

Henrik Granlund  
Herman Torland  
Sondre Gunneng  
Øyvind Wold

# RFC: A user-friendly system for managing and automating network changes in a secure zone model

Bachelor's thesis in Digital Infrastructure and Cyber Security  
Supervisor: Jia-Chun Lin  
May 2022



Henrik Granlund  
Herman Torland  
Sondre Gunneng  
Øyvind Wold

# **RFC: A user-friendly system for managing and automating network changes in a secure zone model**

Bachelor's thesis in Digital Infrastructure and Cyber Security  
Supervisor: Jia-Chun Lin  
May 2022

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Dept. of Information Security and Communication Technology





---

## Acknowledgements

We would like to thank Intility for providing us with this assignment and arranging assistance and guidance throughout the project. A special thanks go to Tobias Bryhn-Bjølgerud, Marius Engan, and the rest of the *Network Services Automation* department for the continuous support and sharing of knowledge. We would also like to thank our supervisor at NTNU, Jia-Chun Lin, who has been incredibly helpful and enthusiastic in guiding us through the project phase.

---

## Abstract

Intility is a fully managed platform service used by more than 600 companies. The exponential growth of this platform makes scalability a necessity to keep up with the increasing customer base. Intility strives to automate and streamline as many processes as possible to achieve seamless scalability. Due to their multi-domain secure zone model, this requires special considerations for data flow. This thesis will showcase the development of a web application that allows Intility's customers to submit requests to make changes to their networks. These requests contain sufficient information in a standardized format that allows for automating the network changes across the secure zones. The system will release capacity from technicians and customer support, allowing them to focus on more resource-intensive tasks. The thesis will cover both technical details about the developed system and outline the development process throughout the project.

---

## Sammendrag

Intility er en ende-til-ende plattformtjeneste som benyttes av mer enn 600 selskaper. Den eksponentielle veksten til denne plattformen nødvendiggjør skalerbarhet for å kunne holde tritt med den økende kundebasen. For å oppnå sømløs skalerbarhet, streber Intility etter å automatisere og effektivisere så mange prosesser som mulig. På grunn av deres sikker-sone modell med flere domener, kreves det spesielle tiltak med tanke på dataflyt. Denne rapporten vil vise utviklingen av en applikasjon som lar Intilitys kunder opprette forespørsler om nettverksendringer. Disse forespørslene inneholder tilstrekkelig informasjon i et standardisert format, noe som gjør det mulig å automatisere nettverksendringen på tvers av de sikre sonene. Systemet vil frigjøre kapasitet fra teknikere og kundestøtte, og la de fokusere på mer ressurskrevende oppgaver. Rapporten vil både dekke tekniske detaljer om det utviklede systemet, samt vise utviklingsprosessen gjennom hele prosjektet.

---

# Table of Contents

<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Code Snippets</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background about Intility . . . . .	1
1.2 High-level problem description . . . . .	1
1.3 Target Audience . . . . .	2
1.4 Project organization . . . . .	3
1.5 Thesis structure . . . . .	5
<b>2 Background</b>	<b>6</b>
2.1 Detailed problem description . . . . .	6
2.2 Purpose . . . . .	7
2.3 Tool survey . . . . .	8
2.3.1 Bifrost . . . . .	8
2.3.2 Database . . . . .	8
2.3.3 Docker . . . . .	9
2.3.4 FastAPI . . . . .	10
2.3.5 Gitlab and git . . . . .	10
2.3.6 JavaScript . . . . .	11
2.3.7 Kong . . . . .	12
2.3.8 Lucidchart . . . . .	12
2.3.9 Microsoft 365 . . . . .	13
2.3.10 OPA . . . . .	13
2.3.11 PyCharm . . . . .	13
2.3.12 React . . . . .	13



---

<b>3</b>	<b>Requirements specification</b>	<b>14</b>
3.1	Functional requirements . . . . .	14
3.2	Non-functional requirements . . . . .	15
<b>4</b>	<b>System design</b>	<b>16</b>
4.1	Architecture of RFC . . . . .	16
4.2	Features . . . . .	19
4.2.1	WLAN changes . . . . .	19
4.2.2	Add a new device to MAC Authentication Bypass (MAB) .	21
4.3	User interface design . . . . .	22
<b>5</b>	<b>Implementation</b>	<b>26</b>
5.1	Backend architecture . . . . .	26
5.2	API . . . . .	28
5.3	Frontend . . . . .	33
<b>6</b>	<b>Development process</b>	<b>36</b>
6.1	Process framework . . . . .	36
6.2	Documentation . . . . .	37
<b>7</b>	<b>Evaluation</b>	<b>39</b>
7.1	Evaluation of functional requirements . . . . .	39
7.2	Evaluation of non-functional requirements . . . . .	42
7.3	Implemented changes . . . . .	52
<b>8</b>	<b>Closing remarks</b>	<b>54</b>
8.1	Future work . . . . .	54
8.2	Learning outcome . . . . .	55
8.3	Evaluation of the project . . . . .	56
8.4	Conclusion . . . . .	57

---

<b>Bibliography</b>	<b>58</b>
<b>A Vocabulary</b>	<b>62</b>
<b>B Log</b>	<b>63</b>
B.1 Week log . . . . .	63
B.2 Timesheet . . . . .	65
B.3 Minutes of meetings . . . . .	68
<b>C Assignment description from Intility</b>	<b>71</b>
<b>D Project plan</b>	<b>73</b>
<b>E Project agreement</b>	<b>86</b>

---

## List of Figures

1	The current request for change flow at Intility . . . . .	6
2	The desired request for change flow . . . . .	7
3	Virtual machines vs containers . . . . .	9
4	Pre-commit results with failed test. . . . .	11
5	Metro vs Kong . . . . .	12
6	System architecture of RFC . . . . .	17
7	Technician interface for approving requests . . . . .	18
8	Conceptual database model . . . . .	19
9	WLAN modal window with collapsed change SSID/Password forms.	20
10	WLAN modal window with opened change SSID/Password forms.	20
11	MAB and ISE . . . . .	22
12	Theme toggle . . . . .	23
13	Customer landing page with different themes . . . . .	23
14	Animated skeleton while retrieving WLAN information from the backend . . . . .	24
15	Tooltip explaining SSID requirements . . . . .	24
16	User feedback for use of invalid SSID format . . . . .	24
17	User feedback for too short password . . . . .	25
18	Form for adding a new device with forced order of input and MAC address sanitization . . . . .	25
19	Flow of data through Kong with plugins . . . . .	27
20	The chosen process framework throughout the project . . . . .	37
21	Thesis workbook in Excel. . . . .	37
22	Utilization of docstring for PyCharm tooltip. . . . .	38
23	Auto-documented endpoints in Swagger . . . . .	40
24	Pre-commit checking backend code. . . . .	41
25	OWASP guidelines for access control. . . . .	42
26	Authentication and authorization flow. . . . .	43
27	<i>Guest WiFi settings</i> interface on different device types. . . . .	50
28	Redesigned technician interface containing company name . . . . .	53
29	Interface for making a request on behalf of a customer . . . . .	53

---

## List of Tables

1	Elapsed time for each technician task in whole seconds . . . . .	45
2	Number of button clicks for each technician task . . . . .	45
3	Elapsed time for each customer task in whole seconds . . . . .	45
4	Number of button clicks for each customer task . . . . .	46
5	Loading time in seconds to submit and approve a request. . . . .	51

---

## List of Code Snippets

1	OPA configuration for access to different endpoints based on roles	27
2	Endpoint that returns all requests from the database . . . . .	28
3	Class with netmiko config . . . . .	29
4	Method for changing WLAN password . . . . .	30
5	Method for changing WLAN SSID . . . . .	32
6	React component for the feature card in the menu . . . . .	33
7	Costumer landing page with usage of the FeatureCard component.	34
8	Hook for fetching all requests from the backend . . . . .	35

---

# 1 Introduction

This chapter will introduce the assignment from Intility and their problem area on a high level. In addition, the target audience, project organization, and structure of the thesis will also be presented.

## 1.1 Background about Intility

Intility [1] is a Norwegian company that offers an end-to-end IT platform for their customers, which are companies of various sizes. The company was founded in the year 2000 and has about 500 full-time employees spread across four offices. Intility's business model is managing their customers' IT stack, taking full responsibility for their network infrastructure, client devices, monitoring, security, error handling, and support. As of May 2022, Intility manages over 600 customers at over 2000 locations, which demands that they have an organized and secure architecture. This project was conducted for the Network Services Automation department at Intility, which is a team of DevOps engineers that works with automating network-related tasks.

## 1.2 High-level problem description

Intility divides all their customers into isolated secure zones. Each zone consists of its own physical network that contains the customer's equipment at their locations, as well as virtual networks to run their services in data centers. By doing this, Intility ensures that an issue in one zone does not affect the other zones. Additionally, for Intility to make changes to services on behalf of their customers, they need to use a virtual machine running on a secured workstation that is configured to follow their internal guidelines.

Today, whenever customers want to change their network settings, they need to contact Intility to create a support ticket. This ticket will be forwarded to a technician, who in many cases will need to log onto the networking equipment and perform the changes manually through the command line. This time-consuming and repetitive workflow can occupy the technicians for extended periods, which becomes an issue when many customers ask for minor changes at the same time since each request must be handled by the technician individually. This led to the following problem statement:

"How can Intility streamline, automate and propagate requests for network changes in their secure zone model?"

---

Intility requested us to make a proof-of-concept web application that offers their customers an easy and intuitive way of requesting minor changes to their own networks. The developed system is internally named **Request For Change** (RFC) which should not be confused with the existing abbreviation for *Request for comment*. A change request can, for example, include changing the guest WiFi password and Service Set Identifier (SSID), which is the name of the network. Each request must be approved by one or more technicians at Intility, which further triggers automated services in the background that perform the changes. To enable this service to work between a customer's secure zone and Intility's secure zone, the application uses the Kong API Gateway (Kong) [2]. Kong acts as a central hub that forwards different Application Programming Interface (API) calls to various services. Furthermore, Kong is configured with plugins to implement extra security features. The specific application requirements from Intility were:

- The frontend should be built using React [3], a JavaScript library for building user interfaces, in conjunction with components from Intility's design system.
- The backend should preferably be written in the programming language Python.
- RFC should use Cisco DNA Center [4], a network management platform, and Cisco Identity Service Engine [5], a security administration tool to change the network configurations.
- RFC should communicate between secure zones using Metro, which is Intility's messaging system.

Further requirements specifications are described in Chapter 3.

The main focus of the project was implementing the fundamental structure of sending change requests and establishing data flow between the different secure zones. There was also a focus on implementing three core features: changing guest network SSID, changing guest network password, as well as adding new devices to the network identity group. Since the assignment was to make a proof-of-concept solution, deploying it for production has been out of scope.

### 1.3 Target Audience

This report is targeted toward people in the educational sector, mainly the examiner of this project. It is also written for students who might be interested in software engineering, including process frameworks, continual improvement, automation, and web development. The product itself is targeted toward two different user groups. Internally at Intility, their technicians will use it to approve or decline requests. Externally, Intility's customers will access a customer interface in the same system and use this to create new change requests. These two groups have different requirements for functionality and usability.

---

## 1.4 Project organization

The group consists of four members studying Digital Infrastructure and Cyber Security at The Norwegian University of Science and Technology (NTNU) in Gjøvik. The members of the group are:

- Sondre Gunneng  
*s.gunneng@gmail.com*
- Herman Torland  
*hermantorland@gmail.com*
- Øyvind Wold  
*oyvindw@gmail.com*
- Henrik Granlund  
*henrik.granlund@icloud.com*

The group has knowledge of a broad spectrum of topics within information technology, such as networking, programming, operating systems, infrastructure, and cyber security. However, API development, Python and React are all new topics that the group lacked previous knowledge of. The group members have been collaborating as a team on various projects since 2019. While large parts of the project were done conjointly, there was still beneficial to delegate roles of responsibilities within the group. In addition, the project had several external stakeholders, both from Intility and from NTNU. The project included the following roles:

Internal roles:

- Project lead: Herman Torland
- Project manager: Øyvind Wold
- Backend lead: Sondre Gunneng
- Frontend lead: Henrik Granlund

External roles

- Product owner: Marius Engan, Head of Edge Infrastructure at Intility.  
*marius.engan@intility.no*
- Supervisor at Intility: Tobias Bryhn-Bjølgerud, DevOps Trainee at Intility.  
*tobias.bryhn-bjolgerud@intility.no*
- Supervisor at NTNU: Jia-Chun Lin, Assistant Professor at NTNU.  
*jia-chun.lin@ntnu.no*



---

Role descriptions:

- Project lead: Made sure all the deadlines were met and conflicts were solved. The project lead was also the primary communicator with the stakeholders of the project.
- Project manager: Took the lead role in the absence of the project lead and made sure the group met the deadlines. The project manager was also responsible for facilitating Scrum and organizing the documents.
- Backend lead: Had the overall responsibility for the backend code, including making sure it was of good standard and structure.
- Frontend lead: Had the overall frontend responsibility, ensuring the user interface was of a good standard and followed Intility's guidelines.
- Secretary: Wrote minutes of meetings. This role was rotated between all internal group members every week.
- Product owner: Had a supervisory role with suggestions, feature requests, and executive decisions.
- Supervisor at Intility: The main point of contact at Intility, who was involved in day-to-day decisions and organization of the system's production.
- Supervisor at NTNU: Attended weekly meetings to provide feedback and assistance on both the system features and the project thesis.

The project had a duration from January 10th to May 20th, 2022. Before starting the main project, a project plan had to be approved. This plan contained our proposed schedule in the form of a Gantt chart (See appendix D), as well as the initial requirements. During the project phase, the group has followed a digital-physical hybrid workflow, combining online and physical meetings at Intility's headquarters in Oslo. The group has strived to treat the project as a regular job by working every day during core business hours. To work within Intility's environment, we received equipment from them to use. For documentation during the project, we have mainly used the Microsoft 365 suite [6], as well as keeping track of tasks using Trello [7], which is an online issue tracking tool. The final report is written in LaTeX using Overleaf [8]. More information about the project process and documentation is outlined in Chapter 6.

---

## 1.5 Thesis structure

1. Introduction: This chapter will introduce the assignment from Intility and the high-level problem description. In addition, the target audience, project organization, and structure of the report will also be presented.
2. Background: This chapter will introduce the problem in greater detail and outline the chosen tools for the project. In addition, some technical details and explanations of the different tools and technologies are also included.
3. Requirements specification: This chapter will present the requirements for the system, which includes requirements from Intility and our contributions to the project. There are both functional and non-functional requirements. The functional requirements are the concrete specifications and features the system must include, whereas the non-functional requirements are more abstract requirements on how the system should perform.
4. System design: The group has developed RFC according to Intility's problem and requirements. This chapter describes the architecture of RFC and the different features that are implemented on top of it. The chapter also covers some design decisions that were taken.
5. Implementation: This chapter will go into greater technical details on the implementation of the system, including some code snippets with explanations of how they work. The chapter will also cover some of our decisions during the programming phase.
6. Development process: This chapter will describe the process during the different phases of the project. In addition, it will outline the chosen process frameworks and how the various elements of the project are documented.
7. Evaluation: This chapter contains an evaluation of how the developed system performs according to the requirements specified in Chapter 3.
8. Closing remarks: This chapter will outline future work that could be done in the project. The learning outcome and conclusion of the project are also included.

---

## 2 Background

This chapter will introduce the problem in greater detail and outline the chosen tools for the project. In addition, some technical details and explanations of the different tools and technologies are also included.

### 2.1 Detailed problem description

As described in Chapter 1, Intility currently has a labor-intensive and manual process for handling minor change requests. For example, if a customer wants to change the SSID of their guest WiFi, they will need to contact Intility support and request this change. The support department will thus create a support ticket with the required information and forward it to a technician at Intility's Network Services department. This technician will manually log into the relevant wireless LAN controller (WLC) using Secure Shell (SSH). Furthermore, the technician will need to type a total of 13 commands into the terminal in order to perform the SSID change. The technician will then provide a response to the support department, which ultimately delivers confirmation to the customer. A similar flow applies to other changes like modifying the guest WiFi password or adding a new device to the customer's network identity group. The design and functionality of the different features are further described in Chapter 4. The current request for change flow is visualized in Figure 1.

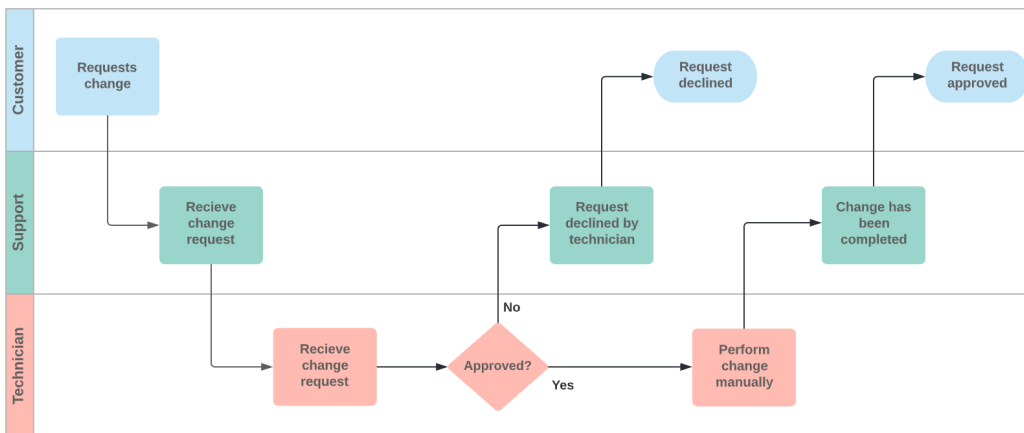


Figure 1: The current request for change flow at Intility

The manual and repetitive process described in Figure 1 could be automated and streamlined. Intility wanted this process to bypass their support department to release capacity for tasks requiring more human input. The proposed solution was to create a dedicated web portal where Intility's customers could make new change requests using an intuitive graphical user interface. These requests are sent directly to a service technician who can quickly glance over the request and

---

approve it with a simple click of a button. This triggers services in the background that automatically log into the customer equipment and perform the change. This desired flow is visualized in Figure 2.

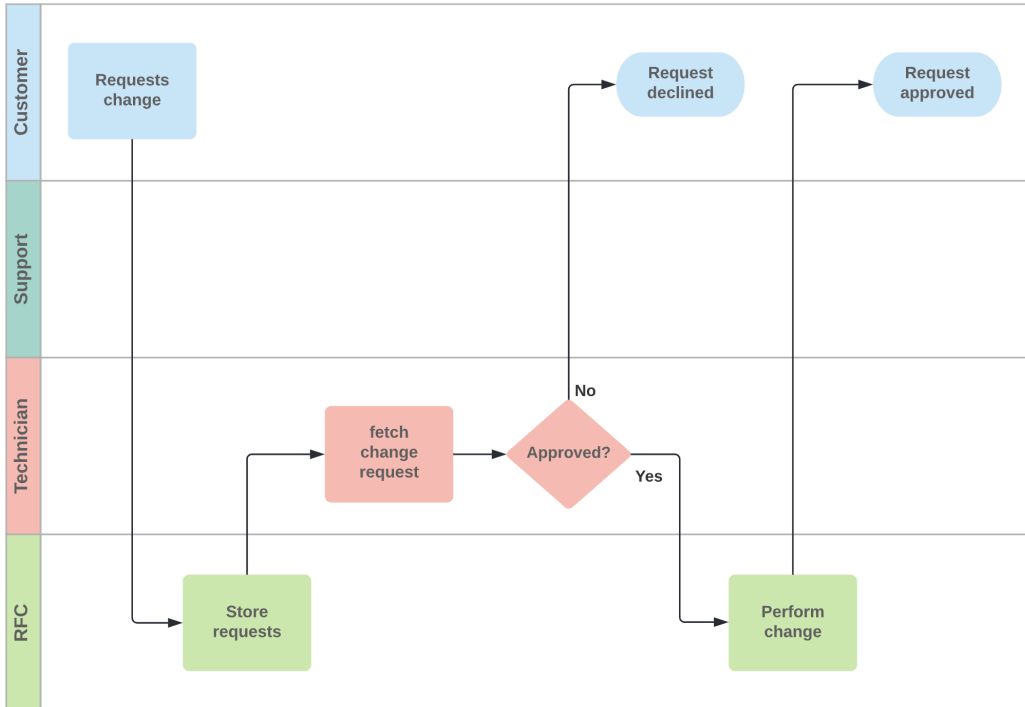


Figure 2: The desired request for change flow

## 2.2 Purpose

This thesis aims to outline our thoughts and process throughout the project. The report describes our research, preparation, and implementation of a web application and is meant to document each phase of the project. The report also describes our design decisions, problems, and workflow using project process frameworks. The purpose of the project itself was to utilize and build upon everything we have learned throughout the different courses of our degree and use this knowledge to develop a new system while also documenting the entire process. By following a process framework like Scrum and working closely with an IT business, we have gained insight into how software engineering is done in the real world.

---

## 2.3 Tool survey

In order to familiarize ourselves with the different tools and technologies available for our use case, we decided to perform a tool survey in the introductory phase of the project. The following section outlines our chosen tools and technologies for the project, as well as some alternatives that we considered.

### 2.3.1 Bifrost

Bifrost [9] is Intility’s design system developed by their user interface department. This system outlines different design guidelines and provides pre-made React components such as buttons, loading bars, and input fields. This made it easy to import components from Bifrost into the code base and make it look professional and fit into Intility’s aesthetic. There are many different design systems available for React. However, Bifrost was used in this project due to the specific requirement from Intility, as stated in Chapter 3. Using such a design system is beneficial for reducing development time while keeping the design visually coherent.

### 2.3.2 Database

The system needed a database solution to store all customer requests. After looking at the different available options, the following database stack was chosen for the project. The database is running PostgreSQL [10], which was selected because it is a robust open-source relational database with a wider variety of data types than MySQL [11]. In addition, the development database is hosted in Microsoft Azure [12] because Intility already uses Azure as their cloud provider. This also has the added benefit of sharing test data between the group members, as everyone uses the same database instance.

