Thesis

NHL Stenden University of Applied Sciences

In association with:

Quality ICT B.V.

Written by:

Christopher Sulistiyo (4850025)

Submission:

1-July-2024

Work placement lecturer(s):
Frans Huijgens
Arvid Fens

Company Supervisor(s):
Manuel Weidijk
Mark Kolk

Version Control

Version	Activities	Date
Initial version	Draft version	28 May 2024
Second Version	Implemented feedback from Frans regarding Research Design, Introduction, Research, and overall structure of the document	6 June 2024
Final draft	Implemented feedback from both Niels and Frans	1 July 2024

Remarks

Any changes and new developments that have a significant impact on the project proceedings will be noted here.

Company Supervisor Approval

This graduation work placement research paper report made by the author, Christopher Sulistiyo, an international student of NHL Stenden University of Applied Science as part of his graduation requirement for the ICT & CT Information Technology Bachelor program in Emmen branch, is hereby approved by:

Company Supervisor

Mark Kolk

General Manager / Advisor

Quality ICT B.V.

Summary

Q-ICT is a small cybersecurity consultant firm based in Emmen, Drenthe, the Netherlands. They act as a trusted IT provider that will handle all clients' IT infrastructure. The services they offered are network, storage and server management, reselling Microsoft 365 subscriptions as an official partner of Microsoft, data backup and recovery, financial data analytics to help clients gain valuable insights into their business, cloud migration and services, technical remote and on-site support by their helpdesk services, customized software development and system integration, cybersecurity consulting and strategies, and more. They have over 500 clients, all spread throughout the Netherlands, and their main target audience is small to medium-sized businesses, ranging from the employee size of 1-100, that do not have an IT department, therefore lacking the technical knowledge. Q-ICT is therefore given the responsibility to manage and securing clients IT infrastructure, and clients will ask for their supports in case of technicality issues arise.

Q-ICT have their own internal system made by themselves, called the QaaS App, which is a web application integrated with various cybersecurity, management and financial tools that Q-ICT uses to manage their business processes regarding their clients, subscriptions, products, and services. It has 5 APIs connected to it. Together, these APIs provide Q-ICT with the tools to be used by both Q-ICT employees and the clients. A client of Q-ICT can access the QaaS app and see the status of their network, security of their endpoints, their financial status, their subscriptions within Microsoft 365, etc. The QaaS App therefore acts as a customer insight management tool for the clients of Q-ICT. For the employees of Q-ICT it is a centralized operation platform designed to streamline and manage all of Q-ICT's day-to-day business activities. Last year, Q-ICT has purchased the rights to a SentinelOne subscription with the help of one of their partner companies, Pax8. Since then, they have been utilizing this product to provide better endpoint security to their clients, migrating from Bitdefender, a traditional antivirus solution, into more modern solutions such as EDR system provided by SentinelOne.

SentinelOne is an American cybersecurity company that has a product of the same name. It is an Endpoint Detection and Response platform, which purpose is to detect and respond to threats of an endpoint in real-time, replacing the traditional antivirus. Its Agents, which serves as a next generation antivirus, utilize AI and ML to detect and respond to threats, and it is deployed on the endpoint itself. Traditional antivirus uses methods that often struggles against new, unknown (zero-day) threats and malware that changes frequently. SentinelOne, on the other hand, and other similar next-generation antivirus solutions use behavioural analysis to identify malware based on its actions rather than its code. This allows them to detect and respond to new, unknown threats that traditional AV solutions would miss. The purpose of an EDR platform is to have a centralized location to monitor and respond to threats on all endpoints in a network, in a company. With this new cybersecurity tools in their system, Q-ICT hope to gives one of the best world-class cybersecurity solutions from Gartner magic quadrant to small and medium business enterprises in the Netherlands, therefore contributing to the improvement of cybersecurity standards in the Netherlands.

Q-ICT then also would like to take one step further and provide transparency to their client to maintain the trust in their relationship with their clients and reinforcing their confidence in Q-ICT's services by displaying client's SentinelOne data exclusively to them in the QaaS App. This integration of SentinelOne to the QaaS App will prevent Q-ICT from creating accounts to each user of their 500 client companies, and increasing the sell value of the QaaS App. Q-ICT also would like to have more control over what data to display to the clients, as almost all of them are non-technical users, therefore providing them with a dashboard that is more user-friendly with visualization on the data to simplify them to the common users. Q-ICT also would like to have more control on adding additional capabilities on the QaaS App based on what SentinelOne system can do, such as adding files into blocklist or whitelist, remediate and rollback an endpoint operating system, network isolation, and more.

The integration of the SentinelOne API into the QaaS app is possible through SentinelOne REST API, with the use of QaaS app existing framework and services, Flutter and Firebase. Once a SentinelOne Agent is properly installed on an endpoint, a SentinelOne token from the parent environment with all the read permission needs to be created and stored in Google Secret Manager, where it is there in the cloud, secured and encrypted. An API rate limiting

needs to be created to not flood the API server with requests each time a user refreshes the page. After that, client's SentinelOne data needs to be connected with their already existing information in the QaaS environment, proper limitations need to be set in the user's permission to prevent them from accessing data that do not belong to them, especially when doing filtering or searching. The new module needs to be deployed in the live environment and tested with client's real data and elevated privileges in the SentinelOne environment.

This project would be the first iteration to the integration of SentinelOne module to the QaaS App. Further user pilot testing by the real clients and additional requirements from the inside market are recommended to further improve the module and add additional features.

Contents

Glossary **10**

Acronyms **12**

1 Introduction **15**
1.1 Project Background . . . 15
1.2 Background Introduction . . . 16
1.2.1 The Company . . . 16
1.3 Problem Statement . . . 20
1.4 Project Objectives . . . 21
1.5 Reading Guide . . . 21

2 Research Design **22**
2.1 Research Topic . . . 22
How can Quality ICT B.V. effectively integrate and leverage SentinelOne EDR platform for continuous cybersecurity monitoring? . . . 22
Sub-question #1: What are the primary front and back-end technologies and components utilized in the QaaS web application, and how do they function together to as a system? . . . 22
Sub-question #2: What is SentinelOne EDR Platform? . . . 22
Sub-question #3: How can SentinelOne be integrated within the QaaS App? . . . 22
Sub-question #4: How can SentinelOne data best be displayed to non-technical users? . . . 22
2.2 Research Methodology . . . 22
2.2.1 Method of Data Collection . . . 22
2.2.2 Selected Measuring Instruments . . . 23
2.2.3 Method of Data Analysis . . . 23
2.2.4 Reliability, Validity, and General Applicability . . . 23

3 Research Results **25**
3.1 Sub-Question #1: . . . 25
3.1.1 The QaaS App Infrastructure . . . 25
3.1.2 Conclusion . . . 31
3.2 Sub-Question #2: . . . 31
3.2.1 Conclusion . . . 34
3.3 Sub-Question #3: . . . 35
3.3.1 SentinelOne API Token . . . 35
3.3.2 SentinelOne Data Types . . . 35
3.3.3 Conclusion . . . 37
3.4 Sub-Question #4: . . . 37
3.4.1 Comparison to other EDR solutions . . . 38
3.5 Conclusion . . . 44

4 Realization **45**
4.1 Methodology and Approach . . . 45
4.2 Product . . . 45
4.3 Development . . . 45
4.3.1 The Back-end . . . 46
4.3.2 The Front-end . . . 47

4.4 Deploying everything to the Live environment . . . 48

5 Conclusion and Recommendation **49**
5.1 Conclusion . . . 49
5.2 Recommendation . . . 49
5.2.1 Integration with N-Central data . . . 49
5.2.2 Redis middleware NoSQL database caching . . . 50
5.2.3 Additional requirements gathering . . . 51
5.2.4 User pilot testing . . . 51
5.2.5 Training and User Manual . . . 51
5.2.6 SentinelOne Aggressive Mode . . . 51

Bibliography **53**

A FD (Functional Design) **55**

B TD (Technical Design) **103**

C Deployment Manual **148**

D Project Plan **158**

E Research Proposal **194**

F Initial Description of Graduation Project **209**

G Interview SentinelOne Transcript **215**

H Interview QaaS App Transcript **220**

List of Figures

1.1 Flat (hierarchical) structure of Q-ICT, along with its departments and their managers. 17
1.2 The infrastructure of the QaaS app 18

3.1 All the products offered by Firebase (*Kubernetes, 2018*) 27
3.2 The *Scheduled Function* in the QaaS app that is scheduled to run every 7 days to keep the API data up-to-date (*Cronitor, n.d.*) 29
3.3 All the extensions that are used in the QaaS app, mostly about Algolia 29
3.4 SentinelOne in the Gartner magic quadrant that proves that it is one of the leading solutions in cybersecurity 32
3.5 Visualization widgets in CrowdStrike 38
3.6 Adding a widget in CrowdStrike 39
3.7 Create scheduled report in CrowdStrike 39
3.8 Dashboard in Trend Micro 40
3.9 A Tab in Trend Micro can contain up to 10 widgets 40
3.10 Change the layout of the dashboard in Trend Micro by dragging and dropping the widgets 41
3.11 Filter the data by date and time range in Trend Micro 41
3.12 Add a tag to the data in Trend Micro 42
3.13 The dashboard of Heimdal® 42
3.14 The dashboard of Huntress 43
3.15 Seeing incidents in Huntress 43

5.1 How the infrastructure would look if Redis is implemented 50
5.2 Redis providers now cannot distribute Redis any more without a direct partnership with Redis, therefore Redis itself controls the pricing. 51

Listings

3.1 Example of a typical onRequest function 27
3.2 Example of onCall function with authentication check 28

List of Tables

3.1 Comparison between the 1st and 2nd Generation of Cloud Functions in Firebase 26

Glossary

API is a software intermediary that allows two applications to talk to each other. In other words, an API is the messenger that delivers request to the provider that was requested from and then delivers the response back (*IBM, n.d.-a*) . 13

CAPTCHA is a type of challenge-response test used in computing to determine whether or not the user is human (*Google, n.d.-a*) . 9, 25

CRM is a process or strategy that companies use to manage and analyze customer interactions and data throughout the customer ' lifecycle. CRM technology is a software system that helps organizations easily track all communications and nurture relationships with their leads and clients *Salesforce, n.d.* . 16

JSON is a lightweight data-interchange format that is easy for humans to read and write while being also easy for machines to parse and generate too. It is based on a subset of the JavaScript programming language and is commonly used for representing structured data. JSON is often used for transmitting data between a server and a web application, as well as for storing configuration data (*JSON, n.d.*) . 9

NoSQL is a category of DB that provides a mechanism for storage and retrieval of data that is modelled in ways other than the tabular relations used in RDMS (*MongoDB, n.d.*). NoSQL DBs are typically designed to handle large volumes of unstructured or semi-structured data, such as JSON, XML, or binary objects, and they offer a flexible data model that can evolve over time. Both databases in Firebase, Firestore, and Real-time Database are NoSQL databases. . 24

Pentest A.K.A. ethical hacking, is a simulated cyberattack where professional ethical hackers break into corporate networks to find weaknesses (*Cloudfare, n.d.*) . 14

REST is a software architectural style for providing standards between computer systems on the web, making it easier to communicate with each other. It is often characterized by how they are stateless and separate the concerns of client and server (*IBM, n.d.-b*) . 9, 16

RESTful API is an API that follow REST principles (Fielding and Taylor, 2000). It considered the gold standard for scalability and are highly compatible with microservice architecture. It is the often used protocol in the context of building APIs for web-based applications . 16

RMM is a software platform that helps MSPs remotely monitor and manage their clients' endpoints, networks, and computers (*TechTarget, n.d.*). 16

SOAP API is an API strictly based on XML for the message structure and HTTP for the protocols (*Indeed, 2023*) . 16

XML is a markup language that defines a set of rules for encoding documents in a format that is both human and machine-readable. It provides a way to structure data in a hierarchical format using tags to define elements and attributes within those elements (*AWS, n.d.*) . 16

Cron Job is a job scheduler responsible on scheduling tasks on repeated schedule on a server (*Kubernetes, 2024*) . 28

reCAPTCHA is a CAPTCHA system owned by Google to enable web hosts to distinguish between human and automated access to websites (*Google, n.d.-b*) . 25

Threat Hunting A.K.A. cyberthreat hunting, is a proactive approach to identifying previously unknown, or ongoing non-remediated threats, and searching for signs of potential cyber threats within an organization's network or system (*SentinelOne, n.d.-a*) . 31, 34, 40

Acronyms

2FA Two-Factor Authentication. 24, 25, 47

A.K.A. Also Known As. 9, 31

AI Artificial Intelligence. 3, 13, 19, 28, 30–32, 35, 39, 51, 52

APA American Psychological Association. 20

API Application Programming Interface. 3, 9, 13, 15–17, 19, 22, 24–30, 35, 38, 39, 49–53

ARP Address Resolution Protocol. 33, 34

AV Anti-Virus. 3, 16–18, 31, 33, 39

AWS Amazon Web Services. 40

B.V. Besloten Vennootschap. 0

BaaS Back-end as a Service. 24

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart. 9, 25

CDR Content Disarm and Reconstruction. 16

CEH Certified Ethical Hacker. 14

CISO Chief Information Security Officer. 14

CMEK Customer-Managed Encryption Keys. 28

CRM Customer Relationship Management. 9, 16

CVE Common Vulnerabilities and Exposures. 35, 53

DB Database. 9, 18, 26, 28, 31, 52, 53

DHCP Dynamic Host Configuration Protocol. 34

e.g., exempli gratia. 34

EC-Council International Council of E-Commerce Consultants. 14

EDR Endpoint Detection and Response. 3, 13, 17, 19, 30, 31, 33, 35, 39, 44

ERP Enterprise Resource Planning. 16, 30

etc., et cetera. 16, 24, 32

ETDR Endpoint Threat Detection and Response. 31

GB Gigabyte. 25

GCP Google Cloud Platform. 24, 28

GUI Graphical User Interface. 32

HTTP Hypertext Transfer Protocol. 9, 24–27, 30, 39, 49

HTTPS Hypertext Transfer Protocol Secure. 24

i.e., id est. 18

IAM Identity and Access Management. 28

ICT Information and Communication Technology. 15, 21

ID Identifier. 33, 50, 51, 53

IDS Intrusion Detection System. 29, 31

IoT Internet of Things. 31, 33

IP Internet Protocol. 34

IPS Intrusion Prevention System. 31

IT Information Technology. 13, 14, 16, 19, 29, 30, 32, 35, 40, 52

ITIL Information Technology Infrastructure Library. 14

JS JavaScript. 25

JSON JavaScript Object Notation. 9, 16, 22, 28, 50, 51

MAC Media Access Control. 34

MCDA Multi-Criteria Decision Analysis. 22

MDR Managed Detection and Response. 31, 34

MFA Multi-Factor Authentication. 25

MGMT Management. 32, 33, 35

MIB Management Information Base. 34

MKB Midden- en Kleinbedrijf. 15, 18

ML Machine Learning. 3, 13, 19, 30–32

MS Microsoft. 14–16, 18

MSP Managed Service Provider. 9, 16

NDR Network Detection and Response. 31

NGAV Next-Generation Antivirus. 31

NHL Noordelijke Hogeschool Leeuwarden. 0

NIC Network Interface Controller. 34

NIST National Institute of Standards and Technology. 35

NoSQL Not Only SQL. 9, 24, 28, 53

OS Operating System. 16, 44

OTP One Time Password. 24

Pentest Penetration Testing. 9, 14

Pub/Sub Publisher/Subscriber. 27

Q-ICT Quality ICT. 0, 3, 13–19, 22, 24, 28, 30, 33–35, 39, 50, 52, 55

QaaS Quality as a Service. 3, 13, 15–19, 21, 22, 24–30, 35, 39, 50, 52–54

RAM Random Access Memory. 25

RDMS Relational Database Management System. 9

REST Representational State Transfer. 3, 9, 16, 26, 35, 39

RMM Remote Monitoring and Management. 9, 16

SaaS Software as a Service. 15

SDK Software Development Kit. 27

SEM Security Event Management. 31

SIEM Security Information and Event Management. 31, 32

SIM Security Information Management. 31

SME Small and Medium-sized Enterprises. 14, 16

SOAP Simple Object Access Protocol. 9, 16

SOAR Security Orchestration, Automation, and Response. 32

SOC Security Operations Center. 13, 19, 32, 34, 40, 44

SWOT Strengths, Weaknesses, Opportunities, and Threats. 22

TS TypeScript. 25

UI User Interface. 14

UX User Experience. 16, 50, 53

vCPU Virtual Central Processing Unit. 25

VLAN Virtual Local Area Network. 33

XML Extensible Markup Language. 9, 16, 22

Chapter 1

Introduction

1.1 Project Background

In today's rapidly evolving digital landscape, cybersecurity remains a paramount concern for organizations across all industries. With the proliferation of sophisticated cyber threats and the increasing complexity of IT infrastructures, business are constantly seeking new and innovative ways to protect their digital assets and fortify their defences and safeguard sensitive data. In this pursuit, cybersecurity consultant firms have emerged as a critical ally for organizations, providing expert guidance and support in the development and implementation of robust cybersecurity strategies and playing a pivotal role in offering expertise and guidance to help organizations navigate the intricate realm of cybersecurity.

One of the key strategies employed by cybersecurity consultants is the integration of third-party security APIs into their arsenal of tools and technologies. These APIs provide invaluable functionalities, ranging from vulnerability assessment and security scans, subscriptions monitoring, financial ticketing system to device health monitoring and threat intelligence analysis by AI. By leveraging these APIs, cybersecurity consultants can enhance their capabilities and provide a more comprehensive and effective security solution to their clients, streamline their operations, provide clients with robust, proactive security measures, and improve their overall service delivery.

When talking about cybersecurity solutions, SentinelOne emerges as one of the leading organizations in the field (*Gartner, n.d.*), offering cutting-edge endpoint protection and threat intelligence solutions. It leverages advance AI and ML algorithms, providing comprehensive protection against malware, ransomware, and other cyberthreats.

SentinelOne provides comprehensive data for cybersecurity analysts and SOC teams by delivering comprehensive insights into malware analysis, threat hunting, and threat detection. Its advanced EDR capabilities enable real-time monitoring and automated threat response, allowing cybersecurity experts to quickly identify and neutralize potential threats. The platform's detailed telemetry data aids uncovering the behaviour and origin of malicious software. For threat hunting, SentinelOne's robust search capabilities and contextual data provide the tools needed to proactively identify and mitigate security risks.

Additionally, its automated detection and response mechanism enhances protection and prevention efforts, ensuring that security teams are prepared to act swiftly and effectively in the event of a real-time cybersecurity incident. This kind of information and functionality empowers the blue team to develop informed strategies and execute precise, timely interventions to safeguard organizational assets. It is one of the must-have protection tools for business organizations looking to bolster their cybersecurity posture and safeguard their digital assets.

As the main topic of this graduation work placement project, the author has been tasked to integrate SentinelOne endpoint protection capabilities into the client company internal application, the QaaS app. The main objective of this project is to promote transparency more into Q-ICT clients by providing them with real-time data and visibility into their organizations, enabling them to make informed decisions and take proactive measures to protect their own digital assets and mitigate security risks.

1.2 Background Introduction

1.2.1 The Company

Q-ICT, is a small cybersecurity consultancy based in Emmen, Drenthe, the Netherlands. The company follows a flat organizational structure, which means that there are few or no levels of middle management between the staff and every communication directly goes with the director. Their customers are SME companies with employees ranging from 1 to 100. As their cybersecurity consultant, Q-ICT is responsible for monitoring their clients' devices and ensuring the overall security of their systems, IT infrastructure, and digital assets. As of now, Q-ICT has over 500 active customers. Some notable clients are: Kigpolis Fietsverzekeringen, CLS Europe, and Heli Holland.

Departments

The company consists of multiple departments in its behalf, each with their own functions and responsibilities. Those departments are the following:

1. *Service Help Desk Department*: it serves as a frontline support function responsible for addressing client inquiries, troubleshooting technical issues, and providing guidance and support to clients in resolving their technical challenges. It contains 2 sub-departments, the first level help desk and the second level help desk. The First Level Help Desk is the first point of contact for clients seeking technical assistance and support, and it is responsible for managing and resolving client issues in a timely and efficient manner. The Second Level help desk is responsible for providing more advanced technical support and troubleshooting for complex technical issues and consists of Senior System Engineers. It has 2 services, indoor and outdoor customer services. With indoor customer services, the company provides remote support to clients, while with the outdoor customer services, the company provides on-site support to clients.
2. *Cybersecurity Department*: it's responsible for implementing procedures that will be used throughout the company's system, especially in Help Desk Department, to ensure that the company's information technology infrastructure is secure. It also develops methodologies and best practices related to cybersecurity, as well as integrating ITIL and NIS2 principles, CompTIA certification ISO 27001 standard and product management practices into the company's operations. Conducting regular security scans for clients' networks, systems, and applications to identify vulnerabilities and security risks and performing Pentest that involves simulating cyberattacks to identify weaknesses in the client's defences and assess their ability to withstand and respond to real-world cyber threats. This department mainly consist of IT managers, Pentesters, CISOs, and cybersecurity specialists equipped with CEH certification from EC-Council.
3. *Software Development Department*: it addresses Q-ICT clients' need by creating custom software solutions tailored to their specific requirements. It consists of software developers that work closely with clients to understand their unique cybersecurity challenges and design solutions that effectively address those concerns, while utilizing their expertise in programming languages, software frameworks, and cybersecurity principle to develop secure and reliable applications. **This is the department where the author is currently working on his graduation work-placement project**. Mainly, this department uses Dart with Flutter as the main front-end for building the UI, and Firebase as the main back-end cloud solution. Furthermore, it utilizes Google Firebase as the main cloud solution for the applications it develops as it works together with Flutter, but it expresses its desires to expand more into MS Azure in the future.
4. *Financial Department*: it is responsible for managing the financial aspects of the company to ensure its financial health and stability. It includes Accountant and Financial Advisor, and they are responsible for analyzing financial data, identifying trends, and making strategic financial decisions to support the company's growth and objectives.

Figure 1.1: Flat (hierarchical) structure of Q-ICT, along with its departments and their managers.

Recently, as since of last year, Q-ICT has obtained the rights to the SentinelOne environment and official permission to sell its subscription to Q-ICT customers under their own name. Q-ICT will then act as the management of the client's domain in SentinelOne that subscribes to them through Q-ICT. Because SentinelOne is relatively new to the Q-ICT, it has not fully integrated with the internal system yet. This will be discussed further in the Problem Statement section.

Internal System and Tools

The QaaS App

The QaaS app is web application that is used by Q-ICT and its clients. For Q-ICT's clients, it is a SaaS that is used to see what kind of information are available regarding their ICT infrastructure, based on the subscriptions that they have with Q-ICT. For example, a customer that subscribes on N-Central and SentinelOne will be able to see the status of their devices from N-Central API, and the health of their devices from SentinelOne API. For Q-ICT employees themselves, it serves as an internal system that is used to manage the clients and their ICT infrastructure.

APIs and Tools

The QaaS app also utilizes several APIs and technologies to help with its operations. It needs to manage and make connection different sort of APIs to help with the operations as an ERP application. Those APIs are the following:

- Resello: is used for Q-ICT's MS subscriptions owned by Pax8 (*LinkedIn, n.d.*). It is a cloud marketplace that simplifies the way SME companies buy, sell, and manage cloud solutions through automation. It provides a single platform to manage the entire cloud customer lifecycle, from quote to cash to support, thus simplifying the process of buying, selling and managing cloud solutions. Furthermore, it normally uses SOAP API for its communication.
- SnelStart: is used for Q-ICT automation of financial and accounting system software, such as managing invoices, etc., for SMEs. It offers a range of products and services to help businesses manage their finances, including accounting software, invoicing software, and financial management tools.
- Bodyguard.io: is a CDR tool used for security tab. It is a product from a Dutch company that filters and scrutinizes downloads from web browsers to detect and prevent malicious files with real-time download scanning capabilities. It normally uses RESTful API for its communication.
- N-Central: is a product from N-Able and is used for monitoring clients' devices and ensuring the overall security of their systems, IT infrastructure, and digital assets. It is a RMM platform designed to help MSP and IT professionals

to remotely monitor and manage their clients' devices and networks. It provides a comprehensive set of tools and features for monitoring, managing, and securing clients' devices and networks, including patch management, AV, backup and disaster reovery, virus scanner, network topology mapping, disk encryption, and Windows update that supports showing OS, disk, and status update. The return response from this API is in XML and JSON format, making it both a REST and SOAP API.

- PerfectView: is used for CRM software (*PerfectView, n.d.*). It is designed to improve business relationships with customers, assist in customer retention, and drive sales growth. In the QaaS app, it is used to manage the relationships and interactions with the app's users, which could include tracking user interactions, managing customer support requests and analyzing user data to improve the app's functionality and UX.

Below is how each entity connected to each other to form the QaaS app:

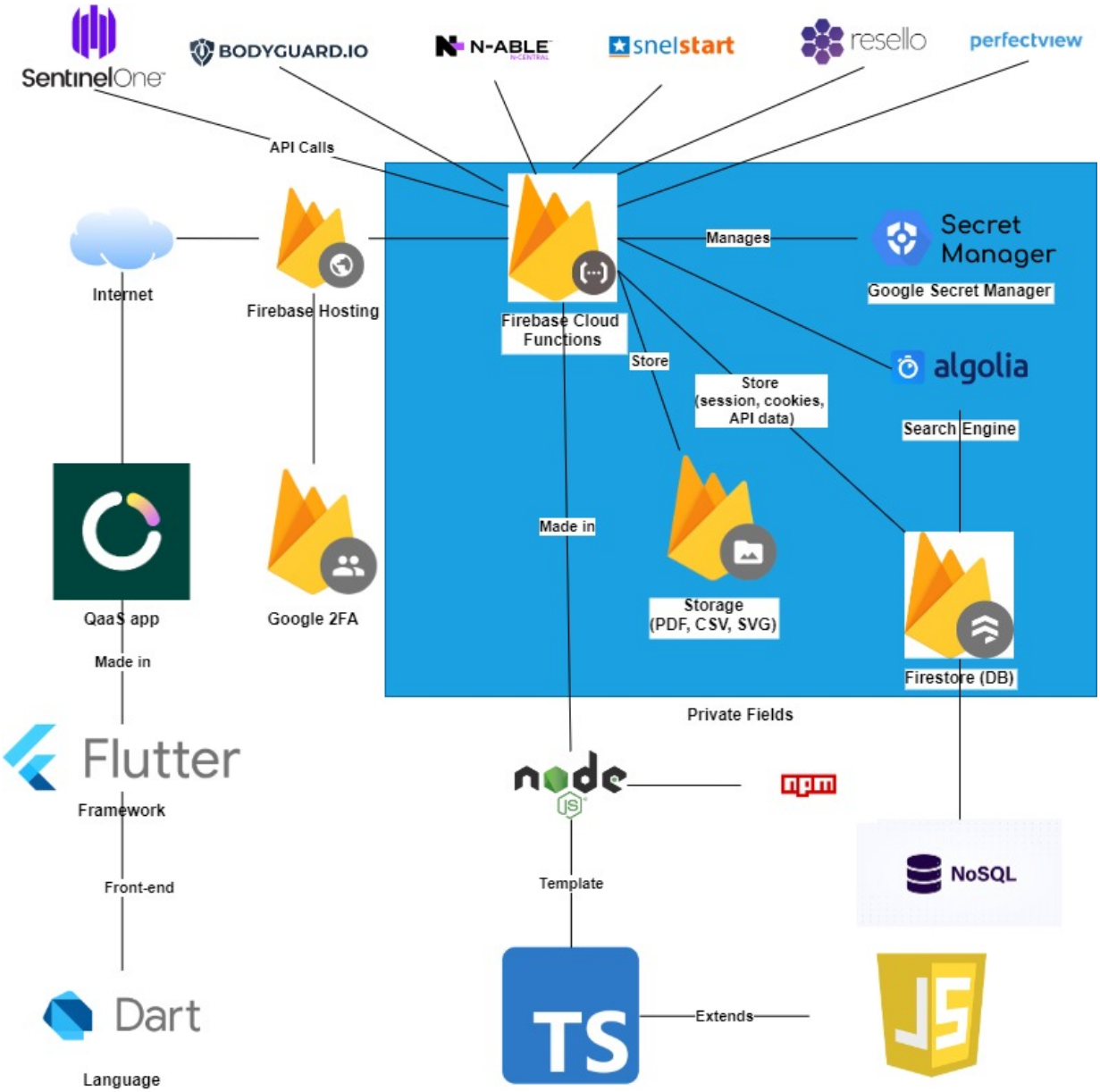


Figure 1.2: The infrastructure of the QaaS app

Besides all the 5 internal APIs that Q-ICT uses, the company also uses several technologies to help them with their operations. Those tools are:

- Computicate (now newly named Acronis): is used as their ticketing system, providing ticketing solution services

to customers for a wide range of events and activities (*Acronis, n.d.*).

- TOMTelecom: is used for their company's phone system. It is responsible for structured process of call routing on incoming calls from customers to the appropriate department or individuals, ensuring effective communication and issue resolution (*TOMTelecom, n.d.*).

Subscriptions

Q-ICT has monthly subscription that it offers to its clients. Subscription within the Q-ICT is divided into smaller proportions which are the following:

1. QaaS Safe: includes security package, patch management, monitoring for workstations or laptop, and traditional AV like BitDefender. This traditional AV product will be removed soon as SentinelOne EDR platform itself is properly integrated to the QaaS environment since it is already an upgrade from traditional AV. This subscription is required for every user in a company to be the official client of Q-ICT. SentinelOne payment subscription will also be included in this after the integration.
2. QaaS Safe+: for advance security protection. Bodyguard.io is included. This subscription is still proof of concept within the stakeholders.
3. QaaS Help: is used for a free call service to the helpdesk between working hours (Monday to Friday, 08:30 - 17:00). This subscription will utilize an external tracking system to check whether a user has this subscription to ensure proper billing. Then a financial and ticketing system like SnelStart and Computicket is used for billing users who call the Helpdesk who do not have this subscription.
4. QaaS Back-up: for back-ups from the data of the customer from MS 365 Cloud provider. It is for restoring if data is lost. The data will be stored off-site in OneDrive as the cloud personal storage since Q-ICT is a Microsoft partner. This subscription has also a collaboration with N-Central for its back-up and disaster recovery solutions.

Products and Services

Some products of Q-ICT are already discussed briefly in the subscription section. But here is the full review:

- Customized software development: this field is one of the main responsibilities of Q-ICT itself, it provides customers with tailor-made software solutions that are designed to meet their specific needs and requirements. The company works closely with clients to understand their unique challenges and develop software applications that effectively address those concerns.
- Online-backup: the company offers a service where the clients can store copies of their files, documents, and data on remote servers via the internet. These remote servers are hosted in secure data centres. Furthermore, MKBiT also provides DB migration in case any event of disasters occurred.
- Active monitoring: the company will offer its customers the chance to oversee, track, and manage their computer systems, networks, or applications in real-time. It serves various purposes, including ensuring system stability, optimizing performance, enhancing security, and providing valuable insights into the usage patterns and behaviours of users or devices. In the future, it wants to bring more automation to its system, providing customers with automatic messages when a disk space becomes full, or when there is an error message in clients' computers.
- Anti-spam software: a specific type of software application that is designed by MKBiT to detect, prevent, and block unsolicited and unwanted e-mail messages, commonly known as spam, from reaching clients' e-mail inboxes. They are typically sent in bulk to many recipients without their consent, often containing advertisements, phishing attempts, malware, or other malicious content. It has features such as whitelist, blacklist, and filter e-mails per e-mail account or e-mail address.
- Microsoft 365 products: the company is also providing access to Microsoft's suite of cloud-based productivity tools and services, including installation services, to its clients. These include Microsoft Word, Excel, PowerPoint, Outlook, and more, which are all accessible online through a web browser or can be installed on local devices such as computers and smartphones.
- AV software: it resells and gives clients advice and comparisons on good antivirus software from manufacturers like McAfee and Bitdefender. This software is designed to detect, prevent, and remove malware from computer systems.

- Cloud services: the company also provides clients with a tailor-made cloud solution based on their specific wishes and problems, whether it is using a public, private, or hybrid cloud. It delivers a range of computing resources, applications, and services over the internet through cloud computing technologies. These services enable the clients to access and use computing resources without the need to own or maintain physical hardware and software infrastructure, enabling their employees to access their files anytime, anywhere, without others having to access them.

In addition, for the software development, Q-ICT also uses other MS technologies such as Power Platform (i.e., Power Automate, Power BI, Power Apps, Power Virtual Agents, and Power Pages), Dynamics 365, and Azure to help with their software development, therefore the clients who are asking for the product to be developed with the help of those MS technologies will also have those subscriptions of the products, therefore paying extra charge.

Security Compliance

Q-ICT is committed to delivering exceptional IT services and cybersecurity solutions to its clients. To ensure client's system meet the highest security standards and comply with regulatory requirements, Q-ICT is actively pursuing several key cybersecurity certifications and compliance framework. By having these certifications, the client can have a proof that they can give to the government in case of inspection that they comply with the latest cybersecurity standards through Q-ICT. The certifications are:

- **NIS2**: is an EU directive aimed at enhancing cybersecurity across member states by ensuring a high common level of security for network and information systems (*Overheid, n.d.*).
- **ISO 27001**: is an international standard for quality assurance of information systems. It specifies requirements for establishing, implementing, maintaining, and continually improving an information security management system (ISMS).
- **CompTIA Security+**: is a globally recognized certification that validates the baseline skills necessary to perform core security functions, concepts, and practices
- **CEH (Certified Ethical Hacker) Training by EC Council**: is a certification that validates an individual's understanding of the tools and techniques used by malicious hackers to compromise network security. It is designed to help cybersecurity professionals think like a hacker and identify vulnerabilities in network systems.

1.3 Problem Statement

The company currently manages numerous third-party APIs mentioned above for various purposes. Currently, the company has already offered SentinelOne endpoint protection subscription to its clients. However, the clients cannot access to SentinelOne dashboard without Q-ICT making the necessary accounts for every single individual first. Making accounts for the clients can be a time-consuming process, seeing that the clients reach up to 500 active companies, therefore an integration of SentinelOne to Q-ICT's own internal system is more advantageous to the company by allowing clients to see their SentinelOne data related to their system and endpoint health directly through Q-ICT's platform. Moreover, it is not beneficial from a business perspective to direct customers to other external security solutions, as most of the data in SentinelOne is in English and Q-ICT want to be the primary resource for cybersecurity assistance for its clients around the Netherlands.

Furthermore, the SentinelOne dashboard is not user-friendly to the clients. It is made to directly assist cybersecurity specialist like threat analysts working in SOC department or IT administrators. Therefore, Q-ICT also wishes that this integration to be designed with the usability to the audience that have limited or no knowledge at all about cybersecurity. Thus, the integration of SentinelOne to the QaaS App simply adds a layer of functionality to the clients, enabling them to see and easily understand the data. In essence, SentinelOne will complement the QaaS app by providing enhancement to the app so that it can show, visualize and display the intricate data related to endpoint security such as real-time insights, comprehensive audit trails, threat intelligence, endpoint security logs, configuration details, user activity logs, telemetry data, alerts, remediation actions, reports and analytics, and more. This will then add more value to the QaaS App as an advance threat detection platform providing user-friendly security dashboard that offers detailed insights into device health and status. This project will not however, make any changes related to how the internal system architecture of SentinelOne, including the structure of its Agents and the algorithms driven by AI and ML.

By this integration, the QaaS App will also complement SentinelOne by promoting it to the broader audience, especially here in the Netherlands. By embedding SentinelOne's advanced security features into the QaaS App, Q-ICT

showcases the robustness and reliability of SentinelOne's solutions to its extensive client base. This integration acts as a powerful endorsement, demonstrating SentinelOne's effectiveness in real-world scenarios and highlighting its value in providing comprehensive cybersecurity protection.

Thus, the company has tasked the author with the development of the SentinelOne security threat platform integration for continuous cybersecurity monitoring within the QaaS app as the main topic of his graduation work placement project.

1.4 Project Objectives

In the end of this project which consist of 90-99 working days, the following objectives should be achieved:

1. Effectively integrates and leverages the SentinelOne EDR platform for continuous cybersecurity monitoring within the QaaS app by showing the data from SentinelOne in the QaaS app.
2. The QaaS app should have a way to display the data retrieved from the SentinelOne API in a user-friendly manner in order for the client users helpdesk support, financial department, cybersecurity department, software development department, and other employees within Q-ICT departments to see the data easier.

1.5 Reading Guide

This report is structured as follows:

- **Summary**: provides a brief and concise overview of the entire report, including the research questions, key findings, and conclusion. Its purpose is to provide readers with a quick and comprehensive understanding of the report.
- **Introduction**: provides an overview of the project background, the research topic of the company, and the project objectives. It introduces the context of the research and outlines the structure of the report.
- **Research Design**: outlines the methodologies employed during the research process. It describes the research approach, data collection methods, selected measuring instruments, data analysis techniques used to address the research questions, reliability, validity, and general applicability.
- **Research Results**: presents the research results, including the research methodology, the findings, and the analysis of the research questions. Firstly, it describes the methodologies employed during the research, and then it provides a detailed account of the research process and the outcomes of the research.
- **Realization**: provides a detailed description of the software end-product developed during this work placement project. It outlines insights into design, development, and implementation phases. It also highlights key features, functionalities, and technology specifications used in the project.
- **Conclusions and Recommendations**: the Conclusion summarizes the key findings and results achieved during the research and realization phases. The Recommendation section outlines the proposed next steps and future research areas to further enhance the project and address any outstanding issues. It discusses potential areas for further exploration or refinement.
- **References**: lists all the sources cited in the report following the appropriate to APA 7th edition citation style.
- **Appendices**: includes any additional supplementary information, data, or materials that are relevant to the report but not included in the main body of the report.

Chapter 2

Research Design

2.1 Research Topic

In research, it is paramount to have the formulation of a clear research topic, research main question, and research sub-questions. The main question serves as the focal point around which the research revolves, encapsulating the primary objective or purpose of the study. The following main research question will be used throughout the research:

"How can Quality ICT B.V. effectively integrate and leverage SentinelOne EDR platform for continuous cybersecurity monitoring in their internal web application, the QaaS App?"

Research Sub-Questions

The research sub-questions are then used to function as a pathway that dissects the main question into smaller, more manageable components, which can then be addressed individually. This approach allows for a more comprehensive and in-depth analysis of the research topic, ensuring that all relevant aspects are covered and that the research is conducted in a systematic and organized manner. This research main question is therefore expanded in the following research sub-questions:

1. What are the primary front and back-end technologies and components utilized in the QaaS web application, and how do they function together to as a system?
2. What is SentinelOne EDR Platform?
3. How can SentinelOne be integrated within the QaaS App?
4. How can SentinelOne data best be displayed to non-technical users?

2.2 Research Methodology

In this research, different research methods have been used to answer the research questions. This research will be based on the six ICT research methods defined by HBO-I (Vogel, 2023). A research method for each sub-question is then defined along with how the results are considered valid and reliable:

2.2.1 Method of Data Collection

- Sub-question #1: desk research of Literature Study will be conducted, with the goal of creating infrastructure information that displays the structure of the QaaS app and all its dependencies. Furthermore, Interview with key stakeholders involved in the development, maintenance, and usage of the QaaS app will be conducted to gain insights into the current situation of the app. Analysis on the current QaaS app and its capabilities.
- Sub-question #2: information gathering through various reliable sources on the Internet, such as Gartner and IBM for technical definitions.
- Sub-question #3: technical assessments and will be conducted, under the supervision of the Company Supervisor, to evaluate the technical feasibility, compactibility, and alignment of SentinelOne's features with the QaaS app environment.

- Sub-question #4: research into existing techniques for endpoint data coming from other EDR solutions through Literature Study. Analyze existing data visualization tools and platforms that are available in Flutter and Firebase. Gather requirements from project stakeholders regarding data visualization preferences and usability and do data analysis and usability testing.

2.2.2 Selected Measuring Instruments

- Sub-question #1: structured interview guide, document report checklist analysis and review, observation, analysis tools for codebase and logs, and quite possibly supplemented by surveys or questionnaires.
- Sub-question #2: document analysis tools for literature review. Structured questionnaires for requirement interviews regarding functionality rating scale and compare the response against industry standards and best practices. Observation of existing API monitoring tools. Prioritize functionalities based on importance, feasibility, and impact on the QaaS app.
- Sub-question #3: technical assessments and requirement workshops will be conducted. Furthermore, API documentation review, document analysis tools, security impact risk assessment, and feasibility checklist assessment with the Company Supervisor will also be overseen.
- Sub-question #4: the selected measuring instruments for this sub-question will be through observation of existing EDR solutions techniques and structured questionnaires to end-user interviews, and usability testing heuristics.

2.2.3 Method of Data Analysis

- Sub-question #1: a qualitative thematic SWOT analysis of interview transcripts and documentation for operational insights to identify strengths, weaknesses, and areas for improvement in the current situation of the QaaS app.
- Sub-question #2: comparative analyze survey/interview responses using MCDA and compare against industry standards and best practices. Prioritize functionalities based on importance, feasibility, and impact on the QaaS app.
- Sub-question #3: evaluate the technical feasibility, compactibility, and alignment of SentinelOne's features with the QaaS app environment. Analyze potential integration challenges and mitigation strategies and assess the performance of the integration through prototyping and testing. Technical analysis for the API documentation and thematic analysis for interview data.
- Sub-question #4: technical tool analysis by reviewing and evaluating the suitability of different visualization techniques for representing the data processed and received by the QaaS app in XML and JSON formats from the API considering factors such as clarity, interpretability, and user engagement. Do a user-centered design and cognitive load analysis by analyzing feedback from company stakeholders, supervisor, and end-users.

2.2.4 Reliability, Validity, and General Applicability

- Sub-question #1: the reliability of the data can be ensured by triangulation of data from multiple sources and conducting interviews with stakeholders from different departments with structure questionnaires to ensure that the data is consistent and accurate. The validity of the data will be ensured by cross-referencing with the existing literature or industry best practices or other sources and through information obtained from interviews with the QaaS app developers to ensure that the data is accurate and reliable. The general applicability of the data will be ensured by ensuring that the information obtained is relevant and applicable to the research question and that it can be used to draw meaningful conclusions and make informed decisions, furthermore by comparing findings with industry standards and best practices or similar case studies or projects.
- Sub-question #2: ensure reliability through sampling techniques and representative stakeholder involvement, with comprehensive literature review and multiple sources of information. Validate priorities against real-world scenarios or case studies involving diverse expert panels, like the Company Supervisor. General applicability can be assessed by comparing prioritization with similar projects or frameworks and considering scalability and adaptability of the integration with representative user samples.
- Sub-question #3: the validity of this sub-question will be through pilot integration unit testing or proof of concept documents and ensuring alignment with cybersecurity standards and best practices. The reliability will be to consider future needs such as adaptability and scalability of the integration and focus on Q-ICT user context and

needs. General applicability can be assessed by comparing integration strategies with industry standards or expert opinions such as from the Company Supervisor.

- Sub-question #4: reliability can be defined by ensuring future adaptability with comprehensive literature review and multiple sources of information. Validity can be achieved through validating visualization choices through data-driven approach in usability testing or prototyping, ensuring alignment with best practices in data visualization and involving the expert, like the Company Supervisor on the field. General applicability can be assessed by accessibility considerations by comparing proposed visualization techniques with similar applications or domains.

Furthermore, the reliability of these results is anchored in the application of selected research methods, including qualitative research, analysis and stakeholder interviews, which were evaluated through the CRAAP test can be seen in the Quality Assessment chapter of Project Plan located in the Appendix of this document (D).

Chapter 3

Research Results

This chapter presents the results of the research conducted to answer the research questions. The chapter is divided into two sections, each corresponding to one of the research questions. The first section presents the results of the research conducted to answer the first research question, while the second section presents the results of the research conducted to answer the second research question.

3.1 Sub-Question #1:

What are the primary technologies and components utilized in the QaaS web application, and how do they function together as a system, including both front-end and back-end infrastructure?

There are 2 main parts of the QaaS app, the front-end and the back-end. The front-end is made in Flutter, and the back-end is made in Node.js with TypeScript as the template. The back-end is hosted on Firebase Cloud Functions which are used to connect and make HTTP calls to the internal APIs, and the front-end is hosted on Firebase Hosting.

3.1.1 The QaaS App Infrastructure

Firebase

Firebase is a comprehensive platform for developing and managing web and mobile applications, created by Google and is part of GCP services. It is a BaaS that provides developers with a variety of tools and services to help with both back-end infrastructure and front-end capabilities without worrying about managing servers or infrastructures. The services offered by Firebase (*Firebase, n.d.-a*) are many, but for this thesis, it will only discuss the ones that are used by the QaaS app. They are listed in the following:

- Authentication: is an easy-to-understand authentication services that support various authentication methods like email/password, phone number, with identity providers such as Google, Facebook, Twitter, Apple, GitHub, etc., along with utilizing 2FA authentication factors to enhance security by requiring additional factor, such as an OTP code that is sent to the user's phone or security key. This functionality is used for the login screen of the QaaS App.
- Database:
  - Firestore Database: Firestore is a NoSQL database that is part of the Firebase platform. It is a flexible, scalable database for mobile, web, and server development. It keeps data in sync across client apps through real-time listeners and offers offline support for mobile and web, so the developers can build responsive apps that work regardless of network latency or Internet connectivity. This will be used to store the user data from different APIs, for example client's Microsoft subscription from Resello or Q-ICT financial data over the years from SnelStart.
- Cloud Functions: often just called Functions in the Firebase console, it allows developers to run back-end code in response to events triggered by Firebase features and HTTPS requests. The code is stored in Google's cloud and runs in a managed environment. It is a serverless framework that allows developers to build and deploy serverless functions that automatically scale up and down based on demand. The available programming languages are Node.js (JS and TS), Python, Go, Java, and .NET (C#). Cloud Functions offers 2 product versions: the original version (1st gen), and the 2nd gen which is built on Cloud Run and Eventarc to provide an enhanced feature set.

