Thesis

NHL Stenden University of Applied Sciences

In the department of:

ICT & CT Information Technology Bachelor Emmen

In association with:

Q-ICT B.V.

Written by:

Christopher Sulistiyo (4850025)

Submission:

1-August-2024

Work placement lecturer(s):

Company Supervisor(s): Manuel Weidijk
Mark Kolk

Summary

Contents

List of Figures

Listings

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 (Wikipedia, n.d.-b). . 8

Pentest A.K.A. ethical hacking, is a simulated cyberattack where professional ethical hackers break into corporate networks to find weaknesses (Wikipedia, n.d.-f) . 9

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. (Wikipedia, n.d.-b) . 16

RMM is a software platfomr that helps MSPs remotely monitor and manage their clients' endpoints, networks, and computers (Wikipedia, n.d.-g).. 18

Acronyms

2FA Two-Factor Authentication. 14

A.K.A. Also Known As. 5
AI Artificial Intelligence. 8, 19
APA American Psychological Association. 11
API Application Programming Interface. 5, 8, 10, 12–19

B.V. Besloten Vennootschap. 0, 8, 12
BaaS Back-end as a Service. 14

CDR Content Disarm and Reconstruction. 18
CISO Chief Information Security Officer. 9
CMEK Customer-Managed Encryption Keys. 15
CRM Customer Relationship Management. 5
CSF Cybersecurity Framework. 8
CT Creative Technology. 0

DB Database. 5

etc., et cetera. 14, 16

FCM Firebase Cloud Messaging. 14
FIPS Federal Information Processing Standards. 8

GCP Google Cloud Platform. 14, 15

HTTP Hypertext Transfer Protocol. 14, 16, 17, 19
HTTPS Hypertext Transfer Protocol Secure. 14

IAM Identity and Access Management. 15
ICT Information and Communication Technology. 0, 12
ISO International Organization for Standardization. 9
IT Information Technology. 8–10, 12, 18, 19
ITIL Information Technology Infrastructure Library. 9

JS JavaScript. 14

JSON JavaScript Object Notation. 12, 13, 18

KPI Key Performance Indicator. 19

MCDA Multi-Criteria Decision Analysis. 13

ML Machine Learning. 15

MS Microsoft. 8, 9, 18

MSP Managed Service Provider. 5, 18

NIST National Institute of Standards and Technology. 8, 9

OAuth Open Authorization. 15

OS Operating System. 14

OTP One Time Password. 14

Pentest Penetration Testing. 5, 9

Q-ICT Quality ICT. 0, 8–10, 12–15, 18

QaaS Quality as a Service. 10, 12–16, 18

QL Query Language. 17

REST Representational State Transfer. 5, 16–18

RMM Remote Monitoring and Management. 5, 18

RPC Remote Procedure Call. 18

SIEM Security Information and Event Management. 9

SME Small and Medium-sized Enterprises. 8, 18

SOAP Simple Object Access Protocol. 16, 17

SQL Structured Query Language. 14

SWOT Strengths, Weaknesses, Opportunities, and Threats. 13

TS TypeScript. 14

UI User Interface. 14

URI Uniform Resource Identifier. 16, 17

XML Extensible Markup Language. 12, 13, 16, 18

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, 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 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 measurers, and improve their overall service delivery.

1.2 The Company and the QaaS App

Q-ICT B.V., is a small cybersecurity consultancy based in Emmen, northeast of the Netherlands. It recognizes the critical importance of proactive API monitoring in safeguarding its clients' digital assets. Their customers are SME companies with employees ranging from 1 to 100. Q-ICT is therefore asked to monitor their clients' devices and ensuring the overall security of their systems, IT infrastructure, and digital assets. They typically engage in various activities, including:

- **Continuous Monitoring and Maintenance**: implementing tools and processes for continuous monitoring of clients' systems, devices, networks, and systems to detect and respond to security threats in real-time and address emerging threats and vulnerabilities.
- **Vulnerability Assessment**: conducting regular vulnerability assessments and penetration testing to identify weaknesses in clients' systems and infrastructure
- **Incident Response**: developing and implementing plans and protocols for responding to and mitigating cybersecurity incidents effectively and efficiently.
- **Penetration Testing**: simulating cyberattacks to identify weaknesses in the client's defences and assess their ability to withstand and respond to real-world cyber threats.
- **Security Incident Investigation**: conducting thorough investigations into security incidents to identify the root cause and impact of the incident and develop strategies to prevent future occurrences.

Furthermore, the company also serves as a MS provider....

1.2.1 Working Methodology

The company currently utilizes the five functions defined by NIST as part of its CSF as the framework to help the company manage and improve their cybersecurity risk management processes. These five functions are part of the FIPS 199. All functions serve as level categories for organizing cybersecurity activities within an organization and are as follows (NIST, 2023):

- **Identify**: develop the organizational understanding to manage cybersecurity risk to systems, assets, data, and capabilities. This includes the development of

an organizational understanding to manage cybersecurity risk to systems, assets, data, and capabilities. It also includes the development of a cybersecurity risk management strategy that is aligned with the organization's mission, goals, and objectives and the establishment of a governance structure to ensure that the strategy is effectively implemented and maintained.

