, User Authentication.

Preconditions:

- Actor must have administrative permission to the system.
- Actor must be connected to the internet.

Steps:

1. Actor authenticates with username and master password.
2. System hashes the master password.
3. System checks validates credentials.

Postconditions:

- System has authenticated the user, who can see all its authorized service providers.

4.5 Requirement Specifications

In this section of the thesis' requirement specifications will be covered. The results and findings of this section has been derived from the 3 State of the Art chapter and 4 Analysis chapter. The requirement specifications that will be described in this section are created in order to better define the ID-Connect system and give a way of validating it aswell.

4.5.1 Functional Requirements

The functional requirements of this system are a set of parameters which are set for the system. The requirements are also a description of what the system must be able to do. Setting these requirements also gives a great way of checking if the system is up to par with the expectations that are set for it. These functional requirements can be seen below.

The system must be able to:

4.5.1.1	**Create a user:** must be able to have users in the system.
4.5.1.2	**Distinguish between regular users and administrative users:** to have administrative accounts.
4.5.1.3	**Authenticate users to service providers:** must be able to have the user work on the service provided by the service provider.
4.5.1.4	**Login to multiple service providers without needing the user to use multiple passwords:** to give users a secure and easy login method.
4.5.1.5	**Automatically choose what login method to use:** to improve the usability for the users.
4.5.1.6	**Securely login to service providers without using the same password:** to not let users have the same password and make logging in more secure.
4.5.1.7	**Securely store the user's passwords:** to minimize the risk of abuse.
4.5.1.8	**Manage which users have access to each service provider:** to give the administration of a company the ability to access manage easily.

4.5.2 User Requirements

The user requirements of this system are created to be used to specify in detail what a user interacting with the system should be able to accomplish. These user requirements are mainly derived from the early Analysis chapter, more specifically the 4.3 Scenario and 4.4 Uses Case section. Just as with the functional requirements, these requirements are also created in order to better specify the details of the ID-Connect system and have something to validate. The user requirements of this section can be seen listed below.

The user should be able to:

4.5.2.1	**Create a new user:** to be able to access the system.
4.5.2.2	**Use one password to access any service provider:** to be able to login to a service provider without having to remember multiple passwords.
4.5.2.3	**Login to various service providers from one portal:** to reduce clicks needed for the user.
4.5.2.4	**Manage existing users:** to give the administration of a company the ability to access manage easily.

4.5.3 Non-Functional Requirements

The non-functional requirements can be argued to be some of the most overall descriptive requirements. These requirements can be said to be the key characteristics of the overall ID-Connect system. Since these requirements are the overall characteristics of the system they are a great asset in the construction of the system. This is due to the requirements basically being the selling points of the system. These requirements are therefore derived partially from the research in chapter 3 State of the Art , but mainly from the 4.1 Stakeholders section of this 4 Analysis chapter. These non-functional requirements can be found listed below.

4.5.3.1 Availability

Availability of the system is always important, but in a case where a system is acting as a reversed bottleneck it is even more so. This is due to the fact that if the ID-Connect system is unavailable everything the system gives access to is also unavailable. This is therefore one of the most crucial non-functional requirements needed for the system.

4.5.3.2 Scalability

Another important requirement is the scalability aspect. Since the systems idea is to be applicable to various sizes of companies with different needs, the system needs to be able to scale and form into the desired form for each individual company. In a way where the users, access and service providers need to be able to be customized to each company's needs.

4.5.3.3 Security

Since the system is dealing with users passwords and is a gateway to service providers, security is an essential requirement. This is due to the storage of personal passwords which will need to be stored and handled in a way where the security is taken into consideration. The security of the application itself is also a must to ensure the trust of both the users and companies using the ID-Connect system.

4.5.3.4 Privacy

Privacy and security often go hand in hand, and this is no exception. One of the ways of helping with the security of the system is to also focus on the privacy of the system data. Just as with the security aspect, the privacy requirement is a must to ensure the trust of both the users and companies using the ID-Connect system.

4.5.3.5 Portability

The ID-Connect system is meant to be the gateway to various different service providers from one place and is meant to be able to be used in a vast amount of companies. Since each company has their own policy on what operational software they are running it is important to make sure that all of these can use the ID-Connect system. This is also the reason why portability is an important requirement for the system to fulfill.

4.5.3.6 Interoperability

As explained, portability as a requirement is important due to the ID-Connect system being used across multiple companies, which also is the reason for interoperability being a requirement for the system. This is in case of a company using multiple different systems to run the ID-Connect system on and still would like to be able to have one unified system.

4.5.3.7 Usability

As the last non-functional requirement there is usability. Usability is always a great asset to have for a system when the system is meant to interact with users. Together with this and the fact that the ID-Connect system is meant to make using multiple service providers easier, is the reason for usability being an important requirement.

4.6 Traceability Matrix

For the checking of the requirements a traceability matrix is created. This matrix will allow the easy checking of if the conceptual idea is fulfilling the requirements set in the Requirement Specifications section above. The way this works is by taking the three sequence diagrams from section 4.4 Sequence Diagrams and holding them up against the set requirements. This is done by having each sequence diagram in the top if the matrix in a column and the requirements on the left side as a row. Then each of the requirements can be checked up against each diagram. Each fulfilled requirement will be marked with an "X", the non fulfilment will be marked with an empty space and the requirement not being relevant for the diagram will be marked with an "/". This process can be seen in Table 1 below.

Requirement Identifiers	**Requirements Tested**	**4.4.1 Sequence Diagram 1 User Creation**	**4.4.2 Sequence Diagram 2 User Management**	**4.4.3 Sequence Diagram 3 User Authentication**
Test Cases	27	8	8	11
4.5.1.1	1	X	/	/
4.5.1.2	3	X	X	X
4.5.1.3	1	/	/	X
4.5.1.4	1	/	/	X
4.5.1.5	1	/	/	X
4.5.1.6	1	/	/	X
4.5.1.7	3	X	X	X
4.5.1.8	1	/	X	/
4.5.2.1	1	X	/	/
4.5.2.2	3	X	X	X
4.5.2.3	1	/	/	X
4.5.2.4	1	/	X	/
4.5.3.1	/	/	/	/
4.5.3.2	3	X	X	X
4.5.3.3	3	X	X	X
4.5.3.4	3	X	X	X
4.5.3.5	/	/	/	/
4.5.3.6	/	/	/	/
4.5.3.7	/	/	/	/

Table 1 - Traceability Matrix

From the traceability matrix it can be seen that there are no requirements that have gone unfulfilled. But there are some requirements which could not be tested due to them not being testable on the three sequence diagrams. These requirements were 4.5.3.1 Availability, 4.5.3.5 Portability, 4.5.3.6 Interoperability and 4.5.3.7 Usability. The reason for these requirements not being able to be tested is that they cannot be very well done on a theoretical basis. For example the 4.5.3.1 Availability requirement will heavily depend on the amount of users and what hardware the system is running on. The two requirements 4.5.3.5 Portability and 4.5.3.6 Interoperability might be able to be seen if they could work on a theoretical basis but before testing with a physical product, it would not be certain. The last 4.5.3.7 Usability requirement is a requirement which would be impossible to test successfully on an only theoretical basis. This is due to the systems usability being fairly subjective from user to user and therefore cannot be done without user testing of the UI.
All in all it can though be seen that all the requirements which are set for the system are being fulfilled in at least one of the sequence diagrams per requirement, with the exception of the requirements that were not possible to be theoretically tested.

5 Proof of Concept

After the finalization of the 4 Analysis chapter the proof of concept can be presented in the 5 Proof of Concept chapter. This chapter will focus and go into detail about how the actual system. The section contains 5.1 Choice of Technology, 5.2 Development Methodology, 5.3 Graphical User Interface, 5.4 System Architecture and 5.5 Technical Documentation. The chapters contents have been formed from the outcome of the 3 State of the Art research and the 4 Analysis chapter.

5.1 Choice of Technology

The 5.1 Choice of Technology section of the 5 Proof of Concept chapter has the goal of clearing up which technologies are used and why they were chosen. The section will feature the data storage considerations, SSO management, database structure and local- vs web-based client.

5.1.1 Hashing vs Encryption

The first consideration for the system is weather the system should be using hashing or encryption in order to store sensitive data. As mentioned earlier in the 3.3.6 Secure Data Storing section, the hashing function would be the most secure but would have the drawback of not being able to be unhashed. The encryption method would therefore in cases where the data would need to be readable be a better choice.
For the ID-Connect system the sensitive data which would be stored would be the master and the secondary password of the user. Therefore the choice would be which would suite best for the individual password type.
The choice for the master password can be stored as a hash digest since the system would not need to know what the actual password is but only if it is correct. This means that when the system stores the hash digest of the password, the hash digest can later be compared to the hash digest of the inputted password. The use of hashing would therefore be best suited for the use for the master password.
For the secondary password which is not known by the user hashing would not be an option. This is due the the fact that the password is randomly generated by the system and will be needed in plaintext for the formfill function of the system. This means that the secondary password ciphertext needs to be reversible. The choice for this is therefore to encrypt the secondary password with the master password as a key, instead of hashing it.

5.1.2 Integrated vs Third Party SSO Management

As part of the login process SSO for all the users was determined to be a functionality which would be very beneficial to the system. For the use of SSOs it could be seen that the system could have it integrated and built into the system itself or it could take use of a third party program.
It was fairly early in the process it could be seen that the implementation of a third party product to manage the use of SSOs could be favorable, due to the system merely building onto the shoulders of giants instead of recreating something that is already created and functional. The choice therefore fell onto the third party, WSO2.
The only concern there was with using WSO2 was the storage of sensitive data used for the formfill process. This data should be protected better than what WSO2 would be able to enable for the storage, and therefore parts of the data stored was moved to a separate external database and not the one integrated onto WSO2.

5.1.3 SQL vs NoSQL

The second last technology consideration was the creation of the database structure. This structure had the choice between being a very static SQL structure or a more dynamic NoSQL structure.
For the choice of the database structure the usage of the database was looked into. What could be seen was that the data inside the database would be mostly consisting of data which would not constantly change. It could also be seen that the best way of creating the database structure for the system would need to take usage or relations between the databases.
These two factors were therefore the deciding factor for the choice of an SQL relational database structure.

5.1.4 Local- vs Web Client

The last major choice that has been considered is whether the system should be based locally or as a web-based system. To figure out which choice would be best for the system the existing similar systems have been taken a look at which identified that most of them were web based. It could also be seen that since all the service providers that would be accessed are online based the required internet access for the web based system would not be an issue.
An other advantage of using a web-based system would be that the user does not need to download anything and everything will always be up to date.
These are also the main considerations and the reasoning for making the system in a web based form.

5.2 Development Methodology

For the 5.2 Development Methodology section of the 5 Proof of Concept chapter, the focus will be on how the development of the ID-Connect system itself was structured. The reason for this section is due to the fact that the development method for the actual programming of the system deviated from what was used for the entirety of the thesis itself. The development method that was used instead of the Scrum method was Extreme Programming. The reasoning of use and explanation of what Extreme programming is, will be covered in this section.

5.2.1 Extreme Programming

Just as Scrum, Extreme Programming is a development method helping to give structure in the process of creation for normally a piece of software. Extreme Programming works in a way where the development team will have various short development cycles in order to incrementally improve the software, followed by testing and evaluation of this cycle.
The perk of having even smaller development cycles than Scrum is that when working in a smaller development team these cycles can easily and quickly be summarized and everyone in the development team will be on the same page. This is due to the cycles being small and therefore limiting the size of the changes or tasks that can be done. This reason together with the easiness of working alongside one another provided by Extreme Programming is the reason for using the method over Scrum for the actual coding of the ID-Connect system.

5.3 Graphical User Interface

To get a visual representation of ID-Connect, a low-fidelity Graphical User Interface (GUI) (Sørensen L., 2016) was created both for the administrator and employee.

5.3.1 Login page

Figure 32 illustrates the login page on the left, where the administrator or an employee would sign in with their username and master password. When entered correctly, ID-Connect would prompt the user to enter a One Time Password (OTP), which would be sent to another user-agent. The prompt for OTP can be seen on the right on Figure 32.

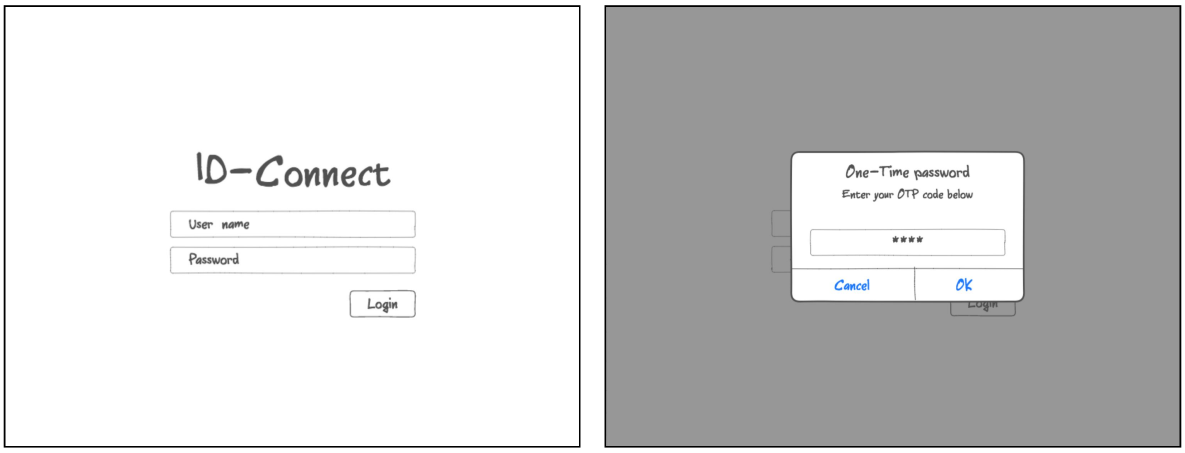


Figure 32 - Login page for ID-Connect both for administrator and employee.

5.3.1 Authentication management

<u>Administrator</u>

If the user is an administrator and is successfully authenticated, ID-Connect presents a list of all the employees the administrator is authorized to manage. If it selects an employee it can see all the service providers the selected employee is authorized to use. The administrator can then, on behalf of an employee, authenticate or deauthenticate an employee. The page can be seen on Figure 33.