The database migrations are done through a tool called Alembic [13], which stems from a project called SQLAlchemy [14]. Migrations can be thought of as version files describing all the details of the database and information about the changes that are made. When working with databases, it is beneficial to describe the structure separately from the database software, since this makes changing database software in the future more manageable. SQLAlchemy has Object-relational mapping (ORM) [15], which is utilized with SQLAlchemyModel [16]. An ORM is often used with relational databases to create a mapping between the database and objects in a programming language like Python. Because the group has limited experience with migration tools and ORMs, DBBeaver [17] was used to visualize, troubleshoot, and manually add data to the database.

When it came to choosing a database solution, many different factors came into play. Even though the selected solution ended up being a SQL database, the NoSQL alternatives also had to be considered. One of the differences between a SQL and NoSQL database is how the data is structured and queried. Another

---

thing to consider is the support for the ACID properties [18], which are the following: **Atomicity** meaning that either all or none of the actions are performed. **Consistency** makes sure no data is lost in a transaction. **Isolation** is making sure the same data do not exist in more than one place at a time. Lastly, **durability** will make certain fulfilled transactions are not undone upon system failure. The fact that SQL databases have good support for the ACID properties is the most important reason that SQL was chosen over NoSQL for this project. This is because of what the application will be used for, which is making system changes on behalf of customers and the changes can be approved by multiple technicians, creating a scenario where two people accepting the change at the same time is possible. To avoid the change being run twice, the isolation property of ACID is important.

### 2.3.3 Docker

Docker [19] is a virtualization tool that allows the creation of isolated environments called containers. These containers are based on images built using a recipe called a Dockerfile. Using containers leads to more efficient use of the host machine hardware and improves security both locally and in production. An alternative to containers is running a virtual machine (VM) for each application. However, this is considered less efficient due to the need for an operating system (OS) for each VM. A visual representation can be seen in Figure 3, where each of the color-coded columns represents the requirements for running an instance of an application. This figure is based on an illustration in the docker documentation [20] For this project, Docker is used to simulate the production environment by running all the different services locally during development. This also makes it easier to share the environment between the group members and simplifies the process of deploying the system to production.

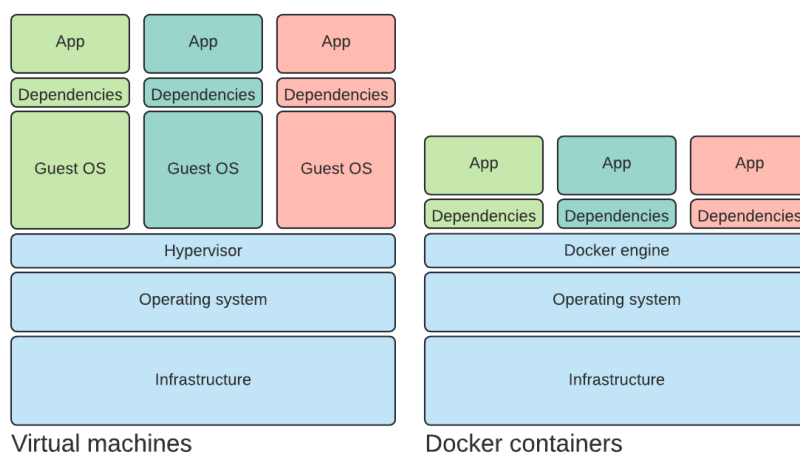


Figure 3: Virtual machines vs containers

---

### 2.3.4 FastAPI

FastAPI is a web framework written in Python and is built upon another framework called Starlette [21], which is built with asynchronous (async) support. Async means that the CPU can divert its resources while the async function is running. This can, for example, be that the function needs to fetch something from an external source, such as over a network connection or an IO operation. FastAPI implements pydantic [22] to facilitate the creation of web APIs. Another significant point is that FastAPI is built using Python 3.6, and that type hinting was introduced in Python 3.5 [23]. The use of type hints in the source code of FastAPI integrates well with Integrated Development Environments (IDE). This improves the developer experience by providing a more detailed description of data, functions, and objects. As stated in the official documentation for pydantic “pydantic enforces type hints at runtime, and provides user-friendly errors when data is invalid” [22]. This leads to more stable code because type checking will be done by python when the application starts but can also be done when data is passed to each endpoint. This makes it possible to use type checking for data validation. Lastly, a feature worth noting is support for the OpenAPI standard, enabling automatic generation of documentation with Swagger [24].

Django [25] was an alternative to FastAPI that was considered to create the backend but was ultimately discarded. Django enforces the Model-View-Controller (MVC) pattern, which is a way to structure the code snippets based on their purpose. Django is considered to have all batteries included, which means it chooses all tools, such as the ORM, for the developer. Furthermore, Django does not prioritize APIs, which means it is recommended to use a third-party package called *djangorestframework* (DRF) [26] to facilitate API development. Another noteworthy thing is that the same team develops Starlette and DRF, and according to their *Encode* project page on GitHub [27] Starlette has more activity than DRF, which means it is more frequently maintained. Using FastAPI makes it faster and easier to get started because it lets developers quickly write endpoints without previous knowledge. It also enables the developer to organize the structure based on project-specific needs. FastAPI also makes the learning process more straightforward because the developer can implement features piece by piece without having any framework-specific knowledge.

### 2.3.5 Gitlab and git

Git [28] is a version control system that keeps track of changes made to the code, as well as provides tools for creating branches and merging conflicts. Gitlab [29] is an online platform that incorporates git for collaborating on coding projects. In addition, Gitlab provides a comprehensive suite of project management tools like issue tracking, assigning tasks, and tools for continuous integration and delivery (CI/CD). This is a methodology based around automating the process of deploying new code to a project, such as running pipelines that, for example, perform tests

---

on the code. The challenge with multiple developers working on the same project is that everyone has different preferences for structuring their code. One way to avoid this becoming a problem is by following a coding standard like PEP-8 [30]. However, learning everything about a coding standard can steal time away from development. To alleviate this, pre-commit [31] was used, which is a tool that runs scripts to verify that the code is structured correctly every time someone tries to create a commit. Pre-commit can be configured to automatically fix minor issues like line breaks and blank line removal and highlight more severe problems for the developer to fix manually. Figure 4 shows the pre-commit failing due to an unremoved print statement on line 63 of the *cisco\_wlc.py* file, which should be removed before pushing the code to Gitlab.

```
>> rfc-backend git:(main) 09:39 git commit -m "refactor"
black..... Passed
Check for case conflicts..... Passed
Fix End of Files..... Passed
Trim Trailing Whitespace..... Passed
Check python ast..... Passed
Check JSON.....(no files to check) Skipped
Check for merge conflicts..... Passed
Detect Private Key..... Passed
Fix double quoted strings..... Passed
flake8..... Failed
- hook id: flake8
- exit code: 1

app/services/cisco_wlc.py:63:9: T001 print found.

isort..... Passed
mypy..... Passed
>> rfc-backend git:(main) 09:39 |
```

Figure 4: Pre-commit results with failed test.

### 2.3.6 JavaScript

Following Intility's requirements, the application is written using the React library. This is a library that uses JavaScript. JavaScript is a programming language with a large web-development community, making it easy to find documentation online. There was also an option to use TypeScript [32], which Intility mainly uses. TypeScript adds type annotations to JavaScript, as well as enhances browser compatibility. However, since none of the group members had earlier experience with TypeScript and the project did not demand the extra functionality, the group chose to write the application in JavaScript. Should the need arise in the future, it is relatively straightforward to convert the codebase to TypeScript, making this decision less impactful.



---

### 2.3.7 Kong

As stated in the assignment description, the original plan was to use Metro to deliver the requests across the secure zones. Metro is a messaging system developed by Intility that is built on Azure Service Bus [33]. Metro can be simplified as a radio station that broadcasts messages across different frequencies. Any service on the network can tune into a given frequency to receive the messages broadcasted on it. Instead of going between the secure zones, Metro goes around them. Internally at Intility, this was in many ways considered a temporary solution that solved the problem in a sub-optimal way. There was a wish from Intility to find a more permanent and more secure solution to this problem.

After researching different options, the decision was made to implement the Kong API Gateway [2]. This acts as a centralized hub for multiple APIs, which means that rather than needing separate routes to each API, the applications can access a single IP address for all API endpoints. Furthermore, Kong handles each API request individually and forwards it to the correct service. One of the main benefits of Kong is the built-in security functionalities. There are, among others, plugins for authentication, traffic control, and logging, which can easily be installed.

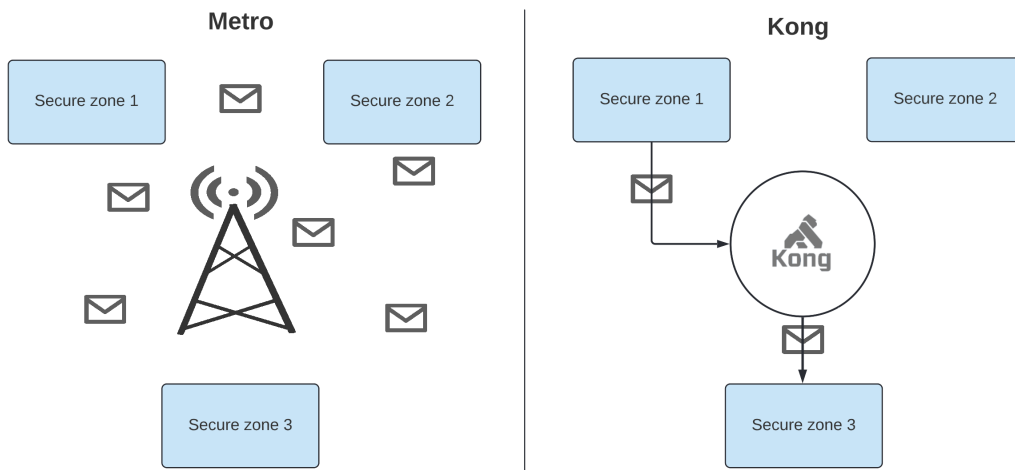


Figure 5: Metro vs Kong

### 2.3.8 Lucidchart

In order to create figures, diagrams, and tables, we chose to use Lucidchart [34] due to its low latency and live collaboration. Lucidchart offers a free university license which gives us access to more features and allows us to create charts without any limitations. An alternative would be to use Draw.io, which offers many of the same features, but is slower and lacks the same ability for live collaboration. Lucidchart also has a broad spectrum of pre-made templates and icons.

---

### 2.3.9 Microsoft 365

Microsoft 365 [6] is a suite of productivity tools for creating documents, sharing content, and collaboration. For this project, the group has mainly used Word for text-based documents, Excel for spreadsheets, OneDrive for shared backup and storage, and Teams for collaboration. We have mainly worked within Intility's Microsoft environment to simplify collaboration with their employees and make use of the end-to-end encrypted video calls and screen sharing.

### 2.3.10 OPA

Open Policy Agent (OPA) [35] is an open-source policy engine that enforces policies specified with code. OPA works by offloading policy decision-making from other systems, which makes it suitable for microservices. For this project, OPA is used to enforce policies whenever a request is sent to the API gateway. In order to personalize the user interface, as well as ensure system integrity, token management was implemented. The requesting entity is identified as either a technician or a customer by the unique user ID and role sent together with the request.

### 2.3.11 PyCharm

For the IDE, PyCharm [36] was used by all group members. PyCharm supports all the technologies and languages used in the project and provides useful features like "Code-with-me", which allows multiple developers to connect to one editor remotely. In addition to this, our supervisor at Intility already used PyCharm, and had a lot of valuable tips for configurations and extensions.

### 2.3.12 React

Intility required us to use React [3] for the frontend. React is a JavaScript library developed by Meta, which makes it easy and fast to create user interfaces for web applications. React is based on components that contain all their functionality and styling. These components can be imported within each other to create interfaces in a modular way. This makes it easy to use third-party design libraries with React, such as Intility's design system; Bifrost.

---

## 3 Requirements specification

This chapter will present the requirements for the system, which includes requirements from Intility and our contributions to the project. There are both functional and non-functional requirements. The functional requirements are the concrete specifications and features the system must include, while the non-functional requirements are more abstract requirements on how the system should perform.

### 3.1 Functional requirements

#### **Programming languages**

RFC should be a web application with a frontend developed in React in conjunction with JavaScript and a backend made in Python. The main reason for these requirements is to ensure that the system can be easily developed further by Intility at a later stage.

#### **API**

RFC should be an API-driven application, where the frontend communicates with a REST-API backend using fetch requests. This API should be created using the FastAPI framework and provide several valuable endpoints.

#### **Kong**

The Kong API gateway should be implemented to allow communication between the frontend and backend.

#### **UI design**

Intility requires that their general developers' guidelines regarding UI design should be followed. This should be ensured by using the Bifrost design system.

#### **Policies**

RFC should use OPA to enhance security and enforce policies. This will, for instance, be responsible for checking that only authorized technicians can approve a given request.

#### **Features**

RFC should be able to trigger several different features once a request is approved. The scope of this project includes implementing a feature for changing network SSID and password using Cisco DNA Center, as well as adding a new device to the network identity group using Cisco Identity Service Engine.

---

## 3.2 Non-functional requirements

### **Security**

RFC should follow good security practices. For instance, the authentication should be done server-side.

### **User experience**

The interface of RFC should be intuitive and natural to use in order to allow users to move around without feeling confused or lost.

### **Adaptive to different screen sizes**

A technician might work on a desktop, tablet, or mobile. Therefore, the interface of RFC should be adaptive to different screen sizes in order to facilitate the different work environments of the end-users.

### **Responsiveness**

The interface should be responsive and perceived as fast. This should be solved by implementing performance techniques like caching and loading animations to visualize necessary background activity.

### **Reusability**

The code should be easily reusable for future development. This should be achieved by using the same languages, frameworks, and code structures that the developers at Intility are familiar with. There should also be a modular approach for each service, and the different components of the code should be written with reusability in mind where possible.

---

## 4 System design

The group has developed RFC according to the above-mentioned requirements. This chapter describes the architecture of RFC and the different features that are implemented on top of it. The chapter also covers some design decisions that were taken. More technical details, including code snippets, are covered in Chapter 5.

### 4.1 Architecture of RFC

The planned architecture of RFC was agreed upon with Intility at an early stage of the project. Figure 6 shows the overall architecture of RFC, including multiple customers, in order to illustrate how the same method is used for each secure zone. RFC is centered around the Kong API Gateway, which acts as a communication hub between the different services and secure zones. This is necessary because of the isolated nature of each zone, which significantly limits the ability to communicate between them.

By implementing Kong, each secure zone can send API calls to a centralized hub, which will forward the request to the correct service. In a simplified way, Kong punches a hole through the firewall of each zone and further governs what traffic that passes through this hole. The rules for what type of traffic Kong allows are defined using another service called OPA, which is used to verify each API call and validate it accordingly. The backend is designed to be flexible towards what kinds of features and services it triggers, thus making the system easily scalable for new features. The designs behind the currently implemented features are described in Section 4.2.

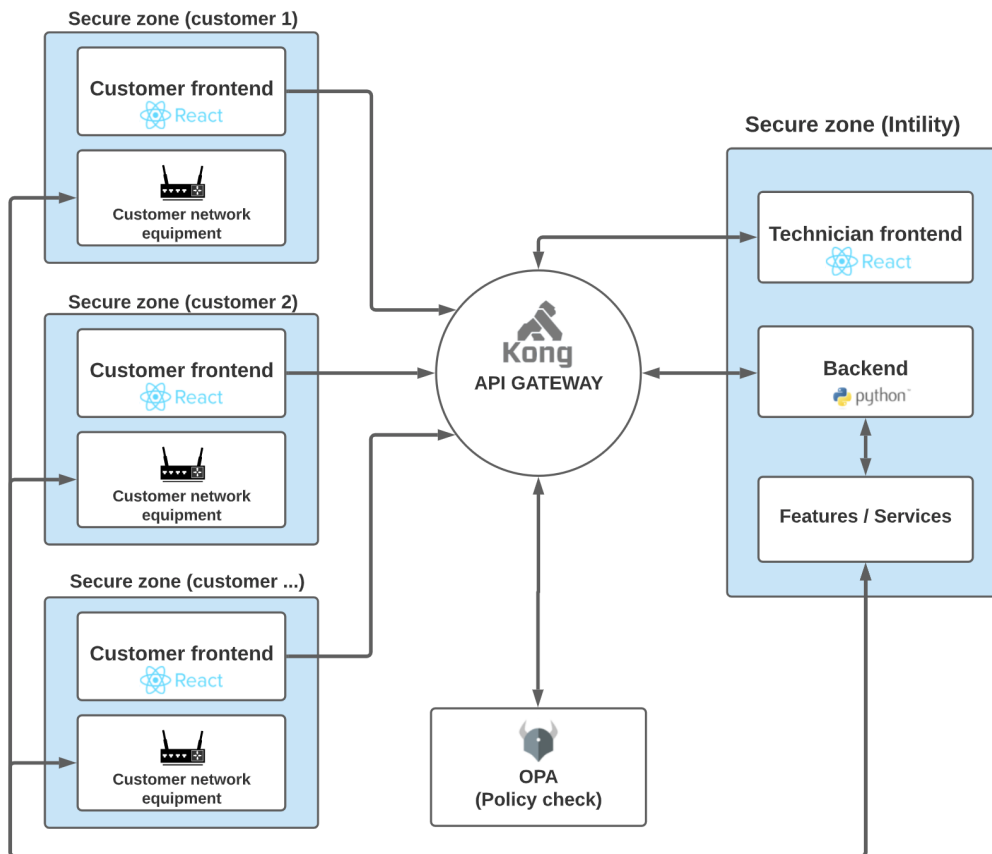


Figure 6: System architecture of RFC

The project's minimum viable product (MVP) was a solution that allows a customer to create a new change request through a web interface. This request is sent directly to a technician rather than through Intility's customer support, thus increasing customer support's capacity to handle more complicated cases. RFC is implemented as a frontend written in React, communicating with an API developed with FastAPI. This communication goes through Kong and is authorized by OPA. The API provides several endpoints that the frontend interface communicates with.

The frontend contains separate views for customers and technicians, containing different features based on their needs. The alternative was to create two separate frontend projects for the two different user groups. The main reason this was considered was to make the separation of concerns clearer. It could also be argued that this would have given more granular control over authorization on an infrastructural level because the two frontends would run in separate containers.

However, in order to avoid maintaining two different projects, it was ultimately decided to develop both views in the same project and provide the correct interface based on which authorizations the user has. These authorizations are based on which roles the users are assigned in Azure Active Directory (Azure AD), which is a cloud-based identity and access management service [37]. This information is decoded from the JSON Web Token that is attached with each API call. Figure 7 shows the landing page when a user with the "technician" role logs in.

ID	Type	Value from	Value to	Created	Last updated	Status		
> 1	SSID	XYZ	CompanyX	12:22   27.04.22	12:25   27.04.22	Pending	Accept 1/2	Decline
> 5	SSID	admin	Seminar2022	09:30   28.04.22	09:30   28.04.22	Pending	Accept	Decline
> 9	SSID	lorem	Lobby_Guest	10:58   03.05.22	10:58   03.05.22	Pending	Accept	Decline

Figure 7: Technician interface for approving requests

In order to support this architecture, a database was needed to store the requests, customer sites, and endpoint identity group tags. The *requests* relation contains information about who made the request, who approved the request, when it was created, when it was updated, the type of change, a comment, the action that will be taken, and the current state of the request. There is also a field for the number of approvals since some changes might need multiple approvals by different technicians. The user information is retrieved from Intility's Azure AD, specifically *tenant\_id* which will be mapped to *customer\_id*, and *oid* (unique object id) which will be mapped to *technician\_id*. There is also a relation containing all the customers with their affiliated sites (office locations). In addition to this, there is a relation that contains each customer's Endpoint identity group, with a corresponding friendly name (tag). Figure 8 shows the conceptual model of the database.

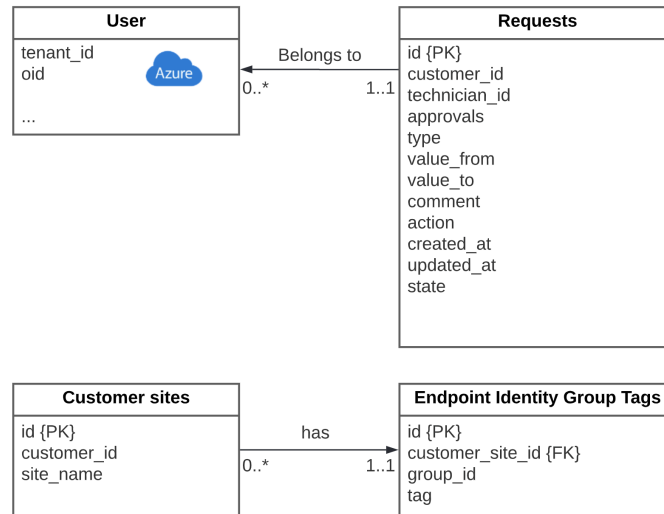


Figure 8: Conceptual database model

## 4.2 Features

In addition to developing the core RFC functionality, the assignment from Intility also involved creating one or more features that utilize the fundamental system described in Section 4.1. This was a more flexible phase of the project where the group could provide input and ideas for what type of features could be implemented. The decided and currently implemented features in RFC are described below.

### 4.2.1 WLAN changes

The purpose of Wireless LAN (WLAN) related changes is to give Intility's customers the ability to make changes to their networks through a user-friendly web interface. Since most customer networks follow Intility's naming conventions, the scope of this feature only includes changes made to the guest network at each customer location. The supported WLAN changes in RFC are **SSID change**, **password change**, and **a toggle to turn on or off the WLAN itself**.

Currently, the supported WLAN changes are few and simple from the end-user's perspective. This helps to make the interface clutter-free and user-friendly. However, more WLAN features could be added with minimal effort. All the WLAN settings are combined into a modal window called *Guest WiFi Settings* (See Figure 9). This modal contains information about the current SSID, as well as how many client devices that are connected to it.