- **Protect**: develop and implement the appropriate safeguards to ensure delivery of critical infrastructure services. It can help the company assists clients in implementing security controls, encryption mechanism, access controls, and other security measures to protect their systems and data from unauthorized access, disclosure, and alteration or modification.
- **Detect**: develop and implement the appropriate activities to identify and detect the occurrence of a cybersecurity event in timely manner to facilitate rapid response and mitigation efforts. This includes the development of a cybersecurity event detection capability that is integrated with the company's incident response and recovery capabilities. It will help the company to implement monitoring and detection mechanisms, such as intrusion detection systems, log analysis tools, and SIEM systems, to detect and identify to cybersecurity incidents promptly.
- **Respond**: develop and implement the appropriate activities to act in responding regarding a detected cybersecurity event, containing the impact, and restoring normal operations. It involves activities such as developing incident response plans, conducting incident response drills and exercises, establishing communication channels with stakeholders, and implementing recovery strategies to minimize the impact of cybersecurity incidents on business operations and services.
- **Recover**: develop and implement the appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity incident and event, along with implementing improvements to prevent future incidents. In this activity, the company should be able to develop and implement recovery plans, conduct post-incident reviews and analysis, identify areas for improvement, and implement measures and improvements to enhance resilience and prevention future incidents.

Furthermore, the company also uses the ISO 27001 as the main standard for information security management and the NIST 800-53 as the main standard for security and privacy controls for federal information systems and organizations.

1.2.2 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 subdepartments, 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, mainly the indoor and outdoor customer services. With the 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 principles 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.
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 in his graduation work placement project. Mainly, this department uses Dart with Flutter as the main front-end development framework, and Node.js with TypeScript template as the main back-end development framework. Furthermore, it utilizes Google Firebase as the main cloud solution for the applications it develops as it works hand-in-hand 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.

1.3 Problem Statement

The company currently manages numerous third-party APIs for the above-mentioned purposes. Currently, those APIs are managed manually and without a standardized implementation in their internal application, the QaaS app, which is a time-consuming and error-prone process. This has led to several problems, namely:

- Inefficient and fragmented approach to API management.
- Lack of user-friendliness, and slow and unclear navigation.
- Inconsistent integration of APIs into the application.
- Poses a significant challenge in error handling and debugging, as disparate error reporting mechanism across the APIs hinder efficient troubleshooting and resolution processes.
- Difficulty to maintain and update APIs.
- Lack of clear and concise documentation.
- Lack of a centralized API management system.
- Inadequate security measures, as the company risks inconsistent data retrieval and analysis across its APIs, potentially leading to incomplete insights into client IT systems and infrastructure.

Thus, the company has tasked the author with the development of a standardized API monitoring system within the QaaS app, with features such as error handling, connection status warranty, debugging, and the effective integration of the SentinelOne security threat platform 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. Develop a standardized API monitoring system within the QaaS app, with features such as error handling and logging, connection status warranty, debugging, secure storage of API keys, handling of expired API keys, and external validation of API connections. Additionally, this functionality should also support displaying the respective status code of error connection from the APIs along with error messages, and conducting testing to these unsuccessful API calls to diagnose issues such as expired credentials, while also being able to inspect the payload for further analysis.
2. Furthermore, the QaaS app should be able to serve as a versatile template application for integration with further upcoming third-party APIs in the future for importing new data.
3. Effectively integrates and leverages the SentinelOne EDR platform for continuous cybersecurity monitoring within the QaaS app.
4. The QaaS app should have a way to visualize 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.
5. Ensure proper unit testing, code refactoring, commenting, and adherence to the overall code conventional guidelines and best practices in both the test and live environments of the QaaS app.

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**: 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 Results

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 enhance API monitoring within its internal application while integrating and leveraging SentinelOne EDR platform for continuous cybersecurity monitoring while still ensuring adherence to the highest security standards?"

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:

- What is the current situation of the QaaS app of Quality ICT B.V.?
- What functionalities should be prioritized in the development of monitoring and managing third-party APIs within the QaaS app while ensuring real-time monitoring, error detection, and insight generation regarding API connections?
- How can SentinelOne be integrated into the QaaS app environment, especially aligning with the API monitoring functionality, while still utilizing their key features and capabilities in context of cyber-threat detection and remote IT infrastructure management?
- What are the most suitable visualization techniques for displaying the data processed and received by the QaaS app in XML and JSON formats to ensure clear and insightful representation of threats detected by SentinelOne API?

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 of 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.

- Sub-question #2: Literature Study on various articles on the Internet, interviews, expert reviews, and requirement elicitation techniques such as use case analysis and user stories. Analysis on the current QaaS app and its API monitoring capabilities.