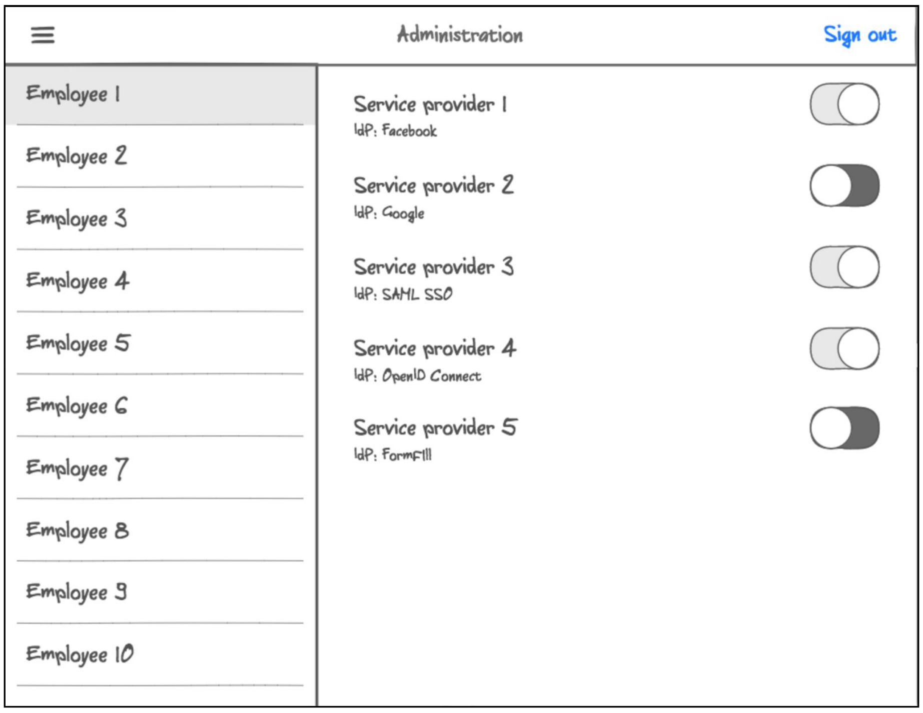
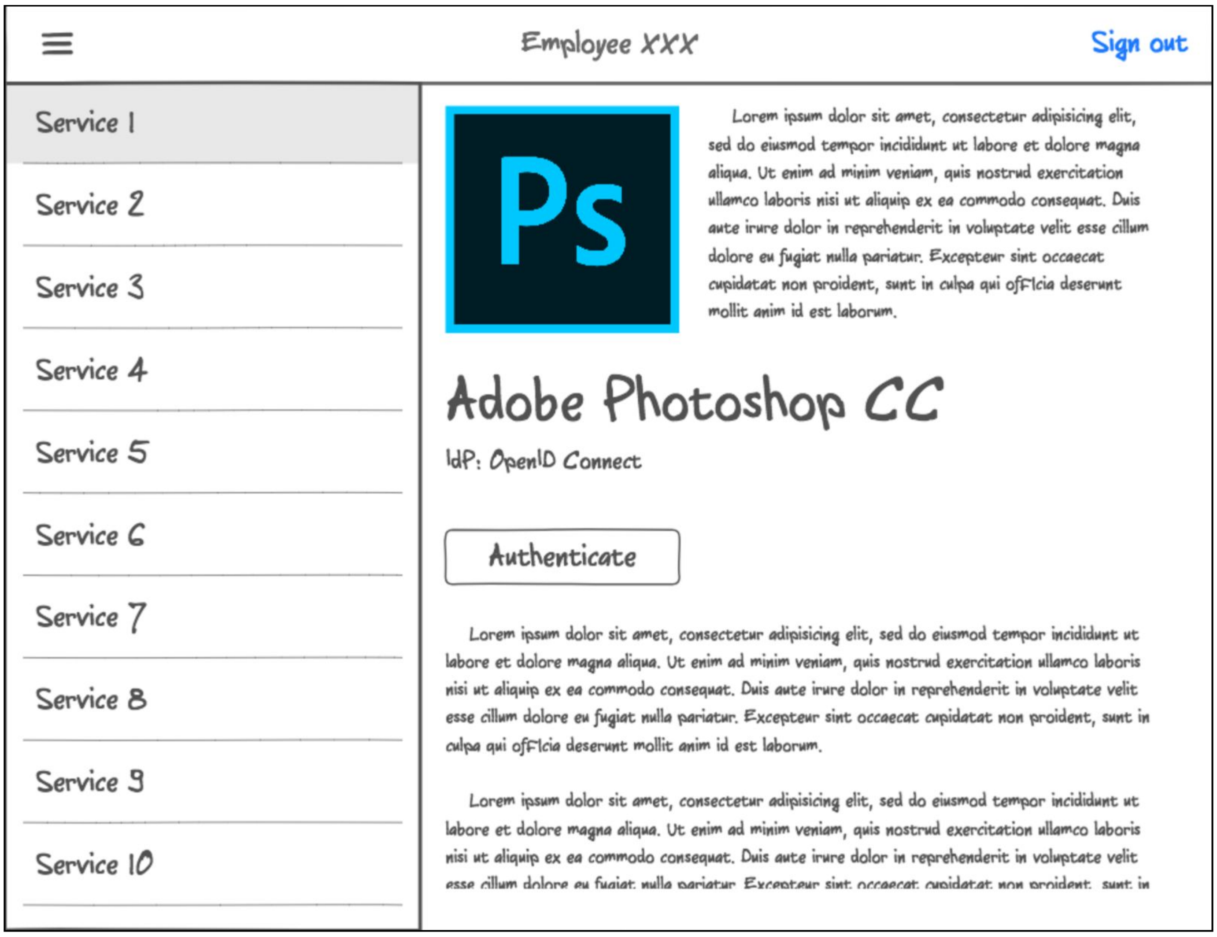


Figure 33 - Authentication management for an administrator.

<u>Employee</u>

The menu for managing an employee's authentication is similar to the administrators. Instead of seeing a list of employees, the employee sees a list of authorized services. When the employee selects a service, a new screen appears and the employee can then read about it, but most importantly authenticate to the selected service. The user will not see all the authentication mechanisms behind-the-scenes when pressing *Authenticate,* but only a loading screen since the idea is to keep the complexity away from the user. The screen is illustrated on Figure 34.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in

Figure 34 - Authentication management for an employee.

5.4 System Architecture

The 5.4 System Architecture section of the 5 Proof of Concept chapter will elaborate on all the architectural aspects of ID-Connect. This section will contain the Overall- , WSO2- , Java- and Database Architecture.

5.4.1 Overall Architecture

This section of the 5 Proof of Concept chapter will shed some light on the overall architecture of the ID-Connect system. The overall architecture which will be explained is meant to be a more high level view of the system architecture. In later sections the more detailed individual explanations of each section of the system architecture can be found.

The overall architecture have briefly been touched upon in the 1.5 Conceptual Idea section. Even though this was an early idea of how the system architecture would look, it was not too far of from how the ID-Connect system architecture ended up looking.
The major difference from the Conceptual Idea is the more detailed definition to what the system architecture contains. This means that the system can now be seen to include four major components, an *Administrator interface*, *User interface*, *WSO2* and the *ID-Connect database*. Each of these components contributes to the system, which can be seen on Figure 35.

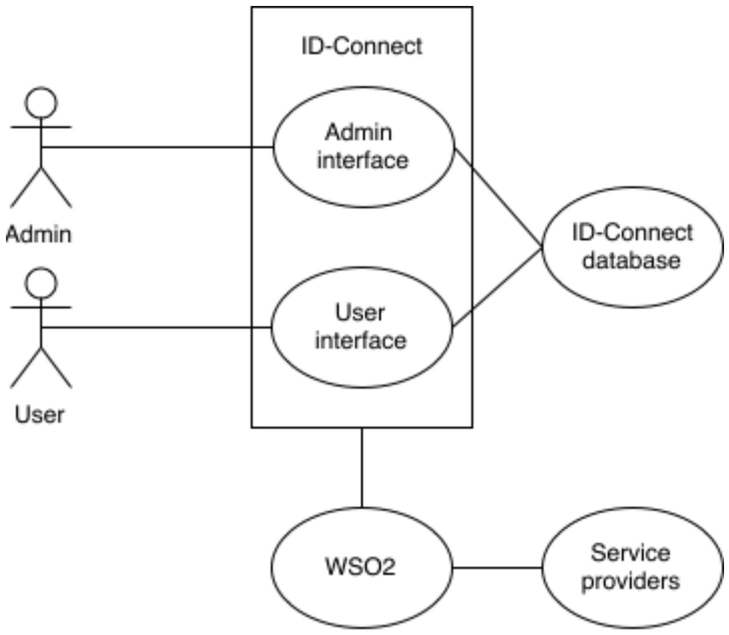


Figure 35 - Overall ID-Connect System Architecture.

Each of the four major components that make up the system will be further explained in detail, in the rest of the System Architecture section of this chapter.

5.4.2 WSO2 Architecture

A major component of the overall architecture was WSO2 IdS. IdS was the backbone of the authentication framework that transparently handled the authentication between an employee and service providers. Figure 36 illustrates the architecture of IdS together with ID-Connect and the information flow will be explained.

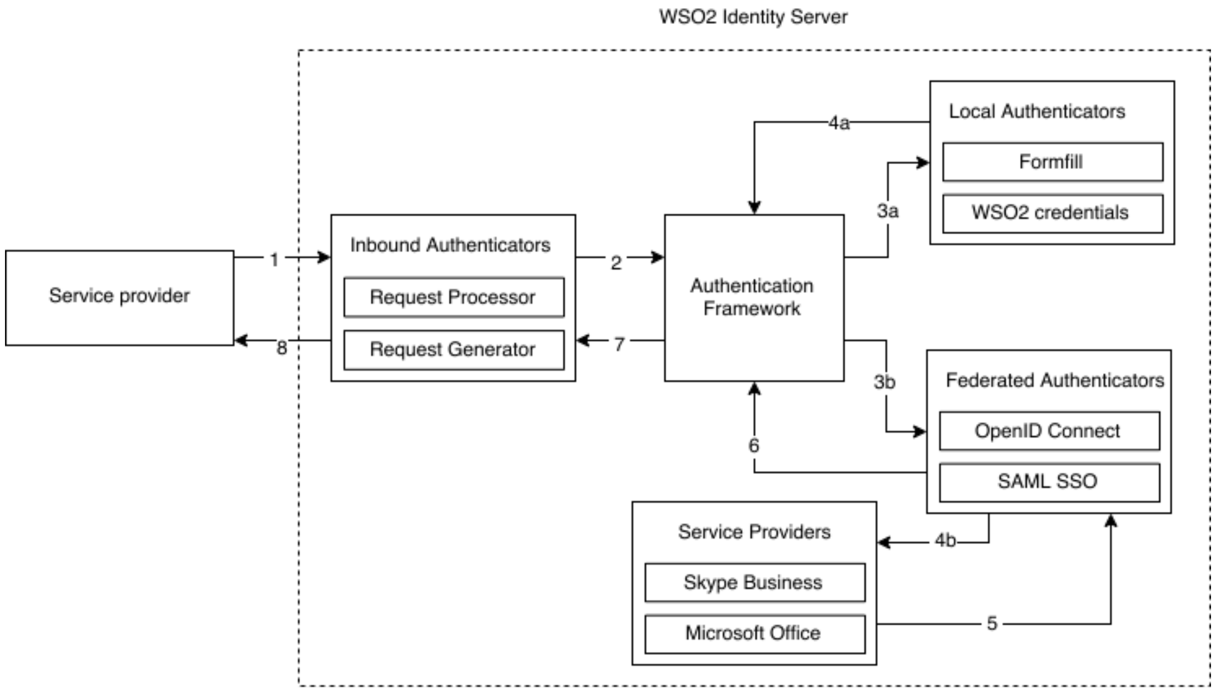


Figure 36 - Architecture of WSO2 IdS within his project's system.

Step 1

An authorized employee selects a service provider from *ID-Connect* which it wishes to authenticate. The service provider sends a request to the *Request Processor* inside the *Inbound Authenticator* component located on the IdP. This component evaluates if the request is of type SAML-SSO, and if so, *Request Processor* continues to the next step otherwise it rejects the request. Metaphorically, *Request Processor* is like a doorman making sure only the listed guests are entering the building.

Step 2

Request Processor sends a request to the *Authentication Framework*, which is the heart of IdP. This request consists of the claims the service provider wishes to access restricted resources which *Authentication Framework* can handle with the function called *claim mapping*. From the request and claims the *Authentication Framework* can determine whether the authenticator is local or federated.

Step 3a
If the authenticator is local, IdS chooses the correct IdP depending on service provider's configuration within the *Management Console*. However, due to this project's limitations the local-authenticator will always be *FormFill*.

Step 3b
If the authenticator is federated, the selected service provider will be authenticated through any IdP, such as Facebook or Google, supporting the federated authenticators that could be OpenID Connect, SAML SSO, WS-Federation etc. For instance, the selected service provider supports Facebook-login hence *Federated Authenticator* will authenticate with Facebook.

Step 4a
Formfill returns the response message from the web page, where the formfill was located, to the *Authentication Framework*. For the sake of simplicity the framework determines whether the response was of type *success* or *error*.

Step 4b
The *Federated Authenticator* redirects the users to the selected service provider, where the actual authentication between the employee and the service provider occurs. The way IdP communicates with IdS are through the configurations made in *Management Console*.

Step 5
After authentication, the *Service Provider* returns a token, which could be an ID-token depending on which type of IdP the selected service provider supports to *Federated Authenticator*.

Step 6
Authentication Framework receives the token extracts the scopes, and matches them with the claims defined by the selected service provider inside *claim mapping*. If correctly matched

Step 7 and 8
The *Authentication Framework* passes the token to the *Request Generator*, which returns back to the *Service provider*. Now the employee is authenticated with the selected service provider and can use its services.

5.4.3 Java Architecture

This subsection of the System Architecture section is dedicated to explain the architecture of the Java part of the developed system. This subsection will be split into the administrator and user side of the system. These two architectural structures will be explained in detail followed with their technical documentation in the next section, 5.5 Technical Documentation.

5.4.3.1 Administrator Side

The first part of this section is the administrator side of the system. The goal of the administrator side is to give an administrator access to manage which users have access to which service providers. This is achieved with three steps, login, user choice and user access control. These three steps can be seen in Figure 37 below.

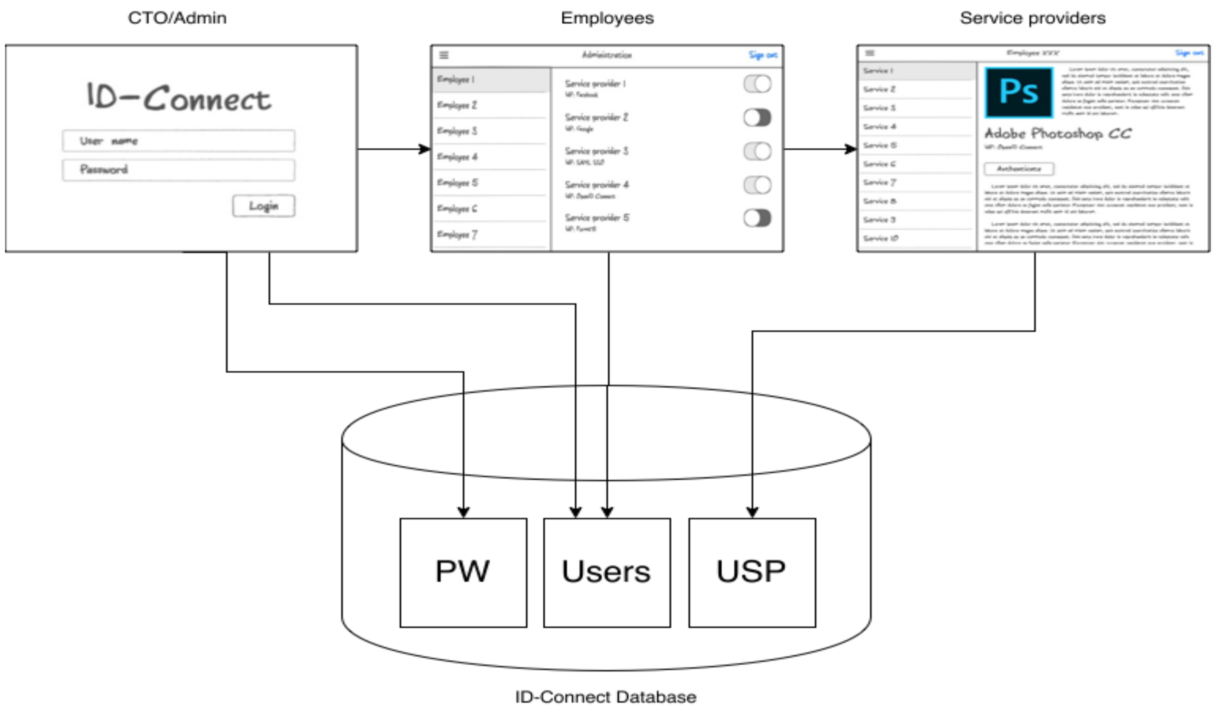


Figure 37 - Administrator Side Architecture.

As mentioned the first step was to login. This process firstly isolates the users which do not have administrative access and have wrong passwords, then determines what company the administrator is from.
The first step is done by SHA-256 hashing the inputted password and checking in the database if the hashed password digest matches with the username. After this is done the system checks if the user has access to the system, by seeing if the user is an administrator. This first step can be seen on the left side of Figure 37.

Step two is when an administrator is logged in. The administrator will be presented with all the users registered under the same company as them. This is done by having the system check in the database which company the administrator is representing and pulls all other users in this company. This step can be seen in the middle of Figure 37.
The last step is when the administrator has chosen a user to manage. The system will then pull all the service providers that are available from the database and compare them to which are active for the chosen user. The service providers that are active for the user is also stored in the database. When this is done the administrator has the possibility to grant and revoke access to each of the service providers for the user. This process will then refresh the database with the new permissions for the user. The third and last step can be seen on the left side of Figure 37.

5.4.3.2 User Side

The second part of the Java architecture is the user side of the system. The goal of this second part is for the user to be able to login, see which service providers they have access to and login to them. This can be accomplished in three steps, primary login, service provider choice and secondary login. The flow of this can be seen in Figure 38 below.

