# Outpatient Monitoring Kit for Clinical Research

EngD THESIS REPORT

# Outpatient Monitoring Kit for Clinical Research

Akash Arora
October/2022
Department of Mathematics & Computer Science

EngD SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

# Outpatient Monitoring Kit for Clinical Research

Akash Arora

October 2022

Eindhoven University of Technology
Stan Ackermans Institute – Software Technology

EngD Report: 2022/071

Confidentiality Status: Public

**Partners**


Koninklijke Philips N.V.


Eindhoven University of Technology

**Steering Group**

Ir. Paul Dillen
Dr. Yanja Dajsuren, EngD
Dr. Renata Medeiros de Carvalho

**Date** October 2022

Composition of the Thesis Evaluation Committee:

Chair: Dr.ir. T.A.C. Willemse

Members: Ir. P. Dillen

Ir. B. Golsteijn, EngD

Dr. Y. Dajsuren, EngD

Dr. R.M. de Carvalho

Dr. K. Huizing

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Contact Eindhoven University of Technology
Address Department of Mathematics and Computer Science
MF 5.080A, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31 402743908

Partnership This project was supported by Eindhoven University of Technology and Koninklijke Philips N.V..

Published by Eindhoven University of Technology
Stan Ackermans Institute

EngD-report 2022/071

Preferred reference Outpatient monitoring kit for clinical research. Eindhoven University of Technology, EngD Report 2022/071, October 2022

Abstract Currently, the cumbersome and resource-intensive process of conducting clinical research is creating significant burden for hospitals. They need to allocate staff, maintain files for each subject, schedule meetings, and track feedback. Philips does not have a solution to provide their customers with an environment to facilitate clinical research or where they can evaluate clinical algorithms. The algorithms are proprietary to Philips and pose a security risk of reverse engineering even when just a software executable is revealed. This report describes a cutting-edge system developed by Philips that reshapes the clinical research domain, providing customers with an evaluation solution that is second to none. The system provides an easy way to track subjects and to collect and monitor their health data, manage wearable devices, and enable data visualization on the collected data. The system comprises wearable devices, a Linux communications hub, and a cloud-based infrastructure, which includes a visualization portal, a service that processes algorithms in the cloud, and a service that stores large quantities of metric data. We recommend testing the system in a real clinical research study and improving it accordingly where necessary.

Keywords Outpatient monitoring, data visualization, clinical research, wearable devices, analytics as a service, microservices

# Foreword

May I invite you on a virtual journey. Imagine a world in which people can keep doing what matters most to them also when they get ill and need care. Grandparents spending time with their grandchildren at home. Children playing with their friends. Parents expecting a baby. Only having to go to the hospital when there is a real need and going home again exactly when possible – not earlier, not later.

This promise is enabled by remotely monitoring patients (their heart, respiration, activity, sleep…: today's list is already endless) unobtrusively and privately by wearable devices and cloud technology and are only 'bothered' when really needed. And not only will remote monitoring yield increased quality of life for patients; it will also ease the often-stressful lives of caregivers, improve medical treatments, and reduce costs of our ever-growing healthcare demand.

Akash can be proud to have contributed significantly to this fulfilling vision during his 10 months EngD assignment at Philips Research.

Before remote monitoring products can be used by patients, they first need to be developed and thoroughly tested. For this, a remote monitoring kit is under development at Philips, allowing to remotely collect data from patients in clinical studies. Scientists can use the data to develop screening, diagnosis, or treatment applications. And other companies can evaluate Philips technology for their own products. As part of the 'Patience' subsidy project, Akash has greatly contributed to the development of the user interface of the remote monitoring kit, making it easy to use and tailored to its job, with a great and beautiful result.

In the development team, Akash was a kind of 'quiet force', soft spoken and often working silently. It sometimes made me ask if everything was fine, but with that simple question, Akash would produce solid, well-structured, and convincing work products: figures, documents, and software, radiating 'don't you worry, it is all handled'. All in a dependable manner, both technically and timewise, and meanwhile eager to learn and grow. I am confident he will gradually step up more into the spotlight.

**Eindhoven University of Technology**

We are grateful for all of Akash's dedication, persistence, and drive for quality, which eventually will make a difference to people's lives. I wish Akash all the best for his further career and do hope that our paths will cross again. Shukria!

Paul Dillen
October 2022

# Preface

This report summarizes the project carried out by Akash Arora, as the graduation project of Engineering Doctorate (EngD) program in Software Technology at Technische Universiteit Eindhoven (TU/e). The project duration was ten months and was conducted at Philips, Eindhoven. The project was supervised by Paul Dillen, at Philips Research, Yanja Dajsuren and Renata Medeiros de Carvalho at TU/e.

The project is aimed to develop a better solution for clinical research in conjunction to catering the solution to Philips' technology evaluation and licensing use case. This was achieved by developing the Studykit system. The system caters to clinical study workflow and thus reduces the workload on the hospitals, while providing the customers of Philips with a technology evaluation environment.

This report summarizes the clinical workflow, Philips' business case, requirements, architecture, design, implementation, verification, and validation of the system. Chapter 1 and 2 lays down the context, domain, and problem analysis. Chapter 3 to 7 cover the requirements, architecture, design, implementation and verification and validation of the system. Chapter 8 covers the project management aspect of this project.

Akash Arora
October 2022

# Acknowledgements

Prima facie, I would like to thank all my supervisors, mentors, and the people I collaborated with during the term of this project.

Firstly, I want to express my sincere gratitude to Paul Dillen, from Philips, for giving me this opportunity to be a part of such an immense project. You have guided me every step of the way, neglected my mistakes and always kept me positive. I really think I couldn't have gotten a better company supervisor than you.

Secondly, I want to thank my TU/e supervisors Yanja Dajsuren and Renata Medeiros de Carvalho, for their insightful comments and questions. As busy as a bee you were, you still made time for me and gave the support and guidance that was needed for this project.

I would also like to thank Martijn van Welie, Reinder Haakma, Dave Boshoven, Carlijn Vernooij, Bram Hoendervangers, Gertjan Maas, from Philips, whom I worked alongside during the term of this project.

I am also grateful to the committee members, Bart Golsteijn, Kees Huizing, and Tim Willemse for participating in my thesis defense, taking the time to read through every detail and evaluating me on my work.

October 2022

# Executive Summary

Philips is a global leader in personal health, connected care, and diagnosis and treatment. Philips' healthcare devices work with other software and hardware products to create a complete set of products. Many hospitals worldwide use Philips products to provide the best healthcare for patients. A lot of companies apply Philips' technology in their products as well.

In the absence of a smart monitoring kit, which would benefit the patients as well as the hospitals, Philips is set on a mission to develop an outpatient monitoring kit for clinical research. Currently, some problems of the current way of working for clinical research are:

- It is error prone
- It is dependent on patient feedback
- It is difficult to maintain
- It is time consuming for the hospital staff

The project goal is to develop the Studykit that enables healthcare to track patients, regardless of whether they are bed-ridden or ambulatory, i.e., outside the hospital, living day-to-day life. The Studykit system includes multiple types of wearable devices, a Linux communications hub, and a cloud-based app. The wearable devices will be able to monitor patients with the help of the Linux machine. The cloud-based app will enable healthcare professionals to track patients.

This report presents the process and approaches taken to handle the issues of clinical research, such as data collection and feedback management, with the help of Studykit system. The Studykit system automates the process of collecting feedback and maintaining the data, eliminating errors, and reducing valuable time spent while collecting and maintaining the feedback. We recommend testing the system in a real clinical research study and improving it accordingly where necessary.

# Table of Contents

# List of Figures

# List of Tables

# 1.Introduction

The importance of remote monitoring of patients has increased significantly in the past two years because of the COVID-19 pandemic. Patients were quarantined at home, but there was no way of detecting if their health was improving or deteriorating. With the advent of remote monitoring, patients can be monitored from the comfort and safety of their own home, allowing them to take more control of their health and wellbeing, while healthcare providers can track and intervene whenever necessary. Remote monitoring has many advantages over in-person monitoring, including the ability to track multiple patients simultaneously and make precise decisions over a patient's health.

Philips is developing a patient-centric holistic suite of products where the data is collected irrespective of the device or system and stored in a unified central repository. Furthermore, any algorithm or system can use the data from this central repository to enhance the assistance towards the patient. This patient-centric, holistic suite of products will help to improve the patient experience by allowing for a more seamless flow of information between devices and systems.

Currently, an outpatient monitoring kit is already under development at Philips. The kit comprises a wearable device, a communications hub, and cloud-based software. As a first step, this kit is being developed for clinical research to gauge patient responses and monitor outcomes. This iterative approach serves as a proof-of-concept and lays the groundwork for future enhancements that will make the kit more user-friendly and efficient.

This report consists of ten chapters. The next chapter discusses the domain and problem analysis. Chapter 4 and 5 discuss the architecture and design of the system based on the requirements gathered in Chapter 3. Implementation details are discussed in Chapter 6, followed by verification and validation of the system in Chapter 7. Chapter 8 discusses the project management of this project over the ten-month period. Chapter 9 discusses the results of this project followed by recommendations for future work. Finally, Chapter 10 provides a project's reflection from the author's perspective.

# 2.Domain and Problem Analysis

To understand the problem, it is essential to evaluate the clinical workflow and Philips' business case for licensing technology. Section 2.1 describes clinical research and the workflow that is carried out between Philips and a hospital that conducts clinical research. Section 2.2 describes Philip's technology licensing workflow that enables them to work with different companies. Subsequently, problem definition, goal and scope are discussed between Section 2.3, 2.4 and 2.5 respectively.

## 2.1 Clinical research

Clinical research is a branch of healthcare science that determines the safety and effectiveness of medications, devices, diagnostic products, and treatment regimens intended for human use [1]. Clinical research is usually carried out by a hospital or a collaboration between hospitals or in collaboration with a clinical research institute. Clinical research is of two kinds, observational and interventional (clinical trials).

In an observational study, a group of people are observed when they are either ambulatory or bed-ridden in a hospital. The aim of an observational study is to observe patients and track health outcomes over time rather than test potential treatments. The treatment towards the group of people does not change based on the observation.

Interventional studies or clinical trials, on the other hand, aim to find or validate a new potential drug, medical device, or procedure. An interventional study is usually carried out in different phases of varying group sizes.

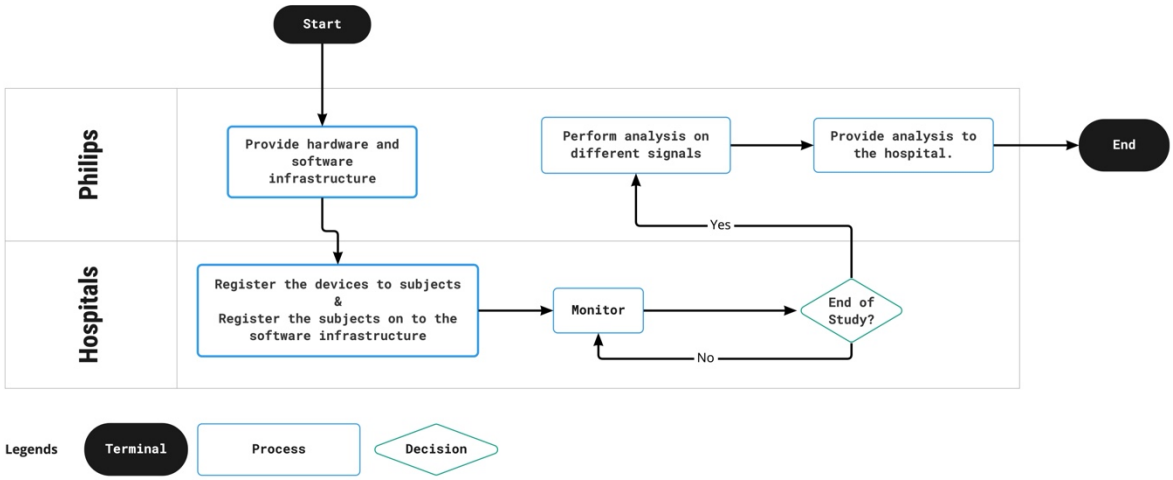



*Figure 1 - Clinical workflow relationship between Philips and a hospital (observation study)*

From Figure 1, we can understand the relationship between Philips and a hospital conducting clinical research (observational). Philips provides the software and hardware infrastructure necessary to conduct observational research. The software infrastructure includes a portal where users are monitored. The hardware infrastructure includes wearable devices such as Philips Data Logger (PDL), Philips Health Band (PHB), and Philips Healthdot.

Due to privacy and regulatory issues, careful management of personal information is required. Careful management usually entails providing access to personal information by certain staff members of Philips and only when necessary. Hence, hospitals use a code for each subject. Since the algorithms require certain attributes (such as height, weight, age, and the wearing position of the device) of a particular subject, written consent by the subject is required beforehand. The consent allows access to the subject's characteristics.

After the hospital has registered devices to subjects and associated the subjects with the relevant study, progress can be monitored. At the end of a study, Philips performs an analysis on the collected data. Subsequently, the analysis as well as the collected data is handed over to the hospitals. In certain cases, dependent upon the agreement and consent of the participants of the study, Philips can also retain the data for future use.

## 2.2 Philips' business case – Technology Licensing

Over the years, Philips has constantly upgraded their Biosensing technology application, comprising of algorithms to deliver the most accurate output/derived metrics. These algorithms are either incorporated into wearable devices and process the measurements directly or they are handled by an external system.

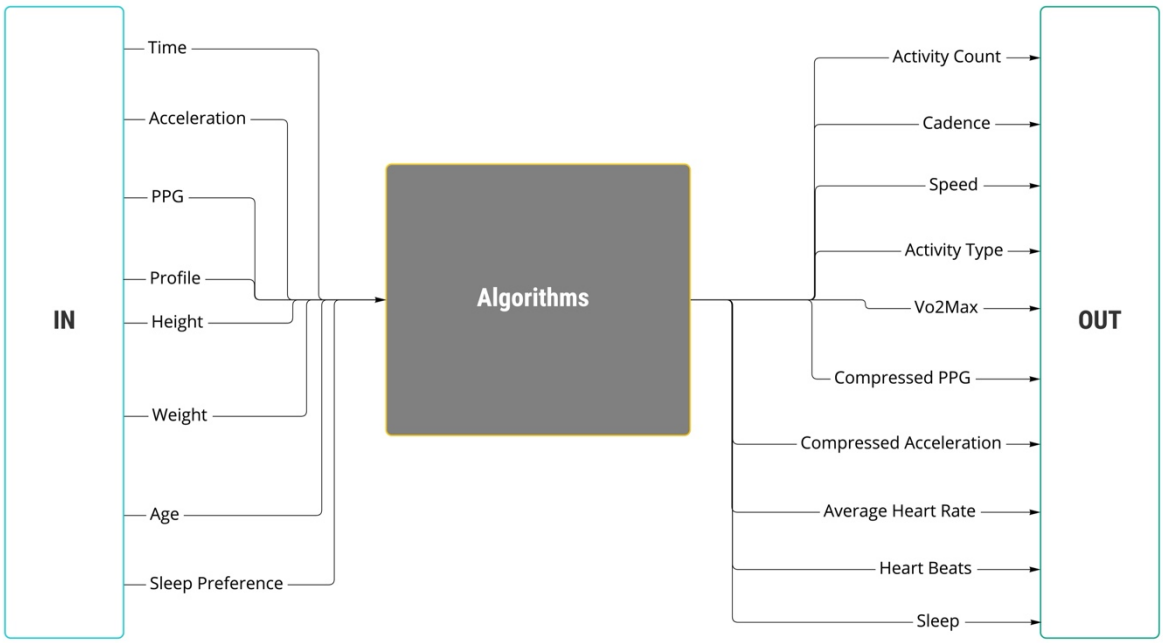



*Figure 2 – Algorithms under the Biosensing technology*

As shown in Figure 2, algorithms are responsible for generating the output/derived metrics from the input/raw metrics. We provide an example to visualize how raw metrics from a wearable device such as PDL, Healthdot, or PHB, are used to generate over ten different derived metrics. Devices such as PHB and Philips Healthdot run on-board algorithms that yield derived metrics and discards the raw metrics. The PDL is currently the only device that does not aggregate raw metrics over time. The algorithms are designed to deliver the same result for the given input even when the data is aggregated.