To reduce the number of elements on the screen, the functionality for requesting an SSID or password change is contained behind two different buttons with slide-



---

down content. Clicking one of these buttons exposes two input fields for the user to enter the new SSID/Password as well as an optional comment if they wish to provide some extra information to the technician. Figure 10 shows the *Guest WiFi Settings* modal window in different states.

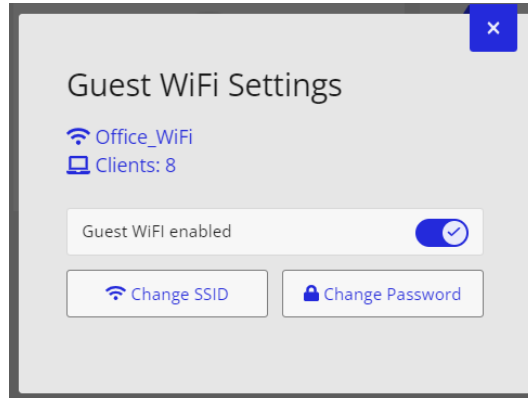
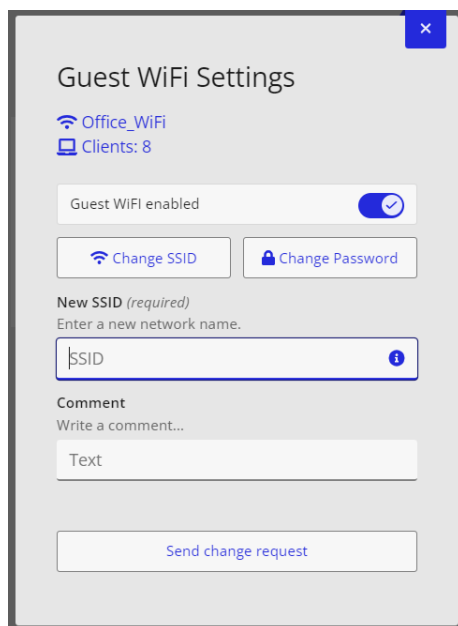
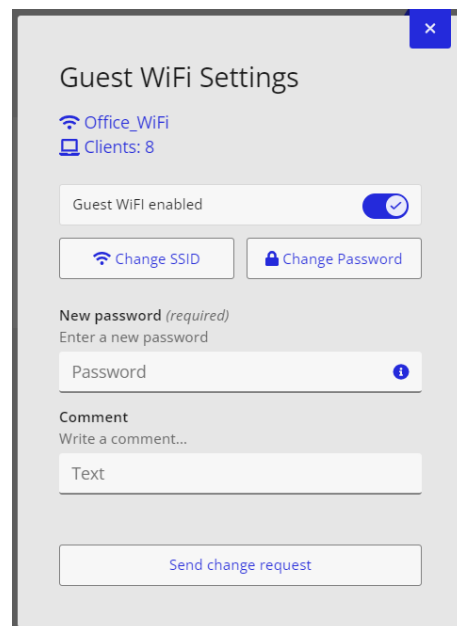


Figure 9: WLAN modal window with collapsed change SSID/Password forms.



(a) Form for requesting an SSID change.



(b) Form for requesting a password change.

Figure 10: WLAN modal window with opened change SSID/Password forms.

The planned solution for implementing the different WLAN features was to use Cisco DNA Center (DNAC), which is a management platform for Cisco networking equipment. It is used for planning, configuring, monitoring, and updating networks built using Cisco products. In addition to providing a graphical user interface,

---

DNAC also has an API that can be used to integrate its functionality into other systems [4].

Because Intility mainly uses Cisco products in their networks and already has DNAC configured, RFC was initially planned to use the DNAC API. However, this solution turned out to have some undesirable side effects. For example, when DNAC pushes a change to the WLC, a provisioning process will get started. This provisioning process has a high processing overhead and may cause disruptions across the network. In practice, this means that a simple change to a WLAN could cause a loss of network access for many devices over a short period.

After discussions with Intility's lead WiFi engineer, it was agreed that this overhead was unacceptable. The alternative solution implemented in RFC was using SSH to connect directly to the WLC and modify the configuration file for the network. This process is automated using a python package called Netmiko [38]. The technical details of this solution are described in Chapter 5.

#### **4.2.2 Add a new device to MAC Authentication Bypass (MAB)**

This feature allows a customer to add new devices to their networks, such as a security camera or a smart light bulb. This feature is necessary because of the MAB mechanism that is configured on Intility's networks. This is a security feature that allows a device to authenticate on the network based on its MAC address. This is a layer two address in the Open Systems Interconnection (OSI) model [39] that uniquely identifies a network-connected device. The MAC address comprises six pairs of hexadecimal characters, separated by colons.

In order to allow a device to communicate on the network, it needs to be added to the list of approved devices in MAB. The current scope of this feature only involves adding a single device in each request. However, this could be extended in the future by making it possible to upload a spreadsheet file with multiple devices or allow the user to choose the number of input fields dynamically.

Because Intility uses Cisco as their network equipment provider, they also use the Identity Service Engine (ISE) [5], which is a service that simplifies access control and policy enforcement. ISE allows the creation of Endpoint Identity Groups (EIG), which are a collection of devices that utilizes the same policy set. The policy set is a set of rules for what type of traffic is allowed on a particular EIG. The need for separate EIGs arises when different types of devices should follow different rules on the network. For instance, a security camera might need different rights on a network than a light bulb. After grouping devices in an EIG, the EIG is assigned to a Virtual LAN (VLAN). This grants network access to all the devices in the EIG on that particular VLAN. Figure 11 shows how a security camera gets authenticated on the network using MAB and ISE.

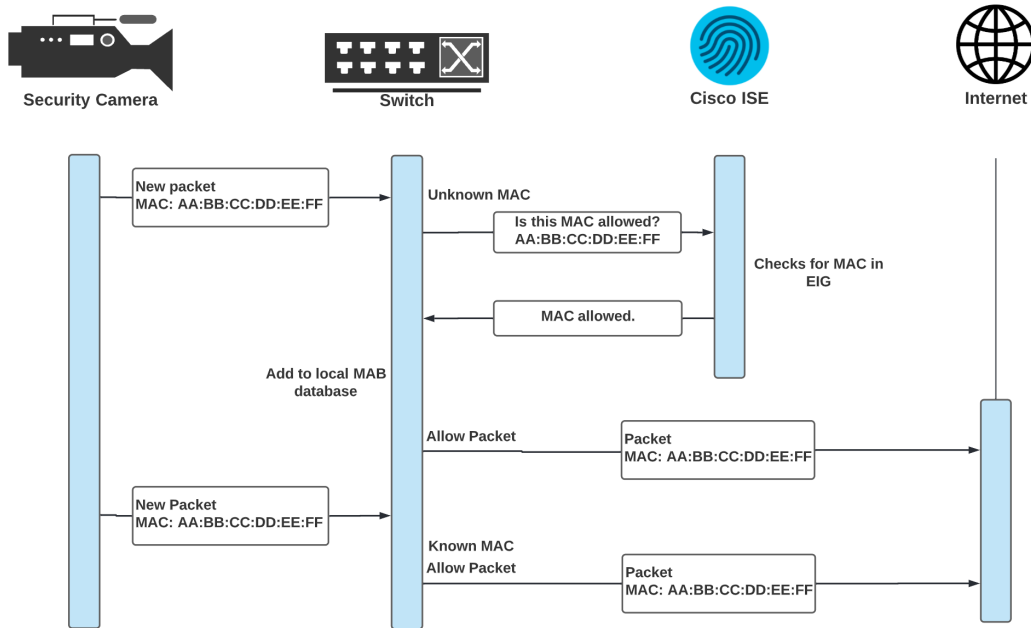


Figure 11: MAB and ISE

In order to add a new device, the user needs to select which site (location) and Tag (EIG) the device should get added to. Furthermore, the user must enter the MAC address, as well as an optional description for the device.

The site names are retrieved from a database and are named according to Intility’s conventions. For instance, the site ZZ-0D-Oslo is composed of the customer code (ZZ) and the location (0D-Oslo). The available tags are based on which site the customer chooses and are retrieved from the ISE API. The API returns a Universal Unique Identifier (UUID) for the EIG, which is a long and complex string (for instance 87840c90-5749-11ec-b9a5-4e04ca7e7bcf) that is hard to conceptualize for the end-user. In order to keep the system user-friendly, this UUID is replaced in the frontend by a friendly tag name that is stored in the RFC database. This tag name could, for instance, be *Security Cameras*.

### 4.3 User interface design

The user interface of RFC is built using Bifrost, which was one of the requirements from Intility. Bifrost is Intility’s design system, which is a React component library developed in-house by their engineers and UI designers. Using a component library like Bifrost makes it fast and easy to create professional-looking user interfaces that are coherent and matches Intility’s company aesthetic. By using Bifrost, the code for components like buttons and input fields could be copied directly into the RFC code. This vastly decreased the development time of the system.

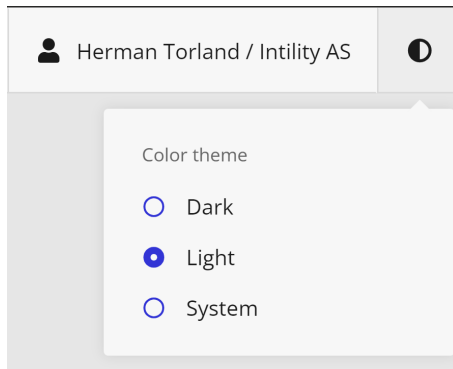
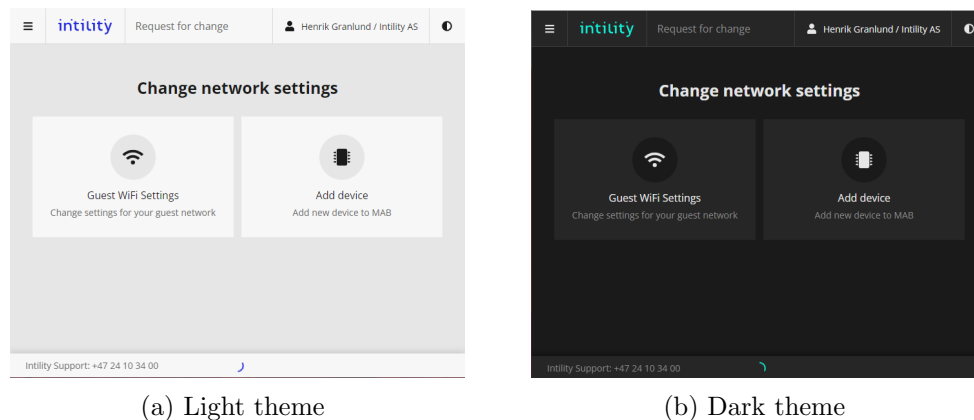


Figure 12: Theme toggle

Bifrost also makes it easy to implement features like a toggle between dark and light themes and make the website adaptive to different screen sizes. RFC utilizes both of these advantages, and Figure 12 shows the theme selector of RFC, and Figure 13 shows the effect of this toggle on the customer landing page.



(a) Light theme

(b) Dark theme

Figure 13: Customer landing page with different themes

An essential aspect of user-centered design is providing feedback and guidance to the user. RFC is designed to be easy to use and always give the user a sense of what is happening. This is done by implementing loading animations and tooltips and providing live input sanitization. Figure 14 shows how RFC presents an animated loading skeleton for the WLAN information that is yet to be retrieved by the backend. This provides the user with a sense of progress rather than showing a static interface that might give the perception of the website hanging.

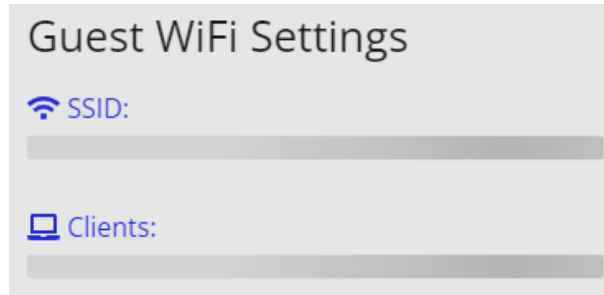


Figure 14: Animated skeleton while retrieving WLAN information from the backend

Figure 15 shows how the user can hover their mouse above an information icon in order to see more detailed information about what type of input is allowed. This will guide the user toward inputting a correctly formatted SSID name. By the zero-trust principles, there are also input validation rules on each field in case a user still tries to submit an invalid request. Figure 16 shows how RFC responds to a user submitting an invalid SSID change request, and Figure 17 shows the response to a password change request that does not contain enough characters.

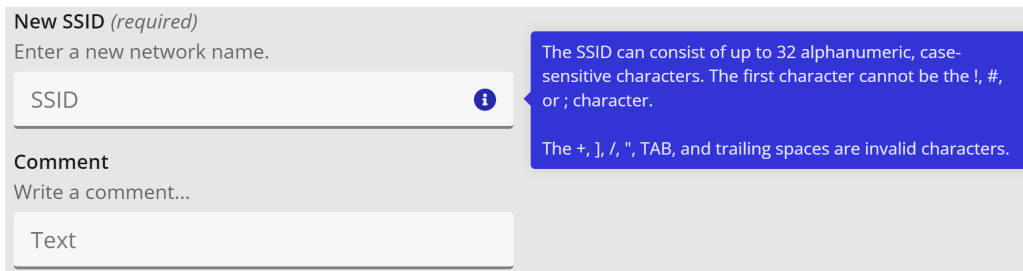


Figure 15: Tooltip explaining SSID requirements

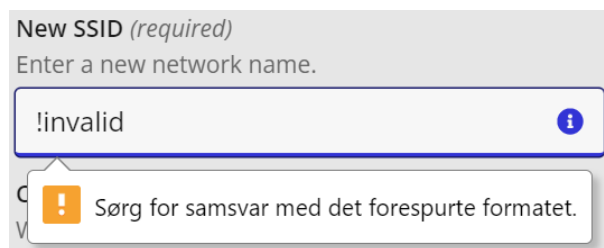


Figure 16: User feedback for use of invalid SSID format

New password (required)  
Enter a new password

short

! Du må forlenge denne teksten til 8 tegn eller mer (for øyeblikket bruker du 5 tegn).

Figure 17: User feedback for too short password

The form for adding a new device to the network is another feature that required some particular UI considerations. Since the list of tags is dependent on which site the customer chooses, this form requires the user to enter the information in chronological order. To achieve this, each field of the form is disabled and greyed out until the preceding field is filled out. In addition to this, the form will perform sanitization of the MAC address field to ensure the address is in the correct format. The user will only be able to enter hexadecimal characters, and the colons between each pair of hexadecimals are injected automatically. Once a valid MAC address is entered, the user will be provided with a green checkmark and the ability to submit the request. By implementing these measures, we can ensure that the users can only submit requests in the right format while also guiding the users through the form. Figure 18 shows the user interface for adding a new device.

(a) Empty Add device form

(b) Filled Add device form

Figure 18: Form for adding a new device with forced order of input and MAC address sanitization

---

## 5 Implementation

This chapter will go into greater technical details on the implementation of the system, including some code snippets with explanations of how they work. The chapter will also cover some of our decisions during the programming phase.

### 5.1 Backend architecture

As described in Section 4.1 the core architecture of RFC is built using Kong and OPA. Kong is used as a gateway between the frontend and backend, and its functionality can be extended with plugins for authentication, security, and logging, among others. On a technical level, Kong intercepts the API calls, which means it can read and manipulate the requests. The difference between how Kong and a traditional firewall operates is that Kong can read the request body. This is possible because the frontend (localhost:3000) sends the request directly to Kong (localhost:8000), which further checks the configured routes to forward it to the backend (localhost:3010). In the case of a firewall, it sits in the middle and can not read the body because it is encrypted. By utilizing Kong with plugins, the system can perform different actions on the content of the requests. Currently, there are implemented plugins for Cross-Origin Resource Sharing (CORS) [40], OPA [35] and OpenID Connect [41].

The CORS plugin is used to configure which domains other than the backend the browser should allow resource loading from. For the current development environment, CORS is configured to only allow resources from the backend running at localhost:3010 and is enforced by the web browser. After the plugin has attached the CORS header to the request, Kong moves on and forwards the request to the OPA plugin. The OPA plugin provides authorization in the application by checking each request against a list (see Code snippet 1) of allowed endpoints based on different roles. For instance, a customer needs to make a POST (write) to the `/requests` endpoint to create a new change request.

The MVP implementation of RFC utilizes three different roles: customer, technician level 1, and technician level 2. The different technician levels define whether a technician can immediately approve a request or just increment the number of approvals needed to trigger the change. Each user's role is provided in the JSON web token that is passed with each request. These tokens are decoded and validated by the OpenID Connect plugin, which verifies the token's signature to ensure integrity. Figure 19 visualizes the flow of a packet through Kong and its plugins, while Code snippet 1 shows the configuration file in OPA for what endpoints a user may access. The asterix represents a wildcard which means it can take any value at that location in the string.

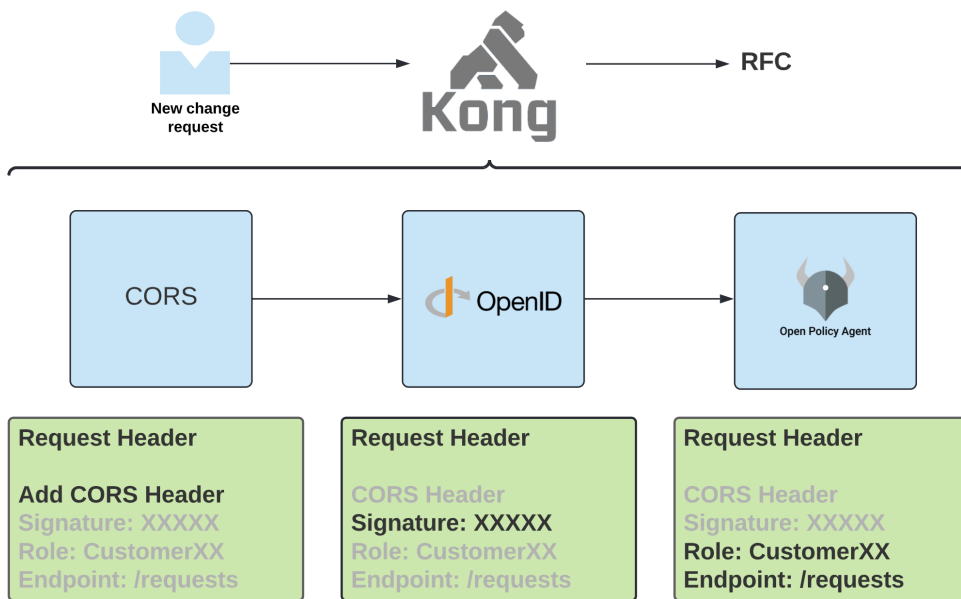


Figure 19: Flow of data through Kong with plugins

```

nsa_rfc:
  role_access:
    customer:
      read:
        - /requests
        - /wlc/*
        - /ise/*
        - /customer-sites
      write:
        - /requests
        - /wlc/*
        - /ise/*
    technician1:
      read:
        - /requests/*
      write:
        - /requests/*/accept
        - /requests/*/decline

```

Code snippet 1: OPA configuration for access to different endpoints based on roles



---

## 5.2 API

RFC is an API-driven application, which means that the API was created first, followed by the frontend that utilizes it. The API for RFC is developed using the FastAPI python framework, which makes writing endpoints a relatively fast and easy process. Code Snippet 2 shows the implementation of the GET /requests endpoint and how it returns all the requests in the database.

```
@router.get('/requests', response_model=list[RequestRead])
async def get_requests(
    session: AsyncSession = Depends(get_session)
) -> list[RequestRead]:
    """List all requests in the database"""

    result = await session.exec(select(Request))
    return [
        RequestRead(**request.dict()) for request in result.all()
    ]
```

Code snippet 2: Endpoint that returns all requests from the database

The first argument in the `@router.get` decorator is the path for where the endpoint listens for requests. The `response_model` describes how the response from the endpoint will be represented, in this case a list with `RequestRead` models. These models act as a filter that defines what type of data will be returned and does some validation to make sure the expected data is returned. It is not always the case that a function wants to return all the data from a source. An example would be retrieving a user from a database, where the user's password should not be included in the response. In the main body of the function, all the requests from the database are fetched through the async database `session`. Since this function is async, the response from the database needs to be awaited to make sure a response has been returned. If it is not awaited the application will in the best case crash, or in the worst case use the wrong data in the next step. Finally, all the requests returned in the response get iterated through and constructed into a list returned from the endpoint.

The different features of RFC are implemented as services in python that the endpoints can run. The current implementation includes two services, WLC and ISE, that the various features utilize. Code snippet 3 shows an example of how the configuration for the WLC service is written in the backend.

---

```
@dataclass
class CiscoWLC:
    """
    Modify network settings through Wireless Lan Controller
    using SSH
    """

    config = {
        'device_type': 'cisco_ios',
        'host': settings.WLC_HOST,
        'username': settings.WLC_USERNAME,
        'password': settings.WLC_PASSWORD,
    }
```

Code snippet 3: Class with netmiko config

The CiscoWLC service is written as a class with a *dataclass* decorator. The *dataclass* decorator removes a lot of boilerplate code and helps with following the “*don't repeat yourself*” principle, as it generates the constructor automatically with the properties. The different attributes are imported from the settings file, retrieving and validating constants provided in a .env file. In addition, the .env file contains environment variables, which are dependent on where the backend is running and makes it easy to switch between production and development environments. It is important to remember when working with .env files that it is not included in git when commits are made, as these files tend to include secrets like passwords.

The different features of each service are written as methods of the class. For the CiscoWLC service, methods for changing SSID, password, seeing WLAN status, and turning it on or off are implemented. Code snippet 4 shows the method for changing passwords on a given WLAN.