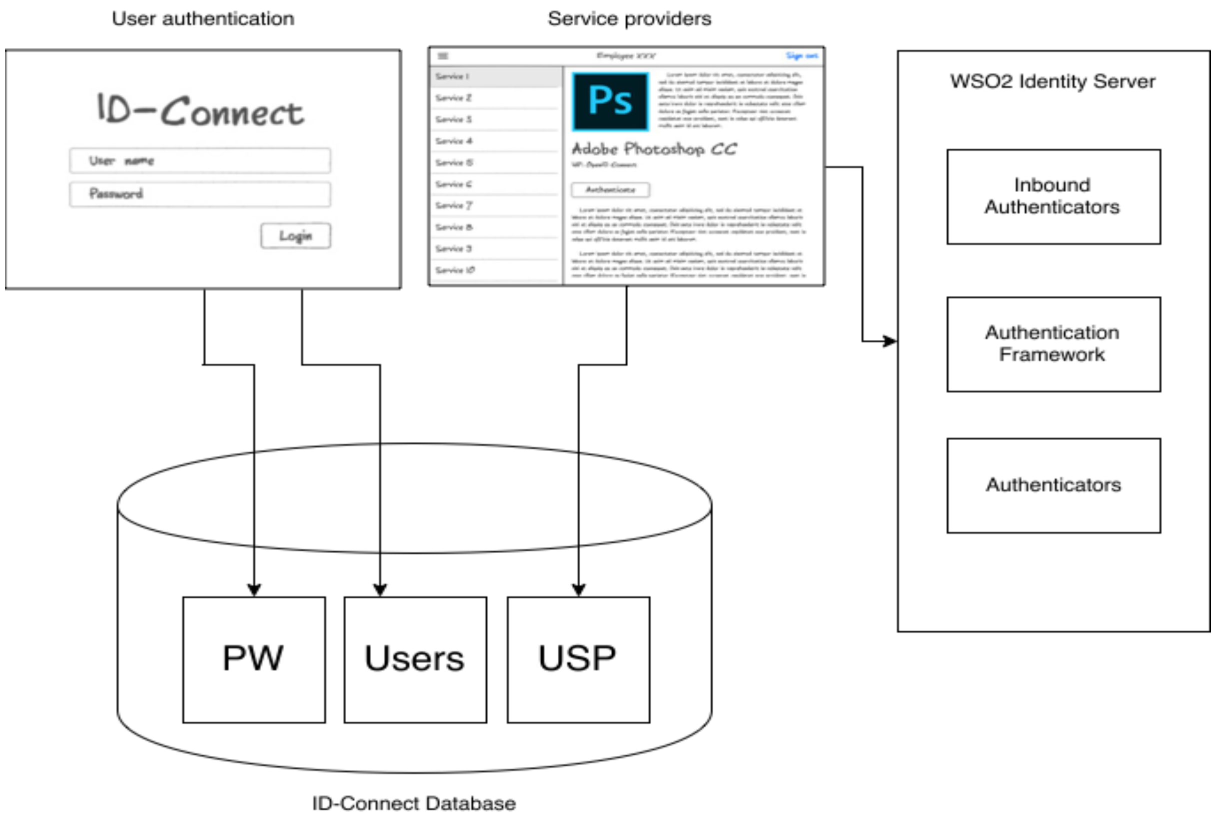


Figure 38 - User Side Architecture.

The first step was for the user to login. This is done by the user entering a username and a password. The password is then hashed with SHA-256 and together with the username checked for a match in the database. This can be seen in the left side of Figure 38.
The second step is the the user to retrieve a list of service providers the user has access to. The way this is accomplished is by looking in the database after the specific user and seeing which service providers they are connected to. These service providers are then shown for the user. This can be seen in the middle of Figure 38.
The last part of the user side is choosing a service provider and logging in. This is done by the system searching for the chosen service provider and seeing which login method is available for the service provider for the user in the database. This could be various SSOs or with the formfill function of the system. The secondary login and method is done automatically by the system, by what is available. WSO2 automatically handles if the authentication happened through federated identity such as SSO. If the formfill function is used the system would check for the formfill data that is needed for the chosen service provider. This data is then given to the service provider and the user is logged in. This process can be seen on the right side of Figure 38.

5.4.4 Database Architecture

The last section of the System Architecture chapter is the Database Architecture. This section will describe the inner working of the database and its structure. The main parts of the database can be split into two major parts, the *User Table* and the *ServiceProvider Table*. Both of these major parts are sharing some tables between each other. The *User Table* is the table storing the user data and is connected to the *Company*- and *PW Table* with a foreign key to know which user belongs to which company and master password. Together with that the *User Table* is also connected to the shared tables *PW2*- and *USP Table*. These two tables contain the secondary passwords in the *PW2 Table* and the data for what user is connected to which service provider in the *USP Table*. The other major part is the ServiceProvider Table which contains the service provider data and is connected to the *FormFill Table* in order to tie together the data needed for the form fill to the specific service provider. The *USP*- and *PW2* Tables are also used to tie the user data and the secondary password to the service provider data. The database architecture is designed with from the needs of the system together with the database rules of normalization.
An overview of this can be seen the the Entity Relationship (ER) diagram shown in Figure 39. In Figure 39 the ER diagram will illustrate the databases entities with all their attributes in relation to each other.

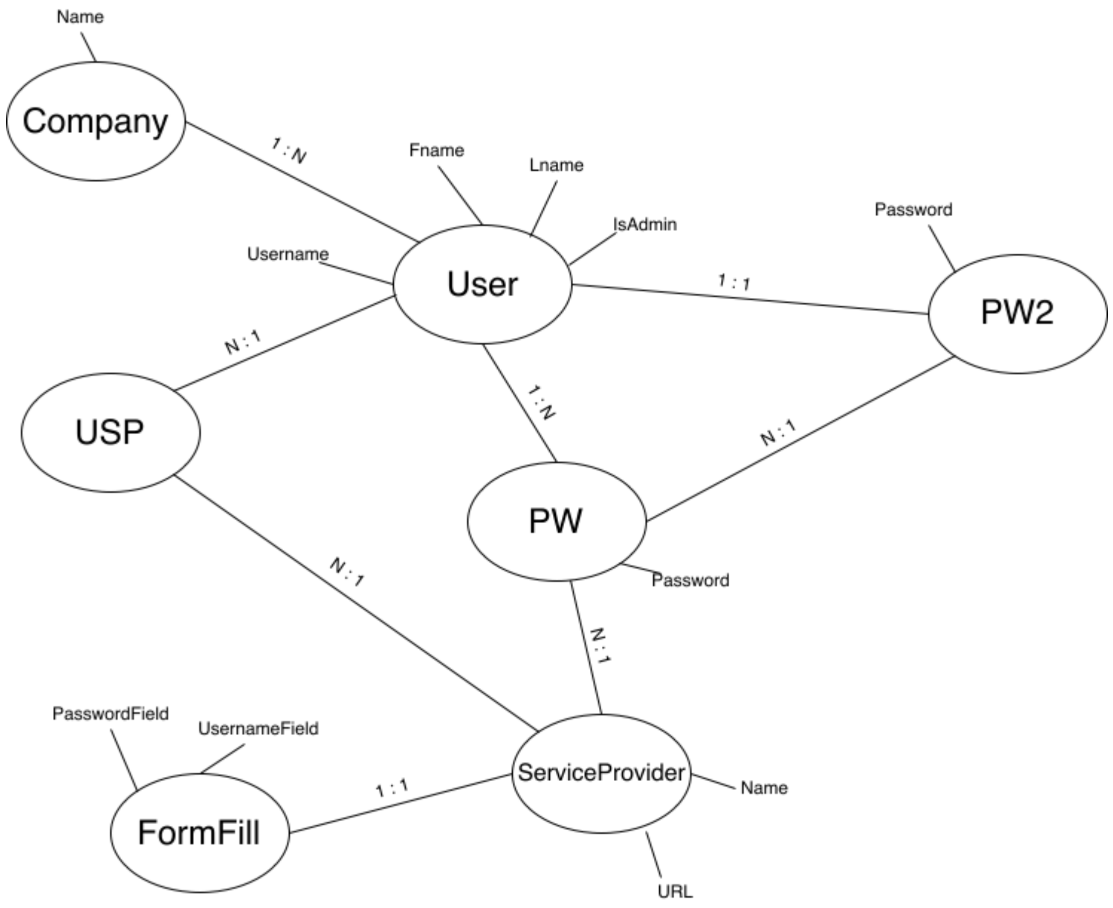


Figure 39 - ER diagram of Database Architecture

5.4.4.1 User Table

As previously mentioned the *User Table* is one of the main table in the database. The *User Table* has the main purpose to keep track of what users there are in the system. The attributes that the *User Table* has is a *UID* as a primary key, a *Fname, Lname* and *Username* to store the user's first name, last name and username. The table also has *IsAdmin* as an attribute, which determines if the user is an admin. In the table there can also be found two foreign keys, *PID* and *CID*. These two foreign keys are referring to the primary keys of the *Password Table* and the *Company Table*. An example of the table can be seen in Table 2 below, where Alice and Bob have been created as users.

UID[PK]	Fname	Lname	Username	IsAdmin	PID[FK]	CID[FK]
0	Alice	Anderson	aander17	1	0	1
1	Bob	Boberson	bbober17	0	1	1

Table 2 - User Table

5.4.4.2 Company Table

The *Company Table* is dedicated to handle all data about a company. In this case the only parameter that is stored about the registered companies are their names. This means that the only thing that the table contains is the *CID* primary key referenced by the *User Table*, and the *Name* attribute containing the name of the company. The example of this can be seen in Table 3 below, where the companies Facebook and Google have been created. It can also be seen in Table 3 that both Alice and Bob are tied to the Google company.

CID[PK]	Name
0	Facebook
1	Google

Table 3 - Company Table

5.4.4.3 ServiceProvider Table

The second major part of the database is the *ServiceProvider Table*. This table is designed to keep track of the data related to the service providers. The table contains a *SPID* primary key, a *Name* attribute containing the name of the name of the service provider and a *URL* attribute storing the URL of the service provider login page. The table also has a *FID* foreign key referencing to the *FormFill Table*. An example of this table can be seen in Table 4 below, where Silvan and Twitter have been registered with their URLs as service providers.

SPID[PK]	**Name**	**URL**	**FID[FK]**
0	Silvan	https://mit.silvan.dk/	1
1	Twitter	https://twitter.com/?lang=en	0

Table 4 - ServiceProvider Table

5.4.4.4 FormFill Table

The next table is the *FormFill Table*, which enforces and ties to the service provider information. The main purpose of the table is to keep track of what each of the login field and password field are called for each service provider. The table contains the *FID* primary key which is referenced from the *ServiceProvider Table* and the *UsernameField* and *PasswordField* attribute. The two attributes are storing the username and password field names used for the formfill function. An example of this table can be seen in Table 5, where the username and password fields can be seen for Silvan and Twitter.

FID[FID]	**UsernameField**	**PasswordField**
0	signin-email	signin-password
1	logonId	logonPassword

Table 5 - FormFill Table

5.4.4.5 USP Table

The *USP Table* is a combination table that in itself does not contain any data but is used as a reference table. The main goal of the *USP Table* is to keep track of which users have access to which service providers. Therefore the table only contains a *USPID* primary key and a *UID* and *SPID* foreign key. The *USPID* primary key is used by the administrative part of the system to show which users have enabled specific service providers. The two foreign keys refers to the *User Table* and *ServiceProvider Table* primary keys. This can be seen as an example in Table 6 where the user Alice Anderson has been tied to the service providers Silvan and Twitter.

USPID[PK]	**UID[FK]**	**SPID[FK]**
0	0	0
1	0	1

Table 6 - USP Table

5.4.4.6 PW Table

The second last table is the *PW Table* which contains the user's master password. As mentioned, this password is the master password which the user uses to get into the system itself. This table contains the *PID* primary key and the *Password* attribute. The *Password* attribute stores the hashed digest of the users. This can be seen in Table 7, where the passwords for Alice and Bob are stored as a 32-byte SHA-256 digest.

PID[PK]	**Password**
0	9b8c062b29bb52d0aafee34a1b806cb01ab23f3a5b3898e27 d2f2adfc4ac02ef
1	7ccadbd2468818c4e8f1a8db0aeba7efe16daaa12606e72e3a 6d4b36177474e4

Table 7 - PW Table

5.4.4.7 PW2 Table

The last table is the *PW2 Table* which has the duty to store the users randomly generated secondary passwords. These passwords were the randomly created passwords used for the formfill for the service providers. The table contains the *P2ID* primary key, *Password* attribute, and the three foreign keys, *UID, SPID* and *PW*. The *Password* attribute contains the main password encrypted secondary password. The three foreign keys are referring to the primary key of the *User-*, the *ServiceProvider-* and *PW Table.*
In the example in Table 8 there can be seen two secondary password, which are tied to user Alice as seen from the *UID* foreign key. This also means that the passwords that are stored are AES-256 encrypted with Alice's master password, which also can be seen by the *PW* foreign key. These two secondary passwords are for are for the formfill at Silvan and Twitter as it can be seen from the foreign key *SPID*.

P2ID[PK]	**Password**	**UID[FK]**	**SPID[FK]**	**PW[FK]**
0	+Pm5Hc3iVMALSJIURyyxmsTLlCbtpF995t18hn1GdA8=	0	0	0
1	Ix4CGzXoJ/zR0hVxImwd5wOG/31DggfqqChmRozcPIs=	0	1	0

Table 8 - PW2 Table

5.5 Technical Documentation

This section explains how the proof of concept was developed and implemented in this project. It will walk through how the IdS in WSO2 was installed and implemented and how the Java implementation was merged with the IdS.

5.5.1 WSO2 Identity Server

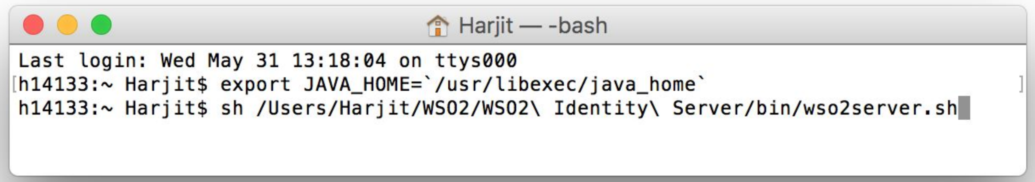
The first section of the 5.5 Technical Documentation will cover the WSO2 Identity server. This contains the Installing, Launching and Configuration.

5.5.1.1 Installing IdS

The IdS on WSO2 was available as open source provided by WSO2 and required no registration or purchases to use their source files. Installing the product was straightforward and only required to be downloaded to a file storage, which was located on one of the group member's PC (Macbook Pro running MacOS Sierra).

5.5.1.2 Launching IdS

Since the server was installed on a Mac, starting the server was done through the *Terminal.* Before starting the server the Java environment variables must be correctly set up (ORACLE, 2015). This was done with the first command illustrated on Figure 40. Straight after, the next command on Figure 40 would launch the server.



```
Last login: Wed May 31 13:18:04 on ttys000
h14133:~ Harjit$ export JAVA_HOME=`/usr/libexec/java_home`
h14133:~ Harjit$ sh /Users/Harjit/WSO2/WSO2\ Identity\ Server/bin/wso2server.sh
```

Figure 40 - Launching the IdS from Terminal.