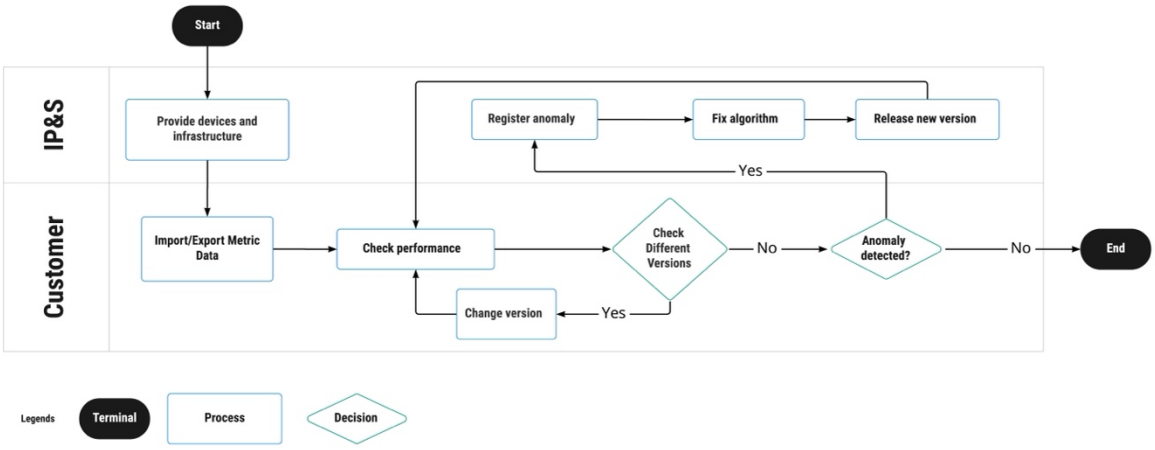



*Figure 3 - Workflow between Philips and third-party companies*

Philips licenses these wearable devices as well as the algorithms to other companies (customers) who want to use this technology. As shown in Figure 3, we visualize the interaction between the Intellectual Property & Standards (IP&S) department of Philips and their customers. The IP&S department is responsible for licensing the products at Philips.

IP&S provides the hardware and software infrastructure to their customers. Customers can now use their existing dataset and import the data into the system or test the wearable device and extract raw metrics from the device and upload them. Depending on the device, typically a Universal Serial Bus (USB) interface is required for extracting the raw metrics from the device. Now, the customers can evaluate the performance of the algorithms. If they see any anomaly with the metrics, they report it back to IP&S. IP&S investigates the problem, fixes it, and delivers a new version of the algorithm.

## 2.3 Problem Definition

Based on the domain analysis, the following challenges are identified:

- The current technology at Philips is not equipped to handle an influx of devices and is not able to manage thousands of devices effectively.
- The technology currently used at Philips is not extendable and is not designed to be compatible with different devices. Systems such as the Philips Actigraphy Server System (PASS), are created for a singular purpose and singular device, such as the PHB. There is no one-size-fits-all solution that can be used for different devices.

- The process of either embedding algorithms inside the wearable devices or providing a software application to process these algorithms possesses a security threat to the Intellectual Property of Philips.
- As algorithms are updated and changed over time, it becomes increasingly difficult to manage versions among different clients.
- Systems, such as PASS, do not allow users to visualize data.
- Systems such as Healthdot system and PASS do not let users export data from the User Interface.
- Devices do not record raw metrics. Artificial Intelligence (AI) is the future and having the raw metrics as the input will be more beneficial for both the patients as well as Philips.
- Currently, there is no effective way of storing time-series based high-volume data at Philips.
- Currently, there is no safe way for IP&S to provide an evaluation environment to their customers without revealing licensed software.

## 2.4 Goal

The primary goal of the project is to develop a system that reshapes clinical workflow as well as providing the technology licensing customers with an evaluation environment. The goal of the project is to develop a system that enables hospitals to track patients and their data, manage wearable devices, and enable data visualization on the collected data. The system should also be able to provide an environment for customers of Philips to evaluate the Philips-licensed algorithms in a controlled way.

To achieve this goal, an outpatient monitoring kit is already under development at Philips. The architecture of the proposed solution and how it integrates with this kit is detailed under Chapter 4. This kit comprises a wearable device such as a PDL, a communications hub that is based on Linux called MSX (internal name), and cloud-based software. Cloud-based software includes a Studykit Store (based on a time-series database), an analytics engine (based on Apache Airflow), a Clinical Data Repository (CDR) framework, and Studykit Portal. The CDR will interact with the Fast Healthcare Interoperability Resources (FHIR) store, which stores information related to a study, subjects, and devices. A backend Application Programming

Interface (API) microservice manages the interaction between CDR, the analytics engine, and Studykit Store.

The design of Studykit Portal allows for ease of use, management of studies, subjects, and devices with visual data representations for each subject. The portal also has an interface for importing and exporting data and managing algorithm versions for each study. The backend microservice API forms an abstraction layer, hiding all other systems connected to it. The Studykit Store is based on a time-series database, which can effectively store a high volume of data. The analytics engine uses Apache airflow to run algorithms in the cloud. The combination of all these systems along with the MSX hub deals with scalability and extendibility concerns of the existing technology.

## 2.5 Scope

Within the scope of this project, we aim to develop the Studykit infrastructure with the following objectives:

- Gather requirements for the system
- Develop domain models based on requirements
- Develop a Portal with the following features:
    - ability to visualize data
    - ability to manage studies and subjects
    - ability to manage algorithms and different versions
    - ability to Import and Export metrics
- Develop a Backend API with the following features:
    - develop APIs which can communicate with different services.
    - validate the security tokens with each call and prevent un-authorized access.
- Develop Studykit Store with the following features:
    - handle different metric formats
    - allow for traceability for origin of the metric
    - handle high volume data
- Verify and validate the solution

# 3.Requirements Elicitation

## 3.1 Introduction

This chapter discusses the requirements of the Studykit system, based on the stakeholder analysis and scope of the project. The requirements are prioritized using MoSCoW prioritization method. The acronym represents four categories as shown in Figure 4.

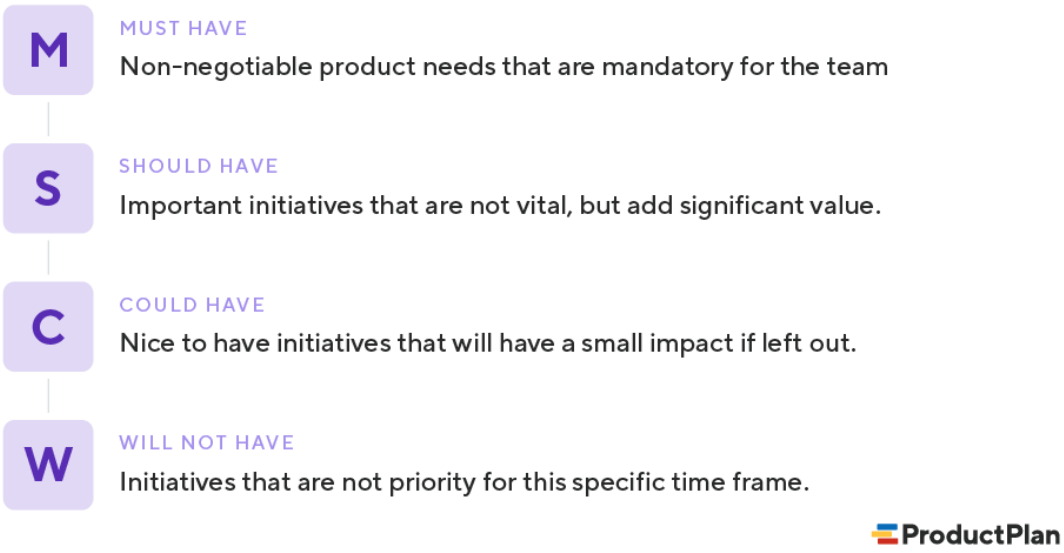



*Figure 4 - MoSCoW acronym [2]*

## 3.2 General requirements

The generic characteristics and actions of the system are presented in Table 1 and are referred to as the General System Requirements.

*Table 1 - General requirements*

| Requirement Id | Priority | Requirement |
|---|---|---|
| **GR01** | Must | The portal should be accessible with any modern browser (i.e., support for ECMAScript v6). |
| **GR02** | Must | The portal should be able to visualize metric data. |
| **GR03** | Must | The portal shall be deployable on Philips' network. |

| | | |
|---|---|---|
| **GR04** | Must | The system shall use Philips' IT-approved tools and technologies. |
| **GR05** | Must | The database should normalize the data before storing. |
| **GR06** | Must | The algorithms should run independently for each subject of a study. |
| **GR07** | Must | The system should store patient/subject information using FHIR. |
| **GR07** | Must | The system should provide a logging and monitoring mechanism. |
| **GR08** | Won't have | The system should be able to live stream data and show visualization. |

## 3.3 Functional requirements

The system's behavior is defined and described by the functional requirements. Functional requirement for each microservice is categorized as following:

*Table 2 - Studykit Portal - Functional requirements*

| Requirement Id | Priority | Requirement |
|---|---|---|
| **FR01** | Must | The portal shall allow the user to import Wearable Sensing Technologies (WeST) data metric files. |
| **FR02** | Must | The portal shall allow the user to export WeST data metric files. |
| **FR03** | Must | The portal shall allow the user to select a particular date for data visualization. |
| **FR04** | Must | The portal shall allow the user to have a high-level view if the data is present for a particular subject or not. |
| **FR05** | Must | The portal shall allow the user to select multiple metrics for visualization. |
| **FR06** | Must | The portal shall inform the user if the visualization data is aggregated |

| | | |
|---|---|---|
| **FR07** | Should | The portal shall inform the user regarding the state of algorithm processing after successful import. |

*Table 3 - Studykit Store - Functional requirements*

| Requirement Id | Priority | Requirement |
|---|---|---|
| **FR08** | Must | The store shall parse different metric data files based on their specification when importing. |
| **FR09** | Must | The store shall construct different metric data files based on their specification when exporting. |
| **FR10** | Must | The store shall selectively map out of range data into their valid range. |
| **FR11** | Must | The store shall track the source information of each metric. |
| **FR12** | Must | The store shall be able to parse metric files irrespective of the white space character (i.e., tab or spaces) |

*Table 4 - Analytics Engine - Functional requirements*

| Requirement Id | Priority | Requirement |
|---|---|---|
| **FR13** | Must | The engine shall provide an API interface for Apache airflow |
| **FR14** | Must | The engine shall store configurations for each subject and study |
| **FR15** | Won't have | The engine shall support live streaming and processing of metric data. |
| **FR16** | Should | The engine API shall provide progress update on the algorithm state post data manual import. |

## 3.4 Non-functional Requirements

A non-functional requirement (NFR) is a requirement that defines criteria rather than specific behaviors that may be used to assess how well a system performs. Non-functional requirements define how a system is supposed to be, in contrast to functional requirements, which specify what a system is supposed to do. For example, an NFR might state that the system must be able

to handle 100 calls per hour, while a functional requirement might state that the system must be able to process 10 calls per minute. Table 5 shows the non-functional requirements of the system.

*Table 5 - Non-Functional requirement*

| Requirement Id | Priority | Aspect | Description |
|---|---|---|---|
| **NFR01** | Must | Security | The system shall authenticate and authorize user and each request using Philips' Identity Access Management (IAM) Client. |
| **NFR02** | Must | Privacy | The system shall be deployed as per regional privacy laws. |
| **NFR03** | Should | Accessibility and Usability | The system shall adhere to Philips' digital Design Language System (dDLS) specification for the user interface. |
| **NFR04** | Should | Compliance | The system shall adhere to all compliances of the respective country, where it is deployed. |
| **NFR05** | Should | Maintainability | Each microservice of the system should be developed with the best practices so, it can be maintained easily over the lifetime of the system. |
| **NFR06** | Should | Extensibility | Each microservice of the system should be modular such that it can be extended in the future without any architectural changes. |

# 4.System Architecture

System architecture is the structural design of the entire system. It is the foundation of the entire system. An architecture of the Studykit system is modeled based on the functional and non-functional requirements mentioned in Chapter 4. This chapter discusses system architecture using the 4 + 1 architecture view model by Philip Kruchten [3]. The architecture view model uses the logical view, process view, deployment view, and physical view with the context of use case scenarios to describe the rationale for each decision.

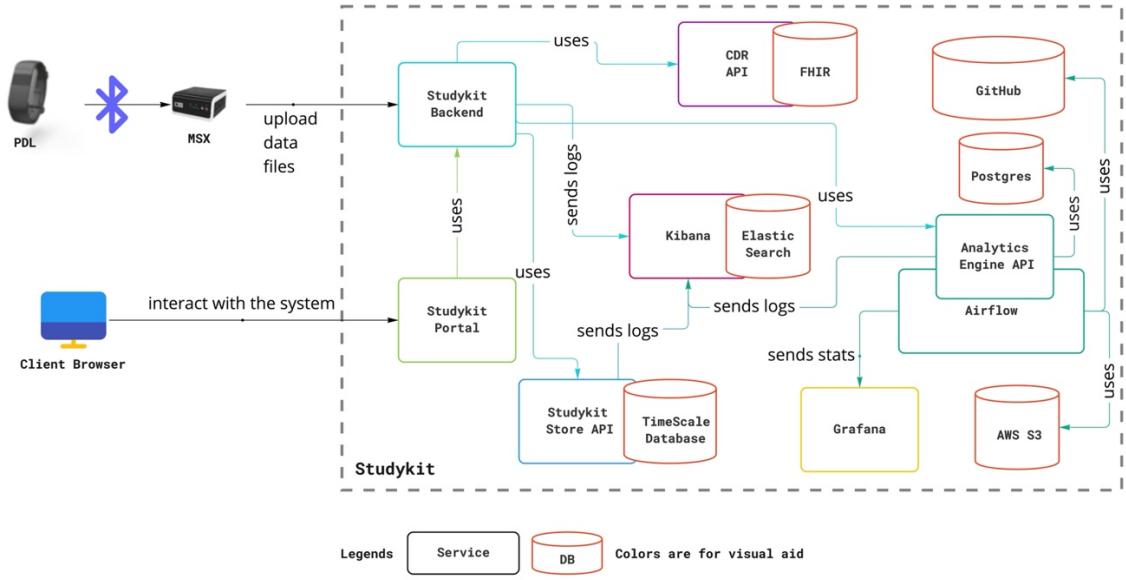



*Figure 5 – High-level architecture*

The Figure 5 illustrates the different components of the system and how they interact with each other. Different components of the system are:

- **PDL:** For the Minimum Viable Product (MVP) the system primarily uses the PDL as the wearable technology to test the automatic data transfer in the system. The system is still extendable to other wearable devices as well, Philips' wearable devices as well as selected third-party devices.
- **MSX:** The MSX is a communications hub based on Linux. It is responsible for collecting data from wearable devices and converting it to WeST specific data files. It uses Bluetooth technology to communicate with the wearable devices.

- **Studykit Portal:** The portal is based on React JavaScript (JS) and Typescript and allows the users of the system to interact with the system. The portal interacts with the Studykit backend API to perform a lot of different tasks, such as retrieving data and storing it, processing information, and interacting with specific microservices.
- **Studykit Backend:** The backend is the center core of the entire system. The Backend handles all the API requests from different microservices. It is also responsible for proxying the data files uploaded by the MSX to the Studykit Store.
- **CDR-API and FHIR:** CDR is Philips' internally developed technology that is designed to efficiently manage and track patient data. Using FHIR, CDR stores data in a standardized format that can be easily accessed and utilized by authorized personnel.
- **Kibana:** Kibana is an open-source technology that is used to maintain logs from different microservices.
- **Grafana:** Grafana is an open-source technology that allows users to create interactive dashboards by connecting to different data sources. It is used for monitoring the work orchestration carried out by Airflow and provides users with valuable insights into their data.
- **Analytics Engine API and Apache Airflow:** Apache Airflow is a work orchestration application that allows users to queue up different tasks and, based on resource availability, execute them. Analytics Engine API is a wrapper around Airflow which allows other microservices to interact with it.
- **Studykit Store API and TimeScale Database:** TimeScale Database is a time-series based database that can handle large volumes of data. The Studykit Store API is designed to assist the TimeScale database in Create, Read, Update, and Delete (CRUD) operations.
- **GitHub for Algorithms:** Although GitHub is a tool that enables version control of codebases, it also can act as a store. Every new release of the algorithm can be tagged and used directly from GitHub.
- **Amazon Simple Storage Service (AWS S3):** AWS S3 is used as a Binary Large Object (BLOB) storage for the application that stores the various algorithm states. With each run, the algorithm recalls the previous state before running and hence, it becomes more accurate over time.