- 1st Generation: Most of the Firebase Cloud Functions that are used in the QaaS app is in this version. The company wishes to migrate all the functions to the 2nd generation in the future. Furthermore, for the integration of SentinelOne with the QaaS app, the company wishes to utilize the 2nd generation of Cloud Functions.
- 2nd Generation: The company wishes that the author's graduation project will utilize the 2nd generation of Cloud Functions.

Table below shows the feature comparison between the 1st gen and the newest 2nd gen cloud function from Firebase

Feature	1st Gen	2nd Gen
Image registry	Container Registry or Artifact Registry	Artifact Registry only
Request timeout	Up to 9 minutes	• Up to 60 minutes for HTTP-triggered functions • Up to 9 minutes for event-triggered functions
Instance Size	Up to 8GB RAM with 2 vvCPU	Up to 16GB RAM with 4 vCPU
Concurrency	1 concurrent request per functions instance	Up to 1000 concurrent requests per function instance

Table 3.1: Comparison between the 1st and 2nd Generation of Cloud Functions in Firebase

Cloud functions are the main back-end infrastructure of the QaaS app. It is used to connect and make HTTP calls to all the internal APIs. There are different types of functions in Firebase Cloud Functions, and they will be discussed later. Some functions are called within the app, and some outside of the app. Some functions are also used to listen to the Firestore collections, in case of any changes in the data.

- Hosting: a service that allows developers to host static websites, dynamic web apps, mobile apps, and microservices on Firebase's infrastructure. The QaaS app is currently hosted on Firebase Hosting.
- Cloud Storage: offers secure, scalable, and reliable file storage and sharing for Firebase apps. It is designed to help developers quickly and easily store and serve user-generated content, such as photos or videos. It is used by the QaaS app to store and serve user-generated content, such as profile pictures, documents, and other files.
- Extensions: it consists pre-built, open-source software packages that extend the functionality of a Firebase project (*Firebase, n.d.-b*). They are designed to automate common development tasks, such as sending notifications, integrating with third-party services, and performing back-end operations, without requiring users to write custom code. The QaaS app uses the Extension mainly for Algolia.
- App Check: it is a security feature that, on top of Firebase 2FA Authentication, that helps protect the project from abuse, such as billing fraud or phishing, by ensuring that only the app that is registered can have access to the Firebase project's resources (*Firebase, n.d.-c*). In the case of the QaaS app, it uses it reCAPTCHA to ensure extra protection in the MFA.

Below are all the products offered by Firebase to their clients as a cloud development solution:

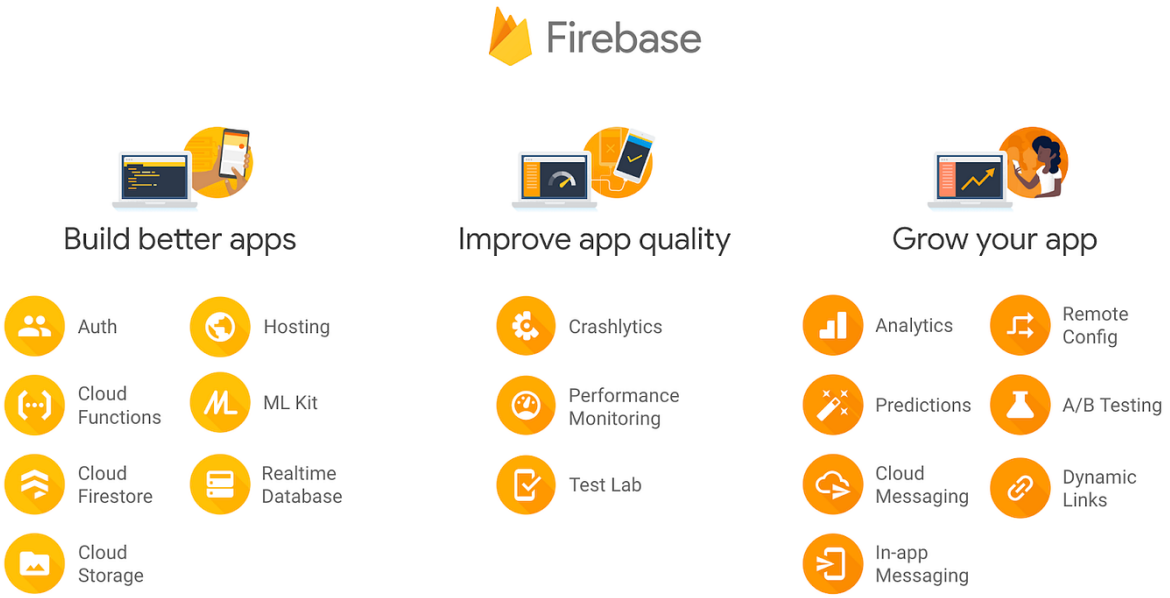


Figure 3.1: All the products offered by Firebase (*Kubernetes, 2018*)

Different Types of Cloud Functions in Firebase

Firebase has a lot of different types of Cloud Functions that developers can use. Just like the previous Firebase product explanation, this thesis will only focus on the following types of Cloud Functions that are used in the QaaS app:

HTTP Triggers: these are functions that are triggered by HTTP requests. They are used as API endpoints of the QaaS app, allowing server-side logic execution in response to HTTP requests from client-side applications or external services. The requests are GET, POST, PUT, DELETE, and PATCH, and they are used to creating, reading, updating, and deleting data in either the Firestore DB or the correlated API environment itself.

- onRequest: this method is used to create an HTTP function that is triggered by an HTTP request. They are more general-purpose compared to Callable Functions and can be used to create RESTful APIs, handle form submissions or perform any other server-side operations that require HTTP requests. Unlike Callable Functions, onRequest functions do not handle authentication or data serialization, so the developers need to manage this aspect manually. This functions only accepts `Request` and `Response`, but not `NextFunction`. In the context of QaaS app, this function is only used for testing purposes, and the developer will need to migrate to onCall when deploying on the Live environment. Below is an example of an onRequest function which needs to take 2 parameters, request and response:

Listing 3.1: Example of a typical onRequest function

```
import { Request, Response } from 'express';
import * as functions from "firebase-functions/v2";

const region = "europe-west1";

export.getData = functions.https.onRequest({ region: region }, async (request: Request, response:
    Response): Promise<any> => {
    try {
        const response = SentinelOneAPICall();
        return response.status(200).json(data: response.data);
    } catch (ex: unknown) {
        if (error instanceof Error) {
            // Error object, log message and stack if available
            console.error(`[${context}] An error occurred: ${error.message} \n Stack:
                ${error.stack}`);
        } else {
            // Non-Error object, log with a generic message.
```

```
                console.error(`[${context}] An unknown error occurred:`, error);
            }
            response.status(500).send("Failed to retrieve agents");
        }
    });
```

- onCall: is a little different from `onRequest`. Instead of using `Request` and `Response`, it uses `data` and `context`. In version 2.0, it only accepts requests as `CallableRequest<any>` that can get any headers and body of the request sent by user. It is used to create Callable Functions, and they are designed to be called directly from client applications, such as mobile or web apps. They automatically handle authentication and data serialization, making it easier to secure call backend code from client applications, and this is what the QaaS app primarily uses for calling its internal APIs as it ensures that the client is authenticated and authorized to make the call. Callable functions are triggered by an HTTP request but are specifically designed to be called from Firebase client SDKs. Below is an example of onCall, which only takes 1 parameter and has authorization check to make sure not any party can access the cloud function.

Listing 3.2: Example of onCall function with authentication check

```
import * as functions from "firebase-functions/v2";
const region = "europe-west1";

const getData = functions.https.onCall({ region: region }, async (request: CallableRequest<any>) =>
    {
    try {
        // Checking that the user is authenticated.
        if (!context.auth) {
            // Throwing an HttpsError so that the client gets the error details.
            throw new functions.https.HttpsError('failed-precondition', 'The function must be
                called while authenticated.');
        }
        const response = SentinelOneAPICall();
        return {
            data: response.data,
        };
    } catch (ex: unknown) {
        if (error instanceof Error) {
            // Error object, log message and stack if available
            console.error(`[${context}] An error occurred: ${error.message} \n Stack:
                ${error.stack}`);
        } else {
            // Non-Error object, log with a generic message.
            console.error(`[${context}] An unknown error occurred:`, error);
        }
        throw new functions.https.HttpsError("unknown", "Failed to retrieve agents", ex);
    }
});

async function SentinelOneAPICall() : Promise<any> {
    const apiUrl = 'https://sentinelOne.example.com/users/123';

    // Define the request parameters
    const requestOptions = {
        method: 'GET', // HTTP method (GET, POST, PUT, DELETE, PATCH, etc.)
        headers: {
            'Content-Type': 'application/json', // Set the content type of the request
        }
    };

    // Make the API request
    fetch(apiUrl, requestOptions)
        .then(response => response.json())
        .then(data => console.log(data)) // Process the response data
        .catch(error => console.log('error', error)); // Handle any errors that occurred during
            the request
}

export = getData;
```

Pub/Sub Triggers: are the functions triggered by messages published to a Pub/Sub topic. It is a messaging service that enables decoupling of applications by sending messages between independent components.

Scheduled functions: this is a Firebase's own term for Cron Job (*Firebase, n.d.-f*). They are used within the QaaS app to run tasks at regular intervals, such as sending reminders, keeping data up-to-date with the internal APIs (like customer list), or performing maintenance tasks.

Figure 3.2: The *Scheduled Function* in the QaaS app that is scheduled to run every 7 days to keep the API data up-to-date (*Cronitor, n.d.*)

Algolia

Algolia is used for search functionality. It is a search-as-a-service platform that enables developers to integrate and build fast, relevant search functionality into their applications and websites (*Algolia, n.d.*). It provides a range of features and capabilities for building and managing search functionality, including full-text search, typo tolerance, and relevance tuning, as well as analytics and monitoring tools to help developers understand how users are interacting with their search functionality in real-time.

How it works basically is that an extension should be installed on the Firebase that will create a customized indices (mirror table) reflecting the data on the Firestore database. An Application ID needs to be provided from Algolia to Firebase to that it will know which environment to use. Next to that, an API key needs also to be given. A developer can create their own API key that will have a proper read, write, or delete as they see fit. It is also important to note that it is not considered best practice to always give admin permission to every extension, because some limitations need to be set to users so that they cannot do such actions that can harm their own data. Below are the Algolia extensions that are already installed and used within the QaaS App in its Firebase channel:

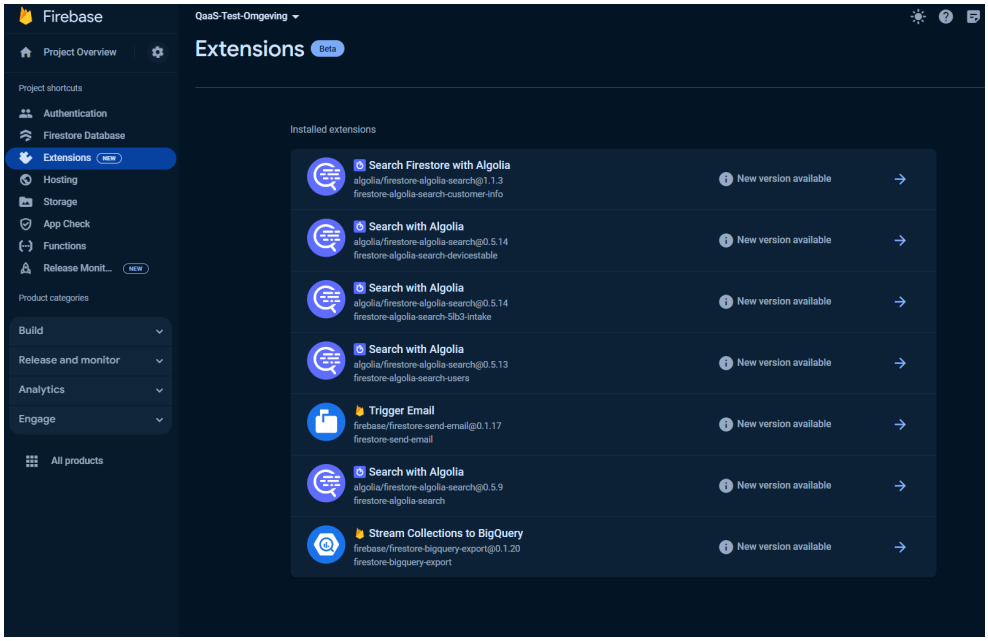


Figure 3.3: All the extensions that are used in the QaaS app, mostly about Algolia

The reason as to why Q-ICT uses Algolia is that the nature of Firebase search engine is quite often proven to be inaccurate and slow. It uses the traditional search engine that will look through fields in JSON format because of its NoSQL nature. It often struggles with misspelled words or incorrect queries. Algolia has a standout feature that to handle user typos and errors gracefully through its AI-powered search algorithms. Users often-times do not like to type the exact words for something that they want to search for. This means that they make typos or enter incomplete search terms or their abbreviation. Algolia's intelligent algorithms can interpret the intent behind the search and return accurate, relevant results.

The Company Supervisor has expressed a desire to use Algolia in the QaaS app in multiple interviews the author has

undertaken for the analysis and designing of this project. It is a nice-to-have feature that promotes user-friendliness and ease of use.

Google Secret Manager

It is a fully managed service provided by GCP that allows developers and organization to securely store, access, and manage sensitive information such as API keys, passwords, certificates, DB credentials and other credentials used in throughout the lifecycle of their applications (*Google, n.d.-c*). It is not part of Firebase, and it helps the QaaS app to centralize and secure its secrets in scalable and easily manageable way. Key-features of Secret Manager include:

- Secure Storage: it encrypts the secret values using CMEK, ensuring the sensitive data is protected both at rest and in transit.
- Audit Logs: it provides and manages audit logs that record all access and modification of activities, helping developers meet compliance, better accountability and regulatory requirements.
- Versioning and Automatic Rotation: it supports versioning of secrets, allowing developers to store multiple versions of the same secret. This means that the developers get to keep multiple versions of secrets and easily revert or roll back to a previous version if needed, which will help in auditing and tracking changes to secrets over time. This feature enables automatic seamless rotation of secrets at regular intervals without disrupting the applications, which improves the security part of the application by ensuring that secrets are regularly updated without manual intervention.
- Access Control: it provides fine-grained access control using Google IAM, allowing developers to specify who can access and manage the stored secrets and what they can do with them.
- Centralized Management: it stores and manages all secrets in one place, simplifying access and control.

Google Secret Manager comes from GCP, and GCP and Firebase are a separate cloud solution. But, because both are part of Google, Firebase Cloud Functions can typically access GCP Secret Manager by editing that specific function that the developer wanted to grant access to. In the context of the QaaS App, Secret Manager is mostly used to store the API keys and tokens.

Modules, User Tags, and Templates of the QaaS App

The QaaS has several user tags that are used to build the app. The tags can be interpreted as level of access to the app, showing what permission a user has to the app. Because of coding inconsistency, there are 2 naming of functionality, which are user tags and user roles, both serving the same purpose. User tags can be assigned or revoked to a user by the IT admin willingly. User Tags can be revoked or assigned to users by IT admins in ”Account Beheer” module. The user tags in the QaaS App are the following:

- Account Management
- Orderer
- Budget Adjusment
- Budget Overview
- Helpdesk
- IDS (for seeing IDS module)
- Intake Overview
- Intake Deletion
- Packing Slip Admin
- Period Comparison Overview
- Update Period Comparison
- User

Modules are the page division of the QaaS app. They are interlinked with User Tags to create the navigation logic of the app. All pages will have different functions, corresponding to its interconnected API. Examples of modules in the QaaS app are:

- Customer Connection
- IDS System
- Budget Overview
- Intake Overview
- Account Management
- Periode Comparison
- Clients
- Security (Bodyguard.io)
- Apparaten (N-Central)
- Subscriptions
- Billing
- Intake
- Packing Slips

Templates are about how the styling and the placement of elements of each page will look based on client's company ID, user tags, or permissions. Officially, a user with a common "User" tag can only see 2 templates of the App, which are light and dark mode. The templates might also vary to different clients, for example Kavos ICT, that has requested Q-ICT to consolidates the QaaS App capabilities into a single unified interface that simplifies navigation and enhances user workflow. They also have requested that the QaaS App to follow their colour palette more than Q-ICT (which namely consist of green, yellow, purple, and white).

3.1.2 Conclusion

The QaaS app is an ERP web application that is used by Q-ICT and its clients. It is made in Dart with Flutter as the front-end framework, and Node.js with the back-end framework with TypeScript as a template to ensure type safety. The app uses several internal APIs and technologies to help with its operations, such as Resello, SnelStart, Bodyguard.io, N-Central, and PerfectView. They all have different purposes that in which make up the functionalities of the QaaS app. The back-end of Q-ICT is Firebase that serves the cloud solution. Firebase Cloud Functions are the primarily components used for making different HTTP connections to those 5 APIs. Firebase Hosting is used to host the front-end web application from the Flutter framework. The app also uses other Firebase products, such as Authentication, Firestore Database, Cloud Functions, Hosting, Cloud Storage, Extensions, App Check, and Google Secret Manager. The app also uses Algolia for search functionality. The app has several templates and modules that are used for the authentication and authorization of the app. The User tags can be viewed as a level of authorization within the QaaS App, and the modules are the pages with different functionalities that the user can have access to. A user tag can be assigned to a module so that user accounts that have that user tag can view the module. Together, they with the control flow of the app's features and functionalities, and the IT Admin has the highest level of access and control over the app.

3.2 Sub-Question #2:

What is SentinelOne EDR Platform?

SentinelOne is a cybersecurity platform that provides endpoint protection, detection, and response capabilities to help organizations defend against advanced cyber threats. It leverages AI and ML to analyze and respond to security threats in real-time, providing organizations with comprehensive protection against malware, ransomware, and other cyber threats. It also provides visibility into clients' IT systems and infrastructure, enabling organizations to gain insights into potential security risks and vulnerabilities and take proactive measures to address them. SentinelOne

itself has emerged as one of the leading EDR platforms in cybersecurity. Figure below shows the SentinelOne place in the Gartner magic quadrant about cybersecurity solutions

Figure 3.4: SentinelOne in the Gartner magic quadrant that proves that it is one of the leading solutions in cybersecurity

In this research, it will delve into what EDR solution is and how it can help organizations protect their IT systems against more modern and advanced cyber threats. Some terminology that the readers need to be familiar with before diving deeper into SentinelOne:

Endpoint

Endpoint can be defined as any remote computing device that receives incoming communications and sends outgoing messages to the network it is connected to. Examples of endpoints include desktops, laptops, smartphones, tablets, servers, workstations, and other IoT devices that is connected to a network. They are the first-line of defence for the Blue Team today.

Examples of endpoints are:

- Computer (workstations, desktops)
- Laptop
- Server
- Mobile devices

EDR

EDR A.K.A. ETDR, is a group of integrated endpoint security solutions that combine data collection, data analysis, forensics, and Threat Hunting with the end-goal of identifying and stopping any potential security breaches in due time. EDR solutions can recognize any suspicious patterns that can be investigated later, as they have been purposefully created to detect and respond in an active manner to advance malware, ransomware, and other cyber threats (the Response EDR). EDR, as the name suggest, were developed specifically for endpoints, and not networks, thus operate only on endpoint level.

The number one thing that sets apart EDR from traditional AV is that traditional AV relies on signature-based detection, usually having a defined set of list in their DB, where known malware signatures are compared against files or processes to identify threats. EDR on the other hand, uses a combination of signature-based detection, such as behavioural analysis, machine learning, and anomaly detection to identify and respond both known and unknown threats. EDR solutions focus on detecting malicious activities at the endpoint level, including file modifications,

process execution, and network connections, focusing on malicious behaviour compared to only concerning with malicious software like what traditional AV does. While this seems similar to NGAV, what sets apart EDR from NGAV is that EDR solutions are more focused on detecting and responding to threats after they have infiltrated the network, whereas NGAV aims only to prevent threats from executing in the first place through advanced detection techniques, a functionality that is crucial in an AV. Figure 3.6 tells how EDR connects to other modern cybersecurity solutions.

SentinelOne Console

SentinelOne MGMT Console is the GUI provided by SentinelOne for managing and administering the platform. It serves as the primary dashboard for security operators and administrators to interact with SentinelOne product suite, enabling them to monitor, configure, and respond to cybersecurity threats across their organization's endpoints, servers, and cloud environments.

Global

The Global tab refers to the environment of the highest level of advisors. It consists of organizations that have access to global level that propagate down to the tenant level, which means they can put policies, scripts, and other configurations that will be applied to all consoles and all their customers.

Site

Site is just a name that refers to logical grouping of devices that are managed together under a single administrative domain, or a company. A company customer can have multiple sites, and each site can have multiple devices. Each site ID is unique and is used to identify the site in the SentinelOne console.

Group

Group refers to logical grouping of devices within SentinelOne MGMT Console. It is a way to organize and manage collections of endpoints, servers, and other entities within the platform. It allows admins to apply policies, deploy agents, and perform other management tasks across a defined set of assets in an organized and efficient manner. This hierarchical structure is crucial for managing large-scale deployments and ensuring that security policies and configurations are consistently applied across relevant assets. Groups can be created and deleted by the admin user, and devices can be moved between groups. An example of a group in Q-ICT site is:

- Default Group
- Laptops
- Workstations
- Servers

Agents

An Agent is a software program, part of SentinelOne product, that is deployed to each endpoint, including desktop, laptop, server or virtual environment, and runs autonomously on each device, without reliance on Internet connection, enabling data gathering, detection, and response to actions. Agents can be interpreted as an AV, collecting relevant security telemetry such as:

- Running processes
- Connected servers
- Open files

This information can be useful to detect the presence of a threat or to use in forensic analysis and investigation after an attack has occurred (Recovery).

Ranger

Ranger is an add-on SentinelOne product that provides a way of detecting other devices (computers and IoT devices) that are on the client's computer network (*SentinelOne, n.d.-b*). If a malicious attacker comes in and plugs his device into the network, all the other SentinelOne agents are going to read the network traffic, determine and classify whether that is a new device or a rogue device. If a device has Ranger on that network subnet, SentinelOne can gather and detect technical information regarding the device.

- Not reviewed

- Not trusted
- Under analysis
- Allowed

Ranger is designed to detect and take out malicious actors that are in the local subnet, as they can a lot of information about the devices (*see ARP Poisoning Veracode, n.d. and man-in-the-middle-attack Wikipedia, n.d.*). In the real-business scenario, a lot of the times, a company has a very secure perimeter firewall (*the big outer castle wall in castle-and-moat network security model Taylor, 2021*), but the inside network are wide open (unless they are doing logical separations of their local network, such as using VLAN) for attacks inside the network.

Sentinels

Sentinels refer to a term that describes SentinelOne ability to deploy and manage security agents on endpoints within an organization, not part of SentinelOne product, like Ranger. This is part of the SentinelOne EDR capabilities, where the user admin can deploy the agents on the endpoints and manage them from the SentinelOne console. The admin can also create policies, scripts, and other configurations that will be applied to all the agents in the network.

Visibility

Unfortunately, Q-ICT does not have Visibility turned on its tenant, thus this feature is also off limit for this project scope. But in general, Visibility allows users to do deep querying to be able to identify different things, such as attributes of a machine. It is useful when user is looking for threats or any additional information in Incident Response or Threat Hunting, which in most cases is proactive, Visibility can give the user a lot of power to use queries, and interrogate all the information or telemetry that SentinelOne has pulled off from the connected machines.

Incidents

Incident is just the name of a page in SentinelOne dashboard that provides a list of cyber-incidents overview that have happened on all endpoints detected by Agents in the network connected to SentinelOne by Ranger. The page typically offers an overview of all detected security incidents, categorized based on severity levels of types of threats.

Once a threat is detected, the user can take the following actions in the dashboard, if they have enough privileges to do so:

- Kill: stops all processes related to that threat
- Quarantine: encrypts and moves the threat and its executables
- Remediate: deletes all files and system changes created by the threat
- Rollback: restores files and configuration that the threat changed. This step is usually taken when a malware has executed its script and has made changes to the system, e.g., a ransomware has encrypted all the files and asked for a ransom. By taking this step, all the three previous steps will also be undertaken as well. This will then reboot the system and restore it to the safe state before the malware has been executed.

Reports

The reports in SentinelOne provide users with insights into the security posture and threat landscape across their organization's endpoints. The reports offer customizable reporting capabilities, allowing users to generate reports tailored to their specific requirements. Users can then choose from a variety of predefined report templates or create custom reports based on their unique needs.

Furthermore, the users can also choose for the report to be made automatically, instead of manually filling them themselves. They can schedule automated report generation at regular intervals, such as daily, weekly, or monthly.

3.2.1 Conclusion

SentinelOne is a cybersecurity organization that provides EDR solutions for endpoint protection, detection, and response capabilities to help organizations defend against advanced cyber threats. It leverages AI and machine learning to analyze and respond to security threats in real-time, providing organizations with comprehensive protection against malware, ransomware, and other cyber threats. SentinelOne provides a wide range of features and capabilities, including endpoint protection, detection, and response, as well as visibility into clients' IT systems and infrastructure. SentinelOne has multiple components, such as Agents that are responsible for collecting security telemetry of an endpoint, Ranger that is responsible for detecting other devices on the network, and other components

that work together to provide comprehensive security solutions for organizations. It is famous for its autonomous features and cloud-native architecture, make it well-suited for securing modern IT environments against evolving cyber threats.

The integration of SentinelOne to the QaaS App will enhance its monitoring capabilities by providing real-time, comprehensive visibility into the security status of clients' devices. This allows for aggregation of detailed threat intelligence and device health information, enabling the app to display critical security data such as threat detections, mitigations, and the overall security posture of each monitored device. By leveraging SentinelOne's advanced threat detection and response features, the QaaS App can offer more proactive and efficient incident management, ensuring that potential threats are swiftly identified and addressed. This not only boosts transparency for clients, allowing them to view and understand insights into their device security, thereby enhancing their trust and confidence in the cybersecurity provided by Q-ICT.

3.3 Sub-Question #3:

How can SentinelOne be integrated with the QaaS app?

The integration of SentinelOne into the QaaS App is viable, leveraging the company's subscription through Pax8, a partner of both Microsoft and Q-ICT. This integration involves several key steps and considerations to ensure a seamless and secure implementation.

Firstly, the deployment of SentinelOne Agents across the clients' endpoints is essential. These Agents are responsible for collecting comprehensive endpoint data, including system health information, malware incidents, network connections, and outdated applications posing security risks according to latest CVEs from the NIST and MITRE. This data is then transmitted into the SentinelOne platform, forming a robust dataset for threat analysis and endpoint management.

Once the Agents are operational and data is flowing into the SentinelOne ecosystem, the integration process involves retrieving the real-time data into the QaaS environment itself. This is where the SentinelOne API come from. SentinelOne provides a RESTful API server to allow users to integrate SentinelOne with other security tools and systems.

SentinelOne also provides a comprehensive documentation for its RESTful API that allows users to interact with the SentinelOne platform programmatically. The API provides a wide range of functionalities, including the ability to retrieve information about devices, incidents, and threats, as well as the ability to perform actions such as quarantining devices, remediating threats, and generating reports. The API is designed to work with the Agents, Sentinels, Ranger, and other components of the SentinelOne platform.

The API sends requests to the associated Management Server and responds with the data that the management pulled from the Agents or from the management database. Therefore, all the data that is displayed in the SentinelOne Console dashboard is also available through the API.

3.3.1 SentinelOne API Token

To be able to access SentinelOne APIs, the user needs to have an API token to prove their identity and that their organization has SentinelOne subscription, therefore has right to access data. API tokens can be created in the SentinelOne MGMT Console.

The API token will be shown only once; therefore, the user needs to store it in a secure place. The token generated by the user is time-limited. To renew the token, the user must generate a new one with the same above-steps. The API token will need to be placed in secure environment, where it is there in the cloud to be accessed anywhere but not any parties can have access to it by having an encryption. This is where Google Secret Manager will come into play.

3.3.2 SentinelOne Data Types

There are more than 300 API calls that are available in SentinelOne's server. The types of API call that can be done can be seen in Appendix F of this document (SentinelOne API documentation itself). The author will need to further with Company Supervisor and Project Stakeholder in the Realization phase to discuss the data types that will be useful to show to the users, mainly with the cybersecurity department to ensure that no sensitive information such as internal IP addresses will be shown in the dashboard. The list below will provide an overview of what kind of data or interactions that can be done using SentinelOne API:

GET requests:

The data that will be showed in the dashboard mainly are the following:

- **Agents**: containing information about their devices, such as are they infected or not, their firewall status, installer types, hardware configuration (regarding RAM, CPU types and core), are they active, their SentinelOne Agent versions, their external and internal IP address.
  - Agent applications: will be about the applications installed on the specific endpoint. This data will be useful so that the manager of client's company can see and have access to what kind of applications his/her employees installed in their work laptop.
- **Threat**: containing information about the incidents that occurred on the client's endpoint that have been prevented by SentinelOne. This data will be useful so that the clients will know what are previous malware that tried to infect their system along with their details and how they were prevented. There will be a date filtering function to this data that user can choose from, so that it can display the threats that have happened one week, month, or year ago. This is because not all previous data are valuable concerning the threats that have happened long in the past (5 years ago).
  - Alerts: are notifications generated by SentinelOne when the system detects a potential security incident or suspicious activity that are not yet solved that might have a severe impact on an endpoint. They are designed to draw attention to events that may require further investigation or remediation.
  - Threat timeline: provides a chronological view of events related to a detected threat. This will be in form of event sequence of all events associated with the threat, providing with detailed analysis and information for each event in the timeline, including process execution, file changes, and network connections. This will help the client identify the root cause analysis by tracing the event back to the initial point of compromise.
  - Threat process events: are the individual actions or activities that occur as part of a detected threat. These events include various actions such as:
    1. Files: are events involved in actions taken on files within the system. This includes modifications, deletions, creations, and other actions that might indicate malicious activities. It is important to monitor these changes of files to help maintain data integrity and detect any tampering.
    2. Process: related actions taken by or against running processes. This includes process creation, execution, termination, and other interactions that may indicate malicious behaviour. It is crucial in identifying unusual or unauthorized process activities that deviate from normal behaviour.
    3. Indicators: also known as IoC, are pieces or forensic data that identify potentially malicious activity. These can include file hashes, IP addresses, domain names, URLs, and other artefacts associated with threats.
    4. Network actions: network-related events such as IP connect, data transfers, port scanning, DNS request, and other malicious activities.
- **Potential Risky Applications**: detailing lists of outdated software that have proven to have unknown vulnerabilities in the cybersecurity world. With this data, clients can see which applications on their devices are outdated, allowing them to prioritize patching and updates and ensure compliance.
  - CVE data: will show the CVE details issued by NIST and MITRE so that the user can get a better understanding on how the outdated application might be dangerous. The explanation is very technical, iterating over security bugs that may be related to the outdated application.
- **Ranger**: provides information about network security especially devices connected to the network, including those not managed by SentinelOne. This data helps to identify rogues devices or network that might pose security risk
- **Rogue**: refers to unauthorized or unmanaged devices on a network. These devices may not have Agents installed but are connected to the network, making them potential security risk as they can harbour vulnerabilities or be used as entry points for attacks.
- **Activity Logs**: captures all actions and events monitored by Agents, including user activities, system changes, and security incidents. These will be useful to the clients tracking user activities for user accountability and giving them better overview of their operational insights to their system.

POST, PUT, PATCH, and DELETE requests

These types of API requests will modify data within the system. Some of API calls that can be requested from SentinelOne API are namely:

1. Create, update, and delete exclusions which will flag a certain file within the whitelist. This will then prevent SentinelOne Agents to quarantine any files that the AI deems suspicious. It can create exclusions/exceptions based on file hash (SHA1), path or file name specified, types of browser the files are coming from, file types (.txt or .docx), and which vendors or certificate the files are coming from.
2. Various Agent actions, such as:
   (a) Decommission
   (b) Disable
   (c) Enable
   (d) Restart
   (e) Shutdown
   (f) Uninstall certain files
   (g) Update the Agent
   (h) Fetch files and firewall logs
   (i) Displaying alert
   (j) Initiate a scan within a network
   (k) Quarantine a file
   (l) Remediate and rollback the operating system
3. Create, update, and delete blocklist items which will block certain files from being downloaded from the internet. The use case of this call is the exact opposite of what exclusions (whitelisting) do.
4. Create, update, expire, and delete accounts
5. Create, update, delete Groups in SentinelOne
6. Update and revert policies
7. Update threat analyst verdict
8. Update the status of a threat

3.3.3 Conclusion

SentinelOne provides a RESTful API that allows users to interact with the SentinelOne platform programmatically, enabling integration with other security tools and systems. The API provides a wide range of functionalities, including the ability to retrieve information about devices, incidents, and threats, as well as the ability to perform actions such as quarantining devices, remediating threats, and generating reports. The author will use the official SentinelOne API documentation to integrate SentinelOne with the QaaS app to display the data of Q-ICT and its 500 clients in the QaaS app. The functionality such as filtering, sorting, and pagination will also be considered to make the data more user-friendly. The author will utilize different kinds of Firebase Cloud Functions (HTTP onCall, Pub-/Sub Triggers, Schedule functions) to call the SentinelOne APIs and display data. Firestore will also be used to store the user's preference from the QaaS app, such as the user's choice of what data they want to see, and visualization types. The author will also utilize Google Secret Manager to store the API token and other sensitive information securely, so it will not be displayed in the code to avoid any security risks. Furthermore, AI powered search engine Algolia will be used to search for specific data in the QaaS app, as well as Firestore to make the search faster and more accurate.

3.4 Sub-Question #4:

How can SentinelOne data best be displayed to non-technical users?

3.4.1 Comparison to other EDR solutions

To determine the best techniques to display SentinelOne data to the users, a comparison to other EDR solutions is needed. In this section, the author has looked into other alternatives to SentinelOne, which are a single EDR solution, created as a complete replacement of legacy AV. Please keep in mind that in this sub-question, only visualization techniques and features of the solutions will be assessed, compared, contrasted, evaluated, and discussed. Factors such as pricing and technologies will not be discussed in this sub-question. The author also does not have access to the actual dashboards; therefore, pictures of the dashboards are likewise taken from the Internet or are available from the official website. This is because Q-ICT is not utilizing these solutions, setting up an account and getting access to the dashboards will raise the project's budget, which is not feasible.

CrowdStrike

CrowdStrike is an American cybersecurity company that provides cloud workload and endpoint security, threat intelligence, and cyberattack response service. In the dashboard, CrowdStrike offers fully customizable widgets to ensure that the various roles on security teams are provided with the information that they need. The dashboard allows static prebuilt widgets that can be moved around on a page, and customizable data views that enable each individual widget to be tailored for the users.

Features of CrowdStrike dashboard including:

- **Role assignment**: allows the user to assign roles to the users, so that they can see the data that is relevant to their role. For example, IT admins are allowed to have an eye on the status of the CrowdStrike Falcon Sensors within the environment. This enables them to understand if there are any gaps and allows them to remediate any issues. A SOC analysts, executives, and vulnerability analysts can easily see any security threats detected or blocked.
- **Fully customizable**: widgets can be customized to display just the right data needed. Users can add filters and change the lens with which to view the data, from donut to line charts.
- **Advance dynamic filtering options**: filters can be applied dynamically to the dashboard, with data from a single source. This could help analysts focus on the most important criteria in their Threat Hunting.
- **Customized Scheduled Reports**: reports based off dashboards or other data sources can be generated and sent automatically on a specified schedule.

Figure below is the CrowdStrike dashboard looks like.

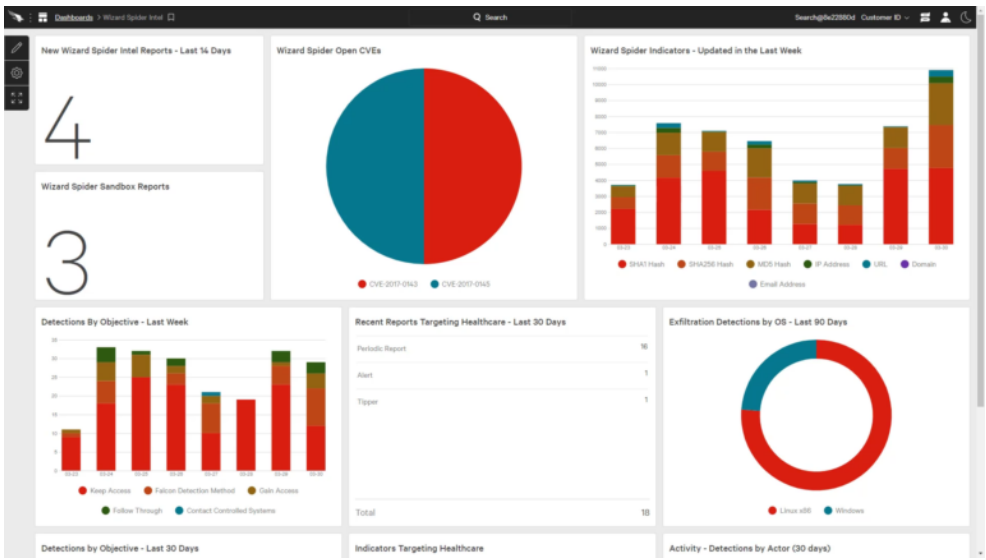


Figure 3.5: Visualization widgets in CrowdStrike

Figure below shows us how to add and edit widgets in CrowdStrike.

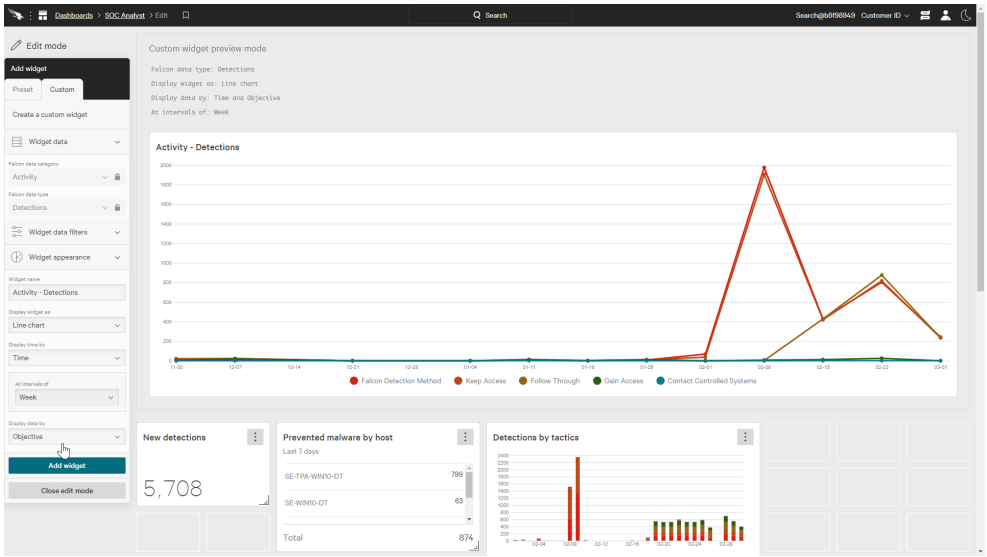


Figure 3.6: Adding a widget in CrowdStrike

Figure below shows us how to create a scheduled report in CrowdStrike.

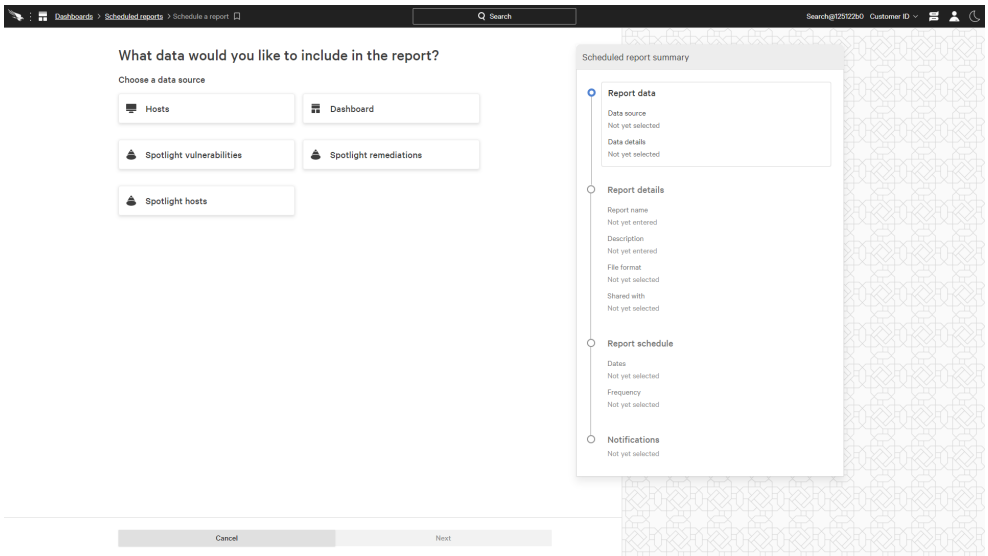


Figure 3.7: Create scheduled report in CrowdStrike

Trend Micro

It is an American Japanese cybersecurity software company, providing cloud and enterprise cybersecurity in environments like AWS, Microsoft, and Google Trend Micro Apex One has customizable contents and layout of their dashboard. Workload Security uses Session to save user's settings and remember the last view of the dashboard the next time the user logs in. The colour shown in the dashboard is more basic compared to the other solution with only the typical red, green, blue, and yellow. Components of the Trend Micro dashboard is:

- **Tabs**: are the container for widgets. Tabs can be added or modified as needed. A dashboard can support up to 10 tags, and each tab can hold up to 10 widgets.
- **Widgets**: like other solutions, widgets represent the core components of the dashboard. They contain visual charts and graphs that track threats and associate them with log statistics aggregated from one or several gateways.
- **Event**: contains a log of record when a protection module rule or condition is triggered by Deep Security Agents. CrowdStrike Agents and Deep Security Manager also records when administrative or system-related events occur

(a ”system event”), such as administrator logging, or agent software being upgraded. Even data is used to populate the various reports and graphs in Deep Security Manager.

- **Filter by tags**: in CrowdStrike Deep Security®, a tag is a unit of meta-data that can be applied to an Event to create an additional attribute for the Event that is not originally contained within the Event itself. Tags can be used to filter Events to simplify the task of the Event monitoring and management. A typical use of tagging is to distinguish between Events that require action and those that have been investigated and found to be benign.
- **Filter by date and time range**: allows the user to filter the data by date and time range. For example, data can be displayed from either the last 24 hours, or the last 7 days.

Figure below shows us how Trend Micro dashboard looks like.

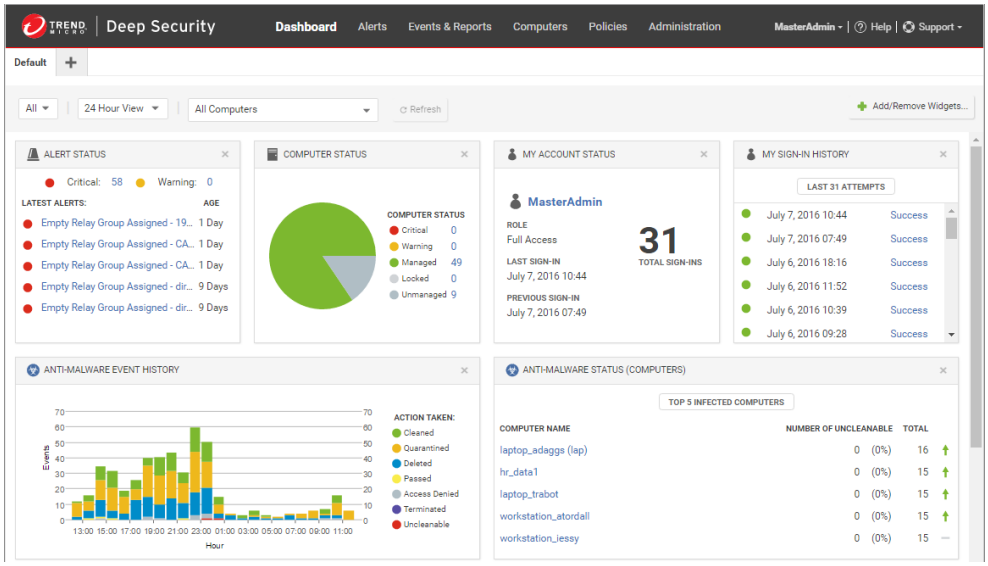


Figure 3.8: Dashboard in Trend Micro

In Trend Micro, tabs can be added and deleted. It will contain up to 10 visualization widgets, which are shown in the following:

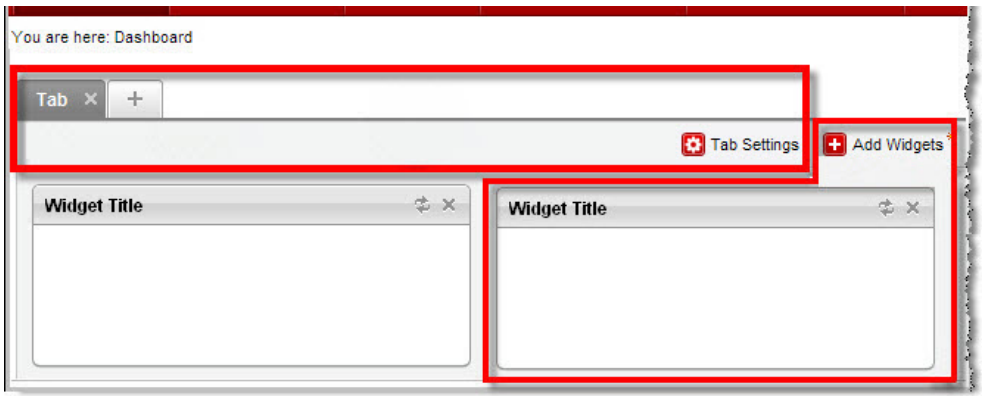


Figure 3.9: A Tab in Trend Micro can contain up to 10 widgets

Here are the layout of the dashboard, where users can drag-and-drop their selected widgets.