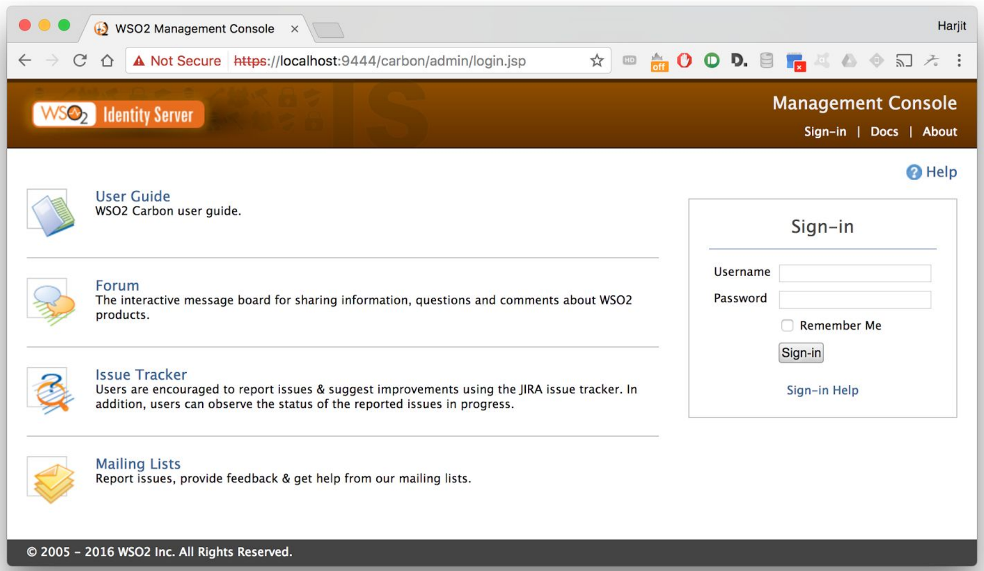
If the IdS was successfully initiated the following message appeared:

```
X509 Certificate Servlet activated successfully...
```

To verify if the server was running correctly the group members opened the URL associated with the running server. The URL was *localhost* since it runs locally on the machine.

```
https://localhost:9444
```

The *Management Console* would appear when visiting the URL, which is displayed on Figure 41, if the server is correctly initiated.


WSO2 Management Console

Not Secure https://localhost:9444/carbon/admin/login.jsp

Management Console

Sign-in | Docs | About

Help

User Guide
WSO2 Carbon user guide.

Forum
The interactive message board for sharing information, questions and comments about WSO2 products.

Issue Tracker
Users are encouraged to report issues & suggest improvements using the JIRA issue tracker. In addition, users can observe the status of the reported issues in progress.

Mailing Lists
Report issues, provide feedback & get help from our mailing lists.

Sign-in

Username

Password

Remember Me

Sign-in

Sign-in Help

© 2005 – 2016 WSO2 Inc. All Rights Reserved.

Figure 41 - Management Console of the IdS will show if the server was correctly started.

5.5.1.3 Configuring IdS

After initiating the server and signing into to super-administrator, the group members could access the *Management Console*. The console is the frontend of IdS allowing users, which would be those who administrate the whole ID-Connect backend, to manage all users, service providers, identity providers, policies etc.

Due to the focus area for this project, the group members did not use the console extensively, but mostly to verify when the server was running.

5.5.2 Java Development

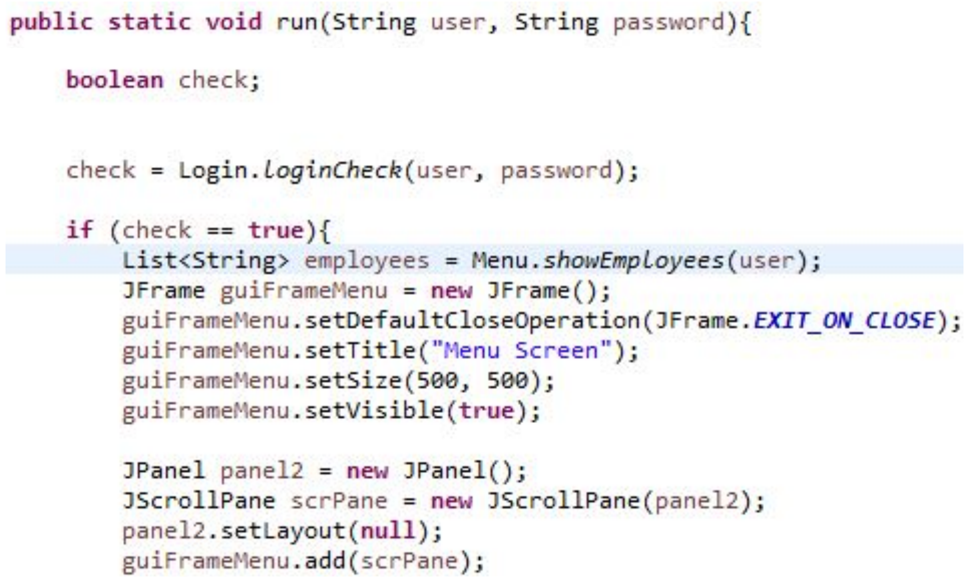
The Java Development section consists of the code which has been created for the proof of concept to come to creation, and will elaborate of the inner workings of this. The section consists of the Administrator Side, User Side and WSO2 part of the system.

5.5.2.1 Administrator Side

The Administrator side of the system can be boiled down into six different classes, *Main, Login, Menu, UserSpec, ToggleServiceProvider* and *ServerConnect.* These six classes will in this section be elaborated upon.

Main Class

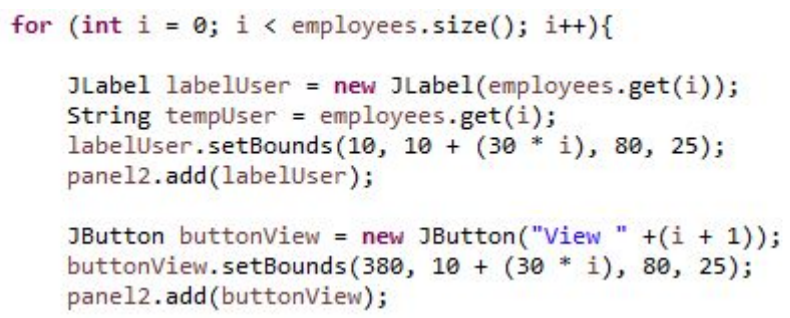
The *Main* class is the first class which will be executed when the system starts up. This class starts by creating the layout of the system and leads to the methods used for the system.
The first action of the system is when the user clicks the login button. When this happens the method *loginCheck* in the *Login* class is executed with the input from the username and password field. The *Login* class will be elaborated later in the section. The reason for calling this method is to check if the username and password match, if they do the method will return a *True* statement, if not a *False* one. If the method returns *False* the system will tell the user that the password does not match the username. If the method on the other hand returns *True* the systems logs the user in and moves on the the next step. This can be seen in Figure 42 below.



```
public static void run(String user, String password){

    boolean check;


    check = Login.loginCheck(user, password);

    if (check == true){
        List<String> employees = Menu.showEmployees(user);
        JFrame guiFrameMenu = new JFrame();
        guiFrameMenu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        guiFrameMenu.setTitle("Menu Screen");
        guiFrameMenu.setSize(500, 500);
        guiFrameMenu.setVisible(true);

        JPanel panel2 = new JPanel();
        JScrollPane scrPane = new JScrollPane(panel2);
        panel2.setLayout(null);
        guiFrameMenu.add(scrPane);
```

Figure 42 - Main Class, Login Check

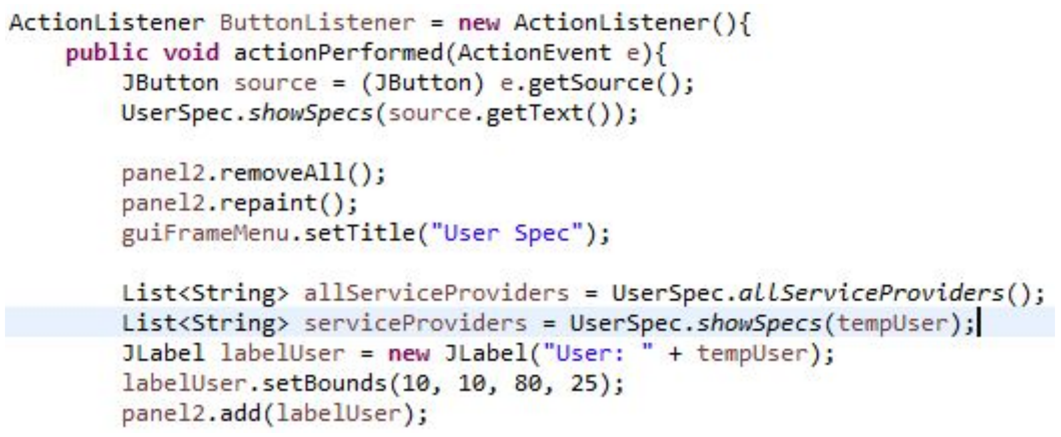
As Figure 42 also shows, the next step which is taken is pulling and displaying all the users for the company which the logged in user is representing. This is done by *Main* class calling the *showEmployees* method inside the *Menu* class, with the given username as input. The *Main* class will also be elaborated on later in this section. What the *showEmployees* method outputs is an array of employees from the company which the inputted user is from.
The returned array with the employees is then used to create the view of employees for the logged in user. This can be seen in Figure 43 below.



```
for (int i = 0; i < employees.size(); i++){

    JLabel labelUser = new JLabel(employees.get(i));
    String tempUser = employees.get(i);
    labelUser.setBounds(10, 10 + (30 * i), 80, 25);
    panel2.add(labelUser);

    JButton buttonView = new JButton("View " +(i + 1));
    buttonView.setBounds(380, 10 + (30 * i), 80, 25);
    panel2.add(buttonView);
```

Figure 43 - Main Class, Employee Pull

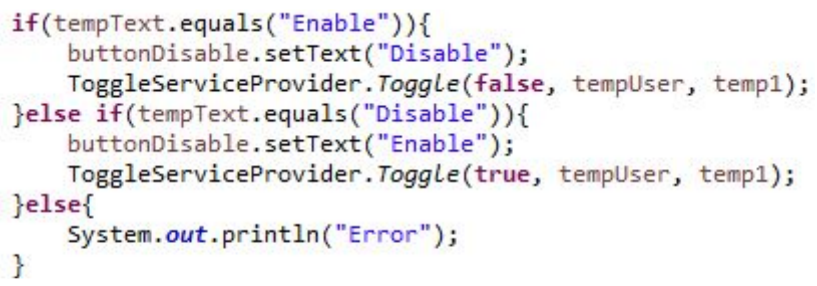
The way the overview of employees is created is by having a loop the size of the returned array. This creates a label for the username and a view button to edit the permission of the user which is created.
Each of the buttons have an actionlistener attached to them to be able to give them an action when clicked. Each of the actionlisteners have the task to create a new view of the selected employees active service providers. This works by calling two methods, *allSericeProviders* and *showSpecs* from the *UserSpec* class. The reason for calling these methods is to first pull all the available service providers and then pull which service providers are already enabled for the user. These two methods will be elaborated on later in this section. The code snippet for this step of the system can be seen in Figure 44 below.



```
ActionListener ButtonListener = new ActionListener(){
    public void actionPerformed(ActionEvent e){
        JButton source = (JButton) e.getSource();
        UserSpec.showSpecs(source.getText());

        panel2.removeAll();
        panel2.repaint();
        guiFrameMenu.setTitle("User Spec");

        List<String> allServiceProviders = UserSpec.allServiceProviders();
        List<String> serviceProviders = UserSpec.showSpecs(tempUser);
        JLabel labelUser = new JLabel("User: " + tempUser);
        labelUser.setBounds(10, 10, 80, 25);
        panel2.add(labelUser);
```

Figure 44 - Main Class, Employee Service Provider Pull

The last part of this class is the toggling of each service provider permission. This is done by, as mentioned pulling all service providers and thereafter the once that are enabled for the user. Then the system will allow the user to revoke access if the access is there or provide access if it is not.
This is done with the *ToggleServiceProvider* classes *Toggle* method, with the toggle change, employee and service provider at parameters. A code snippet of this can be seen below in Figure 45.

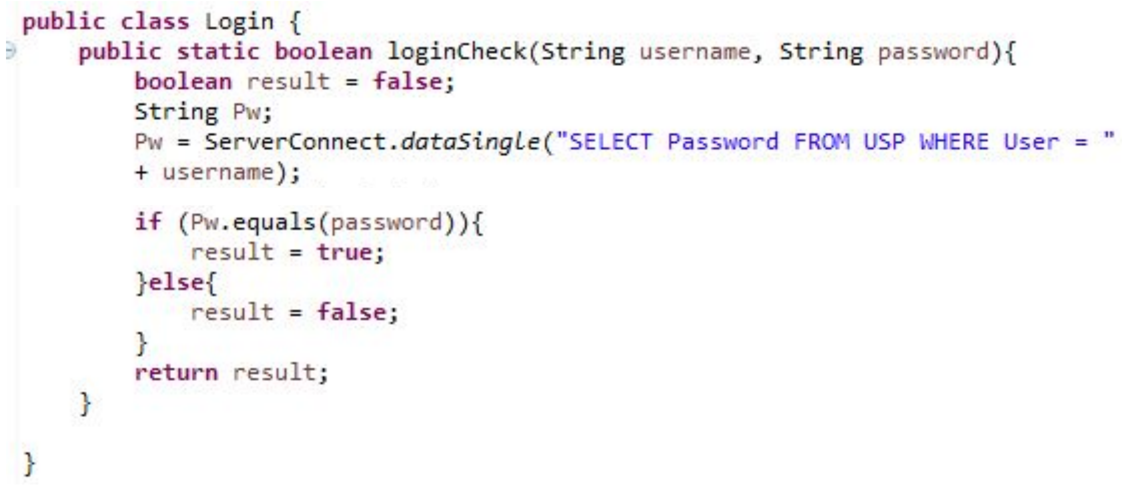


```
if(tempText.equals("Enable")){
    buttonDisable.setText("Disable");
    ToggleServiceProvider.Toggle(false, tempUser, temp1);
}else if(tempText.equals("Disable")){
    buttonDisable.setText("Enable");
    ToggleServiceProvider.Toggle(true, tempUser, temp1);
}else{
    System.out.println("Error");
}
```

Figure 45 - Main Class, Toggle Service Provider Permissions

Login Class

The *Login* class is the second class of the system and only contains the *loginCheck* method. The methods main responsibility is the check if a username is correlated to a user. This is done by calling the *dataSingle* method in the *ServerConnect* class with the username as a parameter, to return the password for the user. When this is done the password inputted can be compared with the one gotten from the *dataSingle* method. If the password is a match the method will return a *True* boolean and if not a *False* one. This piece of code can be seen below in Figure 46.

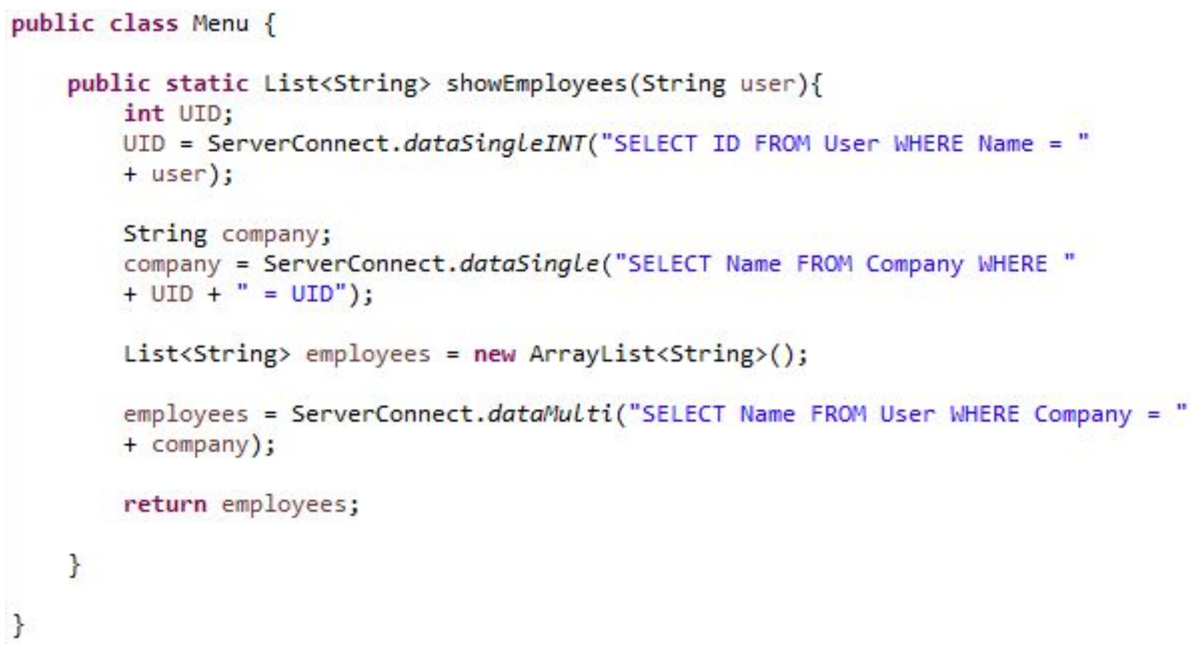


```
public class Login {
    public static boolean loginCheck(String username, String password){
        boolean result = false;
        String Pw;
        Pw = ServerConnect.dataSingle("SELECT Password FROM USP WHERE User = "
        + username);
        if (Pw.equals(password)){
            result = true;
        }else{
            result = false;
        }
        return result;
    }

}
```

Figure 46 - Login Class

Menu Class

The *Menu* class has again only one method, the *showEmployees* method. This methods task is to return an arraylist filled with all the users from the inputted users company. For this task to be possible the method needs to know three things, firstly what user id does the user havde, what company does that user id correlate to and finally which users work for that company. This is done by first calling the *dataSingleINT* method inside the *ServerConnect* class to return the to return the user id. The user id is then used to call the *dataSingle* method to return what company the user works for. To end the method the *dataMulti* method is called with the company name. The *dataMulti* method returns an arraylist with all the employees for the inputted company, which thereafter is returned by the *showEmployees* method. This can be seen in Figure 47 below.