## 4.1 4 + 1 Architecture Model

### 4.1.1. Logical View

The logical view primarily describes the functional requirements of the system, i.e., what the system should provide to its users. The system caters to two different use cases as discussed in Chapter 2, i.e., Clinical Research, and Philips' business case for technology licensing and evaluation. Figure 6 shows how their relationships are mapped based on the requirements. Naturally, the system had to be generic in approach and the weaker relationship of the IP&S use case was implemented.

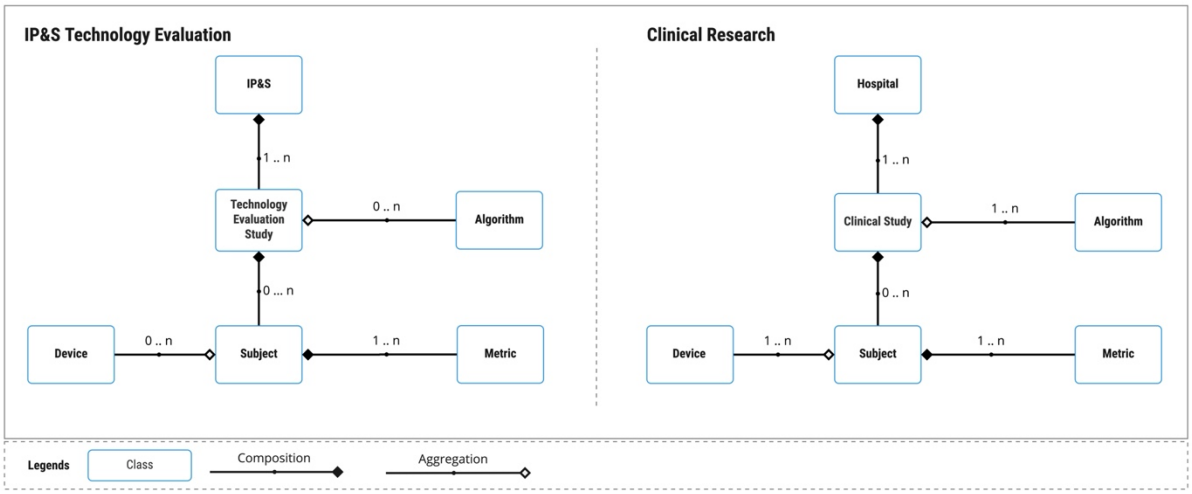



*Figure 6 - Class Diagram describing the different use cases of the Studykit system*

Figure 6 consists of the following classes:

- **User:** This varies based on the use case, can either be IP&S (and their customer) or Hospital. The user can be assigned to multiple studies.
- **Study:** Every user is assigned to a study which consists of different subjects.
- **Algorithm**: The algorithms are assigned to each study and can vary based on the use case. Sometimes, a customer may just want to verify the raw metrics and not utilize any of the algorithms. In this instance, the customer would forego any of the benefits that the algorithms provide. In the case of clinical research, at least one algorithm is utilized.
- **Subject:** Subject is a generic term used in this system. Not all participants in clinical study are ill and might be participating for other reasons. (e.g., clinical research being

conducted on athletes). Hence, the term patients cannot be used. Also, IP&S customers might be wearing the device themselves for testing purposes.

- **Device:** One or more devices can be assigned to a particular subject for clinical research. In the case of Philip's evaluation case, a customer might not assign a device to a particular subject. Just importing and exporting data rather than using a device can achieve their use case. Hence, the metrics are tied to a subject and not to a device, but a reference is maintained.
- **Metrics**: These are the raw and derived metrics of a particular subject during the study.

The Studykit Store design must cater for the centricity of the subjects. Figure 7 describes the relationship between different entities. The Metadata entity is the single-entry point as it has the primary key. All other entities except the DataSource entity depend on Metadata to provide the correct relationship for association. The Metadata entity also has an external source foreign key for "subjectId" from the FHIR database, but it is not represented as a foreign key in the diagram since the relationship cannot be demonstrated. The DataSource entity is separated from the metric relation because it is responsible for traceability. Each metric data has an origin, which can be the device or the algorithm, and the DataSource tracks the origin with each entry.

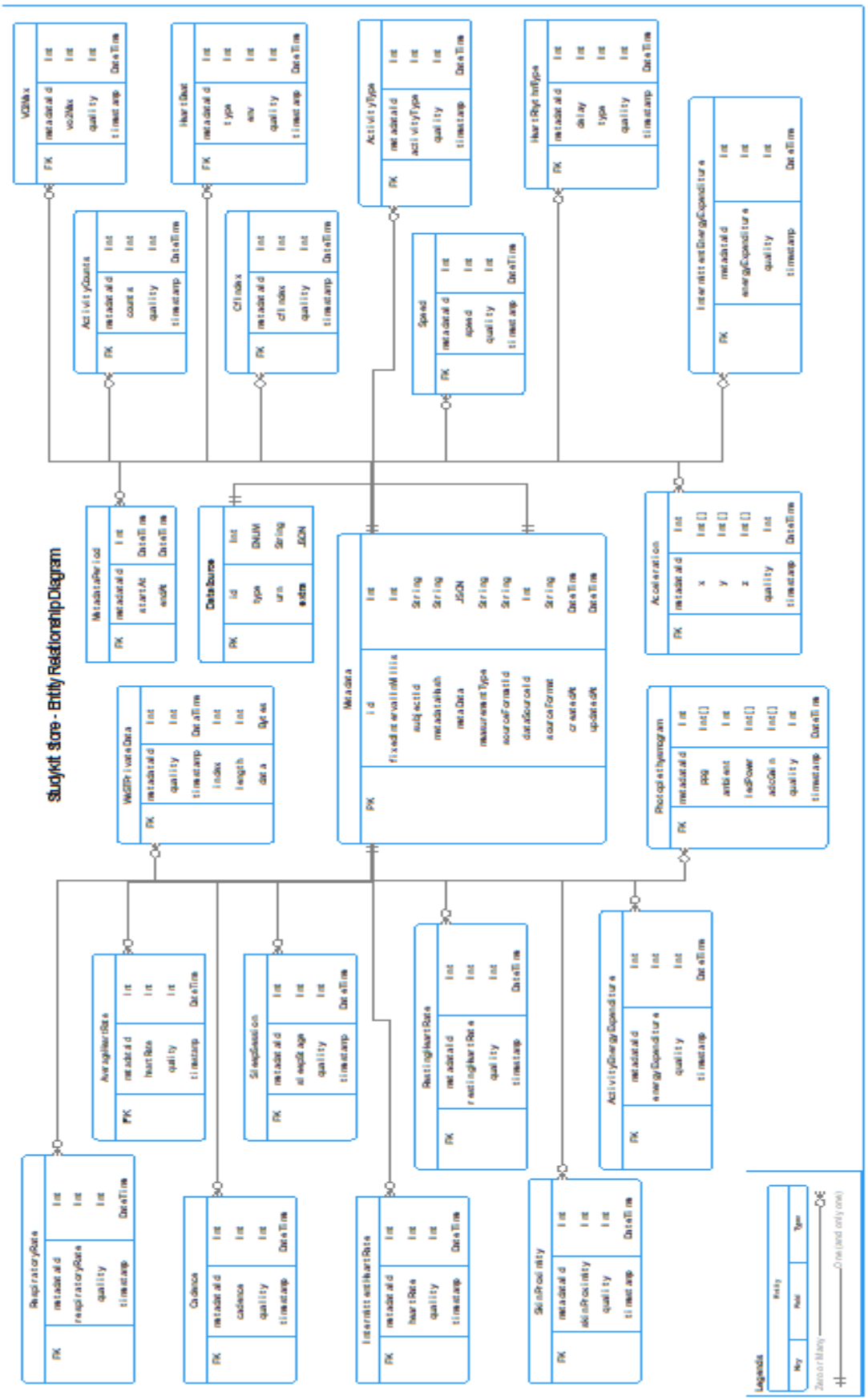



*Figure 7 - Entity Relationship diagram of Studykit Store*

### 4.1.2. Process View

The process view is associated with the system's dynamic components and focuses on the system's run-time behavior. It shows fundamental processes and how they interact.

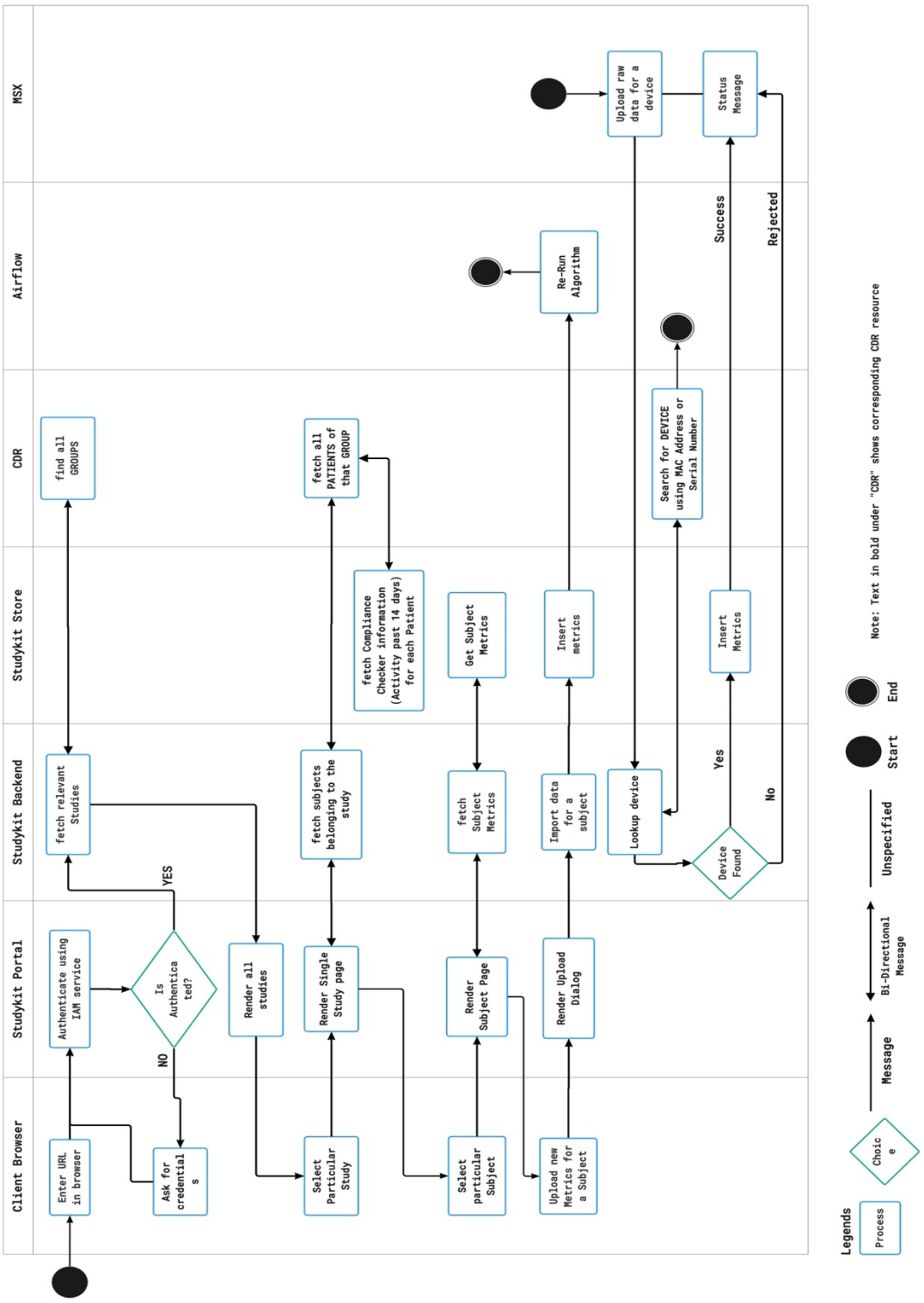



*Figure 8 - System Activity Diagram*

As shown in Figure 8, the system activity diagram covers two important features: the MSX upload process and user interaction with the system. The figure illustrates the interaction between different microservices to perform different tasks (requested by the client browser or when MSX uploads data). All requests go through the Studykit backend because it is the only open facade and thus acts as a proxy service for the entire system. The system is designed using the API Gateway Design Pattern of Microservices architecture. The benefit of having this design pattern is to enable scalability of the system.

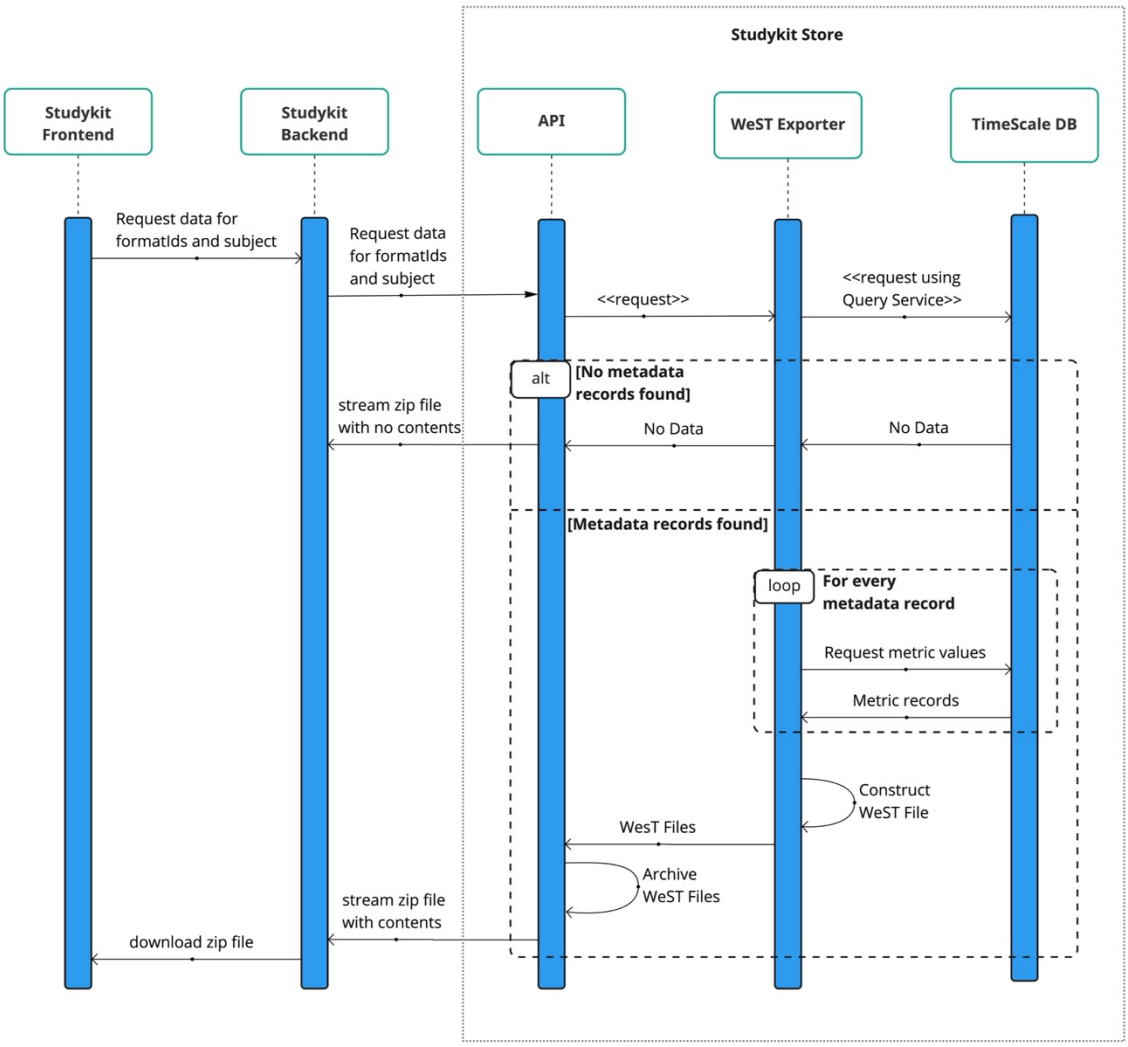



*Figure 9 - Sequence diagram - Metric export*

To visualize the use case of exporting metrics, we can refer to Figure 9. It shows how the request originating from Studykit frontend to export metrics for certain format ids and a particular subject is proxied from the Studykit backend to the Studykit Store and its various components.