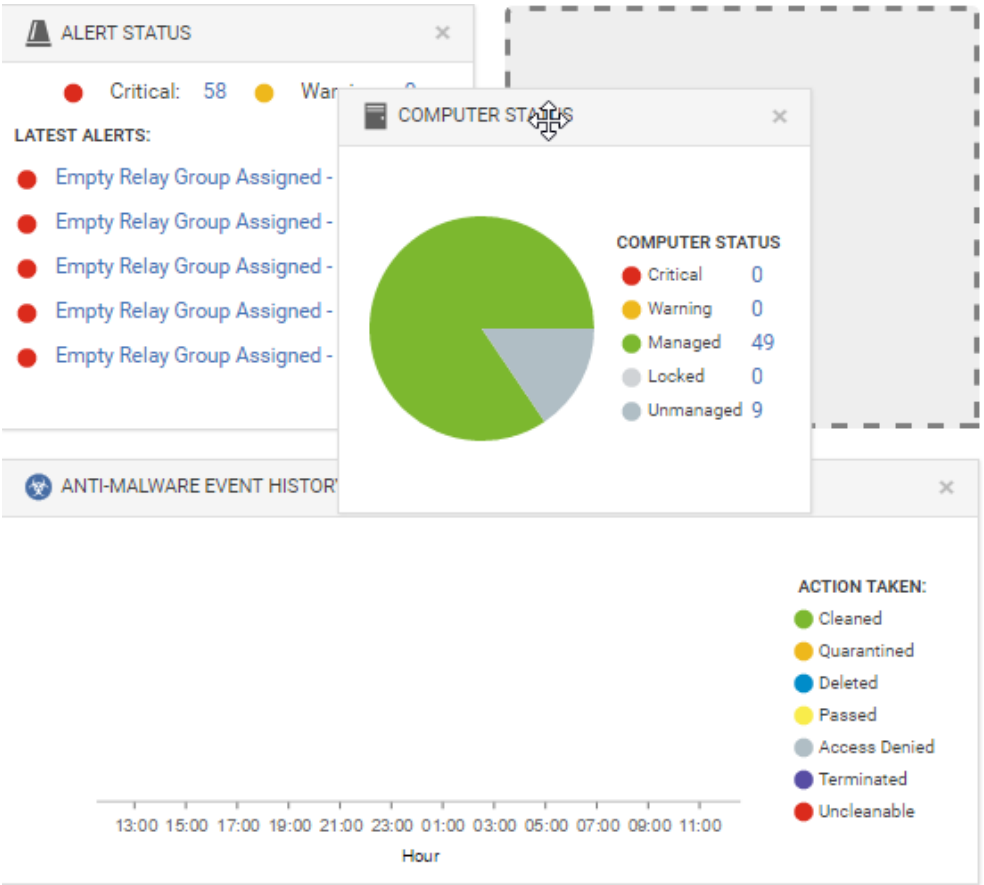


Figure 3.10: Change the layout of the dashboard in Trend Micro by dragging and dropping the widgets

The threat data can be filtered by their date and time range, to ensure the user will get the latest data, and not outdated threat data that has been collected in the distance past, like 5 years ago, which are shown in the following:

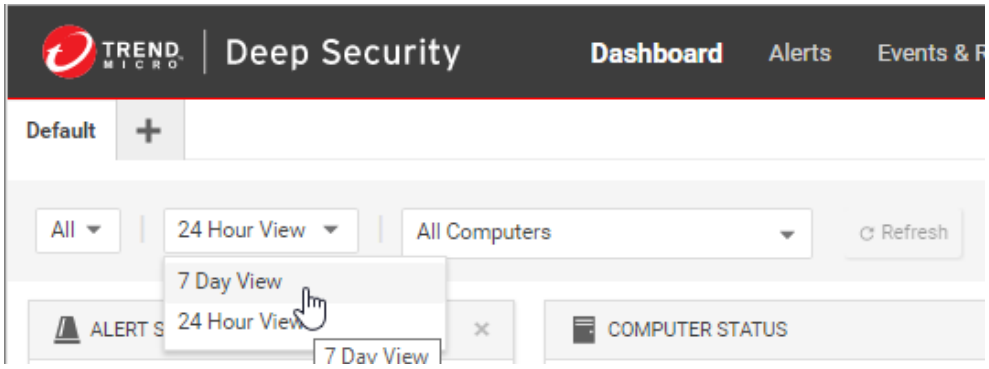


Figure 3.11: Filter the data by date and time range in Trend Micro

TrendMicro data can also be filtered by tags, which can be used to add additional attributes to the data. Figure below shows us how to add a tag to the data in Trend Micro:

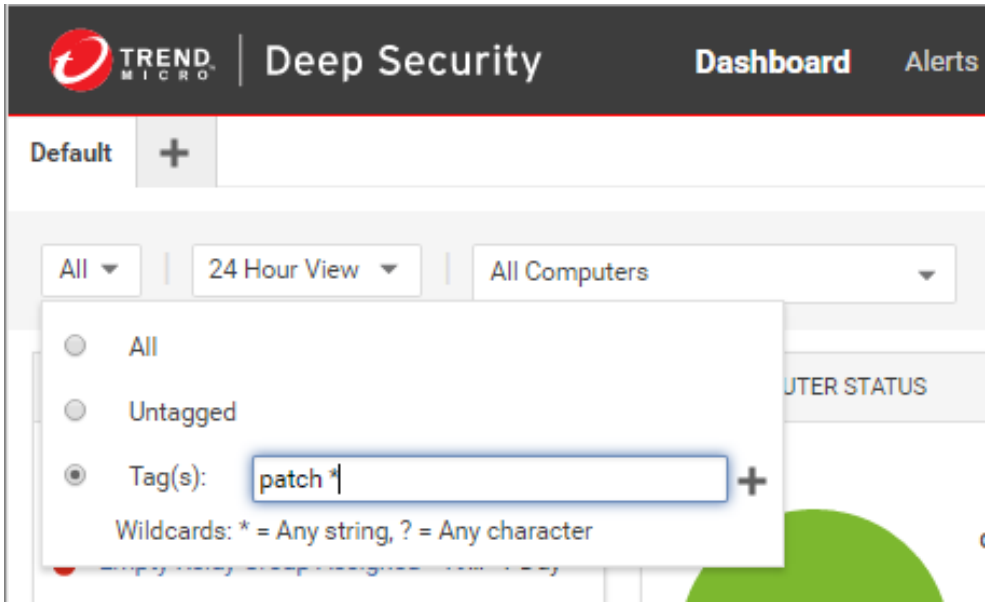


Figure 3.12: Add a tag to the data in Trend Micro

Heimdal® And Huntress

Other EDR solutions to compare are Heimdal and Huntress. The author is not going to get deep into details of the 2, as they have have similar features to the previous 2 solutions. Heimdal is a Danish cybersecurity company that provides a range of security solutions, including endpoint security, network security, and email security. Huntress, on the other hand an American cybersecurity company that provides EDR solution that is fully managed and monitored 24/7 by human SOC, who detect and investigate alerts, triage alerts, and provide actionable remediation steps or one-click solutions. On the homepage, information about each of the products/modules that are active under the current customer account is shown. The chart, which can be seen include a variety of Bar charts, Pie charts, and Line charts, include data regarding attacks, vulnerabilities, detections, infected/quarantined files, blocked/allowed processes, third-party application vulnerabilities or OS updates, and quarantined/rejected e-mails. Like the previous 2 solutions, they also have customizable dashboard that allows users to add, remove, and rearrange widgets to suit their needs. The dashboard also has filtering based on date time or other criteria. Furthermore, there are access roles within Heimdal (admin and customer) that allow the user to see only the data that is relevant to their role.

Figure below tells us about the look of Heimdal dashboard

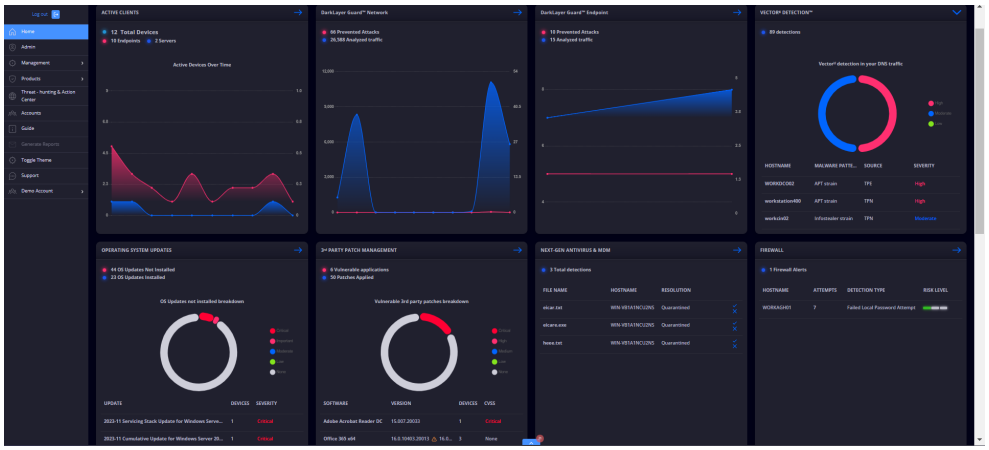


Figure 3.13: The dashboard of Heimdal®

Figure below tells us about how Huntress dashboard looks, which has more casual look that uses only base colours and avoids fancy designs

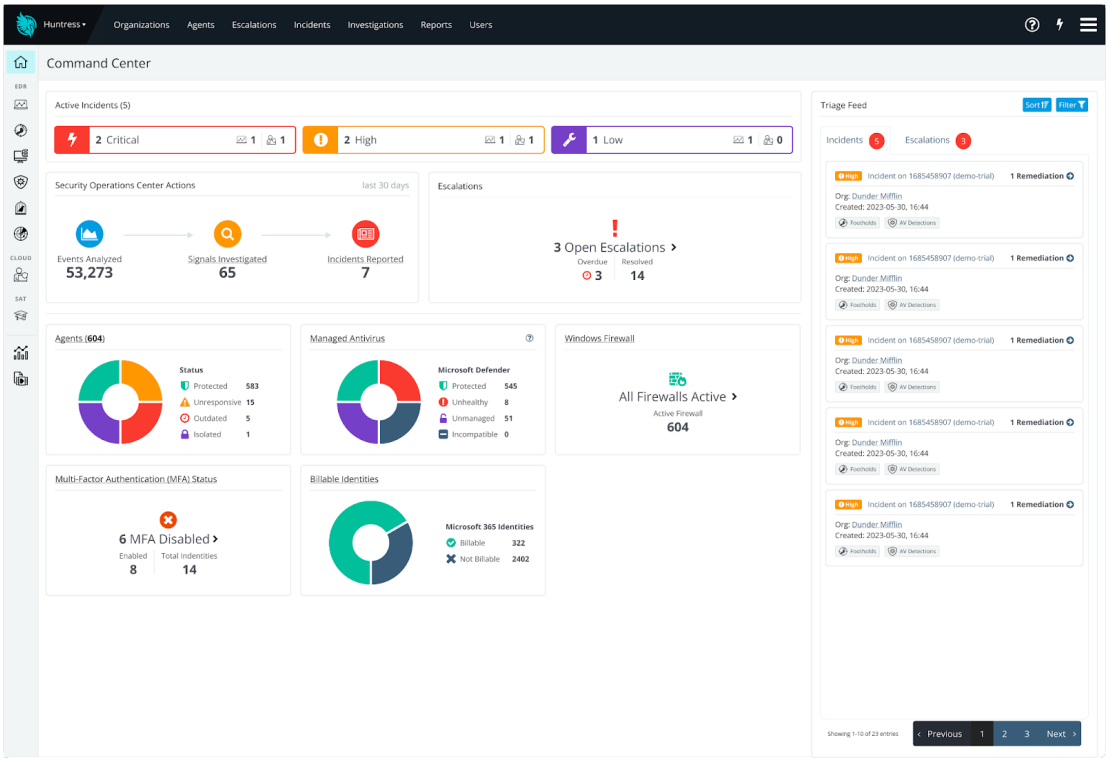


Figure 3.14: The dashboard of Huntress

Figure below tells us how Huntress display incidents to the users. Normally, a non-technical user did not to know all about the minute details of the incident, only seeing if there are active or severe threats or not in their devices should be sufficient.

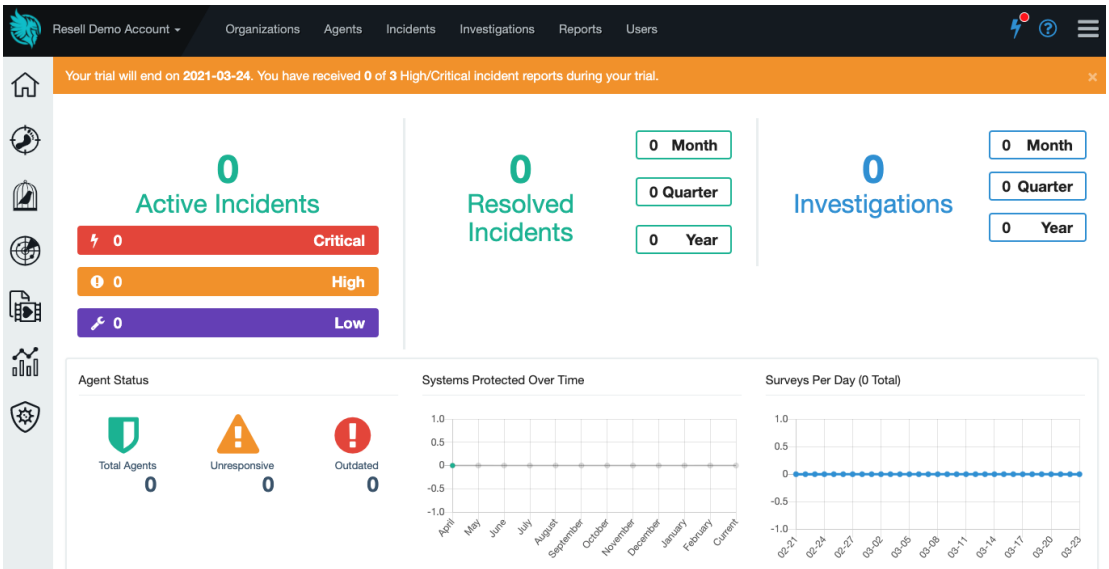


Figure 3.15: Seeing incidents in Huntress

Conclusion

Based on looking at other EDR security solutions, it is concluded that visualization of security data is crucial, as they all have the same functionality. Normally, the first thing users want to see about the security is that have their devices been infected or not. If yes, what infected the device and has the threat been mitigated yet. User customization options such as drag-and-drop and changing widget settings are essential for the to make the infomration accessible and understandable for all clients, regardless of their technical expertise.

3.5 Conclusion

Cybersecurity is a complex field that keeps on changing. Old cybersecurity solutions cannot be continued to be used and trusted against the defence of the latest more persistent and advanced malware. Every business, regardless of the size, is a potential target for cyberattacks. In small to medium business, cybersecurity is often overlooked or neglected. One potential breach by the attackers can cause significant damage to the business, often making it impossible for the business to recover. Despite operating with smaller budgets and fewer resources compared to larger enterprises, they remain attractive to threat actors in their way of seeking illegal financial gain.

The best EDR solution should help cybersecurity department, such as SOC analysts, with the ability to monitor and analyze occurred threats, generate report automation, visualizations, showing activity logs on the network, incident timelines, threat intelligence feeds, real-time threat alerts, and system health and performance metrics.

The integration of SentinelOne to the QaaS App helps the software to be one step closer into becoming an EDR system, in which for employees of Q-ICT, especially those in the cybersecurity department to use. This is essential as Q-ICT wishes to make the QaaS app as their centralized platform for monitoring clients devices and performing their day-to-day business operations. The dashboard module of SentinelOne in QaaS will be helpful especially to the cybersecurity department, so that they can provide better explanation in case of a threat happen on a client's console, in which the clients can also see the data for themselves, providing them with visual aid to avoid miscommunication and maintaining consistency as both parties can look at the same data to reduce misunderstandings.

For Q-ICT users, this integration will increase the value of the QaaS App as their customer insight management, in which Q-ICT as their trusted ICT provider, can show transparency to the clients, therefore showing them that their ICT infrastructure is managed in good hands. By providing comprehensive reporting coming from SentinelOne itself, Q-ICT ensure that the clients can be prepared to demonstrate their compliance to the government and tax authorities. This is not only fosters trust and accountability but also protects clients from negative publicity or consequences of non-compliance, security their operations and reputations.

The integration of SentinelOne to the QaaS App will help Q-ICT to control the data flow coming from SentinelOne. Not all technical users need to know the specifics about their endpoint security. By having this integration, Q-ICT can then change how much data they want to show to the customers based on their permissions. Q-ICT will also have more control in giving the functionalities of what SentinelOne can do, such as adding files into blocklist or whitelist, remediate and rollback a device operating system, and more.

Chapter 4

Realization

4.1 Methodology and Approach

The modulebook guideline for this graduation project allows the author to choose among different SLDC methodologies, such as SCRUM, the traditional Waterfall, or Kanban. Aligning with the company's current working methodology, SCRUM is chosen.

The working flow of the project is explained as follows: the author does a Sprint Planning meeting with the Company Supervisor, usually after the last week Retrospective, to discuss what features will be added to the project. In there the newest User Stories from the stakeholders will be assessed, and the Sprint Backlog will be updated. The author can then start a new iteration of a sprint, which usually lasted for 1-2 weeks. During a sprint, the author needs to work on a selected task in the Sprint Backlog, usually the highest priority and keeping update on the current status. A daily stand-up meeting is held to update other developers that are involved with the collaboration of the project. The Sprint Review, and Retrospective are held once every week, usually on Monday at 09:30. Here, the team will demonstrate the completed work to the Product Owner or stakeholders and gather feedback. The author then will discuss what went well and what didn't. A document regarding this should already be prepared 30 minutes to Company Supervisor and stakeholders before the Retrospective. The session also includes Code Review with other developers that will showcase the code created and discuss the more detailed part of the Sprint within the Development team.

An environment in Azure DevOps is set at the beginning of the project as the project management tool, which is used for visual aid and a planning tool. Tasks were created and moved around, as needed, and the columns were used to keep track of the progress of the project. The board also consist of list of bugs that were found and needed to be fixed immediately before the next release or planning of a next sprint.

All the project activities and in-depth explanation of how the SCRUM methodology is used are already described in Project Plan in the appendix of this paper (D).

4.2 Product

The product delivered by the author, is a fully operational dashboard module inside the QaaS App that has integrated with SentinelOne API. The dashboard is designed to simplify and display various kind of data coming from SentinelOne, and visualized them in widgets based on what kind of data they want to be visualized and the graph types. The widgets are customizable based on user preferences, and their latest session will be stored in the database, where the user can get their latest settings about the dashboard the next time they are logged in. Factors such as API rate limitation, renewing API tokens through scheduled cronjobs, making a secure connection by sending only user's Firebase Token ID (which will be a useless sequence of numbers if intercepted by malicious actors) and doing the calculation in the back-end itself to get the user and company ID are considered during the development of this project.

4.3 Development

This section will provide general explanation on the development of this project, mainly in the back-end and front-end. It will discuss the system architecture, tools and technologies used, and the data flow. The requirements analysis

and acceptance criteria are defined before to ensure

The overview about the components and entities of this project can be seen in the Class Diagram is also located in there (A) as well. The more detailed explanation about how the system and database infrastructure are designed can be found in the Technical Design at the appendices of this project (B).

4.3.1 The Back-end

Firebase Cloud Functions and Cloud Firestore were the main technologies that built the infrastructure of the back-end of this project. The types of functions that are used throughout the project are the following:

- `onCall`: This type of function is used to handle callable functions, which are called from the client-side. It is used to handle requests from the client-side. This is the main function in which the SentinelOne data is fetched and processed.
- `onRequest`: This type of function is used to handle HTTP requests, which are called from the client-side. It is mainly for initial testing, as setting this function up is easier and faster. However, the author does not recommend using this in the production, as it lacks authentication and security for the end-user.
- `onSchedule`: This type of function is used to handle scheduled functions, normally called cron-jobs, which are called at a specific time. It is used to handle scheduled tasks that need to be executed at a specific time. There is only one scheduled function in the project, which is used to replace the old SentinelOne API key with a new one that is securely stored in Google Secret Manager, therefore ensuring proper connection and access to the secret vault, every month.
- `Cloud Firestore Triggers`: this function is used for the Algolia extension, which is used to listen to changes in the Firestore database and update the Algolia index accordingly. This function is used to keep the Algolia index up-to-date with the Firestore database, ensuring that the search functionality works correctly.

Setting up permissions with Google Secret Manager

Because every call to the SentinelOne API requires an authorization through a valid SentinelOne API key, it is crucial to store this key securely on the Internet, where not everyone can access it. Following the guideline from the Company Supervisor, in whom is not a big fan of storing sensitive information in .env files or directly in the code, the author stored the SentinelOne API key in a Secret within Google Secret Manager. The number one reason as to why is because Firebase Cloud Functions work well with Secret Manager, as they are both Google Cloud products. Therefore, the author can easily edit the settings of a specific cloud function to have access to the latest version of a specific Secret, providing that they are stored within the same project directory.

Service Account

The first crucial step in setting up the permissions with Google Secret Manager is to create a Service Account. A Service Account is a special type of Google account that belongs to the application or a virtual machine, instead of to an individual end-user. It is used to operate and manage services without sharing user credentials (*Firebase, n.d.-d*). Whenever a function is created in Firebase, it is automatically assigned a Service Account, the admin can edit the permissions of the Service Account to have access to the Secrets stored in the Secret Manager. With the proper access level granted to a specific Service Account assigned to a desired function, the author can then easily configure on which Secrets his cloud functions can have access to.

Firestore database structure

Inside the Firestore, there are various collections (tables) that hold different data. Most of the collections are used to store the data that is fetched from the SentinelOne API, and one collection is used to store the user's session and preference data. SentinelOne data itself is divided into 2 categories: general SentinelOne data and specific data, both can have varying fields depending on how SentinelOne structures the data in its JSON format. General SentinelOne data only has 1 specific field in common, which is the siteId. The idea is to store all the clients' data in one collection, therefore differentiating the data based on client's company, an index of client's site ID (which is unique from each other) is used to separate the data.

The second category is the specific SentinelOne data, win which the data does not only belong to a specific site ID, but also to a specific ID from another SentinelOne data. For example, each threat can have their own unique timeline of events that have happened. Therefore, another field is required that is used as a second index to reference the desired data. The first and second category will then have different logic in the cloud function to fetch the data.

Modelling in the back-end

The JSON data is modelled in the back-end first, then in the front-end, to ensure that the data is consistent across the application. The data is modelled using interfaces and classes, which are defined in the Models directory. The interfaces define the structure of the data, and the classes provide methods to serialize and deserialize the data. This ensures that the data can be easily converted to and from JSON.

Setting up permissions with Firestore

This has got to do with the Firestore database and the access level of permission that a Firebase Cloud Function has. In the SentinelOne integration project, the table that contains SentinelOne data cannot be changed by the client, only by fetching new data from the SentinelOne API and replacing the old data. Therefore, only one table remains that can be changed accordingly by the user, the User Preference table. This table can be changed freely by the user if they remain logged in to the QaaS app.

Connection to Algolia

As originally stated, the author and Company Supervisor directly intended on using Algolia search as the SentinelOne API search, although far from being bad, does have type intolerance. The purpose of this is to enhance the UX because not all users can remember the exact name of the data they want to search for and makes the search more accurate and faster even with false inputs.

Q-ICT itself already has the test environment project set up in Algolia, because they are also using it in every search functionality that they have in the QaaS app. For this specific project, the author's work Q-ICT Microsoft e-mail account was invited to join the project. In there, the author can already see all the duplicate indexes coming from Firestore that serves as the point of reverence when letting the AI Algolia engine to do the searching. From Algolia dashboard, several default API keys can be seen, including:

- Application ID: unique application identifier
- Search API key: usable for search queries and for listing the indices.
- Write API key: is used to create, update, and delete indices. Cannot be used to manage other API keys.
- Admin API key: like write keys but can be used to manage other API keys.

The way the extension itself works is that it acts as a pre-written cloud function like Cloud Firestore Triggers (*Firebase, n.d.-e*), where it listens to the collection in a Firestore. When the data changes this cloud function passes it through Algolia. It is a script Firebase made together with Algolia. On this extension, 2 parameters are needed to be included: the Application ID and the API key. The Application ID is used to point to the right project in Algolia. The API key is used to authenticate the connection between Algolia and Firebase. For every Algolia extension, it is recommended to create a separate API key and not to use the admin key, so that the key can be revoked if needed. The author is using the custom API key that was made by the Company Supervisor, that has restriction on the index that it can access, and the operations that it can do.

4.3.2 The Front-end

In the front-end side of this project, it is just a matter of calling the Cloud Functions, modelling the JSON response coming from the Cloud Functions (whether it is from Firestore or SentinelOne), making the visualization graphs using the available Flutter package, setting the logic on how to navigate between pages passing the correct data, the logic for how to display the data correctly in paginated data tables and visualization widgets, and lastly, the searching, filtering, and filtering functionality.

After ensuring that the logic of the front-end works perfectly for showing the data, the last step that was needed to be undertaken was polishing, which includes: checking for bugs, translation error, and making sure that the malfunctioning part that were not yet fixed due to time constraint are not shown and testing for responsiveness on different devices. The last part was proven to be quite a long process, because almost every widget that the author has created needed to be wrapped around another Flexible widget that can expand and shrink based on the screen size, and to make sure that there is no infinity width or height that can cause the widget to be cut off or not shown at all.

4.4 Deploying everything to the Live environment

In the final phase of the project, after making sure that everything works and tested properly, the author needed to deploy the project from the test environment to the live environment, so that other users can also access it.

The full guide on how to deploy the application from the test environment to the live environment is available in the appendix C.

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

Cybersecurity is a complex field and threat actors are ever evolving their trade-craft in clever ways. Every single running business is not too small to be attacked, but they may be too small for them to make the news. Therefore, transparency to the clients is important, because lack of visibility to the clients can lead to a loss of trust. This graduation project serves as the first iteration by Q-ICT about the integration of SentinelOne to its internal application, the QaaS app.

The purpose of this project is to research into SentinelOne and its API capabilities, and develop a dashboard that serves as one of the module for the QaaS app. It keeps a transparent relation with customers by informing them about the aspect of their IT system, especially about the security status. The project has 2 main phases, the research and realization parts. The research was aimed at studying the technologies behind the QaaS App, SentinelOne API functionalities, and visualization techniques to display the data to the non-technical users. The realization was aimed at displaying the data in the QaaS App with functionalities such as widget visualization based on user preference, Algolia extension integration, API rate limiting, data filtering and pagination.

All the objectives set and agreed in the beginning by the stakeholders and the author were met, and the project can be considered successful. The author has completed the research, as this document proves, the realization phase has been successfully developed, and all mandatory and agreed-upon documents has been sent to Q-ICT and the educational institution, NHL Stenden.

The research and realization has provided valuable insights into Q-ICT about the kind of data that SentinelOne provides. The author has gained deeper understanding into the field of cybersecurity, especially about SentinelOne, and furthermore familiarity with Flutter and Firebase. The research has answered the main question with positive answer, that the integration of SentinelOne into the QaaS app is possible.

The new SentinelOne module to the QaaS App has no breaking changes, only adding more data and features to the database and the app. In the end, the product were brought to the live channel where it was introduced to the live data from 500 companies using the API token with higher authority. It was then monitored closely by the development team and bugs were tracked and noted. Tools like Azure DevOps as the Agile project provider, and Git as the version control. The new module has been tested with in Preview channel on the live environment, with the data from 500 companies and a higher permission in the API token to be able to see the data.

In the back-end, several functions with different functionalities need to be created to handle the data that is fetched from different SentinelOne API endpoints. The user's session and preference data are stored in the Firebase Firestore DB, and as well as AI powered Algolia search engine to make the search faster and more accurate. Additional features such as filtering, sorting, and pagination will be handled by inputting the correct headers to the API request.

5.2 Recommendation

5.2.1 Integration with N-Central data

In the research phase of this project, the Company Supervisor mentioned that there are some issues regarding the data coming from N-Central API (which also one of the API used for security) that sometimes can cause data discrepancy.

The author has looked into said N-Central data and noticed some similarities with some SentinelOne data structure. The idea of this recommendation is to combine the data from SentinelOne and N-Central that have similar structure, to then make the data to be more accurate and reliable. This will serve as further development for the continuation of this project.

For example, N-Central has one API that can automatically check for an update of an application of a device, given the correct application ID and device ID. If a device has an outdated application, it can either update the application automatically, or notify the corresponding user. If the device also has a SentinelOne Agent installed, the Agent can then detect all the applications that are installed and know immediately if there is an outdated application, with checking for the outdated application that may pose a security risk according to the fastest CVE database. Combining two of these functionalities would then make sense, as SentinelOne can then prioritize outdated applications on a device, and N-Central can then notify the user to update the said application. This way, the user can then have a more secure device, and the QaaS app can then provide a more comprehensive security solution to the user.

5.2.2 Redis middleware NoSQL database caching

Redis is a powerful tool for DB that uses in-memory data structure to provide caching and message broker. It supports various data structures (*Redis, n.d.*), such as strings, hashes, lists, sorted sets, with range queries, bitmaps, hyperlogs, geospatial indexes with radius queries, and streams.

Because the main idea in Redis is to provide caching of the data, coupled with the fact that it is a NoSQL DB, Redis is exceptionally fast to query due to its in-memory nature in contrast to traditional disk-based DBs. This makes it suitable for use cases where low latency and high throughput are required, making it perfect for getting and storing data that is frequently accessed.

The idea is to put SentinelOne data into the Redis DB too, with the limit of 5-10 minutes before the cache is deleted, and then fetch the data from the Redis DB instead of directly fetching it from Firestore and SentinelOne API constantly. Implementing Redis will make querying the data faster than Firestore, providing better UX for the users. Below is how the connection infrastructure would look if Redis is implemented:

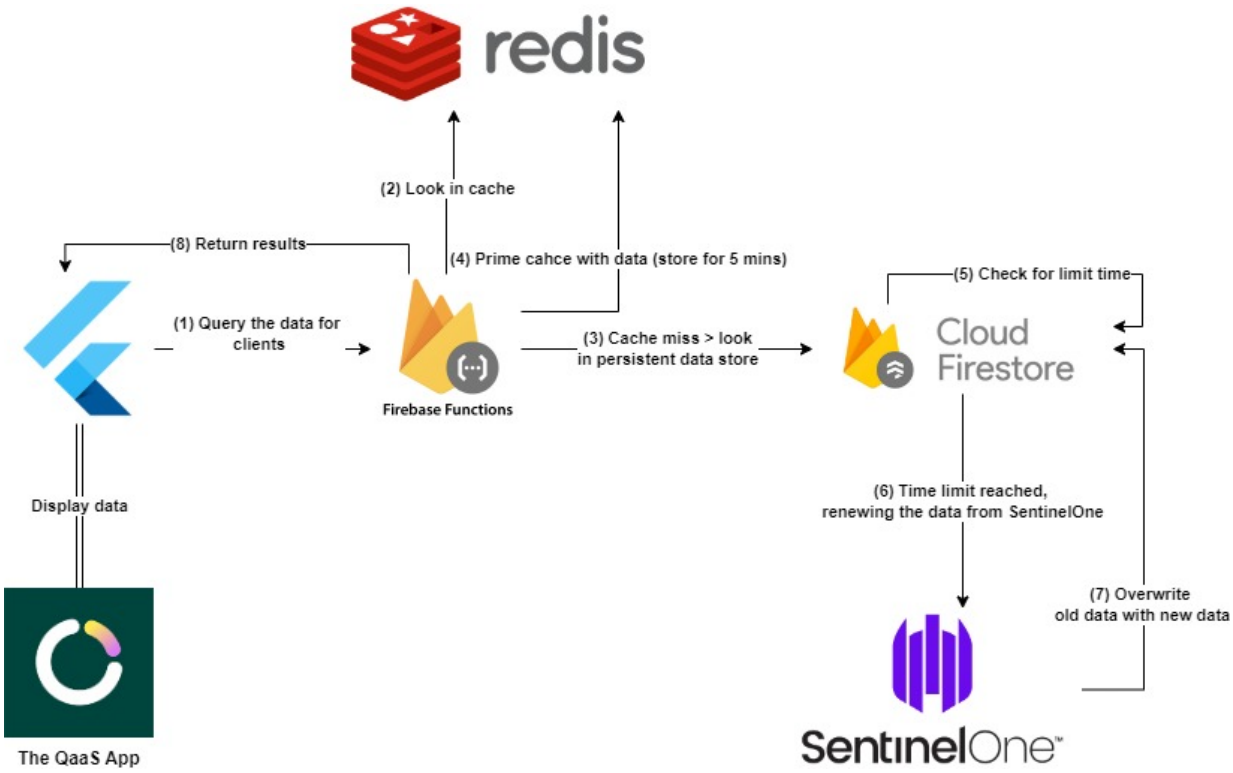


Figure 5.1: How the infrastructure would look if Redis is implemented

However, there is one potential downside that the author sees when using Redis in this project. According to these articles (*Trollope, 2024*, and *Shams and Valderrama, 2024*), as of March 2024, Redis is no longer open source. They

changed the official licence for all future versions. This has serious consequences for the open-source community and some companies providing managed Redis. Therefore, the initial development of the Redis middleware to this project can no longer use the free version of Redis from a Redis provider without a direct partnership with Redis. Nonetheless, with a direct paid subscription to Redis, the author believes that in the long run, using Redis as a means of caching the data will be beneficial for the QaaS app, especially when the number of customers have grown and the SentinelOne data gets called more frequently. Figure 5.2 describes the Redis providers that can no longer provide Redis server to their service without direct partnership with Redis.

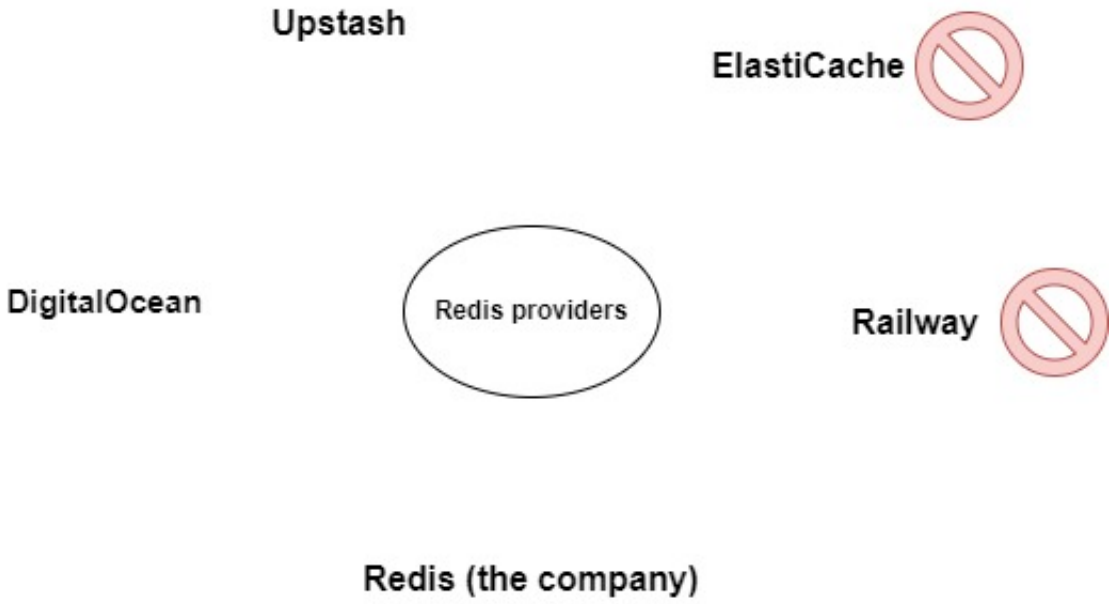


Figure 5.2: Redis providers now cannot distribute Redis any more without a direct partnership with Redis, therefore Redis itself controls the pricing.

5.2.3 Additional requirements gathering

Additional insights such as market requirements should be gathered to identify other desired features and functionality. This ensures the dashboard meets user needs and maintaining trust relationship between the clients and Q-ICT.

5.2.4 User pilot testing

Because the quality assurance testers of this project has been the employees within Q-ICT so far, the author recommends the company further conduct user testing with their actual customers, to gather feedback and identify any missing features or unknown bugs. A representative group of users should be selected and interviewed upon based on their user experience to refine the end-product.

5.2.5 Training and User Manual

While the purpose of this project is to promote user-friendliness while showing technical data, Q-ICT should consider proper user manual creation to briefly explain the purpose of this next version of the QaaS App, ensuring clear insights for the audience on what to expect.

5.2.6 SentinelOne Aggressive Mode

SentinelOne has a lot of features that can be beneficial for an organization, such as its ability to detect and respond to threats in real-time. This, however, does not come without its own potential downsides. SentinelOne Agents often exhibit aggressive behaviour, flagging benign and legitimate files or processes as threats and take actions such as quarantining or mitigating them. This makes them susceptible to false positives, especially in their default configuration. This can lead to unnecessary disruptions, loss of productivity, and distrust in the security protocol. Organizations may need to invest some time and resources in investigating and resolving false positives alerts. The

author therefore would suggest Q-ICT to treat it very carefully and to make sure that the SentinelOne Agents are configured properly to avoid unnecessary disruptions.

Bibliography

Acronis. (n.d.). Computicate. *solutions.acronis.com*. https://solutions.acronis.com/en-us/integrations/computicate/

Algolia. (n.d.). https://www.algolia.com/

AWS. (n.d.). *What is XML?* https://aws.amazon.com/what-is/xml/

Cloudfare. (n.d.). *What is penetration testing? | What is pen testing?* https://www.cloudflare.com/learning/security/glossary/what-is-penetration-testing/

Cronitor. (n.d.). *Crontab Guru*. https://crontab.guru/#0_0_*/7_*_*

Fielding, R. T., & Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*. https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

Firebase. (n.d.-a). *Firebase Products - Google*. https://firebase.google.com/products-build

Firebase. (n.d.-b). *Firebase Extensions*. https://firebase.google.com/docs/extensions

Firebase. (n.d.-c). *Firebase App Check*. https://firebase.google.com/docs/app-check

Firebase. (n.d.-d). *Firebase Service Accounts Overview*. https://firebase.google.com/support/guides/service-accounts

Firebase. (n.d.-e). *Cloud Firestore triggers*. https://firebase.google.com/docs/functions/firestore-events?gen=2nd

Firebase. (n.d.-f). Schedule functions. *Firebase.google.com*. https://firebase.google.com/docs/functions/schedule-functions?gen=2nd

Gartner. (n.d.). *EDR (Endpoint Detection and Response) Solutions Reviews and Ratings*. https://www.gartner.com/reviews/market/endpoint-detection-and-response-solutions

Google. (n.d.-a). *What is CAPTCHA?* https://support.google.com/a/answer/1217728?hl=en

Google. (n.d.-b). *What is reCAPTCHA?* https://www.google.com/recaptcha/about/

Google. (n.d.-c). *Secret Manager*. https://cloud.google.com/security/products/secret-manager

IBM. (n.d.-a). *What is an API (application programming interface)?* https://www.ibm.com/topics/api

IBM. (n.d.-b). *What is a REST API?* https://www.ibm.com/topics/rest-apis#:~:text=A%20REST%20API%20(also%20called,transfer%20(REST)%20architectural%20style).

Indeed. (2023). *What Is SOAP API? (Plus Comparison to REST API and Benefits)*. https://www.indeed.com/career-advice/career-development/what-is-soap-api#:~:text=SOAP%20API%2C%20or%20simple%20object,Extensible%20Markup%20Language%20(XML).

JSON. (n.d.). *Introducing JSON*. https://www.json.org/json-en.html

Kubernetes. (2018). *CronJob*. https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/#:~:text=A%20CronJob%20creates%20Jobs%20on,schedule%2C%20written%20in%20Cron%20format.

Kubernetes. (2024). *CronJob*. https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

LinkedIn. (n.d.). *Resello: powered by Pax8*. https://www.linkedin.com/company/resello-bv/

MongoDB. (n.d.). *What is NoSQL?* https://www.mongodb.com/nosql-explained

Overheid, D. (n.d.). *NIS2-richtlijn*. https://www.digitaleoverheid.nl/overzicht-van-alle-onderwerpen/nis2-richtlijn/

PerfectView. (n.d.). *PefectView CRM Online | Krachtige Nederlandse CRM*. https://www.perfectviewcrm.nl/

Redis. (n.d.). *Data structures*. https://redis.io/redis-enterprise/data-structures/

Salesforce. (n.d.). *What is CRM (Customer Relationship Management)?* https://www.salesforce.com/eu/learning-centre/crm/what-is-crm/#:~:text=Customer%20relationship%20management%20(CRM)%20is,relationships%20to%20grow%20your%20business.

SentinelOne. (n.d.-a). *What is Threat Hunting? A Cybersecurity Guide 101*. https://www.sentinelone.com/cybersecurity-101/threat-hunting/

SentinelOne. (n.d.-b). *SentinelOne Ranger*. https://nl.sentinelone.com/resources/sentinelone-ranger/

Shams, K., & Valderrama, T. (2024). *RIP Redis: How Garantia Data pulled off the biggest heist in open source history*. https://www.gomomento.com/blog/rip-redis-how-garantia-data-pulled-off-the-biggest-heist-in-open-source-history

Taylor, C. (2021). *Castle-and-Moat Network Security Model*. https://cyberhoot.com/cybrary/castle-and-moat-network-model/

TechTarget. (n.d.). Techtarget it channel. *RMM software (remote monitoring and management software)*. https://www.techtarget.com/searchitchannel/definition/RMM-software-remote-monitoring-and-management-software

TOMTelecom. (n.d.). *Wie zijn wij*. https://www.tomtelecom.nl/wiezijnwij/

Trollope, R. (2024). *Redis Adopts Dual Source-Available Licensing*. https://redis.io/blog/redis-adopts-dual-source-available-licensing/

Veracode. (n.d.). *ARP Spoofing*. https://www.veracode.com/security/arp-spoofing

Vogel, J. (2023). Ict research methods — methods pack for research in ict. *HBO-i, Amsterdam*. https://ictresearchmethods.nl/

Wikipedia. (n.d.). *Man-in-the-middle attack*. https://en.wikipedia.org/wiki/Man-in-the-middle_attack

Appendix A

FD (Functional Design)

Functional Design

Christopher Sulistiyo (4850025)

Christopher.sulistiyo@student.nhlstenden.com

ICT & IC Information Technology Department Emmen

Client

Quality ICT B.V.

Version 1.1 – 21/02/2024

VERSION CONTROL **1**

REMARKS **1**

INTRODUCTION **2**

BACKGROUND AND CONTEXT 2
PROBLEM STATEMENT 2
OBJECTIVES 2

CHAPTER 2 – DESCRIPTION OF THE INFORMATION SYSTEM **3**

SYSTEM OVERVIEW 3
FUNCTIONAL REQUIREMENTS 3
TECHNICAL REQUIREMENTS 4
BUSINESS REQUIREMENTS ..... **ERROR! BOOKMARK NOT DEFINED.**
USER STORIES 5
High priority 6
Low priority 7
USES CASES 8
ACCEPTANCE CRITERIA 11
MOSCOW ANALYSIS 13

CHAPTER 3 DATA MODEL **15**

DESIGN PATTERNS USED. 19
USER INTERFACE AND SYSTEM DESIGN 25

CHAPTER 4 – OUTPUT **32**

CHAPTER 5 REQUIRED INPUT **38**

CHAPTER 6 MENU STRUCTURE AND AUTHORIZATION **42**

CHAPTER 7 TECHNICAL CONSEQUENCES **44**

Version control

Version	Activities	Date
Initial version 1.0	Draft version	06-February-2024
Version 2.0	Added Class Diagram, Functional and Technical Requirements	10-June-2024

Remarks

Any changes and new developments that have a significant impact on the project proceedings will be noted here.

Introduction

Background and Context

This document is meant to provide an overview of the systems and data flow of the new features regarding SentinelOne EDR API integration to the QaaS app project. The project is carried out as a graduation project to Christopher Sulistiyo, a 4th year ICT student in NHL Stenden as a necessary requirement to achieve his diploma.

This project serves as the first proof of concept, making it the first attempt in the iteration.

Problem Statement

The company currently just purchased SentinelOne subscription alongside its Vigilance package and want to implement it to the QaaS app. Q-ICT currently already has 5 APIs to manage their customer's device health, finance, and Microsoft subscription, which namely are: Bodyguard, N-Central, SnelStart, PerfectView, and Pax8. The goal of integrating SentinelOne is to show more transparency to their customers, so that they can also know the information about their servers, devices, and computers.