- Sub-question #3: technical evaluations of SentinelOne's capabilities and APIs will be conducted, the API documentation and integration guideline will be read and review with Literature Study method and case studies of similar integrations. Requirements from the cybersecurity experts from the Q-ICT department responsible for SentinelOne's technical support will be gathered and analyzed. Furthermore, Prototyping with proof-of-concept prototypes on a test environment will be conducted to test different integration scenarios, assess feasibility, identify potential challenges, refine the approach, and evaluate the performance of the integration.

- Sub-question #4: research into existing visualization techniques for XML and JSON data, especially the data coming from SentinelOne API 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 data visualization tools, literature review through reading the studies of the best visualization suitability matrix techniques, 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 panel, 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 visualiza-

tion 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.

2.2.5 Research Limitations

The project and research in general will be limited on the API request methods, in which the author is allowed to do only GET requests. This is due to the fact that the author is not allowed to do any PATCH, POST, PUT, DELETE, or any other HTTP request methods that could potentially change the state of the QaaS app or the API that is being requested. This limitation is because the author is not a full-time employee of Q-ICT and is not allowed to make any changes to the QaaS app or the API that is being requested. Therefore, the author is limited to do research in the best practices and possible answer for API monitoring and integration for the GET request method only.

Moreover, the author is also limited to the non-disclosure agreement signed within the initialization period of the graduation work placement. This means that any confidential information that the company deems as confidential will not be disclosed in this research. This includes any information that is not publicly available, such as any financial data or security data pertaining to the internal system or the QaaS app internal code.

2.3 Research Sub-Question #1

The QaaS app is an ERP web application that is used by Q-ICT and its clients. For Q-ICT's clients, it is a SaaS that is used.... For Q-ICT itself, it is an ERP system that is used to manage the clients and their ICT infrastructure. It is made in Dart with Flutter as the front-end framework. 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.

2.3.1 QaaS App Infrastructure

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 (Wikipedia, n.d.-d).

Cloud Solutions

Firebase

Firebase is a comprehensive platform for developing and managing web and mobile applications, created by Google and is party of GCP. It was originally an independent company founded by Firebase, Inc. in 2011. It was then acquired by Google in 2014. Since then, it has become an integral part of Google's broader ecosystem of cloud services (Wikipedia, n.d.-c).

Firebase 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 including (Firebase, n.d.)

- Databases:
  - 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.
  - Real-time database:
- 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.
- 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 are in this version. The company wishes to migrate all the functions to the 2nd generation in the future.
  - 2nd Generation: The company wishes that the author's graduation project will utilize the 2nd generation of Cloud Functions. Features in the 2nd generation including:

* Longer request processing times
* Larger instance sizes
* Traffic management
* Eventarc integration
* Broader CloudEvents support

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 2.1: Comparison between the 1st and 2nd Generation of Cloud Functions

- 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 Messaging/FCM: allows developers to send push notification and targeted messages to a client app that new email or other data is available to sync, or send notification messages to drive app interaction, adding more user re-engagement and retention. It is a cross-platform messaging solution that lets developers reliably deliver messages at no cost.
- Cloud Storage: offers a 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.
- Remote Config: allows developers to dynamically control and the change behaviour and appearance of their apps without publishing app updates and requiring users to download them.
- Performance Monitoring
- Analytics: provides free and unlimited analytics solutions for understanding app usage, user engagement and user behaviour by tracking event logging, user demographics, and funnel analysis to gain valuable insights to improve the app.
- Crashlytics: is a lightweight, real-time crash and error reporter that helps developers track, prioritize, and fix stability issues that erode to the quality of the developer's app.
- Test Lab: enables automated testing of the developer's app on real devices in the Google data center.
- Dynamic Links: consist of deep links that dynamically route users to the content they are interested in, across platforms and devices. It helps developers create and share links that work the way they want, on the platform they want, and whether users have their apps installed.
- ML Kit: provides a set of ML APIs that can be easily integrated developer's app. This feature is still in beta testing.

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 applicaitons 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 supports 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 Compactibility: it is designed to be backwards compactible 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.

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 (Wikipedia, n.d.-a). 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.

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.

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, OAuth credentials, DB credentials and other credentials used in throughout the lifecycle of their applications (Google, n.d.). 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