The frontend handles user requests for different metrics that need to be exported. It translates the name of the metric into a WeST-specified format ID and then requests the information from the Studykit backend, which proxies the call to the Studykit Store. The Studykit backend only verifies the security token and not the request. The Studykit Store API verifies the request and assigns the appropriate service to handle the request (i.e., the WeST Exporter). The WeST Exporter service of the Studykit Store subsequently requests data from the TimeScale DB. If the database returns metadata records, the service then requests data for each metadata record. Finally, it constructs a WeST file from the metric records and creates a zip file. If no metadata is returned, the service still creates a zip file, but without any content. This is because the source client is expecting a BLOB as a response.

### 4.1.3. Development View

The development view, also known as the implementation view, is concerned with software management and portrays a system from the standpoint of a programmer. It uses the Unified Modelling Language (UML) Component diagram to describe system components. Figure 10 shows different components of the entire system.

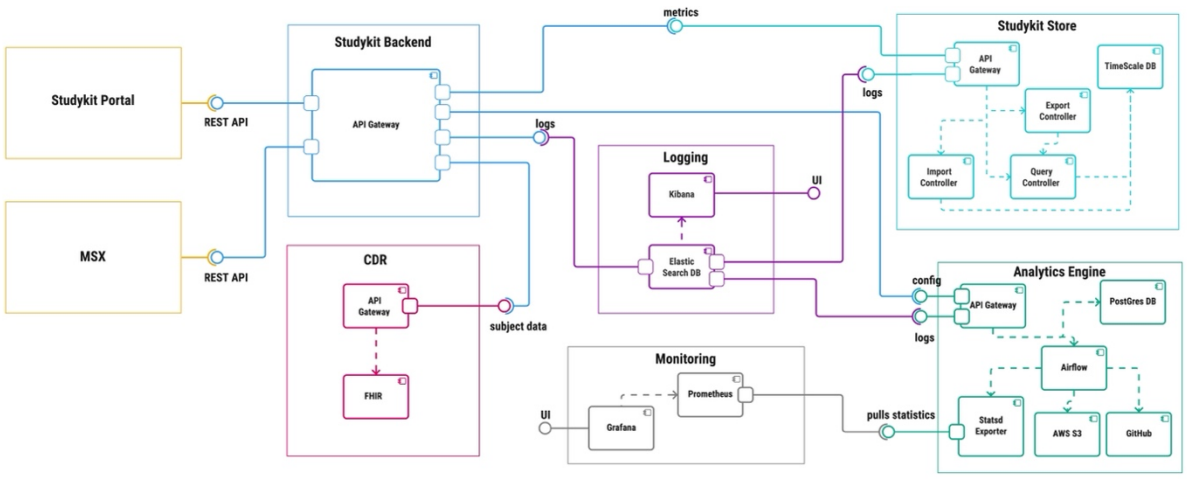



*Figure 10 - Component Diagram of the system*

### 4.1.4. Physical View

The physical view, also known as the deployment view, depicts the system from the perspective of a system engineer. The physical layer is concerned with the structure of software components as well as their physical connections.

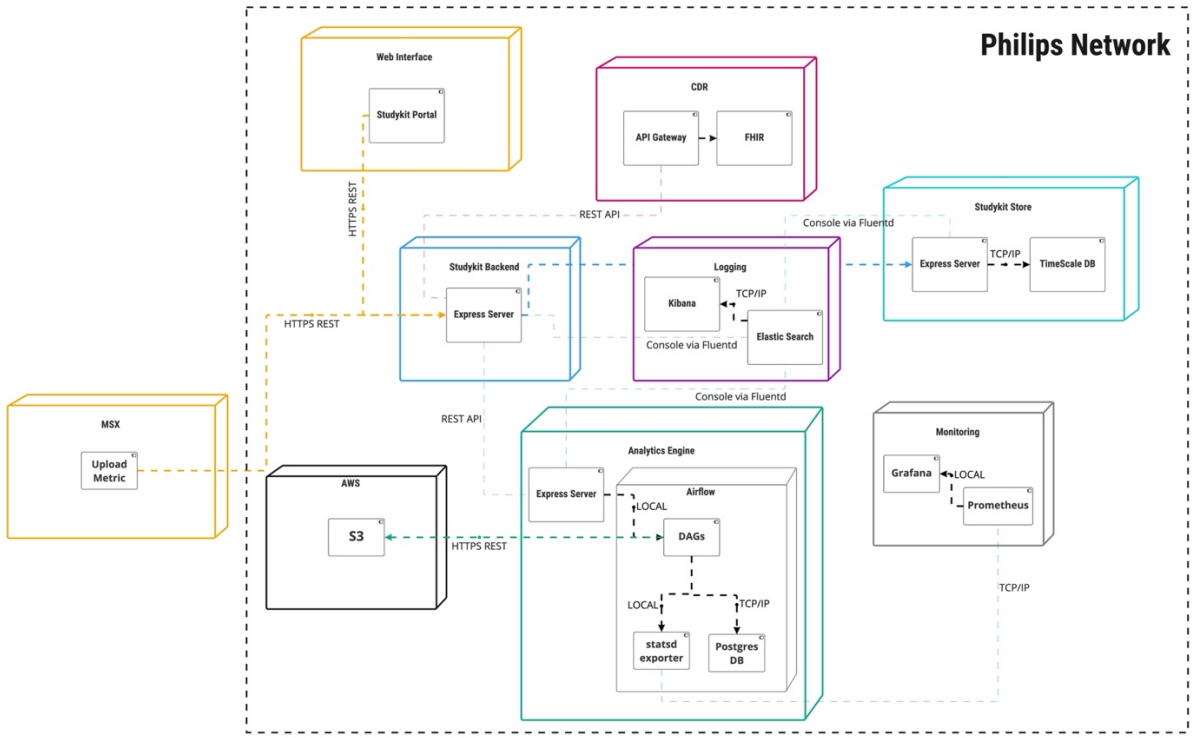



*Figure 11 - Deployment Diagram of the system*

As shown in Figure 11, the entire system is deployed inside the Philips Network except for MSX, which is at the subject's end. The Philips Network is essentially Philips' Health Suite Digital Platform (HSDP). HSDP is Philips' solution designed for cloud infrastructure with appropriate licenses and regulatory compliances for multiple regions. HSDP is a cutting-edge deployment tool that's been deployed on AWS and supports most of the services offered by AWS. HSDP is essentially a wrapper that lets users securely deploy services in the cloud, with the added benefit of being able to do so in a repeatable, automated way.

### 4.1.5. Use Case Scenarios

The different views illustrate the system well, but they do not clarify how users will use it. Use case diagrams add value to the different views by clarifying how users use the system and how the system should behave.

**Use case: CRUD study and subject**

Figure 12 illustrates the use case for creating, updating, reading, and deleting a study and subject.

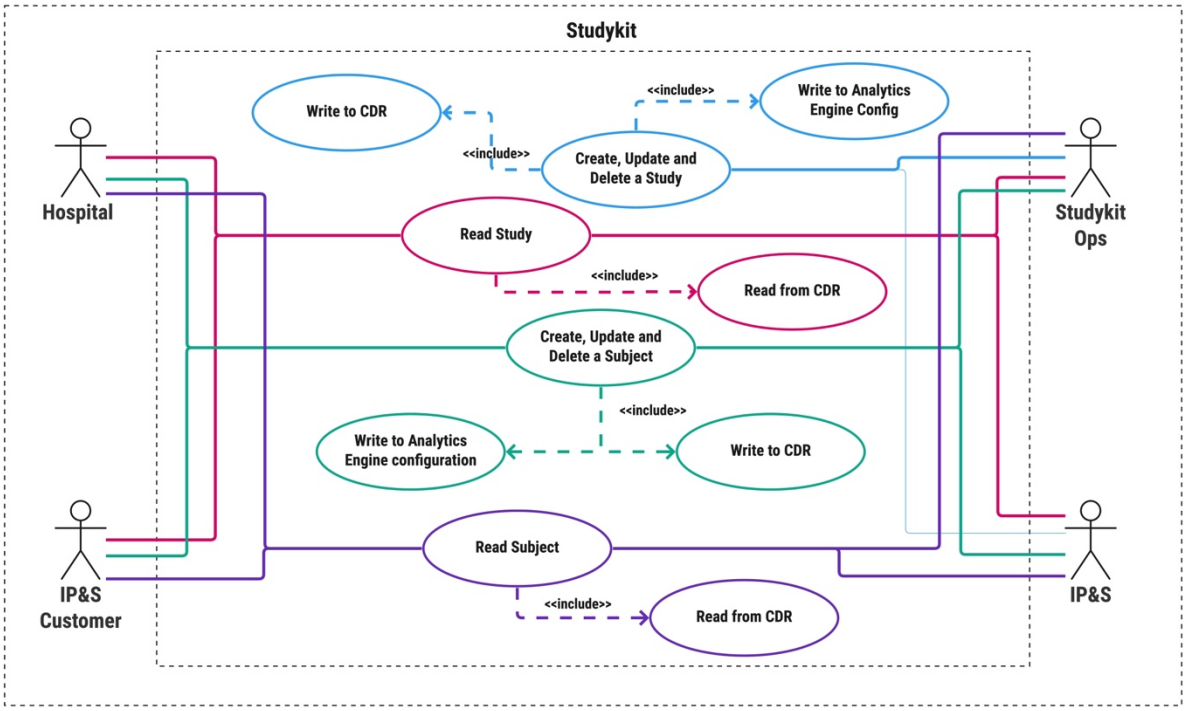



*Figure 12 - Use Case: CRUD study and subject*

**Create, update, and delete study**

1. When the user wishes to create, update, or delete a study, the request is sent to the backend with appropriate headers and data.
2. The request is forwarded to the CDR store, where the Groups resource of FHIR makes the appropriate changes.
3. The request is forwarded to Analytics Engine as well, where the changes to configurations are reflected. Airflow works in the context of the study configuration.

**Read study**

1. When the user wishes to view the information of the study, the request is sent to the backend with appropriate headers and data.
2. The request is forwarded to the CDR store, where it performs a lookup on the requested group resource.

**Create, update, and delete a subject of a study**

1. When the user wishes to create, update, or delete a subject of a study, the request is sent to the backend with appropriate headers and data.
2. The request is forwarded to the CDR store, where the Patient resource of FHIR makes the appropriate changes. In the case of creating, a link is created with the Group resource.
3. The request is forwarded to Analytics Engine as well, where the changes to the configuration of that study are reflected.

**Read subject**

1. When the user wishes to view the information of a subject of a study, the request is sent to the backend with appropriate headers and data.
2. The request is forwarded to the CDR store, where it performs a lookup on the requested patient resource.

### Use case: Import and Export metric data

Figure 13 illustrates the use case for importing and exporting metric data files.

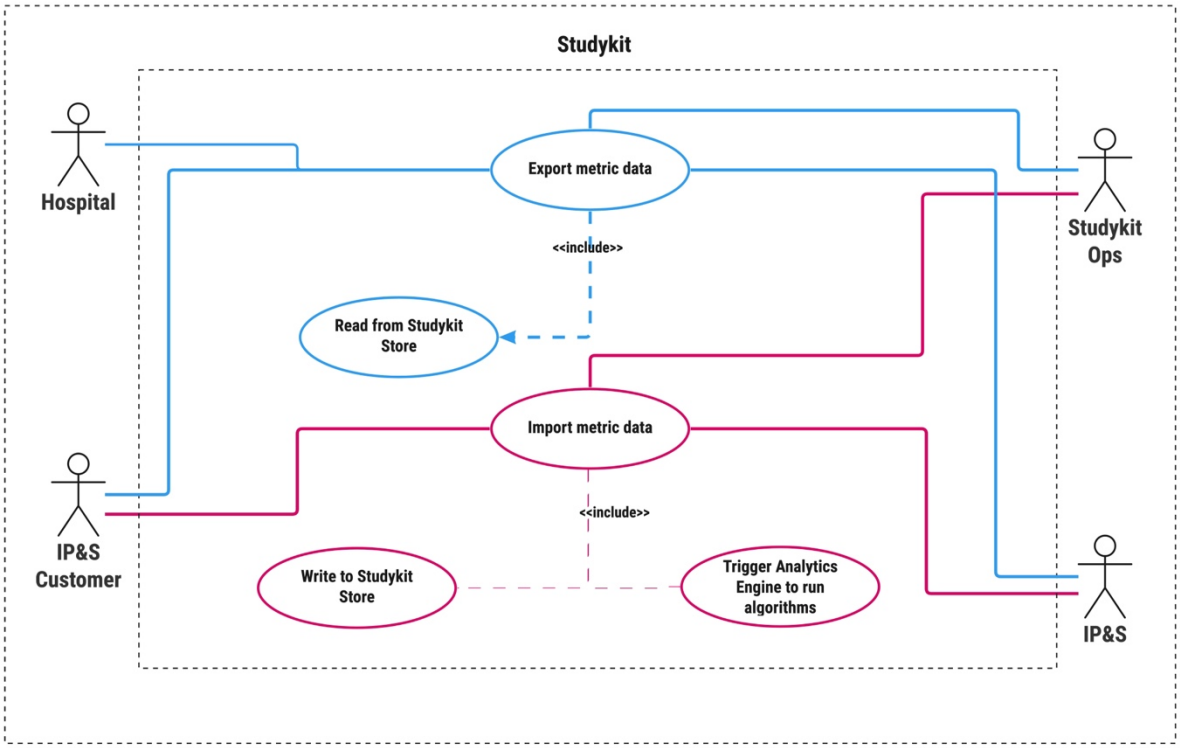



*Figure 13 - Use case: Import and Export metric data*

### Import metric data

1. When the user wishes to import metric data into the system, the user should select the appropriate subject of the study.
2. The system accepts WeST file format for importing the data.
3. The files are proxied from the backend to the Studykit Store importer service.

### Export metric data

1. When the user wishes to export metric data from the system, the user should select the date of the data and the metrics.
2. The request is proxied from the backend to the Studykit Store exporter service.
3. If data exists for the selected date and metrics, a zip file is returned with all metric files.
4. If no data exists for the selected date and metrics, an empty zip file is returned.

### Use case: Visualize subject metric data

Figure 14 illustrates the use case for visualizing subject metric data.

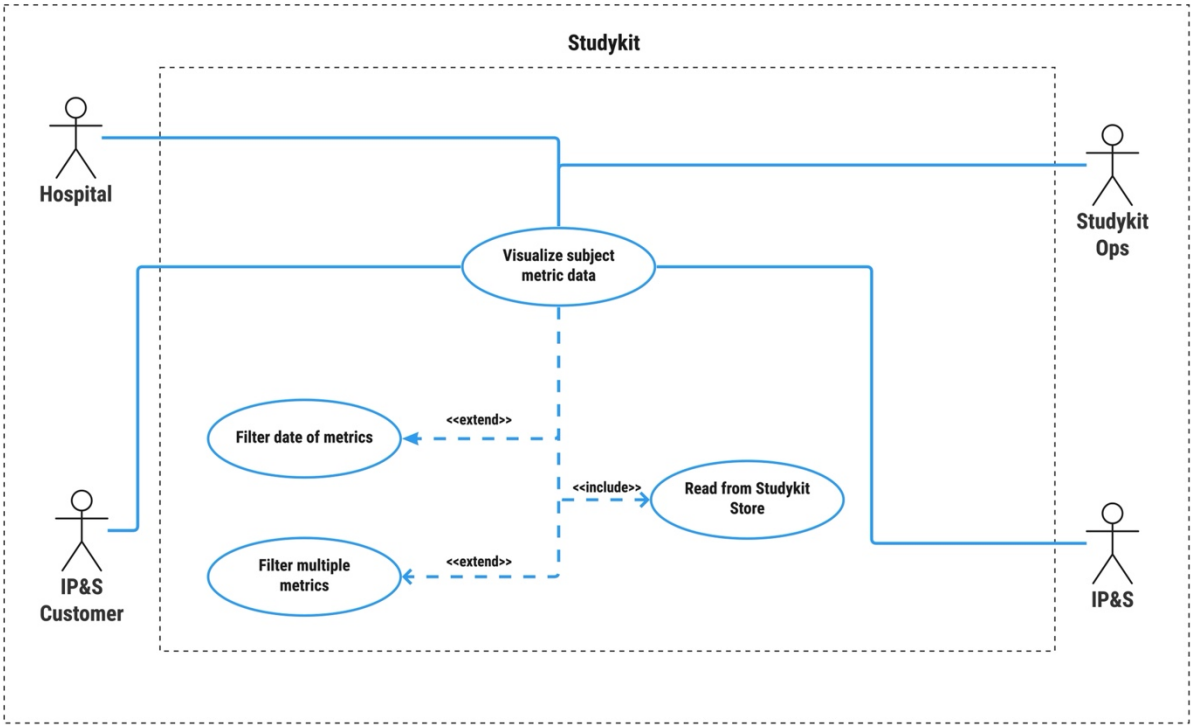



*Figure 14 - Use case: Visualize subject metric data*

1. When the user wishes to visualize metric data of a subject, the user should select the date of the data and the metrics.
2. The request is proxied from the backend to the Studykit Store query service.
3. If data exists for the selected date and metrics, a JavaScript Object Notation (JSON) response is sent with the data as well as the aggregation period. Large volumes of data are practically impossible to visualize. Hence the Studykit Store automatically aggregates the data.
4. If data does not exist for the selected data and metric, a JSON response is sent with no data.