Objectives

The goal of this project is to create a new functionality on top of an existing web application that can display all the necessary data coming from SentinelOne APIs. The data is then be represented in graphs, charts, or table. The graphs and charts themselves are customizable, whereas the table can sort, filter, and search the data based on user input. The application will also make use of stored session based on user's preferences on each user and store the session on the database (Firestore) securely with scalability in mind.

The integration to the QaaS app should also synchronize with the other 5 API data that in the system, whether they are stored in the company's Firestore database or not.

Additionally, the company also wishes to express the SentinelOne data to also collaborate with N-Central API data, in which it is an RMM (Remote Monitoring Management) application related to cybersecurity too, to manage clients' IT infrastructure, including servers, workstations, mobile devices, and network devices.

Chapter 2 – Description of the Information System

The project description is explained better in the Thesis of the author's graduation work placement project. For better information, please refer to the Chapter 1 of the Thesis.

System Overview

The product in question, the QaaS app, is an ERP-like web application used by Q-ICT and its clients. An ERP (Enterprise Resource Planning) application is a software platform that integrates the core of business processes and functions into a single system to streamline operations and improve efficiency. It facilitates cross-functional collaboration by centralizing data and automating workflows across different departments. The QaaS app enables Q-ICT to make data-driven decisions, improve productivity, reduce costs, and enhance customer satisfaction.

Functional Requirements

Functional Requirements describe what the system should do. They detail the specific behaviours and functions of the system. These requirements outline the system's functionality from a high-level perspective and are often derived from business requirements. The functional requirement of this project is as follows:

1. **Device Health and Status Information Display**
    a. The system must display the health status of all devices monitored by SentinelOne
    b. Users should be able to view detailed status for information for each device
    c. Users should not see any data not relevant to their company (Site ID)
2. **Threat Audit Trail**
    a. The system must provide an audit trail of all threats detected on the monitored devices
    b. Users should be able to filter and search through the audit trail based on various criteria (e.g., device, date, threat type)
3. **Integration with SentinelOne API**
    a. The system must integrate with SentinelOne API to retrieve real-time data about device health and threats
4. **User-friendly Interface**

a. The front-end must be intuitive and easy to navigate for users with varying levels of IT and cybersecurity knowledge

Technical Requirements

1. **Technology Stack**
    a. Front-end: Flutter framework
    b. Back-end: Node.js with TypeScript as a template and ES6 linting
    c. Cloud Solution: Firebase (including Firebase Cloud Functions, Firestore, and Google Secret Manager)
2. **Data Security and Privacy**
    a. Ensure all HTTP request between the QaaS, the Cloud Functions, and SentinelOne are encrypted
    b. Implement appropriate security measures to protect user data and privacy
3. **Integration with SentinelOne API**
    a. The system must be able to make secure API calls to SentinelOne to fetch data
    b. Data retrieved from SentinelOne must be processed and stored in a way that aligns with the QaaS App's data structure
4. **Scalability**
    a. The system and the cloud functions should be able to handle data from over than 400 clients without performance degradation keeping in mind that one user can make up to ±40-50 requests per day
    b. The database should be able to hold data from over than 400 clients regarding the devices, malwares, and applications that they have in their endpoints
5. **Error Handling and Logging**
    a. The system must include comprehensive error handling and logging to troubleshoot and resolve issues effectively
6. **Real-time Data Processing**
    a. Implement real-time data processing capabilities to ensure up-to-date device status and threat information
7. **Compliance**
    a. Ensure the system complies with relevant industry standards and regulations for cybersecurity and data protection

8. **Documentation**
    a. Provide thorough documentation for the codebase, API integrations, and user guidelines to ensure maintainability and usability
9. **Backup and Recovery**
    a. Implement backup strategies for critical data and configurations related to SentinelOne integration. Develop recovery procedures to quickly restore services in case of failures.
10.

User Stories

User stories are short, simple description of a feature told from the perspective of the end-user. They usually follow a format and are used in agile development to capture in a lightweight, user-centric manner. The format of user stories is usually as follows:

"As a [user role], I want [goal] so that [reason]".

The user stories of this project will be divided into 4 categories based on the importance, which will be:

- High priority
- Low priority

Furthermore, it will also have sub-categories which relates to which 5 perspectives of the user does it come from. The perspectives are as follow:

- **The clients** are the end-users of the QaaS app coming from customers from different companies. They are the businesses or individuals who have entrusted Q-ICT to monitor the security of their devices. They have quite important needs to the user stories as they need to access information about their device health and status and require transparency and assurance about the security measures in place. Their user stories will focus on features that allow clients to easily understand and monitor their devices' security status without needing in-depth IT knowledge.
- **The helpdesk** is part of Q-ICT's employees to provide support to clients. They trouble shoot issues and offer guidance when there is an issue with the clients regarding their Microsoft subscription or products. They require access to detailed device and threat

information to assists clients effectively. Their user stories will focus on features that enable helpdesk to support clients promptly and accurately.

- **The cybersecurity analysts** are part of Q-ICT's employees that are responsible for monitoring and analysing security threats. They ensure that the devices are protected and respond to incidents. They need to make sure that this integration does not show too much of technical data to the customers as to not overwhelm them, also to make sure that no sensitive data are being shown that would provide risks to their own safety such as devices IP addresses. Analysts need detailed and real0tie data about threats and device health. Their user stories will focus on features that support threat analysis and response.
- **The IT admin (developers)** manage and maintain the QaaS App. They are responsible for integrating the SentinelOne
- **The company (Q-ICT)** contain the stakeholders and project managers that oversee the development of the project.

High priority

For Clients:

- As a customer, I want to be able to see the overview status of all my devices in my network so that I know which endpoints are infected with malware and which devices are healthy.
- As a customer, I want to be able to see the audit trail and the timeline of every threat detection and mitigation.
- As a customer, I want to be able to see detections and mitigations within a specific period (week/ month) per device.
- As a customer, I want to be able to customize the widgets responsible for visualizing the data shown from SentinelOne (which types of graphs or charts I would like to select, with adding and deleting the visualization widgets), so that I can have more control over what I see on the screen to help me understand the context more with my representation preferences.
- As a user I want to be able to have filtering, sorting, and searching functionalities for the SentinelOne data shown in tables so that I can choose which data to be shown, so that I can understand the data more.

For Helpdesk

- As a helpdesk employee, I want the system to display more detailed comprehensive technical information about our clients' devices, so that I can facilitate effective and efficient monitoring to their health status and provide timely support. I should be able to see the health status of all the devices of the customers.
- As a helpdesk employee, I want the app to be able to access comprehensive threat response guidance in real-time from SentinelOne whenever a security threat is detected, so that I can promptly and effectively assist our clients.

For IT admins:

- As an IT admin, I want to have full control of my QaaS web application, by having the same read permission as the Helpdesk and being able to do POST, PUT, and PATCH requests.

For Q-ICT:

- As a cybersecurity consultant company, I want to be able to show more transparency to my customers by showing them the data that SentinelOne gives regarding their endpoint security
- As a cybersecurity company, I would like to be able to have more control over the flow of the data and how much information is shown to the customers.
- As a cybersecurity company, I would like to be able to control the functionalities that the user can do in SentinelOne through the QaaS App

Low priority

For Helpdesk

- As a helpdesk employee, I want a system to automatically send an e-mail to me in case of a high severity unsolved cyber threat happens to one of our clients, so that I can be notified.
- As a helpdesk employee, I want the system to be able to show me what steps do I need to take in case of a cybersecurity threat happening so that I can assists the distraught client better.

For Q-ICT:

- As a cybersecurity consultant company, I want to leverage SentinelOne Vigilance package to be able to offer my customers a 24/7 real-time proactive threat mitigation services and checks that is included for showing real-time metrics and mitigations that

are available so that we can effectively safeguard their digital assets and mitigate cybersecurity risks.

Uses Cases

Use cases describes how users will interact with the system to achieve specific goals. They are detailed scenarios that map out the step-by-step interactions between a user and the system. Use cases focus on the user's perspective and often include multiple paths, include alternate and exception flows.

- **Device Health Monitoring**
  - Actor: clients
  - Use case: the QaaS app displays real-time health information about each device (e.g., laptops, workstations, servers) monitored by SentinelOne agents installed in those devices. This includes metrics such as installed applications, CPU core count, CPU version, RAM memory space, status of the health, is active or not, last active date, etc,
  - Precondition:
    - Client is logged into the QaaS app.
    - The client has devices installed and monitored by SentinelOne Agents EDR.
  - Main flow:
    - The client navigates to the SentinelOne dashboard page section of the QaaS app
    - The system retrieves the latest device health and status information from the SentinelOne EDR.
    - The system displaye device health and status in a user-friendly format.
  - Implementation: retrieve device health data from SentinelOne API and display it in a user-friendly format within the QaaS app dashboard or device management section.
  - Postcondition:
    - The client can view the health and status of their devices.
  - Alternative flow:
    - If the system cannot retrieve the device health and status information, if the data is null, if an error occurred on the server or on the client, it

should handle the error and display the error message appropriately to the client (in a user-friendly way, without displaying too much technicality).

- **Threat Detection and Alerting**
  - Actor: the client, the IT administrators.
  - Preconditions:
    - The client is logged in into the QaaS app
    - The client has devices installed with and monitored by SentinelOne Agents EDR
  - Main flow:
    - The system monitors the devices for any changes in health or status.
    - If a significant change is detected (e.g., a device's health status changes from "Healthy" to "Unhealthy" or "Infected", the system sends an alert to the IT admin and the infected client)
  - Use case: SenntinelOne EDR platform detects a security threat (e.g., malware, ransomware, viruses, worms, spyware, etc.,) on a client's device.
  - Implementation: The threat will immediately be removed from the endpoint by SentinelOne and be listed and displayed in the dashboard as the detected threats that have happened for the Helpdesk and Security Experts of Q-ICT to analyse and document later. Optionally, a notification in the form of e-mail or from the QaaS should be delivered to the concerned user.
  - Postcondition:
    - The client is notified of any significant changes in their device's health or status.
  - Alternative flows:
    - The client can choose to silent or opt-out from receiving the alert notification.
- **Audit Trail and Incident Response**
  - Actor: the client
  - Precondition:
    - The client is logged in into the QaaS app
    - The client has devices monitored by SentinelOne Agents EDR

- Main flow:
    - The client navigates to the Threats section of the QaaS app
    - The system retrieves the audit trail of threats from SentinelOne API
    - The system displays the audit trail in chronological order, with details of each threat.
- Use case: a security incident occurs on a client's device, such as a successful malware infection or unauthorized access attempt.
- Implementation: Retrieve audit trail data from SentinelOne via the API to track the sequence of events leading up to the incident. Present this information in a chronological order within the QaaS app, allowing administrators to investigate and respond to security effectively.
- Postconditions:
    - The client can view the audit trail of threats that have occurred on their devices
- Alternative flows:
    - If the system cannot retrieve the audit trail, it displays an error message in a user-friendly way.

- **Policy Management and Compliance**
    - Actor: the company
    - Preconditions:
        - The company has subscription to the QaaS app and has its own environment registered along with its clients.
        - The company has their SentinelOne agents installed to the desired endpoints.
        - The company has a certified cybersecurity expert to monitor those devices in case of a threat happens.
    - Use case: Q-ICT and its clients need to enforce security policies and ensure compliance with international industry regulations and standards regarding security (ISO 27001).
    - Implementation: With integrating real-time compliance-related data from SentineOne, such as security posture assessments and compliance reports, it

will help Q-ICT and its clients demonstrate adherence to regulatory requirement.

- Postcondition:
  - Q-ICT and its clients now have more protection on their IT systems from an EDR and XDR, which are an upgrade from traditional AV.
- Alternative flows:
  - The clients of Q-ICT can choose to unsubscribe from SentinelOne through Q-ICT if they want to take a different EDR platform.

- **User-friendly Reporting**
  - Actor: the client, IT admin
  - Preconditions:
    - The client is registered by Q-ICT with SentinelOne subscription.
    - The client does not have to be logged in to the QaaS app.
  - Use case: clients want to view comprehensive reports on their device security posture and threat landscape.
  - Implementation: generate customizable reports within the QaaS app using data obtained from SentinelOne. These reports can include metrics such as threat detection rates, incident response times, and overall devices security scores. Ensure that the reports are easy to understand for clients with varying levels of IT and cybersecurity knowledge.
  - Postconditions:
    - The clients can now download reports in form of PDF or HTML.
  - Alternative flows:
    - Should display an error if failed to download reports

Acceptance Criteria

Acceptance criteria are conditions that a software product must meet to be accepted by user, customer, and stakeholder. They are specific and detailed conditions written alongside user stories that defined what the system must do to satisfy the user story. They help ensure that the requirements are testable and that the delivered functionality works as expected.

- **Real-time Threat Detection:** The QaaS app must integrate with SentinelOne EDR API to receive real-time alerts and notifications about detected security threats, including but not limited to:

    - Number of detected threats
    - Threat severity levels
    - Endpoint health status
    - Compliance posture
- **Threat Severity Identification:** The app should categorize each threat based on its severity level, incident status, and security verdict to prioritize response efforts.
- **User-friendly Interface:** For the clients, the UI should be intuitive and user-friendly, ensuring that the customers can easily navigate through page.
- **Customization and Flexibility:** The app should allow for customization of data visualization and response guidance based on the user preference, organization's policies, procedures, and specific client requirements.
- **Technical Data Display:** The system should display detailed technical information about each device, including but not limited to:
    - Endpoint name and category type (server, computer, laptop, workstation)
    - Operating system type and version
    - Hardware specifications (e.g., CPU, RAM, local storage type and details)
    - Network connectivity status (last activated date, IP address, is it connected to the internal network, last username who used it)
    - Installed applications and their versions (along with their dependencies and potential risks)
- **Health Status Indicators:** Each device entry should include health status indicators (e.g., color-coded icons) to quickly identify devices that require attention based on predefined criteria such as:
    - Connection status (offline/online)
    - Performance metrics (e.g., CPU usage, memory usage)
    - Security compliance (e.g., up-to-date AV definitions)
- **Filtering and Sorting:** The system should allow the users and personnel to filter and sort devices based on various criteria (e.g., customer name, device type, health status) to streamline monitoring efforts.
- **Real-Time Updates:** The system should update device information in real-time to ensure that helpdesk personnel and clients have access to the most current data.

MoSCoW Analysis

Must have	Should have	Could have	Would have
Showing SentinelOne data in tables	Search functionality	User session to store user's preference over the visualization they picked from the last session	Integration with N-Central API data
Visualization of data in the form of graphs and charts	Pagination functionality in tables to improve speed performance	Responsiveness to other screen resolution like on mobile with iPhone 11, Galaxy Note 20, etc,	Utilizing SentinelOne Vigilance package
Scheduled functions in Firebase to renew SentinelOne API key once it's about to expire	Sorting functionality	Unit test all the widgets of the SentinelOne pages	Dark mode, light mode
Securely store the API keys and other important information in Google Secret Manager	Filter column functionality	Utilizing Redis DB to improve querying speed	Different languages option (German, English, etc,)
Visualizing timeline of threats from SentinelOne data	Showing the CVE information from outdated applications according to NIST	Showing the latest SentinelOne news feed	Drag and drop functionality to widgets

Ensure proper permission for different User Tags			Resize the widgets' size (width and height)
Ensure that users from another company can only see the data specific of their own company			

Legend:

- Search functionality: utilizing Algolia AI powered search functionality (storing the data in and retrieving data from Firestore DB)
- Pagination: will make use of limit and skip functionality of SentinelOne API. In the page, there should be an array that stores the initial first n data retrieved and when the user clicks to show the next page, another API call should be called to for the next n data, and the array should store those as well. In the end when a user clicks to the end of the page, all the data has been retrieved and store nicely. When the user is at the end of the page all the data has been fetched and stored on the array, so when the user navigates back, there is no need to fetch data from the API anymore. This will be done in the back end from the Firebase cloud functions.
- Sorting: by column names and ascending or descending functionality of the table, making use of SentinelOne API onSort and sortOrder functionality

Chapter 3 Data Model

This chapter describes what data will be used in the project. It also shows how the data will flow and what are the relationships between each component in the system.

This project will utilize Firebase Firestore as its primary database. Firestore itself is a NoSQL document-oriented database. Because the nature of NoSQL databases that are designed to store data that do not have a fixed structure that is specified prior to developing the physical model, the focus is shifted on the physical data model. The developers who use NoSQL typically developing applications for massive, horizontally distributed environments. This puts emphasis more on figuring out how the scalability and performance of the system will work. But they also still need to think about the data model they will use to organize the data.

NoSQL DBs do not have a schema in the same rigid way that relational databases have a schema. There are 4 types of NoSQL database: document-based databases, key-value stores, column-oriented databases, and graph-based databases. NoSQL being the document-oriented database, typically store data in JSON, BSON, and XML format. Because the nature of SentinelOne API calls, only JSON and XML file format are the focus of the development. In a document-oriented database, documents (or items) can be nested, and elements can be indexed for faster querying.

The tables in relational databases are called collections in NoSQL database. They are the containers for documents that share a common structure or purpose. Unlike the traditional RDBMS, collections do not enforce a schema, allowing documents withing the same collection to have different fields or structures or types. Documents are the basic unit of data storage in NoSQL, and each is a JSON-like object that contain key-value pairs. Documents are stored within collections and represent individual records or entities. They can contain nested objects and arrays, providing flexibility for storing complex data structures. Each document would then consist of key-value pairs, where the key is a field name, and the value is the corresponding data that wanted to be stored. Key-value pairs are like rows in relational database, but with more flexibility in terms of data structure. The value can be various types, including strings, numbers, Boolean values, arrays, nested objects, and even binary data.

The most widely adopted NoSQL document-databases are usually implemented with a scale-out architecture. Providing a clear path to scalability of both data volumes and traffic.

Instead of ERD, this document will present Class Diagram to represent the structures and relationships of each entity and classes in this software development project, as well as DFD to visualizes the data flow between different types of users and the data sources/ sinks.

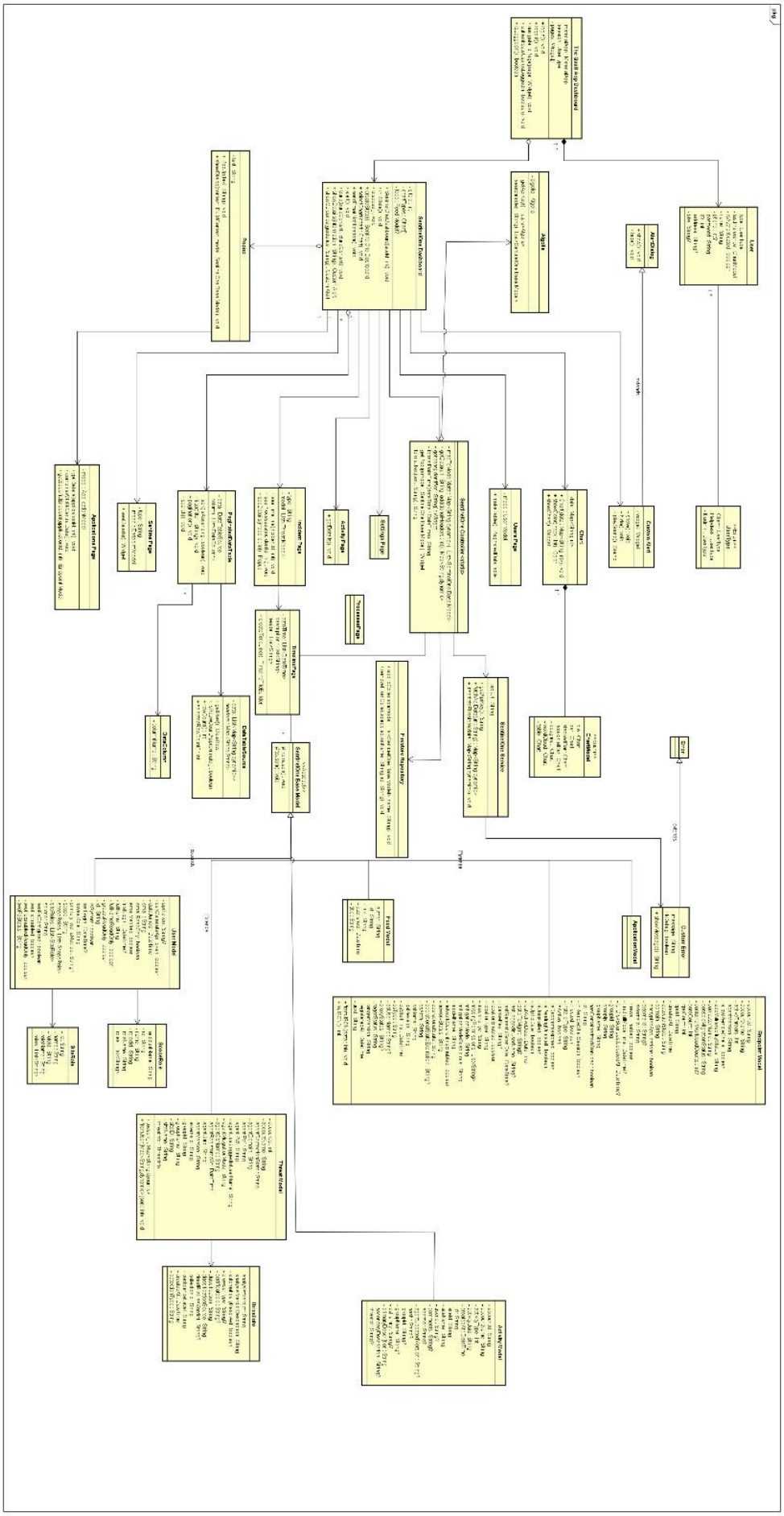


Figure 1 Class Diagram

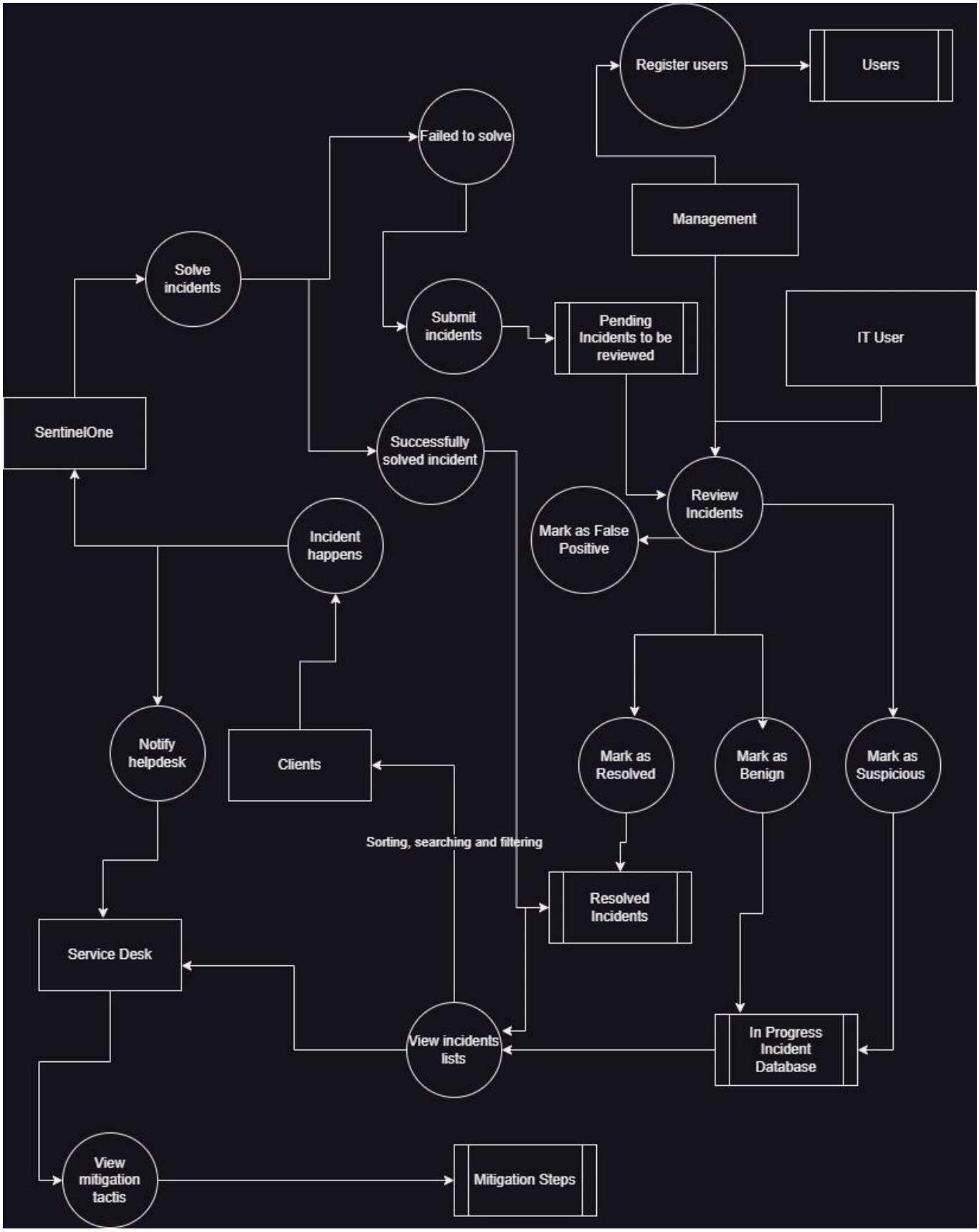


Figure 2 The DFD

Design Patterns used.

Flutter front-end

MVC (Model-View-Controller)

It is a software architectural pattern used usually for designing UIs and organizing code in applications. It promotes a clean separation of concerns, making applications easier to read, develop, scale, and maintain, while also promoting efficient code organization. It also supports modularity and reusability since each component can be developed independently of the others. MVC is widely used in web development (ASP.NET, Ruby on Rails), desktop applications, and mobile app development. The components that will be used in the graduation project are the following:

- Model: it represents the data and the business logic of the application. It encapsulates the data and behaviour of the application domain and responds to requests for information about its state (usually from the view) and instructions to change state (usually from the controller). It interacts with the database or any other data source to fetch or store data. It promotes relational modelling such as ORM for SQL, and ODM for NoSQL. It also is widely used in Repository pattern, where.
- View: represents the presentation layer of the application. It is responsible for displaying the data provided by the model in a user-friendly format. Views can be anything from a simple text output to a complex graphical UI web page. Views can receive data from the model and present it to the user, but they do not directly interact with the Model, only with the Controllers.
- Controller: it acts as an intermediary between the Model and the View. It receives user input from the View, processes it (potentially interacting with the Model to retrieve or modify data), and updates the View accordingly. The Controller interprets user actions (i.e., mouse clicks, keyboard inputs) and translates them into commands for the Model or View. Controllers control the flow of the application, orchestrating interactions between the Model and the View.
- Service: this directory is for the files that will be making calls to Firebase Cloud Functions on various types of triggers, such as HTTP Triggers (Get request), Firestore Triggers (insert, create, update Firestore documents), Callable Functions (On Call

function), Authentication Triggers, Cloud Storage Triggers, Pub/Sub Triggers, Analytics Triggers, etc.,

- Middleware: this directory is made for custom errors and loggings that the author might have when starting this project. The purpose of this customized error is for easier readability during testing and debugging.
- Utility: it provides functions or modules that are used commonly across the application. These utilities are often generic, reusable pieces of code that perform specific tasks or offer helper function to simplify development.
- Shared Widgets: this directory functions as a mix between the Utility and the View. It provides commonly used UI Widgets that are exported to classes or methods to simplify or shorten the code file in the View directory.
- Test: it contains the unit test files that are a crucial aspect of software development that ensures applications behave as expected, catches bugs in early development process, and maintain code quality over time. The test will include widget test to test the UI components, functional test that test the methods in Controller and Service, and Integration test that verify that different parts of the application work together correctly.

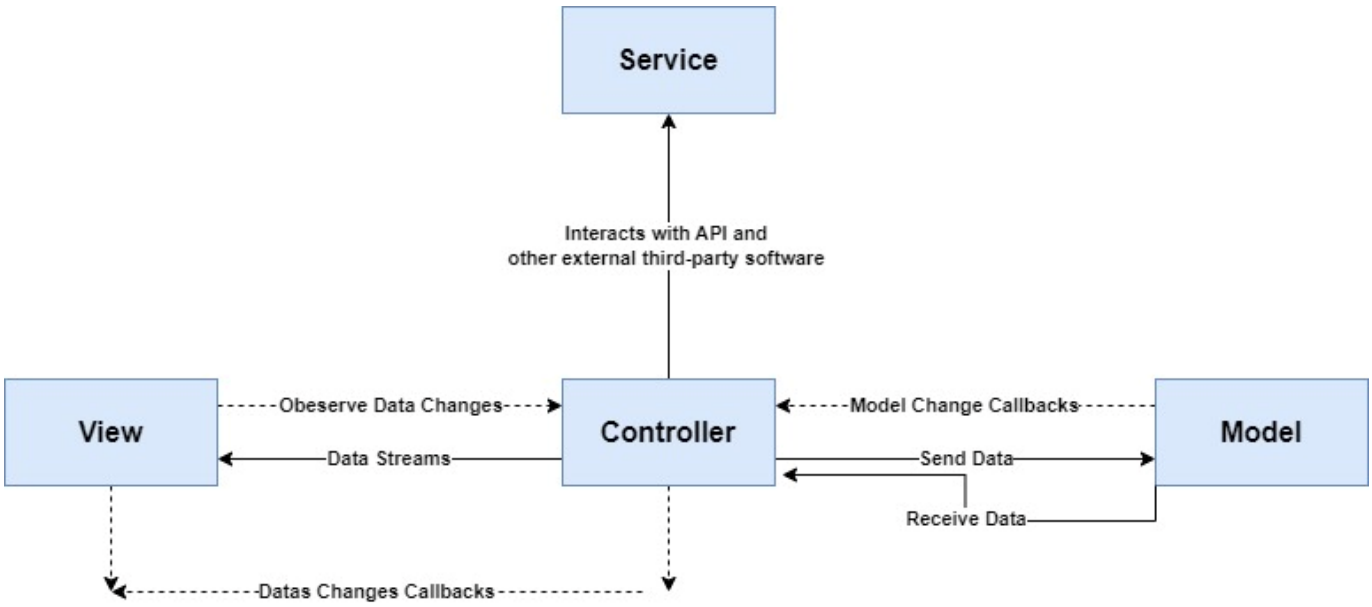
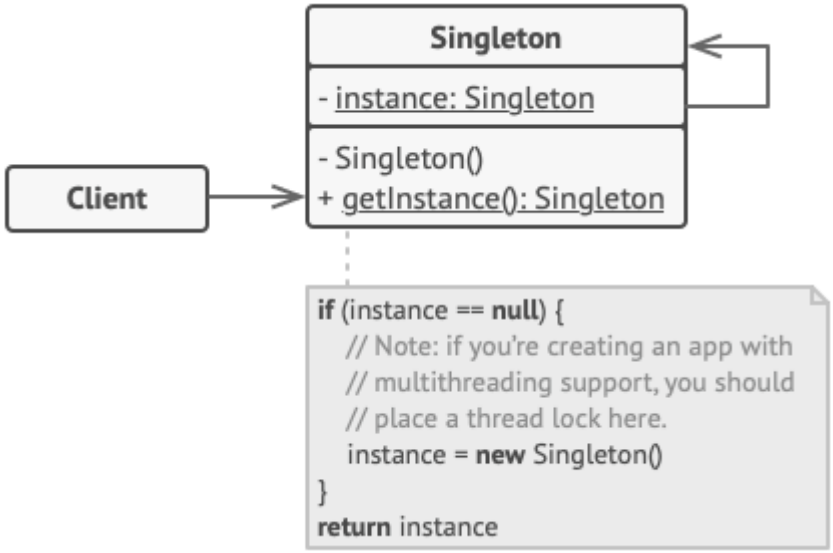


Figure 3 The MVC pattern in the Flutter front-end framework of the QaaS app

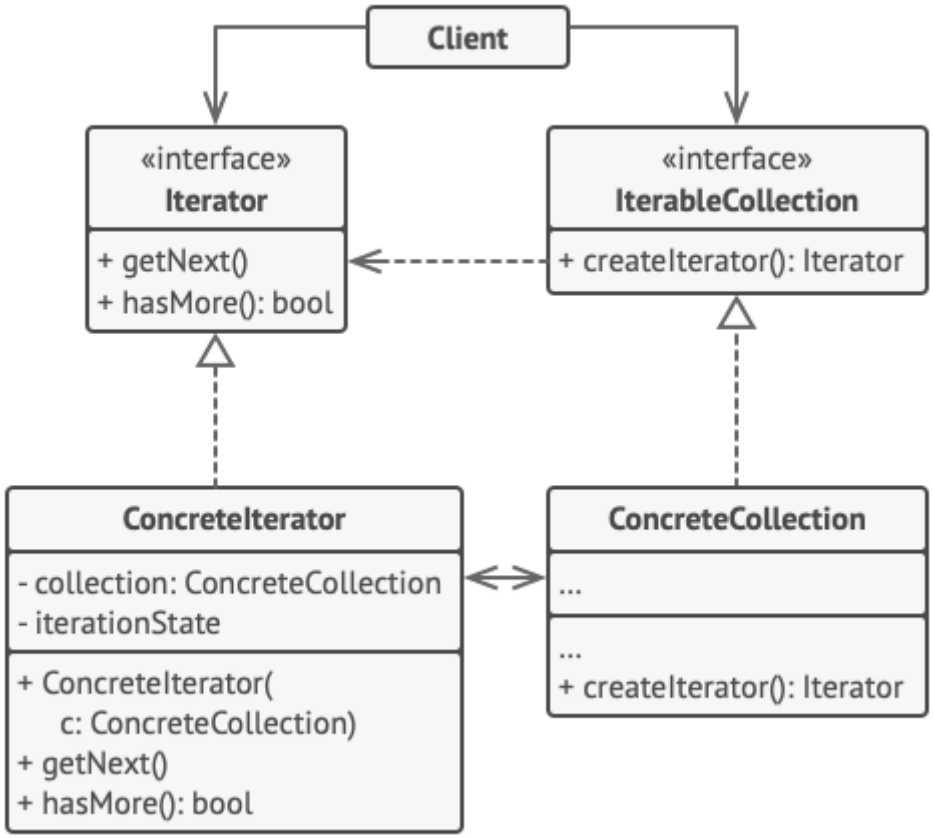
Singleton Pattern

The Singleton Pattern will be used for the Service, Controller, and View files to make sure that they are only called once in the web application and will have to be terminated (clear out of memory) before initiating them again. This would be helpful especially when dealing with View files that contain multiple Widgets, as they need to be disposed first on close after the user leave the page to make sure that there are no memory leaks within the application and to maintain high performance optimization.



Iterator

It is a behavioural pattern that provides a way to access the elements of a collection sequentially without exposing its underlying representation. It mainly allows the developers to traverse through the elements of a collection one by one without needing to know the internal structure of the collection. This pattern will be used as SentinelOne data models will be represented in one abstract base model, and SentinelOne pages of the QaaS app will have many lists or collection of them, sometimes even more than one.



Node.js back-end

Application Element:

Controllers

Controller folders are responsible for storing files that contain the logic of the server. This does not necessarily mean to be files related to API calls, as they are already stored in the Service folder, but

Helpers

This folder is used to store functions that provide common functionality or assists in various tasks throughout the application. It may contain variety of use cases such as constants and enumerations, custom validators, helper classes, and utility functions such as API data validation, date formatting, encryption, String manipulation, etc.,

This folder is responsible for error handling utilities, including error formatting functions, error logging utilities, or custom error classes.

Routers

Routers are used for the files that contain the definitions of routes for handling incoming HTTP request. These routes determine how the server responds to different types of requests (e.g., GET, POST, PATCH, PUT, DELETE) on specific endpoints (URL paths). Because the nature of Firebase Cloud Functions to have allowing independent separation of functions, this directory will not be used a lot in this project, but it is good to have in the future use in case Q-ICT wants to have a Node.js Express server inside one of their Firebase Cloud Functions. The server, because it lives inside a cloud function, will have cold start latency, shorter limited uptime and resource limits compared to the traditional hosted Node.js Express server, but it may also have several upsides that are not yet explored.

Models

The Models contains the data models or schema definitions for interacting with database or other data storage mechanisms. These modes represent the structure and behaviour of the data within the application and are often implemented using ORM or ODM library or a database query builder. It will be used mostly for providing the abstraction later between the application logic and external services (SentinelOne and NoSQL Firestore database), making it easier to manage changes to the external services (such as changing the API version from the current version 2.0 of SentinelOne API, in case the JSON data changes), unit testing, and maintenance.

Middleware

The Middleware is a supplementary directory, serving a very close functionality to Helper folder. It will contain functions that access the request (`req`) and response (`res`) objects for authentication (verifying the authenticity of incoming requests by checking Firestore ID tokens, web sessions, or other authentication mechanisms), and authorization (whether a user has a permission to perform a certain action based on roles (clients, helpdesk, IT administrator)). It is a good to have (Could have in MoSCoW analysis for the application's request-response cycle.

Services

The Service folder contain files that are responsible for making API calls to SentinelOne. The purpose of this folder is made so that there is separation of error messages between the

Controller and the Service, to make it easier for the developer to debug the application when an API call went awry.

Views

This directory will be responsible for storing the files that contain HTML elements to show the JSON response to the user in the server. The files of this directory will not be a lot.

Config

This directory will contain the configuration files that are used throughout the back-end application. The purpose of this directory is very similar to the one in the Flutter front-end application, only this configuration files concern more information to the back end, like the Q-ICT SentinelOne domain name, and the secret names of all the Google Secret Manager secrets that contain the SentinelOne API keys.

Public

This directory will contain images in the form of SVG, JPG, PNG, or a custom-made HTML 404 (not found) or 401 (unauthorized) pages in the Node.js server to show to the user. In case of a malicious attacker trying to access the web server, it should always display the custom-made HTML pages.

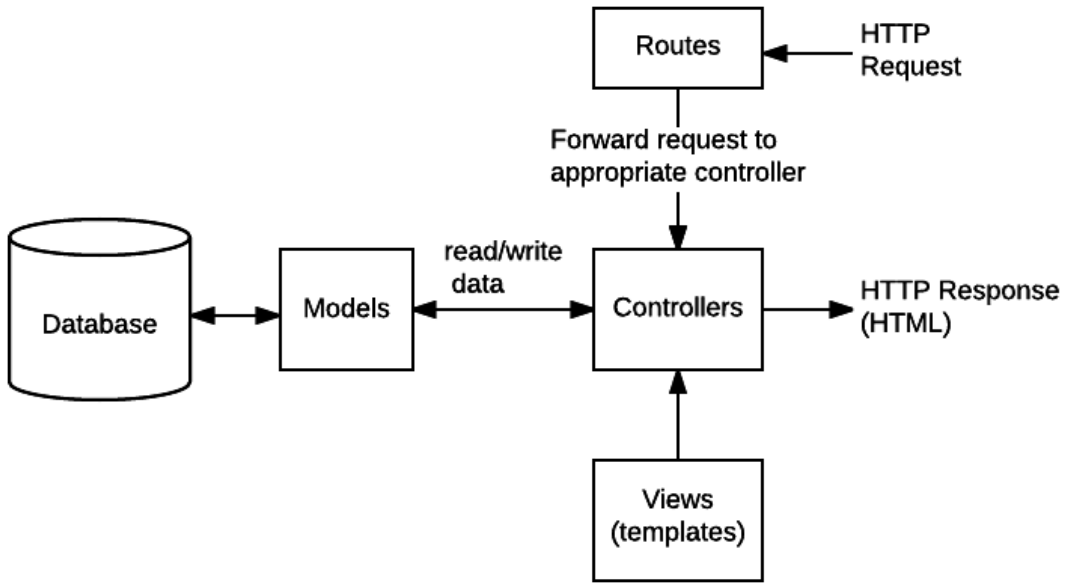
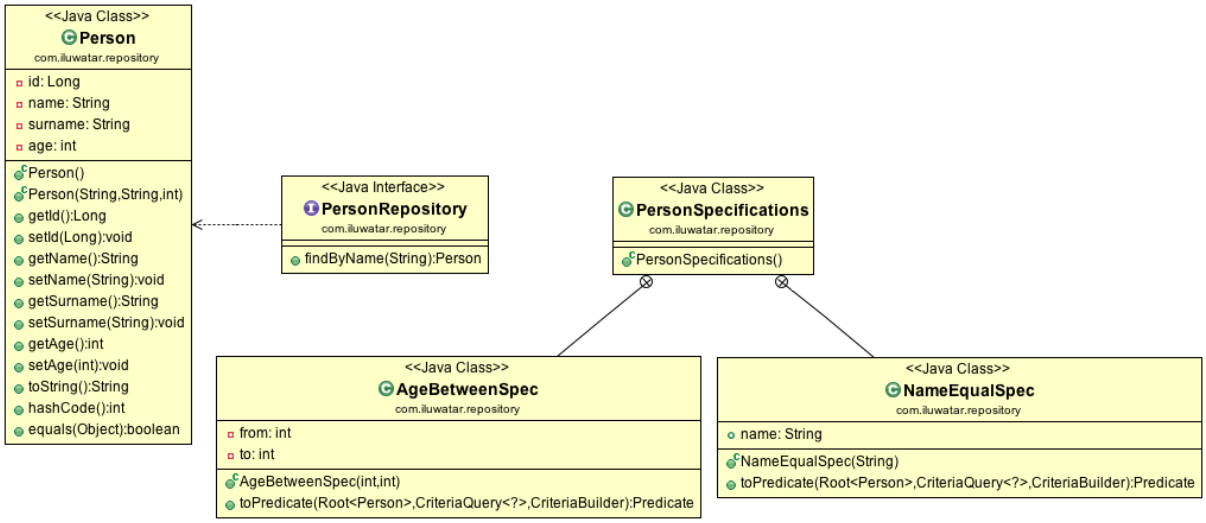


Figure 4 The back-end pattern in Firebase Cloud Functions of the QaaS app

Repository Pattern

This pattern provides abstraction layer between the data source and the business logic, making it easier to switch between different data sources (Firestore, SentinelOne APIs) without affecting the rest of the application. It provides a collection-like interface for accessing domain objects. This pattern is commonly used for managing the data flow when having to deal with large database and REST APIs. This pattern will be very powerful if combine with object mapping concepts like ORM and ODM.



ODM (Object Document Mapping)

It is a concept derived from ORM (Object Relational Mapping). It is a technique used in software development to bridge the gap between object-oriented programming and document-oriented databases. It is a way to map objects in the code to the documents stored in NoSQL database like Firestore. Flutter provides a package for this ODM to Firestore by using *cloud_firestore* and *algolia* for Algolia's mirror NoSQL database for the search functionality. In the back end, Node.js provides *firebase-admin* package that will be used for logging in and adding/deleting data to the correct collection.