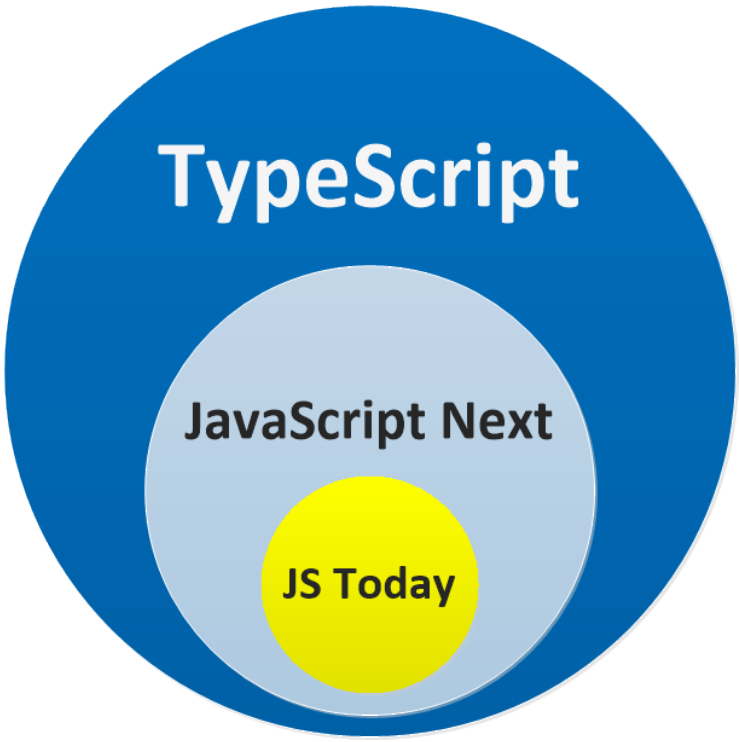

Figure 2.1: Venn diagram of TypeScript and JavaScript

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.

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 RDMS. 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 (MongoDB, n.d.):

- 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 in order 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.

”Serverless Architecture”

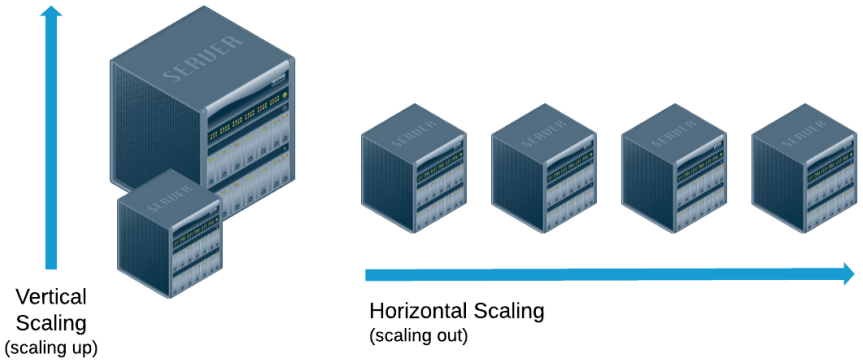

Figure 2.2: Vertical (SQL) vs. Horizontal (NoSQL) Scaling

Some examples of NoSQL databases include MongoDB (document-oriented), Cassandra (wide-column store), Redis (key-value store), and Neo4j (graph database). Some examples of SQL databases include MySQL, PostgresQL, and MS SQL Server.

2.3.2 Q-ICT Internal APIs

What is an API?

API is a software intermediary that allows two applications to talk to each other. They are an accessible way to extract and share data within and across organizations.

Web APIs

A web API is an API that can be accessed using the HTTP protocol. Not all APIs are web APIs; some APIs are used only to communicate between two applications on the same computer, never making use of a web connection. But in practice, when developers talk about APIs, they are almost always talking about web-based APIs used to

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 2.1: 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"
        }
    }
```

XML format 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. It is often used for store, design, and representing structured data in a portable and platform-independent way, making it easy to create, exchange, and process information between different systems, applications, and platforms.

XML documents consist of text-based data enclosed in tags, similar to HTML, but allows users to define their own customized tags and document structures. This flexibility makes XML suitable for representing a wide variety of data types and structures.

XML is often used in various domains such as web services, configuration files, data interchange between different software systems, and more.

Listing 2.2: Example of a XML response