# 5.System Design

System design focuses on detailing the low-level design of the system based on the architecture as described in Chapter 5 to satisfy the requirements mentioned in Chapter 4. In this chapter, we discuss the design choices for different technologies used to develop the system.

## 5.1 Design choices

**Technology Choice 1:** React JS for frontend

**Brief**: JavaScript is currently the most popular choice for frontend development. It offers many different frameworks and libraries for different solutions.

**Rationale:**

1. The team is quite familiar with React JS.
2. Offers one-way data binding, hence the UI components cannot change without updating the state.
3. Uses a virtual Document Object Model (DOM) to manipulate the object model.

**Discarded Choices:**

1. **Angular JS**: Angular is another popular Single-Page Application (SPA) choice, but it is important to note that it is a framework and not a library. Hence, it is built-in with a lot of features (also called Batteries) which end up not being utilized. Also, it uses real DOM to manipulate the object model, which is quite slower because of this feature, it enables two-way data binding.
2. **Vue JS**: Vue has become quite popular in the recent years due to its simplicity. Although Vue shares a lot of features with React, it does not support function declarative style of programming, nor provides essential hooks out of the box.

**Technology Choice 2:** Node JS for different backend microservices

**Brief:** Currently the market has a lot of very good backend programming languages to offer, both for lightweight and heavyweight use. It really comes down to how the system scales and interacts with the underlying hardware to manage the core load efficiently.

**Rationale:**

1. Built-in cluster module which efficiently manages load across all CPU cores and enables scalability.
2. Node JS is built using an event-driven architecture and thus makes it easier to perform synchronous operations.
3. Philips in-house technology, such as the IAM client, is built using Node JS and integrates flawlessly with other node-based applications.

**Discarded Choices:**

1. **Django:** Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design [4]. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support. Unfortunately, there is no support for Philips' in-house technology.
2. **Laravel**: Even though Laravel is built using Hypertext Preprocessor (PHP), it is slower than Node and Django. Over the years, PHP has developed quite a lot with the newer PHP versions, but it still has an overhead regarding scalability. Additionally, it does not have any support for Philips' in-house technology.

**Technology Choice 3:** TimeScale DB for high volume time-based data

**Brief**: The system initially used a SQL database using the Barista system at Philips, but it was not suited to handling large data volumes. Some devices, such as a PDL, record raw metrics every second. This implies that every second there are 41 data points for Photoplethysmography (PPG) (1 for quality, 4 for Analog to Digital Converter (ADC) gain, 4 for LED power, and 32 for PPG) and 97 data points for Acceleration (1 for quality, 32 for x-axis, 32 for y-axis, and 32 for z-axis). Given the high volume of data being recorded every second, it is more suited towards time-series databases.

**Rationale:**

1. Timescale DB is the first open-source time-series database that scales horizontally and easily processes billions of data points per second while retaining full Structured Query Language (SQL) functionality.
2. The software supports continuous aggregates, from Postgres materialized views, to refresh the query automatically continually and incrementally in the background. The computation is only performed on data that needs to be changed, which makes the process more efficient.

3. TimeScale DB can easily store metadata in any format (even JSON). Also, TimeScale DB supports Array as a valid data type to store data.

**Discarded Choices:**

1. **Influx DB**: Launched in 2013 — Influx DB is a quite popular time-series database choice for many applications since it is schema-less. The downside of Influx DB is that it is quite slow over historical data.
2. **Amazon Web Services Timestream**: Launched in 2020 by Amazon, it is relatively new and not industry ready. It does not support individual updating or deleting of rows. Also, time to ingest data is relatively higher than TimeScale DB and Influx DB.

## 5.2 Benchmark of different time-series databases

A benchmark was required to determine the optimum time-series database for the Studykit use case. Multiple factors were considered to evaluate the different time-series databases, such as:

- Performance
- Schema structure
- Array support
- Binary support
- HSDP support
- Backup capabilities
- CRUD operations

**Performance**

To evaluate performance, a test was conducted where data for ten subjects were uploaded for seven days. The ingestion time taken for this dataset is as follows:

*Table 6 - Timeseries database performance evaluation – ingestion*

| Influx DB | TimeScale DB | AWS Timestream |
|---|---|---|
| 8 hours | 6 hours | 17 hours |

Subsequently, another evaluation was carried out to determine the read speed of the different databases. The databases were also tested on their ability to aggregate data over time. The results are as follows:

*Table 7 - Timeseries database performance evaluation - read*

| Query | Period | Interval | Iterations | Influx DB (seconds) | TimeScale DB (seconds) | AWS Timestream (seconds) |
|---|---|---|---|---|---|---|
| **Get all ppg and acc** | 30 mins | None | 10 | 0.633739 | 0.15798 | 1.4799 |
| **Get all ppg and acc** | 1 hour | None | 10 | 1.27104 | 0.314785 | 1.5255 |
| **Get all ppg and acc** | 6 hours | None | 1 | 7.71209 | 1.94029 | 3.74605 |
| **Get single metric (acc)** | 5 mins | Mean (interval – 1 min) | 20 | 0.017913 | 0.00101 | 0.184031 |
| **Get single metric (ppg)** | 5 mins | Mean (interval – 1 min) | 20 | 0.017066 | 0.00093 | 0.157554 |
| **Get single metric (acc)** | 1 day | Mean (interval – 1 min) | 20 | 2.74532 | 0.0294056 | 0.335798 |
| **Get single metric (acc)** | 1 day | Median (interval – 1 min) | 20 | 2.81018 | 0.0360835 | 0.30192 |
| **Get single metric (acc)** | 7 days | Mean (interval – 5 min) | 4 | 15.9272 | 0.0862099 | 0.938409 |
| **Get single metric (acc)** | 7 days | Median (interval – 5 min) | 4 | 16.0279 | 0.213224 | 1.08088 |
| **Get single metric (acc)** | 7 days | Mean (interval – 1 day) | 4 | 15.0114 | 0.0792669 | 0.922445 |
| **Get single metric (acc)** | 7 days | Median (interval – 1 day) | 4 | 15.3372 | 0.207651 | 0.983989 |

Other evaluations are listed as below:

*Table 8 - Timeseries database evaluation – multiple factors*

| | **Influx DB** | **Timescale DB** | **AWS Timestream** |
|---|---|---|---|
| **Schema Structure** | Schema less | Schema based | Schema less |
| **Array Support** | No | Yes | Yes |
| **Binary Support** | No | Yes | No |
| **HSDP Support** | No | No | No |
| **Backup capabilities** | Yes | Yes | No |
| **CRUD operations** | All | All | No support for update or delete of individual records. |

# 6.Implementation

## 6.1 Introduction

This chapter presents the implementation of various microservices of the system based on the architecture and design as mentioned in chapter 5 and 6 respectively.

## 6.2 Studykit Portal

The portal uses technologies such as React JS and TypeScript and the design is based on the dDLS from Philips. The dDLS is a complex specification that covers different web components, color schemes, asset libraries, motion elements, etc. The dDLS also specifies the margin and padding between different elements. Since, the UI components are common between different systems at Philips, it greatly reduces the learning curve. Figure 15 shows the overview of the Studykit Portal designed using the dDLS.


*Figure 15 – Studykit Portal*

React JS is currently the most popular frontend library of choice for many companies. It allows users to create User Interfaces by combining JS with Hyper Text Markup Language (HTML).

This process of combining JS with HTML is called JavaScript eXtensible markup language (JSX). React is only concerned with state management and rendering that state to the DOM.

Managing the state of a React JS application has been complex since its inception, and hence, it has required an external library to manage the global state of the application. With the continuous development and improvement of React JS, it is no longer complex. But it does come with its own caveats. The current way of dealing with the state can be expressed as:

- Application state - refers to the local state of the application, independent of the server data.
- Server state - refers to the state of the server and is dependent on the data returned by the server.

Studykit Portal maximizes both these aspects of state management. For server state management, the portal uses Stale-While-Revalidating (SWR) library. SWR is an open-source library by Vercel, which does the following tasks:

- fetches data from the server
- re-fetches data without re-rendering the application
- maintains the old cache of the fetched data
- automatically fetches data upon re-focusing on the application, after certain amount of time defined by the user, and when the user reconnects to the internet.

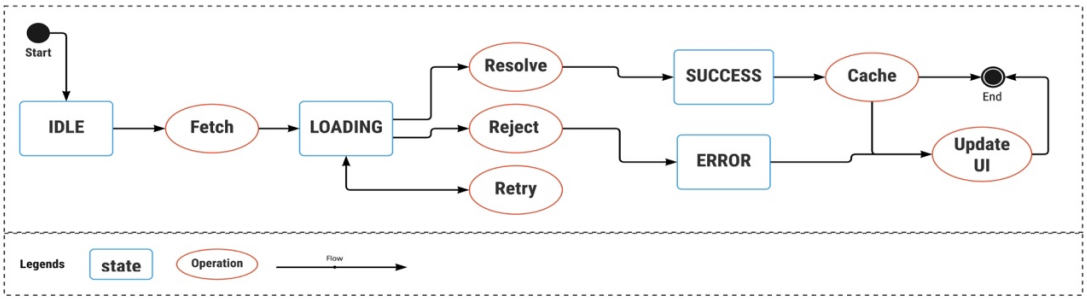



*Figure 16 - State management for Server-side data*

Figure 16 shows the state management of the server state achieved using the SWR library.

For application state, the portal uses the new useContext hook provided out of the box by React JS. The useContext hook uses the Provider and Consumer model to share the application state between different components. Every component i.e., consumer that wishes to use the application state should be wrapped inside the provider.

With these techniques, modern JS frameworks and libraries don't have the problem called "prop drilling". Imagine a scenario where the parent component needs to pass some data (props) to a child component four levels down. This process of passing down props is defined as prop drilling. Communication between different components is made possible with the help of passing props from one component to the other. Since the components are individual pieces of code that can be used at multiple places in the application or inside another component, it leads to non-modular code and, greatly reduces the maintainability of the code.

## 6.3 MSX

The MSX is a powerful communications hub, built using a stable and secure Linux environment. It uses Bluetooth to connect wirelessly to the wearable device. It pairs automatically when it encounters the PDL device. Upon pairing, it fetches the raw metrics from the device. Subsequently, the MSX fetches the data from the device every minute after the initial fetch. The received data is in binary format and it is the responsibility of the MSX to convert it to WeST format files.

The files are uploaded to the cloud using the 4G data dongle attached to the MSX. To ensure security at all levels, the MSX is equipped with a security token. This security token has access to upload the files to the cloud only. Additionally, the MSX does not know which subject it belongs to; thus, it only uploads the device identifier and the metric files. The device identifier can either be a MAC address or a serial number.

Inside the cloud, the endpoint is pointing towards the backend service. This service looks up the device identifier in the FHIR database using the Clinical Data Repository (CDR) interface. CDR is another in-house technology from Philips, which was developed to efficiently and securely use FHIR. As per the Device and Patient specification of FHIR, any device can be

associated with a particular subject. Making use of this association, the backend service performs a global level look-up on all CDR instances to look for that device identifier.

Upon successful query, it forwards the raw metrics along with the newly retrieved subject id to the Studykit Store. Studykit Store is another microservice application developed for this project. It comprises a service layer on top of the TimeScale DB. The service layer enables three top level functionalities are query, import, and export.

The backend service never interprets the files that are uploaded by the MSX. The purpose of the backend service is to perform a look-up for subject identifier based on the received device identifier. To maintain performance while importing, it uses the same upload stream as received from the MSX and proxies it to the Studykit Store. The Studykit Store has a parser for the files. The parser reads the metadata of each file to identify the metric type based on the format id field. Based on the format id, it parses the files. The parser is thorough and individually checks each data point.

## 6.4 Work Orchestration

The algorithms are the most crucial, integral, and important aspect of this entire system. They reside under the Analytics Engine. The Analytics Engine consists of Apache Airflow and Node JS-based service written on top of it to interact with the Airflow remotely via the Representational State Transfer (REST) API.

Apache Airflow is an open-source platform, written in Python, developed by AirBnb in 2014. It solves the problem of hard-to-manage complex workflows. It allows easy management of schedulable jobs with the ease of scalability and extensibility.

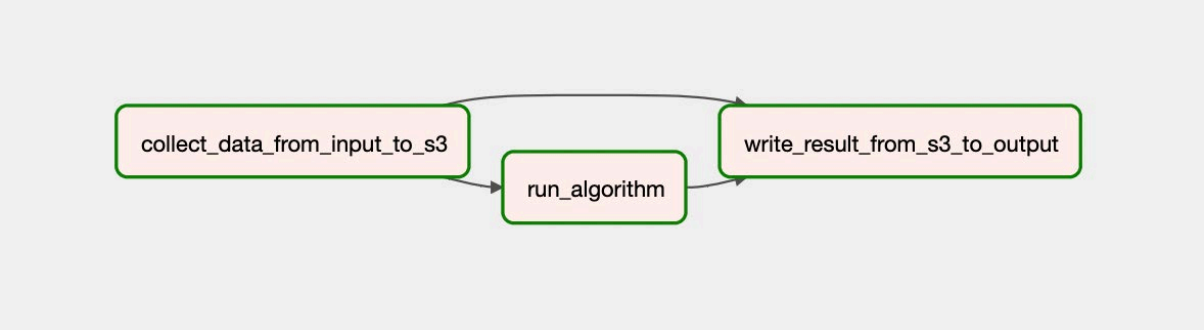



*Figure 17 - DAG*

A *DAG* (Directed Acyclic Graph) is the core concept of Airflow, collecting tasks together, organized with dependencies and relationships to say how they should run [5]. As seen in Figure 17, it dictates the order of how three tasks run. The directionality between the graphs describes the dependency. These tasks can be scheduled to run periodically or once as per the user's configuration. For periodic scheduling, Airflow uses the Linux cron command syntax. The syntax for creating is illustrated by Figure 18.

The cron expression is made of five fields. Each field can have the following values.

| * | * | * | * | * |
|---|---|---|---|---|
| minute (0-59) | hour (0 - 23) | day of the month (1 - 31) | month (1 - 12) | day of the week (0 - 6) |

*Figure 18 - Cron Syntax [6]*

The Node JS-based service is designed to cater to three solutions:

1. To maintain the algorithms, connections, and their states with the help of a database
2. To ensure other microservices can interact with airflow
3. To perform CRUD operations on the DAGs.

The Analytics Engine works in combination with the Studykit Store Amazon S3. The Studykit Store is responsible for delivering the data for a particular subject for a given date and time, whereas Amazon S3 is responsible for storing the state of the algorithms. Algorithms maintain a state file, which allows the algorithm to become more accurate over time.