User Interface and System Design

Wireframes

Sentinel One Dashboard

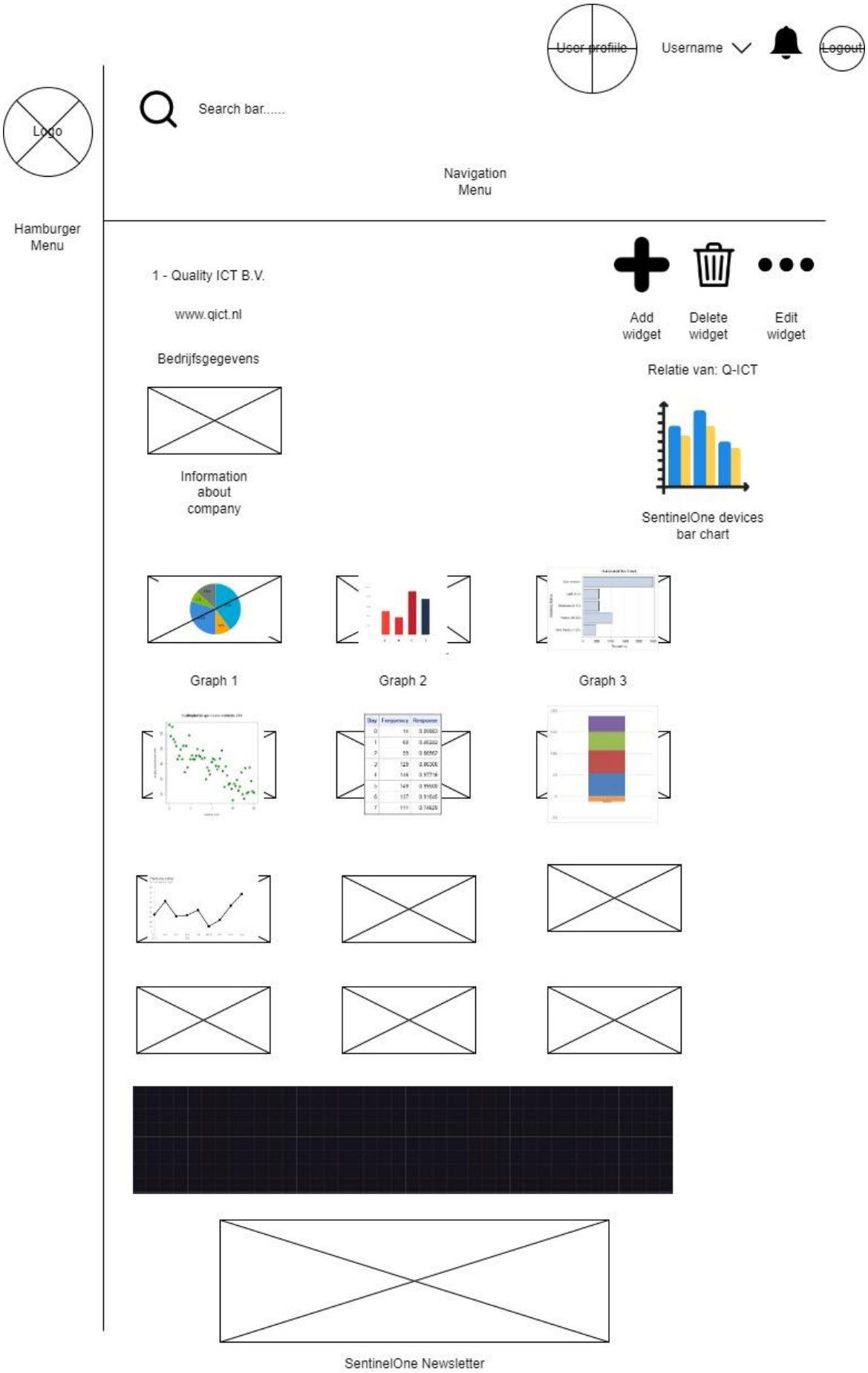

User profiile
Username
Logout
Logo
Search bar.....
Navigation Menu
Hamburger Menu
1 - Quality ICT B.V.
www.qict.nl
Bedrijfsgegevens
Information about company
Add widget
Delete widget
Edit widget
Relatie van: Q-ICT
SentinelOne devices bar chart
Graph 1
Graph 2
Graph 3
SentinelOne Newsletter

Figure 5 Main SentinelOne Dashboard of the QaaS app with the customizable visualization widgets

Used SentinelOne API(s):

- Get Agents (filter: Site ID)
- Get Threats (filter: Site ID)
- Get Applications (filter: Site ID)
- Get Ranger Table
- Get Rogues Table
- Get S1 RSS Feed (general/ no filter)

Widgets:

- Pie chart
- Vertical bar graph
- Vertical stack bar graph
- Horizontal bar graph
- Line chart
- Scatter chart
- Word cloud
- Table

SentinelOne data specific page:

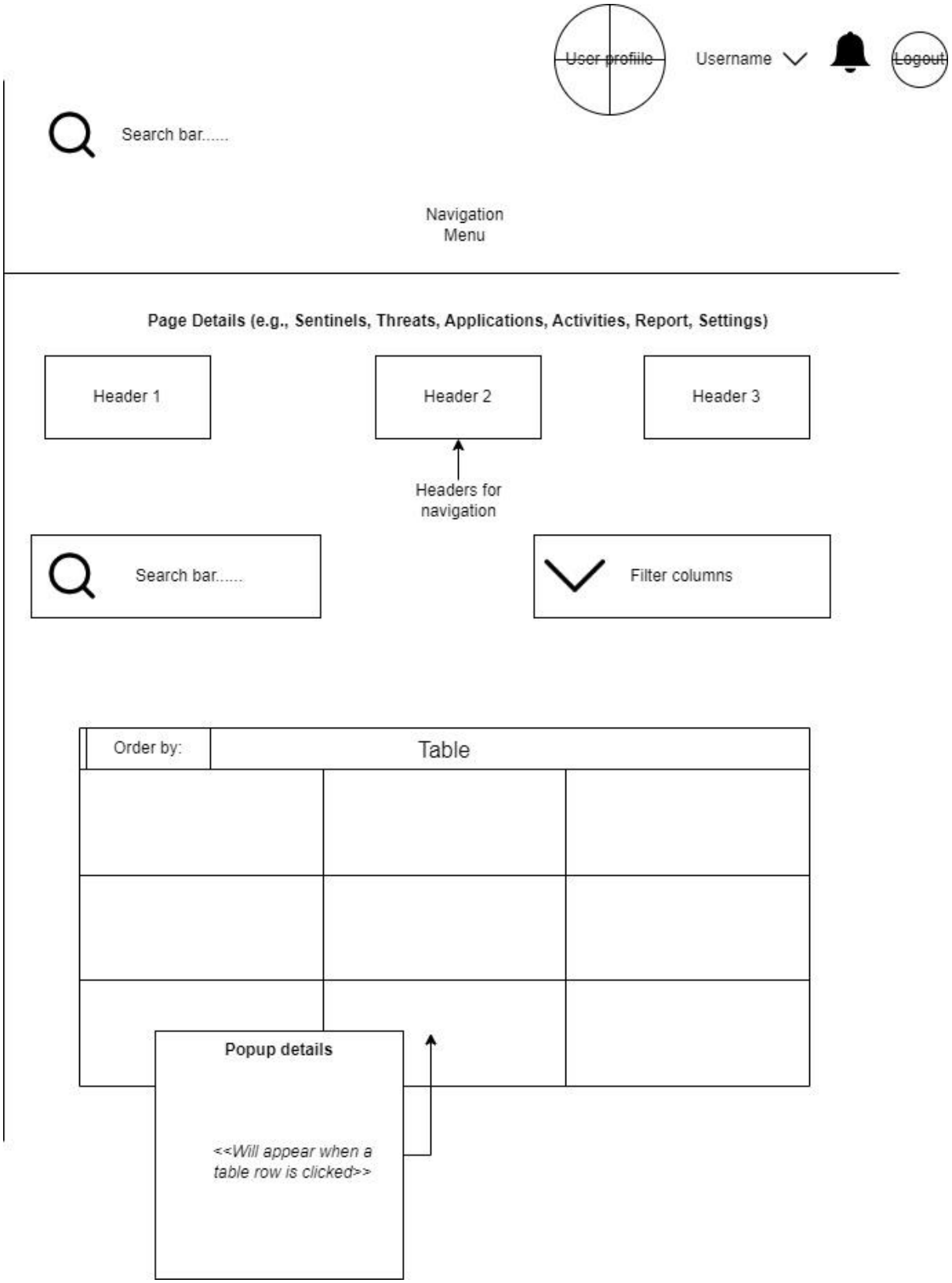


Figure 6 SentinelOne data specific page, will show popup with more details if a row of the table is clicked. WIll contain information about endpoints, threats, applicaitons, and activities.

Used SentinelOne API(s)

- Endpoints:
    - Get agents
    - Get endpoint applications (filter: agent id)
    - Get application process (filter: agent id)
- Threat
    - Get threats
    - Get alerts
    - Get threat notes (filter: threat id)
    - Get agents (filter: agent id)
- Application:
    - Get application with risks
    - Get aggregated applications with risks
    - Get application inventory endpoint
    - Get application inventory
    - Get application CVE
- Activity
    - Get activities
- Miscellaneous
    - Get latest SentinelOne news feed
    - Get customizable notes

Threat Processes Page

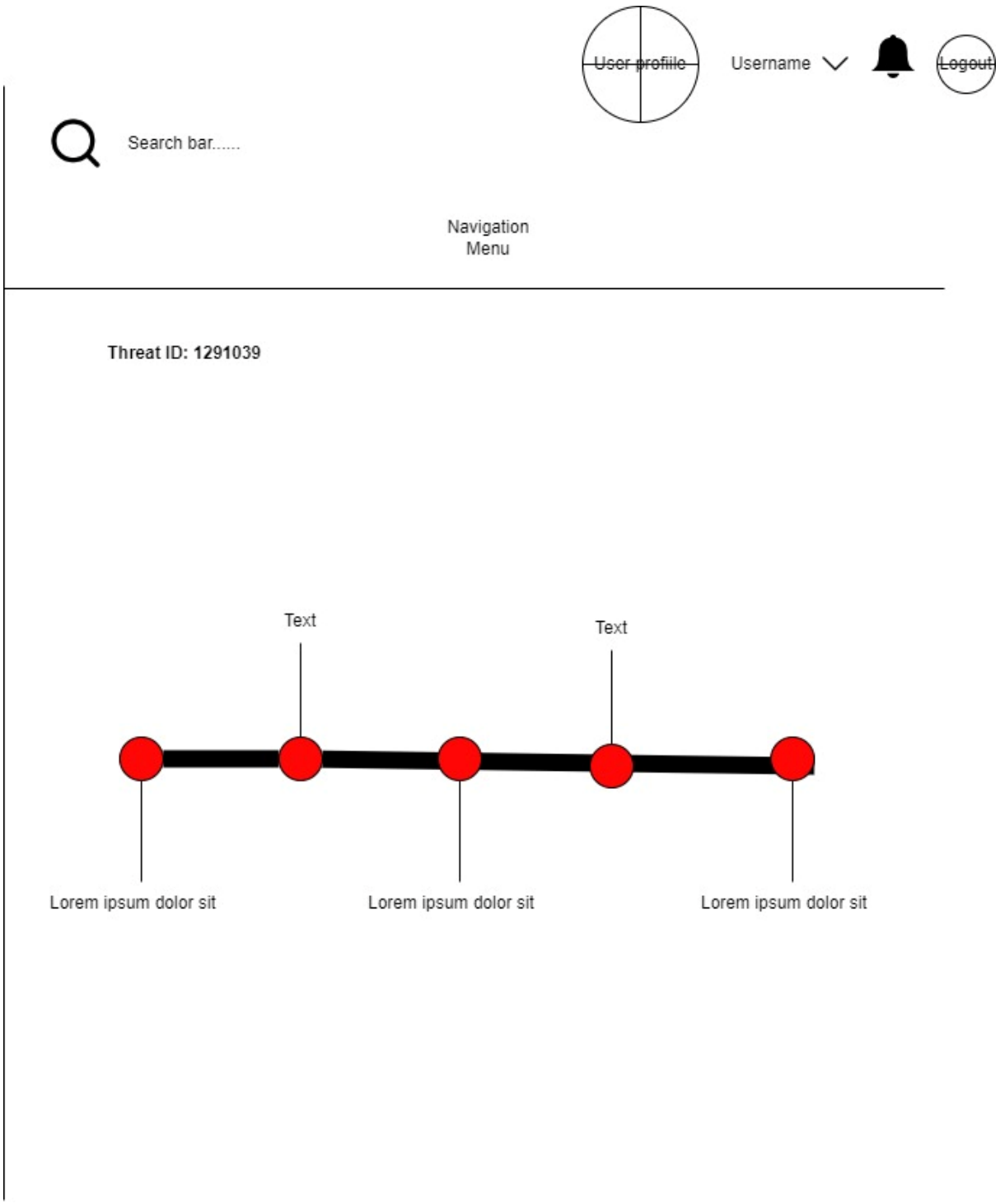
Used SentinelOne API(s):

- Get threat process (filter: threat id)

Widgets:

- Threat process widgets

Threat timeline page



Used SentinelOne API(s):

- Get Threat Timeline (filter: threat id)
- Get activity types

Widgets:

- Timeline widget

Settings Page

Used SentinelOne API(s):

- Get Global Policy
- Get Site Policy (filter by: Site ID)
- Get Group Policy (filter by: Group ID)
- List users

Mockups

Chapter 4 – Output

Any system is designed to produce an output, which is based on the input. In the tables below this output is detailed according to each entity and the input provided. It also goes into detail about what information users can expect to be able to get out of the system.

Code Output Product	**O-01**
Name	Overview of device status and health information
User	Clients, Helpdesk, IT admins
Objective	To provide information and export information about endpoint (on user request)
Frequency	Always present on the website and should be updated automatically once the page is refreshed
Data to be exported	Widgets (graphs, charts, table)

Code Output Product	**O-02**
Name	Information regarding a specific endpoint
User	Clients, Helpdesk, IT admins
Objective	To show more detailed information about an endpoint
Frequency	Should make an API call once every day to keep the system up to date (through cron jobs)
Data to be exported	JSON XML

Code Output Product	**O-03**

Name	General information about mitigated threats happened
User	Clients, Helpdesk, IT admins
Objective	To show all the threats that have happened on a specific Site (client's network)
Frequency	Every time the page loads
Data to be exported	Table

Code Output Product	**O-04**
Name	Specific information about a threat
User	Clients, Helpdesk, IT admins
Objective	To more detailed information about a threat that have happened in a specific Site
Frequency	Every time the page loads
Data to be exported	JSON XML

Code Output Product	**O-05**
Name	Threat timeline
User	Clients, Helpdesk, IT admins
Objective	To show the threat timeline and how it is handled by SentinelOne
Frequency	Every time the page loads
Data to be exported	Widgets (UI elements)

Code Output Product	**O-06**
Name	Activities
User	Clients, Helpdesk, IT admins

Objective	To show all activities that have been done in an endpoint for threat assessment
Frequency	Every time the page loads
Data to be exported	JSON XML

Code Output Product	**O-07**
Name	Conclude suspicious activity
User	Helpdesk, IT admins
Objective	To summarize and conclude when the threat happened based on the infected endpoint activities for incident identification
Frequency	On user's input
Data to be exported	JSON XML

Code Output Product	**O-08**
Name	Installed applications
User	Helpdesk, IT admins
Objective	To show all installed application on a specific endpoint
Frequency	Every time the page reloads
Data to be exported	JSON XML

Code Output Product	**O-09**
Name	Risky outdated installed application
User	Helpdesk, IT admins

Objective	To show the users what are the risk in their environment by showing outdated applications that may impose potential risks
Frequency	Every time the page reloads
Data to be exported	JSON XML

Code Output Product	**O-10**
Name	Application Inventory
User	Clients, Helpdesk, IT admins
Objective	To show application inventory to the users
Frequency	On user's input
Data to be exported	JSON XML

Code Output Product	**O-11**
Name	Settings
User	Helpdesk, IT admins
Objective	To show SentinelOne settings to the user
Frequency	On page reload
Data to be exported	JSON XML

Code Output Product	**O-12**
Name	Reports
User	Helpdesk, IT admins
Objective	To show monthly overview reports of the environment that the employee of Q-ICT or SentinelOne has made automatically to the users

Frequency	On page reload
Data to be exported	JSON XML

Code Output Product	**O-13**
Name	Download reports
User	Clients, Helpdesk, IT admins
Objective	To download reports from the dashboard
Frequency	On user's input
Data to be exported	PDF

Code Output Product	**O-14**
Name	E-mail notification and web alert
User	Helpdesk, IT admins
Objective	To notify the associate user in case of a cyber threat happening
Frequency	Every time a cyber threat happened on a client machine
Data to be exported	Emails, web alert, and sound

Code Output Product	**O-15**
Name	Cyber threat emergency guidance
User	Helpdesk
Objective	To inform the helpdesk about the instructions regarding what actions can be undertaken in case of an unmitigated cyber threat happening on a customer/company computer from SentinelOne
Frequency	Every time a cyber threat happened on a client machine

Data to be exported	Web pop-up

Code Output Product	**O-16**
Name	Newsletter
User	Clients, Helpdesk, IT admins
Objective	To show to the users the SentinelOne newsletters from the API
Frequency	Every time the page reloads
Data to be exported	JSON XML

Chapter 5 Required Input

The aim of this chapter is to show the desired input. It will consist of a table for each function, that is receiving input. The table will have these categories: Code Output (a unique code for every function), Name (name of the function), Authorization (the users that will have access to use this function), Objective (the goal of the function, what data is it supposed to received), Description (where the data will be inputted on the screen), Frequency (how often will the input be put in), Screens Used (which parts of the web application (tabs) will be used when entering the input).

Input Task Login

Code Input Task	**Input Task IT-01**
Name	User ID
Authorization	Clients, helpdesk, and IT admins
Objective	To log in into the QaaS app and present the system with the ID for give it the right authorization
Description	The following data must be inserted into the login screen: • Username (email). • Password. • TOTP code sent through mobile phone. If the user does not have a login credentials, that means they are not an official user/client of Q-ICT, and therefore should contact the IT admins for help
Frequency	Anytime a user wants to log in. The credentials can be saved in a cookie. When the user closes the browser or log off their device, the application should automatically log them off.
Screens used	Login Screen

Code Input Task	Input Task IT-02
Name	Site ID
Authorization	Clients
Objective	To give the system with the Site ID. It is required for the client user only, as they can only see security information and status of their own devices.
Description	In SentinelOne, "Sites" refer to **logical grouping** of endpoints (devices or systems protected by SentinelOne), within an organization's network infrastructure. Sites are typically organized by SentinelOne based on geographical locations, departments, or other criteria that makes sense for the organization's structure and management needs.
Frequency	Whenever the client wishes to navigate to SentinelOne page.
Screens used	QaaS app Dashboard Screen

Code Input Task	Input Task IT-03
Name	Incident/Threat ID
Authorization	Clients, helpdesk, and IT admins.
Objective	To be able to specify a threat to view more detailed information
Description	To be able to show each threat more detailed information along with the timeline on how that threat is mitigated by SentinelOne, the app will take the user to a different page.
Frequency	Once per user's request
Screens used	SentinelOne Threat Timeline Page

Code Input Task	Input Task IT-04
Name	Agent ID

Authorization	Clients, helpdesk, and IT admins
Objective	To pass in the Agents ID, to view the device information in more detailed
Description	Agents in SentinelOne refers to the AV itself that is installed in a specific endpoint to provide comprehensive security protection. These agents utilize the behavioural AI and continuous monitoring; therefore, it is crucial that the Agents get access to the highest permission possible to give real-time information to the central system.
Frequency	Once per user's request
Screens used	SentinelOne Device page

Code Input Task	**Input Task IT-05**
Name	Input search/filter/ sort
Authorization	Clients, helpdesk, IT admins
Objective	To filter, search, and sort the data in either the front-end/back-end.
Description	The data which will be displayed in tables, need to have functionalities such as searching, filtering, and sorting based on various criteria (customer name, device type, health status) to streamline monitoring efforts.
Frequency	Once per user's request
Screens used	SentinelOne Overview page

Code Input Task	**Input Task IT-06**
Name	Visualization type
Authorization	Clients, helpdesk, and IT admins
Objective	To be able to select different visualization types of the data displayed (pie chart, bar, graphs, etc,)

Description	The app will visualize the SentinelOne API data (whether it is about the endpoint or threat information) to make it easier to analyze
Frequency	Once per user's request
Screens used	SentinelOne Overview page

Chapter 6 Menu Structure and Authorization

This project will have 3 different layers authorization: the Customers, the Helpdesk, and the Developers. Because of the security compliance regulation as the intern is not a full-time employee of the company (as stated in the Chapter 2 in Thesis), this project will only handle the GET request (Read operations) of the API call and nothing more than that, as any additional request would have a substantial adversity effect on the system.

The clients have the most basic authorization and authentication across the platform. They need to be registered on the Q-ICT database, therefore making them an official client, and register their user email, password, and phone number to the system. Once registered and the phone number verified, they can log in to the QaaS web application by entering their registered email username and password. Once they put in the right information, an OTP verification code will be sent to their registered phone number, thus ensuring MFA. When viewing the SentinelOne page, the clients can only view their own device security information given by SentineOne. The page

The helpdesk has more rights and authority than the client user. Once logged in with 2FA, they can see the overview of all the client's device health and status.

The IT user has the most rights in the QaaS app where they can update the modules and templates of the QaaS app, limiting what the user can see. The IT point of view will be mostly out of scope for this project, as the author will have limitation of access in both the QaaS app permission and SentinelOne requests.

Please note *that the scope of this project, as defined in the Thesis will not include doing POST, PUT, PATCH, and DELETE requests from the API. This table represents the ideal complete implementation of the SentinelOne integration and will treat the author to have a full access to the API. Otherwise, the whole entirety of the table will only include Read operation.*

Main	**Sub-menu**	**Authorization**		
		Clients	Helpdesk	IT admins
Dashboard Page	Incident Page Endpoint Page	C, R, U, D	C, R, U, D	C, R, U, D

Incident Page	Incident Detail Page Alert Page	R	R, U	C, R, U, D
Endpoint Page	Application Page Endpoint Detail Page	R	R	R
Alert Page		R	R	R
Incident Detail Page	Incident timeline	R		
Activity Page		R	R	R
Application Inventory Page		R	R	R
Policy Page		R	C, R, U, D	C, R, U, D
Settings Page		R	C, R, U, D	C, R, U, D
Helpdesk page		N/A	C, R	C, R

Chapter 7 Technical Consequences

Are any extra workplaces required, and if so, with which technical facilities?

The author will use Q-ICT's official office as his main workplace along with other premises such as NHL Stenden University Emmen campus, the Library Emmen, and the author's own office in his room if some situations desire the author to not to be there. A supply of secure and good internet connection will be provided by Q-ICT, in the form of an internal network connection, linked to his docking stations. Additionally, the company will provide 2 additional ultra-wide screens to the author's workstation, with additional resources available for disposal upon request.

Which special other technical equipment is required? (E.g., bar code equipment)

Besides the customer's own device that they want to install the SentinelOne agents on, there is no special equipment required.

On which software or stand-alone computer will the software be used?

The project itself is a continuation of a pre-existing infrastructure, if said infrastructure is missing or not up to the required specifications, an update will be recommended. There should not be a need to purchase any other server or equipment, as the new website should be able to run on the same server that the existing platform is currently running, and therefore it will be using the same domain name.

Which internet facilities are required for the software?

The product should work on any cross-platform web browser, but in order the user to get the best performance and all its existing functionality, Microsoft Edge and Google Chrome is recommended by the author, as Flutter and Firebase is powered by Google. Another thing that should be taken into consideration is constant internet connection to be able to use the application, as some of the pages use StreamBuilder in which it requires constant internet connection to streamline the data. For the system to run perfectly and qualitive enough to fulfil its purpose, users are required to have at least 250Mbps connection with below 10 milliseconds response time to the web server.

Is special system software or other technical equipment required to run the software?

No, the user only needs to be registered as an official Q-ICT customer by subscribing to one of our offered subscriptions and have an Internet connection to access it.

How will the system be supported?

The back-end code will be hosted by Google in the form of Firebase Cloud Functions, as the data communication facilities. The QaaS app itself is also hosted by GCP (Google Cloud Platform), and the author will be working on the test environment. Once the company sees that his work has fulfilled as expected with the stakeholder's requirements, it will be moved to the live environment in which it can be accessible to Q-ICT and its clients to access on the Internet.

Which data communication facilities are required?

E-mail addresses, passwords, and phone numbers are required for creating an account.

Which (special) printers are required or is a network printer sufficient?

No printers are required to access the web application.

Which back-up facilities are required? Are separate back-up facilities required or can the back-up be included with the other data from the network?

The Firestore DB from Firebase will serve as the back-up facilities in case of data loss.

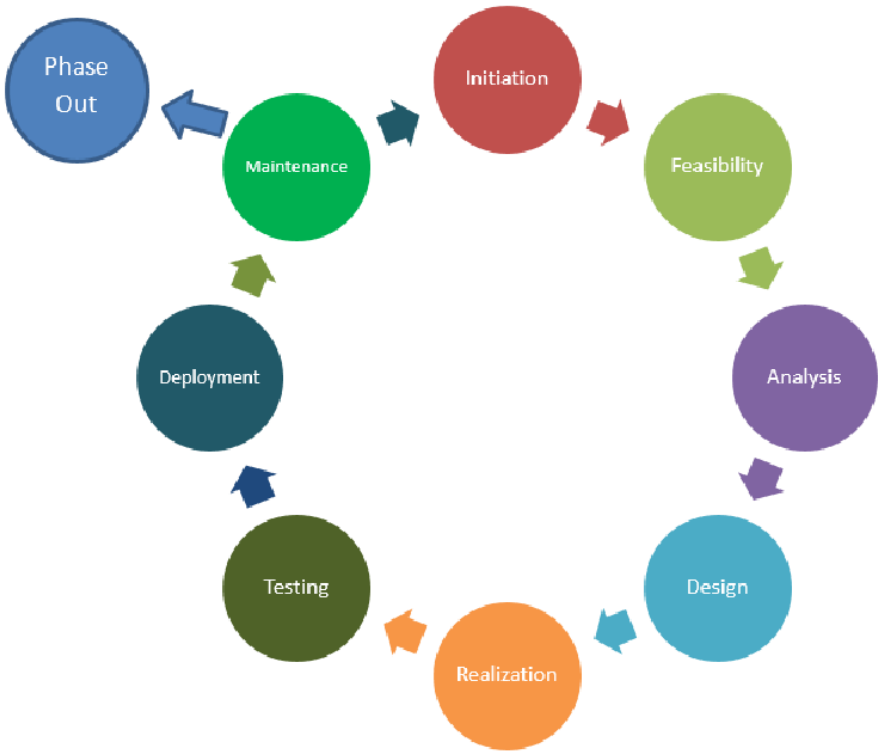
Are extra security measures required for data protection?

Upon logging in (authorization) the app will do a Captcha check to make sure that the user is a human and not a bot.

Who will take care of implementation and maintenance?

The maintenance of the final product is left to the Q-ICT software development department. In the end, the author ought to produce a comprehensive documentation (in the form of README.md and code commenting or User and Developer Manual), as the means of product usage, counsel, and recommendation for the Q-ICT official developers and upcoming interns to ensure its continuity. However, the author will not be taken into any account regarding the maintenance or phasing out of the product beyond completion as specified in the graduation

project module-book (*See Graduation Project Manual Information Technology 2023-2024 Appendix C*).



Appendix B

TD (Technical Design)

Technical Design

Christopher Sulistiyo

Quality ICT B.V.

Christopher.sulistiyo@student.nhlstenden.com

ICT & IC Information Technology Department Emmen

Version 1.1 - 21/02/2024

NHL STENDEN university of applied sciences

Version Control

Version	Activities	Date
Initial version 1.0	Draft version	06/02/2024

Remarks

Any changes and new developments that have a significant impact on the project proceedings will be noted here.

Contents

1 Introduction **4**
1.1 Background and Context . . . 4

2 System Requirements **5**
2.1 Client-Side Requirements . . . 5
2.2 Sever-Side Requirements . . . 5
2.3 Integration Requirements . . . 6
2.4 Database Requirements . . . 6
2.5 Deployment Requirements . . . 6
2.6 Non-Functional Requirements . . . 7

3 SentinelOne API Architecture **8**
3.1 Endpoints . . . 10
3.2 Threats . . . 12
3.3 Ranger . . . 17
3.4 Rogues . . . 18
3.5 Miscellaneous . . . 19

4 Database Design **20**
4.1 How Collection is Stored . . . 20
4.2 Data Types . . . 22
4.3 Collection Types . . . 22
4.4 Index Types . . . 22

5 Back-end Architecture **25**
5.0.1 Compliance with ESLint compiler . . . 27
5.1 SentinelOne NPM packages . . . 31
5.2 Downloading PDF and HTML report files from SentinelOne to the client . . . 32

6 Front-End Architecture **36**
6.1 Flutter . . . 36
6.2 Architecture Components . . . 37
6.2.1 State Management . . . 37
6.2.2 Routing and Navigation . . . 38
6.2.3 UI/UX Design Principles . . . 38
6.2.4 Security . . . 38
6.2.5 Performance Optimization . . . 38

7 Test Strategy **40**
7.1 Test Objectives 40
7.2 Test Scope 40
7.3 Test Types 40
7.3.1 Functional Testing 40
7.3.2 Performance Testing 41
7.3.3 Usability Testing 41
7.4 Test Environment 41
7.5 Test Schedule 41

8 Security Considerations and Author's Remark **42**
8.1 Structure of the front-end codebase 42
8.2 Translation 42
8.3 Future modelling 42

Chapter 1

Introduction

1.1 Background and Context

This document serves as a comprehensive blueprint for the development and integration of this project. Therefore, this document will outline the technical specifications, architecture, methodologies, and implementation strategies necessary to achieve the project's objectives, as it is essential that every artefact is documented in the software development lifecycle to provide a detailed roadmap for future developers, project managers, and stakeholders for the making of new features for this project. Additionally, it ensures that all team members have a clear understanding of project's scope and its objectives in terms of software development, which is to outline the architecture, components, and implementation details of the SentinelOne integration into the QaaS App. Specifically, this document will:

- Provide an overview of the project's objectives, scope, overall architecture, and design principles guiding the integration process.
- Describe the project's architecture and components involved in the integration, including the client-side front-end built with Flutter, and the back-end powered by Firebase Cloud Functions and Node.js.
- Explain the workflow by illustrating the data flow and interaction between various components, ensuring seamless integration and real-time updates.
- Outline the implementation details and methodologies.
- Provide a detailed explanation of the database design.
- Discuss and highlight the security considerations, measurements and future improvements. It will further emphasize the security protocols and best practices implemented to maintain data integrity and privacy.
- Outline the development process by detailing the approach, including the use of design patterns and how it facilitates maintainability and scalability.

Chapter 2

System Requirements

System requirements are essential for the integration of SentinelOne EDR to the QaaS App. It serves as the foundation for the entire development process. It provides a clear understanding of what the system must achieve, guiding developers in creating a solution that meets the specified needs.

2.1 Client-Side Requirements

- Browser Compatibility: the web application should be compatible with modern web browser such as Google Chrome, Mozilla Firefox, Microsoft Edge, Opera, and Safari.
- Device Compatibility: the web application should be responsive and function correctly on various devices, including desktops, tablets, and smartphones.
- Minimum Screen Resolution: the application should support a minimum screen resolution of 1280x720 pixels.
- Performance: to ensure a smooth performance with minimal latency, even on devices with lower hardware specifications.
- Security: implement client-side security measures such as HTTPS, secure cookies, and Content Security Policy (CSP) to ensure data protection and security against common web vulnerabilities.
- Usability: design the UI to be intuitive and easy to navigate for non-technical users, incorporating accessibility standards.

2.2 Sever-Side Requirements

1. Firebase Cloud Functions:
   - Version: Use the latest stable version of gen 2.0 of Firebase Cloud Functions
   - Memory Allocation: at least allocate between 256 MB to 2 GB of memory per function, based on performance requirements.
   - Timeout Settings: set appropriate timeout settings for functions (default is 60 seconds), depending on the function complexity in case an infinite loop occurs or other unexpected behaviour.
   - Dependencies: include necessary Node.js dependencies, such as `firebase-admin`, `firebase-functions`, `axios`, and other project-specific packages.

Additionally, ensure proper logging, monitoring, and error handling mechanism are in place to maintain system reliability and quickly address any issues.

2.3 Integration Requirements

- SentinelOne API Integration:
  - Authentication: implement secure API authentication by using API key that have the appropriate access to fetch the wanted data.
  - Data Fetching: schedule regular intervals for fetching data from SentinelOne API to ensure up-to-date information is available in the application. In case of an error, do not show the technicality in the client instead make an appropriate user-friendly error page and message and redirect the user to that page.
  - Error Handling: Implement robust error handling for API calls to manage failures and retires gracefully. Also, document what was the source of the error on the Firebase function logs for better debugging.
- Firebase Authentication: ensure that the user is always logged and have necessary permission to access the SentinelOne dashboard of the application. Authentication can be implemented by using:
  - User Management: utilize Firebase Authentication for managing user sign-up, sign-in, and password recovery.
  - Role-Based Access Control: implement role-based access control to restrict access to sensitive data and actions.
- Database:
  - Firestore: use Firebase Firestore for real-time data storage and retrieval.
  - Data Security: implement Firestore security rules to control access to the data.

2.4 Database Requirements

- Database Type: use a NoSQL database such as Firebase Firestore for storing user data, logs, and other relevant information.
- Data Structure: design a schema that efficiently stores and retrieves data, ensuring data consistency and integrity.
- Security: implement database security measures such as encryption, access control, and data validation to protect against unauthorized access and data breaches.
- Scalability: design the database to scale horizontally to accommodate increasing data volume and user traffic.

2.5 Deployment Requirements

- Deployment Environment: deploy the application on a cloud platform, in this project's case Firebase Hosting.
- Continuous Integration/Continuous Deployment (CI/CD): set up CI/CD pipelines to automate the deployment process and ensure consistency across environments.
- Monitoring: implement monitoring tools on the said cloud provider such as Firebase Performance Monitoring to track system performance, detect issues, and optimize resource utilization.

2.6 Non-Functional Requirements

- Scalability: design the system to scale horizontally to handle an increasing number of clients and devices.
- Reliability and Availability: ensure high availability of the system with minimal downtime. Implement robust error handling and recovery mechanisms to maintain system reliability.
- Security: protect sensitive data using encryption in transit and at rest. Using Google Secret Manager to store sensitive information such as API keys and credentials where those data will be encrypted in the cloud. Do not include those in the front-end or back-end code to avoid unintended exposure.
- Documentation: provide comprehensive documentation for developers, administrators, and end-users to understand the system architecture, functionality, and usage.
- Testing: conduct thorough testing, including unit tests, integration tests, and end-to-end tests, to ensure the application functions as expected and meets the specified requirements.
- Compliance: ensure the application complies with relevant regulations and standards, such as GDPR, HIPAA, and PCI DSS, to protect user data and maintain legal compliance.
- Support: provide ongoing support and maintenance to address issues, implement updates, and enhance the application's functionality based on user feedback.
- Maintainability: follow the web architectural design patterns like MVC, MVVM, or MVP to ensure codebase maintainability. Document the code and APIs thoroughly in a README file or code commenting for future developers.
- Usability: conduct usability testing to ensure the application is easy to use for non-technical clients. Provide help and support documentation such as User Manual and/or Developer Manual within the application.
- Compability: ensure the Flutter application works on every screen including mobile screens like Android, iOS, and tabloid devices. Ensure compatibility with various web browsers for the web-based components.

Chapter 3

SentinelOne API Architecture

Before diving deeper into the project's architect itself, it is important to understand what kind of API does SentinelOne provides to its audience and how SentinelOne structures its API.

JSON format JSON is a lightweight data-interchange format that is easy for humans to read and write while being also easy for machines to parse and generate too. It is based on a subset of the JavaScript programming language and is commonly used for representing structured data. JSON is often used for transmitting data between a server and a web application, as well as for storing configuration data.

JSON data is organized into key-value pairs, where keys are strings and value can be strings (enclosed in double quotes '""'), booleans ('true' or 'false'), numbers, objects (unordered collections of key-value pairs enclosed in curly braces "), arrays (ordered collections of values enclosed in square brackets '[]'), or null.

Listing 3.1: Example of a JSON response

```
{
    "name": "John Doe",
    "age": 30,
    "isStudent": false,
    "cars": [
        {"name": "Ford", "models": ["Fiesta", "Focus", "Mustang"] },
        {"name": "BMW", "models": ["320", "X3", "X5"] },
        {"name": "Fiat", "models": ["500", "Panda"] }
    ],
    "hobbies": ["Reading", "Gaming", "Traveling", "Cooking", "Photography",
        "Painting", "Gardening"],
    "address": {
        "street": "Sesame Main Street",
        "city": "New York",
        "zip": "10001"
    }
}
```

REST API: if SOAP is like an envelope, RESTis like a more lightweight postcard. REST APIs are considered the gold standard for scalability and are highly compatible with microservice architecture. It is the often used protocol in the context of building APIs for web-based applications. REST itself is not a protocol, but an architectural style for

designing networked applications, defining a set of constraints and principles that define how web services should be structured and interact with each other.

APIs that follow REST principles are called RESTful APIs. The are RESTful as long as they comply with the 6 guiding constraints of a RESTful system:

- **Client-server architecture**: the architecture is composed of clients, servers, and resources, and it handles requests through HTTP.
- **Statelessness**: no client is stored on the server between requests. The server should process and complete each request independently. Information about the session state is, instead, held with the client. The clients can do this via query parameters, headers, URIs, request body, etc.,
- **Cacheable**: simply, the clients should be able to determine whether this response is cacheable from their side, and if so, for how long. If a response is cacheable, the client has the right to return the data from its cache for an equivalent request and specified period, without sending another request to the server. A well managed caching mechanism can eliminate the need for some client-server interactions
- **Layered system**: client-server interactions can be mediated by additional layers. These layers could offer additional features like load balancing, shared caches, or security.
- **Uniform interface**: this is the core to design RESTful APIs. There should be a uniform and standard way of interacting with a given server for all client types. The uniform interface helps to simplify the overall architecture of the system. This includes 4 facets:
  - Resource identification in request: resources are uniquely identified in requests and are separate from the representations that are returned to the client using URI.
  - Resource manipulation through representations: clients receive files of a uniform that represent resources. These representations must have enough information to allow modification or deletion of the resource's state in the server, as long as they have the required permissions.
  - Self-descriptive messages: each message returned to a client contains enough information to describe how the client should process the information further, such as additional actions that can be performed on the resource.
  - Hypermedia as the engine of application state: after accessing a resource, the REST client should be able to discover through hyperlinks all other actions that are currently available.
- **Code on demand (optional)**: servers can extend the functionality of a client by transferring executable code.

REST APIs are high-performing (especially over HTTP2), time-tested, and support many data formats. They also decouple the client and server, making sure of independent evolution. However, building a true REST API is difficult because it requires a disciplined adherence to the Uniform Interface constraint. Some organizations trade off the long-term benefits of a truly REST API for other HTTP API protocols that have similar benefits but adhere to REST constraints more liberally. REST requests typically include these key components:

- **Endpoint**: the uniform resource identifier that locates the resource on the internet is part of this component. URLs are the most common type of URI.
- **HTTP Method**: this component outlines the four basic processes that a resource can be subjected to: POST (create a resource), GET (retrieve a resource), PUT (update a resource), and DELETE (remove a resource).
- **Headers**: data related to the server and the client are stored in this component. Like in SOAP, one can also use REST headers to store authentication measures such as API keys, server IP addresses, and the response format.

- **Body**: this component contains additional information for the server, such as data that needs to be added or replaced.

Listing 3.2: Different HTTP methods in REST

```
GET /users Retrieve list of all users
GET /users/{id} Retrieve details a specific user by their ID
POST /users Create a new user
PUT /users/{id} Update a specific user by their ID
DELETE /users/{id} Delete a specific user by their ID
```

Listing 3.3: REST's Example Request

```
GET https://api.example.com/users/123
```

There are basically 5 main components of the data that are useful for the user to see in the QaaS App:

3.1 Endpoints

Endpoints within an organization's network that have SentinelOne Agents installed will have their information collected and analysed. This various types of information plays a vital role in maintaining the security integrity of the network. Some overview about an endpoint that SentinelOne gathers and analyses are:

1. **System Information** :
   (a) **Endpoint Name**: the name assigned to the device.
   (b) **IP Address**: the unique identifier assigned to the device on the network.
   (c) **MAC Address**: the unique identifier assigned to the device on the network.
   (d) **Operating System**: details about the OS version and type (Windows, macOs, Linux, etc.).
   (e) **Hardware Specifications**: information about the CPU, memory, disk space, and other hardware details.
   (f) **Last Seen**: the last time the endpoint was active and communicated with SentinelOne Console.
   (g) **Agent Version**: the version of SentinelOne Agent installed on the endpoint.
   (h) **User Account**: The user account of the endpoint.
   (i) **Device Type**: the type of device (e.g., desktop, laptop, server, workstation, mobile device, IoT device, etc.,).
2. **Security Status**:
   (a) **Agent Status**: whether the Agent is active, inactive, decommissioned or needs an update.
   (b) **Health Status**: the overall health of the endpoint from a security perspective.
   (c) **Threat Level**: any current threats and their severity.
3. **Performance Metrics**:
   (a) **CPU Usage**: the percentage of CPU resources being used by the endpoint.
   (b) **Memory Usage**: the percentage of memory resources being used by the endpoint.
   (c) **Disk Usage**: the percentage of disk space being used by the endpoint.

4. **Response Actions**:
   (a) **Quarantine Status**: information on files or processes that have been quarantined.
   (b) **Mitigation Actions**: actions taken by SentinelOne to mitigate security threats (e.g., killing processes, rolling back changes).
5. **Application Information**
   (a) **Installed Applications**: list of applications installed on the endpoint, including versions and publishers.
   (b) **Running Processes**: real-time information on running processes and their resource usage.
   (c) **Active Sessions**: current sessions associated with user accounts, indicating who is actively logged in.
6. **Activity Logs**:
   (a) **Process Activity**: logs of processes that have run on the endpoint, including those flagged as suspicious.
   (b) **Network Activity**: details of network connections made by the endpoint, including potentially malicious connections.
   (c) **User Activity**: information about user logins and actions that could be relevant for security monitoring.
7. **Endpoint Configuration**:
   (a) **Installed Software**: list of software installed on the endpoint.
   (b) **Running Services**: services running on the endpoint.
   (c) **Startup Programs**: programs configured to run at startup.
8. **Threat Data**:
   (a) **Detected Threats**: information on threats detected on the endpoint, including malware, exploits, and suspicious activities.
   (b) **Threat Details**: specific information about each threat, including type, severity, and status (e.g., quarantined, mitigated).
   (c) **Threat Timeline**: the history of threat detection and mitigation actions taken on the endpoint.
9. **User Information**
   (a) **User Accounts**: list of user accounts active on the endpoint, including their names, roles, groups, and privileges.
   (b) **Login Activity**: records of login attempts, successes, and failures, along with timestamps.
   (c) **User Active Sessions**: list of current sessions associated with user accounts, indicating who is actively logged in.
10. **Policy Compliance**:
    (a) **Policy Adherence**: whether the endpoint adheres to the security policies configured in SentinelOne. The data is in *boolean* format.
    (b) **Policy Violations**: detected deviations or violations of security policies, indicating areas of concern.
11. **File and Content Information**
    (a) **File Hashes**: unique identifiers for files, useful for detecting known malicious files.
    (b) **File Types**: information on the types of files being accessed or transferred.

(c) **Content Analysis**: textual content of documents and e-mails, scanned for signs of phishing, malware, or other threats.

12. **Behavioural Analytics**
    (a) **Anomaly Detection**: identifies unusual patterns of behaviour that could indicate a security issue.
    (b) **User Behaviour Baseline**: established normal behaviour patterns for users and devices, aiding in anomaly detection.
    (c) **Device Reputation**: assesses the safety and reputation of devices based on their behaviour and interactions.
13. **Threat Intelligence**
    (a) **Threat Reports**: incorporates external threat intelligence feeds to enrich the detection capabilities.
    (b) **Signature Updates**: regular updates to malware signatures and indicators of compromise (IOCs).
14. **Security Events**
    (a) **Alerts**: generated notifications of potential security issues, requiring investigation and response.
    (b) **Incident Logs**: detailed records of security incidents, including timestamps, affected systems, and actions taken.
15. **Compliance and Policy Enforcement**
    (a) **Policy Violations**: detected violations of organizational security policies.
    (b) **Audit Trails**: records of policy enforcement actions and compliance checks.
16. **Hardware Information**:
    (a) **Device Model**: the model and specifications of the hardware.
    (b) **Device Serial Number**: the unique identifier assigned to the device on the network.
    (c) **BIOS/UEFI**: information about the BIOS and UEFI firmware versions.

3.2 Threats

Threats represent all the previous types of cyberattacks that have happened on endpoints that are connected in a single network, therefore forming a site group in SentinelOne Console. All the threats that are listed have been detected by SentinelOne agents, therefore they no longer possess any danger to the environment. These threats are categorized on their behaviour and potential harm they cause to the endpoints. It is part of the SentinelOne capabilities to provide comprehensive visibility and control over the security of endpoints within an organization's network. Some key threat categories that SentinelOne addresses are:

1. **Malware**
    (a) **Virus**: self-replicating code designed to cause damage to a computer system.
    (b) **Trojan Horse**: malicious software disguised as legitimate software.
    (c) **Worms**: autonomous program that replicate themselves by modifying the host program.
    (d) **Spyware**: software designed to gather personal or confidential information without consent.
    (e) **Ransomware**: encrypts files on a victim's computer or network, demanding a ransom payment for the decryption key.

2. **Adware and PUPs (Potentially Unwanted Programs)**: are software that automatically delivers advertisements, often bundled with free software downloads. They are usually not malicious but unwanted by the user.
3. **Exploits**: are pieces of code specifically crafted to take advantage of a bug or vulnerability in software or hardware to gain unauthorized access or privileges. It can be broken down into 2 types:
   (a) **Zero-Day Attacks/Exploits**: attacks that exploit unknown vulnerabilities in software or hardware to those who should be interested in mitigating them, including developers and users.
   (b) **Known Vulnerabilities**: attacks that exploit known vulnerabilities in software or hardware that have been disclosed to the public, but have not been patched or fixed by the vendor.
4. **Phishing and Credential Theft**: involves sending fraudulent communications, usually e-mails, to trick recipients into revealing sensitive information, such as passwords and credit card numbers.
5. **Botnets**: are networks of compromised computers that are controlled by malicious actors, often being used to perform DDoS attacks, send spam, or steal data.
6. **Advanced Persistent Threats (APTs)**: are long-term targeted attacks on specific systems, often state-sponsored, aimed at stealing intellectual property or sensitive information.
7. **Fileless Malware**: malicious malware that do not rely on files and instead work directly in computer's memory and exploit existing legitimate tools or scripts on the system (e.g., PowerShell, WMI)
8. **Insider Threats**: come from individuals within an organization who have access to sensitive information, either intentionally or unintentionally causing harm.
9. **Network Attacks**:
   (a) **Man-in-the-Middle (MitM)**: an attack where the attacker secretly intercepts and possibly alters the communication between two parties.
   (b) **Denial-of-Service (DoS)**: an attack that attempts to make a machine or network resource unavailable to its intended users.
10. **Suspicious Activities**
    (a) **Suspicious Processes**: unusual or potentially harmful processes running on the system.
    (b) **Suspicious Network Traffic**: unusual outbound or inbound network traffic that may indicate malicious activities.
11. **Rootkits and Bootkits**: are malware designed to gain unauthorized root or administrative access to a system and hide its presence.
12. **Insider Threats**: are threats that come from individuals within an organization who have access to sensitive information, either intentionally or unintentionally causing harm.

The types of information regarding the threat that can be provided by SentinelOne for analysing and responding to these security incidents are the following:

1. **Threat Overview**
   (a) **Threat Name**: the name or type of the detected threat (e.g., Trojan, Ransomware, Phishing Attack).
   (b) **Detection Source**: the method or technology used to detect the threat (e.g., behavioural AI, signature-based detection, threat intelligence).
2. **Threat Details**

(a) **Severity Level**: the assigned severity of the threat (e.g., low, medium, high, critical).