---

```

@classmethod
def change_password(
    cls, wlan_name: str, password: str
) -> None:
    """
    Changes the password on the guest network of a given WLAN
    Parameters:
        wlan_name: The name of the WLAN to change
        password: The new password for the guest WiFi
    """

    config_commands = [
        f'wlan {wlan_name}',
        'shutdown',
        f'security wpa psk set-key ascii 0 {password}',
        'no shutdown',
    ]

    with ConnectHandler(**CiscoWLC.config) as net_connect:
        net_connect.send_config_set(config_commands)
        net_connect.save_config()

```

Code snippet 4: Method for changing WLAN password

The list *config\_commands* contains the necessary SSH commands to change the WLAN password on a Cisco WLC. Two of these commands include variables that are passed to the method as parameters. These variables are injected into the command string to make the feature dynamic.

The SSH connection is managed using the *with* statement as a convenient way to handle data streams. The statement is a context manager in python, and it deals with raised exceptions and errors. In essence, the *with* statement is just a wrapper for the *try/finally* block, which opens a connection and, if it gets interrupted by a raised exception, makes sure to close the connection properly. By using the *with* statement, better readability and less code repetition are achieved.

*ConnectHandler* is a class from the SSH configuration package Netmiko, which supports many different vendors such as Cisco. To configure the connection, the *ConnectHandler* takes some options in its constructor, which configures it to communicate with a specific device. As seen in Code snippet 3, Netmiko is set to use *cisco\_ios* in this case, as well as using the WLC IP, username, and password from the settings file. The SSH session is named *net\_connect* during its lifetime and is used to send commands to the WLC. Finally, a *save\_config()* gets sent, which

---

tells the WLC that the running configuration file should also be written to the startup configuration file. This is a Cisco-specific way of ensuring configuration persistence by saving it to non-volatile storage on the device.

Another feature of the CiscoWLC service is changing the SSID of a WLAN. There is currently no way of renaming an existing SSID on a Cisco WLC. Thus, the solution is to delete the WLAN and create a new one with the new name. However, since the WLAN might contain important configurations that should be kept, the backend first needs to retrieve the running configuration of the WLAN before deleting it. A new WLAN with the passed name can then be created by modifying and using the retrieved config. Code snippet 5 shows the implementation of this solution.

The running configuration is retrieved using Netmiko and saved in the *output* variable. The data is received as a long string with line breaks indicated by `\n`. Since netmiko expects a list, this data needs to be split on the `\n` characters. The SSID name is declared as the fourth word on the first line of the configuration file. Thus the first element of the list gets split on spaces and rebuilt using the new SSID name. Once the configuration file is finished, the existing WLAN gets deleted before a new one gets built using the altered configuration file.

---

```

@classmethod
def change_ssid(
    cls, wlan_name: str, new_ssid: str
) -> None:
    """
    Changes the SSID of a given WLAN
    Parameters:
        wlan_name: The WLAN that should be renamed
        new_ssid: The new SSID name of the WLAN
    """

    with ConnectHandler(**CiscoWLC.config) as net_connect:
        output = net_connect.send_command(
            f'show run wlan {wlan_name} '
        )

        running_config: list = output.split('\n')
        running_config[0] = (
            running_config[0].split(' ')[0]
            + ' '
            + running_config[0].split(' ')[1]
            + ' '
            + running_config[0].split(' ')[2]
            + ' "'
            + new_ssid
            + '"'
        )
        running_config.append('no shutdown')

        output = net_connect.send_config_set(
            [f'no wlan {running_config[0].split(" ")[1]}']
        )
        output += net_connect.send_config_set(running_config)
        output += net_connect.save_config()

```

Code snippet 5: Method for changing WLAN SSID

---

## 5.3 Frontend

The frontend of RFC is built using React, which utilizes custom and pre-defined components to build user interfaces. As described in Section 4.3, RFC utilizes the Bifrost design system, which is a set of React components that conform to Intility's design guidelines. This subsection will show some of the components that RFC is built with and how the frontend communicates with the backend.

```
import { Card } from "@intility/bifrost-react";

const FeatureCard = (props) => (
  <div className={"bfc-base-3-bg"}>
    <Card
      align={"center"}
      onClick={props.open}
      className={"card-hover bfl-padding"}
    >
      <Card.Logo icon={props.icon} />
      <Card.Title>{props.feature}</Card.Title>
      <Card.Content>{props.description}</Card.Content>
    </Card>
  </div>
);

export default FeatureCard;
```

Code snippet 6: React component for the feature card in the menu

Code snippet 6 shows the implementation of a custom component called *FeatureCard*. This component is used to display the clickable cards in the menu of RFC. An example of these cards can be seen in Figure 13a. This component contains the structure of an icon, title, and description and is a wrapper for the *Card* component imported from Bifrost. The main reason for creating this component was to achieve reusability and flexibility to add new features, which was also a part of the project requirements. This reusability is achieved by utilizing the *props* argument, which becomes an object that can be assigned different properties. These properties can be data like text, images, and functions. The *FeatureCard* component takes properties for the icon, title, and description, as well as a function for the *onClick* action. In this case, the passed-in function is *props.open* which will be an event that is responsible for showing the correct modal window once the card is clicked. An event is in simple terms a way to call a function when some action occurs, in this case, when a user clicks the button. By creating this component, new features can be added to the menu on the customer landing page simply by

---

importing the *FeatureCard* component and defining its properties. This usage is illustrated in Code snippet 7.

```
const Home = () => {
  const [
    openGuestNetwork, setOpenGuestNetwork
  ] = useState(false);
  const [openAddDevice, setOpenAddDevice] = useState(false);

  return (
    <div className="App bfl-page-padding" id="home-root">
      <h3>Change network settings</h3>
      <Grid cols={2}>
        <FeatureCard
          feature="Guest WiFi Settings"
          icon={faWifi}
          description="Change settings for your guest network"
          open={() => setOpenGuestNetwork(true)}
        />
        ...
      </Grid>

      <ChangeGuestNetwork
        open={openGuestNetwork}
        setOpen={setOpenGuestNetwork}
      />
      ...
    </div>
  );
};
```

Code snippet 7: Customer landing page with usage of the *FeatureCard* component.

Code snippet 7 shows the layout of the customer landing page. Screen responsiveness is achieved by using the *Grid* component from Bifrost, as well as a Bifrost class for dynamic page padding. This page also makes use of the *FeatureCard* component created in Code snippet 6, and the custom properties for each feature are passed in respectively. The icons come from a large icon library called Fort Awesome [42], which Intility has a license to use. An important part of this snippet is opening and closing the modal window using React hooks [43]. React hooks is a way to share state between components. When a hook is created, it will return two values, the first is the current state of the hook, and the second is a function for changing this state. Using the *open* property on the feature card, which is an

---

alias for the *onClick* event, it is possible to change the state of *openGuestNetwork* through the function *setOpenGuestNetwork*. The addition of *set* in front of the state name is a naming convention for React state hooks.

```
import { useQuery } from "react-query";
import { authorizedFetch } from "@intility/react-msal-browser";

export default function useRequests() {
  return useQuery("requests", () =>
    authorizedFetch(
      "http://localhost:8000/rfc-dev/api/v1/requests", {
        method: "GET",
      }).then((res) => res.json())
  );
}
```

Code snippet 8: Hook for fetching all requests from the backend

Code snippet 8 shows how the data is being fetched from the backend through the use of a custom hook. The naming convention for custom hooks in React is to put *use* in front of the function name. This hook makes use of another hook called *useQuery* which is imported from React Query, which is a library used for caching and background fetching. The first argument is a name for the cache key, which makes it possible to use the data in other components by referencing it with said key. When it comes to the second argument *useQuery* expects a function to be passed, which is going to handle the actual data fetching. In this case, a helper function called *authorizedFetch* has been used. This was used instead of the default *fetch* function to add an authentication token to the request. This request then gets sent to Kong before the data from the response is returned and added to the cache.



---

## 6 Development process

This chapter will describe the process during the different phases of the project. In addition, it will outline the chosen process frameworks and how the various elements of the project are documented.

### 6.1 Process framework

The project was divided into four phases, which was beneficial because the individual phases of the project had different goals and purposes. The group decided to use different process methodologies for each phase to achieve more structure and increased efficiency.

The beginning phase of the project featured quite a few introductory meetings and onboarding courses with Intility, where we got to know their team, tools, and processes. Therefore, it made sense to complete these activities as a team in chronological order, which led to choosing the waterfall model for this phase.

The waterfall model is a traditional linear model that requires clearly defined requirements and goals before proceeding to the next step. When working with waterfall, all planning is done in advance, followed by actually doing the planned activities. When one stage is completed, the next stage can be started. As this is a linear method, all activities are only carried out once, and significant changes cannot be made during the process.

For the development phase, the group chose the Scrum framework. Scrum is a flexible and agile process framework for conducting projects, allowing for changes to be made underway. Scrum divides the projects into fixed periods called sprints, which are typically two weeks. Each sprint starts with a sprint planning, where the tasks, goals, and requirements for that sprint are decided. This is followed by the sprint itself, where the team works towards the determined goals. During the sprint, there are typically daily stand-ups which are short meetings where the members share their progress and potential difficulties. A sprint is concluded with a sprint review, where the team showcases and evaluates their results. This iterative workflow makes it possible to introduce changes to the project requirements and goals during the implementation itself to accommodate feedback or new needs. [44]

For this project, we became a part of Intility's Scrum process, where we planned sprints every other Monday, conducted daily stand-ups, and attended a show and tell with the department every Friday. There was also a Scrum evaluation called "Retro" every other Friday, where the team could evaluate the Scrum process itself. Scrum was also chosen for the report phase since writing the report is an activity that needs continual improvement in the form of refinements and implementation of feedback from our supervisor. Finally, for the end phase, the waterfall method was again chosen to accommodate sequential steps that were only going to be done

once. Figure 20 illustrates our chosen process framework throughout the project.

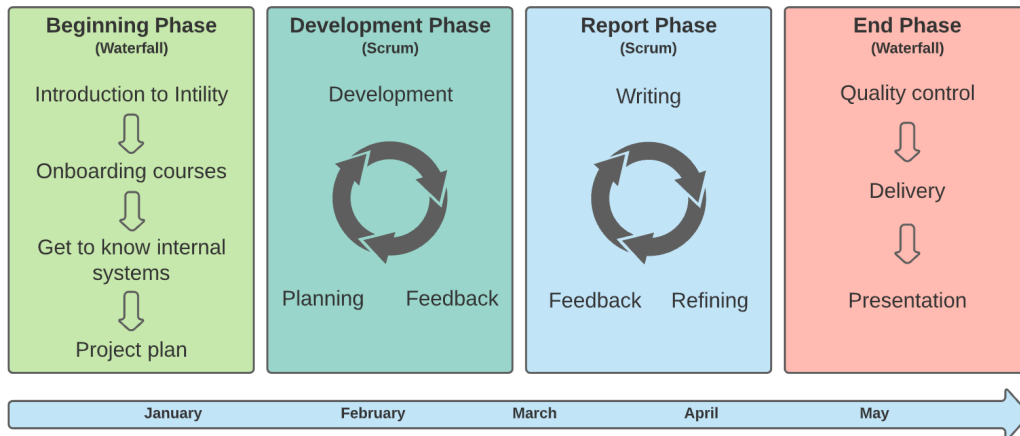


Figure 20: The chosen process framework throughout the project

## 6.2 Documentation

A vital part of large projects is the documentation of the different elements. Having good documentation makes it easier to describe the system to stakeholders, debug potential problems, and simplify further development.

The beginning phase of the project included making a thorough project plan with a schedule for the project. This schedule was visualized in the form of a Gantt chart, which was created in Microsoft Excel. In addition, we have documented all meetings, status reports, design decisions, and hour logs throughout the project. All of this information is stored in a shared Excel spreadsheet.

ID	Product / Feature	Theme	Context	Decision	Rationale
1	Thesis	Language	The language of a Thesis may impact the richness of the language and the availability of commonly used terminology within the subject. It may also impact the crowd in which the thesis may reach.	English (American)	English is a richer language and is the commonly used language in all programming. Provides easier continuity of wording and expressions. A richer language will also make it easier to elaborate on difficult themes.  Alternative: Norwegian. The writer's primary language will make writing easier, but fall short on the above mentioned aspects.
2	Frontend	Number of frontends	The frontend is the part of the application that will face the end-user.	One frontend for all users of the portal, but using OPA policies to manage end-user experience.	Only need to create one frontend for all users. OPA will allow the developer to manage the end user experience with ease and with focus on security.  Alternative: Two frontends. Would potentially allow the developer to differ between end user and admin frontends.
3	Frontend	Programming language	JavaScript VS TypeScript	Since the implementation of JavaScript is not a big part of the project and that it is possible	JavaScript is a language that has a large community and there are lots of examples online.

Figure 21: Thesis workbook in Excel.

---

For the individual text-based documents, we have mainly used Microsoft Word. For the main report, we have used Overleaf [8], which is an online LaTeX editor. The main reason why we chose to use Overleaf was because of the powerful and easy-to-use live collaboration tools, as well as the built-in version control. In addition to this, NTNU provides a template for LaTeX that makes writing a thesis more convenient and gives the report a professional look. The template contains several configuration settings and packages that are helpful for writing a report. To simplify and explain concepts, we have created illustrations, which are made and stored in Lucidchart.

The code is stored in Gitlab, where all code versions are documented with a commit message. This makes it easy to look up any changes and see precisely what changes were made, when they were made, and who made them. The code is documented using python docstrings for all functions and comments to explain important or complex code snippets further. Figure 22 shows how PyCharm uses the docstring of the *change\_password* method to present a tooltip with its description and expected parameters when hovered.

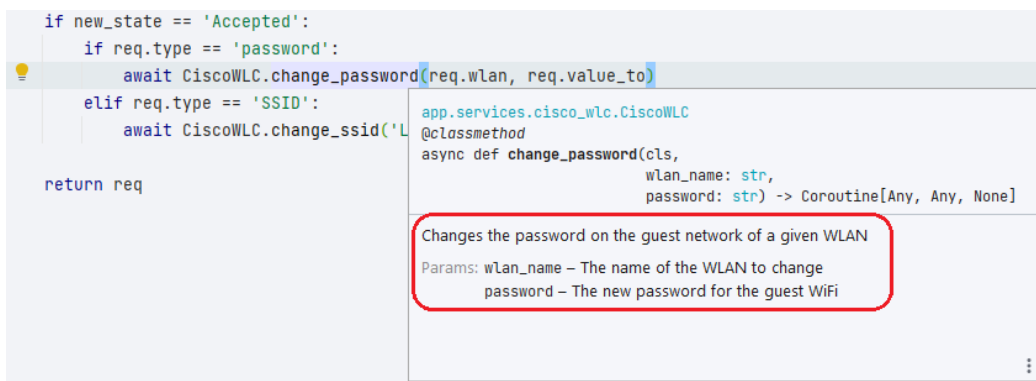


Figure 22: Utilization of docstring for PyCharm tooltip.

---

## 7 Evaluation

This chapter contains an evaluation of how RFC performs according to the requirements specified in Chapter 3, as well as some changes that were made to RFC as a result of the evaluation.

### 7.1 Evaluation of functional requirements

In this section, we restate each functional requirement and provide the corresponding evaluation result.

#### 7.1.1 Programming languages

**Requirement description:** RFC should be a web application with a frontend developed in React in conjunction with JavaScript and a backend made in Python. The main reason for these requirements is to ensure that the system can be easily developed further by Intility at a later stage.

**Evaluation result:** The requirement has been followed in the development of RFC. As discussed and shown in Chapter 4, RFC was implemented as a web application in the browser. The frontend is implemented in React in conjunction with JavaScript, and the backend is implemented in Python.

#### 7.1.2 API

**Requirement description:** RFC should be an API-driven application, where the frontend communicates with a REST-API backend using fetch requests. This API should be created using the FastAPI framework and provide several valuable endpoints.

**Evaluation result:** RFC was built as an API-driven application, with a frontend that fetches information from the backend through a REST-API written with the FastAPI framework. Figure 23 shows the automatically documented endpoints in Swagger.

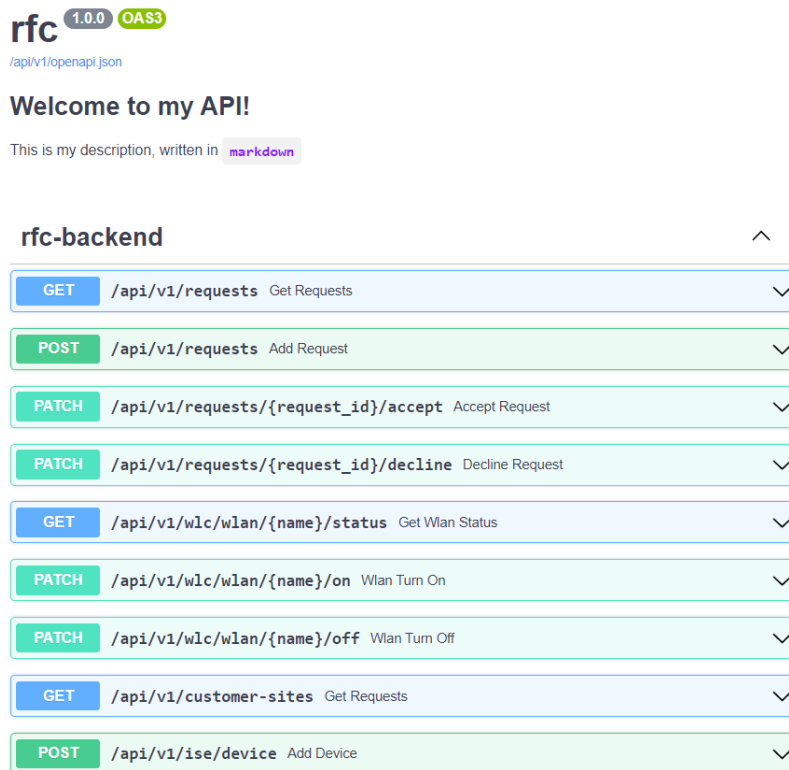


Figure 23: Auto-documented endpoints in Swagger

### 7.1.3 Kong

**Requirement description:** The Kong API gateway should be implemented to allow communication between the frontend and backend.

**Evaluation result:** To allow communication between the secure zones, RFC communicates through the Kong API-Gateway. Currently, an instance of Kong runs locally in a Docker container. However, in the future, RFC will communicate through a common Kong instance for all APIs at Intility.

### 7.1.4 UI design

**Requirement description:** Intility requires that their general developers' guidelines regarding UI design should be followed. This should be ensured by using the Bifrost design system.

**Evaluation result:** As requested, the user interface was built using Intility's design system, i.e., Bifrost. Please see Section 4.3 for more details. In order to follow the guidelines for code structuring, a plugin called Prettier [45] was used for formatting the frontend code, while changes to the backend code were run through

---

the pre-commit tool before pushing to git. Figure 24 shows the pre-commit tool checking the finished backend code with every test passing.

```
>> rfc-backend git:(main) 13:44 git commit -m "test"
black.....Passed
Check for case conflicts.....Passed
Fix End of Files.....Passed
Trim Trailing Whitespace.....Passed
Check python ast.....Passed
Check JSON.....(no files to check)Skipped
Check for merge conflicts.....Passed
Detect Private Key.....Passed
Fix double quoted strings.....Passed
flake8.....Passed
isort.....Passed
mypy.....Passed
```

Figure 24: Pre-commit checking backend code.

### 7.1.5 Policies

**Requirement description:** RFC should use OPA to enhance security and enforce policies. This will, for instance, be responsible for checking that only authorized technicians can approve a given request.

**Evaluation result:** In order to ensure that only authorized technicians can approve a request, OPA was implemented for the authorization of API calls. Furthermore, since Kong already handles each API call, OPA was configured as a plugin for Kong. This way, for each API call Kong receives, it will check with OPA that it is allowed before forwarding the API call.

### 7.1.6 Features

**Requirement description:** RFC should be able to trigger several different features once a request is approved. The scope of this project includes implementing a feature for changing network SSID and password using Cisco DNA Center, as well as adding a new device to the network identity group using Cisco Identity Service Engine.

**Evaluation result:** According to the requirement, we have implemented RFC so that it is able to trigger several different features once a change is approved in the core request system. The core RFC system was developed with flexibility in mind and can be extended with new features in the future. Following this project's scope, the currently implemented features are changing network SSID, changing network, toggling WLAN state, and adding a new device to the network identity group through ISE. They are shown in Section 4.2. The WLAN features were built using SSH rather than DNAC, which was the original requirement. This is explained in Section 4.2.1

---

## 7.2 Evaluation of non-functional requirements

In this section, we restate each non-functional requirement and provide the corresponding evaluation result.

### 7.2.1 Security

**Requirement description:** RFC should follow good security practices. For instance, the authentication should be done server-side.

**Evaluation result**

In order to fulfill the requirement for good security practices, RFC was designed according to the Secure Coding Practices Quick Reference Guide [46], published by the Open Web Application Security Project (OWASP)[47]. OWASP is a nonprofit foundation that works to improve software security, and their guide has been a valuable resource to confirm that all fundamental security practices are followed in RFC. The guide covers essential actions to consider when processing, transferring, and storing data. Even though the guidelines were written in 2010, they are still relevant today. Figure 25 shows an example of the OWASP guidelines and how we can use them to verify that our system is secured to basic standards.

**Access Control:**

- Use only trusted system objects, e.g. server side session objects, for making access authorization decisions
- Use a single site-wide component to check access authorization. This includes libraries that call external authorization services
- Access controls should fail securely
- Deny all access if the application cannot access its security configuration information

Figure 25: OWASP guidelines for access control.

As seen in Figure 25 the first two points state the importance of having authorization done exclusively on systems running in trusted environments. Because of this, the RFC frontend can not be used to deal with authorization on its own, since the end-users can manipulate the source code. Therefore the frontend needs to get permissions from somewhere else. Thus, the authentication in RFC is handled by the Microsoft Authentication Library (MSAL) [48] in conjunction with Azure AD, while Kong and OPA do the authorization.

By doing this, the authorization process is performed externally from the application itself and can be abstracted away from the application logic as illustrated in Figure 26. Additionally, by sending all communication through Kong, the requests can be authorized by the OPA plugin. The current development environment utilizes a local Kong instance. However, in the future, it will utilize a centralized Kong instance that will handle authorization for all applications at Intility.