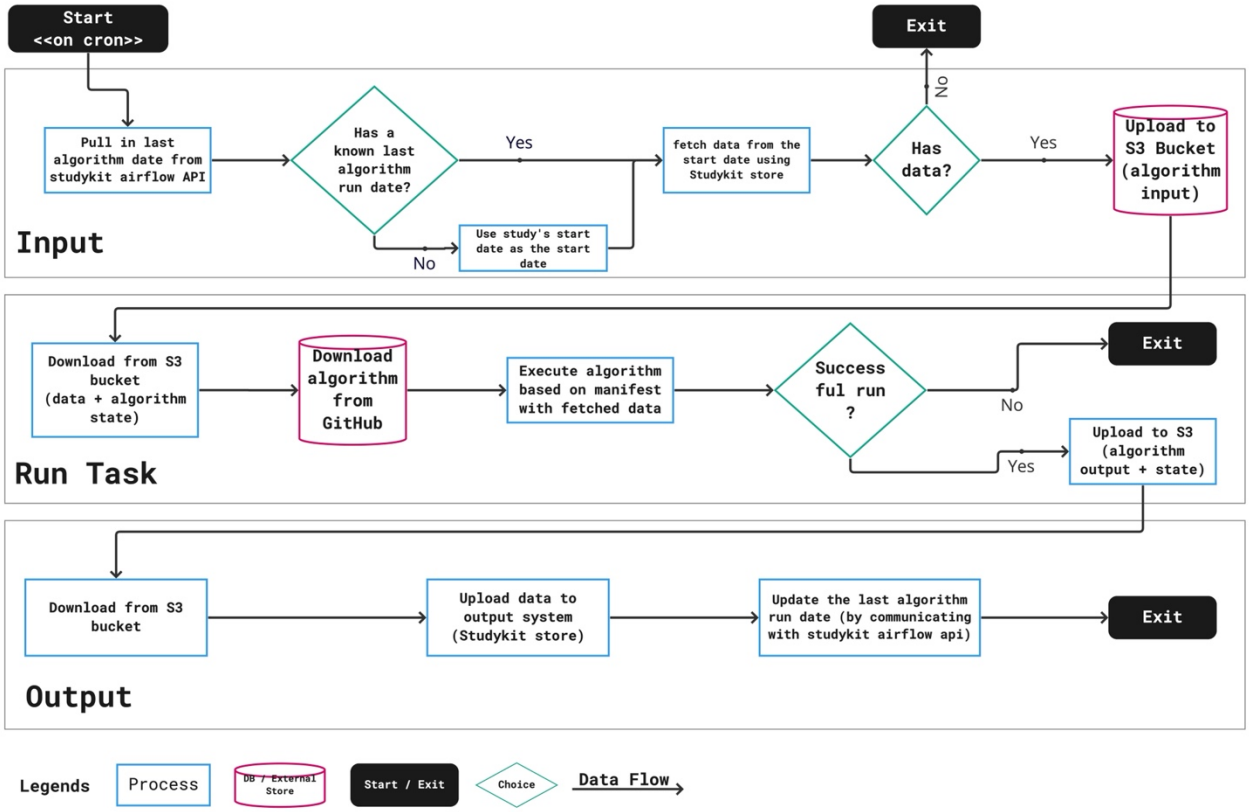



*Figure 19 - Work Orchestration*

Figure 19 shows the work orchestration broken down into three steps: Input, Run Task, and Output. As shown, the process starts when the job is queued based on the cron specification inside the Input Task. From the database using the Studykit API, the last known algorithm run is fetched, and the new start date is calculated. If this is the first instance of the algorithm run, then Study's start date is used as the start date. Then we query the Studykit Store based on the start time and retrieve the data. The entire work orchestration process ends if no data is returned by the Studykit Store. If Studykit Store returns the data, subsequently, it is uploaded to an Amazon S3 bucket.

Inside Run Task, data along with the last state of the algorithm (if exists) is retrieved from S3. Subsequently, the algorithm is downloaded from GitHub. GitHub is used as an external store where the algorithms are maintained. It allows for maintaining the code history of the algorithms and maintaining the versions of the algorithms. After the download is complete, the algorithms run on the raw data that was downloaded. The execution is dependent upon the manifest that is returned by the Studykit Store along with the data. The manifest includes metadata from the algorithm as well as the personal attributes of the subject. If the run is not

successful, the work orchestration process exits with a failed status. If the algorithm runs successfully, the derived metrics along with the new metrics states are uploaded to Amazon S3 bucket.

Finally, inside Output Task, the derived data is downloaded from the Amazon S3 bucket and uploaded to Studykit Store. Subsequently, the last algorithm run date is updated using the Studykit API and the work orchestration ends with a success state.

## 6.5 Studykit Store

The Studykit Store represents another crucial sub-system, which deals with the storing of metrics. The Studykit Store is the first ever solution at Philips, which is built using a time-series based database. The nature of healthcare data is largely dependent upon time, and hence relational databases do not add a lot of benefit. TimeScale DB is built on top of Postgres SQL and provides the functionalities of a relational database as well as a time-series database.

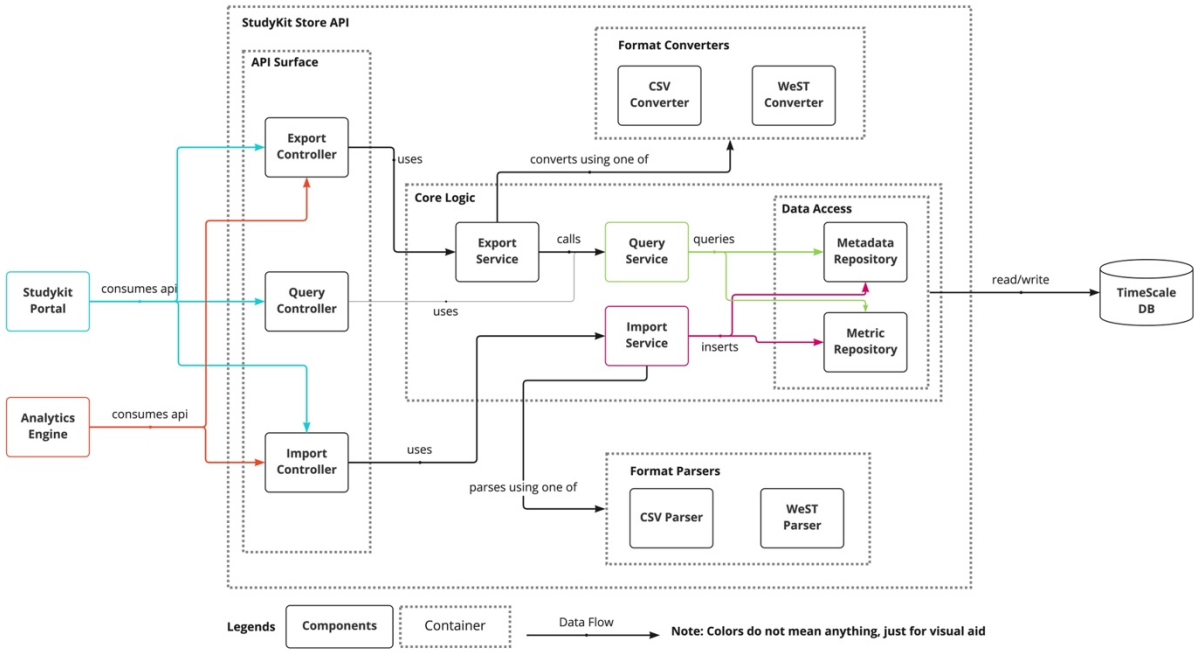



*Figure 20 - Studykit Store Container Diagram*

The Studykit Store also has an API layer, which allows other microservices to interact with the database. Two different services, i.e., Import service and Export service, handle the parsing of the data. The import service takes care of the files that are uploaded by MSX and Manual

Upload via the Studykit Portal. The export service takes care of allowing users to extract all or selected data from the database into a downloadable file. Lastly, there is the Query service, which allows for just reading of the data. The export service under the hood uses the query service to retrieve data. Figure 20, outlines the entire structure in detail. Note, the Studykit Portal and Analytics Engine do not directly communicate with the Studykit Store. This is just for visual representation purposes.

## 6.6 Logging and Monitoring

All microservices log to allow for speedy debugging and to guarantee that any issues are immediately detected and remedied. Tracing and managing server logs can be an incredibly tedious process, so we use an external system to ensure proper management and search indexing. Elastic Search by Elastic enables us to track and visualize server logs in an efficient way. It maintains logs from different sources, sorts them based on time, and allows for quick search and analysis. Kibana is a User Interface by Elastic, which enables us to fully utilize the power of Elastic Search. Both tools are open-source and have great community support as well. Figure 21 shows an example of Apache Airflow log.

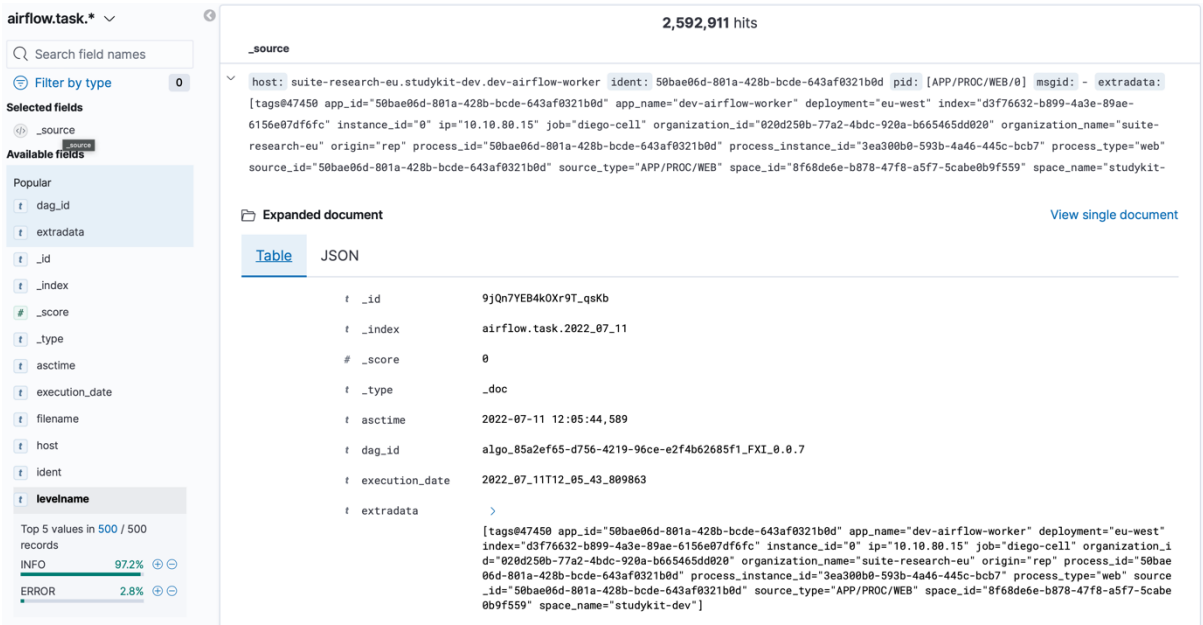



*Figure 21 - Kibana log*

For monitoring the Airflow DAGs, another open-source technology is used, called Grafana. Grafana allows you to query, visualize, alert on, and understand metrics no matter where they

are stored. Grafana lets the user connect data sources and create dashboards using the user interface. It shares a lot of features with Kibana, but the key difference is that it does not rely on Elastic Search under the hood. It is independent of the data source and can connect to any relational or non-relational database.

To enable Grafana's connection with Airflow, another open-source tool called Prometheus is used in conjunction with statsd_exporter. Prometheus is an open-source system monitoring and alerting toolkit originally built at Soundcloud. Statsd_exporter, as the name implies, is a stats exporter which runs as a daemon in the background. Airflow has built-in support for statsd_exporter. Moreover, Prometheus also has support for statsd_exporter and hence, it pulls the data and passes it to Grafana.

Figure 22, illustrates the stats of the development airflow cluster for the past one hour. Various stats are shown, such as Scheduler heartbeat (fetched every 30 seconds), Dagbag size (total number of DAGs), Zombie killed (a process is defined as zombie when it is terminated yet still has record on the process table), total failed tasks, executor open slots (maximums DAGs that can run at a given time).


*Figure 22 - Grafana dashboard*

# 7.Verification & Validation

This chapter discusses the verification and validation of the system developed. The difference between the two terms is mostly brought about by the function of specifications. Verification is the process of ensuring the software adheres to specifications, whereas validation is the process of making sure the specification captures the customer's needs.

## 7.1 Verification

Verification of the system requirements along with their status (satisfied, not satisfied and partial) is as follows:

Table 9 shows the verification status of general requirements.

*Table 9 - Verification - General requirements*

| Requirement Id | Priority | Requirement | Status |
|---|---|---|---|
| **GR01** | Must | The portal should be accessible with any modern browser (i.e., support for ECMAScript v6). | Satisfied |
| **GR02** | Must | The portal should be able to visualize metric data. | Satisfied |
| **GR03** | Must | The portal shall be deployable on Philips' network. | Satisfied |
| **GR04** | Must | The system shall use Philips' IT-approved tools and technologies. | Partial |
| **GR05** | Must | The database should normalize the data before storing. | Satisfied |
| **GR06** | Must | The algorithms should run independently for each subject of a study. | Satisfied |
| **GR07** | Must | The system should store patient/subject information using FHIR. | Satisfied |

| | | | |
|---|---|---|---|
| **GR07** | Must | The system should provide a logging and monitoring mechanism. | Satisfied |
| **GR08** | Won't have | The system should be able to live stream data and show visualization. | Not satisfied |

Reason for partial satisfaction:

- GR04: Since Studykit is the first system at Philips, which uses TimeScale DB, it is currently not listed under Philips' IT approved tools and technologies. However, the team is currently in the process of getting TimeScale DB approved.

Table 10, 11, and 12 show the status of functional requirements of different microservices.

*Table 10 - Verification – Studykit Portal – Functional requirements*

| Requirement Id | Priority | Requirement | Status |
|---|---|---|---|
| **FR01** | Must | The portal shall allow the user to import data metric files. | Satisfied |
| **FR02** | Must | The portal shall allow the user to export data metric files. | Satisfied |
| **FR03** | Must | The portal shall allow the user to select a particular date for data visualization. | Satisfied |
| **FR04** | Must | The portal shall allow the user to have a high-level view if the data is present for a particular subject or not. | Satisfied |
| **FR05** | Must | The portal shall allow the user to select multiple metrics for visualization. | Satisfied |
| **FR06** | Must | The portal shall inform the user if the visualization data is aggregated. | Satisfied |
| **FR07** | Should | The portal shall inform the user regarding the state of algorithm processing after successful import. | Satisfied |

*Table 11 - Verification - Studykit Store - Functional requirements*

| Requirement Id | Priority | Requirement | Status |
|---|---|---|---|
| **FR08** | Must | The store shall parse different metric data files based on their specification when importing. | Satisfied |
| **FR09** | Must | The store shall construct different metric data files based on their specification when exporting. | Satisfied |
| **FR10** | Must | The store shall selectively map out of range data into their valid range. | Satisfied |
| **FR11** | Must | The store shall track the source information of each metric. | Satisfied |
| **FR12** | Must | The store shall be able to parse metric files irrespective of the white space character (i.e., tab or spaces) | Satisfied |

*Table 12 - Verification - Analytics Engine - Functional requirements*

| Requirement Id | Priority | Requirement | Status |
|---|---|---|---|
| **FR13** | Must | The engine shall provide an API interface for Apache airflow | Satisfied |
| **FR14** | Must | The engine shall store configurations for each subject and study | Satisfied |
| **FR15** | Could | The engine shall support live streaming and processing of metric data. | Satisfied |
| **FR16** | Should | The engine API shall provide progress update on the algorithm state post data manual import. | Satisfied |

Table 13 shows the status of the non-functional requirements of the system.

*Table 13 - Verification - Non-functional requirements*

| Requirement Id | Priority | Aspect | Description | Status |
|---|---|---|---|---|
| **NFR01** | Must | Security | The system shall authenticate and authorize user and each request using Philips' Identity Access Management (IAM) Client. | Satisfied |
| **NFR02** | Must | Privacy | The system shall be deployed as per regional privacy laws. | Satisfied |
| **NFR03** | Should | Accessibility and Usability | The system shall adhere to Philips' dDLS specification for the user interface. | Satisfied |
| **NFR04** | Should | Compliance | The system shall adhere to all compliances of the respective country, where it is deployed. | Satisfied |
| **NFR05** | Should | Maintainability | Each microservice of the system should be developed with the best practices so, it can be maintained easily over the lifetime of the system. | Satisfied |
| **NFR06** | Should | Extensibility | Each microservice of the system should be modular such that it can be extended in the future without any architectural changes. | Satisfied |

## 7.2 Validation

Validation is the process of checking if the developed system satisfies the needs of the stakeholders. The Studykit system was validated iteratively within the team during the development phase. The system was also validated in collaboration with stakeholders, both internal, and external. A User Interface mockup of the Studykit portal was developed using a tool called Sketch. The dDLS has a kit for Sketch, that allows the user to use the required building blocks. The demo of the User Interface was given to the internal (IP&S) and external (contact people at hospitals) stakeholders. Subsequently, feedback received from the stakeholders post the demo was taken into consideration for taking the next steps. Naturally, not all the suggestions are considered directly, because the aim of this project is to develop the MVP, and not the final solution. The suggestions were not ignored but added to the backlog for future enhancements.