```
<response>
    <status>200</status>
    <message>OK</message>
    <data>
        <user>
            <id>123</id>
            <username>johndoe</username>
            <email>johndoe@example.com</email>
            <role>user</role>
        </user>
        <user>
            <id>456</id>
            <username>janedoe</username>
            <email>janedoe@example.com</email>
            <role>admin</role>
        </user>
    </data>
</response>
```

OpenAPI Standard Previously known as Swagger, OAS is a specification for writing a public API, with guidelines for details like endpoint naming conventions, data formats, and error messaging (Swagger, n.d.). The standards required by OpenAPI and its automation of some tasks make it easier for a developer to start working with an API without needing to read through a complex code base. For API producers, OpenAPI standard offers access to a wide variety of tools based on the standards. API teams can use these tools to quickly up mock servers and create high-quality documentation, among other tasks. APIs are generally categorized into different types based on their audience, architecture, and protocols (Bahl, 2022):

Different Types of API By Audience

Public API: also called external or Open API, and as the name suggests it is available to everyone. They are open for the public to use and integrate with their applications. Developers can quickly implement them using little to no authorization, with few require sign-up and generation of the API Keys to access them. This can make them to not be the best regarding security - just because ”public” means expanded visibility - but sharing data with them is easier.

Examples of a public API are OpenWeatherMap, Google Maps navigation, Facebook and the Twitter API, which the latter allows developers to access Twitter’s functionality and data.

Internal API: also called Private API, and is used within a private organization to make internal apps ”talk” to each other. To interact with the data, a developer needs to be actively granted permission to access it, because the data and functionality available through the API are proprietary to the company. They are often set up with extensive logging and load-balancing capabilities because they must have greater fault tolerance and security than public APIs. They also do not follow the OpenAPI standard as consistently as public APIs, since their producers and consumers typically work together closely, data formats can be negotiated based on specific use cases. As they are built by specifically the company; it will only have API protocol types that the organization wants to support. All the APIs managed by

the QaaS app fall into this category. This solution tend to be very secure, as they are entirely internal.

Partner API: also called Shared API, this API is made considering the scalability while developing the business, which will share a few APIs across a few other licensed organizations, enabling service offerings across business (B2B). This API is shared only with the intended users; others might not have access to them because they are not shared publicly, thus making it exists somewhere between public and private APIs. They often function to share data between two companies or organizations for a specific business purpose, while still ensuring strict privacy protection.

These APIs indeed require authorization to access them (like having a PayPal account or an API key). All the clients who are part of the business can access and integrate using those APIs. Few APIs will only provide read access, and few will provide read/write access via shared APIs. This depends on the business process model.

For example, travel booking APIs are shared with travel agencies to increase their visibility and booking. Websites like Expedia, Make My Trip, and Trivago are excellent examples of this kind of API.

Composite API: it connects multiple APIs into a single data or interface or process to streamline the development operations, therefore allowing developers to bundle and send requests from different APIs as they do not have to write separate code for every individual API. Usually, one or more related APIs are combined to improve performance. They reduce the load to the server as they are treated as a single call. In the microservice-based architecture (*Different Types of API by Architecture*), a single user action might involve multiple calls, which can come as a drawback as they generate enormous number of individual API calls. In that scenario, the orchestration of this type of API is very helpful as this work faster and is easier to implement. It is more secure than using multiple, individualized solutions, and it may also support multiple API protocol types.

A good example of this type is Shopify API, which provides synchronization to large volumes of data and actions with other platforms that are dependent on each other in one request, such as Etsy, eBay, and Amazon. Rather than having multiple APIs to manage the automation of inventory, shipping, and taxes across multiple shopfronts, a single API can be used for syncing and integration. Similar in ordering via a food app, the customers can use multiple GET calls, retrieving the food, and finally placing the request as a POST call.

Different Types of API by Architecture

Monolithic API: this is a single, coherent codebase type of architecture, providing access to a complex data source. Most public APIs are monolithic APIs, because it is familiar to most web developers, and they often closely follow the MVC architecture of a relational DB. They provide predictable functionality across a range of resources, and they generally remain fairly stable over time because they serve so many use cases for many users.

However, as the name suggests, they are monolithic, and therefore can be difficult to scale or refactor, because so much data is interconnected with them. When developers worry about releasing ”breaking changes”, they are often working with monolithic architectures, where changing even minor details can have unpredictable consequences.

Microservices API : this is the main alternative to monolithic API architecture. This architecture is more common for internal and partner API, though public API may also be part of an organization’s overall microservice architecture. Most development teams using a CI/CD process make use of many microservices as part of their code lifecycle, each serving a discrete, independent purpose. For example, an e-commerce company might have an internal microservice that provides inventory data, and another to validate employee geolocation on changes to inventory data, while software developers pushing code automatically call microservices for testing and governance. As workflows change, individual microservices can be swapped out, updated, or replaced without affecting the overall parts of the system.

Unified API: this type is common among Partner APIs. It serves similar purpose to Composite APIs, but instead it bundles related calls to multiple different APIs, instead of to multiple endpoints on a single API.

Different Types of API Protocols

SOAP APIs: are strictly based on XML for the message structure and HTTP for the protocols. SOAP itself is a protocol and sending a SOAP request is similar to using an envelope to send a message. SOAP APIs consume extra overhead and more bandwidth, and require more work on both the client and server ends. That being said, like envelopes, SOAP encloses more stringent security compared to REST. XML-encoded SOAP messages use the format defined below:

- **Envelope**: the root element of the message, which encapsulates the entire SOAP message. It ’envelopes’ the message by placing tags at the start and the end.
- **Header (optional)**: defines specific additional message requirements, such as authentication.
- **Body**: the request of response is included here.
- **Fault (optional)**: information about errors that might arise during the execution of the API call or response is highlighted here, along with information on how one can address these errors.

Listing 2.3: Example of a SOAP request

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:example="http://example.com">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <example:GetUser>
            <example:UserID>123</example:UserID>
        </example:GetUser>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

REST API: if SOAP is like an envelope, REST is 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 (*see REST*).

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 (Fielding and Taylor, 2000):

- **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 (Hazaz, 2022). 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 2.4: 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 2.5: REST's Example Request

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

Listing 2.6: Example of REST request in JavaScript

