

Quality ICT B.V.

Client

Technical Design

Christopher Sulistiyo (4850025)

Christopher.sulistiyo@student.nhlstenden.com

ICT & IC Information Technology Department Emmen

Version 1.1 – 21/02/2024

# 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

[Version control 2](#_Toc168395982)

[Remarks 2](#_Toc168395983)

[Chapter 1 Introduction 4](#_Toc168395984)

[Background and Context 4](#_Toc168395985)

[Additionally, it ensures that all team members have a clear understanding of project’s 4](#_Toc168395986)

[Chapter 2 System Requirements 5](#_Toc168395987)

[Client-Side Requirements (Flutter Web App) 5](#_Toc168395988)

[Chapter 2 – Architecture Overview 6](#_Toc168395989)

[Schemas 6](#_Toc168395990)

[Chapter 3 – Technology Stack 9](#_Toc168395991)

[Schema 9](#_Toc168395992)

[Deployment Plan 9](#_Toc168395993)

[Risk Assessment 9](#_Toc168395994)

[Mitigation 9](#_Toc168395995)

[Chapter 4 – System Components 10](#_Toc168395996)

[Technologies Used 10](#_Toc168395997)

[Database Design 11](#_Toc168395998)

[Back-end Development 12](#_Toc168395999)

[Front-end Development 13](#_Toc168396000)

.

# Chapter 1 Introduction

## 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 artifact 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

# Chapter 2 System Requirements

## Client-Side Requirements (Flutter Web App)

There are several user 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

## Server-Side Requirements (Firebase and Node.js)

* Firebase Cloud Functions:
  + **Version:**
  + **Memory Allocation:**
  + **Timeout Settings:**
  + **Dependencies:**
* Node.js Server:
  + **Runtime Environment**: Node.js 14 or higher.
  + **Memory and CPU**: sufficient allocation based on the expected load (2 vCPUs and 4 GB of RAM as a minimum requirement).
  + **Storage**: ensure adequate storage for logs and temporary data.
  + **Dependencies**: required Node.js such as `axios`

Additionally,

# Chapter 2 – Architecture Overview

## Schemas

A diagram of a flowchart

Description automatically generated

Figure 2 Flowchart Diagram

A diagram of a diagram

Description automatically generated

Figure 3 UseCase Diagram

A diagram of a computer

Description automatically generated

Figure 4 QaaS App Sequence Diagram

A diagram of a project

Description automatically generated

Figure 5 How XDR and EDR work together on a user's machine Sequence Diagram

# Chapter 3 – Technology Stack

# Schema

## Deployment Plan

## Risk Assessment

## Mitigation

# Chapter 4 – System Components

## Technologies Used

Algolia

N-Central API

Firebase Cloud Functions

Firebase Firestore

Flutter

Mock-up test framework

Firebase cloud function console logging

In flutter, try catch

Cost (how much it costed)

Security rules playground, test security rules

# Database Design

**How the collection is stored**

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.

**Data Types**

* Endpoints: this type of
* Threats
* Applications
* Miscellaneous: contains SentinelOne news feed article and customizable notes.

**Collection types**

There are 2 types of collection that are used throughout the project

* SentinelOne Data:
* 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

**Index types**

Moreover,

* 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

# Back-end Development

As was stated in the Research Report, Q-ICT wishes to utilize the 2nd generation of Firebase Cloud Functions to integrate SentielOne to the QaaS App, therefore a new Firebase project was created and stored on the cloud using Azure DevOps. The author also needs to decide 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` 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 behaviour 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 \texttt{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.

# Front-end Development