## 7.3 Security Checks

Philips uses a proprietary technology called Black Duck from Synopsys. Black Duck essentially helps teams manage the security, quality, and license compliance risks that come from the use of open source and third-party code in applications and containers [7].

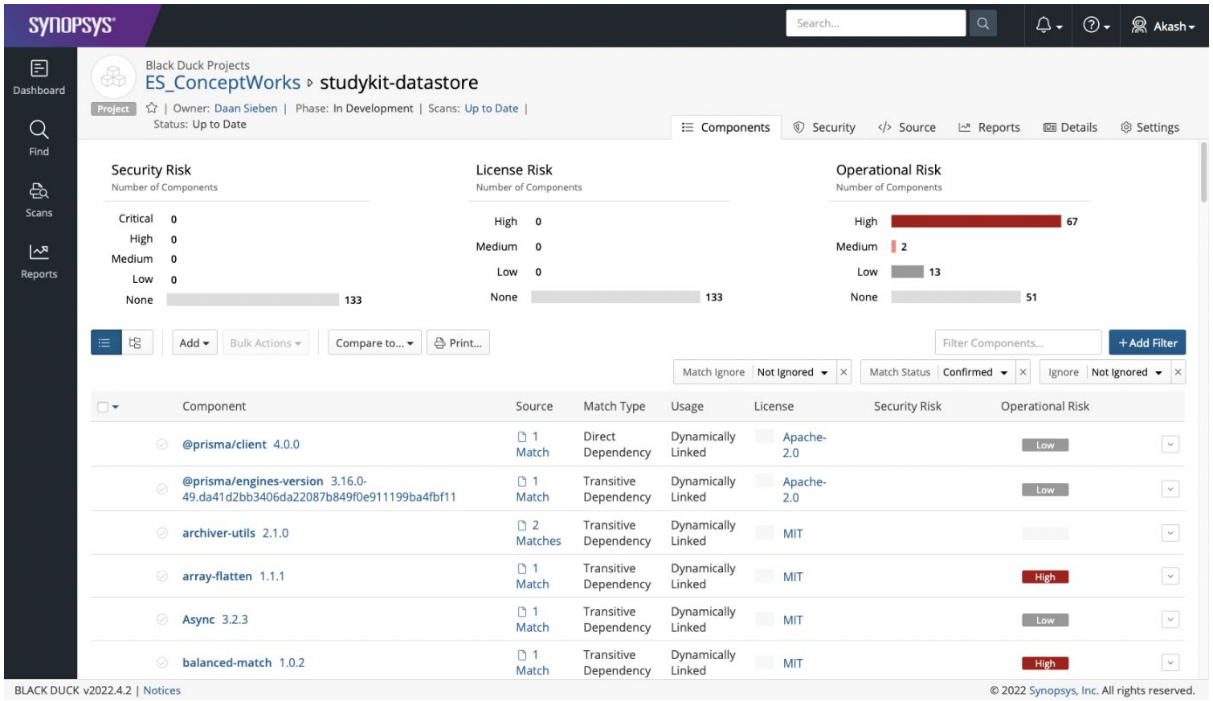



*Figure 23 – Black Duck security analysis of Studykit Store*

Figure 23 shows the security analysis of the Studykit Store. As seen from the figure, even though there is no security risk involved, there is still an operational risk. This is because, it also analyzes the dependencies of the different libraries and when each of the dependencies were last updated.

## 7.4 Clean Code

Philips uses an industry-leading open-source technology called SonarQube to maintain and improve different applications and tools for clean code. SonarQube is a self-managed, automatic code review tool that systematically helps you deliver Clean Code [8]. SonarQube and Black Duck are a part of Studykit's Continuous Integration and Continuous Deployment (CI/CD) pipeline.

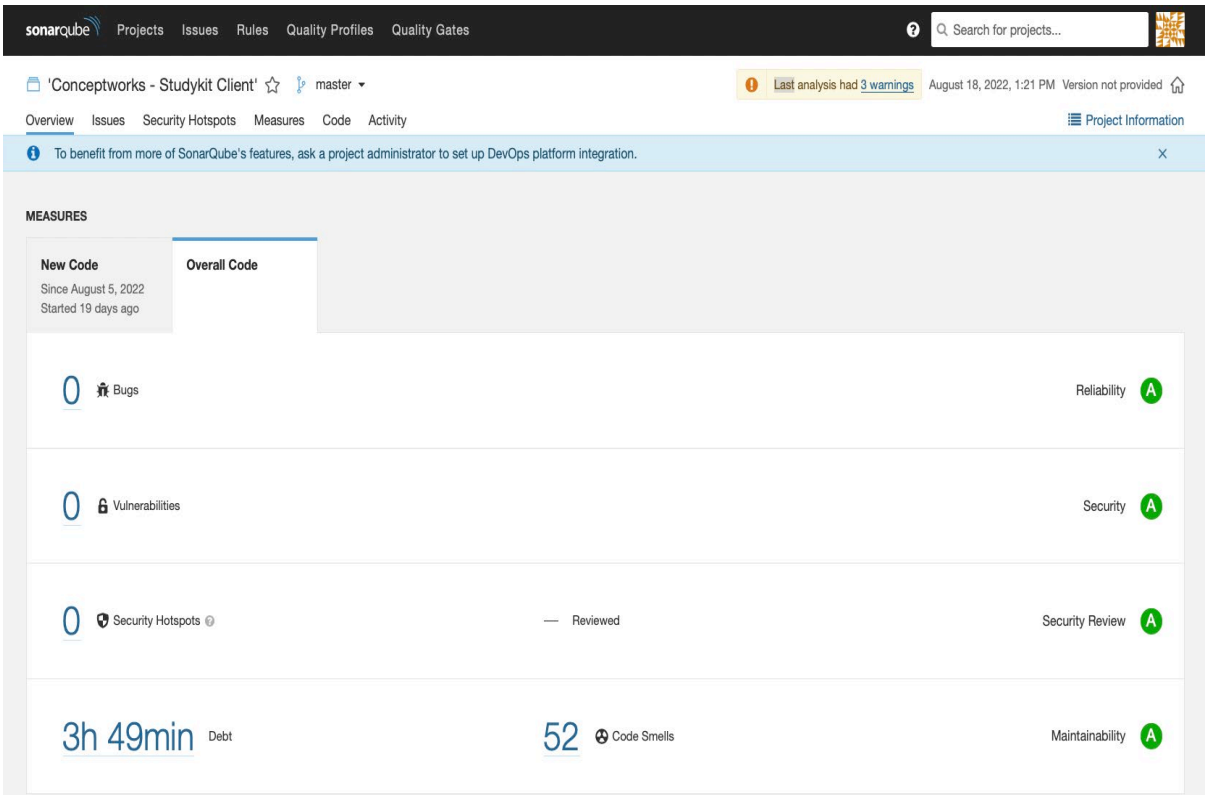
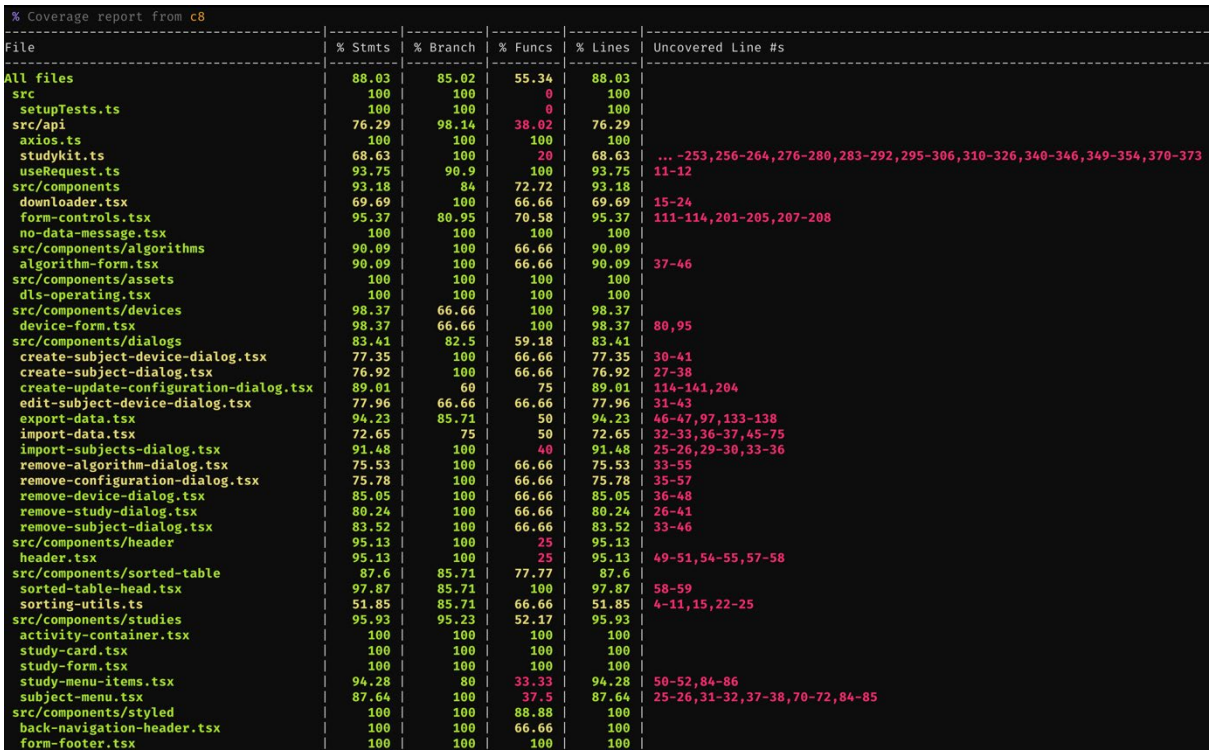



*Figure 24 - SonarQube Analysis of Studykit Client*

Figure 24 shows the SonarQube analysis of Studykit Client. As seen from the figure, it gives a rating on bugs, vulnerabilities, security hotspots, debt, and code smells.

## 7.5 Code Coverage

Code coverage is a metric that reflects how thoroughly the program's source code has been tested. The coverage tool generates a report, usually expressed in percentages, of all the files of a given project. Code coverage is a good way to measure the quality and maturity of a project's code and is essential for understanding the health of the project's code base. There is no industry standard for the minimum percentage of coverage, but anything above 75 percent of coverage is considered good. Figure 25 shows code coverage for the Studykit Portal, which currently stands at 88.03 percent. Similarly, other microservices are tested and checked for code coverage before deployment. Currently, the code coverage for Studykit Store and Studykit Backend is at 68.9 and 70.9 percent respectively. However, we made sure that the critical parts are covered.



% Coverage report from c8

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
|---|---|---|---|---|---|
| All files | 88.03 | 85.02 | 55.34 | 88.03 | |
| src | 100 | 100 | 0 | 100 | |
| setupTests.ts | 100 | 100 | 0 | 100 | |
| src/api | 76.29 | 98.14 | 38.02 | 76.29 | |
| axios.ts | 100 | 100 | 100 | 100 | |
| studykit.ts | 68.63 | 100 | 20 | 68.63 | ...-253,256-264,276-280,283-292,295-306,310-326,340-346,349-354,370-373 |
| useRequest.ts | 93.75 | 90.9 | 100 | 93.75 | 11-12 |
| src/components | 93.18 | 84 | 72.72 | 93.18 | |
| downloader.tsx | 69.69 | 100 | 66.66 | 69.69 | 15-24 |
| form-controls.tsx | 95.37 | 80.95 | 70.58 | 95.37 | 111-114,201-205,207-208 |
| no-data-message.tsx | 100 | 100 | 100 | 100 | |
| src/components/algorithms | 90.09 | 100 | 66.66 | 90.09 | |
| algorithm-form.tsx | 90.09 | 100 | 66.66 | 90.09 | 37-46 |
| src/components/assets | 100 | 100 | 100 | 100 | |
| dls-operating.tsx | 100 | 100 | 100 | 100 | |
| src/components/devices | 98.37 | 66.66 | 100 | 98.37 | |
| device-form.tsx | 98.37 | 66.66 | 100 | 98.37 | 80,95 |
| src/components/dialogs | 83.41 | 82.5 | 59.18 | 83.41 | |
| create-subject-device-dialog.tsx | 77.35 | 100 | 66.66 | 77.35 | 30-41 |
| create-subject-dialog.tsx | 76.92 | 100 | 66.66 | 76.92 | 27-38 |
| create-update-configuration-dialog.tsx | 89.01 | 60 | 75 | 89.01 | 114-141,204 |
| edit-subject-device-dialog.tsx | 77.96 | 66.66 | 66.66 | 77.96 | 31-43 |
| export-data.tsx | 94.23 | 85.71 | 50 | 94.23 | 46-47,97,133-138 |
| import-data.tsx | 72.65 | 75 | 50 | 72.65 | 32-33,36-37,45-75 |
| import-subjects-dialog.tsx | 91.48 | 100 | 40 | 91.48 | 25-26,29-30,33-36 |
| remove-algorithm-dialog.tsx | 75.53 | 100 | 66.66 | 75.53 | 33-55 |
| remove-configuration-dialog.tsx | 75.78 | 100 | 66.66 | 75.78 | 35-57 |
| remove-device-dialog.tsx | 85.05 | 100 | 66.66 | 85.05 | 36-48 |
| remove-study-dialog.tsx | 80.24 | 100 | 66.66 | 80.24 | 26-41 |
| remove-subject-dialog.tsx | 83.52 | 100 | 66.66 | 83.52 | 33-46 |
| src/components/header | 95.13 | 100 | 25 | 95.13 | |
| header.tsx | 95.13 | 100 | 25 | 95.13 | 49-51,54-55,57-58 |
| src/components/sorted-table | 87.6 | 85.71 | 77.77 | 87.6 | |
| sorted-table-head.tsx | 97.87 | 85.71 | 100 | 97.87 | 58-59 |
| sorting-utils.ts | 51.85 | 85.71 | 66.66 | 51.85 | 4-11,15,22-25 |
| src/components/studies | 95.93 | 95.23 | 52.17 | 95.93 | |
| activity-container.tsx | 100 | 100 | 100 | 100 | |
| study-card.tsx | 100 | 100 | 100 | 100 | |
| study-form.tsx | 100 | 100 | 100 | 100 | |
| study-menu-items.tsx | 94.28 | 80 | 33.33 | 94.28 | 50-52,84-86 |
| subject-menu.tsx | 87.64 | 100 | 37.5 | 87.64 | 25-26,31-32,37-38,70-72,84-85 |
| src/components/styled | 100 | 100 | 88.88 | 100 | |
| back-navigation-header.tsx | 100 | 100 | 66.66 | 100 | |
| form-footer.tsx | 100 | 100 | 100 | 100 | |

*Figure 25 - Code Coverage: Studykit Portal*

# 8.Project Management

In this section, we describe how the system described in the previous chapters was managed. This section describes the planning and execution of the entire project.

## 8.1 Project Organization

Philips uses the world-renowned Scaled Agile Framework (SAFe) to manage and promote agile practices at an enterprise scale [9]. SAFe is a unique tool that promotes alignment, collaboration, and delivery across many agile teams. A Value Stream (VS) in SAFe represents the series of steps that an organization uses to implement solutions that provide a continuous flow of value to the customer/end-product.