```
const apiUrl = 'https://api.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
```

All three of the REST, SOAP, and GraphQL use the HTTP protocol for communication therefore falls into HTTP APIs category. They are the commonly used for web services and allow applications to interact with each other over the internet. HTTP is superbly suited for applications following a request-response paradigm.

REST	SOAP
Works with XML, JSON, HTTP and plain text	Works with XML by design
Loose and flexible guidelines based on architectures	Strict, clearly-defined rules
Modest security	Advanced security
Works well with data	Works well with processes (actions)
Uses low bandwidth and is highly scalable	Uses more bandwidth with limited scalability

Table 2.2: Comparison of REST and SOAP

GraphQL API: is contract-driven and come with introspection out-of-the-box. It was developed by Facebook for internal purposes in 2012, and it was open-sourced in 2015. Now it is maintained by the GraphQL Foundation (Wikipedia, n.d.-e). Building an API with GraphQL is very easy in comparison to true REST APIs, which require extensive knowledge of HTTP to build intelligently.

The downside is, however, that they do not scale well and require tight coupling between the client and the server. GraphQL queries get more expensive to parse and execute plans for as they get bigger and lack certain concepts native to HTTP, such as content and language negotiation.

Listing 2.7: GraphQL's Schema Example

```
type Query {
     user(id: ID!): User
     posts: [Post!]!
}
type User {
     id: ID!
     name: String
}
type Post {
     id: ID!
     title: String!
     body: String!
}
```

Listing 2.8: GraphQL's Request Example to Specific Data

```
{
     posts {
          title
          user(id: "123") {
               name
          }
     }
}
```

Listing 2.9: GraphQL's Return Data Example

```
{
     "data": {
          "posts": [
                    {
                         "title": "My first post",
                              "user": {
                                   "name": "John Doe"
                              }
                    }
               ]
     }
}
```

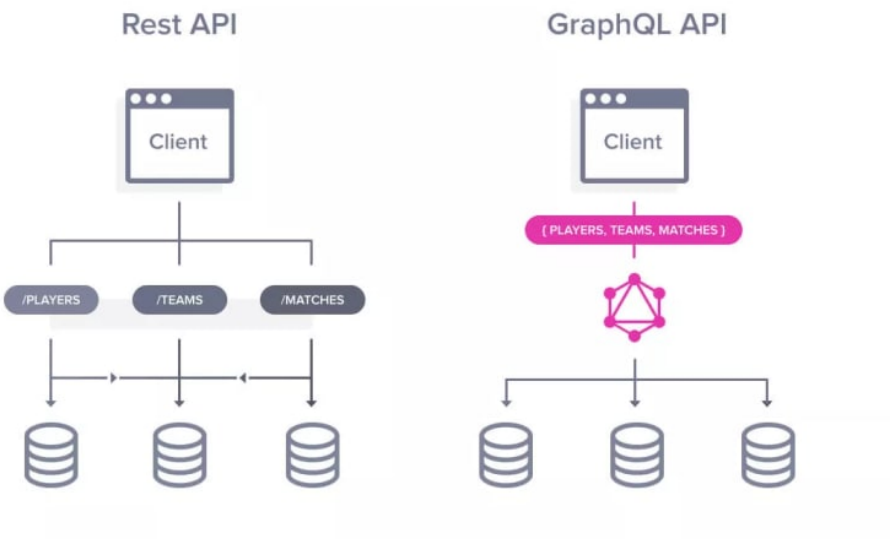

Figure 2.3: GraphQL vs. REST Architecture

RPC API : RPC is a protocol return XML or JSON response. This protocol calls a method rather than a data resource, and remains one of the oldest and most stable protocols for APIs. While RESTful API returns a document, the response from a RPC server is a confirmation that the function was triggered, or an error indicating why it failed to run.

RPC is much faster than REST, but the specifics depend on implementation. Unlike REST or SOAP, the message format varies. RPC is tailored toward a client-server architecture and generally over a network.

Components of a RPC system:

⋆ Client: the requesting device.

⋆ Client stub: how the client will pack/unpack its materials.

⋆ RPC runtime: the messaging system (a courier between the client and server).

⋆ Server stub: how the server will pack/unpack its materials.

⋆ Server: the supplying device.

The popular frameworks of RPCs are Apache Thrift, gRPC, and JSON-RPC, and XML-RPC.

gRPC: developed by Google and released to public in 2015 to use, gRPC is an open-source RPC architecture that can operate in numerous environments.

The gRPC transport layer primarily relies on HTTP. The ability for developers to specify custom functions that allow for flexible inter-service communication is a significant feature of gRPC. This API protocol also offers extra features such as timeouts, authentication, and flow control.

In the gRPC protocol, data is transmitted in protocol buffers, a platform and language-agnostic mechanism that allows for data to be structured intuitively. This mechanism defines the service and then the data structures that the service will use. Compiling is taken care by protoc, the protocol buffer compiler.

The output of this process is a comprehensive class containing the user's defined data types and basic set of methods in the chosen development language. Users can implement in-depth API operations using this class.

Listing 2.10: gRPC Service For a Calculator In a Protobuf File