(b) **Status**: the current status of the threat (e.g., detected, mitigated, undetected).

(c) **Detection Time**: the time at which the threat was detected.

3. **Threat Indicators**

   (a) **Indicators of Compromise (IOCs)**: specific attributes associated with the threat, such as file hashes, IP addresses, URLs, and domain names.

   (b) **Malicious File Details**: information about malicious files, including file path, file name, file hash, and file size.

4. **Threat Behaviour**

   (a) **Process Information**: details of the process involved in the threat, such as process names, IDs, parent processes, and command-line arguments.

   (b) **Execution Flow**: the sequence of actions taken by the threat, including process creation, file modifications, registry changes, and network connections.

   (c) **Behavioural Indicators**: behavioural patterns observed during the threat's execution, such as attempts to evade detection, persistence mechanism, and system modifications.

5. **Affected Endpoints**: endpoint details as was already described in (*1*)

6. **Mitigation and Response Actions**:

   (a) **Quarantine Actions**: information on whether the malicious file or process has been quarantined.

   (b) **Killed Processes**: details of any processes that were terminated as a part of the mitigation efforts.

   (c) **Rollback Actions**: if supported, information on any rollback actions taken to revert the system to its pre-infected state.

   (d) **Manual Interventions**: details of any manual actions taken by admins, such as isolating the endpoint or initiating a deep scan.

7. **Threat Timeline**

   (a) **Detection Events**: a chronological list of events related to the detection of the threat.

   (b) **Mitigation Events**: a timeline of mitigation actions taken by SentinelOne and admins.

8. **Threat Analysis**

   (a) **Threat Classification**: classification of threat based on its behaviour and characteristics (e.g., file-based, fileless, memory-based) as was listed above.

   (b) **Threat Intelligence**: additional context from threat intelligence sources, such as associations with known threat actors or campaigns.

9. **Clarification**

   (a) **Attack Storyline**: a visual representation of the threat lifecycle and its activities within the endpoint environment.

   (b) **Process Tree**: A visual depiction of the process hierarchy and how the threat interacted with other processes.

10. **Contextual Information**:

    (a) **Related Threats**: information about other threats that are similar or related to the detected threat.

(b) **Threat Evolution**: insights into how the threat has evolved over time and any changes in its behaviour or tactics.

Additionally in the API, threats can also be filtered from:

Analysis Verdict (performed by user, marking it as)	Threat Mitigation Status	Incident Status	A.I. Confidence level
• False positive • Suspicious • True positive • Undefined	• Mitigated • Not mitigated • Marked as benign	• Resolved • Unresolved • In progress	• Malicious • Suspicious • N/A

Table 3.1: Different filters that can be applied to the Incidents page #1

Engine	Cloud Provider	OS	Classification
• SentinelOne Cloud • On-Write Static A.I. - Suspicious • Behavioral A.I. • On-Write Static A.I. • Reputation • Cloud Detection • User-Defined Blocklist • Documents, Scripts • Anti Exploitation / Fileless • Intrusion Detection • Potentially unwanted application • Lateral Movement • Remote Shell • Manual • Application Control • Threat Intelligence • Watctower • Driver Blocking	• Azure • AWS • GCP • OCI • ESXi	• Windows • Linux • Mac • Windows Legacy	• Malware • PUA • Virus • Infostealer • Hacktool

Table 3.2: Different filters that can be applied to the Incidents page #2

Initiated by	Time	Free text search

• Agent Policy • Full Disk Scan • Local agent command • Deep Visibility Command • Management console API • On-Demand Scan • Custom Rule • Custom Alert • Cloud Detection • Threat Intelligence • Watctower	• Recent • Last 24 Hours • Today • Last 48 Hours • Last 7 Days • Last 30 Days • This Month • Last 2 Month • Last 3 Month • Last Year • Custom Range	• Content Hash • Cloud Account • Cloud Image • Cloud Instance ID • Cloud Instance Size • Cloud Location • Cloud Network • AWS Role • AWS Security Groups • AWS Subnet IDs • Azure Resource Group • GCP Service Account • Threat Details: e.g., ”This is a non-Microsoft binary that masquerades as a Microsoft executable” • File Path: e.g.,: ”\Device\HarddiskVolumeA\Users\Chris\Downloads\malicious.exe” • Endpoint Name: e.g., ”LT-Christopher”, ”LT 10-08” • UUID: e.g.,: ”0ff2a3409f284776a23432e9f6894afa” • Agent Version (at detection) e.g.,: ”23.1.4.650” • Agent Version (current) • Domain (at detection time) • Command Line Arguments • Initiated By (username): e.g., ”LuukAdmiraalMKBIT” • Storyline • Originated Process • Cluster Name • Node Name • Namespace Name • Namespace Labels • Controller Name • Controller Labels • Pod Name • Pod Labels • Container Name • Image Name • Container Labels • External Ticket ID • Node Labels

Table 3.3: Different filters that can be applied to the Incidents page #3

3.3 Ranger

Ranger in SentinelOne is an advanced network discovery and visibility feature designed to identify unmanaged and potentially vulnerable devices within an organization's network. It is showing information more about the network of the organization. It therefore extends SentinelOne's capabilities beyond endpoint protection, providing comprehensive visibility into the entire network environment. Some data that can be gathered are:

1. **Device Discovery**: information that are already discussed in (*1*).
2. **Network Visibility**:
   (a) **Network Segments**: information about the network segments or subnets where the devices are located.
   (b) **Connection Details**: information about the network connections, such as connected switches, routers, and access points.
   (c) **Communication Patterns**: details about the communication patterns of the device, including inbound and outbound connections.
3. **Device Inventory**:
   (a) **Firmware Version**: the version of the device's firmware, if applicable.
   (b) **Hardware Details**: information about the device's hardware, such as manufacturer, model, and serial number.
4. **Security Posture**:
   (a) **Managed vs. Unmanaged**: identification of whether device is managed (i.e., has the SentinelOne Agent installed). The data is in *boolean* format.
   (b) **Vulnerability Assessment**: details about known vulnerabilities associated with the device, based on its OS, firmware, or software.
5. **Policy Compliance**: information that are already discussed in (*10*).
6. **Device Behaviour**:
   (a) **Behavioural Analysis**: insights into the device's behaviour, including any suspicious or anomalous activities.
   (b) **Risk Assessment**: the risk level associated with the device based on its behaviour and security posture.
7. **Response Capabilities**:
   (a) **Mitigation Actions**: informa
   (b) **Notification Alerts**: alerts and notifications related to the device's security status, generated for any discovered vulnerabilities or suspicious activities associated with the device.
8. **Device Inventory Management**:
   (a) **Agent Deployment**: options to deploy SentinelOne agent to unmanaged devices directly from the console.
   (b) **Device Grouping**: ability to group devices based on criteria such as device type, location, or risk level for better management.
9. **Activity Logs**:
   (a) **Discovery Logs**: logs of discovery events, including timestamps and the nature of each event.
   (b) **Historical Data**: historical records of the device's presence on the network and any changes in its status.

10. **Reporting** :
    (a) **Reports and Dashboards**: customizable reports and dashboards that provide summaries and insights into the network's security posture and device inventory.
    (b) **Network Maps**: connection representations of the network, showing the relationships and connections between devices.

Unfortunately for this project scope, Q-ICT has turned off the Ranger feature, so this will be out of scope. But this document can then serve as a future blueprint in case they want to implement it later on the application.

3.4 Rogues

Rogues in SentinelOne is unmanaged devices or endpoints that have been detected on the network but do not have the SentinelOne Agent installed. Identifying rogue devices is crucial for maintaining network security and ensuring comprehensive endpoint coverage. The information shown about rogues are the following:

1. **Device Identification, Type and Classification**: containing the data that has been discussed in (*1*).
2. **Network Information**:
    (a) **Network Segment**: the network segment or subnet where the device is located.
    (b) **Connected Access Point**: information about the access point or switch the device is connected to.
3. **Discovery Details**:
    (a) **Discovery Date**: date and time the device was discovered on the network.
    (b) **Last Seen**: the last time the device was active on the network.
4. **Security Posture**:
    (a) **Risk Assessment**: an evaluation of the potential risk posed by the rogue device based on its behaviour and security posture.
    (b) **Vulnerability Detection**: information about any known vulnerabilities associated with the device's OS or software.
5. **Behavioural Analysis**:
    (a) **Suspicious Activities**: details of any suspicious activities or behaviours observed on the rogue device.
    (b) **Communication Patterns**: information about the device's communication patterns, including inbound and outbound connections.
6. **Policy Compliance and Violations**: information that are already discussed in (*10*).
7. **Mitigation and Response Actions**:
    (a) **Isolation Options**: options to isolate the device from the network to prevent potential threats.
    (b) **Agent Deployment**: options to deploy Agent to the rogue device for better management and security.
    (c) **Manual Actions**: records of any manual actions taken by admins, such as initiating a scan or quarantining the device.
8. **Reporting**: data that have been discussed in (*10*).
9. **Historical Data**

(a) **Past Incidents**: information on any previous incidents involving the rogue device.

(b) **Activity History**: historical activity logs for the device, showing patters over time.

Rogues are very closely related to Ranger, but the difference is that Ranger is more about the network itself, while Rogues are more about the devices. Because the Ranger functionality is turned off, this project scope will also not include Rogue data to the application.

3.5 Miscellaneous

This is the optional functionality provided by MGMT console, within it the user can see the following:

1. **Live Feed News**: the cybersecurity the latest news from SentinelOne.
2. **Customizable Notes**: customizable notes that can be added to the application by the users.

Chapter 4

Database Design

NoSQL Database

is a category of DB that provides a mechanism for storage and retrieval of data that is modelled in ways other than the tabular relations used in RDBMS. NoSQL DBs are typically designed to handle large volumes of unstructured or semi-structured data, such as JSON, XML, or binary objects, and they offer a flexible data model that can evolve over time. Both databases in Firebase, Firestore, and Real-time Database are NoSQL databases. Some characteristics of NoSQL DBs include:

- Data Model: NoSQL uses dynamic schema, which allows data to be inserted without having to define the schema first, while SQL uses a fixed schema to store data.
- Data Structure: NoSQL DBs use a variety of data structures such as key-value pairs, document-oriented, graphs databases, and columns-oriented; unlike SQL which only uses tables to store data.
- Querying: NoSQL uses variety of query languages, such as MongoDB query language or Cassandra's CQL, while SQL databases only use SQL as the unified language to query data.
- Data Consistency and ACID Compliance: databases that are ACID compliant means that they follow a set of rules to ensure that database transactions are processed reliably and securely. SQL databases are a good example of this, while NoSQL databases sacrifice some ACID properties to achieve higher performance and scalability.
- Use Case: NoSQL DBs are ideal for applications that need to handle large volumes of data and require high scalability and flexibility, such as social media platforms, real-time analytics, and content management systems.
- Scalability: NoSQL databases are traditionally designed for horizontal scalability, meaning they can easily scale out multiple servers or clusters to handle volumes of data and high throughput, which is more cost-effective and allows for better scalability. They are well-suited for distributed and cloud-based environments. SQL databases are traditionally designed for vertical scalability, meaning a single server is scaled up with more powerful hardware and power (CPU, RAM) to a single server. This can become expensive and limit scalability.

4.1 How Collection is Stored

Before diving into how the database is designed, a basic overview of the terminologies of Firestore is needed to understand the basis of the hierarchy of the database.

1. Collection: a group of documents that have the same structure. It is equivalent to a table in a relational database. Collections can contain multiple documents, and each document can contain multiple fields.

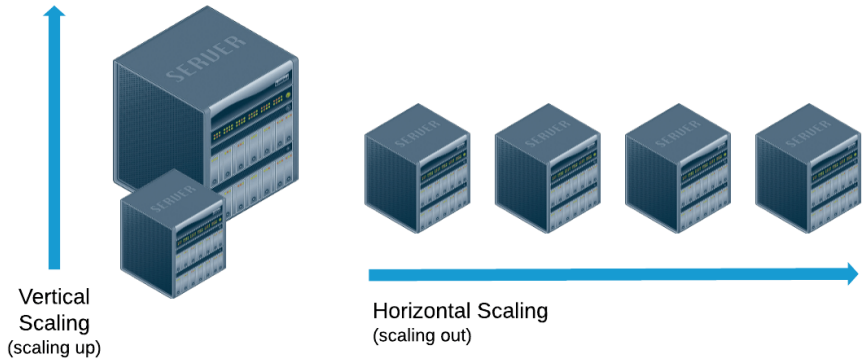


Figure 4.1: Vertical (SQL) VS Horizontal (NoSQL) Scaling

2. Document: refers to a single record in a collection. A document is essentially a JSON object stored in a collection. This JSON object can contain various fields, which are represented as key-value pairs. Each document in a collection can have a different structure. This is because Firestore is schema-less, meaning that it allows a flexible schema, and the data can be stored without any restrictions. Each document is identified by a unique ID, which is automatically generated by Firestore or can be set manually by the user. The document ID is used to uniquely identify the document in the collection.

3. Field: each document will then contain fields, which are essentially key-value pairs that hold the data for that document. Fields will have different types of data, such as the following:

   - String: the most common type of data. It can hold any text data, including numbers and special characters.
   - Number: can hold any numerical data. It can be integer or float.
   - Boolean: can hold 2 values which are true or false value.
   - Map: a collection of key-value pairs. It can hold any data type listed in this list.
   - Array: a collection of data of any type. It can hold any data type listed in this list.
   - null: the absence of any value.
   - Timestamp: a date and time value. It will be in the form of number of milliseconds since January 1, 1970 (Unix Epoch).
   - Geopoint: a geographical location.

Because Cloud Firestore is a NoSQL database, a different approach is needed to managing data compared to traditional SQL databases. It is built for automatic scaling, while still offering high performance and ease of application development. NoSQL is also schema-less, meaning that it allows a flexible schema, the data can be stored without defining the structure first. NoSQL databases are also scaled horizontally, by adding more servers to distribute the workload, instead of adding more power to a single server. Firestore NoSQL does in fact however, allow indexing to the database. There are 2 ways of how the author approach indexation for this project:

1. Document ID: is the usual form of indexation. Document IDs can be automatically generated by Firestore or generated manually by the user, keeping in mind that each document ID must be unique to each other.

2. Field ID: is the form of indexation that can be created by the user. The document itself is using a JSON format, and the user can create an index by making a specific field in the JSON document format, and then query it by making a filter to that field. This is useful for the author to store the user preferences in the Firestore database.

4.2 Data Types

- Endpoints: contains comprehensive set of information related to each endpoint (desktops, laptops, servers) that is being monitored by SentinelOne platform. This data will help administrators and security teams to manage, monitor, and respond to security events effectively.
- Threats: the list of malware that have been detected on endpoints that are connected to a specific network. The type of data that can be gathered are already discussed in 3.2
- Applications: contains information about outdated applications that can pose security risks to an endpoint according to the CVEs regularly posted by MITRE and NVD.
- Miscellaneous: contains SentinelOne news feed article and customizable notes.

4.3 Collection Types

There are 2 types of collection that are used throughout the project. These types were created by the author for the purpose of storing the required data for the application.

1. SentinelOne Data: containing the data from fetched from SentinelOne API, each unique to their respective Company Site ID. The data is further classified into types like what was discussed above (*4.2*). This collection will store its data in different way than in SQL. A new document will be created for each data model of the respective JSON response, and an indexation field will be created that will be based on the Site ID relative to the company. Firestore itself will determine the Document ID, to ensure that each document is unique.
2. User Preference: containing the data from each individual users about their choice of graphic types per widgets, types of data that wanted to be displayed, editing the widget title, table header columns customization, and pagination. This collection will store its data in the same way in SQL. Document IDs will be determined based on user ID, for the patency of Cloud Functions being able to determine which data from which user is being fetched. It is therefore easier to add, edit, delete fields of specific documents. The User Preference Collection also has a defined structure for each of its document, which mean they will contain the same fields that will be discussed later below.

4.4 Index Types

Moreover, indexes play a huge role in the database. Indexes are used to identify which data belongs to which, so that the Cloud Functions can know exactly what to fetch to a specific customer. Indexes also allow for faster search result if the request requires filtering of some data. There are 3 types of indexes that are used in the Firestore database:

- Site ID: is mainly used for indexing the SentinelOne data. The purpose of storing the site ID in the field of a document is to differentiate data across different organizations that are part of Q-ICT's SentinelOne environment (the customers). This separation of data is mainly has got to do with the fact that a customer from a specific organization cannot see SentinelOne data outside of their own organization.
- User ID: is mainly used for the collections that store the user preferences. The main purpose of this index is to further differentiate the data between users of the same organization as users will have different customization to their widgets between each other.
- Data ID: this indexation is used to further differentiate certain varieties of SentinelOne data in Firestore. This is because each data can contain an additional model data specific to that data ID. For example, each threat that have

happened have their own timeline about what processes were undertaken by SentinelOne to mitigate the threat. This will then cannot be generalized by just the Site ID, but instead need more specific indexation.

Making of User Preference Table

A special table in Firestore is created to store the user's preference, each document' ID is the user's ID, therefore ensuring uniqueness. In a document, there are 3 different fields:

- timestamp: containing date time in the format of timestamp. It is used as a reference for the Cloud Function to know when the data is last updated, and if it exceeds a certain threshold, the Cloud Function will fetch the data from the SentinelOne API again. This will be the field that the fetching cloud function needs to do its if-check first to determine if the data is outdated.
- widgetIds: containing an array of strings, each string is the ID of the widget that the user wants to see in the dashboard. This is used for the convenience of adding, updating, and deleting widgets.
- widgetTitle: containing also an array of strings, which serves as the title that the user chooses for their widgets. A widget title can have the same name as another widget, if the widget ID is different.
- widgets: containing a map of 3 different fields:
  - category: containing a string, which is the category of the widget. The category is used to group the widgets in the dashboard, so the user can easily find the widgets that they want to see. Currently, there are 3 categories: "Threats", "Endpoints", "Applications", and "Miscellaneous". In the future, additional 2 categories can be added: "Ranger (Network Discovery)" and "Rogues". Ranger is a feature that is used to discover the network of the client's machine, and Rogues is a feature that is used to detect any unauthorized devices that are connected to the client's machine. These 2 features are currently deactivated in the Q-ICT site environment for internal reasons, thus the author does not include them in the project.
  - graphicType: containing a string, which is the type of the graphic that the user wants to see in the widget. Currently, the graphics that the user can utilize are: "Doughnout", "Pie Chart", "Vertical Bar", "Stacked Vertical Bar", "Horizontal Bar", "Stacked Horizontal Bar", "Line Chart", "Scatter", "Bubble", "Pyramid", "Funnel", and "Table".
  - widgetType: Inside the 4 main categories, the selection is further divided into separate types according to which categories that wanted to be visualized. Agent category can show the status of the agents, such as the number of agents that are connected, disconnected, or in quarantine. Threats category can show the status of the threats that are detected by SentinelOne, such as the number of threats that are detected, resolved, or in quarantine. Applications category show the number of outdated applications that can potentially be a threat to the client's machine, according to the latest CVEs from the NIST and MITRE. The widget categorization is divided as follows:
    * Endpoint: Agent versions, pending updates, agent by OS, console migration status, domains, encrypted applications, endpoint connected to management, connection status, endpoint health, endpoint list, endpoint summary, installers, load saved filters, local configurations, location IDs, machine types, manual actions required, network health, OS arch, pending uninstall, and scan status.
    * Threats: Analyst verdicts, confidence levels, engines, external ticket exists, failed actions, incident status, initiator, mitigated preemptively, note exists, pending actions, reboot required, threat classification, threat list, threat status, threat summary of the week, and unresolved.
    * Applications: Risk levels, and most impactful applications according to SentinelOne Vulnerability Score.
    * Miscellaneous: SentinelOne news feed, and free text.

```
timestamp: 1717414859062
widgetIds
    0 "1"
    1 "52:*(bh5d2"
    2 "&?m*imbd9s"
    3 "?:!llp/#td"
    4 ")*rcx3f034"
widgetTitle
    0 "Date Time - Threats"
    1 "Risk Levels - Applications"
    2 "Agent Versions - Endpoints"
    3 "Live Feed - Miscellaneous"
    4 "Free Text - Miscellaneous"
```


Figure 4.2: User Preference document for a specific user will contain timestamp, widgetIds, and widgetTitle


```
widgets
    0
        category: "Threats"
        graphicType: "Line"
        widgetType: "Date Time"
    1
        category: "Applications"
        graphicType: "Doughnout"
        widgetType: "Risk Levels"
    2
        category: "Endpoints"
        graphicType: "Doughnout"
```


Figure 4.3: The widgets field of the userPreference document

Chapter 5

Back-end Architecture

JavaScript

JS is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a multi-paradigm programming language, supporting OOP, imperative, and functional programming styles and is primarily used for client-side web development. It is also used for server-side development with Node.js, and for mobile app development with frameworks like React Native, Vue.js, Angular.js, and NativeScript. JS is now one of the core technologies of the web, along with HTML and CSS, and is supported by all modern web browsers. Some key features of JS include:

- Client-side Scripting: it is mainly used for client-side scripting in web browsers, allowing programmers to create dynamic content that interacts with the browser and the user. It can manipulate the content and behavior of HTML elements, respond to user actions without the need to reload the page for interacting with DOM to update page content dynamically. This enhances the UX by making web applications feel more responsive and integrated.
- Cross-Platform Compactibility: it is supported by all modern web browsers, including Chrome, Firefox, Safari, Opera, and Edge, and can be used to build cross-platform web applications that run on any device or platform, like Android or iOS.
- Server-Side Scripting: with the advent of Node.js, it has also become popular for server-side scripting. Node.js is a runtime environment that allows developers to build scalable network applications, including web servers, APIs, and real-time applications like chat servers. This has expanded the versatility and use cases of JS beyond the browsers, making it a full-stack development language.
- Dynamic Typing: it means that variables do not have predetermined types. Instead, types are determined at runtime based on the assigned values.
- Functional Programming: it supports concepts such as first-class functions, high-order functions, and closures. This allows programmers to write more concise and expressive code by treating functions as first-class citizens.
- Event-Driven Programming: it follows an event-driven programming paradigm, where actions or events trigger specific functions or code execution, makes it well-suited for building interactive UI and handling user interaction.

Node.js

is an open-source, cross-platform, server-side runtime environment built on Chrome's V8 JS engine that allows programmers to run JS code outside a web browser, enabling the development of server-side and networking applications.

TypeScript

is an open-source programming language developed and maintained by Microsoft. It is a superset of JS, meaning that any valid JS code is also valid in TS. It adds optional static typing to JS, which allows developers to annotate their code with type information, catch errors early in the development, and improve the maintainability or large codebases. Some developers prefer using TS over JS for some of its key-features:

- Static Typing: it checks the types of variables at compile-time, which enable developers to specify the types of variables, functions parameters, and return values. This introduces type safety, in contrast to JS which is dynamically typed, which means that the types of variables are determined at runtime.
- Enums: it provides support for enums, allowing developers to define a set of named constants with associated values.
- Generics: it supports generics, enabling the creation of reusable components that can work with a variety of data types.
- Decorators: it supports decorators, which are a way to add annotations and a meta-programming syntax for class declarations and members. Decorators can be used to modify classes, methods, and properties at design time, providing a powerful way to extend or modify the behaviour of classes and their members.
- Interfaces and Classes: it supports OOP concepts such as classes, interfaces, inheritance, and access modifiers, making it easier to organize and structure code.
- ES6+ Features: it supports many features introduced in ECMAScript standards beyond ES5, such as arrow functions, classes, async/await syntax, and modules.
- Backwards Compatibility: it is designed to be backwards compatible with JS, which means that developers can use it alongside JS code and integrate it into existing JS projects and environments, including, browsers, Node.js, and other JS runtime environments.

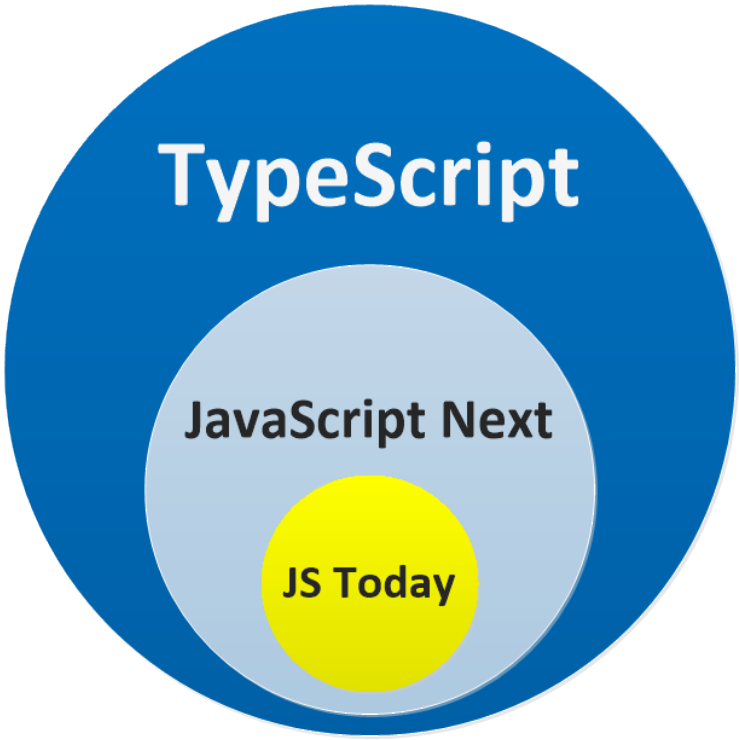


Figure 5.1: Venn diagram of TypeScript and JavaScript

As stated in the Research Report, Q-ICT wishes to utilize the 2nd generation of Firebase Cloud Functions, therefore a new Firebase project was created and stored on the cloud (*using Azure DevOps*). The author also needs to make a decision in regards on how the infrastructure of the codebase should be structured. Because Q-ICT is always critical and open to feedback, the author is given access to the old Firebase codebase (*utilizing version 1.0*), and find any potential upsides and downsides of that project repository. The author then, by an informed decision, is allowed to decide whether to structure the new codebase in the same way as the old one. The author has certainly decided to create some new adjustments to the new codebase. For example, instead of stacking all functionalities that a Firebase Cloud Function might have, the author has decided to improve the codebase especially regarding the separation of concerns. Specifically, the response from the API is modelled using Interfaces and Classes object within the Models directory, allowing for a consistent reuse across different functions. Additionally, distinct Routers and Controllers directories has been established, each with specific responsibilities. The Routers directory primarily handles external communications with the API, utilizing `Axios` (*npm, n.d.*) for handling the GET requests. The Controllers directory, contains the back-end logic of the cloud function, bridging Models and Routers and ensuring comprehensive error documentation. This structure defines the behavior of the Firebase Cloud Function version 2.0 of the QaaS app.

Moreover, there are Utilities and Middlewares directories, which are used to handle common functionalities and errors. The Utilities directory contains functions that are used across the codebase, such as logging errors, and the Middlewares directory contains functions that are used to handle the request and response of the cloud function. For example, the `logError` function in the Utilities directory is used to log errors in the console, and the `handleError` function in the Middlewares directory is used to handle errors in the response of the cloud function. This structure allows for a more organized and maintainable codebase.

5.0.1 Compliance with ESLint compiler

Additinally, the back-end project utilizes Node.js as the runtime environment, with TS as the programming language. This has caused longer compilation time as it involves additional steps like type checking and transpiling the code to JS. However, it is still believed to be the better practice as JS in nature is a loosely typed language, and it is easy to make mistakes in the code as its variables do not have a fixed type. While this sometimes can be beneficial as it makes the development faster and more flexible, it also introduces challenges, particularly in debugging and maintaining the code. To prevent this, both the author and the Company Supervisor have decided to use and adhere to TS and ESLint rules, to ensure static typing, code quality and consistency to the coding standards.

Listing 5.1: Example of Class and Interface defined in Models

```

export interface Customer {
  customerId: string;
  customerName: string;
  orgUnitType: string;
  parentId: string;
  city: string;
  stateProv: string;
  country: string;
  county: string | null;
  postalCode: string;
  contactEmail: string;
}

/**
 * Class for CustomerModel.
 */
```

```
18  export class CustomerModel {
19    private readonly firestore: FirebaseFirestore.Firestore;
20    private readonly collectionName = "customers";
21    private readonly context = "CustomerModel";
22
23    constructor() {
24      this.firestore = admin.firestore();
25    }
26
27    serializeCustomerToJson(customer: Customer): string {
28      return JSON.stringify(customer);
29    }
30
31    /**
32     * Deserializes a JSON string to a Customer object.
33     * @param {string} json The JSON string to deserialize.
34     * @return {Customer} The deserialized Customer object.
35     */
36    deserializeJsonToCustomer(json: string): Customer {
37      return JSON.parse(json) as Customer;
38    }
39
40    /**
41     * Creates a new customer in Firestore.
42     * @param {Customer} customer Customer to add to Firestore.
43     * @return {Promise<FirebaseFirestore.DocumentReference<FirebaseFirestore.DocumentData>> | undefined}.
44     */
45    async createCustomer(customer: Customer):
          Promise<FirebaseFirestore.DocumentReference<FirebaseFirestore.DocumentData> | undefined> {
46      try {
47        return this.firestore.collection(this.collectionName).add(customer);
48      } catch (ex: unknown) {
49        logError(ex, this.context + ".createCustomer");
50      }
51      return undefined;
52    }
53
54    /**
55     * Retrieves a customer by its ID from Firestore.
56     * @param {string} customerId The ID of the customer to retrieve.
57     * @return {Promise<Customer | undefined>} A promise that resolves with the customer object if found,
          or undefined if not found or
58     an error occurs.
59     */
60    async getCustomer(customerId: string): Promise<Customer | undefined> {
61      try {
62        const doc = await this.firestore.collection(this.collectionName).doc(customerId).get();
63        if (!doc.exists) {
64          return undefined;
65        }
66        return doc.data() as Customer;
67      } catch (ex: unknown) {
68        logError(ex, this.context + ".getCustomer");
69        return undefined;
70      }
71    }
72
73    /**
74     * Updates an existing customer in Firestore.
```

```
75     * @param {string} customerId The ID of the customer to update.
76     * @param {Partial<Customer>} customer The partial customer object containing updates.
77     * @return {Promise<void>} A promise that resolves when the update is complete.
78     */
79    async updateCustomer(customerId: string, customer: Partial<Customer>): Promise<void> {
80      try {
81        await this.firestore.collection(this.collectionName).doc(customerId).update(customer);
82      } catch (ex: unknown) {
83        logError(ex, this.context + ".updateCustomer");
84      }
85    }
86
87    /**
88     * Deletes a customer from Firestore.
89     * @param {string} customerId The ID of the customer to delete.
90     * @return {Promise<void>} A promise that resolves when the customer is successfully deleted.
91     */
92    async deleteCustomer(customerId: string): Promise<void> {
93      try {
94        await this.firestore.collection(this.collectionName).doc(customerId).delete();
95      } catch (ex: unknown) {
96        logError(ex, this.context + ".deleteCustomer");
97      }
98    }
99  }
```

Listing 5.2: An example of Firebase HTTP Request onCall Cloud Function version 1.0

```
1  import * as functions from "firebase-functions";
2  import * as admin from "firebase-admin";
3  import { Secrets } from "../Firebase/Secrets";
4  import { FirebaseCall } from "../Firebase/FirebaseCall";
5  import axios from 'axios';
6
7  export default functions
8    .region("europe-west1")
9    .https.onCall(async (data, context) => {
10       try {
11         // context.app will be undefined if the request doesn't include a valid
12         // App Check token.
13         if (context.app === undefined) {
14           throw new functions.https.HttpsError(
15             "failed-precondition",
16             "The function must be called from an App Check verified app."
17           );
18         }
19       } catch (error) {
20         console.error("An error occurred in the Firebase HTTP Request onCall Cloud Function: ", error);
21         throw new functions.https.HttpsError("internal", "An error occurred while processing the
               request.");
22       }
23 });
```

Listing 5.3: onCall Cloud Function version 2.0, where the parameters of the function is less and the way it handles the authorization is different

```
1  import axios from "axios";
```

```
import * as functions from "firebase-functions/v2";
import { CallableRequest } from "firebase-functions/v2/https";
import admin from "firebase-admin";

import { region, sentinelOneURL, sentinelOneApiVersion } from "../../../config";
import { logError } from "../../../Middleware/LogError";
import { getSecret } from "../../../Util/Secret";

export const getSentinelOneData = functions.https.onCall({ region: region }, async (context:
    CallableRequest<any>) => {
  try {
    // Authentication check
    if (!context.auth) {
      throw new functions.https.HttpsError("unauthenticated", "The function must be called while
          authenticated.");
    }
    const data = context.data;

    // // Checking if the request contains the necessary data
    if (!data || !data.siteId || !data.userId || !data.dataType) {
      throw new functions.https.HttpsError("invalid-argument", "The necessary requirement(s) are
          missing or invalid in your request.");
    }
  } catch(ex: unknown) {

  }
});
```

Listing 5.4: LogError functionality in the Utilities folder

```
export function logError(error: unknown, context: string): void {
  if (isAxiosError(error)) {
    // AxiosError object, log detailed request and response information
    console.error(`[${context}] Axios error occurred: ${error.message}`);
    console.error(`Response data: ${error.response?.data}`);
    console.error(`Status code: ${error.response?.status}`);
    console.error(`Headers: ${JSON.stringify(error.response?.headers, null, 2)}`);
  } else if (error instanceof Error) {
    // Standard Error object, log message and stack
    console.error(`[${context}] An error occurred: ${error.message} \n Stack: ${error.stack}`);
  } else {
    // Non-Error object, log with a generic message
    console.error(`[${context}] An unknown error occurred:`, error);
  }
}
```

Listing 5.5: An example of how that utility is used in the Cloud Function, the reason why exception type is unknown is to comply with the ESLint rules that does not recommend to use the any type to the variables

```
try {

}
catch (ex: unknown) {
  console.log(`An error occured in the getFirestoreData function: ${ex}`);
  logError(ex, "getFirestoreData");
} finally {
```

```
  throw new functions.https.HttpsError("internal", "An error occurred while getting the data in
      Firestore.");
}
```

Listing 5.6: Example of challenges in JavaScript because of its loosely typed nature that creates lack of type safety, type coercion, and type conversion

```
function add(a, b) {
 return a + b;
}
console.log(add(5, '2')); // "52" instead of 7
```

Listing 5.7: Another example of JavaScript challenges

```
C:\Users\ChristopherSulistiyo>node
Welcome to Node.js v22.1.0.
Type ".help" for more information.
> true === 1
false
> true + true + true === 3
true
> 0 == "0"
true
> "0" ==- null
true
> -null
-0
> -0 == "0"
true
> parseInt(0.00000005)
5
> typeof(NaN);
'number'

> let a = []
undefined
> a.length == a
true
```

5.1 SentinelOne NPM packages

NPM packages are a necessary component when developing a project in JavaScript, as they are used for managing dependencies in Node.js projects. Below are the NPM packages that are used in the project:

- `axios`: Used for making HTTP requests from the Firebase Cloud Functions to the SentinelOne API.
- `firebase-admin`: Used for interacting with the Firebase services, such as Firestore and Secret Manager.
- `firebase-functions`: Used for creating Firebase Cloud Functions. Version 2.0 is used in the project. Some differences between version 1.0 and 2.0 are the way the functions are defined, the way the functions handle the request and response, and the way the functions handle the authentication. This package has many types of functions, and the ones that are used are: CallableRequest, onRequest, onSchedule, and onCall.
- `algoliasearch`: Used for integrating the Algolia search engine with the Firebase Cloud Functions. Algolia is used to make the search faster and more accurate.
- `firebase-functions-test`: Used for testing the Firebase Cloud Functions. The package provides a way to test the functions locally before deploying them to the Firebase Cloud.

5.2 Downloading PDF and HTML report files from SentinelOne to the client

At almost the end of the realization phase, based on a work item requested by the stakeholders on the SCRUM board, the author has decided to include showing the reports that have been made automatically or manually by SentinelOne or the cybersecurity actors on Q-ICT's SentinelOne environment. The process initially was quite low on the priority list (Could have on the MOSCOW), SentinelOne already has an API that will give the ASCII characters of the PDF, but calling this directly on the client itself is not possible due to CORS blocking policy. Therefore, a Cloud Function is needed as a middleman for getting the PDF file to the client.

First on the back-end, when using `axios` to call the appropriate SentinelOne URL, another header needs to be included to specify that the expected response format is PDF. Additionally, the response type also need to be set to `arraybuffer` to hand;e binary data efficiently.

Lastly, upon receiving the request, the binary data is converted into a base64-encoded string. This is necessary to ensure that the binary data can be safely transmitted and displayed within the web client. Additionally, the content of the MIME type of the response is extracted from the headers to facilitate proper handling of the data on the client side.

When the client accepts the response, it is necessary the base64-encoded string is decoded into a byte of array. Then, to facilitate the download of the PDF, a `Blob` is created from the decoded byte array. An object URL is then generated from this `Blob`, and an anchor element is programmatically created and clicked to trigger the download.

This way, a PDF or HTML file can be downloaded on the client side using any type of web browser when running the QaaS app.

```

class Data extends BaseModel {
  Data({
    required this.id,
    required this.name,
    required this.age,
  });

  factory Data.fromJson(Map<String, dynamic> json) {
    return Data(
      id: json['id'] as String,
      name: json['name'] as String,
      age: json['age'] as int,
    );
  }

  final String id;
  final String name;
  final int age;

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'name': name,
```

```
25          'age': age,
26        };
27      }
28    }
29
30    static Future<List<Data>> getData() async {
31      try {
32        final HttpsCallable callable =
33            FirebaseFunctions.instanceFor(region: 'europe-west3').httpsCallable(
34          'getData',
35          options: HttpsCallableOptions(
36            timeout: const Duration(seconds: 540),
37          ),
38        );
39
40        final Map<String, dynamic> headers = <String, dynamic>{
41          'dataType': 'abcd',
42        };
43
44        final HttpsCallableResult<dynamic> results = await callable.call(headers);
45        final Map<String, dynamic> responseData =
46            results.data as Map<String, dynamic>;
47        final dynamic data = responseData['data'];
48
49        if (data is List<dynamic>) {
50          return data
51              .map((dynamic json) =>
52                  Data.fromJson(json as Map<String, dynamic>))
53              .toList();
54        } else if (data is Map<String, dynamic>) {
55          return <Data>[Data.fromJson(data)];
56        }
57      } on FirebaseFunctionsException catch(e) {
58        if (kDebugMode) {
59          print('Firebase error when fetching the data: $e');
60        }
61      } catch (e, stack) {
62        if (kDebugMode) {
63          print('Application error: $e \nStack: $stack');
64        }
65      }
66      return List<Data>.empty();
67    }
```

Listing 5.8: How calling and modelling a response works in Flutter

Listing 5.9: How downloading a PDF/HTML file works on the back-end

```
const response = await axios.get(urlName, {
  headers: {
    "Authorization": `ApiToken ${apiToken}`,
    "Accept": "application/pdf",
  },
  responseType: "arraybuffer",
});

const answer = {
  data: Buffer.from(response.data, "binary").toString("base64"),
  contentType: response.headers["content-type"],
};

return { data: answer };
```

```
static Future<void> downloadPdfReports({
  required String reportId,
}) async {
  try {
    final HttpsCallable callable =
        FirebaseFunctions.instanceFor(region: 'europe-west3').httpsCallable(
      'getReports',
      options: HttpsCallableOptions(
        timeout: const Duration(seconds: 540),
      ),
    );

    final Map<String, dynamic> headers = <String, dynamic>{
      'reportId': reportId,
    };

    final HttpsCallableResult<dynamic> results = await callable.call(headers);
    final Map<String, dynamic> responseData =
        results.data as Map<String, dynamic>;
    final String base64Data = responseData['data'] as String;

    // Decode the base64 string to bytes
    final Uint8List bytes = base64Decode(base64Data);

    // Create a Blob from the byte data
    final html.Blob blob = html.Blob([bytes], 'application/pdf');

    // Create an object URL from the Blob
    final String url = html.Url.createObjectUrlFromBlob(blob);

    // Create an anchor element and trigger a download
    html.AnchorElement(href: url)
      ..setAttribute('download', 'report.pdf')
```

```
34          ..click();
35
36        // Optionally revoke the Blob URL after some time to release resources
37        Future<void>.delayed(
38            const Duration(seconds: 5), () => html.Url.revokeObjectUrl(url));
39
40        if (kDebugMode) {
41          print('PDF download triggered');
42        }
43      } on FirebaseFunctionsException catch (ex) {
44        if (kDebugMode) {
45          print('Firebase error when downloading the PDF: $ex');
46        }
47      } catch (e, stack) {
48        if (kDebugMode) {
49          print('Application error: $e \nStack: $stack');
50        }
51      }
52    }
```

Listing 5.10: How downloading a PDF/HTML file works on the front-end

Chapter 6

Front-End Architecture

The front-end architecture is a critical aspect of the QaaS App, ensuring that the UI is intuitive, responsive, user-friendly and secure. The front-end is responsible for handling user interactions, displaying data, and communicating with the back-end services. This chapter outlines the front-end architecture, including the technologies used, structure, design patterns, design principles and key components used to develop the client-side application, primarily using Flutter.

As was said before, the front-end of the QaaS App is built using Flutter, a modern UI framework that enables the development of high-performance applications for multiple platforms into a single codebase. The architecture in this project follows the MVC (Mode-View-Controller) architectural pattern to separate concerns and improve maintainability, scalability, and testability. The front-end is responsible for handling user interactions, displaying data, and communicating with the back-end services.

6.1 Flutter

Flutter is an open-source framework made by Google in 2017. It used as an UI toolkit for building natively compiled applications for mobile, web, and desktop (Windows, macOS, Linux) from a single codebase.

Widgets

Widgets in Flutter refer to the basic building blocks used to build UIs. Everything in Flutter, from buttons to layout containers to text inputs, is a widget. They are lightweight, reusable components that define the structure, appearance, and behavior of UI elements in a Flutter application. When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description to determine the minimal changes for rendering in the UI. There are only 2 types of Widgets in Flutter:

- **Stateless Widget**: is immutable, meaning that their properties cannot change once they are initialized. They are used for static content that does not require user interaction. Examples of a stateless widget would be text labels, buttons, icons, and static images.
- **Stateful Widget**: it maintains state, allowing their properties to change over time in response to user interactions or other events. They are used for dynamic content that require updates, such as animations, form inputs, and interactive elements. Stateful widgets consist of two classes: the widget class itself and a corresponding state class that manages the widget's state. Examples of stateful widgets are: text input fields, scrollable lists, tabs, and progress indicators.

- **Inherited Widgets**: facilitate state management by allowing state to be passed down the widget tree efficiently. This is particular useful for sharing state across many child widgets.

Flutter apps are built using a hierarchy of nested widgets, forming a ”widget tree”. The root widget defines the overall app structure, and other widgets branch out to create more complex UIs.

```
// Stateless Widget
class MyTextWidget extends StatelessWidget {
	final String text;

	MyTextWidget({Key key, this.text}) : super(key: key);

	@override
	Widget build(BuildContext context) {
	return Text(text);
	}
}

// Stateful Widget
class MyCheckbox extends StatefulWidget {
	@override
	_MyCheckboxState createState() => _MyCheckboxState();
}

class _MyCheckboxState extends State<MyCheckbox> {
	bool isChecked = false;

	@override
	Widget build(BuildContext context) {
		return Checkbox(
			value: isChecked,
			onChanged: (bool value) {
				setState(() {
				isChecked = value;
			});
		});
}
}
```

Listing 6.1: Example of a stateful and stateless widget

6.2 Architecture Components

6.2.1 State Management

Effective state management is crucial for building responsive and performant applications. The following state management solutions are considered when making this project:

- Provider: a wrapper around Inherited Widgets, providing a simple way to manage and access state throughout the application.
- Riverpod: an improvement over Provider, offering a more robust and testable state management solution.
- Bloc (Business Logic Component): a reactive state management library that uses streams to manage state changes, ensuring a clear separation between business logic and UI.

6.2.2 Routing and Navigation

Flutter provides a robust navigation system to manage routes within the application. The following approaches are used for routing:

- Pushing the stack: navigating to a new screen by pushing it onto the navigation stack. It is a traditional imperative approach for managing navigation stack and transitions between screens.
- Pop and push: navigating to a new screen by popping the current screen and pushing it onto the navigation stack. It is a declarative approach offering more control over the navigation stack, suitable for complex routing scenarios.
- Navigation with arguments: passing arguments to the next screen.

6.2.3 UI/UX Design Principles

- Responsive Design: ensure the application layout adapts to different screen sizes and orientations using Flutter's layout widgets such as `Flexible`, `Expanded`, and `MediaQuery`.
- Material Design: follow Google's Material Design guidelines for consistent and intuitive user interfaces and experience. Use Flutter's built-in Material components.
- Accessibility: ensure the application is accessible to all users and meet accessibility standards. Use Flutter's accessibility features such as `Semantics` and `FocusNode`. Features included are screen reader support, high-contrast themes, and accessible navigation to ensure the app is usable by all users.

6.2.4 Security

- HTTPS: ensure all data communication between the client and server is encrypted using HTTTPS whether it is at transit or at rest. This will be mainly used when talking to the Firebase Cloud Functions.
- CORS: use Cross-Origin Resource Sharing to enable cross-origin resource sharing. This is important when the app is deployed on a web server.
- Input Validation: validate all user inputs on the client side to prevent common security vulnerabilities such as cross-site scripting (XSS), and data tampering and to ensure data integrity.