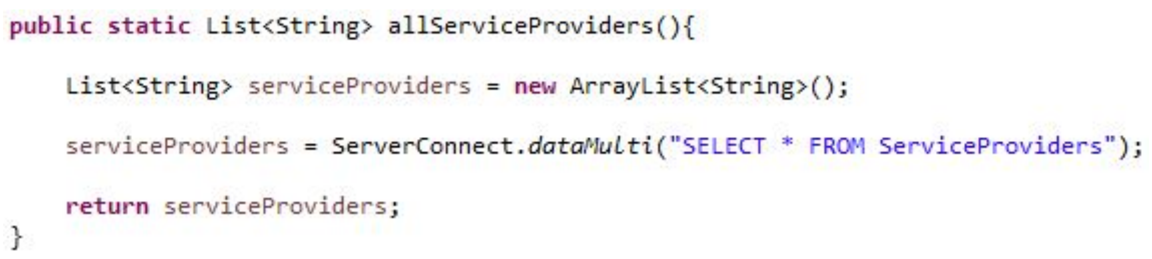
```
public class Menu {

    public static List<String> showEmployees(String user){
        int UID;
        UID = ServerConnect.dataSingleINT("SELECT ID FROM User WHERE Name = "
        + user);

        String company;
        company = ServerConnect.dataSingle("SELECT Name FROM Company WHERE "
        + UID + " = UID");

        List<String> employees = new ArrayList<String>();

        employees = ServerConnect.dataMulti("SELECT Name FROM User WHERE Company = "
        + company);

        return employees;

    }

}
```

Figure 47 - Menu Class

UserSpec Class

The fourth class of the system is the *UserSpec* class. This class consists of two methods, the *allServiceProviders* and *showSpecs*.

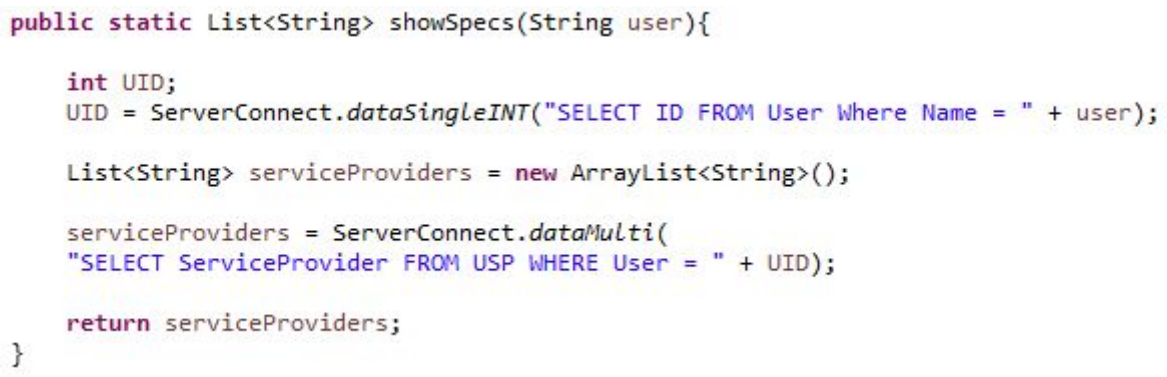
allServiceProviders method is a method created in order to return all service providers registered. The method calls a the *dataMulti* method inside the *ServerConnect* class which returns all the service providers in the database inside of an arraylist. This list is then returned by the *allServiceProviders* method. This method can be seen in Figure 48 below.



```
public static List<String> allServiceProviders(){

    List<String> serviceProviders = new ArrayList<String>();

    serviceProviders = ServerConnect.dataMulti("SELECT * FROM ServiceProviders");

    return serviceProviders;
}
```

Figure 48 - UserSpec Class, allServiceProviders Method

The other method inside of the *UserSpec Class* is the *showSpecs* method. This method will return the service providers enabled for the user that is inputted into the method. For this to be possible the method first calls the *dataSingleINT* method to get the user id of the inputted user. Thereafter the method calls the *dataMulti* method to return all the service providers for that given user id and at the end the *showSpecs* method returns the arraylist of service providers. This can be seen in Figure 49 below.

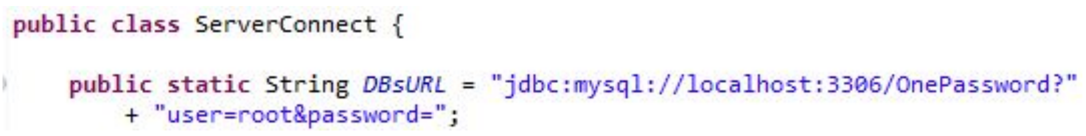


```
public static List<String> showSpecs(String user){

    int UID;
    UID = ServerConnect.dataSingleINT("SELECT ID FROM User Where Name = " + user);

    List<String> serviceProviders = new ArrayList<String>();

    serviceProviders = ServerConnect.dataMulti(
    "SELECT ServiceProvider FROM USP WHERE User = " + UID);

    return serviceProviders;
}
```

Figure 49 - UserSpec Class, showSpecs Method

ToggleServiceProvider Class
The *ToggleServiceProvider* class only has the *Toggle* method used for either removing a user from the USP database table or adding them there with the given service provider.

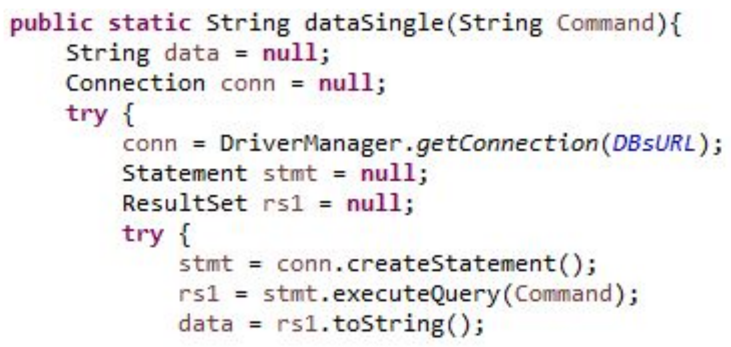
ServerConnect Class
The *ServerConnect* class consists of three different methods which all connect to the system database and return data there from. The three methods of the class are the *dataSingle, dataSingleINT* and *dataMulti*. These three method return a single string value, single integer value or multiple string values. Before these method can be used the *ServerConnect* class will define the database URL and password. This can be seen in Figure 50.



```
public class ServerConnect {

    public static String DBsURL = "jdbc:mysql://localhost:3306/OnePassword?"
        + "user=root&password=";
```

Figure 50 - ServerConnect Class Database URL

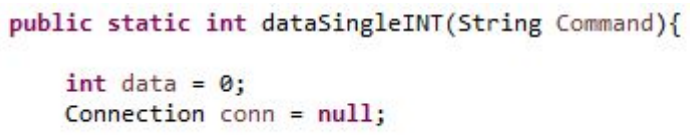
The first *dataSingle* method starts by first creating a connection to the database with the defined database URL. Thereafter the method relays the inputted command along to the database. This can be seen in Figure 51 below.



```
public static String dataSingle(String Command){
    String data = null;
    Connection conn = null;
    try {
        conn = DriverManager.getConnection(DBsURL);
        Statement stmt = null;
        ResultSet rs1 = null;
        try {
            stmt = conn.createStatement();
            rs1 = stmt.executeQuery(Command);
            data = rs1.toString();
```

Figure 52 - ServerConnect, dataSingle Method

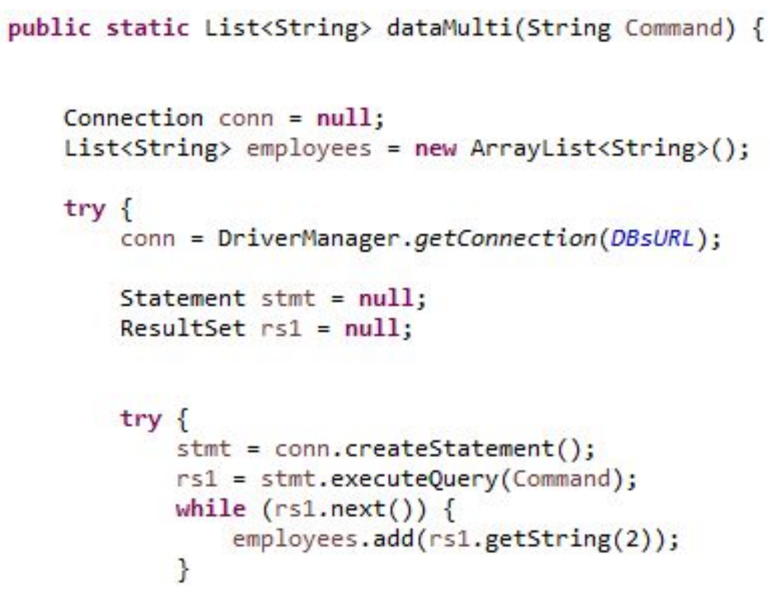
The second *dataSingleINT* method works in the same way as the *dataSingle* method with the only difference being that the returned value is an integer. The change can be seen in Figure 53 below.



```
public static int dataSingleINT(String Command){

    int data = 0;
    Connection conn = null;
```

Figure 53 - ServerConnect, dataSingleInt Method

The third and last method in the *ServerConnect* class is the *dataMulti* method. The difference with this method compared to the two previous methods is that this method returns an arraylist instead of a single value. The way this works is by again connecting to the database with the database URL and then define a string arraylist. Thereafter the inputted command will be relayed to the database and all the pulled strings from the database will be put one by one into the arraylist. In the end the arraylist is returned by the *dataMulti* method. This can be seen in Figure 54 below.



```
public static List<String> dataMulti(String Command) {


    Connection conn = null;
    List<String> employees = new ArrayList<String>();

    try {
        conn = DriverManager.getConnection(DBsURL);

        Statement stmt = null;
        ResultSet rs1 = null;


        try {
            stmt = conn.createStatement();
            rs1 = stmt.executeQuery(Command);
            while (rs1.next()) {
                employees.add(rs1.getString(2));
            }
```

Figure 54 - ServerConnect Class, dataMulti Method

5.5.2.2 User Side

The user side of the system works in a very similar way as the administrator side. The user side of system also has three steps, login, choice of service provider and the login to the service provider.
The first login step works in exactly the same was as for the administrator side. The choice of service provider works in a similar way as the choice of permissions for the administrator side, with the change that the toggle revoke and grant permission is changed for the login to service provider. Authenticating to service providers was done with a custom authenticator available in WSO2 to authenticate via formfill which is explained in the coming sections.

5.5.2.3 WSO2 Formfill

Formfill was implemented by creating a custom authenticator within IdS's source code, so ID-Connect easily could choose between federated authenticators or formfill when authenticating employees. This was done with *Authenticator API* available from WSO2's source code, which is a Java *interface* containing required methods to authenticate a user with a custom authenticator. The class diagram for *ApplicationAuthenticator* along with the classes used for formfill. is illustrated on Figure 55.

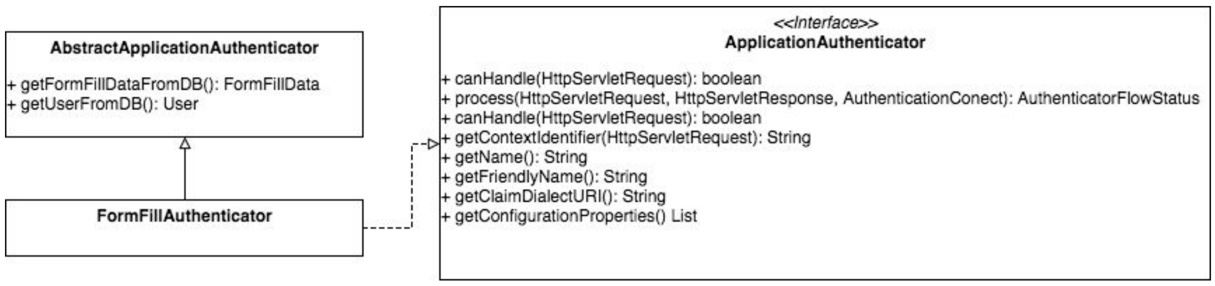

AbstractApplicationAuthenticator
+ getFormFillDataFromDB(): FormFillData
+ getUserFromDB(): User
FormFillAuthenticator
<<Interface>>
ApplicationAuthenticator
+ canHandle(HttpServletRequest): boolean
+ process(HttpServletRequest, HttpServletResponse, AuthenticationConect): AuthenticatorFlowStatus
+ canHandle(HttpServletRequest): boolean
+ getContextIdentifier(HttpServletRequest): String
+ getName(): String
+ getFriendlyName(): String
+ getClaimDialectURI(): String
+ getConfigurationProperties() List

Figure 56 - Class diagram of formfill implementation.

The methods in the *interface* are described below.

<u>canHandle</u>
This method validated whether the authentication mechanism was a formfill or another authenticator.

<u>process</u>
The purpose of *process* method was to fetch the formfill and its text-fields from a webpage. *process* establishes a connection to the server WSO2 is stored, which was *localhost* for this project. After the connection the formfill mechanism could be used to fetch the formfill from the web page and insert user-data in the form. To implement formfill in Java, a library from *Gargoyle Software Inc.* called *HtmlUnit*, was imported due to its capabilities to easily access a formfill on a website without developing comprehensive boilerplate code (Gargoyle Software Inc., 2017). Figure 57 illustrates the relevant code-snippet of how the formfill was implemented.

```
FormFillData formData = getFormFillDataFromDB();
User user = getUserFromDB();

String callbackUrl = IdentityUtil.getServerURL();
final HtmlPage page1 = webClient.getPage(callbackUrl);
final HtmlForm form = page1.getFormByName(formData.getFormID());

final HtmlTextInput formItemFirstName = form.getInputByName(formData.getFirstName());
final HtmlTextInput formItemLastName = form.getInputByName(formData.getLastName());
final HtmlTextInput formItemGmailAddress = form.getInputByName(formData.getEmail(););
final HtmlTextInput formItemPasswd = form.getInputByName(user.getPassword());
final HtmlTextInput formItemPasswdAgain = form.getInputByName(user.getPassword());

formItems = new ArrayList<>();
formItems.add(formItemFirstName);
formItems.add(formItemLastName);
formItems.add(formItemGmailAddress);
formItems.add(formItemPasswd);
formItems.add(formItemPasswdAgain);
```

Figure 57 - Code-snippet of formfill in Java in the method called process.

First the IDs of each item in the formfill were loaded from the table *FormFill* and inserted into the object *FormFillData*. The specific user's information were also loaded from storage and inserted into *User* object. Then the HTML page, where the formfill was located, was loaded from a URL retrieved from the *ServiceProvider* table, The actual form could then be located by its ID with help of *HtmlForm* object. This object contained a method called *getInputByName,* which accepts an argument of String that would be an item's ID. The formfill could now be sent to *getConfigurationProperties* methods, which would insert correct data into the formfill.