```
syntax = "proto3";

package calculator;

service Calculator {
     rpc Add(AddRequest) returns (AddResponse);
     rpc Subtract(SubtractRequest) returns (SubtractResponse);
}

message AddRequest {
     int32 a = 1;
     int32 b = 2;
}

message AddResponse {
     int32 result = 1;
}

message SubtractRequest {
     int32 a = 1;
     int32 b = 2;
}

message SubtractResponse {
     int32 result = 1;
}
```

Apache Thrift: developed by Facebook, Thrift itself is a lightweight, language-agnostic software stack. This API protocol supports HTTP transmission, along with binary transport formats. Thrift is capable of clean abstraction and implementations for data serialization and transport and application-level processing. Its primary objective is point-to-point RPC implementation.

Apache can support 28 programming languages, allowing programs written in any of these languages to communicate with each other and request remote services using APIs. **WebSocket**:

Listing 2.11: WebSocket's Example

```
// Client-side code
const socket = new WebSocket('wss://example.com/socket'); // Replace with actual server's websocket URL

// Event handler for when the connection is established
socket.addEventListener("open", (event) => {
    console.log('Connected to the server');
    // Send data to the server
    socket.send('Hello, server!');
});

// Event handler for incoming messages from the server
socket.addEventListener("message", (event) => {
    console.log(`Message from server: ${event.data}`);
});

// Event handler for when the connection is closed
socket.addEventListener("close", (event) => {
    console.log('Connection to the server closed');
});

// Event hander for handling errors
socket.addEventListener("error", (event) => {
    console.error(`An error occurred: ${event.message}`);
});
```

Socket: is a software abstraction that allows programs running on different devices to communicate with each other over a network. It provides a standard interface for network communication, enabling data to be sent and received between applications running on separate computers. Sockets are commonly used in networking applications to establish connections and exchange data.

Listing 2.12: TCP Server Example Using Sockets in Python

```
import socket

# Create a TCP/IP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the address and port
server_address = ('localhost', 8080)
server_socket.bind(server_address)

# Listen for incoming connections (max 5 clients in the queue)
server_socket.listen(5)

print('Server is listening on port 8080 for incoming connections....')

while True:
    # Wait for a connection
    client_socket, client_address = server_socket.accept()

    try:
        # Receive data from the client
        data = client_socket.recv(1024)
        print(f"Received data from {client_address}: {data.decode('utf-8')}")

        # Send a response back to the client
        response = 'Hello, client!'
        client_socket.sendall(response.encode('utf-8'))

    finally:
        # Clean up and close the connection
        client_socket.close()
```

MsgPack: is an open standard for compact binary data serialization, ideal for efficient data transfer. MsgPack supports a variety of data types, including integers, floating-point numbers, strings, arrays, maps (key-value pairs), and more. It is designed to be platform-agnostic, meaning that the data can be serialized in one programming language and deserialize it in another without compactibility issues.

Listing 2.13: MsgPack Serialization and Deserialization in Python

```python
import msgpack

# Creating a Python dictionary to represent some data
data = {
     'name': 'John Doe',
     'age': 30,
     'is_student': False,
     'hobbies': ['Reading', 'Gaming', 'Traveling'],
     'address': {
          'street': 'Sesame Main Street',
          'city': 'New York',
          'zip': '10001'
     }
     "scores": [90, 85, 95, 100, 95, 88, 72]
}