6.2.5 Performance Optimization

- Lazy Loading: implement lazy loading of data in Flutter apps by using the `FutureBuilder` widget. This will ensure that data is loaded only when needed, improving app performance and reducing resource consumption. It improves initial load times and reduce memory usage when loading data-heavy components. A spinning MKBiT logo has been chosen as the loading indicator.
- Efficient Rendering: use Flutter's built-in performance profiling tools to identify and optimize rendering bottlenecks.

- Asynchronous Operations: leverage Dart's asynchronous single-threaded programming features to perform network requests and other long-running operations without blocking the UI.

The front-end architecture of the Qaas App is designed to provide a responsive, secure, and maintainable UI. By leveraging Flutter's powerful UI framework and following the MVC architectural patterns, the architecture ensures a clear separation of concerns, efficient state management, and an intuitive UI. This foundation will support the seamless integration of SentinelOne EDR and provide clients with transparent view of their device security information, and providing a clear pathway in the development of future SentinelOne modules that are yet to be integrated.

Chapter 7

Test Strategy

This test strategy outlines the approach for system testing the integration of SentinelOne EDR with the QaaS App. The goal is to ensure that the system functions as expected, meets the specified requirements, and provides a reliable and secure experience for users.

7.1 Test Objectives

The primary objectives of system testing are to:

- Verify that the integration with SentinelOne API functions correctly.
- Ensure that data is accurately fetched, displayed, and updated.
- Validate user interactions and interface elements.
- Confirm that security measures are effective.

7.2 Test Scope

System tests will cover the following components:

1. Device Health and Status Dashboard.
2. Audit Trail and Threat Information.
3. Client Management.
4. Integration with SentinelOne API.
5. Database interactions with Firebase Firestore.
6. Front-end components developed using Flutter.

7.3 Test Types

7.3.1 Functional Testing

Functional tests will verify that each feature works according to the specified requirements.

- Dashboard: verify the display of device health and status information.

- Audit Trail: check the accuracy and completeness of the threat audit trail.
- Client Management: validate the addition, update, and removal of clients and devices.
- API Integration: ensure data is correctly fetched and processed from SentinelOne API.

7.3.2 Performance Testing

Performance tests will assess the system's responsiveness and stabilitty under load.

- Load Testing: test the system's responsiveness under various load conditions by simulating multiple users accessing the system simultaneously to evaluate performance especially in the database and cloud functions.
- Stress Testing: identify the system's breaking point by gradually increasing the load beyond normal usage levels.
- Scalability Testing: evaluate the system's ability to scale horizontally by adding more clients and devices to the system and ensuring the system can scale horizontally to handle increased load.

7.3.3 Usability Testing

Usability test will assess the UX and interface design of the system.

- UI: ensure the UI is intuitive and easy to navigate.
- Accessibility: test the system's accessibility features such as screen reader support and high-contrast themes. Verify that the application meets accessibility standards.
- Responsiveness: ensure the application functions correctly on various devices and screen sizes.

7.4 Test Environment

The test environment will mimic the production environment closely as possible, including:

- Backend: Firebase Cloud Functions and Firebase Firestore.
- Frontend: Flutter application running mostly on web.
- Integration: SentinelOne API.
- User Devices: various desktop browsers, mobile smartphone devices, and tabloid devices.

7.5 Test Schedule

- Planning and Preparation: 1 week.
- Development of Test Cases: 1 week.
- Execution of Tests: 2 weeks.
- Bug Fixing and Retesting: 3-4 weeks.
- Final Verification and Reporting: 1 week.

Please also note that during the development of this project, the author only had access to the test environment of both the server and the client. This means that the author only has limited access to the data that is supposed to be shown and even less permission access to the SentinelOne API like POST, PUT, PATCH, and DELETE.

Chapter 8

Security Considerations and Author's Remark

8.1 Structure of the front-end codebase

There are some issues that the author has noticed with the way the QaaS app code was structured. For example, when assigning modules to a user, sometimes the user can see the modules that are not assigned to them. This may due to the fact that the code is not structured properly, and that they were some bugs left unchecked during the first development of the QaaS app. According to the Company Supervisor, the QaaS app itself is still young, so there are many rooms for improvement, especially regarding the structure of the codebase and the lack of utilization of software design patterns. The author that there should be a separate project, intended directly on refactoring the QaaS app, to make it more robust and scalable in the future.

Furthermore, there are some inconsistencies in the naming of the variables, especially regarding user tags and user roles. They are all doing the same thing, but the naming is different. This due the fact that this project was initially a continuation from an unfinished project, and the current developer did not bother to change the naming of the variables. This inconsistency can lead to confusion for the future developers, and it is recommended to change the naming of the variables to be more consistent.

8.2 Translation

The target audience of this project is Q-ICT's customers, in which all of them are a Dutch company. Therefore, the author and the Company Supervisor have tried to make the SentinelOne page in the QaaS to be in Dutch with the text buttons and the table headers, and the general UI widgets. However, as SentinelOne itself is an American company, the data that is fetched from the API is in English. The author and the Company Supervisor have not yet explored the idea of using an external package in the back-end that can connect to a translation party, like Google Translate or Microsoft Translator API, to translate the data. This feature, as the stakeholders have mentioned, would be a nice addition to have, as most of the clients do not know any technical knowledge of Cybersecurity, and having the data in their native language would make it easier for them to better understand the data.

8.3 Future modelling

The modelling in both the back-end and front-end are done in a way that is suitable for JSON response from the API. However, the author has not yet explored the idea of making the modelling to be more future-proof, by making the

modelling to be more dynamic, by adding other schemas such as XML, YAML, or CSV. This way, the QaaS app can be more flexible in the future, and in case SentinelOne wants to also expand their API server, for example by adding a SOAP server that returns XML response.

Appendix C

Deployment Manual

Qaas app deployment manual

This document is intended as a guide to deploy the Qaas app to the live environment.

Initial Steps:

1. Switch the firebase options to the live environment in the flutter new client
2. Make sure there are no debug app check tokens in the index.html
3. Build the flutter client
4. Deploy the new flutter client to firebase, now the frontend is updated
5. Add / update / delete any firebase cloud functions for the new version into the live environment
6. Create API secrets in the live environment if needed
7. Allow secret access on the specific cloud functions
8. Make sure the firebase security rules are up to date in the live environment
9. Mirror cloud Firestore data in Algolia if needed.
10. Change allowed Algolia indexes in the Algolia API key functions, this decides which indexes youre allowed to search on with this specific Algolia search API key.
11. Build Firestore indexes if needed, for where / order by queries. This is usually done before client deployment

Client code testing on the live environment

1. To test the client code on the live environment, create an App check debug token and place it in the index.html file. The path to the index file: web\index.html When going back to the test environment make sure to put the App check debug token back.
2. Proceed with doing step 2 and 3 from **Client code deployment to live environment**

Building Firestore indexes

If you do not use indexes, you can skip this step.

- Building Firestore indexes is something I recommend doing when the client code is connected to the live environment. If you have a try catch block around your Firestore query, it will just give you the link to build the needed index.

- Firestore indexes only need to be build when you use orderBy within your query and some other scenarios. Indexes you can see as a copy of that collection where they sort the data beforehand. This is for performance reasons.

Secret manager deployment (API keys)

1. Login to the Google Cloud console and make sure the right environment is selected **QaaS-Live_omgeving-V2** https://console.cloud.google.com/+
2. Navigate to **Secret Manager** you can do this by typing secret manager in the search bar. At this page you see an overview of all the secrets that the selected environment has access to.
3. Click **Create Secret**
4. Give the secret a name and fill in the secret value. The secret value could be a password, API key. There are cases where usernames are also defined here. The reason for this is that usernames then can be changed without redeploying the whole code base.
5. Make sure to check **Manually manage locations for this secret** and then select **Europe-west3** be sure to always deploy resources within Europe.
6. When you are done click **Create secret** in the bottom of the page to save the secret.

Adding a new secret version (API keys)

1. Follow step 1 to 2 from **Secret manager deployment (API keys)**
2. On the Secret manager page click the action button on the secret you desire to add a new version on. This is an icon with triple dots

3. Then click on **Add new version**, this will open a dialog where you can fill in the new secret value.
4. Make sure you Disable all past versions of the secrets.
5. Click **Add new version** to save the new version

Cloud function code deployment

1. Make sure you are in the root of the Cloud function directory
2. Login to Firebase through the visual studio code terminal. First make sure you are in the root project folder. Then type **Firebase login**, this will either make sure you are logged in to the right account already or you can log in to the correct Firebase project. If you are already logged into a different project, type **Firebase logout** to logout and then login to the correct project.
3. Make sure you are connected to the correct project through the terminal by using the **Firebase use** command. This will list the current active project. If you want to switch use **Firebase use <project id>** for example **Firebase use qaas-live-omgeving-v2**.
4. Use **firebase deploy --only functions:<functionName>** to deploy a specific function. **Firebase deploy --only functions** would deploy redeploy all the functions, but I would not recommend this. Only do this if necessary and you have a good understanding of what all the functions do.
5. Open the Firebase project in the browser https://console.firebase.google.com/project/qaas-live-omgeving-v2/functions and navigate to the Cloud functions tab. If deployed correctly the function is visible here.
6. Hover over the function and on the right should be an action button visible. Here you can view the **logs** or view the **detailed usage stats**. Any console.log will appear in the logs of the function.
7. In case you have **Secrets** that the cloud function is using from the secret manager, you need to assign these secrets to the cloud function. You do this by going to the **detailed usage stats** of the cloud function. Then on the top you click **Edit**. You will see **Runtime, build, connnections and security settings** click this open. Now you see multiple tabs. Click on **Security And Image Repo**. Here you can add **Secrets** to the cloud function. If you don't do this, you will get permission errors shown in the console logs. Make sure to click next and then click on deploy again, this deploys the functions with access to the assigned secrets. I wonder if there is a better way of doing this in one go. But it does add a security layer where you must do a specific action to grand access.

Cloud Firestore rules deployment

1. Open the firebase web console https://console.firebase.google.com/
2. Make sure you selected the correct environment **QaaS-Live-omgeving-V2**
3. In the Firebase project navigate to **Firestore Database**
4. Click on **Rules** you can find this in the tabs on the top.
5. Here you see all the security rules that are applied on the **Cloud Firestore** of the selected environment. These rules determine who can Read, Write, Update, Delete data from the cloud Firestore. This can be defined for each specific collection.
6. If you want to make change you will probably copy the rules from the test environment into this after a good review. So, you tested your change already in the test environment.
7. When you hit ctrl + s you save the rules, and they will be deployed to the live environment. Beware that it can take several minutes before cloud Firestore rules take effect. The previous rules are still there in the version control on the left of the screen in case we made a mistake, and we need to revert the made changes.

Cloud storage rules deployment

The cloud storage security rules are quite like the Firestore security rules. Most steps are the same.

1. Open the firebase web console https://console.firebase.google.com/
2. Make sure you selected the correct environment **QaaS-Live-omgeving-V2**
3. In the Firebase project navigate to **Storage**
4. Click on **Rules** you can find this in the tabs on the top.
5. Here you see all the security rules that are applied on the **Storage** of the selected environment. These rules determine who can Read, Write, Update, Delete data from the cloud storage. This can be defined for each specific path or document.
6. If you want to make change you will probably copy the rules from the test environment into this after a good review. So, you tested your change already in the test environment.
7. When you hit ctrl + s you save the rules, and they will be deployed to the live environment. Beware that it can take several minutes before cloud Firestore rules.
8. The Cloud storage makes use of custom claims mostly for access. But when deploying these rules, you already should be aware of what these are. If this is not known, here is some information

https://firebase.google.com/docs/auth/admin/custom-claims custom claims are defined for each user when their account is created or updated. This is done in a cloud function.

Extension deployment (Algolia)

1. Open the firebase web console https://console.firebase.google.com/
2. Make sure you selected the correct environment **QaaS-Live-omgeving-V2**
3. In the Firebase project navigate to **Extensions**
4. Click Explore extensions
5. Search for **Search Firestore with Algolia** in this example and click on install
6. Select the correct environment to install the extension on **QaaS-Live-omgeving-V2**
7. Name the extension
8. Fill in the collection path. So, for example you have a collection named **User**
9. You could specify the indexable fields but usually we keep this empty and index all
10. Name the Algolia index name equal to the collection name in Firebase, so in this example it would be **User.**
11. Retrieve the Algolia Application Id
    a. Go to https://www.algolia.com/ and sign in
    b. Make sure you select the correct environment **Qaas Live omgeving**
    c. Click on settings on the bottom left of the page
    d. Click API keys
    e. On the top of the screen the Application ID is visible, copy and paste this into the Firebase extension page.
12. Retrieve the Algolia API key "We recommend creating a new API key with "addObject", "deleteObject", "listIndexes", "deleteIndex", "editSettings", and "settings" permissions. **Do not use the Admin API key**."
    a. You can find this in the same place as the Application ID. But instead navigate to All API keys
    b. Here is an API key with the name **Firebase live omgeving api key**, use this key
13. Optionally you can select Yes on Full index existing documents. But this is only if you already have data in the firestore collection you want to install Algolia on.
14. Make sure you select **Europe-west3** as Cloud function location
15. Click **Install extension** to start installing the extension, this will take several minutes.

Algolia commonly used settings

1. Make sure you have selected the correct Algolia environment **Qaas live omgeving**

2. On the overview page scroll down to the indexes. Only indexes with data will be shown here. So, if the Firestore collection is empty. So will be the index here. Algolia mirrors the Firestore collection.
3. Click on your desired index / collection
4. Click on Configuration
5. Add searchable attributes, so if you want to search on first name only. You add first name here as searchable attribute.
6. Tip: is searching not working in the QaaS app? Go to the browse tab within Algolia on the desired index / collection and use the search bar there for testing. If the result is found there. You know your Algolia settings are at least setup correctly.

Client code deployment to preview channels

Preview channels are used for testing new live versions. Preview channels get their own separate links. They also have an expire time which can be set in the command. When deploying a preview channel, the existing live version will not change.

1. First make sure you do step 1 to 6 from **Client code deployment to live environment**
2. To deploy the client code to a preview channel make use of the following command **firebase hosting:channel:deploy QaaSv0.8 --expires 7d** "QaaSv0.8" is the name of the preview channel, use this for tracking the channel. "—expires 7d" means it will expire in 7d and it will automatically be deleted after that.
3. You will now receive a link where you can access the newly deployed version on the preview channel.
4. To make authentication work on the preview channel you need to add the url to list of allowed urls that can make use of the public API key: https://console.cloud.google.com/apis/credentials?project=qaas-live-omgeving-v2 you need to change this on the **Browser key (auto created by Firebase)** make sure to save your changes
5. You also need to add the url to Google reCAPTCHA https://www.google.com/recaptcha/admin/site/483217617/settings without this the MFA would not be working correctly. It can take some time before these changes take effect. Wait a couple of minutes. Also make sure to do ctrl + f5 to be sure it's not a cache problem.
6. After these steps you should be able to login to the Qaas app on the preview channel
7. If everything is working correctly and you want to deploy the preview channel to the current live channel, all you have to do is **firebase hosting:clone qaas-live-omgeving-v2:QaaSv0.8 qaas-live-omgeving-v2:live**

Client code deployment to live environment

Note: this is **ONLY** client code deployment. This does **NOT** include Cloud functions code, Secrets from the secret manager, Cloud firestore rules, any extensions, Algolia settings, Building firestore indexes, Cloud storage rules, etc.

1. Make sure there is no FIREBASE_APPCHECK_DEBUG_TOKEN in the index.html this looks something like this: <script>self.FIREBASE_APPCHECK_DEBUG_TOKEN = "the token is here";</script>. Remove the whole line including the script tags, be sure to save the document. The path to the index file: web\index.html
2. Check which environment the client is connected to. You can do this by looking at the firebase_options.dart file located at: lib\firebase_options.dart. **ProjectId: 'afstudeerapp-6df3b'**, means its connected to the test environment. The live environment is **Projectid: 'qaas-live-omgeving-v2'**
3. If you want to switch environments in the firebase_options file does one of the following commands **flutterfire configure -p afstudeerapp-6df3b** or flutterfire **configure -p qaas-live-omgeving-v2**
4. Login to Firebase through the visual studio code terminal. First make sure you are in the root project folder. Then type **Firebase login**, this will either make sure you are logged in to the right account already or you can log in to the correct Firebase project. If you are already logged into a different project, type **Firebase logout** to logout and then login to the correct project.
5. Make sure you are connected to the correct project through the terminal by using the **Firebase use** command. This will list the current active project. If you want to switch use **Firebase use <project id>** for example **Firebase use qaas-live-omgeving-v2**.
6. Run **flutter build web** to prepare the client code for deployment.
7. **This next move will deploy the client code to the live environment** which is used actively by users. Run **Firebase deploy** to deploy to the live environment. The **Firebase use** command from before made sure we are deploying this to the correct environment.

Appendix D

Project Plan

Project Plan

Christopher Sulistiyo

Quality ICT B.V.

Christopher.sulistiyo@student.nhlstenden.com

ICT & IC Information Technology Department Emmen

Version 1.1 - 21/02/2024

NHL STENDEN university of applied sciences

Chapter 1

Introduction

This document provides a detailed description of the project planning, research, and execution of the author's graduation work placement in the client company, Q-ICT.

The graduation work placement will be executed on the behalf of NHL Stenden University of Applied Sciences, which is based in the north of the Netherlands and offers a large variety of courses in both Dutch and English. The schooling has a heavy emphasis on group and practical project-based work, supported by a thorough theoretical education.

This document is created as one of the necessary requirements to submit to the School Supervisor within the first 2 weeks after the author started his graduation project. This document represents one of the essential prerequisites for the author's successful completion of the graduation work placement project of NHL Stenden at the ICT & IC Information Technology Department in Emmen, serving as one of the mandatory requirements therein to achieve his bachelor's degree.

With NHL Stenden being an international educational institution, there are campuses across the country as well as abroad. The one of relevance for this project is in Emmen and offers four international courses, ICT & IC Information Technology course is one of which. This graduation project will be solely undertaken by the author of this document, a 4th year international IT student to complete his 4 years study of ICT in Stenden.

As the details of this commission have yet to be finalized, this document is subject to change and any alterations will be shared promptly and marked accordingly.

1.1 Layout of the Project Plan

General information: Provides an overview of both the sponsoring organizations well as the organization in which the project is to be carried out. Project objectives: Consists of the objectives which are determined by Q-ICT as well as the projects' intended results. Project activities: Defines the tasks that need to be performed for the realization of the project, while taking into consideration the prioritization of certain tasks based on significance.

1. Project boundaries: States the project boundaries and the project duration.
2. Products and Interim Results: Specifies the products and results, which extends non-physical products such as client meetings.
3. Quality: Provides an overview of how the quality of the work will be monitored and eventually assessed.
4. Project Organization: Comprised of organizational information regarding the author and Q-ICT and its inner workings, as well as details regarding the correspondence with the sponsoring organization.

5. Planning and Scheduling: will show the schedule of tasks to be done as well as their dependence on other tasks for their completion or quality. Including various charts and diagrams, it will outline the time-scope as well as the desired time-consumption of the tasks.
6. Costs and Benefits: Discusses the time and resources required, while taking the benefits of the sponsoring organization into consideration.
7. Risk Analysis: Gives a detailed account of potential internal and external risks.

Chapter 2

Project Objectives

The primary objective of this assignment is to design and develop a new functionality on top of an existing application that can monitor Q-ICT and its client's networks and endpoints, then visualize and display the data. The feature should allow Q-ICT to control what data can be displayed to their users based on their tags.

The specific objectives of the project are considered as follows:

1. An in-depth research report about SentinelOne and its EDR capabilities and the current technologies behind the QaaS app
2. A dashboard module within the QaaS App that connects to SentinelOne API and Management Console, fully operational
3. To ensure that the product is scalable and can be easily adapted to accommodate future changes and additions to Q-ICT's product line.
4. To make the visualization of the returned data from the API calls in the form of JSON for user-friendliness.
5. To ensure that the product access is secure, adhering to all cybersecurity best practices, and that all sensitive data is protected from unauthorized access or manipulation.
6. To deliver the project on time (90 working days), within spec, and to the satisfaction of the project stakeholders.

Chapter 3

Project Activities

Project Activities details all activities sorted by category that are needed to successfully complete the project.

3.1 Planning

1. Initial contact with Q-ICT (via e-mail and/or LinkedIn).
2. Receive the assignment description from Q-ICT, make a definitive version of the Description of the Graduation Project based on the received information.
3. Make an initial version of the research topic, main question, and 4 research sub-questions and add it to the Description of the Graduation Project document.
4. Send the definitive version of Description of the Graduation Project document to graduation.ict-ct-emmen@nhlstenden.com and to get it approved.
5. After making sure that all the requirements to start the graduation project are met, send the Notification Form document to graduation.ict-ct-emmen@nhlstenden.com to notify them that the author has officially started his graduation work placement and to receive a graduation School Supervisor assigned to the author from the Graduation Committee.
6. Receive a School Supervisor and set up a physical/online meeting with him/her to discuss the graduation work placement and receive feedback.
7. Complete and sign the NDA of the graduation project between NHL Stenden and Q-ICT, along with getting the NDA the signatures of Niels Dorm and the Company Supervisor.
8. Triplicate the NDA, so that the school, the author, and Q-ICT can keep the document.
9. Make a draft version of this document (Project Plan)
10. Discuss the draft version with the Company Supervisor.
11. If required, implement the changes proposed by the Company Supervisor.
12. Make a definitive Project Plan.
13. Make a definitive version of Research Proposal and Planning.
14. Send the definitive version of Project Plan, Research Proposal, and Planning to the assigned School Supervisor within 2 weeks of the initial date of the graduation work placement.

15. Create a GitHub repo and GCP account from Q-ICT, ensuring that the author obtains all necessary permissions to access Firebase cloud service products.
16. Gather information about the project, its boundaries, and goals.
17. Have meetings and interviews with the stakeholders and product owner to discuss and discover further boundaries and project specifics.
18. Research project planning & management, and the technical part of the research topic, study available materials.
19. Set up a project backlog and a SCRUM board in an online DevOps environment.
20. Write User Stories and Tasks.
21. Create Sprint backlogs based on user stories.
22. Receive the previous code and documentation from the previous iteration of this project.

3.2 Research

1. Gather information about the project commissioner.
2. Set up research thesis document on the agreed research topic.
3. Create a definitive version of the research topic, main question and 4 research sub-questions.
4. Discuss research questions with the Company and School Supervisor.
5. Analyze existing product done by the previous developers.
6. Look up programming resources and proprietary integrated development environments.
7. Brainstorming with Company Supervisor to define tasks for the product backlog.
8. Determine feasible tasks.
9. Determine out of scope tasks.
10. Determine out of range tasks.
11. Make a list of the proposed features.
12. Make a list with the required hardware if any.
13. Present the list and ideas to the Company Supervisor and Product Owner.
14. Create an advice report for the clients and stakeholders backed up by theory.
15. Discuss concepts, ideas, and tasks with the client.
16. Write down the findings and discussion in a research report document.
17. Analyse the feedback from the Company Supervisor.
18. Create a feature/task priority document, set up the Backlog board accordingly.
19. Discuss a draft variant of the Graduation Research Report to Company Supervisor.
20. Send the initial draft of the Graduation Research Report to School Supervisor and receive feedback.
21. Implement the feedback and/or task ideas from the client into the task list.
22. Discuss the final concept with Company Supervisor.

23. Finish the research phase and all related documents.

3.3 Design

1. Create a Definition of Done with the Company Supervisor.
2. Create the visual design of the platform.
3. Create a test plan, making sure the platform works as planned.
4. Create a document for User Stories, explaining what the system will do.
5. Create a User Manual, explaining to the users how to use the system.
6. Choose the required hardware.
7. Decide on fonts, color palette and other design elements.
8. Outlining required languages and components
9. Make a definitive choice of our concepts.
10. Sketch the final product.
11. Visualizing the final product with mock-ups (if applicable).

3.4 End Product

1. Finalize and use the definitive version of the concepts and tasks.
2. Develop the product.
3. Develop the responsiveness of the product.
4. Plan weekly sprints and sprints reviews.
5. Test the product according to Definition of Done requirements.
6. Implement the professional product in accordance with the frameworks and requirements of the stakeholders.
7. Make sure the professional product meets the given standards.
8. Respect the testing plan and scheduling.
9. Discuss and implement feedback coming out of the testing phase.
10. Make a report on the final product.
11. Wrap up all the documentation, proofread and finalize.

3.5 End-Presentation/ Colloquium

1. Clean up the project (code).
2. Sort and archive all documents.
3. Make sure the connections, the live version, the database, API connections, cloud functions, and other cloud services are working.
4. Collect essentials for the final presentation.

5. Build a coherent presentation detailing both the product and the road to achieving it.
6. Presenting the definitive product to the project commissioner (first and second school supervisors), as well as clients
7. Receive feedback and the grade for this graduation work placement.
8. Reflect on what has been done, and what could have been done better. (Retrospective)

Chapter 4

SCRUM Activities

This chapter will discuss the SCRUM activities that have been done during the development of this project. SCRUM is an agile work methodology used primarily in software development projects. The implementation of SCRUM in the project is as follows:

4.0.1 SCRUM Roles

According to (*Atlassian, n.d.*), a SCRUM team consist of SCRUM master, Product Owner, and the SCRUM Team.

The Product Owner for this project will be the stakeholders of Q-ICT themselves, which mainly consists of the managers, the security department leader, and the Company Supervisor. They know exactly what they want to achieve with the project, and they make most of the user stories of the project.

The SCRUM Master would be the Company Supervisor who is responsible for facilitating the SCRUM board and ensuring that the project stays on track.

The development team consists only the author himself, who is responsible for the development of the project.

4.0.2 Events

Events in SCRUM are time-boxed iterations, which means each iteration repeats these events and each event has its own time limit. The events used in this project are as follows:

Sprint

A sprint is a consistent iteration where User Stories are completed. It is usually about one or two weeks.

Planning Poker

also known as SCRUM poker, is a consensus-based technique used by the author and the Company Supervisor, to estimate the effort or complexity of tasks in the Product Backlog.

Daily Stand-up

Daily stand-up are short meetings, usually lasted around 15-45 minutes to ensure that everyone is aware of what others are doing. For this project, it serves more to tell the Product Owner what have the author done in the previous day. Except on Monday, the daily stand-up is scheduled at 09:30, usually done online in a Teams call.

Sprint Review

It is a periodic meeting that happens at the end of every Sprint to show the client and the stakeholders the progress of the assignment so that they can either approve it or ask for changes. For this project, a sprint review is set every Monday at 09:00 for the Development team. This is where the Product Owner (Company Supervisor) checks if the sprint has been completed and implemented properly.

Sprint Retrospective

It is a meeting that happens after the Sprint Review to detect and improve a project's obstacles so that the same mistakes will not be made in future sprints. In this project, it is more like a private meeting between the author and the Product Owner to discuss what are the issues that the author has faced during the development of this project or to discuss any problems he faced within the organization as well.

4.0.3 SCRUM Artefacts

They are the documents of value to provide transparency and Opportunities for inspection and adaptation. All of these documents were produced in this project to ensure adherence to the SCRUM principles. This document will only discuss their definitions, which are the following:

Definition of Done

A document stating that User Stories can be considered done when it has passed the stages written in the document. The author has created this document before the beginning of the Realization phase where it is agreed on and signed by both the author and Company Supervisor.

Product Backlog

Is a prioritized list of all the work that needs to be done for the product. It includes features, enhancements, bug fixes, technical work, and knowledge acquisition that are necessary to deliver a viable product. The product backlog are available in Azure DevOps environment of this project.

Burndown Chart

Is a visual representation of the work remaining versus time. It helps the author track progress during a sprint, and predict if the author will complete the work by the end of the sprint. It will be in form of a separate document that the author have created for himself, not in Azure DevOps

Sprint Backlog

Is a subset of product backlog, consisting of the items selected for the current sprint plus a plan for delivering them. The sprint backlog is available in the DevOps environment of this project.

Chapter 5

Project Boundaries

This chapter will give further insight into what the final product will consist of and what is not within the scope of this project. Should there be any changes to the scope of this project, this section will be updated accordingly.

5.1 Duration

The project will take place throughout period 4 of the fourth year of the author's study in NHL Stenden, incisively from 19 February 2024 to 24 June 2024. The duration of the project is 90 working days with a target of 40 hours per week. The author will carry out the project at the host company office building in Kapitein Nemostraat, Emmen.

5.2 Scope

No budget is given for this project.

The scope of the project is limited within SentinelOne capabilities as an EDR solution. This project is, in part, a continuation of feature development of an already existing application, the QaaS App. As this feature is a new addition to the software as per stakeholders request without the acknowledgement of the client, it is expected that the project will be carried further and improved upon in the future. It serves as the Minimum Viable Product (MVP) as the basis pilot module for SentinelOne dashboard in the QaaS App.

5.3 Resources

The author will be given a desk with a docking stations and 2 monitors at the office building of Q-ICT in Emmen. The author will have rights to access the company's network and SentinelOne environment and resources. The author will be provided a supervisor that knows the field of coding, the Senior Developer of Q-ICT, who will be responsible for guidance and supervision to the author during the project.

The technologies used for the implementation will be using the same framework and cloud solution for the current software, namely Flutter, which is in Dart, and Firebase, which is in Node.js with TypeScript/JavaScript, and a NoSQL database from Firestore. The student will be given an environment in Firebase, as well as other existing technologies that will make it into the integration like Algolia and Google Secret Manager.

5.4 Preconditions

Below are listed conditions that should be met for the project succeed:

1. The author needs to implement SCRUM Agile management framework in the project.
2. Deliver the required documents stated in the graduation project module book.

The author has been asked to develop the platform with scalability in mind, as this will not be the final iteration of this product and might be carried out further by other development teams of Q-ICT later down the line.

Chapter 6

Producs and Interim Results

Within this chapter, the expected deliverables and interim results of both periods are listed and briefly explained. This section will be updated should the need for further documentation or products arise during the project.

6.1 Documentation

The documentation of the project will be composed of a project plan, a research report, minutes of the most informative team meeting, advisory documentation based on the research report (potentially as part of one of the aforementioned documents), a product backlog made up of tasks based on the prepared user stories, user manuals and a project report.

The documentation of working hours is done by means of an Excel file that allows the author to enter their individual hours with a brief description what was achieved in that time. Based on the product backlog, that are updated and reviewed weekly, as well as detailed sprint backlogs are created for the realization phase.

6.2 Research and Concept Specifications

This will contain the different concepts thought of during the project and the definitive concept chosen with clear reasoning and arguments. These include - a study title, a working group and investigators, background and rationale, specific aims, and analysis framework.

Furthermore, extensive research will be conducted on scrum and knowledge management, and how they can be used in the realization of the project. This documentation will help to keep track of the progress of the overall product conceptualization and the intermediate products, that will be stated in the final project plan. The backbones of this will be represented by the research report, to ensure educated choices are made for the selection of methods and overall approach for the product development.

6.3 Design architecture

Software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among each other, and properties of both elements and relations.

The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as a blueprint for the system and the developing project, laying out the tasks necessary to be executed by the teams. Dif-

ferent ideas will be explored to see what fits the project the best, and a definitive choice will be made in consultation, together, with the clients.

The database architecture is provided by the client. Database architecture refers to the organization of data within a database system. It defines how data is stored, organized, and accessed by the users and applications that interact with the database.

6.4 Final Product

At the end of the agreed project allocated timeframe, a complete final product will have been delivered.

6.5 End Presentation

Once the allotted time for the project concludes, the results of which will be shared in a manner specified by both the client and the lecturers, to display the results as well as give a general insight into the project timeline and working methods.

6.6 Interim Products Overview

The table below shows the various documents that need to be finished during this period along with their respective deadlines.

Task	**Deadline**
Minutes / Documentation of important meetings/ Interview transcript between the author and the stakeholders	As needed
Daily Scrum meetings	Daily
Monthly Report Document (Accountability)	Monthly overview, fill in daily
Project Plan (Initial Version)	First 2 weeks of the Graduation Work Placement
Research Proposal	First 2 weeks of the Graduation Work Placement
Planning	First 2 weeks of the Graduation Work Placement
Product backlog	Week 1-2
Midterm evaluation	Week 4-5
Sprint review	Week 2-10
Sprint retrospective	Week 2-10
Sprint backlog	Week 1-10
Mutual Peer Evaluation	Week 10
Presentation of realization phase results	Week 10
Thesis: All research necessary for the realization phase compiled in one document	Week 9 (End of the Research Phase)
FO/SRS/PRS	Week 10 (first week of the Realization Phase)
Definition of Done	Per document
Code of Conduct	Week 10 (first week of the Realization Phase)
Version Control	

Chapter 7

Quality

7.1 End Product

The product of this project is to deliver a working, easily scalable and maintainable web application where the client can view the statistics of any machines they have produced, as well as their clients should be able to access these data of their machines. The information can be seen for each machine one by one and can also be visualized from a graph.

This web application must be easily accessible, each component must be created in a separate environment so that each part can be managed easily or scaled up if there is the need to, without breaking the rest of the features or containers.

Special attention needs to be put in the security aspect. Access must be limited to logged users, and each incident listed must be manually curated by approved users. The database must also be protected from any injections or security flaws and the data must be sanitized and uniformised, if possible. More info about the database structure can be found in the Database Architecture document.

7.2 Control

Code wise, everything will be managed in GitHub, as the author will follow the best practices of quality of code guidelines, which consists of several programming conventions such as Comment conventions, Indent style conventions, Line length conventions, Naming conventions, Programming practices and Programming principles and Ethics Principles. To control the quality of the code, uploading to the Main/Master branch requires the code review and authorization from the Company Supervisor, as checking the code pre-emptively will reduce unnecessary time spent fixing the program and thus delays. Changes to the upper branch are forbidden, each new update needs to be committed from a separate branch that has been pulled from the most current safe build.

The interim results such as documentation and various features of the program will be checked by all group members and proofread by the members delegated to this, and feedback will be requested from university staff and the client.

The communication from the author and the company supervisor will be on Teams, Outlook e-mails, and WhatsApp messages, but also physical meetings, in the building of Q-ICT. To make sure that the project is going smoothly, and both parties are on the same page regarding what is being created and what must be made, weekly report sessions will be held between the author and the Product Owner, with the possibility to the school's supervisor to join.

Chapter 8

Research Quality Assessment

The research quality was assessed using the following criteria:

Currency

1. **Timeliness**: The publication date of each source was checked, to ensure that the information is up-to-date. Depending on the research requirements, both contemporaneous materials and current accounts of historical events were taken into consideration.

Relevance

1. **Relevance**: The source was checked to ensure that it is relevant to the project. Journal articles were examined by perusing abstracts and introductory paragraphs, and the publication date was verified to confirm its applicability to the research needs. As EDR solutions are recently a new aspect of cybersecurity, the sources can be ensured to be up-to-date and relevant to the research. The aspect of data visualization can be dated up to 1981, therefore the author will be more focused on the recent material, dating up to 2010, as the audience of this data visualization has changed as well to be more modern and up-to-date.

2. **Source Selection**: the author ensured that the sources were selected from a variety of sources, including journal articles and websites.

Authority

1. **Credibility**: the credentials of each source's author were evaluated to ensure their subject-matter expertise. Biographical reference sources and search engine queries, including tools such as Google Scholarly, were employed to obtain information on the writer's background and qualifications, and the context in which the source was written.

2. **Type of Source**: each source was classified as popular, scholarly, internal or trade, with a clear distinction between the different types of sources. Popular and most convenient sources, such as web articles, were used to gain a general understanding of the research topic. Trade sources, for instance publication standards, were used to gain a better understanding of the research topic and to identify the most relevant source. Scholarly sources, such as journal articles and publications, were used to gain a comprehensive understanding of the research topic and to identify the most relevant sources. Lastly, the internal sources were used to gain a comprehensive understanding of the requirements for the research and execution. Oher types of sources were often not taken into account, due to their irrelevancy and inadequacy, such as Wikipedia articles.

Accuracy

1. **Reliability**: to establish the reliability of each source, the facts and data were cross-referenced with other documents addressing the same subject.
2. **Validity**: the author assessed the validity of each source by examining the author's credentials, source, publication reputation, and the presence supporting the evidence. The author also considered the objectivity of each source to ensure that the information was presented in neutral manner.

Purpose

1. **Purpose and Commercialism**: the purpose of each source was scrutinized to determine its congruence with the research objectives. The author analysed whether the source aimed to persuade, informed, provide an overview, or incite controversy. Similarly, it was assessed whether the source contained advertisement or other forms of commercialism that might influence the information provided. The impact of commercial on the source's content was carefully evaluated. This led to exclusion of sources, such as blogs and personal opinions from web articles, that were deemed to be biased or unreliable, such as those were sponsored by a company or organization with a vested interest in the research topic due to commercial reasons.
2. **Intended Audience**: the author appraised the target audience of each source to ensure its suitability for the research objectives. The source author's style and the intended audience of the source were considered.
3. **Conclusion**: the author ensured that the sources were appropriately used and that the information was cited correctly, in accordance with the APA 7th Edition citation guidelines.

Chapter 9

Project Organization

9.1 Team member, product owner, and contact persons

There is only one member of the team, the author, along with 2 supervisors from Q-ICT and one supervisor from NHL Stenden.

Name	Email	Role	Phone
Christopher Sulistiyo	c.sulistiyo@qict.nl	Author	+31 627553451
Manuel Weidjik	manuel@qict.nl	Senior Software	+31 20 330 2216
Mark Kolk	m.kolk@qict.nl	Company Supervisor	+31 6 85454829
Pierre Kleine Schaars	pierre@qict.nl	Product Owner	+31 20 330 2216
Robbert Remie	r.remie@qict.nl	Stakeholder	+31 20 330 2216
Frans Huijgens	@nhlstenden.com	School	
Niels Doorn		School Lecturer	

9.2 Availability

The author is available on working days between Mondays until Fridays between 08:30-17:00. He can be contacted by e-mail, Teams, WhatsApp, SMS, or a call. Outside of these times the author is not expected to be working on the project but may be reached out due to the flexibility of certain occasions, special occasions, i.e., shortly upcoming deadlines or goals not achieved according to the scrum or activity board. However, the deadline will be kept in mind and the working hours might be adjusted based on that.

Any of the absence days will have to be explained by the author with the Monthly Report. The author will decide how to proceed, putting the project completion as the main objective. Also, all hours that have been done must be recorded in the Monthly Report document with a brief description of what was achieved that will eventually be sent to School Supervisor.

9.3 Team roles

There are a couple of roles within this Graduation Work Placement:

1. The author himself is responsible for coordinating tasks within the group and making sure those tasks are completed on time and meet the quality expectations. Moreover, they are responsible for piloting the project in the

right direction and acting as a bridge in communicating between Q-ICT and the teachers. He also schedules meetings with the teachers and Company Supervisors if needed.

2. The author as the developer is focused on doing the tasks that he himself created.
3. The company supervisor as the SCRUM Master, is focused on facilitating the backlog among the developers, and making sure that the Scrum team carries out every sprint perfectly.
4. Product Owner defines the why is it worth to develop the product, who it is for, and what features, requirements should it contains, and to main the product backlog.

9.4 Reporting

The author will meet at least once a week for a Sprint Review and Sprint Retrospective (usually on Monday at 09:30, at the beginning of a new sprint) with the Company Supervisor to discuss progress for individual tasks and overall project progress. Moreover, both members get to speak up about their concerns about how progress has been made so far. This is to ensure that there are no unspoken conflicts or issues. This meeting usually takes about an hour. All this will be taken down from the 8 work hours that are required by the author daily.

Using the weekly quota, the author can then plan and adjust the needs of the project going forward. Furthermore, the author has decided to use Teams, Outlook, and WhatsApp as the main ways of communication. Outlook e-mail has been chosen for official contacts between the author and the School Supervisor, with MS Teams as an alternative.

Furthermore, a Daily Standup with the Company Supervisor (and additional interns at his disposal) will be held every working day, with the maximum duration of 30 minutes. This usually takes place at 09:30. This Daily Standup is where the author would state what has been done the previous day, what are his plans for this day, and discuss any potential impediment that he is facing. This method will ensure transparency, collaboration, and adaptability within a real working environment, ultimately contributing to the successful delivery of the sprint goal working as a team.

Initially, during the research and documenting phase, a hybrid project planning method will be used. Whenever work must be done, tasks will be created and assigned to one or more team members along with a deadline. To keep track of current and past tasks, a SCRUM environment chosen by Q-ICT itself will be used, along individual tracking of hours and progress. At least 2 weekly meetings will take place where the tasks will be discussed, and new tasks will be added.

During the period when the SCRUM method is being used, the author is expected to keep his details up to date in his Monthly Reports, as well as accurately noting down what feature he has worked on. Once a sprint concludes, the sprint is reviewed together with the Company Supervisor and new user stories for the following one are selected based on the product backlog. This backlog is also updated with any new tasks that may have popped up during the sprint and unfinished tasks are updated and reentered.

Finally, the author came to an agreement to use GitHub as the digital tool for collaborating within the project group. This will enable a smooth workflow for the team members, as well as securing older product versions in case of a mishap. This platform will be able to show the working progress of the project, as well as keep track of the SCRUM tasks by using its Project feature. This method of work will be implemented during the developing stage of this project.

Chapter 10

Planning and Scheduling

As effective planning and scheduling are essential for delivering a high-quality software application within budget and on time, by taking a structured approach to project planning, the development can be ensured that it is working efficiently and effectively towards achieving the project objectives.

Below the prioritization of project deliverables, estimation of their timeframes and a Gantt chart depicting set time frames are described.

The graduation work placement project's duration is 90 working days, which makes up for about 18 weeks. The graduation project is split up into 2 parts, the Research phase, and the Realization phase. The Research phase is where the author will work more into the necessary documentations required as stated in the module book, and in the end will produce a Report. The Realization phase is where the author is more focused on the technical coding part of the project, working closely with the Company Supervisor in SCRUM and in the end will produce a working end-product as stated in the Graduation Project Description document.

Because of the nature of planning which is for time estimation and serves as a roadmap for the project, the author will assume that each phase of the project will take about 9 weeks, thus making up the total number of 18 working weeks. Please keep in mind that this plan is just for guiding the author through each phase of the project lifecycle and ensuring that all parties (Q-ICT and NHL Stenden) are aligned with the project goals, timeline, and expectations. Some future changes might be required as the author sees fit, as one cannot predict encountering random unpredictable risks, thus rendering the process development of this graduation work placement project.

10.1 Research Phase (26/February/2024-29/April/2024)

Code	Task	Weeks	Can only take place after
A	Graduation Work Placement Description	1	-
B	Non-Disclosure Agreement	1	-
C	Graduation Project Notification Form	1	-
D	Project Plan (draft version)	2	A, B, C
E	Research Proposal	2	A, B, C
F	Research Thesis Report Paper Document	6	A, B, C, D, E
G	Monthly Reports	1-9	-

H	Interview Transcripts	5	-
I	Test Plan	7	G
J	Functional Design (FD)	8	D
K	Technical Design (TD)	8	D

10.2 Realization Phase (29/April/2024-08/July/2024)

Code	Task	Weeks	Can only take place after
A	Definition of Done	1	-
B	Product Backlog	Epics	2 - 9
C	Sprint Backlog	3	-
D	Research Document (final version)	8	G
E	Monthly reports	1-9	-
F	Sprint tasks	2-9	G
G	End-product	9	A
J	End-presentation (colloquium)	10	G

It is important to note that unexpected hurdles may arise during the execution phase, leading to deviations from the original plan. These deviations may arise due to factors such as prolonged feedback and approvals from clients or situations where activities require more resources than initially anticipated as well as plain lack of knowledge. Therefore, it is crucial that the author and the planning remain flexible and provide enough room to accommodate changes as required to ensure successful project completion and school and Q-ICT satisfaction.

Chapter 11

Cost and Benefits

This chapter will discuss various costs and benefits relating to this project.

11.1 Cost

For this project the budget is set around 20€ per month in terms of resource consumption on the Firebase environment. However, should the author require anything in terms of software and/or cloud access, Q-ICT may provide this.

Building the project should not prove expensive since the product is a software application. The expenses may come with hosting and maintenance of the product, for which the author is not responsible.

11.2 Benefits

The project will provide benefits in terms of allowing Q-ICT, as well as its clients, to have data visualization of the gathered data for internal use or clients' usage. This functionality will ultimately enable the QaaS app to monitor their internal APIs and enable more automation as the new functionality will be a template for integrating their future APIs.

As for the author, building a web dashboard for a real client will provide the author with real-world experience and an opportunity to apply their knowledge and skills in a practical setting.

Chapter 12

Risk Analysis

This chapter handles the possible risks that may be encountered during the project and ways to handle the problems. This is visualized on the table below.