<u>getContextIdentifier</u>

Figure 58 illustrates a code snippet of data inserted into the formfill.

```
@Override
public List<Property> getConfigurationProperties() {
    List configProperties = new ArrayList();

    Property clientId = new Property();
    clientId.setName(FormfillAuthenticatorConstants.CLIENT_ID);
    clientId.setDisplayName("Client Id");
    clientId.setRequired(true);
    clientId.setDescription("Enter formfill identifier");
    configProperties.add(clientId);

    itemFirstName.setValueAttribute(clientId.getFirstName());
    itemLastName.setValueAttribute(clientId.getLastName());
    itemGmailAddress.setValueAttribute(clientId.getEmail());
    itemPasswd.setValueAttribute(clientId.getPassword());
    itemPasswdAgain.setValueAttribute(clientId.getPassword());
    webClient.close();
    return configProperties;
}
```

Figure 58 - Code-snippet to insert data to a formfill in getContextIdentifier.

Property is holds the authenticator's parameters which can be defined through the *Management Console* in IdS, but was hardcoded due to limited time of this project. The *setName* defines the type of authenticator to manipulate the data from the parameters. This data could therefore be inserted into the formfill via the *setValueAttribute* method accepting a parameter of String, which was retrieved from *Property*. The data was now inserted into the formfill and returned from IdP to ID-Connect.

5.5.3 Unit Testing

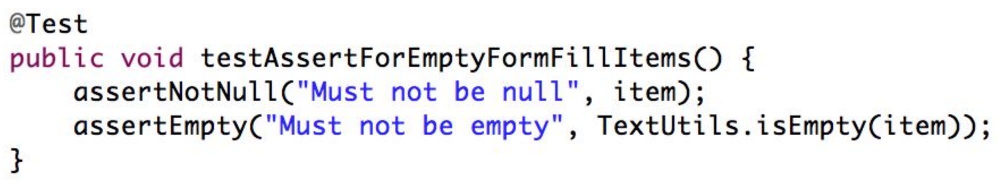
For this project the UI of the system was tested with unit testing with the tool called *JUnit*. The test ensured that no unexpected results occurred when loading data and displaying it on the UI. The testing was done while developing the features to limit the amount of bugs and errors in the Java application. The following sub sections explains the types of testing was done for the system.

5.5.3.1 Checking for null values

If a client-side application sends a request to a malfunctional server or database, the response could return null-values. If the null-checking is handled incorrectly on the client-side, Java will throw a *NullpointerExeption* and cause the system to crash. Some may argue that practically it is overrated to test for null-values for code-snippets that never would return null. This might be true if the code is never changed afterwards, but future updates might require change in the code-base which could break the guarantee of a value never being null.

5.5.3.2 Testing FormFill

Unexpected errors might occur of *FormFill* does not completely fill out the form. Therefore the group members tested if there were any blank items in the requested form. Figure 59 illustrates a code-snippet that tests if the form-items are blank or not.



```
@Test
public void testAssertForEmptyFormFillItems() {
    assertNotNull("Must not be null", item);
    assertEmpty("Must not be empty", TextUtils.isEmpty(item));
}
```

Figure 59 - Test for non-empty FormFill-items.

The first method-call checks if the value inside the item is *null*. If so, *JUnit* prints the message, *Must not be null*, on the first parameter in the *assert* method in the console. This also applies for the second method-call which checks if the item is empty.

5.5.3.3 Testing UI builder

The UI for proof-of-concept was built using the standard Java *swing* library. To test the UI's validity the method *testAssertThatUIIsCorrectlyConstructed* was created which checks if all the items from an array, which holds the UI-components such as text fields and buttons, are added to the array. If they are added the UI is correctly displayed. A code-snippet of the method is displayed on Figure 60.

```java
@Test
public void testAssertThatUIIsCorrectlyConstructed() {
   assertThat(Arrays.asList("Username", "Password", "Login", "Register"),
   hasItems(panel.getAll()));
   ...
}
```

Figure 60 - Test for correctly build UI.

6 Discussion

This section focuses on the reflections of decisions made throughout this thesis. The reflections will mainly be focused on the proof of concept of the ID-Connect since the system was the core element of this project. Afterwards suggestions for future improvements of ID-Connect will be addressed and discussed along with further challenges.

As discovered in the 4 Analysis chapter, the SMEs, who would use ID-Connect, would be the biggest stakeholder, making ID-Connect depending on SMEs to use the system. However, the challenge of convincing SMEs requires tremendous amount of trust which they must give to ID-Connect since the system stores the employees passwords. If ID-Connect gets compromised by hackers it might cause damages to the SMEs. Especially availability is important since ID-Connect would be a reversed bottleneck system leading to all service providers. This means that all the service providers that might be essential for an SME to function is depending on one entry point. This also is one of the most crucial factors which needs to be addressed if the system was launched on the market.

6.1 Drawbacks

Since ID-Connect is built on top of SSO it is amplifying not only the positive parts of SSO but also its drawbacks. The main drawback is the single point of failure. If an attacker obtains the master password he/she has access to all the other service providers the user is authorized to use. ID-Connect's formfill function, might therefore be vulnerable to keyloggers or sniffing on the client-side where the master password is used. If the sniffing of passwords would happen anywhere else the sniffer would only be able to obtain the secondary password which is unique to each service provider.

Other drawbacks and dependencies are the heavy relying on IdS and IdPs to function in the most optimal way. This is due to the system mainly using IdS to authenticate users with IdPs to access services from service providers. Formfill removes some of the dependencies by filling out forms and mimic the SSO process instead of authenticating through IdPs.

Another drawback of ID-Connect is not only if misuse of master password occurs but also if the user forgets the password. Since the password is only stored as a hash digest in the database it will not be able to be recovered. If a user forgets its master password all the secondary passwords for the user become useless since they are encrypted with the master password, and the user has to enroll again. This problem is also common with password managers, and is often solved by giving the user the choice of backing up their master password locally or on the service's servers itself. This would therefore also work for ID-Connect and could be the possible solution for the problem.

The last drawback is the lack of hashing the secondary passwords when authenticating with formfill. As explained in 4.4 Sequence Diagram section and the 5 Proof of Concept chapter, the secondary passwords are only encrypted with the master password, due to the formfill function needing a plaintext password. This in theory means that secondary passwords could potential be decrypted, if both the database which stores the encrypted passwords would be compromised and the master password would be either guessed or gotten hold of. Since all the secondary passwords of one user would be encrypted with the master password as a key, this would give access to all of the secondary passwords of a user. The reasons for not being able to use hashing to store the secondary passwords as explained earlier is that the user itself does not know the secondary password since it is generated by ID-Connect.

6.2 Data Storage

Another consideration for the system that can be made is about the data that is stored. This will especially be relevant with the coming user data initiative, GDPR, which all companies in the EU handling user data must obey. The GDPR has therefore also been one of the aspects that have been kept in mind while creating the ID-Connect system. This is also the reason for trying to minimize the user data which is stored in the system. (Costa, 2015)

6.3 Future Improvements

Studying Innovative Communication Technologies and Entrepreneurship (ICTE) have provided the group members great knowledge to suggest improvements to the ID-Connect, which would be beneficial to be implemented in the future. This section therefore includes suggestions for future improvements to ID-Connect. Since the actual developed product is merely a proof of concept of the entire ID-Connect, there are quite a few things that could be improved upon.

6.3.1 Implementation of a Complete System

The first suggestion is to develop the entire system by including the delimitations described in section 1.6 Delimitations. This means the proof of concept would support registration of new users and include a GUI for the end-users. The registration of new users would benefit the end-users to be able to create users instead of the administrator of the ID-Connect system manually must create them by adding them into the database.

The GUI would benefit the group members by allowing them to do user-testing, validate the solution towards SMEs, and gain new requirements to improve the system. The current GUI was created with Java swing library providing a skeleton-GUI which might not be feasible for users to interact with. The user-testing would not only be interesting with only employees but also with enterprises, who might provide valuable feedback such as the demand of the product from the enterprises to the group members in order to improve the overall system. It would also be interesting to see how many employees from technical enterprises uses the same password or admits how strong their passwords are compared to employees from non-technical companies.

6.3.2 Expand the Target Audience

Another improvement would be expanding the user-base from SMEs to include private users - in particular families. A household would mimic an enterprise, a parent or spouse would be an administrator and the children employees. This could help the family with access management where the parent (administrator) would grant or deny permissions, authenticate and deauthenticate for their children to authorized service providers. Elderly people could also benefit from ID-Connect, where municipalities would be enterprises, nurses would be administrators and the elderly employees. The administrators would manage the authentication for the elderly since the elderly generally lack technological knowledge. However, there would be an issue of administrators authenticating on behalf of the elderly which requires their content, and this issue might add complexity to the system. Granting permission on behalf of other users or use their identity is not only up to the resource owner, but also the service providers such as online banking accounts or governmental applications like Borger.dk. Impersonating other users could potentially be abused where the service provider could be liable and therefore does not allow people completing tasks on the behalf of other people.

6.3.3 Multi-Factor Authentication

As stated in chapter 3 State of The Art, implementing multi-factor authentication when users authenticate with their master password would decrease the chance of attackers compromising the password, since the additional factors are required. The factors could be OTPs sent to the users email or via SMS on their phones, or be inspired by FIDO alliance which uses tokens such as biometric fingerprint, hardware tokens to authenticate users which was introduced in 1 Introduction chapter. Therefore, implementing multi-factor authentication would be a great future improvement.

6.3.4 Taking Advantage of WSO2's Full Potential

WSO2 IdS was used as the backbone of the authentication mechanism in ID-Connect. When an employee chooses a service from the list of service providers and authenticates itself, a request is sent to IdS. This request checks the type of outbound authentication to see if the service is using OpenID Connect, SAML etc. It then authenticates the user and sends a response back to ID-Connect. However, IdS offers many more features and the group members recognized the full potential of WSO2 and its products could be useful for future development of the system.

6.3.4.1 Management Console

The Management Console offers great ways for super administrators (those who own ID-Connect) to create users, define roles with role-based access control (RBAC) to control user’s authorization-level, define policies with standards like eXtensible Access Control Markup Language (XACML). Since WSO2 is open source and delivers the source code these functionalities can also be added to end users, so they can, under the hood, use RBAC, XACML etc. Configuring service providers to define local and outbound authentications are also possible for users via Management Console along with IdPs. IdS offers fine-grain access to which scopes an IdP can select when authenticating a user. This could be great if an enterprise uses a service provider, and think they have requires too many scope, the user can simply turn them off in the Management Console.

The Management Console would therefore be a great future asset for ID-Connect due to its mentioned functionalities.

6.3.4.2 Additional products

WSO2 offers additional open source products. Two products caught the group members attention namely App Manager and API Manager. App Manager is a part of IdS and offers a standalone solution to publish applications, which could be internal to an enterprise that requires authentication with the employees. The solution is similar to an App Store where an authorized user uploads an application and signs it digitally. Another user can then review the application before publishing allowing only valid applications. IdS interpretes these applications as service providers so the benefits of authentication mechanisms also applies to applications. (WSO2, 2017e)

API Manager offers great ways to implement APIs that could be integrated with service providers and applications. The authentication and authorization with the API are handled via IdS so if an employee has been given a specific role he/she can access the API depending on the role. This is great instead of developing the authorization from the ground-up. (WSO2, 2017f)

7 Conclusion

After the research was completed within the project's scope, it became clear that a system like ID-Connect, despite being a proof of concept with many rough edges and lacked functionalities, could become a feasible reality if the future improvements along with the points discussed in chapter 6 Discussion were implemented.
Especially the various similar solution, such as password managers, existing on the market, indicated an interest in such product as ID-Connect. This helped to draw the conclusion that ID-Connect could be a great asset to the market and thereby formed the thesis problem formulation:

How can a system be built that would potentially help SMEs securely managing its employees passwords, so the employees can authenticate or deauthenticate multiple services with one master password?

With the increasing rise of digital services requiring user-profiles, users have difficulties remembering all these passwords as stated in chapter 1 Introduction. To overcome this problem users either tend to create weak passwords that are easy-to-guess, or use the same password for multiple services. This might cause security risk for an enterprise for instance, who needs to protect its resources against cyber criminals.

A survey was made to validate if this problem also applied to employees. Surprisingly almost half of the respondents answered *Yes* to be using the same password for multiple services. This was not expected from employees who were expected to be more professional when working in SMEs and care more about security. Almost half of the respondents answered that the reason of using the same password is the difficulty of memorizing them which validates the research in the 1 Introduction chapter. Solutions to this could be password managers but almost half of the respondents did not have any knowledge or did not use password managers in their enterprises.

The group members investigated the currently existing password managers on the market, which arrived as client-based and browser-based. To collect requirements for ID-Connect, the name of this thesis' proof of concept, an investigation of how the managers stores the master password, which cryptographic algorithms were used to encrypt the master password and how the managers supported enterprises was made. The investigated client-based password managers supported enterprise-versions of their system for a fee where the browser-based did not support adding multiple user-profiles to manage their authentication, hence the group members decided to develop a client-based solution since it was easier to collect requirements from the client-based solutions.

NIST published a report about theories behind passwords and provided a list of how one could mitigate password-attacks. Some of the important points were implementing strong passwords that are long and complex, make use of multi-factor authentication and respect cryptographic standards verified by authorities like FIPS. ID-Connect respected most of the points NIST had like implementing the verified standards SHA-256 and AES-256 improving the integrity of the system, and requiring long, complex passwords.

By thoroughly investigating the workings of the various password managers and SSOs it was noticeable that there was a possibility for combining these two solutions, even though there were some issues with both of these solutions.

The password managers generally had their passwords on display in the case that a user had the master password, which meant that once the master password is compromised all of the passwords will be as well. The problem with SSO was that service provider must implement SSO in their services. This meant that it would require some implementation from the service provider side to make an SSO system work on their service. This in term meant that there are many webpages where SSO login is not an option due to the lack of implementation on the service provider side.

After identifying these shortcomings from the two technologies the focus was turned to creating a system which would provide the same result as the two without their shortcomings, which became the main contribution of ID-Connect. The shortcomings were solved by combining the two technologies having both an SSO way of authenticating and authenticating with a password vault for services without SSO using formfill. This solved the authentication-issue SSOs had with service providers lacking SSO and ID-Connect's dependency.

To solve the issue of having user's master password being compromised without the secondary passwords being retrievable, the ID-Connect system was created with a master password known by the user and a secondary password created by the system and unknown to all except the system. This meant that if the master password is compromised there can be some misuse by logging in, but the actual secondary password cannot be compromised unless the ID-Connect's database is also compromised. This also limits the aftermath of a potential misuse scenario, by the system only having to change the master password and none of the secondary ones.

Thus ID-Connect would be suitable to securely authenticate and manage employees passwords and solving the mentioned shortcomings and the problem formulation.

8 References

Ayushi (2010) *A Symmetric Key Cryptographic Algorithm.* Retrieved from: http://www.ijcaonline.org/journal/number15/pxc387502.pdf Accessed 22/05/2017.

Bellare M., Boldyreva A., Micali S. (2000) *Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements*. In: Preneel B. (eds) Advances in Cryptology — EUROCRYPT 2000. EUROCRYPT 2000. Lecture Notes in Computer Science, vol 1807. Springer, Berlin, Heidelberg
Retrieved from: https://link.springer.com/chapter/10.1007/3-540-45539-6_18 Accessed 22/5/2017.

CA Technologies (2014) *A How-to Guide to OAuth & API Security.* Retrieved from: https://www.moodle.aau.dk/pluginfile.php/650020/mod_folder/content/0/a-how-to-guide-to-oauth-and-api-security.pdf?forcedownload=1 Accessed 30/05/2017.

Costa L. (2015) *REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation)*. Retrieved from: http://data.consilium.europa.eu/doc/document/ST-9565-2015-INIT/en/pdf Accessed 06/06/2017.

Dashlane (2017a) *The password manager, perfected.* Retrieved from: https://www.dashlane.com/features/password-manager Accessed 13/05/2017.

Dashlane (2017b) *Dashlane speeds up the web.* Retrieved from: https://techcrunch.com/2012/04/16/dashlane-speeds-up-the-web-with-instant-logins-automatic-checkout-and-more/ Accessed 13/05/2017.

Dashlane (2017c) *Use Dashlane's Password Generator to create strong passwords.* Retrieved from: https://www.dashlane.com/features/password-generator Accessed 13/05/2017.

