Quantified Student

Transfer and general documentation

Koen Janssen (451394)

Jelle Maas (479146)



# 

# Abstract

This document serves as documentation of the Quantified Student project.

# 

# 

# Version History

| **Version** | **Author** | **Date** | **Description** | **Reviewers** |
| --- | --- | --- | --- | --- |
| 0.1 | K. Janssen | 13.06.2022 | Setup document |  |
| 0.2 | K. Janssen | 23.06.2022 | Added Front-end and Back-end section | J. Maas |
| 0.3 | J. Maas | 28.06.2022 | Add mobile, smartwatch and Huawei Link protocol sections |  |
| 0.4 | K. Petra | 28.06.2022 | Add technical document |  |

# 

# Table of Contents

[**Abstract**](#_8qp6tm2436fr) **1**

[**Version History**](#_ljbasbi3kqdt) **2**

[**Table of Contents**](#_6rh431mk6smz) **3**

[**1 Introduction**](#_1248eu7394b5) **4**

[**2 Dashboard**](#_fjmvsm4bvnf1) **5**

[2.1 LTI](#_khyc2kxahxxm) 5

[**3 Web API**](#_kkms6qaigdd7) **5**

[**4 Mobile**](#_7r79kmhphck4) **6**

[4.1 Smartwatch](#_pcp4s6brvb9l) 6

[4.1.1 Huawei Link protocol](#_u23x6n5atphb) 6

[4.1.2 Huawei Health database](#_5ef77168xtoc) 6

[**5 Documentation**](#_ns32zyzwqyg) **6**

[5.1 Technical Specification](#_o4qanz8d2j60) 6

# 1 Introduction

The Quantified Student (QS for short) project focuses on helping students with their development and optimising their performance with the help of collected data. The collected data will be shown in a dashboard where the student can see it. Afterwards, the student can conclude where and how to improve their workflow. For example, the system can show when it is the best time to work for the student.

# 2 Dashboard

For the Quantified Student Dashboard, we are using the Next.js library, which is a derivative of ReactJS. Next.js comes with some additional features out of the box such as Typescript, SSG, SSR and a compiler that is written in Rust for optimized build speeds.

## 2.1 LTI

After developing a prototype and multiple issues, it has become clear that an LTI integration is something that is now not feasible to have integrated into the application. For now, to add the QS dashboard inside Canvas with the help of another LTI called ”Redirect Tool”. This LTI lets the user integrate an Iframe into the canvas course. This Iframe can be used to integrate a QS dashboard. This is for now the best way to have a working prototype.

Besides this, The LTI development is indeed an issue and needs a lot of work. For the next iteration of QS, there needs to be emphasise that this is something for two people or more to work on. You can't get this working just alone. There are too many moving parts to get this working.

# 3 Web API

The Quantified Student central API uses GraphQL, and Express.js with Apollo Server and is written in JavaScript. The substantiation of the above-mentioned technologies and framework choices can be found in the data mediator and technical document. The source code of the API can be found in the public GitHub repository.

GitHub repository:

<https://github.com/quantifiedstudent/web-api/tree/apollo>

# 4 Mobile

The Android version of the mobile application is developed in Kotlin, using the native toolkit. The purpose of this mobile app is to provide access to any data source which requires a mobile device due to hardware or usability restrictions. For example, connecting to a smartwatch using Bluetooth.

## 4.1 Smartwatch

The mobile app currently supports communication with compatible smartwatches, meaning it can request data from a particular smartwatch. The extracted data is then sent to its own API, since the smartwatch is seen as a separate data source. In conclusion, the application functions as a middleman between the smartwatch and the data collection endpoint.

The team intends to develop an iOS version of the mobile application if user testing indicates that the mobile application would be beneficial to the system.

The source code of the mobile app can be found at the following GitHub repository: <https://github.com/quantifiedstudent/mobile-android>.

### 4.1.1 Huawei Link protocol

The smartwatches which were selected implement the Huawei protocol, and are manufactured by Huawei. The Huawei Link protocol was reverse engineered in order to allow connection to the smartwatches using the mobile application. The reverse engineering process was partially successful in that we are able to connect to the smartwatch and gather a small amount of arbitrary data.

The Huawei smartwatch is currently able to be paired with, however it times out after a certain time. The team believes that this is due to a missing “set date time” packet, which must be encrypted before transmission. The packet can be sent, but issues arise when trying to encrypt it. The team has been unable to resolve the encryption issues thus far, however a solid foundation has been put in place which should make it easier to resolve the issue.

### 4.1.2 Huawei Health database

Due to us not being able to finish the full Huawei Link protocol, we’ve opted for a different approach to get the data. The Huawei health app saves all the biometric data in a file called hihealth\_003.db. This file is only accessible if you have a rooted phone, However the file is encrypted, and it gets encrypted on runtime. With the use of Frida, we can let the application decrypt the database files, and then we can read it out like any other DB file.  
<https://github.com/quantifiedstudent/huawei-data-extraction>

# 5 Documentation

The documents of the Quantified Student project can be found in the GitHub repository at the following link: <https://github.com/quantifiedstudent/docs>.

## 5.1 Technical Specification

The technical specification for the API can be found at