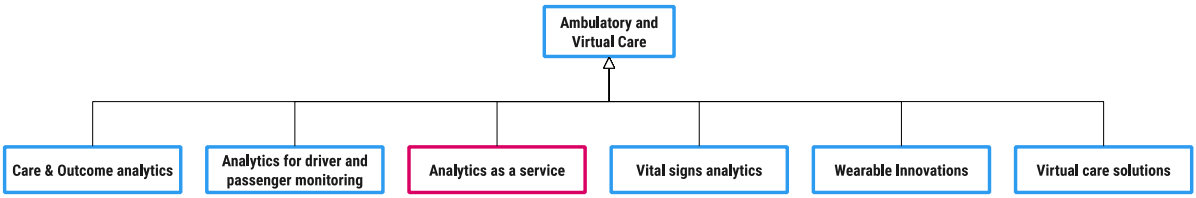
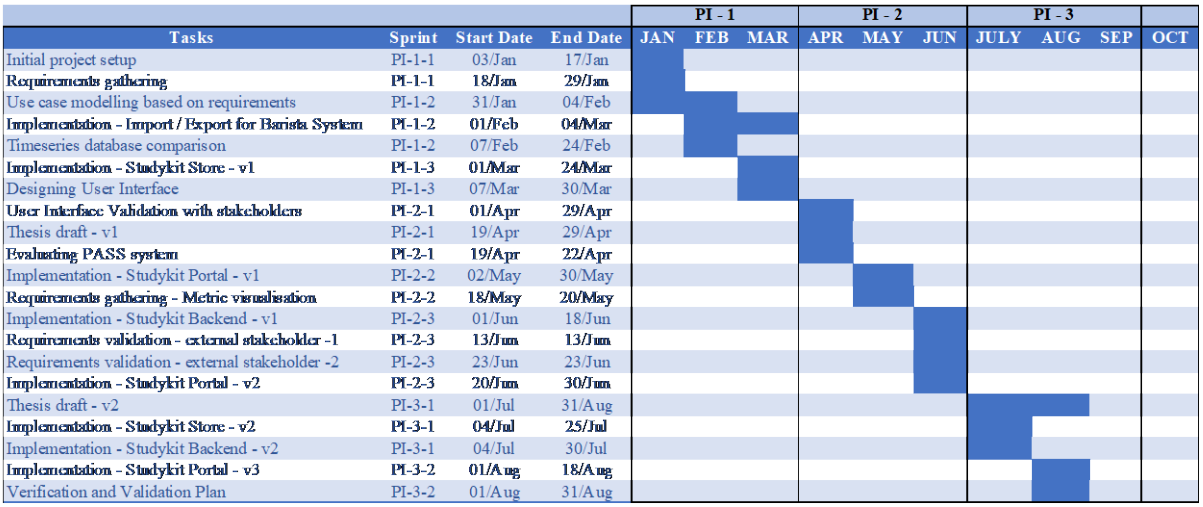



*Figure 26 - Value stream structure*

Figure 26 shows the structure of the Ambulatory and Virtual Care VS. Studykit is a part of the Analytics as a Service epic under the Ambulatory and Virtual Care VS. An epic represents a series of user stories that share a border strategic objective. Each Epic under the VS follows the iterative approach of development. The iteration is identified as a Program Increment (PI) and is three months in duration. The PI is a key milestone in the process and sets the stage for future development. A PI is further split into three sprints, which span for one month each. Sprints ensure that the team is making progress towards a defined goal and that everyone is staying on track. The work done during each sprint is shown by Figure 27.



| Tasks | Sprint | Start Date | End Date | PI - 1 JAN | FEB | MAR | PI - 2 APR | MAY | JUN | PI - 3 JULY | AUG | SEP | OCT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial project setup | PI-1-1 | 03/Jan | 17/Jan | | | | | | | | | | |
| Requirements gathering | PI-1-1 | 18/Jan | 29/Jan | | | | | | | | | | |
| Use case modelling based on requirements | PI-1-2 | 31/Jan | 04/Feb | | | | | | | | | | |
| Implementation - Import / Export for Barista System | PI-1-2 | 01/Feb | 04/Mar | | | | | | | | | | |
| Timeseries database comparison | PI-1-2 | 07/Feb | 24/Feb | | | | | | | | | | |
| Implementation - Studykit Store - v1 | PI-1-3 | 01/Mar | 24/Mar | | | | | | | | | | |
| Designing User Interface | PI-1-3 | 07/Mar | 30/Mar | | | | | | | | | | |
| User Interface Validation with stakeholders | PI-2-1 | 01/Apr | 29/Apr | | | | | | | | | | |
| Thesis draft - v1 | PI-2-1 | 19/Apr | 29/Apr | | | | | | | | | | |
| Evaluating PASS system | PI-2-1 | 19/Apr | 22/Apr | | | | | | | | | | |
| Implementation - Studykit Portal - v1 | PI-2-2 | 02/May | 30/May | | | | | | | | | | |
| Requirements gathering - Metric visualisation | PI-2-2 | 18/May | 20/May | | | | | | | | | | |
| Implementation - Studykit Backend - v1 | PI-2-3 | 01/Jun | 18/Jun | | | | | | | | | | |
| Requirements validation - external stakeholder -1 | PI-2-3 | 13/Jun | 13/Jun | | | | | | | | | | |
| Requirements validation - external stakeholder -2 | PI-2-3 | 23/Jun | 23/Jun | | | | | | | | | | |
| Implementation - Studykit Portal - v2 | PI-2-3 | 20/Jun | 30/Jun | | | | | | | | | | |
| Thesis draft - v2 | PI-3-1 | 01/Jul | 31/Aug | | | | | | | | | | |
| Implementation - Studykit Store - v2 | PI-3-1 | 04/Jul | 25/Jul | | | | | | | | | | |
| Implementation - Studykit Backend - v2 | PI-3-1 | 04/Jul | 30/Jul | | | | | | | | | | |
| Implementation - Studykit Portal - v3 | PI-3-2 | 01/Aug | 18/Aug | | | | | | | | | | |
| Verification and Validation Plan | PI-3-2 | 01/Aug | 31/Aug | | | | | | | | | | |

*Figure 27 - Project timeline*

## 8.2 Project Milestones

This section describes the different milestones that contributed towards successfully delivering the project. Table 14 shows the important milestones and the delivery date.

*Table 14 - Project Milestones*

| Milestone | Estimated Date |
|---|---|
| Time-series database comparison | 24 February 2022 |
| Studykit Store Implementation V1 | 30 March 2022 |
| UI Mockup of Studykit Portal | 29 April 2022 |
| Studykit Portal Implementation V1 | 30 May 2022 |
| Studykit Backend Implementation V1 | 18 June 2022 |
| Studykit Portal Implementation V2 | 30 June 2022 |
| Studykit Store Implementation V2 | 25 July 2022 |
| Studykit Backend Implementation V2 | 30 July 2022 |
| Studykit Portal Implementation V3 | 18 August 2022 |

## 8.3 Risk Analysis

This section describes the risks that were identified during the project. Table 15 represents the different risks that were identified along with their likelihood, impact, and mitigation steps.

*Table 15 - Risk assessment*

| Risk | Likelihood | Impact | Mitigation |
|---|---|---|---|
| The trainee is ill for a duration longer than 10 days | Medium | High | • Negotiate requirements.<br>• Add buffer period in project timeline |
| Delay in progress with respect to expected timeline | Medium | High | • Work on documentation in the free time.<br>• Implement small modules which can be hot plugged into the system later. |
| The trainee does not receive the tools on time (e.g., Laptop) | High | High | • Gather the requirements<br>• Understand the domain |
| Lack of domain knowledge | Medium | Medium | • Learn the required skill proactively.<br>• Pair programming with other engineers to speed up the process. |
| Unscheduled holidays of other team members | Low | Medium | • Be independent enough to take charge in their absence.<br>• If something is blocking, consult other members for a possible solution. |
| Miscommunication among the team members | Medium | High | • Ask open questions to avoid doubts<br>• Whenever in doubt, feel free to reach other members over MS Teams when feasible |

# 9.Conclusions

This chapter summarizes the project achievements and gives some recommendations for future works.

## 9.1 Results

During this project, we developed the Studykit system that reshapes clinical research, while also helping Philips and their customers. The system provides a testing playground for customers of Philips. It enables customers to explore different algorithms offered by Philips, while abstracting the actual implementation of algorithms. The abstraction of algorithms ensures that Philips can securely provide algorithms to their customers without having to worry about any security issues. In clinical research, the system reduces the effort of collecting feedback, maintaining files for different subjects, and managing data from different subjects.

The system also provides a manual way of importing and exporting the data that mitigates the manual effort that was required with the Barista system. The introduction of the MSX hub solves the problem of supporting multiple devices. The Studykit Store can handle large data volumes while not compromising on sharding and compression. Moreover, the Studykit Store also normalizes the imported data, ensuring the data always conforms to the WeST specifications.

Although this is still an MVP, it is a massive upgrade from the previous technology. The results of this project show many improvements to the system currently in place and a potential way for Philips to carry on in future development of tools concerning clinical research (e.g., the Barista system took 15+ seconds to render a single visualization for a metric but the Studykit system takes 4-5 seconds to render about 5 visualization metric graphs).

## 9.2 Recommendations and future work

Given the time constraint of the project, it was not possible to develop the ultimate solution required for the market. Having achieved the MVP, here are some recommendation and future work for this project:

- **Live Streaming of data:** Currently, the metrics are processed every hour even though the MSX uploads the data every minute whenever there is data incoming. For an interventional study, it is required that the system scales up and supports live streaming and processing of metrics.
- **Medically checked and certified:** Provided that the future scope of the system is also to support interventional study. It is necessary to have the system medically checked and certified for safety, privacy, robustness, downtime, etc.
- **Machine Learning on data:** The world is evolving and doing wonders with the help of machine learning. With the Studykit system in place and having the ability to record raw metrics, we believe Philips now has better opportunities for developing sensor technologies and algorithms with the help of machine learning.

# 10. Project Retrospective

This chapter concludes the report by providing a self-reflection on the project from the author's perspective.

## 10.1 Reflection

The past ten months working at Philips has been an absolute delight for me. I never thought I would be working at a project that revolves around medical domain for my EngD thesis. I was incredibly new to this domain and initially found it hard to cope with all the new technical jargon and terms. The team really helped me get acquainted with the domain and guided me through the entire process.

I developed and learnt a lot over the past ten months both personally and professionally. Looking back to January when I started this project and seeing what I am today, there is no doubt I personally see a big difference in myself. When you are studying at the University, you just study about these new emerging technologies and get a gist of the idea, but never really implement anything at scale. I learnt a great deal about agile, microservices, production level code, continuous integration, and deployment, and many more things including working in a team driven by the same ambition.

To conclude, I will always remember this project and cherish it. This project has challenged me in every way possible and broaden my skills. The project has made me confident about learning and adapting to new technologies and situations irrespective of the domain. The project also taught me how to deal with problems at a larger scale and what is the thought process behind the rationale for different choices.

# Glossary

| | |
|---|---|
| ACC | Acceleration |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| BLOB | Binary Large Object |
| CDR | Clinical Data Repository |
| CI/CD | Continuous Integration / Continuous Deployment |
| CRUD | Create, Read, Update and Delete |
| DAG | Directed Acyclic Graph |
| DB | Database |
| dDLS | digital Design Language System |
| DOM | Document Object Model |
| ECMA | European Computer Manufactures Association |
| EngD | Engineering Doctorate |
| FHIR | Fast Healthcare Interoperability Resources |
| FR | Functional Requirements |
| GR | General Requirements |
| HSDP | Health Suite Digital Platform |
| HTML | Hyper Text Markup Language |
| IAM | Identity Access Management |
| IP&S | Intellectual Property & Standards |
| JS | JavaScript |
| JSON | JavaScript Object Notation |
| JSX | JavaScript eXtensible markup language |
| MSX | Internal name for Linux based communications hub |
| MVP | Minimum Viable Product |
| NFR | Non-Functional Requirements |
| PASS | Philips Actigraphy Server System |
| PDL | Philips Data Logger |
| PHB | Philips Health Band |
| PHP | Hypertext Preprocessor |

| | |
|---|---|
| PI | Program Increment |
| PPG | Photoplethysmography |
| REST | Representational State Transfer |
| S3 | Simple Storage Service |
| SAFe | Scaled Agile Framework |
| SPA | Single-Page Application |
| SQL | Structured Query Language |
| SWR | Stale-While-Revalidating |
| TS | TypeScript |
| TU/e | Technische Universiteit Eindhoven / Eindhoven University of Technology |
| UML | Unified Modelling Language |
| USB | Universal Serial Bus |
| WeST | Wearable Sensing Technologies |

# References

[1] Wikipedia, "Clinical Research," [Online]. Available: https://en.wikipedia.org/wiki/Clinical_research.

[2] P. Plan, "MoSCow Prioritization," [Online]. Available: https://www.productplan.com/glossary/moscow-prioritization/.

[3] P. B. Kruchten, "The 4+1 View Model of architecture," *IEEE Software,* vol. 12, no. 6, pp. 42-50, 11 1995.

[4] D. Project, "The web framework for perfectionists with deadlines | Django," [Online]. Available: https://www.djangoproject.com/.

[5] A. Airflow, "DAGs - Airflow Documentation," [Online]. Available: https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html.

[6] Cronhub, "Cron expression generator by Cronhub," [Online]. Available: https://crontab.cronhub.io/.

[7] Synopsys, "Black Duck," [Online]. Available: https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html.

[8] SonarQube, "SonarQube Documentation," [Online]. Available: https://docs.sonarqube.org/latest/.

[9] SAFe, "SAFe," [Online]. Available: https://www.scaledagileframework.com.

# Appendix A.

In this section, the main participants concerned with this project are presented in three categories: Philips, TU/e, and Hospital. The following sections describe the name, role, and tasks of each participant.

**Philips**

*Table 16 - Stakeholder list - Philips*

| Name | Role | Tasks |
|---|---|---|
| **Paul Dillen** | Philips supervisor and Algorithms architect | • Ensure the project success adds value to the Ambulatory and Virtual Care value stream<br>• Review the final project report and presentation<br>• Guiding the EngD trainee |
| **Martijn van Welie** | Cloud architect | • Guiding the EngD trainee with the relevant cloud and technical knowledge |
| **Dave Boshoven** | Embedded architect and SCRUM master | • Guiding the EngD trainee with the relevant embedded and technical knowledge<br>• Guiding the EngD trainee with respect to relevant tasks |
| **Carlijn Vernooij** | Product Owner | • Guiding the EngD trainee with appropriate medical domain knowledge<br>• Guiding the EngD trainee about specific requirements<br>• Review the final project report |
| **Reinder Haakma** | Product Manager (former Product Owner of Studykit) | • Ensure the project success adds value to the Ambulatory and Virtual Care value stream |

| | | |
|---|---|---|
| | | • Guiding the EngD trainee with appropriate medical domain knowledge<br>• Guiding the EngD trainee about specific requirements |
| **Bram Hoendervangers** | Engineer | • Successfully delivering the components of the project<br>• Guiding the EngD trainee with technical knowledge |
| **Gertjan Maas** | Engineer | |
| **Hugo Barrote** | Contact person at Intellectual Property & Standards department | • Ensure the project success adds value to the business.<br>• Providing relevant information to make the problem statement clear. |
| **Jimmy Huang** | | |

**Hospital**

*Table 17 - Stakeholder list - Hospital*

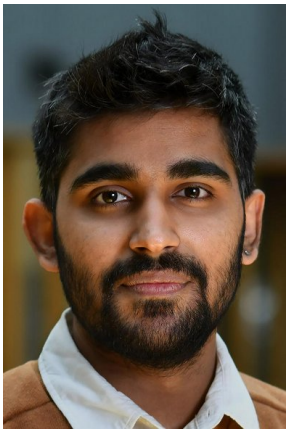
| Name | Role | Tasks |
|---|---|---|
| **Mayra Goevaerts** | Philips' contact person at Maxima Medisch Centrum | • Discuss merits and demerits of PASS system<br>• Provide feedback on Studykit Design |
| **Arthur Bouwman** | Philips' contact person at Catharina Ziekenhuis | • Provide feedback on Studykit Design |
| **Leon Montenij** | | |

**TU/e**

*Table 18 - Stakeholder list - TU/e*

| Name | Role | Tasks |
|---|---|---|
| **Yanja Dajsuren** | TU/e supervisor and EngD ST program director | • Guiding the EngD trainee<br>• Evaluating the project based on TU/e standards<br>• Review the final project report and presentation |
| **Renata Medeiros de Carvalho** | TU/e supervisor | |

# About the Author



**Akash Arora** received his Bachelor's degree in Computer Application from Guru Gobind Singh Indraprastha University, India in 2016. Later he received a Master's degree in ComputerApplications, specializing in Software Development from Symbiosis International University, India in 2019. Akash has also studied Cyber Security at Ritsumeikan University in Japan during the Master's program for the summer of 2018 and the entire research internship was sponsored by the Japanese government. Akash has worked actively during this Bachelor's and Master's program as a freelancer for start-ups as well as multi-national companies, helping them with their tech stack. Nowadays, his interests lie in Software Architecture & Design, Entrepreneurship, and Cyber Security.

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

**EngD SOFTWARE TECHNOLOGY**