Dashlane (2017d) *Why Dashlane Business.* Retrieved from: https://support.dashlane.com/hc/en-us/articles/208865685-Why-Dashlane-Business Accessed 13/05/2017.

Dashline (2017e) *30 day free trial for up to 30 users.* Retrieved from: https://www.dashlane.com/it/business/pricing Accessed 13/05/2017.

Eurostar (2017) *Public employment - Denmark.* Retrieved from: http://ec.europa.eu/eurostat/statistics-explained/index.php/Public_employment_-_Denmark Accessed 14/05/2017.

Facebook (2017) *Permissions Reference.* Retrieved from: https://developers.facebook.com/docs/facebook-login/permissions/ Accessed 27/04/2017.

FireEye (2017) *Small and Midsize Businesses.* Retrieved from: https://www.fireeye.com/solutions/small-and-midsize-business.html Accessed 26/02/2017.

Gargoyle Software Inc. (2017) *HtmlUnit.* Retrieved from: http://htmlunit.sourceforge.net/ Accessed 31/05/2017.

Google (2017a) *Smart Lock security simplified.* Retrieved from: https://get.google.com/smartlock/ Accessed 13/05/2017.

Google (2017b) *Integrating Google Sign-In into your web app*. Retrieved from: https://developers.google.com/identity/sign-in/web/sign-in Accessed 03/06/2017.

Google (2017c) *Google Sign-In JavaScript client reference.* Retrieved from: https://developers.google.com/identity/sign-in/web/reference#googleusergetbasicprofile Accessed 03/06/2017.

Google Chrome Help (2017) *Set or change a sync passphrase.* Retrieved from: https://support.google.com/chrome/answer/1181035 Accessed 13/05/2017.

Google Cloud (2017) *Get Gmail, Docs, Drive, and Calendar for business.* Retrieved from: https://gsuite.google.com/intl/en_ie/ Accessed 13/05/2017.

Guccione (2017) *What the Most Common Passwords of 2016 List Reveals*. Retrieved from: https://blog.keepersecurity.com/2017/01/13/most-common-passwords-of-2016-research-study/. Accessed 26/05/2017.

Guttman B (1995). *An Introduction to Computer Security: The NIST Handbook.* Retrieved from: http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf Accessed 28/05/2017.

Hamill P. (2006) *Unit Testing Frameworks. Tools for High-Quality Software Development*. Retrieved from: https://books.google.dk/books?hl=en&lr=&id=2ksvdhhnWQsC&oi=fnd&pg=PT7&dq=**unit**+test+java&ots=AK6_-8XPF3&sig=un1TXRyoTSWpPmJj27-RuXy89Qo&redir_esc=y#v=onepage&q=unit%20test%20java&f=false Accessed 04/06/2017.

JUnit (2017) *Getting started.* Retrieved from: http://junit.org/junit4/ Accessed 04/06/2017.

Kasahara A., Miura A., S. Hiroshi, Ishida K (2006) *Login system and method.* Retrieved from: https://www.researchgate.net/profile/Patrick_Bours/publication/221567007_A_Login_System_Using_Mouse_Dynamics/links/00b7d5316d600bde10000000.pdf. Accessed 03/06/2017.

Kelly (2002) *Is Single Sign on a Security Risk.* Retrieved from: https://www.giac.org/paper/gsec/811/single-sign-security-risk/101711. Accessed 28/05/2017.

Khajuria S. (2014) *Network and Security (NAS) Lecture 1*. Retrieved from: https://www.moodle.aau.dk/pluginfile.php/370856/mod_folder/content/0/NAS_SKH01_030914.pdf?forcedownload=1 Accessed 28/05/2017.

Khajuria S. (2016) *Identity and Access management (IAM) Lecture 6 ELECTRONIC AUTHENTICATION*. Retrieved from: https://www.moodle.aau.dk/pluginfile.php/650019/mod_folder/content/0/Lecture%206%2C%20SKH%20080316.pdf?forcedownload=1 Accessed 03/06/2017.

Khajuria S. (2017) *Lecture 3, SKH 210217*. Retrieved from: https://www.moodle.aau.dk/pluginfile.php/923559/mod_folder/content/0/Lecture%203%2C%20SKH%20210217.pptx?forcedownload=1 Accessed 28/05/2017.

LastPass (2017a) *How we do it.* Retrieved from: https://lastpass.com/whylastpass_technology.php Accessed 25/05/2017.

LastPass (2017b) *Take LastPass to work.* Retrieved from: https://www.lastpass.com/business Accessed 13/05/2017 Accessed 25/05/2017.

Malalgoda S. (2016) *Customize JSON Web Token Generation with WSO2 API Manager.* Retrieved from: http://wso2.com/library/articles/2014/12/customize-json-web-token-generation-with-wso2-api-manager-1.8.0/ Accessed 04/06/2017.

Mander J. (2015) *Internet users have average of 5.54 social media accounts.* Retrieved from: https://www.globalwebindex.net/blog/internet-users-have-average-of-5-social-media-accounts Accessed 26/02/2017.

Microsoft (2017) *Active Directory Architecture.* Retrieved from: https://msdn.microsoft.com/en-us/library/bb727030.aspx#EFAA Accessed 26/02/2017.

Ofcom (2016) *Adults' media use and attitudes*. Retrieved from: https://www.ofcom.org.uk/__data/assets/pdf_file/0026/80828/2016-adults-media-use-and-attitudes.pdf Accessed 26/02/2017.

Olesen H. (2016a) *Identity and Access Management (IAM) Lecture 7: OAuth 2.0 and OpenID Connect*. Retrieved from: https://www.moodle.aau.dk/pluginfile.php/650020/mod_folder/content/0/Lecture%207%2C%20290316.pdf?forcedownload=1 Accessed 01/06/2017.

Olesen H. (2016b) *Identity and Access Management (IAM) Lecture 5: Identity Management Systems*. Retrieved from: https://www.moodle.aau.dk/pluginfile.php/650018/mod_folder/content/0/Lecture%205%2C%20010316.pdf?forcedownload=1 Accessed 04/06/2017

Olesen H. (2017) *Identity and Access Management (IAM) Lecture 7: OpenID Connect & User Managed Access*. Retrieved from: https://www.moodle.aau.dk/pluginfile.php/923564/mod_folder/content/0/Lecture%208%2C%20OpenID%20Connect%20%20User%20Managed%20Access%2C%20280317.pdf?forcedownload=1 Accessed 01/06/2017.

OneLogin (2017) *Single Sign-On (SSO) for Active Directory.* Retrieved from: https://www.onelogin.com/active-directory-sso Accessed 28/05/2017.

ORACLE (2015) *Environment Variables (The Java Tutorial)*. Retrieved from: http://docs.oracle.com/javase/tutorial/essential/environment/env.html Accessed 31/05/2017.

Paradigm (2017) *Metadata for authenticity: hash functions and digital signatures.* Retrieved from: http://www.paradigm.ac.uk/workbook/metadata/authenticity-fixity.html Accessed 06/06/2017.

Ritzau (2016) *Danske personoplysninger er billigt til salg på nettet*. Retrieved from: http://jyllands-posten.dk/livsstil/digitalt/ECE8340239/Danske-personoplysninger-er-billigt-til-salg-p%C3%A5-nettet/ Accessed 06/03/2017.

Scarfone K., Souppaya M. (2009) *Guide to Enterprise Password Management.* Retrieved from: http://www.tier3md.com/media/800-118.pdf Accessed 25/05/2017.

Schwaber K., Beedle M. (2002) *Agile software development with Scrum.* Retrieved from: http://dbmanagement.info/Books/MIX/Agile_Project_Management_With_Scrum.pdf Accessed 01/06/2017.

Secher M. (2017) *Mand anholdt for stort YouSee-nedbrud nytårsaften*. Retrieved from: http://nyheder.tv2.dk/krimi/2017-01-05-mand-anholdt-for-stort-yousee-nedbrud-nytaarsaften Accessed 04/06/2017.

Seshu V. (2013) *Single Sign On using SAML*. Retrieved from: https://blog.imaginea.com/single-sign-on-using-saml/ Accessed 02/06/2017

Silva J. (2003) *An Overview of Cryptographic Hash Functions and Their Uses.* Retrieved from: https://www.sans.org/reading-room/whitepapers/vpns/overview-cryptographic-hash-functions-879 Accessed 23/05/2017.

Sørensen L. (2016) *Introduction Ideas and Problem formulation. Lecture 1.* Retrieved from: https://www.moodle.aau.dk/pluginfile.php/845373/mod_resource/content/1/PBL%20intro%20andProblem%20formulation%202016.pdf Accessed 03/06/2017.

Trustpilot (2017) *Dashlane reviews.* Retrieved from: https://www.trustpilot.com/review/www.dashlane.com Accessed 13/05/2017.

Twitter (2017) *Implementing Sign in with Twitter.* Retrieved from: https://dev.twitter.com/web/sign-in/implementing Accessed 30/04/2017.

Virk (2017) *Overblik over virksomhedsformer.* Retrieved from: https://startvaekst.virk.dk/opstart/vaelg-virksomhedsform/overblik-over-virksomhedsformer Accessed 31/5/2017.

WSO2 (2017a) *WSO2 Identity Server Documentation.* Retrieved from: https://docs.wso2.com/display/IS530/Overview Accessed 19/03/2017.

WSO2 (2017b) *WSO2 Capabilities.* Retrieved from: http://wso2.com/identity-and-access-management Accessed 19/03/2017.

WSO2 (2017c) *WSO2 App Manager Documentation.* Retrieved from: https://docs.wso2.com/display/APPM120/Introducing+App+Manager Accessed 19/3/2017.

WSO2 (2017d) *WSO2 App Manager Documentation 2.* Retrieved from: https://docs.wso2.com/display/IS530/WSO2+Identity+Server+Documentation Accessed 19/03/2017.

WSO2 (2017e) *WSO2 App Manager*. Retrieved from:
http://wso2.com/products/app-manager/ Accessed 05/06/2017.

WSO2 (2017f) *WSO2 Api Manager Documentation*. Retrieved from:
https://docs.wso2.com/display/AM210/WSO2+API+Manager+Documentation Accessed 05/06/2017.

9 Abbreviations

Term	Usage
AD	Active Directory
AES	Advanced Encryption Standard
API	Application Programming Interface
ApS	Anpartsselskab
A/S	Aktieselskab
CIA	Confidentiality, Integrity, Availability
CSO	Chief Security Officer
FIPS	Federal Information Processing Standards
FTP	File Transfer Protocol
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTTP	Hypertext Transport Protocol
HTTPS	Hypertext Transport Protocol Secure
ICTE	Innovative Communication Technologies and Entrepreneurship
IdP	Identity Provider
IdS	Identity Server
IS	Interessant selskab (Danish)
IVS	Iværksætterselskab (Danish)
NIS	Norton Identity Safe
NIST	National Institute of Standards and Technologies
OS	Operating System
OTP	One-Time Password

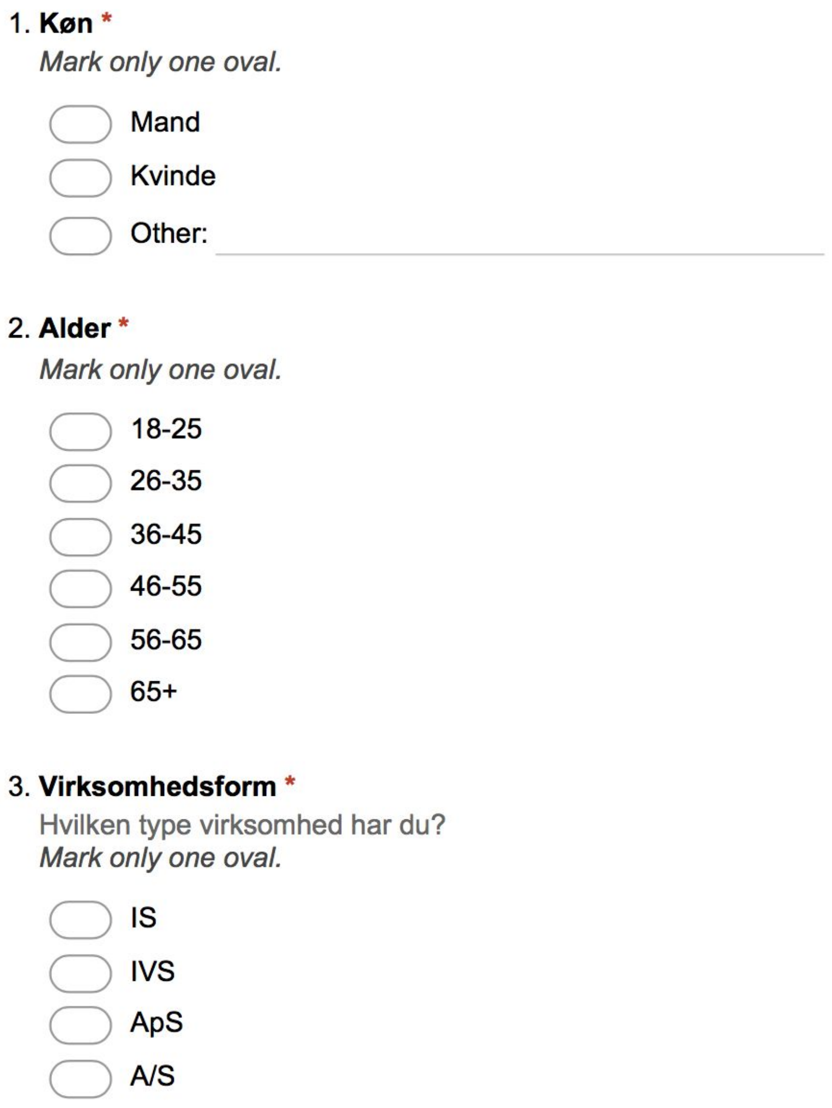
PBKDF2	Password-Based Key Derivation Function
PIN	Personal Identity Number
RBAC	Role-Based Access Control
REST	Representational State Transfer
SME	Small-medium size business
SSO	Single-Sign-On
TDC	Tele Danmark Communications
TLS	Transport Layer Security
VPN	Virtual Private Network
XACML	eXtensible Access Control Markup Language

10 Appendix

10.1 Appendix A

Den gennemsnitlige internetbruger er aktiv på mere end 5 forskellige sociale medier, bruger flere forskellige emails og er registreret på mange online services (fx webshops). Disse services kræver oftest lange og komplicerede adgangskoder, som kan være svære at huske, hvilket resulterer i at brugere enten anvender den samme adgangskode til flere forskellige services, eller bruger mange ikke kompliceret adgangskoder fx 123456. Dette gør det nemt for ITkriminelle at hacke sig ind på fx en brugers bank konto, og kan forvolde stor skade.

Dette spørgeskema vil undersøge omfanget af dette problem i danske små og mellemstore virksomheder.



1. **Køn** *
Mark only one oval.

- ◯ Mand
- ◯ Kvinde
- ◯ Other: ____________

2. **Alder** *
Mark only one oval.

- ◯ 18-25
- ◯ 26-35
- ◯ 36-45
- ◯ 46-55
- ◯ 56-65
- ◯ 65+

3. **Virksomhedsform** *
Hvilken type virksomhed har du?
Mark only one oval.

- ◯ IS
- ◯ IVS
- ◯ ApS
- ◯ A/S

4. **Hvad beskæftiger jeres virksomhed med?** *

Mark only one oval.

- ◯ Byggeri og håndværk
- ◯ Design og kunst
- ◯ Film, teater og musik
- ◯ Forretning, handel og markedsføring
- ◯ IT (hardware)
- ◯ IT (software - herunder webudvikling, appudvikling etc.)
- ◯ Pædagogik
- ◯ Salg
- ◯ Sundhed, pleje og idræt
- ◯ Transport og logistik
- ◯ Undervisning og forskning
- ◯ Other: ____________________

5. **Hvor mange ansatte er der i virksomheden?** *

Ca.

Mark only one oval.

- ◯ 1-5
- ◯ 6-10
- ◯ 11-20
- ◯ 21-35
- ◯ 36-50
- ◯ 50-75
- ◯ 75-100
- ◯ 100+

6. **Overordnet set, hvor stærk/kompleks vil du vurdere dine adgangskoder er?** *

Fx. en stærk adgangskode skal være min. 8 tegn langt, indeholde både store og små bogstaver, tal og special tegn

Mark only one oval.