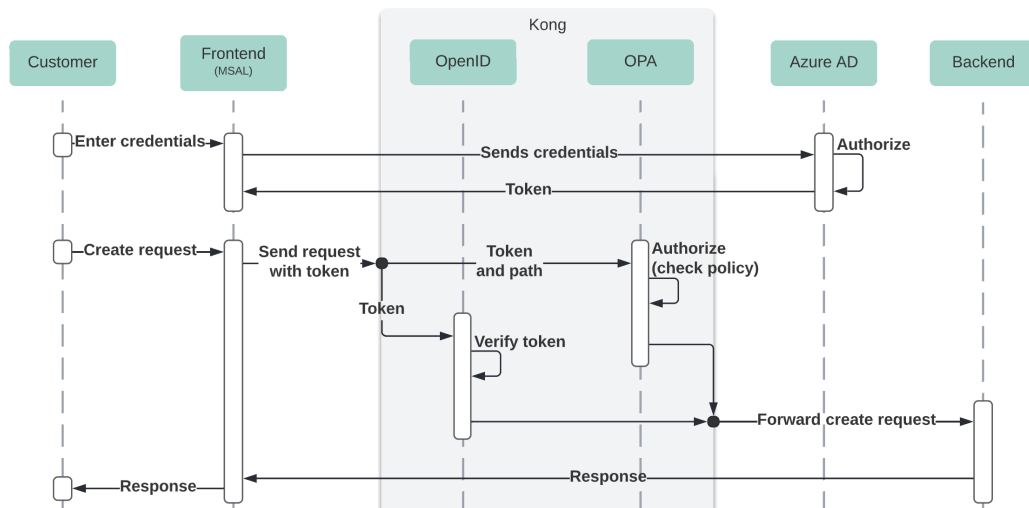


Figure 26: Authentication and authorization flow.

With the OPA solution, the authorization meets the OWASP guidelines by doing it server-side and using a single site-wide system to do the authorization. According to the third and fourth points in Figure 25 this is handled by the way the OPA plugin is configured as a requirement in Kong, which means if Kong can not communicate with OPA, it will stop all communication to any services which requires it. Another way is how the permission logic is set up. OPA will check the token and its roles against a list that describes which endpoints each role has access to and the HTTP method they can use on it, see Figure 26. If OPA can not find the endpoint which is being accessed under the role the user has in the token, the request gets denied. An alternative to using Kong and OPA would be implementing authorization directly in the backend. This would be a possible solution and is a common way of doing it since the backend runs in a safe and controlled environment, such as Azure.

With the increasing focus on privacy for consumers, we wanted to research how general GDPR policies would apply to our system. RFC neither grants access to new data for Intility employees nor for additional people to see this customer data. There is also no personal data stored in the database of RFC. Hence, this system is already compliant with the current regulations that the customer and Intility have agreed on. Therefore, both the group and the supervisors at Intility regard this project as GDPR compliant.

The group had discussions regarding encrypting the WLAN passwords in the database, which are being stored as long as the technician has not approved or denied a request. Currently, the passwords are handled in cleartext both when submitted to Intility and when a technician updates the password on the relevant WLAN. It is important to note that once the application is deployed to production, communication will be done through HTTPS, which means it will be encrypted when traveling over the network. The database is also running in Azure, where all the



---

data stored will be encrypted, so doing it in RFC is unnecessary. Implementing encryption of the WLAN password in RFC would also be unnecessary since the clear text version of the password is handled in other parts of Intility's value streams. As a result of this, we did not see the value of implementing encryption, which was thus given a low priority.

### 7.2.2 User experience

**Requirement description:** The interface should be intuitive and natural to use in order to allow the user to move around without feeling confused or lost.

#### **Evaluation result**

Throughout the design process, careful choices around the user interface were made to optimize the user experience. This was important to let the users navigate the system without feeling cluttered or getting lost.

In order to verify that RFC fulfills this requirement, a user testing was conducted with employees at Intility that will use the system daily, as well as with less technical employees in order to simulate user testing with the customers. According to the principle of ethnography, the user testing was conducted in the form of an in-person session at their workstations in order to see how the system performs under its natural circumstances.

The user testing consisted of two parts and was recorded but anonymized due to subject privacy. The first part consisted of a user observation using the think-aloud protocol, which involves asking the subject to verbalize their thoughts while performing predefined tasks. In order to quantify the test results, the elapsed time and number of button clicks the subject needed to perform each task was logged. The second part of the user testing consisted of a structured interview and an unstructured part where the user was asked questions about their experience of using the product and its current features. This way, both quantitative and qualitative feedback was received.

The optimal scenario would have been to get all the technicians at Intility to attend user testing. However, due to limited resources and a tight schedule, this was not feasible. Therefore, the test was conducted on a pseudo-random group of five technicians, covering a range of ages and competencies. This reduced the scope of their feedback but at the same time, allowing for a deeper and more qualitative understanding of what a technician thinks about the system.

#### QUANTITATIVE RESULTS

The quantitative feedback was logged as the number of button clicks and elapsed time to complete each task. Based on the average values, RFC performs reasonably well, with the most time-consuming tasks being the feature for creating a new SSID request and adding a new device to the network. However, these are reasonable values due to the amount of information that needs to be provided by the users. The results from the user observation are presented in Tables 1, 2, 3, and 4.

---

<b>Task</b>	User 1	User 2	User 3	User 4	User 5	<b>Average</b>
Log in	4	5	3	4	3	3
Find all pending requests	5	3	3	4	3	3
Which customer created a password change request to "Admin123"	6	5	3	4	3	4
Accept SSID change for "Office_Guest"	4	5	4	6	4	5
Change to dark theme	3	2	4	2	3	3

Table 1: Elapsed time for each technician task in whole seconds

<b>Task</b>	User 1	User 2	User 3	User 4	User 5	<b>Average</b>
Log in	2	2	2	2	2	2
Find all pending requests	0	1	1	0	1	1
Which customer created a password change request to "Admin123"	1	0	2	2	0	1
Accept SSID change for "Office_Guest"	2	2	2	1	1	2
Change to dark theme	3	2	2	2	3	2

Table 2: Number of button clicks for each technician task

<b>Task</b>	User 1	User 2	User 3	User 4	User 5	<b>Average</b>
Log in	4	6	4	11	3	6
Create new SSID request	37	41	52	48	57	47
Find the number of connected clients	21	7	11	8	17	13
Turn off guest WiFi	3	5	3	6	3	4
Add new device to network	45	43	57	52	43	48
Change to dark theme	3	13	7	3	6	6

Table 3: Elapsed time for each customer task in whole seconds

---

Task	User 1	User 2	User 3	User 4	User 5	Average
Log in	2	2	2	2	2	2
Create new SSID request	9	7	12	11	8	9
Find the number of connected clients	3	3	5	2	4	3
Turn off guest WiFi	3	2	4	3	3	3
Add new device to network	14	11	13	11	15	13
Change to dark theme	3	3	2	4	3	3

Table 4: Number of button clicks for each customer task

#### QUALITATIVE RESULTS

Based on the results of the user interviews, all the qualitative feedback has been combined into an aggregated list of bullet points. This includes both the positive feedback as well as suggestions for improvement.

Positive feedback:

- Both the customers and the technicians emphasized that the interface looks neat and is simple to use. In addition, the design is similar to the rest of Intility’s websites and portals, which makes it familiar and natural for them to maneuver.
- All the users reported the need for such a system, and they all saw the potential it has to improve and streamline the process of making simple changes to the networks. In addition, many of the users reported that the system could provide value to them even in its current state.
- Many of the users enjoyed the opportunity to choose between light and dark mode to suit their preferences. Especially the technicians enjoyed the dark mode, which makes it possible to adapt the system to fit their work environments.
- The adaptive screen size and support for different devices was a feature that the technicians particularly welcomed. The technicians are currently dependent on their computers to perform an SSID or password change on a customer’s network. RFC will enable the technicians to approve a network change from almost anywhere, which could significantly decrease the processing time of such a request.

---

Improvement suggestions:

- Two of the technicians pointed out a wish to see the name and company of the user that has created a request. This information would be beneficial in special cases where the technicians must contact the customer for additional details. Currently, the system stores and displays customer information in the form of a user-id that is somewhat meaningless for humans and could be replaced by the name and company of the user to improve readability. One of the technicians also mentioned that it would be nice to see information about what company a request belongs to without expanding the row.
- One of the technicians had concerns about how the system works when a customer has more than one guest WLAN. For this edge case, it would have been nice to have the opportunity to choose a specific WLAN from a list of all the available ones.
- Some of Intility's customers have offices at several locations worldwide. In many cases, these offices use the same WLAN and WLC. Thus a change at one location will affect all the other ones. The customer interface should display a warning about this once a customer tries to submit a change request to avoid confusion across different locations.
- During testing of the customer interface, we noticed that a user had difficulties creating a new SSID change request. After further discussion, it came to our attention that they were not familiar with the term "SSID".
- A technician mentioned that it would be beneficial to have filtering and search functionality in the list of handled requests. As time goes on, the log of handled requests may become quite long. Thus, pagination, search, and filtering should be implemented to simplify the process of finding a previously handled change request. A customer also requested access to a similar interface, making it possible to see a log of their previous requests.
- Intility mainly strives to perform all changes during a specific maintenance window and document all planned maintenance in a dedicated portal. One of the technicians voiced concerns about how the system currently executes a change immediately after the technician approves it. Instead, we were advised to put the approved request in a queue that gets applied to the WLC during the maintenance window. It would also have been beneficial if this queue of planned WLC changes were automatically documented in the existing maintenance portal. A similar feature request was made during user testing with a customer. They wanted a feature for scheduling when a change were to be performed in order to get a more predictable behavior of their network. This could be implemented by adding a timestamp field to the "create request" form.

- 
- One of the customers raised concerns about the impact of accidentally submitting a change request or a request containing wrong information. The customer thus wished for an opportunity to cancel or change a submitted request. This could be implemented in conjunction with the previously mentioned schedule feature and allow a customer to undo the submitted request at any time before the maintenance window.
  - When a customer opens a request, it should preferably be logged as a case linked to Intility's existing service ticket system, which houses all cases related to a customer. A technician mentioned that it would be nice if the ID of a request in RFC were synced with the ticket ID, making it possible to click on the ID and be redirected to the corresponding case in the service system.
  - In some cases, it could be helpful for an Intility employee to have the opportunity to create a change request on behalf of a customer. These could, for instance, be cases where the customer contacts support for another issue and mention the wish for a network change as well.
  - One of the technicians mentioned that the current solution for changing the SSID of a WLAN would not work on guest networks that use a captive portal. A captive portal is an alternative authentication method where the WLAN itself does not have a password but instead utilizes a popup window that displays information on how to connect. Such solutions are often present on public guest networks like hotels and restaurants. The current solution changes the WLAN password directly on the WLC. However, renaming a captive network would need to be done using ISE. Although most guest networks on Intility's platform use a traditional WPA2 password, the RFC backend would need to be extended to support captive networks before potentially getting deployed to the customers.
  - In some cases, the technicians have been asked to turn off the guest network for various reasons. As a result, a dedicated button has been implemented, but this feature currently allows a customer to turn off the guest WiFi without any involvement from Intility. Three of the technicians had concerns about giving this power to the customer. They suggested implementing the feature as a request requiring approval, similar to the SSID and Password changes.

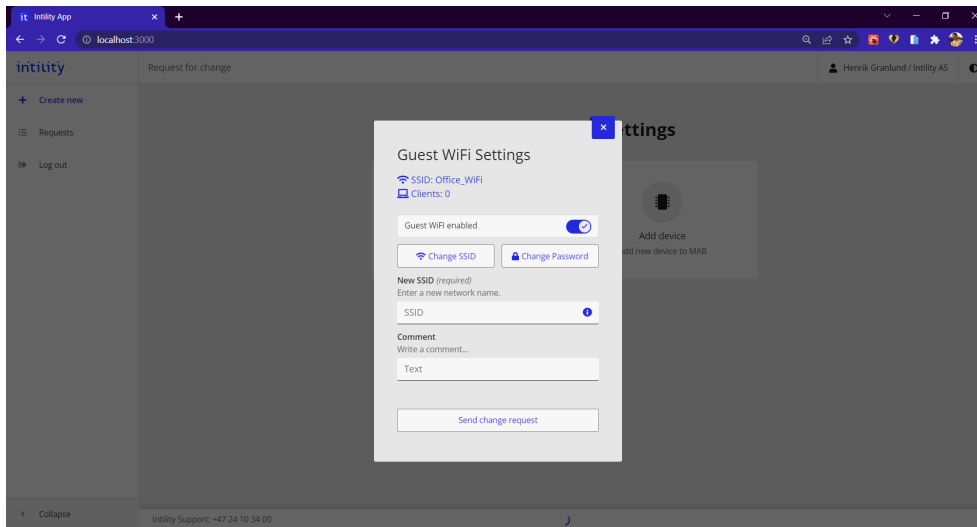
---

### 7.2.3 Adaptive to different screen sizes

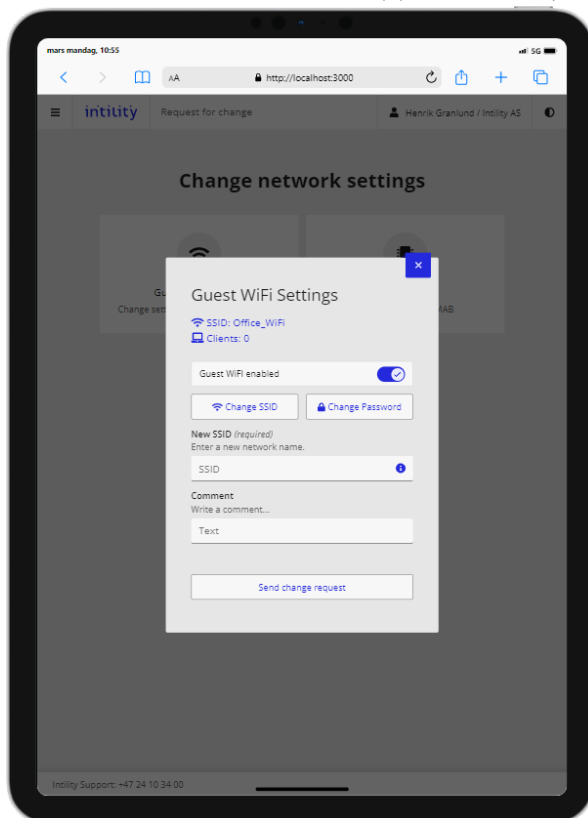
**Requirement description:** The interface should be adaptive to different screen sizes to facilitate the different work environments of the end-users. A technician might work on a desktop, tablet, or mobile. Thus, the application should support all of these form factors.

#### **Evaluation result**

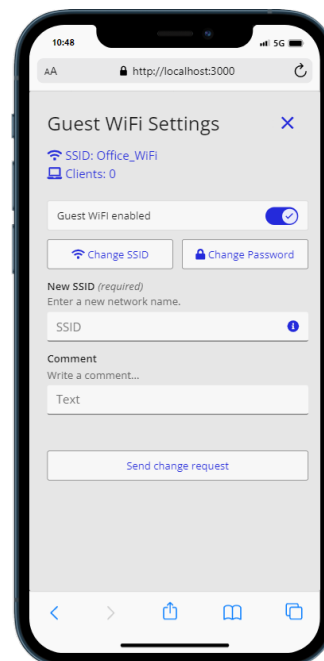
In order to make the experience as user-friendly as possible regardless of work style and environment, there are implemented breakpoints for different screen sizes. Depending on how large the screen is, different elements like the sidebar are either shown or hidden. The breakpoints are designed to accommodate three different types of devices; Desktop, tablet, and mobile. However, each of these size levels is entirely flexible and dynamically scales and positions the elements on the screen according to the current resolution. Figure 27 shows the function for changing guest WiFi settings on the three different device types.



(a) Full screen / PC



(b) Medium sized screen / Tablet



(c) Small screen / Mobile

Figure 27: *Guest WiFi settings* interface on different device types.

---

## 7.2.4 Responsiveness

**Requirement description:** The interface should be responsive and perceived as fast. This should be solved by implementing performance techniques like caching and loading animations to visualize necessary background activity.

### Evaluation result

In order to enhance the user experience, RFC was designed to be responsive and perceived by the users as fast. According to the Nielsen Norman Group [49], there are three important time limits to consider when optimizing application performance. Firstly, the limit for having the user feel that the system is reacting instantaneously is about 0.1 seconds. Furthermore, the limit for users' flow of thought to stay uninterrupted is about 1 second. The absolute limit for keeping a user's attention is 10 seconds, and if an action takes more than this, they will want to perform other tasks while waiting. In these cases, a percent-done indicator would be beneficial. At the same time, actions that take between 2 and 10 seconds should rather use a simple "busy" indicator to avoid violating the principle of display inertia, which says that having many changing and flashing elements on the screen makes the user stressed and unable to keep up with the system. The Nielsen Norman Group also clarifies that a system might react too quickly, making the user unable to keep up with the system. In these cases, a fake loading animation might be beneficial [49].

RFC has been designed with these time limits in mind. In order to create the feeling of instantaneous response, loading animations are triggered within 0.1 seconds of clicking a button. For actions that lead to a context switch to another window, an artificial delay of 1 second is added to allow the users to keep up with the system's actions. Furthermore, each action was designed with the goal of at least taking less than 10 seconds in order to keep the users' attention. To verify that the system performs to our expectations, a test was conducted on the two core features of RFC: creating a change request and approving such a request. Each of the actions was done ten times to ensure realistic results. The results were as follows:

Task	1	2	3	4	5	6	7	8	9	10	Average
Submit request	3.7	1.2	1.4	2.1	0.9	2.2	1.2	2	1	1.4	<b>1.71</b>
Approve request	5.7	6	5.5	4.9	6.1	5.8	5.2	6.2	4.9	5.5	<b>5.58</b>

Table 5: Loading time in seconds to submit and approve a request.

The test uncovered a pleasing result for submitting a request but a somewhat high average loading time for approving a request. Techniques like caching of data and background loading have been implemented to mitigate the loading time. There are also implemented loading animations to give the perception of progress



---

happening. Since the average time for approving a request is between 2 and 10 seconds, this loading animation is simply a "spinning wheel" rather than a progress bar.

However, the system is still perceived as somewhat slow from the technician's point of view. This is because the frontend waits for the backend to confirm that it has received a given request before changing the elements on the screen. Since the latency for communicating with the back-end and the WLC is somewhat high, this has a negative impact on the user experience. A possible solution would be to immediately display a change as "performed" in the interface, even though the change is not entirely done in the backend. By implementing this, the users will rarely experience any loading screens, as long as their network connection is stable. If the backend and frontend communication fails, the transaction can be reverted.

### 7.2.5 Reusability

**Requirement description:** The code should be easily reusable for future development. This should be achieved by using the same languages, frameworks, and code structures that the developers at Intility are familiar with. There should also be a modular approach for each service, and the different components of the code should be written with reusability in mind where possible.

**Evaluation result:** One of the requirements was to make sure the system and its codebase could be re-used or further developed by Intility in the future. By using the same languages, frameworks, and tools that their developers prefer, the project structure and code should be relatively easy to understand for a developer at Intility. This was verified by asking a DevOps engineer at Intility to make minor changes to the system, which they were able to do effortlessly.

## 7.3 Implemented changes

Based on the feedback received during the user testing, a few enhancements were made to the system. These changes were mostly affecting how information is presented to the user, such as showing the name of the company on the requests overview page as well as the name of the customer and technician involved with the request. The original implementation displayed information about the user in the form of a user-id number, but this was changed to improve human readability. The updated solution added information about the user as extra fields in the database schema, which are automatically filled out based on the token from the logged-in user. Additionally, the company information was previously only available when the row was expanded but is now moved to the top level so that the technician can get the information without extra navigation, speeding up the decision process. Figure 28 shows the updated technician interface.

ID	Type	Company	Value to	Created	Last updated	Status
1	SSID	AE - Wold AS	Wold Gjestenett	12:22   27.04.22	12:25   27.04.22	Pending
5	SSID	AB - Regnskap AS	Lobby_Guest	09:30   28.04.22	09:30   28.04.22	Pending
9	SSID	SB - Telefonservice AS	Butikk-Gjest	10:58   03.05.22	10:58   03.05.22	Pending

Figure 28: Redesigned technician interface containing company name

The user testing also revealed that not everyone knew the term "SSID", which led to some confusion for certain users. As a result of this, the terminology in the customer interface was changed from "SSID" to "WiFi name" to facilitate the system for users with less technical knowledge. Additionally, the feature making it possible to toggle the guest WiFi on or off was disabled to avoid customers accidentally turning it off at an inappropriate time.

Another feature that was added after getting feedback from the technicians was the ability for an Intility employee to make change requests on behalf of a customer. This was implemented as an input field with live search functionality. Figure 29 shows the interface of the "Guest WiFi settings" for an Intility employee.

**Guest WiFi settings**

**Customer**  
Select customer to make request on behalf of

A|

- AB - Regnskap AS
- AC - Verksted AS
- AD - Advokatbyrå AS
- AE - Wold AS

Change name    Change password

Figure 29: Interface for making a request on behalf of a customer

---

## 8 Closing remarks

This chapter will outline the future work that could be done in the project. The learning outcome and conclusion of the project are also included.

### 8.1 Future work

Due to the limited time constraint of the project, there are several features, improvements, and technicalities that were not prioritized. Nevertheless, these are changes that we have deemed beneficiary in the future development of this project.

#### **More flexible backend architecture**

During user testing, concerns were raised about how dynamic and flexible the system will be for edge case scenarios and highly customized network architectures. Currently, there is no logic to accommodate requests for customers that have multiple guest-WLANs, customers with captive networks, or customers with several locations. Considering the current trajectory of increasingly complicated network architectures at Intility, there is necessary to implement support for this before deploying RFC to production and making it available for Intility's customers.

#### **Connect to existing service system**

Furthermore, the system does not currently offer any communication with the existing ticket and service system that Intility uses. Although RFC is supposed to alleviate some workload of the customer support team, it is still suggested that an automated support ticket appears in the service system such that these changes are documented in case the customer contacts Intility's support regarding the modifications performed on the network. These tickets should have the same id and hyperlink to the request in RFC.