# Serialize the data to a MsgPack binary format
packed_data = msgpack.packb(data)

# Deserialize the MsgPack binary data back to a Python object
unpacked_data = msgpack.unpackb(packed_data)

# Print the original data and the deserialized data
print("Original data:", data)
print("Unpacked data:", unpacked_data)
```

With that being said, the QaaS app needs to manage and make connection different sort of APIs. Those APIs are the following:

- Resello: is used for Q-ICT MS subscriptions owned by Pax8. It is a cloud marketplace that simplifies the way SMEs 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.
- 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.
- 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 remote monitoring and management, patch management, antivirus, backup and disaster recovery, and network topology mapping.

These different APIs will be discussed further in the next sub-question.

2.4 Research Sub-Question #2

2.4.1 What is API Monitoring?

API monitoring is the process of gathering, visualizing, tracking, analyzing, and alerting on the performance, availability, and the API telemetry data to ensure that API requests are handled as expected.

2.4.2 How Does API Monitoring Work?

API monitoring automatically checks API performance and availability at regular intervals, to ensure that the API runs appropriately. This can be done in a few different ways, depending on the type of the API being monitored 2.

For example, let's take Postman, a popular free API

monitoring tools that allows developers to easily monitor, analyze, and debug their APIs. In this example, the system might look for common errors, such as 500-level responses or timeouts, when making requests to the API. It also checks for latency issues or sudden spikes in traffic that could indicate potential system problems. In addition, this monitoring system can be configured to track specific metrics such as total requests, response times, and other KPIs over time. This will then provide real-time insights into API performance and offers a range of features such as automated alerts, detailed analytics, and comprehensive reporting capabilities. Features that an API monitoring might have are listed in the following (Postman, 2024):

- **Endpoint Surveillance**: it begins by closely tracking the various endpoints of an API system. These endpoints represent specific functionalities or resources that the API provides.
- **Request-Response Analysis**: monitoring tools can simulate API requests by sending predefined inputs and parameters to specific endpoints, then analyzing the responses for relevant factors such as response time or data accuracy.
- **Performance Metrics Measurement**: KPIs, such as response time, latency, and error rates, are measured and tracked over time. This metrics offers insights into the overall health and efficiency of an API.
- **Error Detection and Logging**: actively identifying and log any errors or anomalies in API resources with an API monitoring program. This includes capturing HTTP error codes, unexpected data/file formats, and alerting of any deviations from expected behaviour.
- **Security Checks**: API monitoring assess API transactions for unauthorized access attempts, potential vulnerabilities, and adherence to security protocols.
- **Alerting and Notification System**: automated alerting systems can be configured with high levels of customization to notify relevant stakeholders when specific API performance thresholds are breached or exceeded or when abnormal behaviour is detected.
- **Usage Analysis Gathering**: collect usage analytics for insights into how a given API is being used. This data helps software organizations plan for scalability, optimize resource allocation, and understand user behaviour.
- **Logging for Auditing**: detailed logs of API interactions are maintained for auditing purposes. These API logs are valuable resources for post-incident analysis and tracking historical performance and behavioural trends.
- **Continuous Monitoring**: the tool is a constantly ongoing, real-time process that ensures that issues of any scope are identified and addressed promptly.

With those functionalities in mind, API monitoring tools can then automatically check API performance and availability at specified intervals of time to ensure that the monitored APIs run appropriately. The specific timing of those intervals is unique to each API product.

2.5 Research Sub-Question #3

2.5.1 SentinelOne

SentinelOne is a cybersecurity platform that provides endpoint protection, detection, and response capabilities to help organizations defend against advanced cyber threats. It leverages Artificial Intelligence 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. 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.

2.5.2 SentinelOne Console

2.5.3 Ranger

Ranger is one of the SentinelOne product that provides a way of detecting other devices (computers and IoT devices) that are on the client's computer network. 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 this is a new device, or a rogue device. As long as a device has Ranger on that network subnet, SentinelOne can gather and detect technical information regarding the device. On a network, before a machine is connected and talks to other devices and gateways, it is going to do a broadcast and gives up information about itself. This is called an ARP request. Ranger is going to read that ARP request and determine

2.5.4 Sentinels

Sentinels are the end-points,

2.5.5 SentinelOne Agent

An Agent is a software program, deployed to each endpoint, including desktop, laptop, server or virtual environment, and runs autonomously on each device, without reliance on internet connection.

2.5.6 Vigilance

It is a MDR service - providing threat monitoring, hunting, and response, to its existing customers. It provides

a 24/7 SOC with expert analysts and researchers to give customers near real-time threat monitoring, in-console threat annotations, and response to threats and suspicious events.

2.5.7 Incidents

2.6 Research Sub-Question #4

Chapter 3

Realization

Chapter 4

Conclusion and Recommendation

Bibliography

Bahl, J. (2022). Api types and protocols. *Stoplight*. https://stoplight.io/api-types

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

Firebase. (n.d.). Firebase products - google. *GCP, Google Cloud Platform*. https://firebase.google.com/products-build

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

Hazaz, Y. (2022). Understanding rest apis: Key principles and best practices explained. *Amplication*. https://amplication.com/blog/rest-apis-what-why-and-how/

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

NIST. (2023). The five functions. *NIST Government*. https://www.nist.gov/cyberframework/online-learning/five-functions

Postman. (2024). Monitor health and performance of your apis in postman. *Postman Learning Centre*. https://learning.postman.com/docs/monitoring-your-api/intro-monitors/

Swagger. (n.d.). Openapi specification. *Swagger.io*. https://swagger.io/specification/

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

Wikipedia. (n.d.-a). Algolia. *Wikipedia, free encyclopedias*. https://en.wikipedia.org/wiki/Algolia

Wikipedia. (n.d.-b). Api. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/API

Wikipedia. (n.d.-c). Firebase. *Wikipedia, free encyclopedias*. https://en.wikipedia.org/wiki/Firebase

Wikipedia. (n.d.-d). Flutter (software). *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Flutter_(software)

Wikipedia. (n.d.-e). Graphql. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/GraphQL

Wikipedia. (n.d.-f). Penetration test. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Penetration_test

Wikipedia. (n.d.-g). Remote monitoring and management. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Remote_monitoring_and_management

Appendix A

Planning

Appendix B

Project Plan

Appendix C

FO (Functional Overview)/SRS(Software Requirements Specification)/PRS (Product Requirements Specification)