	1	2	3	4	5	
Meget svag	◯	◯	◯	◯	◯	Meget stærk

7. **Bruger du den samme adgangskode til flere forskellige services fx samme kode til Netflix og til Borger Service?** *

Mark only one oval.

- ◯ Ja
- ◯ Nej

8. **Hvad er grunden til at anvende den/de samme adgangskode(r)?** *
Mark only one oval.

◯ Jeg bruger aldrig den samme adgangskode til forskellige services
◯ Det er svært at huske mange stærke/komplekse koder
◯ Min adgangskode er meget stærk. Folk vil alligevel aldrig kunne gætte/hacke sig frem til den.
◯ Jeg tænker ikke over, at jeg bruger den samme adgangskode til forskellige services
◯ Other: ______________________

9. **Har I lagt et budget for jeres virksomheds IT sikkerhed?** *
Mark only one oval.

◯ Ja
◯ Nej

10. **Hvem er IT-sikkerhedsansvarlig for jeres virksomhed?** *
Mark only one oval.

◯ Ledelsen (CEO, COO...)
◯ IT afdelingen
◯ Den enkelte medarbejder
◯ Ingen
◯ Other: ______________________

11. **Hvordan holder I styr på jeres egne og medarbejdernes adgangskoder i virksomheden?** *
Mark only one oval.

◯ Vi har adgangskoderne i mailkorrespondancer
◯ Skrevet dem ned på et fysisk stykke papir
◯ Skrevet dem ned i et Word dokument eller lign. tekstprogram
◯ Medarbejderne har ansvaret for deres egne adgangskoder
◯ Vi har ikke et system til at holde styr på adgangskoderne
◯ Other: ______________________

12. **Hvis jeres virksomhed skulle oprette en Microsoft Office 365-bruger til jeres nyansatte medarbejder, hvordan vil processen foregå?** *
Mark only one oval.

- ◯ Den IT-sikkerhedsansvarlige opretter en bruger, og afleverer brugernavn og adgangskode til medarbejderen på et stykke papir.
- ◯ Den IT-sikkerhedsansvarlige opretter en bruger, og sender brugernavn og adgangskode til medarbejderens email.
- ◯ Den IT-sikkerhedsansvarlige opretter en bruger, og sender et link til medarbejderen så vedkommende selv vælger sin adgangskode.
- ◯ Den IT-sikkerhedsansvarlige opretter en bruger, men medarbejderen indtaster selv brugernavn og adgangskode.
- ◯ Medarbejderen opretter sig selv, og holder selv styr på sit brugernavn og adgangskode.
- ◯ Medarbejderen opretter sig selv, men sender en kopi af brugernavn og adgangskode.
- ◯ Other: ____________

13. **Hvis en medarbejder stopper, hvordan fjerner I vedkommende fra de tilmeldte services?**
Fx Office 365, email konti, Photoshop licenser, Dropbox etc,
Mark only one oval.

- ◯ IT-sikkerhedsansvarlige afmelder samtlige services manuelt
- ◯ IT-sikkerhedsansvarlige afmelder samtlige services via et program/applikation
- ◯ Ved det ikke
- ◯ Other: ____________

14. **Ved du hvad en Password Manager er?** *
Mark only one oval.

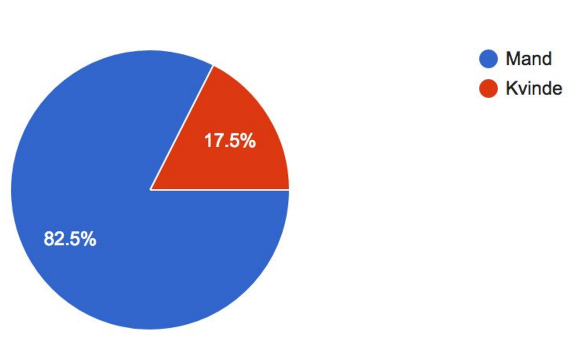
- ◯ Ja
- ◯ Nej

15. **Hvilken password manager har du/I brugt?** *
Tick all that apply.

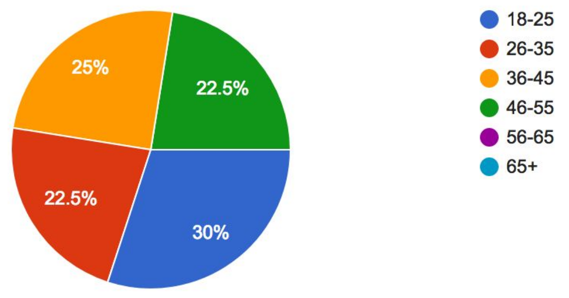
- ☐ Jeg kender/bruger ikke password managers.
- ☐ LastPass
- ☐ DashLane
- ☐ Google Chrome Smart Lock
- ☐ Safari Keychain
- ☐ Firefox
- ☐ Okta
- ☐ OneLogin
- ☐ Logmeonce
- ☐ Ping Identity
- ☐ Other: ____________

10.2 Appendix B

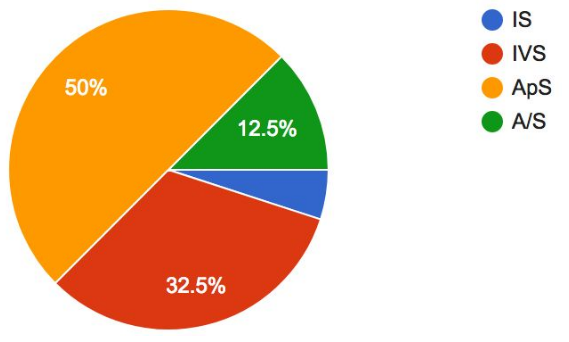
Køn



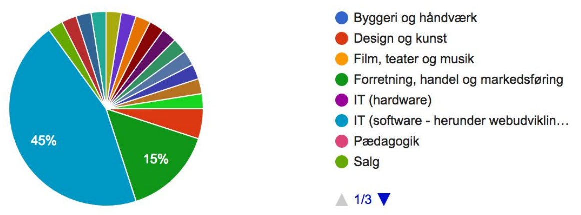
Alder



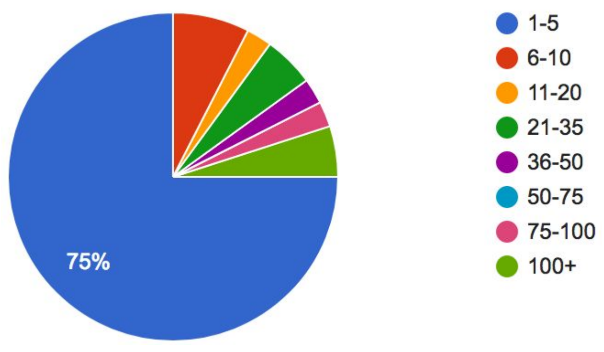
Virksomhedsform



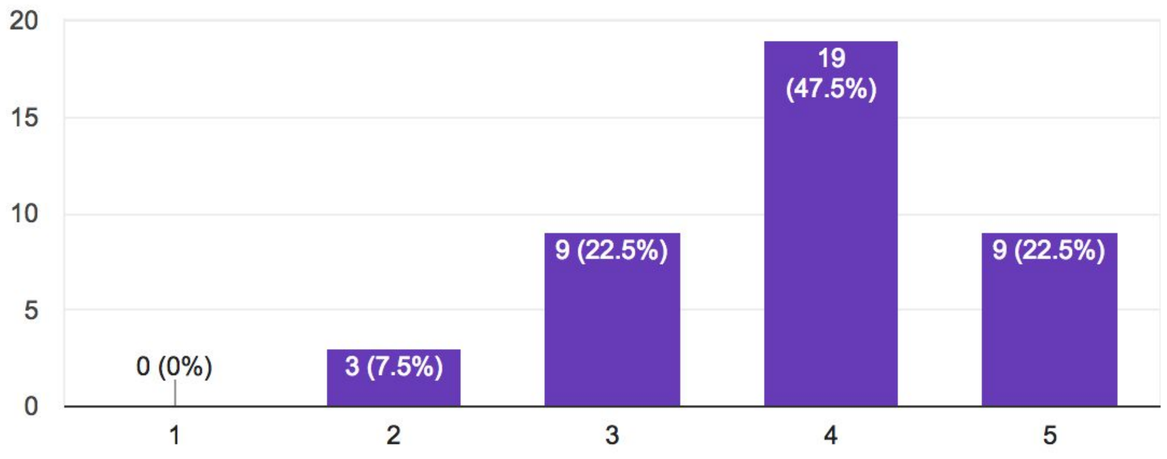
Hvad beskæftiger jeres virksomhed med?



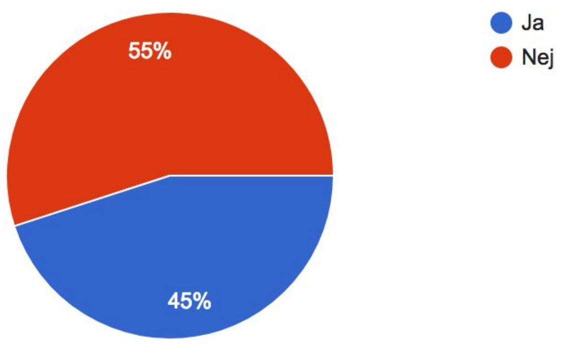
Hvor mange ansatte er der i virksomheden?



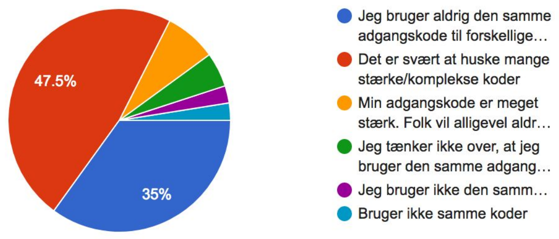
Overordnet set, hvor stærk/kompleks vil du vurdere dine adgangskoder er?



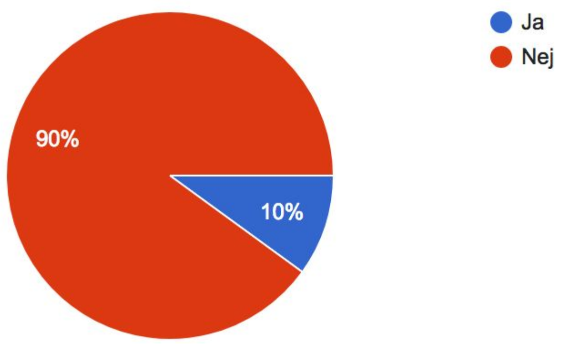
Bruger du den samme adgangskode til flere forskellige services fx samme kode til Netflix og til Borgerservice?



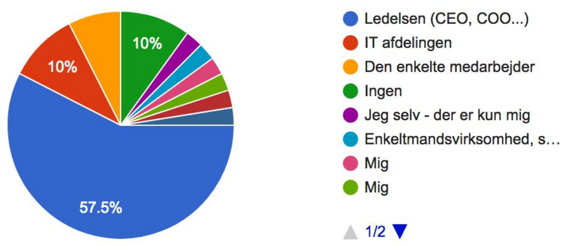
Hvad er grunden til at anvende den/de samme adgangskode(r)?



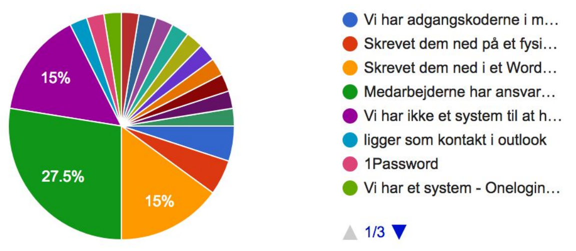
Har I lagt et budget for jeres virksomheds IT sikkerhed?



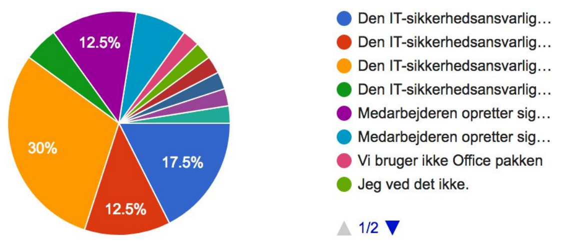
Hvem er IT-sikkerhedsansvarlig for jeres virksomhed?



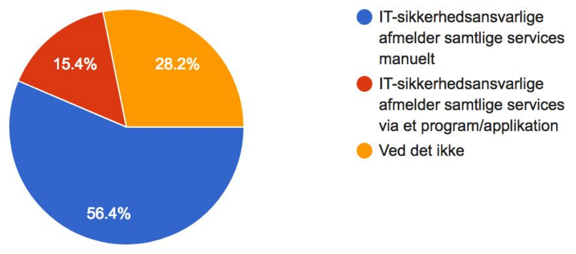
Hvordan holder I styr på jeres egne og medarbejdernes adgangskoder i virksomheden?



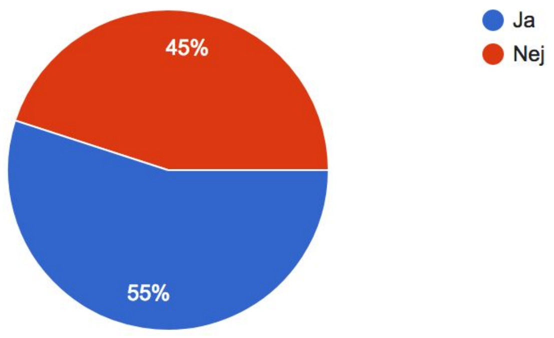
Hvis jeres virksomhed skulle oprette en Microsoft Office 365-bruger til jeres nyansatte medarbejder, hvordan vil processen foregå?



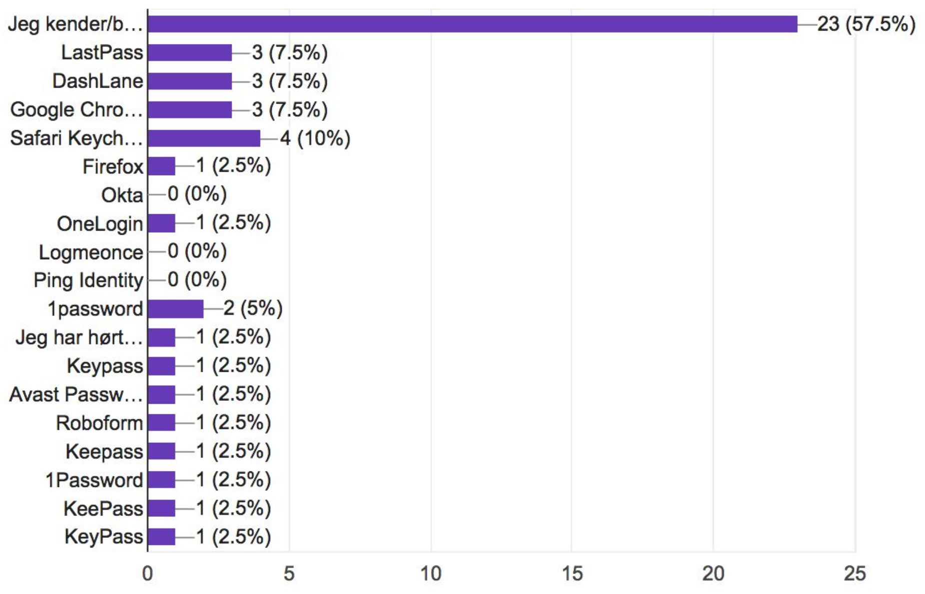
Hvis en medarbejder stopper, hvordan fjerner I vedkommende fra de tilmeldte services?



Ved du hvad en Password Manager er?



Hvilken password manager har du/I brugt?


Jeg kender/b… 23 (57.5%)
LastPass 3 (7.5%)
DashLane 3 (7.5%)
Google Chro… 3 (7.5%)
Safari Keych… 4 (10%)
Firefox 1 (2.5%)
Okta 0 (0%)
OneLogin 1 (2.5%)
Logmeonce 0 (0%)
Ping Identity 0 (0%)
1password 2 (5%)
Jeg har hørt… 1 (2.5%)
Keypass 1 (2.5%)
Avast Passw… 1 (2.5%)
Roboform 1 (2.5%)
Keepass 1 (2.5%)
1Password 1 (2.5%)
KeePass 1 (2.5%)
KeyPass 1 (2.5%)