#### **Search functionality**

Another important feature that was requested is support for search and filtering functionality in the log of handled requests. Currently, all handled requests remain as a list on the same page which can grow to become quite long over time. By implementing a powerful search field and pagination for the results, the technicians can quickly find details about a previously handled request should the need arise.

#### **Scheduled changes**

In the current solution, the approved change requests get applied instantly. Unfortunately, this leads to a somewhat unpredictable behavior for the customers since they do not know when a technician might approve their request. To alleviate this, a suggestion was to implement a maintenance window where all approved changes get performed. In addition, the form for creating a new change request could contain a field for when the change should be performed, where the options are either to select a timestamp for the change or to choose the nearest scheduled maintenance window. The approved and planned changes should also be logged in Intility's maintenance portal, which could be beneficial for the support department

---

in cases where a customer might call because of network issues due to the changes. Another concern from a customer was that there was no option for them to cancel a request. This could be implemented with the schedule feature mentioned above, allowing a customer to cancel their request at any time until the registered timestamp for the change.

## 8.2 Learning outcome

The project has been a valuable learning experience for the group members. According to the course description of the subject [50], the bachelor thesis should conclude the study and combine important parts of the scientific content of the study program. The project fits well with this since it contained relevant topics such as network infrastructure, software development, and security. The course page also outlines what the learning outcome of the subject should be in regards to gained knowledge, skills, and competence.

During the course of this project, we have acquired substantial experience in software development within authentication, authorization, and cloud services, we have also learned entirely new concepts like React, API development, and Software-defined Networking which has been both challenging and rewarding. Furthermore, we have been exposed to working in a larger team environment where we have experienced the importance of working systematically to achieve a satisfying solution to a complex problem statement. Throughout the project, we have been in continuous contact with numerous external stakeholders to create the specification and requirements for the system. By doing this, we have learned to have a systematic approach to the continuous evaluation of the project's progress.

We have also gained skills in systematically documenting and presenting the project results and using relevant literature and documentation. In addition, we have gained experience in solving a delimited problem by developing a customized solution for a general problem statement.

---

### 8.3 Evaluation of the project

Conducting a project as a form of work has been educational and provided valuable insights into how larger projects are done in the industry.

The schedule and organization of the project have worked well. By comparing the conducted work against the proposed GANTT chart in the project plan, we can see that most project tasks have been done in the allocated time frame. The GANTT chart, supplemented by displaying issues in Trello and working in Scrum sprints, turned out to be a good form of workflow management for this project. It has been an enjoyable experience working with Scrum in practice, especially since Scrum, in many ways, is considered the industry standard framework for software projects. The likelihood of encountering Scrum in the industry at a later stage in life is high, which makes the experience from this project even more beneficial.

When it comes to the assigned roles, they turned out to have less impact on the work distribution than initially thought, as the distribution of work happened evenly and naturally. The group has also utilized a lot of live collaboration during the different work to share knowledge and ensure that each member gained a good understanding of the various elements of the project. As a result, the total amount of hours allotted for the project was distributed evenly among the members, which is reflected in the hour log journaled at the end of each workday.

Although the assignment has been both exciting and relevant for our studies, it did turn out to contain a bit more programming than initially assumed, which led to some challenges due to our limited experience with software engineering. However, in hindsight, this challenge contributed to a greater learning outcome overall. Overall, the project has provided great insight into how a larger company works and has contributed to building a social and professional network. Overall the group is pleased with the project and how it contained a practical approach to a real-world and complex problem, allowing us to apply both old and new theoretical knowledge.

---

## 8.4 Conclusion

Today, when a customer wants to make changes to their networks, the request must go through customer support, which can be time-consuming due to the manual work and communication between the customers, support, and technicians at Intility. This led to the following problem statement:

”How can Intility streamline, automate and propagate requests for network changes in their secure zone model?”

By developing a system that provides a graphical user interface with clear and simple terminology, the customers themselves can describe a wanted change with sufficient details so the request can be directly passed to a technician for review. This removes the need for customer support to be involved and saves time by reducing the need for communication between the customers and technicians. By standardizing the format of these requests, the possibility of automating the changes is also achieved, which reduces the chance of human errors. To allow this automation to happen across different secure zones, Kong was implemented as a central hub between the zones and utilizes various plugins for added security. This provided a better data flow than Metro would because the RFC frontend can use the backend API directly instead of listening to changes and interpreting them from a message queue.

In general, automating simple tasks entails increased efficiency, in addition to allowing Intility to focus on more challenging tasks that require human interventions. After discussions with the product owner, he could clearly see the developed system as a good foundation for a future self-service portal for Intility’s customers.

---

## Bibliography

- [1] Intility, “An industrialized it platform, delivered as a service.” [Accessed March 3rd, 2022] Available <https://intility.no/en/the-solution/>.
- [2] Kong Inc., “Kong gateway.” [Accessed February 2nd, 2022] Available <https://docs.konghq.com/gateway/>.
- [3] Meta Inc., “React - a javascript library for building user interfaces.” [Accessed February 7th, 2022] Available <https://reactjs.org/>.
- [4] Cisco Systems, “Cisco dna center at-a-glance.” [Accessed February 3rd , 2022] Available <https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/dna-center/nb-06-cisco-dna-center-aag-cte-en.html>.
- [5] Cisco Systems, “Cisco identity services engine.” [Accessed March 14th , 2022] Available <https://www.cisco.com/c/en/us/products/security/identity-services-engine/index.html>.
- [6] Microsoft, “Microsoft 365.” [Accessed March 29th, 2022] Available <https://www.microsoft.com/nb-no/microsoft-365/products-apps-services>.
- [7] Atlassian, “Trello.” [Accessed March 9th, 2022] Available <https://trello.com/tour/>.
- [8] Overleaf, “Overleaf, online latex editor.” [Accessed May 2nd, 2022] Available <https://www.overleaf.com/>.
- [9] Intility, “Bifrost design system.” [Accessed February 2nd, 2022] Available <https://bifrost.intility.no>.
- [10] The PostgreSQL Global Development Group, “Postgresql: The world’s most advanced open source relational database.” [Accessed March 31st, 2022] Available <https://www.postgresql.org/>.
- [11] IBM, “Similarities and differences in the uses, benefits, features and characteristics of postgresql and mysql.” [Accessed April 29th, 2022] Available <https://www.ibm.com/cloud/blog/postgresql-vs-mysql-whats-the-difference>.
- [12] Microsoft, “What is azure.” [Accessed April 29th, 2022] Available <https://azure.microsoft.com/nb-no/overview/what-is-azure/>.
- [13] M. Bayer, “A database migration tool for sqlalchemy..” [Accessed April 29th, 2022] Available <https://pypi.org/project/alembic/>.
- [14] M. Bayer, “The python sql toolkit and object relational mapper.” [Accessed April 28th, 2022] Available <https://www.sqlalchemy.org/>.

- 
- [15] “Object–relational mapping.” [Accessed March 24th, 2022] Available [https://en.wikipedia.org/w/index.php?title=Object%E2%80%93relational\\_mapping&oldid=1072364996](https://en.wikipedia.org/w/index.php?title=Object%E2%80%93relational_mapping&oldid=1072364996).
- [16] S. Ramírez, “Sqlmodel.” [Accessed March 24th, 2022] Available <https://sqlmodel.tiangolo.com/>.
- [17] DBeaver community, “Dbeaver.” [Accessed March 15th, 2022] Available <https://dbeaver.io/about/>.
- [18] IBM Corporation, “Acid properties of transactions.” [Accessed April 28th, 2022] Available <https://www.ibm.com/docs/en/cics-ts/4.2?topic=processing-acid-properties-transactions>.
- [19] Docker Inc., “Developers love docker. businesses trust it.” [Accessed February 7th, 2022] Available <https://www.docker.com/>.
- [20] Docker Inc., “Use containers to build, share and run your applications.” [Accessed April 29th, 2022] Available <https://www.docker.com/resources/what-container/>.
- [21] Starlette, “The little asgi framework that shines.” [Accessed March 28th, 2022] Available <https://www.starlette.io/>.
- [22] S. Colvin, “Pydantic.” [Accessed February 3rd, 2022] Available <https://pydantic-docs.helpmanual.io/>.
- [23] G. van Rossum, J. Lehtosalo, and Łukasz Langa, “Pep 484 – type hints.” [Accessed March 28th, 2022] Available <https://peps.python.org/pep-0484/>.
- [24] S. Ramírez, “Fastapi.” [Accessed February 2nd, 2022] Available <https://fastapi.tiangolo.com/>.
- [25] Django Software Foundation, “Django makes it easier to build better web apps more quickly and with less code..” [Accessed May 15th, 2022] Available <https://www.djangoproject.com/>.
- [26] Encode OSS Ltd, “Django rest framework is a powerful and flexible toolkit for building web apis..” [Accessed May 12th, 2022] Available <https://www.django-rest-framework.org/>.
- [27] Encode OSS Ltd., “Collaboratively funded software development..” [Accessed May 12th, 2022] Available <https://github.com/encode/>.
- [28] “Git.” [Accessed February 15th, 2022] Available <https://git-scm.com>.
- [29] “Simplify your workflow with gitlab.” [Accessed March 23rd, 2022] Available <https://about.gitlab.com/stages-devops-lifecycle/>.
- [30] B. Warsaw, N. Coghlan, and G. V. Rossum, “Pep-8.” [Accessed February 7th, 2022] Available <https://www.python.org/dev/peps/pep-0008/>.



- 
- [31] A. Sottile, “Pre-commit.” [Accessed March 15th, 2022] Available <https://pre-commit.com/>.
- [32] Microsoft, “Typescript.” [Accessed March 15th, 2022] Available <https://www.typescriptlang.org/>.
- [33] Microsoft, “Azure service bus.” [Accessed March 15th, 2022] Available <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview>.
- [34] Lucid Software Inc., “Lucidchart - intelligent diagramming.” [Accessed March 15th, 2022] Available <https://www.lucidchart.com/pages/>.
- [35] Cloud Native Computing Foundation, “Open policy agent.” [Accessed March 15th, 2022] Available <https://www.openpolicyagent.org/>.
- [36] JetBrains, “Pycharm.” [Accessed March 29th, 2022] Available <https://www.jetbrains.com/pycharm/>.
- [37] Microsoft, “What is azure active directory.” [Accessed March 22th, 2022] Available <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-what-is>.
- [38] K. Byers, “Netmiko.” [Accessed March 21st, 2022] Available <https://github.com/ktyers/netmiko/>.
- [39] G. Harris, “Osi model.” [Accessed May 2nd, 2022] Available [https://en.wikipedia.org/w/index.php?title=OSI\\_model&oldid=1085309819](https://en.wikipedia.org/w/index.php?title=OSI_model&oldid=1085309819).
- [40] Mozilla Foundation, “Cross-origin resource sharing.” [Accessed February 2nd, 2022] Available <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>.
- [41] OpenID, “Welcome to openid connect.” [Accessed March 25th, 2022] Available <https://openid.net/connect/>.
- [42] Fortawesome, “Fort awesome.” [Accessed March 25th, 2022] Available <https://fortawesome.com/>.
- [43] Meta Platforms, Inc, “Introducing hooks.” [Accessed May 18th, 2022] Available <https://reactjs.org/docs/hooks-intro.html>.
- [44] I. Sommerville, *Software Engineering Global Edition*. Pearson Education Limited, tenth ed., 2016.
- [45] “Prettier - opinionated code formatter.” [Accessed March 21st, 2022] Available <https://prettier.io/>.
- [46] OWASP Foundation, “Owasp secure coding practices quick reference guide.” [Accessed March 29th, 2022] Available [https://owasp.org/www-pdf-archive/OWASP\\_SCP\\_Quick\\_Reference\\_Guide\\_v2.pdf](https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf).

- 
- [47] OWASP foundation, “The open web application security project.” [Accessed May 12th, 2022] Available <https://owasp.org/>.
- [48] Microsoft, “Overview of the microsoft authentication library (msal).” [Accessed May 2nd, 2022] Available <https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview>.
- [49] J. Nielsen, “Response times: The 3 important limits.” [Accessed May 2nd, 2022] Available <https://www.nngroup.com/articles/response-times-3-important-limits/>.
- [50] NTNU, “Dcsg2900 - bachelor thesis bachelor of science in digital infrastructure and cyber security.” [Accessed May 4th, 2022] Available <https://www.ntnu.edu/studies/courses/DCSG2900#tab=omEmnet>.

---

## A Vocabulary

- API - Application programming interface.
- EIG - Endpoint Identity Group
- GUI - Graphical User Interface.
- IDE - Integrated development environment. Program with useful tools for writing code.
- Kong - API gateway and microservice platform.
- MAC address - Media Access Control Address. A unique address for an internet-connected device
- MVP - Minimum viable product.
- OPA - Open Policy Agent. An open-source policy engine that enforces policies specified with code.
- RFC - Request For Change. The internal name of the developed system.
- SSH - Secure Shell. Encrypted communication protocol.
- SSID - Service Set Identifier. Generally known as WiFi name.
- UCD - User Centered Design.
- UI - User interface.
- UX - User experience.
- VLAN - Virtual Local Area Network.
- WLAN - Wireless Local Area Network.
- WLC - Wireless LAN Controller. A device to manage other wireless network equipment

---

## B Log

### B.1 Week log

## Week Log

<b>2</b>	Introduction week. This week was spent on getting to know Intility, as well as setting up our computers with the required software and configure the environment. The week also contained a few meetings to agree on the project specifications
<b>3</b>	This week was dedicated to creating the project plan in coordination with our supervisor at NTNU. In addition to this, the frontend project was setup and we started creating the backend API endpoints
<b>4</b>	This week we have configured the system to use Kong as a gateway for all API requests. We have also implemented authentication through the Microsoft Authentication Library, which enables us to limit availability of the system based on Azure tenants. In addition we have implemented some design changes to our frontend.
<b>5</b>	This week we have implemented policy checking using OPA. We had a few problems during the configuration, but after doing some bug-hunting we found a solution. We have also started writing the report structure, as well as starting writing our tool-survey. In addition we have done some minor changes on the frontend.
<b>6</b>	The backend has been connected to DNAC and ISE, and we have started drafting the thesis. Redesign was done of the "create request" forms. Changing password now works through SSH (netmiko)
<b>7</b>	Implementation of the SSID change feature and toggle WiFi state feature (Both utilizing SSH with netmiko). Not so many productive days this due to seminar in the IØ2000 course.
<b>8</b>	Created "add device to MAB" feature, with frontend, backend and database. Customers can now register new devices to a given MAB by using a simple interface.
<b>9</b>	No work done on bachelor due to project and seminar in IØ2000, as well as sickness among the group members
<b>10</b>	Thesis writing, as well as course on thesis writing with Supervisor. The week mainly consisted of structuring and drafting of each section of the final thesis
<b>11</b>	Thesis writing. The structure was finalized, and we started filling out more on each section as well as refining drafts.
<b>12</b>	Thesis writing on the chapters about implementation and evaluation.
<b>13</b>	Refine and finalize the first thesis draft before delivery deadline to supervisor
<b>14</b>	The first half of the week was spent on thesis writing and user testing of the customer interface at Intilitys headquartes. The rest of the week was spent on the IØ2000 project.
<b>15</b>	Easter break
<b>16</b>	User testing of the technician interface at Intilitys headquartes, as well as writing about these results in the thesis.
<b>17</b>	This week was spent on programming the changes that were decided as a result of user testing, as well as fixing minor bugs in the system. After this, the system was declared finalized.
<b>18</b>	This week was dedicated to continual improvement, i.e. refining and restructuring the sections in addition to a thorough grammar review.
<b>19</b>	Revision thesis to comply with the final comments from supervisor. This included capture new screenshots and figures due to bad resolution or errors. In addition we wrote the abstract.
<b>20</b>	Final fixes (chapter spacing, section references, font, margins), as well as generating the final PDF through NTNU Graphics Center. Final read-through, supervisor meeting and delivery.

---

## B.2 Timesheet

## TIMESHEET

Henrik		Øyvind			Sondre			Herman			
Total		Total			Total			Total			
370		351			369			369			
Date	Task	Hours	Date	Task	Hours	Date	Task	Hours	Date	Task	Hours
<b>JANUARY</b>											
10.jan	Intro / set up environment	9	10.jan	Intro / set up environment	9	10.jan	Intro / set up environment	9	10.jan	Intro / set up environment	9
11.jan	Set up environment	1	11.jan	Set up environment	1	11.jan	Set up environment	1	11.jan	Set up environment	1
11.jan	Project plan	2	11.jan	Project plan	2	11.jan	Project plan	2	11.jan	Project plan	2
11.jan	Welcome bachelor meeting	1	11.jan	Welcome bachelor meeting	1	11.jan	Welcome bachelor meeting	1	11.jan	Welcome bachelor meeting	1
11.jan	NTNU lectures	3	11.jan	NTNU lectures	3	11.jan	NTNU lectures	3	11.jan	NTNU lectures	3
12.jan	Set up frontend project	1	12.jan	Set up frontend project	1	12.jan	Set up frontend project	1	12.jan	Set up frontend project	1
12.jan	Learning backend	3	12.jan	Learning backend	3	12.jan	Learning backend	3	12.jan	Learning backend	3
12.jan	Learning frontend	2	12.jan	Learning frontend	2	12.jan	Learning frontend	2	12.jan	Learning frontend	2
13.jan	Coding backend	1	13.jan	Coding backend	1	13.jan	Coding backend	1	13.jan	Coding backend	1
13.jan	User course w/Intility	1	13.jan	User course w/Intility	1	13.jan	User course w/Intility	1	13.jan	User course w/Intility	1
13.jan	Coding back- and frontend	4	13.jan	Coding back- and frontend	4	13.jan	Coding back- and frontend	4	13.jan	Coding back- and frontend	4
14.jan	Company meeting	2	14.jan	Company meeting	2	14.jan	Company meeting	2	14.jan	Company meeting	2
14.jan	Frontend and NSA meeting	4	14.jan	NSA and frontend	4	14.jan	Frontend and NSA meeting	4	14.jan	Frontend and NSA meeting	4
17.jan	Meeting worked project plan	2	17.jan	Meeting worked project plan	2	17.jan	Meeting worked project plan	2	17.jan	Meeting worked on project plan	2
17.jan	Backend and frontend	4	17.jan	Backend and frontend	4	17.jan	Backend and frontend	4	17.jan	Backend and frontend	4
18.jan	Project plan	5	18.jan	Project plan	5	18.jan	Project plan	5	18.jan	Project plan	5
18.jan	Frontend work	1	18.jan	Frontend work	1	18.jan	Frontend work	1	18.jan	Frontend work	1
19.jan	Meeting with supervisor	1	19.jan	Meeting with supervisor	1	19.jan	Meeting with supervisor	1	19.jan	Meeting with supervisor	1
19.jan	Worked on project plan	3	19.jan	Worked on project plan	3	19.jan	Worked on project plan	3	19.jan	Worked on project plan	3
19.jan	OpenShift intro	1	19.jan	OpenShift intro	1	19.jan	OpenShift intro	1	19.jan	OpenShift intro	1
19.jan	Worked on project plan	1	19.jan	Worked on project plan	1	19.jan	Worked on project plan	1	19.jan	Worked on project plan	1
20.jan	DNAC intro	1	20.jan	DNAC intro	1	20.jan	DNAC intro	1	20.jan	DNAC intro	1
20.jan	Bifrost intro	1	20.jan	Bifrost intro	1	20.jan	Bifrost intro	1	20.jan	Bifrost intro	1
20.jan	project plan	3	20.jan	project plan	3	20.jan	project plan	3	20.jan	project plan	3
21.jan	project plan	3	21.jan	project plan	3	21.jan	project plan	3	21.jan	project plan	3
21.jan	meetings with intility	2	21.jan	meetings with intility	2	21.jan	meetings with intility	2	21.jan	meetings with intility	2
24.jan	OPA, Kong intro Ingvald	5	24.jan	OPA, Kong intro Ingvald	5	24.jan	OPA, Kong intro Ingvald	5	24.jan	OPA, Kong intro Ingvald	5
25.jan	Meeting with supervisor	1	25.jan	Meeting with supervisor	1	25.jan	Meeting with supervisor	1	25.jan	Meeting with supervisor	1
25.jan	Kong, OPA and frontend auth	5	25.jan	Kong, OPA and frontend auth	5	25.jan	Kong, OPA and frontend auth	5	25.jan	Kong, OPA and frontend auth	5
26.jan	Kong and Frontend auth	5	26.jan	Kong and Frontend auth	5	26.jan	Kong and Frontend auth	5	26.jan	Kong and Frontend auth	5
26.jan	React App course	1	26.jan	React App course	1	26.jan	React App course	1	26.jan	React App course	1
27.jan	Project plan finalizing	4	27.jan	Project plan finalizing	4	27.jan	Project plan finalizing	4	27.jan	Project plan finalizing	4
27.jan	Frontend login	2	27.jan	Frontend login	2	27.jan	Frontend login	2	27.jan	Frontend login	2
27.jan	Template fix, Overleaf	1	27.jan	Template fix, Overleaf	1	27.jan	Template fix, Overleaf	1	27.jan	Template fix, Overleaf	1
28.jan	Show & tell, OPA, minor dev	5	28.jan	Show & tell, OPA, minor dev	5	28.jan	Show & tell, OPA, minor dev	5	28.jan	Show & tell, OPA, minor dev	5
31.jan	OPA and Azure	5	31.jan	OPA and Azure	5	31.jan	OPA and Azure	5	31.jan	OPA and Azure	5
<b>FEBRUARY</b>											
01.feb	Project plan changes	2	01.feb	Project plan changes	2	01.feb	Project plan changes	2	01.feb	Project plan changes	2
01.feb	OPA policies	2	01.feb	OPA policies	2	01.feb	OPA policies	2	01.feb	OPA policies	2
01.feb	Overleaf Setup	2	01.feb	Overleaf Setup	2	01.feb	Overleaf Setup	2	01.feb	Overleaf Setup	2
02.feb	Meeting with supervisor	1	02.feb	Meeting with supervisor	1	02.feb	Meeting with supervisor	1	02.feb	Meeting with supervisor	1
02.feb	UserID in DB	2	02.feb	UserID in DB	2	02.feb	UserID in DB	2	02.feb	UserID in DB	2
02.feb	Report structuring	3	02.feb	Report structuring	3	02.feb	Report structuring	3	02.feb	Report structuring	3
03.feb	OPA	2	03.feb	OPA	2	03.feb	OPA	2	03.feb	OPA	2
03.feb	Report drafting	5	03.feb	Report drafting	5	03.feb	Report drafting	5	03.feb	Report drafting	5
04.feb	Show & tell	1	04.feb	Show & tell	1	04.feb	Show & tell	1	04.feb	Show & tell	1
04.feb	frontend changes	3	04.feb	frontend changes	3	04.feb	frontend changes	3	04.feb	frontend changes	3
04.feb	database schema update	2	04.feb	database schema update	2	04.feb	database schema update	2	04.feb	database schema update	3
07.feb	Limit approvals based on role	2	07.feb	Limit approvals based on role	2	07.feb	Limit approvals based on role	2	07.feb	Limit approvals based on role	2
07.feb	Report drafting	3	07.feb	Report drafting	3	07.feb	Report drafting	3	07.feb	Report drafting	3
08.feb	Connect backend to DNAC	4	08.feb	Connect backend to DNAC	4	08.feb	Connect backend to DNAC	4	08.feb	Connect backend to DNAC	4
08.feb	Report structuring/writing	4	08.feb	Report structuring/writing	4	08.feb	Report structuring/writing	4	08.feb	Report structuring/writing	4
09.feb	Change wifi password feature	6	09.feb	Change wifi password feature	6	09.feb	Change wifi password feature	6	09.feb	Change wifi password feature	6
10.feb	Report writing	3	10.feb	Report writing	3	10.feb	Report writing	3	10.feb	Sick	3
10.feb	Redesign request page	3	10.feb	Redesign request page	3	10.feb	Redesign request page	3	10.feb	Sick	3
14.feb	Implement SSID changing	6	14.feb	Implement SSID changing	6	14.feb	Implement SSID changing	6	14.feb	Implement SSID changing	6
15.feb	Implement WiFi toggle	5	15.feb	Implement WiFi toggle	5	15.feb	Implement WiFi toggle	5	15.feb	Implement WiFi toggle	5
15.feb	NTNU meeting	1	15.feb	NTNU meeting	1	15.feb	NTNU meeting	1	15.feb	NTNU meeting	1
18.feb	Redesign frontend + meeting	5	18.feb	Redesign frontend + meeting	5	18.feb	Redesign frontend + meeting	5	18.feb	Redesign frontend + meeting	5
21.feb	update DB schema + frontend	5	21.feb	update DB schema + frontend	5	21.feb	update DB schema + frontend	5	21.feb	update DB schema + frontend	5
22.feb	Visiting Intility	7	22.feb	Visiting Intility	7	22.feb	Visiting Intility	7	22.feb	Visiting Intility	7
23.feb	Spec meeting MAB feature	7	23.feb	Spec meeting MAB feature	7	23.feb	Spec meeting MAB feature	7	23.feb	Spec meeting MAB feature	7
24.feb	MAC address sanitation	7	24.feb	MAC address sanitation	7	24.feb	Database MAB backend	7	24.feb	Database MAB backend	7
25.feb	Show and tell, backend	5	25.feb	Show and tell, backend	5	25.feb	Show and tell, backend	5	25.feb	Show and tell, backend	5