Risk	Possibility	Possible Outcome	Prevention	Measure taken
Graduation Work Placement is not finished on time	Likely	The author will be forced to take extended periods in his study in NHL Stenden Emmen	Not possible	
Contracting COVID-19	Less likely	The author will have to quarantine for 2 weeks, and the project will be set back.	Maintain 1.5-meter distance and use face masks when needed	The author will have to work during down time not to fall behind, getting vaccines and respecting measures
Common illness	Less likely	The author will be set back for a moment.	Good health care and communicate properly with the Company Supervisor	
Tasks not finished on time	Possible	Set back until tasks are finished	Proper planning and time management	Sprint reviews, plan next sprints differently, issue warnings if other causes

Insufficient task	Less likely	Set back until task schedule is done correctly	Actively ask members for document / peer review, proof-reading.	Plan next sprint accordingly
Lack of technical knowledge	Less likely	The author might not be able to realize certain parts of the project.	Research proper technical skills needed, practice these and ask for assistance if needed	Research what skills the team is lacking,
Loss of data	Unlikely	Loss of data/files. The amount depends on the most recent back-up.	Frequent back-ups / version control	Restore files from latest back-up / revert to an earlier version
Insufficient testing	Possible	End-product of lesser quality	Structured Test Plans and sufficient time to complete the tests	Review test reports and run tests again
Hardware malfunction	Unlikely	Set back until hardware is replaced	Actively check if hardware is in working condition	Order a new part as fast as possible and replace it
Lack of proper communicat ion with the	Possible	This will slow the development of the project.	Frequent communication with the School Supervisor, follow up	Weekly meetings, multiple ways of communication and reaching out

School Supervisor			emails, messages through WhatsApp, calling on the phone in case of emergency	
Security issues	Unlikely	Personal information could be stolen.	Testing for security issues.	Consultation regarding Secure Programming and following best security practices
Security issues on the libraries used	Possible	Update on the libraries, or change the libraries	Use newer, community trusted libraries, with frequent updates.	Check for updates and be sure that the libraries used can be applied in the future.
Additional research required by the stakeholders	Possible	The progress will be slowed down, causing troubles with keeping the deadline.	Good communication between the author and the stakeholders.	The author will try to communicate as efficiently as possible with the client to avoid any misunderstanding.
Workload Too Heavy	Possible	Due to the possibility of the author being unfamiliar with some of the subject matter, there is a possibility there may be miscalculations	Reassess progress frequently and communicate any setbacks with the	Plan sprints accordingly and have weekly sprints reviews.

		regarding the sprint planning	Company Supervisor	

Table 2: Risks

Appendix

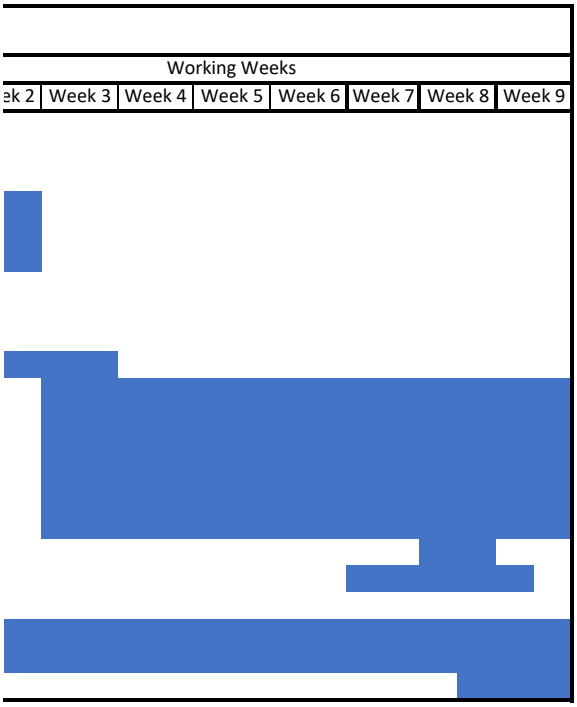
Gantt Chart

Appendix A

Gantt Chart Research Page

Research Phase Gantt Chart (26/February/2024 - 29/April/2024)

Activities	Week 1	Wee
Graduation Project Description		
Graduation Project Notification Form		
NDA signed and triliplicate by 3 parties		
Research Proposal		
Project Plan		
Planning		
Interview with Company Supervisor, Stakeholders, Owner and Cybersecurity Consultant to determine requirements		
Setting up LaTeX environment for document editing		
Initial meeting with the School Lecturer to determine Requirement Analysis		
Set up Azure DevOps environment (developers and stakeholders account) & GCP Firebase E-mail Account		
Studying the QaaS app front-end (Flutter code)		
Studying the back-end code (Node.js server, Firebase console, cloud functions, and other services utilized)		
Studying and reading the SentinelOne API documentation		
Studying and reading the documentations of N-Central for API monitoring		
Studying and reading the documentations of Pax8 for API monitoring		
Studying and reading the documentations of Bodyguard for API monitoring		
Studying and reading the documentations of for API monitoring		
School Lecturer Visit		
Making SRS, FO, PRS, with Class Diagram, Sequence Diagram, Use Case Diagram, Activity Diagram (optional)		
Monthly Report		
Sprint Review for Company Supervisor		
Research Report excluding Realization, Conclusions & Recommendations, and Evaluation chapters		


Working Weeks
ek 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9

Appendix B

Gantt Chart Realization Page

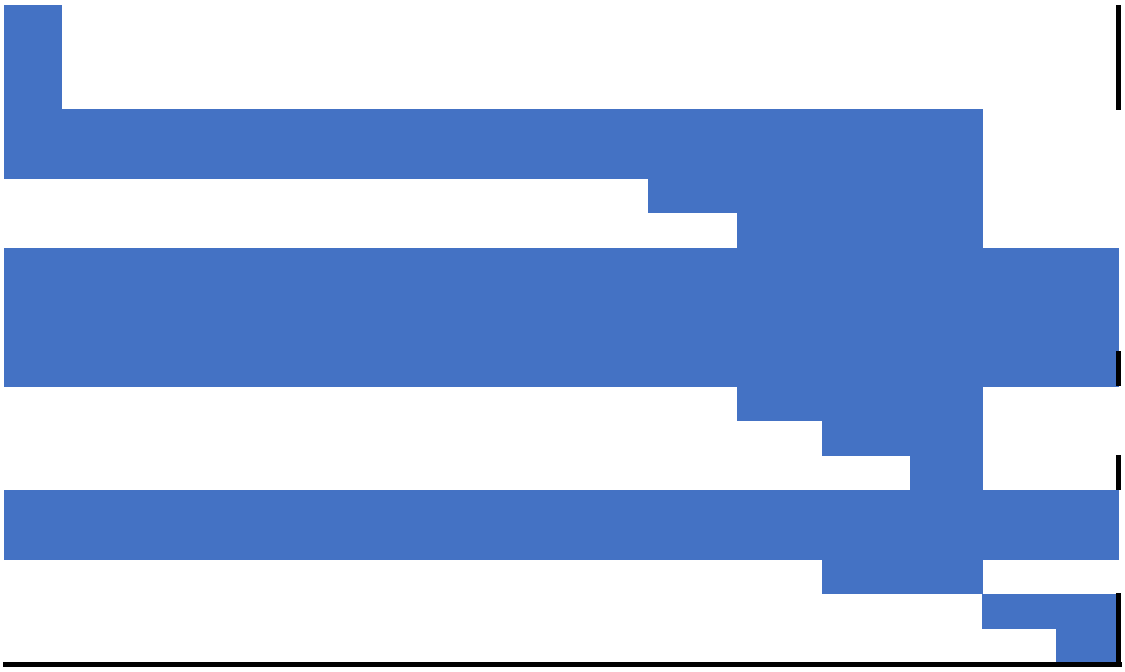
Realization Phase Gantt Ch

Activities	Week 1
Determine Definition of Done with Company Supervisor	
Setting up the SCRUM board in Azure DevOps	
Determine Story Points	
Create Sprints	
Create Product Backlogs	
Create Burndown Charts	
Code in the front-end client-side Flutter test environment	
Code in the back-end server-side Node.js and Firebase cloud environment	
Updating Research Report and filling out the missing chapters	
Research Report check with Company Supervisor	
Daily Stand-ups	
Sprint Restrospective	
Sprint Review	
Unit Testing	
Bugs finding and fixing	
Finalizing the final product	
Deploy the final product in Firebase hosting	
Monthly Report	
Sprint Review document for Company Supervisor	
Research Report (initial draft)	
Research Report (final draft)	
End-presentation (colloquium)	

art (29/April/224 - 8/July/2024)

Working Weeks

Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9



Appendix E

Research Proposal

Research Proposal

Christopher Sulistiyo (4850025)

Christopher.sulistiyo@student.nhlstenden.com

ICT & IC Information Technology Department Emmen

Client

Quality ICT B.V.

Version 1.1 – 21/02/2024

VERSION CONTROL **2**
REMARKS **2**
INTRODUCTION **3**
REASONS FOR THE RESEARCH 3
RESEARCH OBJECTIVES 4
RESEARCH MAIN AND SUB-QUESTIONS 5
CONCEPT THEORY 6
Concept clarification 6
Definitions to apply to the knowledge base. 6
Sources validation within the company or employees 10
RESEARCH SETUP **11**
DESIGN/ DATA COLLECTION 11
POPULATION AND SAMPLE 11
ANALYSIS PROPOSAL 12
INTERNAL & EXTERNAL VALIDITY AND RELIABILITY 12
USABILITY 12
RESEARCH TASKS **14**

Version control

Version	Activities	Date
Initial version 1.0	Draft version	06/02/2024

Remarks

Any changes and new developments that have a significant impact on the project proceedings will be noted here.

Introduction

Reasons for the Research

Q-ICT is a small cybersecurity consultant company based in Emmen, Drenthe, the Netherlands, with clients ranging from small to medium businesses sized companies (SME (small and medium-sized enterpirses) in English or MKB (midden- en kleinbedrijf) in Dutch) with employees ranging from 1-100. It currently manages numerous third-party APIs without a standardized implementation within its internal application, the QaaS app. Those APIs are listed in the following:

- **Snelstart**: is used for company's automation of financial and accounting system software, such as managing invoices, etc,
- **N-Central**: is used for monitoring the clients' devices. It is a RMM (Remote Monitoring and Management) platform designed to help MSPs (Managed Service Providers) and IT professionals to remotely monitor, manage, and support their clients' IT infrastructure.
- **Pax8**: is used for their Microsoft subscription. It is a software that helps Q-ICT to simplify the way to buy, sell, and manage cloud solutions.
- **Bodyguard.io**: used for security tab. It is a software developed by a Dutch company to filter and scrutinize downloads from web browsers to detect and prevent malicious files with real-time download scanning capabilities. This API is still relatively newly added by the company,

These third-party APIs are crucial within Q-ICT, which jobs include offering and advising best solutions and practice for their clients' IT infrastructure by performing security scans and monitoring their devices health. The functionalities of these third-party APIs are included in the following, but not limited to:

- Monitoring endpoint security events and health status to ensure the effectiveness of security measures.
- Manage and provision cloud-based security solutions to protect client infrastructure and data in the cloud.
- Automate security operations and incident response to efficiently address security threats and vulnerabilities.

- Provide a comprehensive cybersecurity solution tailored to the specific needs and constraints of SMB (Small to Medium Business) clients, enhancing their overall security posture.

Currently, the QaaS app does not have a way to manage their numerous third-party APIs without a standardized implementation. This resulted in a lack of user-friendliness, slow and unclear navigation and data visualization, and the difficulty to integrate a new potential API that might be beneficial to the company. The company consequently seeks to implement a new component within the QaaS app, with functionalities and features such as establishing comprehensive monitoring of these internal APIs, ensuring connection status, error handling, handling of expired API keys, secure storage of keys, and external validation of API connections. Q-ICT, therefore, has asked the author to conduct research regarding the industry-standard best practices for developing and implementing this new component to the QaaS app, while keeping in mind the potential impact of this implementation to its existing operational efficiencies, cybersecurity posture, and end-users' perceptions and interactions with already existing functionalities in the QaaS app. Furthermore, Q-ICT also seeks to implement SentinelOne API, a new AI powered cybersecurity platform for cyber threat detection and remote IT infrastructure management yet to be explored by the company to the QaaS app. The author is also therefore asked to make a way for the QaaS to be able to necessitate interpreting the all the APIs' response in the form of XML and JSON files.

Research Objectives

The objectives of this research are listed in the following:

- **Integration with SentinelOne**: this research will also explore how SentinelOne can be integrated into the QaaS app to align with other modules of the QaaS App while leveraging its key features for cyber threat detection and remote IT infrastructure management.
- **Best visualization techniques for SentinelOne data**: this research will also propose suitable visualization techniques for displaying data processed and received by the internal application in XML and JSON formats, focusing on clear and insightful representation of threats detected by SentinelOne API.

- **Follow code conventional guidelines, maintain best cybersecurity practices, and unit testing**: during the development of the final product, the Company Supervisor wishes that the author follow the best code conventional style with MVC architectural style that was already implemented on the QaaS app, as well as following the best cybersecurity standards and implementation of unit testing in both Flutter front-end and Firebase back-end code.

Research Main and Sub-Questions

A research main question is a crucial component when doing research as it serves as the driving force behind the study, guiding the research process, structuring the paper, promoting clarity, driving inquiry, addressing relevance, and facilitating evaluation. The main research question is as follows:

"How can Q-ICT effectively-integrate and leverage SentinelOne security threat platform for continuous cybersecurity monitoring?"

The research sub-questions are crucial in shaping the research process and facilitating the thorough investigation and eventual answer to the main question. They are as follows:

- What is the current situation of the QaaS app, the internal application used within Q-ICT to monitor its third-party API calls, alongside its back-end infrastructure?
- How can SentinelOne be integrated into the QaaS app, while still utilizing key features and capabilities in the context of cyber threat detection and remote IT infrastructure management?
- What are suitable visualization techniques for displaying data processed and received by Security Threat Platforms regarding to SentinelOne APIs compares to other visualization techniques by the same platform?

These research questions do not make substantial statements about the research that will be executed and can be later subject to change after a proper consultation with the school supervisor.

Concept Theory

Concept clarification

In this graduation research, the central question revolves around integrating the SentinelOne security threat platform for continuous cybersecurity monitoring, all while ensuring adherence to the highest security standards. To elucidate this question further, it is imperative to provide stipulative definitions for the key terms and concepts involved in this study.

Firstly, the "SentinelOne security threat platform" encompasses a well-known comprehensive cyber security solution from United States of America designed to detect, prevent, and respond to various security threats within Q-ICT's IT infrastructure powered by AI. This platform utilizes advanced threat detection algorithms, machine learning capabilities, and real-time monitoring to safeguard against cyber threats such as malwares, ransomwares, phishing emails, and unauthorized access attempts.

Secondly, "continuous cybersecurity monitoring" denotes the ongoing process of monitoring and analysing Q-ICT's IT environment for potential security vulnerabilities, breaches, or anomalous activities. This involves real-time monitoring of network traffic, system logs, user activities, and application behaviour to proactively identify and mitigate security risks.

Furthermore, "adherence to the highest security standards" encompasses the implementation of industry best practices, regulatory compliance requirements, and robust security measures to ensure the confidentiality, integrity, and availability of Q-ICT's internal application and data. This includes adhering to the standards such as ISO 27001, NIST cybersecurity framework, and GDPR regulations to mitigate security risks and protect sensitive information.

Definitions to apply to the knowledge base.

Definitions used in the research:

Real-time monitoring: the continuous tracking and analysis of API interactions within Q-ICT's internal application to detect and respond to events as they occur, ensuring timely identification of anomalies or security threats.

Error detection: the process of identifying and flagging any deviations, discrepancies, or failures in API transactions within the internal application, facilitating prompt resolution and maintaining system integrity.

Insight generation: the extraction and analysis of data from API connections to gain actionable insights and inform decision-making processes related to system performance, user behaviour, and security posture.

Cyber threat detection: the process of identifying, analysing, and responding to potential security threats within Q-ICT's IT infrastructure, including those originating from third-party API interactions, utilizing SentinelOne's advanced threat detection capabilities.

Remote IT infrastructure management: The ability to remotely monitor, configure, and manage Q-ICT's IT infrastructure, including API endpoints and security measures, leveraging SentinelOne's centralized management features and remote access capabilities.

Data visualization: methods used to present complex data in a visual format that is easy to understand and interpret, facilitating insights and decision-making. Examples include pie charts, charts, graphs, figures, pictures, and heatmaps in the dashboard.

Internal threats: security risks originating from within Q-ICT's IT infrastructure, including potential vulnerabilities, unauthorized access attempts, and anomalous activities detected by SentinelOne, other third-party APIs, and other security measures.

SentinelOne data: information collected and processed by SentinelOne regarding detected cyber threats, including threat types, severity levels, affected systems, and response actions.

Q-ICT relevant API connections: third-party API interactions within Q-ICT's internal application, including data exchanges, system integrations, and communication channels.

Knowledge Base Application

Before proceeding any further, it is important to know what the definitions of implicit/tacit and explicit/expressive knowledge are. Below are the given definitions:

<u>Implicit knowledge</u>: is a knowledge that is gained through incidental activities, or without awareness that learning is occurring. (*Wikipedia, n.d.*). It helps individuals differentiate themselves from other candidates. When an organization has a wide scope of tacit

knowledge, it can rely on its talented pool of employees to develop innovative solutions, thus helping the organization to become industry leaders. An organization can also recognize its tacit knowledge and develop the appropriate resources to train new hires as effectively as possible.

Explicit knowledge: is a knowledge that is straightforwardly expressed and shared between people (*Wikipedia, n.d.*). It can help employees within an organization share valuable information with their coworkers or customers. Transferring explicit knowledge to others in the workplace allows employees to learn new information that can help them do their jobs effectively.

Best practices:

- The integration of SentinelOne allows for comprehensive threat detection and response capabilities, ensuring the protection of API connections and IT infrastructure against cyber threats. This will algin with the industry best practices to enhance overall security posture.
- Data visualization emphasizes the use of intuitive and interactive visualization techniques to present complex cybersecurity data effectively. This includes techniques such as interactive dashboards, dynamic charts, and hierarchical visualizations that enable users to explore and analyse data related to internal threats by SentinelOne and other API connections.

Implicit knowledge:

- Employees within Q-ICT may process tacit knowledge regarding effective visualization techniques that have been successful in previous cybersecurity initiatives or similar projects. Leveraging this knowledge can inform the selection of visualization techniques tailored to Q-ICT's specific needs and preferences.

Explicit knowledge:

- Technical documentation and case studies from SentinelOne can provide insights into the platform's key features and capabilities relevant to cyber threat detection and remote IT infrastructure management. This explicit knowledge can guide the

integration process and help maximize the utilization of SentinelOne Q-ICT's environment.

- Resources such as academic literatures and case studies provide insights into the best practices and emerging trends in cybersecurity data visualization. This explicit knowledge can guide the selection of suitable visualization techniques that align with industry standards and facilitate effective communication of internal threat data to stakeholders within Q-ICT.

Application

Based on the definitions stated above and its knowledge base application, some example implementation can be drawn up regarding the graduation work placement project.

Sub-question #2:

To answer this sub-question, Q-ICT should strategically integrate SentinelOne into its environment, ensuring alignment with other module functionalities while leveraging key features for cyber threat detection and remote IT infrastructure management. By integrating SentinelOne's threat detection tools, Q-ICT can therefore enhance its ability to detect and respond to cyber threats originating from API connections in real-time. Additionally, leveraging SentinelOne's centralized management features enables remote monitoring and management of Q-ICT's IT infrastructure, including API endpoints, ensuring consistent security across the organization's digital ecosystem.

Sub-question #3:

To address this sub-question, Q-ICT should explore suitable visualization techniques for displaying data processed and received by internal threats detected by SentinelOne and other relevant API connections. Interactive dashboards can be used to provide an overview of detected threats, allowing users to drill down into specific details such as threat types, affected systems, and response actions. Dynamic charts and graphs can visualize trends and patterns in threat data over time, helping stakeholders identify emerging threats and prioritize mitigation efforts. Additionally, heatmaps can highlight areas og high threat activity within Q-ICT's IT infrastructure, guiding targeted security measures. By leveraging these visualization techniques, Q-ICT can effectively communicate internal threat data to

stakeholders and facilitate informed decision-making regarding cyber security strategies and resource allocation.

Sources validation within the company or employees

Lastly, for the validation of the sources within the company or employees, the interviews that will be conducted by the author will be primarily addressed to the developers of the QaaS app, the company owner, and stakeholders. Their respective name will be Manuel Weidijk as Q-ICT senior software developer, Mark Kolk as the owner of Q-ICT, and Pierre Kleine Schaars, Luke Admiraal, Pete Hoekstraa, and Bas Ter Heurne as the stakeholders of Q-ICT, along with its sister companies, MKBiT and QaaS (Quality as a Service).

Research Setup

Design/ Data Collection

In this research, different research methods have been used to answer the above-mentioned research main question. A research methodology must be defined for each research sub-questions to determine how the results are being considered valid and reliable.

- **Sub question 1**: a combination of desk research (Literature study of the library method of ICT Research Methodology) and Interview (from the Field method of ICT Research Methodology) with the creator of the QaaS app will be conducted, with the goal of understanding the infrastructure beneath along with understanding all its dependencies. The author will then gain a substantial amount of understanding from its Flutter-based code, as well as the Node.js TypeScript based RESTful API server behind it, and the Firebase from GCP (Google Cloud Platform) services and functionalities being used within the QaaS app such as Google Secret Manager, Firebase, Cloud Functions, Firestore, Google Authentication, Cloud Messaging, Cloud Storage, Google Analytics, hosting services, real-time databases, etc,.
- **Sub question 2**: the type of data collection that will be used are observation and interviews with the product owner and stakeholders.
- **Sub question 3**: this sub-question will be answered by doing experimental activities, creating reports, and doing desk research. The SentinelOne Internal API documentation will be used as selected measuring instruments.
- **Sub question 4**: this sub-question will also be answered by doing desk research, observation, experimentation, and reporting to the company supervisor and stakeholders.

Population and Sample

Population refers to the entire group of individuals and/or entities that the research aims to study (*Wikipedia, n.d.*). The population of this research mainly consists of a group of individuals ranging from high school students to any graduate from academic degree that are interested in the fields of IT, especially regarding API connections and best cybersecurity practices.

Sample represents a subset of population that is selected for the actual study (*Wikipedia, n.d.*). In this case, the sample would be the Software Development departments of Q-ICT B.V. itself and the teachers and supervisors of NHL Stenden Hogeschool of the ICT & IC Information Technology department in Emmen where the author is conducting this research for.

Analysis Proposal

Within this graduation process, the author aims to study and assess the existing internal application code and features crafted using Flutter for the front-end and Node.js for the server back-end. It seeks to meticulously analyse these components to identify any undisclosed bugs, assess their weaknesses and strengths, and evaluate any potential security vulnerabilities. Additionally, the author intends to explore opportunities for enhancements, identify areas for refinement, and pinpoint potential spaces for development of the proposed features stated in the Graduation Description.

Furthermore, the author endeavours to conduct a comprehensive analysis of Firebase as a cloud solution, examining its full suite of services offered, to ascertain the most suitable offerings to integrate for the proposed new features within the framework of the existing internal application for the graduation assignment.

Internal & External Validity and Reliability

The reliability is validated by interviewing the developers behind the QaaS app, incorporating any feedback to the application that they might give while the author is trying to implement this new feature. To further improve validity, the business and product owner, and additional stakeholders of Q-ICT will also later be questioned to achieve the desired outcome of the new component requested.

Usability

The usability of a research is often determined by the relevance and clarity of its main and sub-questions. It can be assessed as in the following:

- **Practical Implementation**: the results of this research will later be used to generate actionable insights and guidance during the Realization phase of the graduation work placement by the author. The Research Report as the end-result of this research will also provide recommendations to Q-ICT regarding the research topic.

- **Consideration of Best Cybersecurity and Industry Standards**: the inclusion about the needs to implement the best cybersecurity and industry standards demonstrates an awareness of the importance of aligning the research with those existing standards. This consideration enhances the practical applicability and usability of the research in the broader context of industry norms.
- **Impact Assessment**: the research will also discuss about the potential impact on integrating this new component on the operational level of the QaaS app product, indicating an intention to evaluate the practical consequences of the proposed solution. This focus on impact enhances the usability by providing insights into potential benefits and challenges of the main research question.

Research Tasks

The tasks that the author will undertake whilst doing this research are planned and included in the following:

- Interview with the developers behind the QaaS app.
- Interview with the owner of the Q-ICT.
- Interview with the additional 2 stakeholders of Q-ICT.
- Review the Flutter client-side front-end and Node.js server-side back-end code, as well as understanding all the Firebase functionalities used within the internal application.
- Review and study all the listed 5 APIs and SentinelOne API documentations.
- Conduct desk research regarding all the research sub-questions listed above.

Appendix F

Initial Description of Graduation Project

GRADUATION PROJECT DESCRIPTION

Attention: This form must be returned, via email, to the graduation agency (graduation.ict-ct-emmen@nhlstenden.com).

Explanation:

- The size of the graduation project description must be 1-2 A4 and must contain the points mentioned here below. The format of the fields is not fixed and can be resized to fit.
- It must be clear why this graduation project is of sufficient level (think about the complexity).
- The research should serve the realisation (See phase 2 SDLC.)

Name of student:

Student: Christopher Sulistiyo (4850025).....

Institution/company where assignment will be carried out:

Name: Quality ICT (Q-ICT).....
Address: Kapitein Nemostraat 20 .....
Town, post code: Emmen, 7821 AC .....
Department: Software development.....
Telephone: 0591 – 708 004 .....
Name of supervisor: Manuel Weidijk, Mark Kolk .....
E-mail of supervisor: manuel@qict.nl, m.kolk@qict.nl .....
Number of permanent employees: 20.....

Does the company meet the requirements mentioned in the handbook graduation:

(Think about the requirement about company and placement supervisor.)

Both supervisors are master graduates from the field of IT, with the latter has been working on the field of IT and security department for more than 20 years. Additionally, the company will facilitate the interns with the required resources for an adequate working facility such as work laptop and docking station for a proper desktop environment. The company itself is a cybersecurity company, solely dedicated for security advisor and client-made software development and has received many IT interns from NHL Stenden who got their projects approved before working for the company. Moreover, the company will grant the intern access to essential APIs and platforms (such as SentinelOne, N-Central, and Bodyguard.io) required for the monitoring application, alongside a range of GCP (Google Cloud Platform) services encompassing Google Secret Manager, Firebase, Cloud Functions, Firestore, Google Authentication, Cloud Messaging, Cloud Storage, Google Analytics, hosting services, real-time databases, etc. Additionally, access will be provided to a SCRUM board, as well as both test and live environments essential for software development.

Are the required competencies achieved:

(See graduation project manual, appendix E.)

During this project, the intern will analyze company's existing internal application, identify, and offer advice for any security weaknesses and/or bugs, strategize a design aligning with the project's

objectives and scopes for the company's internal application, and endeavor to implement the planned design within the project duration. The intern will be subjected to a comprehensive learning experience encompassing the creation of process documentation such as monthly reports, thesis, and user manual. Moreover, the intern will gain practical exposure to a real work environment focused on ICT and cybersecurity, engaging in SCRUM management, software development, and receiving guidance from the company supervisor throughout the project duration. This alone should prove that the project has received the final qualifications required to achieve a bachelor's degree in the field of Information and Communication Technology in NHL Stenden.

Furthermore, this project endeavors to alleviate QICT's current challenge of disparate and unstandardized API monitoring within its internal application.

Introduction graduation project:

(Think about the context.)

Q-ICT is a cyber-security consultant company, currently manages numerous third-party APIs without a standardized implementation within its internal applicaton, the QaaS app. The company seeks to establish comprehensive monitoring of these internal APIs, ensuring connection status, error handling, handling of expired API keys, secure storage of keys, and external validation of API connections. Those APIs that the company uses on its internal application include:

- Snelstart: is used for company's automation of financial and accounting sytem software, such as managing invoices, etc,. The file returned by this API is in the form of JSON format.
- N-Central: is used for monitoring the clients' devices. It is a RMM (Remote Monitoring and Management) platform designed to help MSPs (Managed Service Providers) and IT professionals to remotely monitor, manage, and support their clients' IT infrastructure. The file received from this API calls will be in the form of SOAP XML format
- Pax8: is used for their Microsoft subscription. It is a software that helps Q-ICT to simplifiy the way to buy, sell, and manage cloud solutions. The response file returned by this API is in the form of JSON format.
- Bodyguard.io: used for security tab. It is a software developed by a Dutch company to filter and scrutinize downloads from web browsers to detect and prevent malicious files with real-time download scanning capabilities. This API is still relatively newly added by the company, The file received from this API is in JSON format.
- PerfectView: is a CRM (Customer Relationship Management) from a Dutch company for its solution to help manage, track, and store information related to QICT's current and potential customer. The file format returned by this API is in the form of SOAP XML.

Project Scope

The project's focus involves enhancing on the QaaS internal app to serve as a versatile template application capable of executingh the aforementioned functionalities. In addition, the company aims to integrate SentinelOne to the QaaS app, a cybersecurity platform yet to be explored by the company. This platform facilititates continuous monitoring the clients' IT environments for any potential cyber threats, utilizing AI tool to analyze endpoint activities in real-time. Upon successful integration without disruption of existing API calls, the company futher desires insights into cyber threats detected by

SentinelOne from its clients' IT infrastructure. These insights, visualized through graphs and charts connecting the API-sourced data, necessitate QaaS app's ability to interpret SOAP XML and JSON file formats. This will then require the intern to create additional front-end pages in Flutter within the internal app to visualize the SentinelOne integration. Moreover, the company expresses its desires to new ideas, including implementing reading capabilities from YAML, TOML, and CSV file format.

Additionally, the company desires the QaaS app to incorporate error logging functionality. This entails the app being capable of listing and exhibiting all the unsuccessful API calls, displaying their respective status codes and messages. It would be also advantageous that the internal app to be able to conduct testing and debugging to these unsuccessful API calls to diagnose issues such as expired credentials, while also being able to inspect the payload for further analysis.

Voluntarily, at the company discretion and if the project duration permits, they desire that the intern to address potential data integrity issues arising from N-Central API calls, which frequently result in errors of the return data.

Consequently, the intern will undertake research to identify and implement optional practices for integrating API monitoring functionalities into an existing application. The internal app, built in Flutter front-end client with a Node.Js TypeScript as the back end server, already encompases functionalities from GCP (Google Cloud Platform) services like Google Secret Manager, Firebase, Cloud Functions, Firestore, Google Authentication, Cloud Messaging, Cloud Storage, Google Analytics, hosting, real-time database, etc,. Additionally, the intern will have access to both test and live environment for DevOps purposes to facilitate continuous integration. Comprehensive unit testing and meticulous documentation are integral components of this project's requirements.

Description research:

(For example, based on a main question and 3-5 sub-questions research must serve the realization. Clearly describe why it's research.

***When approving the assignment, we do not make substantive statements about the main and sub questions. So, keep in mind that these still must be adjusted during graduation and that the main and sub questions in this description are a first step towards**.)*

The following will describe the main research topic question and sub-question that the intern currently has in mind for his graduation project. These main and sub-questions of course can be later a subject to change after a proper consultation with his school lecturers.

The main question is as follows:

"How can Q-ICT effectively enhance API monitoring within its internal application while integrating and leveraging SentinelOne security threat platform for continuous cybersecurity monitoring while still ensuring adherence to the highest security standards?"

The sub-questions are as follows:

- What functionalities should be prioritized in the development of monitoring and managing third-party APIs within an internal application while ensuring real-time monitoring, error detection, and insight generation regarding API connections?
- How can Sentinel One can be integrated into the Q-ICT environment, specifically aligning with the API monitoring functionality, while still utilizing their key features and capabilities in the context of cyber threat detection and remote IT infrastructure management?
- What are suitable visualization techniques for displaying data processed and received by the internal application in XML and JSON formats that ensure clear and insightful representation of threats detected by SentinelOne and other Q-ICT relevant API connections?
- What is the industry-standard best practices for developing and implementing a standardized API monitoring functionality on top of an existing internal application with measurements for scalability, security, and reliability insurance of the monitoring application?
- What potential impact can the implementation of the proposed solution have on Q-ICT's operational efficiency, cybersecurity posture, and end-users' perception and interaction with the already existing internal application?

Furthermore, the intern plans to utilizing LaTeX Overleaf as his choice of software system for document preparation. Engaging in this research alone presents the intern with a novel and captivating exploration of a complex IT topic, one that he has yet to delve into previously.

Description realization:

(What the student delivers.)

The project's realization phase will involve the creation of multiple software deliverables, primarily encompassing a Flutter-based client front-end application and a Node.Js Express TypeScript-template based back-end server application as they were pre-determined by the client to ensure their continuous development. These will incorporate a middleware and functionalities source from GCP (such as Firebase, Google Secret Manager, Cloud Functions, Firestore, Google Authentication, Cloud Messaging, Cloud Storage, Google Analytics, hosting, real-time database, etc.,) as well as Q-CT internal third-party APIs mentioned above. The access to the internal APIs will firstly be 2 APIs, N-Central and Bodyguard.io, both returning file format of SOAP XML and JSON respectively.

To provide additional background for the explanation, the company is currently encountering issues with executing API calls on their Firebase Cloud Functions, which have a one-hour limitation on execution duration. Occasionally, when retrieving specific data (such as a particular customer's data), an API calls surpasses this time limit. This problem primarily occurs within Pax8, Snelstart, N-Central, and the company foresees a potential occurrence with Bodyguard.io as well in the future. Therefore alternatively, the intern has been tasked with finding a solution to resolve this issue for at least 2 of the internal APIs, collaborating well with his company supervisor.

This also tackles an additional concern, namely the necessity for multiple API calls to obtain a desired single set of data. The company is then open to any solution that addresses this matter, prompting the intern to devise a method that executes a single function within the internal application. This function would need to work autonomously and asynchronously to handle the required additional API calls,

thereby reducing the time spent on manually calling these APIs on the screen. A resolution to this issue would align effectively with the previously mentioned concern.

This project will also implement proper unit testing, code commenting, code refactoring, with proper design and architectural patterns for scaling and proper UI/UX implementation to ensure ease-of-use for users. Furthermore, the intern and his supervisor will be working in SCRUM Agile methodology with their own SCRUM board containing definition of done, product backlogs, sprint backlogs, burndown chart, etc. The intern and his supervisor will also have their daily stand-ups, sprint review, sprint retrospective, and sprint planning concurrently.

Description of complexity:

(What makes it that this assignment is suitable as a graduation assignment so that the student can achieve the final competences here.)

The complexity of this project lies in the integration of SentinelOne into the already existing QaaS internal application to work well with other APIs utilizing multiple technologies (Flutter, Node.Js, and GCP services), which is a novel undertaking for the company. While throughout the project duration the intern will be in direct contact and guidance from the company supervisor; however, both the company and the intern will navigate to new programming activities. Challenges in this project include establishing a robust security protocol as it handles with a crucial confidential data for the company from the internal APIs, designing intuitive UI/UX with effective visualization, and making an informed decision regarding software design and architectural patterns for continuous integration, database environment, API key management, unit testing strategies, integration approach, deployment and DevOps practices, technology (frameworks, libraries, and tools) selection, adaptation to changes, and security measures.

In conclusion, a well-managed work period of 90-100 days should be deemed adequate for this project, considering the proper distribution and management workflow. The project's complexity aligns with the expectations set by the Bachelor of Information and Communication Technology program at a University of Applied Science.

Date completed: 8-December-2023

Appendix G

Interview SentinelOne Transcript

What is SentinelOne and what value does it bring to Q-ICT?

SentinelOne is an EDR and XDR security platform that will protect devices from malware, ransomware, and other kinds of cyberattacks.

EDR (Endpoint Detection Response): is a cybersecurity technology that focuses on detecting and investigating security incidents on endpoints such as desktops, laptops, servers, and mobile devices. It is designed to provide real-time visibility into endpoint activities, detect suspicious or malicious behavior, and facilitate rapid response and remediation.

XDR (Extended Detection and Response): it expands on EDR concept, and integrates new solution like multiple security data sources, including endpoints, networks, cloud environments, and applications, to provide better visibility into security incidents and streamline threat detection and response across the entire IT environment.

SentinelOne has Agents, which is basically an AV (anti-virus) which gives protection against cyber threats. It is a more modernized version of the traditional AV, with the latter having a defined set of vulnerabilities. It utilizes AI and ML to respond to threats in real-time, learning the pattern of an attack in a machine, thus providing behavior-based detection and autonomous protection which the traditional AV (like Avast and Avira) cannot do.

Objectives:

To be able to show the customers their device's status (number of devices protected, endpoints that are infected, get the Site info like last active, device type, resource group it belongs to, device name, OS version, health status, network status, etc.) *See the Sentinel page on SentinelOne dashboard*.

By integrating SentinelOne and having Code Conventions, it will make stronger proof that we are complying with ISO 27000 and NIS2 Directive (an EU guideline to cybersecurity that its member states should adhere to).

What are your desired expectations for the Integration of SentinelOne

3 separate pages

Customer Page:

It should show only the details of their specific devices. A customer cannot see other customers' information about their devices. It should be as user-friendly as possible, implementing graphs, charts, bars, images, and tables to help them visualize. Make it as simple as possible, not

overwhelming them, but display enough data to keep them informed and offering them transparency to their security.

This page can be seen on the sidebar of the QaaS app (in between Mijn Dashboard, Mijn Apparaat, etc.,) The purpose of this page is to show the customer what their device's info and status are (based on their Site ID, which is stored in the database in Firestore) and nothing else. No information about other customers should be displayed, or any irrelevant information regarding them. Do not bombard them with too much data.

Pierre: basically, when a customer logs in, he/she only cares whether their device is infected or not. Nothing more.

Helpdesk Page:

Show more technical details than the Customer. Able to see all the Q-ICT customers' devices and their health. The purpose of this page is to provide the Helpdesk with guidance in case of Cyber Incident happens, it should help and guide the Helpdesk o what to do.

This page can be seen in Klanten page, where the helpdesk would need to navigate through all customer and selecting a specific customer, therefore then passing that customer ID and showing to them what SentinelOne can provide regarding their device's information.

Be able to do Analysis Verdict, it will list all the incidents that have happened that have not yet been reviewed, and the Service Desk can mark them as False Positive, Benign, Mitigated, Unsolved, True Positive, Undefined, Suspicious, Not Mitigated, etc. *See Incident page on SentinelOne dashboard*.

Additionally, if a malicious file was detected on a machine, it should send an email notification to alert the selected users and a warning on the screen when the website is open. Optionally, it should also have features like an alert notification sound.

IT (developer) Page:

See the same technical details as the Helpdesk but with more rights to Read and change the Page (Create, Update, Delete?).

Design Choice

Free to choose, do not forget to bear in mind the factor of UI/UX design for the Customers. Look for some ideas online, for example generative AI image generators (like Midjourney)

Stakeholders

Manuel, Robbert, Pierre, and Mark.

What is the next step:

- Implement some design mockups. Do not spend much time on this. Tools that might be used: Adobe XD, Adobe Illustrator, Figma, or Photoshop
- Create UML Diagrams (Class Diagram, ERD for database, Activity Diagram, Relational Diagram, etc., that are necessary for school)
- Together with Manuel, finish and agree upon the Code Convention about Naming Convention (styling), Code Structure, Code Commenting (documentation), Error Handling, Version Control, Dependency Management, Security, Performance Optimization, testing practices, etc. to promote consistency across the codebase (both in the front-end (Flutter) and back-end (Node.js Firebase)). It also helps with code maintenance, scalability, and testability, making it and to collaborate with other developers for better CI/CD pipeline development.
- Together with Manuel choosing the right Design Pattern (e.g., Repository Pattern, Factory, Singleton, Decorator etc.,), and Architectural Pattern (MVC, MVVM, MVP, etc.,) as the general repeatable solution to a commonly occurring problems in programming and software engineering/designing. It will provide a blueprint that the developers can follow and customize to, to solve various problems from simple to complex issues.
- Also look further for more software development OOP (Object Oriented Programming) concepts/ techniques. like ODM and ORM for NoSQL and SQL databases, that provides abstraction to simplify database interactions and helps in mapping objects to Database entities when dealing with large databases.
- Make User Manual (or README.md file) of the application to help the new interns study and understand the code better and faster.
- Implement Unit Testing for Flutter and Node.js (optional, if time permits).
- Can already start on the coding (make some API calls in the back end, show the data in the front-end, choose which graphs or charts are the nicest for the customer to see) and provide MVP (minimal valuable product) for the stakeholder (Robbert) to see.
- Might need to pause/stop on writing the Thesis sub-question about how to integrate SentinelOne to the QaaS app. All the theoretical information has been stated, might be better to continue writing the Thesis after Realization (coding) part has been done.

- Might need another interview after some duration in the Development if any questions/remarks come out or to see how the progress has been made and check if anything is still in order.

Notes

Robert pointed out for me to just take inspiration and reference from the official SentinelOne pages itself. Might need to study and investigate SentinelOne APIs and the documentation further with Manuel ourselves.

Appendix H

Interview QaaS App Transcript

- Can you provide me with details of what kind of QaaS subscriptions we have? (is QaaS Safe, QaaS Help, and QaaS backup)

Subscription within the QaaS app is divided into smaller proportions (all monthly subscriptions):

- QaaS Safe: security package, patch management, monitoring for workstations or laptop, traditional AV (BitDefender) => will be removed soon as SentinelOne EDR itself is already an upgrade from traditional AV. This subscription is required for every user in a company. SentinelOne payment subscription is also included in this.
- QaaS Safe +: for advance security protection. Bodyguard.io is included. Still a proof of concept
- QaaS Help: for free call service to the helpdesk between working hours. For tracking if a user has this subscription or not financial DB like SnelStart and Computicket is used for billing users who do not have this subscription
- Back-up: for back-ups from the data of the customer from MS 365 Cloud provider. It is for restoring data if lost, stored off-site. It has a collaboration with N-Central

- Can you provide me with details of what kind of Microsoft 365 subscriptions we have? Do we only re-sell Microsoft 365 products to the clients? What makes them want to buy them from us and not directly from Microsoft itself?

We resell Microsoft products from Pax8, a Microsoft provider. In general, you can look at what Microsoft offers for their products' subscriptions, and they will be like us.
They want to buy from us because we also provide support, assistance with installation, we can also directly contact Pax8 for help if some issues occur. Meanwhile, if the customers buy the subscriptions from Microsoft themselves, when they need help, they need to wait in the customer service line for a long time.

- Can you provide me with details of what kind of internet connection subscriptions we have? How do they work?

We offer solutions to the customer in terms of their internet connectivity, we also might get into contact with Internet providers, like KPN and Ziggo.

- Can you provide me with details of what kind of password manager subscriptions we have? How do they work?

For that subscription we are utilizing Keeper, which is a password manager product that requires paid subscription. Under the hood, in which most of us cannot see, Mark and Robbert can compare all the passwords being stored there from a user's account. If there is password duplication, or the passwords themselves are weak in complexity or length, they will advise customers to change their passwords.

- When a customer pays us for our subscriptions, does mean that they also pay subscriptions from SentinelOne, N-Central, Bodyguard.io, SnelStart, Resello (Pax8), and PerfectView (also KPN and Ziggo?)?

Yes

- With all the tools that we have in our disposal, how do we come up to our customers and offer them security protection as a cybersecurity consultant company? Can we be seen as an intermediary between SentinelOne and the customers?

Quality ICT is a cyber-security company. It helps customers with security questions, network providers, to guide our customers with security posture, analysis of a customer like cybersecurity incident response, what is security posture in MS 365, what needs to do to be cyber compliant compliant in the future.

MKBiT is an MSP (Microsoft Service Provider) that provides devices MS 365 for customers, answers all their IT related questions. The Helpdesk is not necessarily equiped with

cybersecurity knowledge, despite being part of Q-ICT, for that they need to go to Mark and Robbert.

- Are there any subscriptions that I missed?

SecureMe2: which is a security solution for security appliances. Like Bodyguard.io, it also comes from a Dutch company. It has a machine that connects to a network desired network, it then scans the whole network if sees suspicious traffic then report to Andre or Robbert, if for example a laptop has connection in botnet, if a malicious USB driver has been plugged into a network (same as SentinelOne Ranger?).

Additional questions

- Because we are a Microsoft company, does that mean the protection that we offer only limited to the Microsoft products? So, for example, if an attack happens but on Unix or MacOS machine, does that fall out of our scope then? Can SentinelOne be installed and working on a device that is not in Microsoft Windows?

Our knowledge is focused mainly on Microsoft. Customers with Unix or Mac OS computers may come and subscribe to us, and we will do our best to answer their cybersecurity questions.
But yes, SentinelOne can in fact be installed in a Unix machine.

- In case of a threat happens that is already not detected and mitigated by SentinelOne and caused problems in a customer's devices, what do we do then? Do one of our helpdesk/ cybersecurity experts comes directly to the client to handle it? What happens when they also do not know what to do? How dependent are we on SentinelOne? Do we have back-up for every customer that we have? Because last time Manuel told me we had over 400 customers, so that would make the back-up data very big.

SentinelOne should pick it up when a malicious threat happens.

Protect: What we do is that we inform the customer, turn off some devices, plug out ethernet, we may come to them to assists physically.
Investigation: we write document about the cyber threat that have happened. How many devices infected, how many data is lost, what kind of lessons that we learn from them.
Remediate

- Also relating to the QaaS app, you said it should send a notification to the specific users in the form of e-mails when a threat happens, do you also want the admin to customize the recipient of these e-mails? Do you also want an alert from the QaaS app itself?

For the Helpdesk, it should be used as a single panel of glass. The QaaS App needs to display what kind of incident happens, so the helpdesk knows what to do with it.

- You also said that for the helpdesk, the QaaS app should also guide them on what to do in case of a cyberthreat happens on a client's machine, can you elaborate more on this? Does SentinelOne API itself provide some sort of guidance to this?

By displaying the overview of the incidents that have occurred, the Helpdesk can then know what steps to take themselves.
Also, we expect that the SentinelOne page to be scalable. Adding new modules to the QaaS app will not be that hard, possible to do so in the interface, and delete old modules without crashing the program.

- Can you provide me with an explanation of what SentinelOne Vigilance is? What sort of implementation do you wish to be done for the QaaS app to have this ability as well?

Vigilance is an additional module for SentinelOne. It is 24/7 SOC, that has employees from another country monitors SentinelOne data during the work off-hours from that

specific country. Because hackers are relentless, they are oftentimes making use of work hours to do their malicious activities.

This is still proof of concept. Mark and Robbert are still investigating about it. So, this is an optional, if not out of scope, part of my graduation project.