Quantified Student

Multi-protocol architecture supplemental document

Jelle Maas*

Fontys University of Applied Sciences

Eindhoven, Netherlands

Dated: May 10, 2022

*E-mail: jelle.maas@student.fontys.nl

Abstract

The Quantified Student (QS for short) project focuses on helping students with their development and optimizing their performance with the help of collected data. The collected data will be shown in a dashboard where the student can see it. After which, 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.

Table of Contents

Table of Figures

Revision History

Revision	Date	Author(s)	Description
0.1	April 25, 2022	J. Maas	Initial document setup
0.2	April 26, 2022	J. Maas	Improve goal and add contributions
0.3	May 9, 2022	J. Maas	Improve testing, add version control and architecture sections
0.4	May 10, 2022	J. Maas	Add conclusion

1 Introduction

Quantified Student (QS for short) Watch is a component of the Quantified Student system. The project is intended on the one hand to collect physical data from a student using smartwatches and on the other hand to provide performance insights to a student, generated from that collected data.

To achieve said goal, it is required to have proper communication with a compatible smartwatch. This communication may be used to gather, among other things, stress, and fitness data from signed up students.

The communication is setup with the help from an Android application. This application handles the communication with a compatible smartwatch, which means it can request the mentioned data from the smartwatch. The application acts as the middleman between the smartwatch and data collection endpoint.

2 Goal

The goal of this feature is to support the addition of multiple communication protocols and make it easier to implement said protocols. This solution allows the application to communicate with multiple models of smartwatches (and other peripherals?) simultaneously in a structured manner, which results in increased scalability of the application and thus increases the number of students who can use the platform.

3 Contributions

3.1 Architecture

The architecture is set up in a way that support for new peripherals can be easily implemented and overlooked. One supporting feature for this is the introduction of a protocol collection, in which each protocol has a validation function which checks if the given peripheral is compatible with it. This collection can be looped through when connecting to a peripheral.

The protocol collection is initialized at start-up of the application. Each protocol describes the compatible peripherals of said protocol, based on the manufacturer-specific data, which exists of the manufacturer identifier and the model sequence.

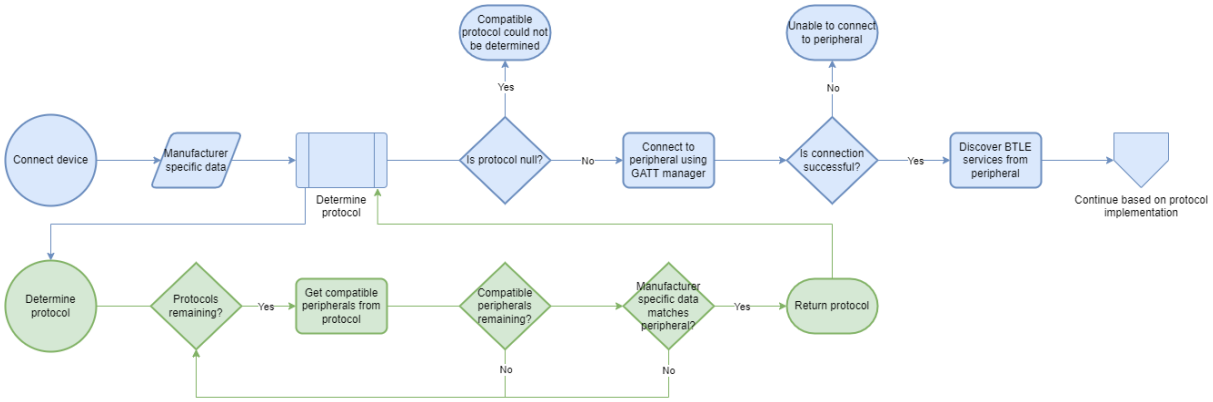

Figure 1 Flowchart protocol determination

As previously described, protocol determination starts when the user connects to a peripheral from the application. The protocol determination is done in the following three steps:

1. The first step in determining the protocol is to look up the manufacturer specific data from the peripheral and match it to a compatible protocol.
2. The second step is to determine whether the returned protocol is null or not. If so, the application couldn't find a compatible protocol for the chosen device. If not, the application will try to communicate with the peripheral using the determined protocol.
3. The third and final step is to continue the application based on the custom implementation of the chosen protocol, which can vary based on given data such as, e.g., manufacturer identifier, model sequence, etc.

3.2 Testing

Parts of the contributed source code of the multi-protocol architecture are covered by unit tests, which ensure that said code meets quality standards before it is included within the version control system. This, among others, ensures a reliable application where quality is paramount. This results in save time and money and helps to write better and more efficient code.

4 Version control

During the development of the feature, the changes were pushed to a separate branch called *"feature/multi_protocol_architecture"*. After the development process was finished, the changes were reviewed by the necessary reviewers via a pull request. When all reviewers approved the contributions, the proposed changes were merged with the main branch of the GitHub repository. The pull request shows all the comments and commits made during the development and can be viewed publicly at https://github.com/quantified-student/smartwatch-mobile-android/pull/6.

Conclusion

After the addition of the multi-protocol architecture, the mobile application is now more scalable, such that it allows easier integration with new communication protocols for Bluetooth peripherals (such as smartwatches). The next step comes when the product is out of the prototyping stage, where the addition of new peripherals will increase the number of students which can use the biometric data source within their Quantified Student dashboard. All in all, this will hopefully result in an overall increase of performance for students that use the biometric data source.