MARCH											
04.mar	Sick	0	04.mar	Report writing	5	04.mar	Report writing	5	04.mar	Report writing	5
07.mar	Report writing	3	07.mar	Report writing	3	07.mar	Report writing	5	07.mar	Report writing	5
08.mar	Report writing	6	08.mar	Report writing	6	08.mar	Report writing	6	08.mar	Report writing	6
09.mar	Supervisor meeting	1	09.mar	Supervisor meeting	1	09.mar	Supervisor meeting	1	09.mar	Supervisor meeting	1
09.mar	Report writing	5	09.mar	Report writing	5	09.mar	Report writing	3	09.mar	Report writing	5
10.mar	Report writing	6	10.mar	Report writing	5	10.mar	Report writing	6	10.mar	Report writing	5
11.mar	Report writing	5	11.mar	Report writing	5	11.mar	Report writing	5	11.mar	Report writing	5
13.mar	Revision/refining	3	13.mar	Revision/refining	3						
14.mar	Structure revision	6	14.mar	Structure revision	6	14.mar	Structure revision	6	14.mar	Structure revision	6
15.mar	Re-writing to be more formal	6	15.mar	Re-writing to be more formal	6	15.mar	Re-writing to be more formal	6	15.mar	Re-writing to be more formal	6
18.mar	Minor revisions and meeting	4	18.mar	Minor revisions and meeting	4	18.mar	Minor revisions and meeting	4	18.mar	Minor revisions and meeting	5
21.mar	Report: Evaluation	5	21.mar	Report: Requirements	4	21.mar	Report: Implementation	5	21.mar	Report: Evaluation	5
22.mar	Report: Database, User test	5	22.mar	Report: Database, User test	5	22.mar	Report: Database, User test	5	22.mar	Report: Database, User test	5
23.mar	User test	5	23.mar	User test	5	23.mar	User test	5	23.mar	User test	5
24.mar	Evaluation	4	24.mar	Evaluation	4	24.mar	Evaluation	4	24.mar	Evaluation	4
25.mar	Rewrite implementation	5	25.mar	Rewrite introduction	4	25.mar	Write Frontend	5	25.mar	Rewrite implementation	5
28.mar	Rewrite implementation	5	28.mar	Rewrite introduction	4	28.mar	Rewrite tool-survey	5	28.mar	Rewrite process	5
29.mar	User test - Technicians	5	29.mar	User test - Technicians	5	29.mar	User test - Technicians	5	29.mar	User test - Technicians	5
APRIL											
22.apr	User test preparation	6	22.apr	User test preparation	6	22.apr	User test preparation	6	22.apr	User test preparation	6
25.apr	User test - Customers	6	25.apr	grammar review	6	25.apr	User test - Customers	6	25.apr	User test - Customers	6
26.apr	Implement changes	6	26.apr	User test / half day	3	26.apr	User test / half day	3	26.apr	User test review	6
27.apr	Implement changes	6	27.apr	Sick		27.apr	Database changes, coding	6	27.apr	User test review	6
28.apr	User test review	6	28.apr	Sick		28.apr	Tool survey (database, docker)	6	28.apr	User test review	6
29.apr	Thesis review	6	29.apr	Thesis review	6	29.apr	Thesis review	6	29.apr	Thesis review	7
MAY											
02.mai	Grammar and revision	7	01.mai	Grammar and revision	7	01.mai	Grammar and revision	7	02.mai	Grammar and revision	7
03.mai	Implemented changes	7	02.mai	Future work	7	02.mai	Implemented changes	7	03.mai	Future work	7
04.mai	Revision/refining	5	04.mai	Revision/refining	5	04.mai	Revision/refining	5	04.mai	Revision / Refining	5
05.mai	Revision/refining	5	05.mai	Revision/refining	5	05.mai	Revision/refining	5	05.mai	Revision / Refining	5
06.mai	Revision/refining	5	06.mai	Revision/refining	5	06.mai	Revision/refining	5	06.mai	Revision / Refining	5
09.mai	Revision/refining	4	09.mai	Revision/refining	4	09.mai	Revision/refining	4	09.mai	Revision / Refining	4
10.mai	Revision/refining	5	10.mai	Revision/refining	5	10.mai	Revision/refining	5	10.mai	Revision / Refining	5
11.mai	Revision/refining	5	11.mai	Revision/refining	5	11.mai	Revision/refining	5	11.mai	Revision / Refining	5
12.mai	Revision/refining	6	12.mai	Revision/refining	6	12.mai	Revision/refining	6	12.mai	Revision / Refining	5
13.mai	Abstract	5	13.mai	Abstract	5	13.mai	Abstract	5	13.mai	Abstract	5
16.mai	Create PDF / Styling	4	16.mai	Create PDF / Styling	4	16.mai	Create PDF / Styling	4	16.mai	Create PDF / Styling	4
18.mai	Final read / deliver	3	18.mai	Final read / deliver	3	18.mai	Final read / deliver	3	18.mai	Final read / deliver	3



---

### **B.3 Minutes of meetings**

## MINUTES OF MEETING

Date	Attendees	Conclusions
17.01.2022	All	We are ahead of schedule, and will use the surplus time to create the deliverable for 31 January
19.01.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- 31.march first draft of thesis</li> <li>- Upload documents on teams with supervisor</li> <li>- Final deadline 12pm 20.may</li> <li>- Email weekly status report at least one day before supervisor meeting</li> <li>(what have been done, any problems, want to discuss, what we will do the next week, are we on track)</li> <li>- Literature survey, tool survey, software/system design/implementation/evaluation, thesis writing, project presentation</li> <li>- Thesis is the most important, start early</li> <li>- Use the supervisor to get feedback along the way</li> <li>- Get details on how the zones are secured (how they work)</li> <li>- What are the requirements (functional: what the client expects. non-functional: how fast is it, security)</li> <li>- User testing of interface</li> </ul>
20.01.2022	Group, Ali Arfan, Tobias	<ul style="list-style-type: none"> <li>- DNAC (digital network...) cisco DNA center</li> <li>- ISE.lab.int.intility.no (port 9060/ers/sdk)</li> <li>- Use DNAC when changing password on guest network</li> <li>- WLC wireless lan controller, controls all access points, can be used for configuration, pushes the configuration to devices <a href="https://dnac.static.aa.cust.xcv.net/dna/platform/app/consumer-portal/developer-toolkit/apis">https://dnac.static.aa.cust.xcv.net/dna/platform/app/consumer-portal/developer-toolkit/apis</a></li> <li>- Cisco ISE Identity service engine</li> <li>- ISE controls identity to all devices</li> <li>- MAC address spoofing</li> </ul>
25.01.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- How the approval process works for requests</li> <li>- What does it mean that a zone is secure</li> <li>- Use bullet points for requirements</li> <li>- Policies should be functional</li> <li>- Non functional security high</li> <li>- More non functional</li> <li>- Add one more zone to topology</li> <li>- How is the frontend accessed? Do you need VPN or can they access it from anywhere.</li> <li>- Describe briefly how the topology works</li> <li>- RFC as title of topology is confusing</li> <li>- Everything added to the report needs a reasoning for being there</li> <li>- All the terminology needs to be cited</li> <li>- Use a page to describe all the terminology</li> <li>- Explain how to read risk matrix</li> <li>- Simplify Gantt, weekly basis instead of daily</li> </ul>
02.01.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- Project plan</li> <li>- Implemented Kong, OPA and authentication</li> <li>- Minor frontend and backend changes</li> <li>- Created Overleaf template for the final report</li> <li>- Feedback projectplan                             <ul style="list-style-type: none"> <li>- Nice!</li> </ul> </li> <li>- Plan to make several policies in OPA</li> </ul>
09.02.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- Dynamic/static</li> <li>- UIX/UCD view</li> <li>- Personal privacy                             <ul style="list-style-type: none"> <li>- Whats necessary for the user to see</li> </ul> </li> <li>- Discuss what services Intlity actually wants</li> </ul>
16.02.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- DEMO RFC</li> <li>- Live feedback</li> <li>- further improvements</li> <li>- presented the backend code and database</li> </ul>
23.02.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- SSID and Passwrord change in GUI</li> <li>- Frontend redesigning</li> <li>- Starting up with MAB next week</li> <li>- On track with Gantt-chart</li> </ul>
02.03.2022	Group, Supervisor NTNU	No meeting due to seminar in IØ2000

09.03.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- Our previously drafted structure fits quite well according to our supervisors tips.</li> <li>- need to add/change a few chapter titles, as well as changing reference style.</li> <li>- Finished core functionality of RFC</li> <li>- Major features implemented</li> <li>- Only minor revisions remain.</li> <li>- Started on report</li> <li>- structure and layout</li> <li>- on track with Gantt chart, and moving to report phase.</li> </ul>
16.03.2022	Group, Supervisor NTNU	No meeting due to seminar in IØ2000 / Supervisor attends Department meeting
23.03.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- Demo RFC</li> <li>- frontend feedback</li> <li>- discuss potential features</li> <li>- questions regarding the user test interview</li> </ul>
30.03.2022	Group, Supervisor NTNU	Supervisor sickness, cancelled
06.04.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- User testing</li> <li>- feedback on usertests</li> <li>- Questions regarding report</li> </ul>
20.04.2022	Group, Supervisor NTNU	No meeting due to seminar in IØ2000
04.05.2022	Group, Supervisor NTNU	Meeting cancelled due to report revisioning.
11.05.2022	Group, Supervisor NTNU	<ul style="list-style-type: none"> <li>- Revised report based on feedback from supervisor</li> <li>- Created new and updated figures to the report</li> <li>- Low quality screenshots replaced with new screenshots in better resolution</li> <li>- Some questions regarding the feedback: <ul style="list-style-type: none"> <li>- "a customer" - is it their/his/her? a customer is a company</li> <li>- "The three most requested WLAN changes" - Why underlined?</li> </ul> </li> </ul>
18.05.2022	Group, Supervisor NTNU	

---

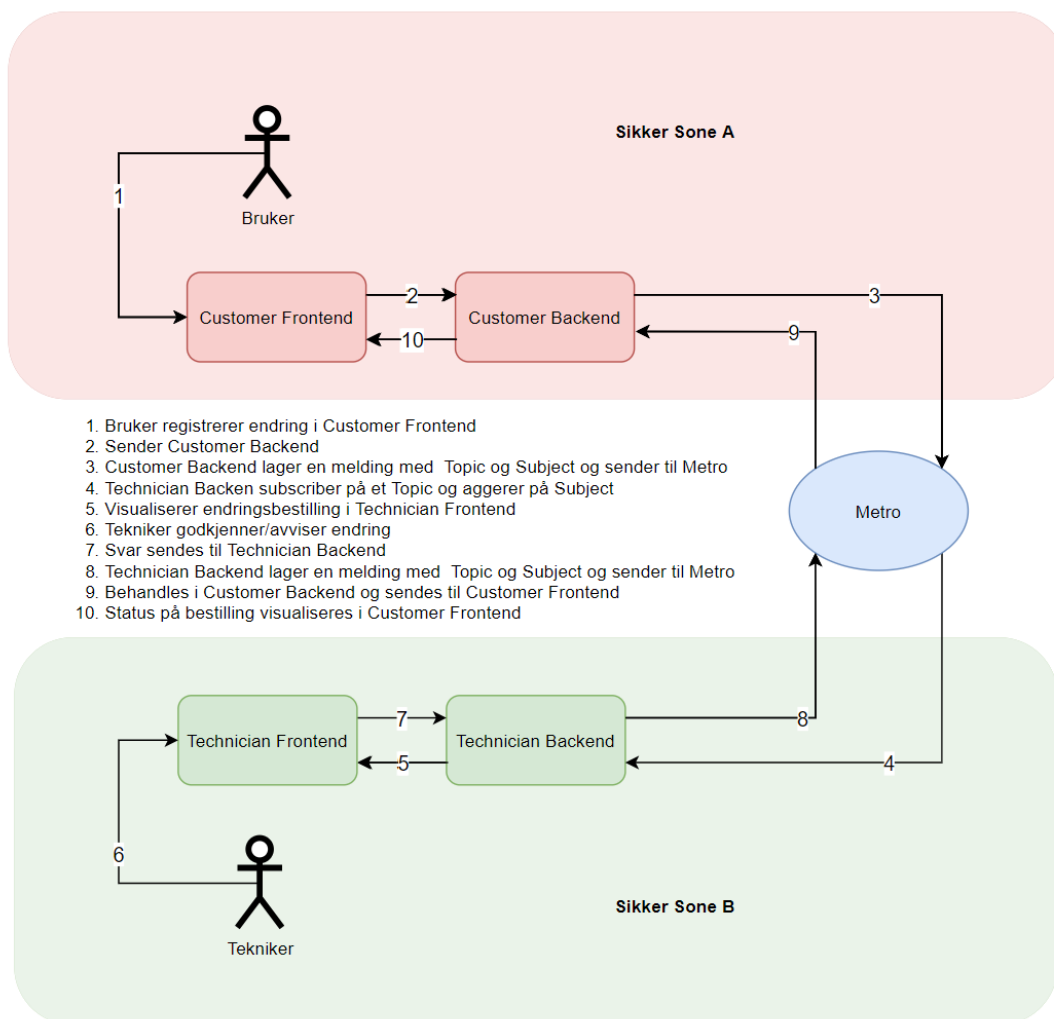
**C Assignment description from Intility**

## BACHELORPROSJEKT INTILITY NETWORK SERVICES

Hensikten med oppgaven er å bygge grunnmuren til et system for endringsbestillinger og tjenestebestillinger som skal gå til Intility. Dette skal realiseres ved å bygge en kundeportal og en teknikerportal, begge front-end og back-end som benytter seg av et Intility sitt eget meldingssystem (Metro) bygd på Azure Service Bus og Event Grid. Metro muliggjøre sikker utveksling av informasjon med hjelp av meldinger på tvers av sikre soner / domener på Intility plattformen.

Hovedrammen av oppgaven handler om å bygge gjøre systemet kapabelt til å ta å sende og abonnere på meldinger gjennom Metro. Som et proof of concept skal systemet sende melding fra kundeside til teknikerside, tekniker godkjenner og kundeside mottar melding og visualiserer dette i portalen. På toppen av fundamentet skal gruppen bygge en eller flere features for bestilling i kundeportal. Her står gruppen mer fritt til å komme med innspill.

Gruppen skal fortrinnsvis programmere i python på Backend, hvor det prefereres at alt skrives async i moderne rammeverk som FastAPI. Det er et krav at frontend programmeres i React og bruk av Intility sitt desinsystem (bifrost.intility.no).



---

## D Project plan



Norwegian University of  
Science and Technology

DCSG2900: Bachelor Thesis Bachelor of Science in  
Digital Infrastructure and Cyber Security

## Project plan

Spring 2022

Herman Torland

Øyvind Wold

Sondre Gunneng

Henrik Granlund

Supervisor: Jia-Chun Lin

Department of Information Security and Communication Technology  
Faculty of Information Technology and Electrical Engineering

Gjøvik, NTNU

## Vocabulary

- API – Application programming interface
- CORS – Cross origin requests
- FastAPI – A framework written in the python language to facilitate the creation of APIs
- OpenAPI – An API standard
- Swagger – A tool for providing automated documentation of API endpoints.
- PEP-8 – Style guide for Python
- IDE – Integrated development environment
- JavaScript – Programming language
- TypeScript – Built on top of JavaScript to add features such as type checking
- Bifrost – Design system built by Intility
- Kong – API gateway. Used to forward requests
- OPA – Policy-based control for cloud environment
- OpenShift – Tool for container administration
- Git – System for version control and DevOps tool
- GitLab – platform for good version control and collaboration
- Overleaf – Online LaTeX editor
- Cisco DNA Center – Management platform for Cisco networking equipment

## 1. Goals

### 1.1 Background

[Intility](#) is a Norwegian company that offers an end-to-end IT platform for their customers. This involves taking full responsibility for their customers' IT stack, including everything from the network infrastructure to client devices and internal IT support. To allow for this model to be organized and secure, Intility has chosen to create a “secure zone” for each customer. These zones are isolated networks that contain all the customers' equipment and devices at their locations, as well as virtual networks to run their services in datacenters. For Intility to make changes to services on behalf of their customers they need to use a virtual machine running on a secured workstation. For a workstation to be considered secure it needs to be configured after Intility's internal guidelines.

Intility has requested us to make a proof-of-concept web app that offers their customers an easy and intuitive way of requesting simple changes to their networks. Such requests can for example include network SSID and password changes. Each request must be approved by a technician at Intility. To enable this service to work between the customers zone and Intility's zone, we have decided to implement the Kong API Gateway. This can act as a central hub that forwards different API requests to different services. Furthermore, we will use Kong with plugins to implement authentication, policies, and CORS, which are all security measures. Further requirements and guidelines from Intility are outlined in Section 1.3.



## 1.2 Project goal

Effect goals:

- Learn to collaborate on a bigger project with more complex requirements than earlier assignments at NTNU.
- Get a deeper understanding of how an IT business works from the inside, including using process frameworks like scrum in practice
- Learn to use modern technologies such as React and Python, that are popular in the IT industry and will be beneficial to know in the future.

Outcome goals:

- Develop a system that fulfills Intility's functional requirements and might potentially be released as a feature to their customers.
- Write and deliver a thorough report that outlines our process from start to finish, as well as describing how our solution will solve a current problem.
  - The report should cover how we have worked together (process framework, minutes of meetings, challenges etc.)
  - The report should cover how the final product is built and what technologies it consists of.

## 1.3 Requirements/constraints

Functional requirements

- Intility has requested that the system should be a web application with a frontend developed in React, as well as recommending the use of Python for the backend.
- The frontend should talk to a REST-API backend using fetch requests. We are recommended to make this API using the FastAPI framework.
- To allow for communication between the frontend and backend, we are advised to use an API-gateway called Kong. This should be done to solve the “secure-zone” problem mentioned in Section 1.1.
- Intility also has a requirement that we follow their general developers' guidelines regarding UI design and code-structuring.
- The system should use Open Policy Agent (OPA) for enhancing security. This will be responsible for checking that only authorized technicians can approve a request.
- The system should be able to trigger several different APIs once a request is approved, to begin with we will implement a feature for changing network SSID and password using Cisco DNA Center, which provides an API for such use cases.

Non-functional requirements

- The system should have good security.
- The interface should be intuitive and natural to use.
- The interface should be adaptive to different screen sizes
- The interface should be responsive and fast.
- The code should be easily reusable for future development.

## 1.4 Planned Architecture

Considering the requirements mentioned in the previous section, we have initially decided on the following architecture. Each customer will be able to access an interface from their own secure zone and use it to request changes to their own already existing networks. These requests will be sent to the Kong API gateway, which will check against policies in OPA. An example of a policy check could be verifying that a customer has the correct rights to request a change. If OPA evaluates the request to be valid, Kong will forward it to the backend and store it in a database. A technician at Intility will be able to see all requests through their own interface where they may approve or decline them. This will again lead to an API call to Kong and a policy check to OPA, before the request is forwarded to the backend. At this point, the system should be able to trigger an API call to several different services, in our case exemplified by, but not limited to Cisco DNA Center (DNAC). DNAC provides an API that allows for the requested changes to be performed on the customers network by utilizing an SSH tunnel. Below is a visual representation of our planned topology.

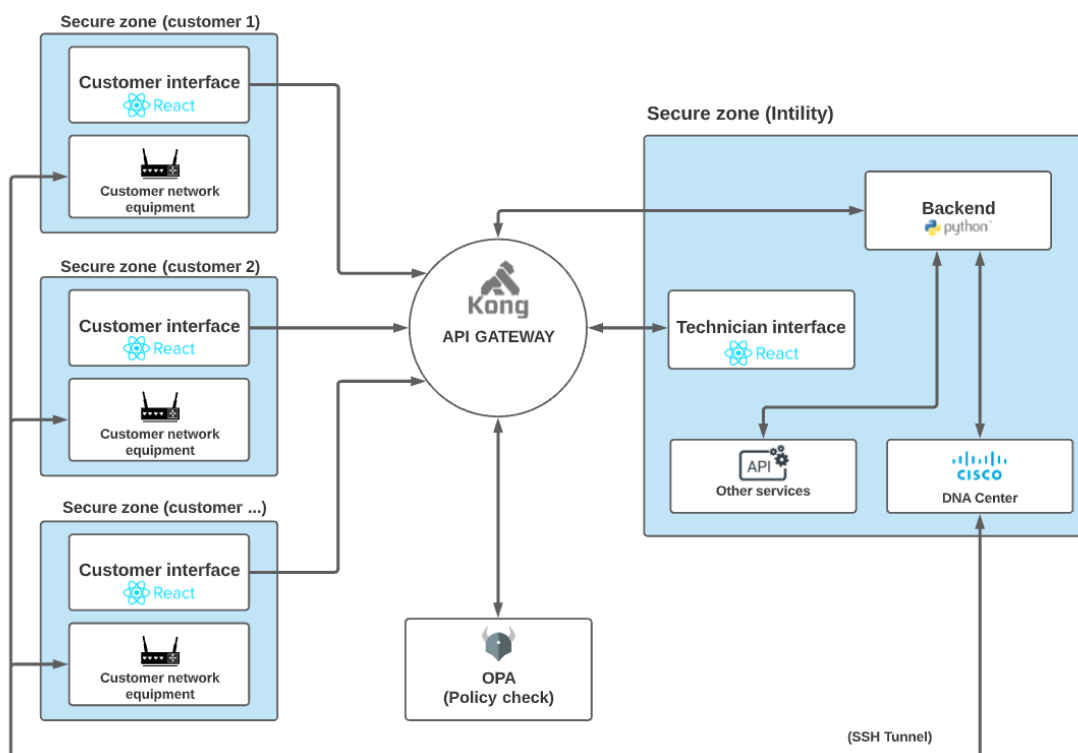


Figure 1. Planned architecture

## 2. Project organization

### 2.1 Roles

- Project lead: Herman Torland – herman.torland@intility.no
- Project manager: Øyvind Wold – oyvind.magnus.wold@intility.no
- Backend lead: Sondre Gunneng – sondre.gunneng@intility.no
- Frontend lead: Henrik Granlund – henrik.granlund@intility.no

### Project lead

Makes sure all the deadlines are met and conflicts are solved. Will also be the main communicator with the project stakeholders.

### Project manager

Will take the lead role in the absence of the project leader and make sure the deadlines are met. Responsible for facilitating scrum.

### Backend lead

Quality check of the backend, such as code quality and that it meets standards.

### Frontend lead

Quality check of the frontend, make sure the UI/UX is of a good standard.

### Secretary

Writes minutes of meetings. The role will be rotated between all group members on a weekly basis.

Week nr.	Responsible
2	Herman Torland
3	Øyvind Wold
4	Sondre Gunneng
5	Henrik Granlund
6	Herman Torland
...	...

*Table 1.0, Secretary role rotation*

## 2.2 Group rules and routines

### 2.2.1 - Team meetings

We will conduct team meetings on a regular basis. All members are required to arrive on time and meet prepared. Every team member should speak during the meetings, and at a minimum give a general status report of how far they have gotten since last time. Each meeting should be documented with minutes by the secretary.

### 2.2.2 - Absence

Although members should always strive to attend scheduled meetings, the group should have an understanding that different issues may arise that can prevent a member from joining. If unable to attend a meeting, the member should inform the rest of the group as soon as possible. Attendance will be reported in the meeting minutes.

### 2.2.3 - Work effort

We have great ambitions for the project, and thus a high work effort is required by all team members. Everyone should try their best to solve their assigned tasks, and the threshold for giving up should be high. However, due to a lot of new concepts and varying degrees of difficulty in tasks, members should not be afraid of asking for help if they are struggling with completing a task, as this is a team effort.

#### 2.2.4 - Disagreements

The group should expect that certain disagreements will arise during the project period. Members are expected to be patient and understanding when someone suggests an unfamiliar perspective or solution to a problem. If any serious disagreements arise, we will try to solve them internally first before potentially contacting our supervisor for an external perspective and guidance.

#### 2.2.5 - Daily work routine

The group has decided to work on the project every day of the week during Intility's core working hours (09:00-15:00). Each day should start with a daily standup where today's tasks will be discussed. Due to Covid-19 restrictions, all work will be conducted digitally to begin with. We will work physically at Intility's headquarters if the possibility arises at a later stage.

#### 2.2.6 - Weekly status report

We have agreed with our supervisor on meetings every Wednesday at 09:00. At least one day in advance a status report should be sent to our supervisor containing the following bullet points:

- What we have done in the previous week.
- Any potential problems and how we solved them.
- What to discuss during the meeting.
- What to do this upcoming week.
- How our progress is going relative to the Gantt chart.

#### 2.2.7 - Backup of documents

Backup of all work should be done regularly by the project manager. These backups should be stored in a shared OneDrive folder that all group members have access to.

#### 2.2.8 - Logging of hours

Every group member is responsible for logging their own working hours into our shared spreadsheet. Each member should strive to log around 30 hours of work per week, including meetings and sessions.

#### 2.2.9 - Sanctions for violations

If a team member does not follow the rules outlined above, an extraordinary meeting will be held where the team member is responsible for explaining themselves. This could result in a warning. If a group member receives two warnings, we will have a meeting with our supervisor to find a solution. If we do not agree or find a solution to the issue, it could lead to eviction of this group member.

### **3. Planning, follow-up, and reporting**

#### **3.1 Main division of the project**

In the beginning of the project, we will base our work around the waterfall method, but we want to move closer to a scrum-based process framework. The main reason for this is that we want to gather as much information as possible in the early phase, ensuring each member achieves a good understanding of all the different elements of the project. With that in mind, we can move on to a more complex and detailed scrum method before the end phase.

### 3.2 Timeline with process framework choice

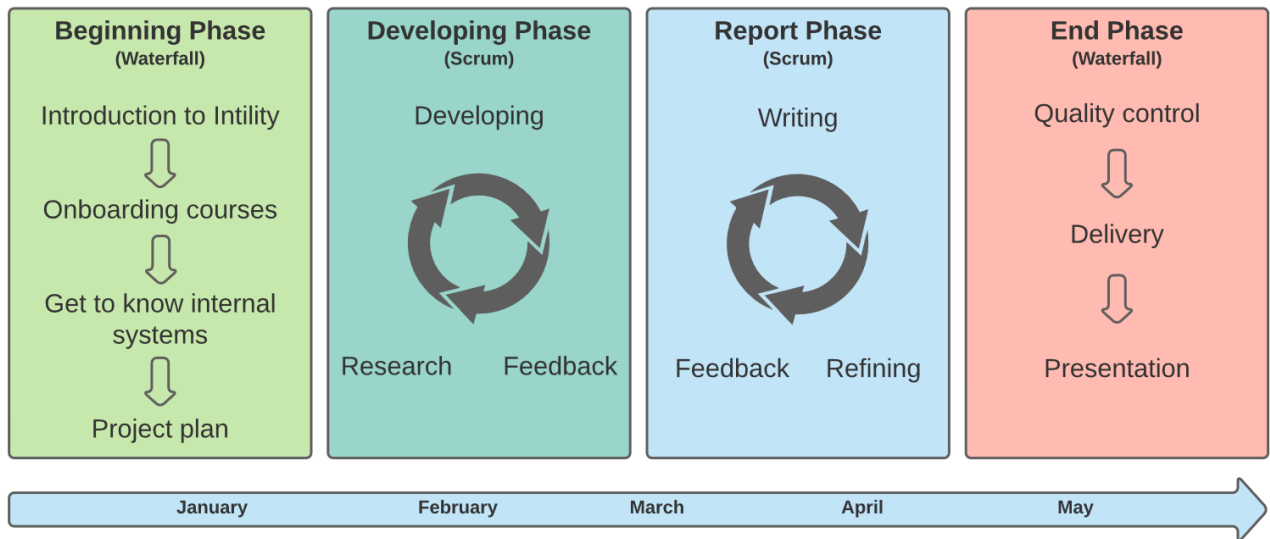


Figure 2. Work process during project

For this project we have considered using two development methods: Scrum and waterfall.

Scrum is a process framework where a team collaborates and break down tasks into several small projects that must be completed within a given period also called “sprints.” Each sprint has its own timeframe.

We have chosen to use Scrum as it is the most appropriate development method for our project. By dividing the timeframe into several sprints, we have a better overview of how much time we have on each task. As we are most familiar with this kind of framework, this has also been the reason that we decided to use scrum.

The waterfall method is a sequential method where the project flows through all the phases of a project as a waterfall. The main aspect of the waterfall method is that the project cannot continue before the first phase is completely done, as each phase depends on the previous phase. In our case this could be especially useful at the start of the project due to introductions to several systems that it is necessary for us to learn. We also see that there is a need for the use of waterfall in the end phase, as we must go through several steps before we can deliver the main project.

In addition to the four boxes in the figure above, we have added a timeline at the bottom of the figure that represents the estimated time we will spend on the project. This is just a rough estimate, as a more detailed timeline is described in the attached Gantt-chart.

### 3.3 Plan for status meetings and points of decision during the period

- Have a meeting with our supervisor (Jia-Chun Lin) once every Wednesday from 09:00-09:30.
- Monday meetings at 09:00 to agree on what needs to be done the following week.
- Attend “show and tell” with Network Services department at 09:00 am Fridays.

## 4. Organization of quality control

### 4.1 Tools, standards, documentation

#### 4.1.1 Backend

The backend is going to be an API the frontend will communicate with and will be developed using the FastAPI framework, which is written in python. One of the benefits of using FastAPI is that it follows the OpenAPI standards which enables auto generated documentation for the endpoints using Swagger. Since we will be using python, we have chosen to stick to its official coding standard PEP-8. To make sure that our code follows this standard, we will make use of a code formatter called Black, which runs before every commit to git using pre-commit hooks.

#### 4.1.2 Frontend

The frontend is going to be the graphical user interface that the customers of Intility and their technicians will use to interact with the backend. We are going to develop it using React (client-side rendering) library. The languages we could choose from were JavaScript or Typescript. We have selected JavaScript because nobody in our group has any experience with Typescript and since the application is small, we do not see any benefit in the extra features Typescript provides. For the user interface Intility have asked us to use their internal design system called Bifrost, which provides components such as tables, notifications, modals, and color schemes.

#### 4.1.3 Development environment and shared tools between the applications

The IDE we use is called PyCharm and is designed for python and has many useful features for autocompletion, navigating the source code and starting the local environment. It also has the benefits of working well with JavaScript which is the language we use in our frontend. Since Intility also uses this IDE internally, it will be easier to get help if we have any issues. The frontend and backend we are working on will be on separate networks and needs a way to communicate, for this we will be using the API gateway part of Kong to forward requests between the secure zones. We will also be using OPA for managing policies which will be our access control, for example to separate customer view from the technician view in the frontend.

#### 4.1.4 Deployment, backup, and test environment

We will be using git for version control in conjunction with GitLab for full integration with project planning features like issues and milestones to help keep things organized and backed up. GitLab will also help with automatic deployments, also called CD (continuous deployments) when we merge branches for issues into the main branch a deployment pipeline will run tests and check the code before deploying to the live environment. To get our application hosted we will be using OpenShift which will pull our built images from GitLab and create pods to deploy in Kubernetes.

#### 4.1.5 Documents and communication

To store and collaborate on documents such as project plans, reports, minutes of meetings, etc. we will be using Teams and Office 365. When it comes to drafting the thesis, we will use Overleaf to live collaborate on Latex documents. Both Office 365 and Overleaf will give us the benefit of automatic backups. Office 365 makes it easy for us to have everything accessible in one place, such as email, shared calendars, and easy meeting invites. In Teams we will have a team for the group which also

the supervisors provided by Intility have access to. A separate team will be used to share documents with NTNU supervisor, this is because we cannot give, due to restricted access from externals in Intility's internal teams/systems. For daily voice communication within the group, we will be using Discord where our supervisor from Intility also has access to the Discord server. The last communication tool is Slack, the reason we have this is to get notifications from GitLab, for example when a pipeline fails or someone merges.

## 4.2 Plan for inspection and testing

To inspect and test our solution, we will work closely with Intility as they have dedicated different people to guide us with user testing, code review, and project processes.

## 4.3 Risk analysis at a project level

- Value:** Project  
**Description:** Data loss  
**Consequence:** High  
**Probability:** Low  
**Action:** We will be using online collaborative tools such as GitLab, Office 365 and Overleaf which all provide automatic online backup.
- Value:** Project  
**Description:** One or more group members become sick for a longer period and cannot do their assigned tasks  
**Consequence:** Medium  
**Probability:** Low  
**Action:** The most important thing is for whoever gets sick to let the group know as soon as possible and then the rest of the group will have to divide workload among themselves.
- Value:** Project  
**Description:** Missing the deadline of 20<sup>th</sup> of May  
**Consequence:** High  
**Probability:** Low  
**Action:** Revise the plan each week to see if we are following the schedule and make any adjustments if needed. The project lead has the ultimate responsibility of ensuring the project is delivered.
- Value:** Project  
**Description:** Any of the supervisors gets ill, either from NTNU or Intility  
**Consequence:** Low  
**Probability:** Low  
**Action:** Even though we might not get the scheduled weekly meeting we can still contact our supervisors through email.

5. **Value:** Project  
**Description:** If a group member must leave the project because of some unforeseen incident.  
**Consequence:** High  
**Probability:** Low  
**Action:** Depending on how far along the project we have gotten, the workload could be increased for all the other group members. And we would have to talk to our supervisor on how to solve it, scale down some parts of the project. Every part of the project should be documented, to simplify the process of transferring a task.
  
6. **Value:** Project  
**Description:** Loss of motivation during the project  
**Consequence:** Medium  
**Probability:** Medium  
**Action:** As time goes by and different problems arise, one or more of the group members may experience a loss of motivation. This might lead to varying quality of the work conducted. To prevent this, we will make sure to rotate what types of tasks each member does. We will also strive to take the weekend off to get a break each week. As motivation, we will strive to take a spring break in April, if we are on track.
  
7. **Value:** One of the programs stops working/ is not supported.  
**Description:** One of the programs that the project depends on does not work or support on our respective systems.  
**Consequence:** High  
**Probability:** Low  
**Action:** If one of the programs we depend on is no longer supported or stops working, we must take a decision together with the supervisor and Intility to find an alternative solution. This can be time-consuming as there are several factors that a new program must go through, to be approved by Intility.

**Table 2. Risk matrix**

Consequence/Probability	Low	Medium	High
Low	1,4,5	2	
Medium		6	
High	3,7		

*The numbers represent the risks, and they are placed according to consequence vertically and probability horizontally*



## 5. Plan of implementation

1	<b>Project setup and intro</b>		10.01.2022	28.01.2022	18	
2	Personal environment setup	All	10.01.2022	12.01.2022	3	
3	Project setup	All	12.01.2022	14.01.2022	3	
4	Project plan	All	17.01.2022	28.01.2022	12	
5	MVP features and UI decisions	All	24.01.2022	27.01.2022	4	
6	Database schema design	All	27.01.2022	28.01.2022	2	
7	<b>Frontend</b>		31.01.2022	01.03.2022	30	
8	Bifrost and React learning	All	31.01.2022	01.02.2022	2	
9	List requests	Henrik	02.02.2022	02.02.2022	1	
10	Create request	Henrik	02.02.2022	02.02.2022	1	
11	Approve and decline requests	Herman	03.02.2022	03.02.2022	1	
12	Communicate with API through Kong	Herman & Henrik	04.02.2022	14.02.2022	10	
13	Create separate customer view and technician view	All	14.02.2022	23.02.2022	10	
14	Supplementary features/visuals	All	24.02.2022	01.03.2022	7	
15	<b>Backend</b>		31.01.2022	01.03.2022	30	
16	API: Create new request	Sondre	02.02.2022	02.02.2022	1	
17	API: Get all requests	Sondre	02.02.2022	02.02.2022	1	
18	API: Decline/Approve request	Øyvind	03.02.2022	04.02.2022	2	
19	Move API to Kong gateway	Sondre & Øyvind	07.02.2022	11.02.2022	5	
20	Authentication of users	Øyvind	14.02.2022	18.02.2022	5	
21	Perform approved changes on network	All	21.02.2022	01.03.2022	8	
22	<b>Project report</b>		02.03.2022	06.05.2022	66	
23	Introduction and problem statement	All	02.03.2022	09.03.2022	7	
24	Literature survey	Øyvind	09.03.2022	16.03.2022	12	
25	First draft delivery	All	02.03.2022	31.03.2022	29	
26	Process	Herman	07.03.2022	18.03.2022	12	
27	Requirement evaluation	Øyvind	21.03.2022	01.04.2022	24	
28	Technicals	Sondre	14.03.2022	25.03.2022	12	
29	Reflection and conclusion	Henrik	28.03.2022	08.04.2022	19	
30	Readover and adjustments	All	02.03.2022	06.05.2022	40	
31	<b>Deliveries and finalization</b>		06.05.2022	20.05.2022	15	
32	Revision and spell-checking	All	13.05.2022	19.05.2022	7	
33	Overleaf formatting/visuals	All	16.05.2022	19.05.2022	4	
34	Gather appendixes and sources	All	06.05.2022	12.05.2022	7	
35	Delivery	All	19.05.2022	20.05.2022	2	
36	Presentation	All	TBA		-	

Figure 3. Gantt Chart

## 6. Confirmation

I hereby confirm that I have read and agreed on the content of this Project plan.

Gjøvik, 27.01.2022

Sondre Gunneng



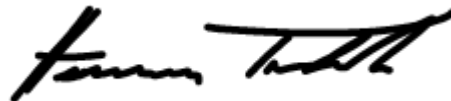
Henrik Granlund



Øyvind Wold



Herman Torland



---

## **E Project agreement**

*Fastsatt av prorektor for utdanning 10.12.2020*

## **STANDARDAVTALE**

### **om utføring av studentoppgave i samarbeid med ekstern virksomhet**

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

#### **Forklaring av begrep**

##### **Opphavsrett**

Er den rett som den som skaper et åndsverk har til å fremstille eksemplarer av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

##### **Eiendomsrett til resultater**

Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

##### **Bruksrett til resultater**

Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

##### **Prosjektbakgrunn**

Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

##### **Utsatt offentliggjøring**

Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

## 1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for informasjonssikkerhet og kommunikasjonsteknologi
Veileder ved NTNU: Jia-Chun Lin e-post og tlf. <a href="mailto:jia-chun.lin@ntnu.no">jia-chun.lin@ntnu.no</a> , +47 458 49 287
Ekstern virksomhet: Intility AS, Network Services Ekstern virksomhet sin kontaktperson, e-post og tlf.: Marius Engan, <a href="mailto:marius.engan@intility.no">marius.engan@intility.no</a> , +4798409649
Student: Herman Torland Fødselsdato: 12.08.1996
Student: Henrik Granlund Fødselsdato: 29.07.1998
Student: Sondre Gunneng Fødselsdato: 30.04.1993
Student: Øyvind Wold Fødselsdato: 06.11.1995

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

## 2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 10.01.2022
Sluttdato: 20.05.2022

Oppgavens arbeidstittel er:  Request for change
---

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

### 3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:
--

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

### 4. Studentens rettigheter

Studenten har opphavsrett til oppgaven<sup>1</sup>. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

### 5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

#### Alternativ a) (sett kryss) Hovedregel

---

<sup>1</sup> Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

<input type="checkbox"/>	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven
--------------------------	--

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

#### **Alternativ b) (sett kryss) Unntak**

X	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt
---	---

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene: Prosjektet er en del av større pågående prosjekt for utvikling av eksterne grensesnitt ut mot sluttbrukere av plattformen.

#### **6. Godtgjøring ved patenterbare oppfinnelser**

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

#### **7. NTNU sine rettigheter**

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

#### **8. Utsatt offentliggjøring**

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

<input type="checkbox"/>	Oppgaven skal være offentlig
--------------------------	------------------------------

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i

denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
<b>X</b>	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

## 9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.






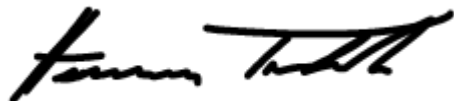
Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.



Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

### Signaturer:

Instituttleder:
Dato:
Veileder ved NTNU: Jia-Chun Lin 
Dato: 31. 01. 2022
Ekstern virksomhet:

Dato: 27.01.2022
Student: Sondre Gunneng 
Dato: 27.01.2022
Student: Henrik Granlund 
Dato: 27.01.2022
Student: Øyvind Wold 
Dato: 27.01.2022
Student: Herman Torland 
Dato: 27.01.2022

