UiT The Arctic University of Norway

DEPARTMENT OF COMPUTER SCIENCE

Secure collection of physical activity data from study participants through mobile solutions

Markus Madsen Fenes

Master's thesis in Computer Science INF-3990 June 2023

Abstract

Biometric and health related data collected through automated features in smartphones and wearables can provide researchers with valuable information about the population.

This thesis explores the possibilities of extracting data from Apple and Samsung wearable devices like watches and smartphones where native built-in support for data extraction and third-party data storage is not available.

Doing so will rely on a separate application (mSpider) on the connected phone, to perform data extraction, while prioritizes privacy and security as the primary concern. The thesis has implemented a proof-of-concept application that fetches existing data from Apple Health. The proof-of-concept application has a low footprint on the participant’s phone when it comes to battery usage and data usage.

The result from this thesis shows that it is possible and feasible to perform automatic and continuous data transfer from smart watches and phones in a larger population while respecting their privacy and security. Due to varying international laws, further research is needed to comply with regulatory requirements across the globe.

In conclusion, this application and the thesis presents a path for collecting research data from smart watches and phones.

Preface

I decided to do a master thesis when the world went into lockdown during the first phase of the coronavirus pandemic. At the time I was finishing my bachelor’s degree at University of Southeast Norway and did not see the pandemic as a good opportunity to get a job, so I decided to look for other opportunities and one of them was this master thesis at UIT.

Firstly, I would like to express my gratitude to my chemistry teacher for his remark, stating that I had an 80% chance of failing the course. To him I would say thank you for igniting a fierce determination within me. Through this challenge, I have attained a bachelor's degree in information technology and information systems, with the potential to achieve a master's degree if all goes according to plan. Looking ahead, I may even consider pursuing a PhD, as my passion for research, learning and the desire to educate others drives me forward in life.

While studying for my bachelor and this master, I have learned a lot of useful things both on and off the computer, that will help me later in life. I have become a very active bicyclist and have clocked in over 4000 kilometers on my bike, and I will continue biking after finishing my master thesis. While I have enjoyed studying, I am now looking forward to gaining some practical job experience and going on new adventures with my bike and in the realm of technology.

I would also like to thank friends and family for helping me move around the country and for motivating me and reminding me why I am doing this in the first place, namely the reasons explained above.

Another group of people I would like to thank is all the wonderful individuals I have met online during my master thesis, you have been an inspiration to me to find new fields of interest and just in general nice to talk to.

Additionally, I like to thank my supervisors, my fellow students, and other teachers at UIT.

And finally, I would like to thank you the reader, for reading this.

Table of contents

Abstract ii

Preface iii

Abbreviations vii

1 Introduction 1

1.1 Background 1

1.2 Problem statement 2

1.3 Limitations 3

1.4 Thesis structure 4

2 Theoretical Framework 5

2.1 Physical Activity 5

2.2 Apple iPhone 5

2.3 Apple Watch 6

2.4 Apple Health 6

2.5 Apple ResearchKit 8

2.6 Samsung Health 8

2.7 Security 9

2.8 Related work 10

2.8.1 mSpider 10

2.9 State of the art 11

2.9.1 Methods 11

2.9.2 Results 12

3 Method 18

3.1 Software and tools 18

3.2 Reach Native 20

3.3 Testing 21

4 Requirement specifications 22

4.1 Functional requirements 22

4.2 Non-functional requirements 26

4.3 Actors 26
4.4 Use-cases 27
4.5 Personas and usage scenarios 30
5 Design 32
6 Implementation 36
7 Results 45
7.1 Data collected and battery usage 45
7.2 User testing 48
7.3 Security 48
7.3.1 Apple 48
7.3.2 Samsung and Google 49
7.4 Limitations with using React Native. 50
8 Discussion 51
8.1 Thesis summary 51
8.2 Data collection 51
8.3 Privacy and security 51
8.4 User testing 52
8.5 Strengths and limitations 52
8.6 Problem statements 53
8.7 Contributions 54
8.8 Future work 55
9 Conclusion 56
References 57

List of figures

Figure 1 Apple Watch series 8 on the author's hand 6
Figure 2 Prisma diagram 13
Figure 3 Use case diagram. 29
Figure 4 Persona 1 Mike Smith 30
Figure 5 Persona 2 Robert 31
Figure 6 iOS 32
Figure 7 Android 32
Figure 8 Database diagram 34
Figure 9 Sequence diagram of token use 35
Figure 10 Permissions request on iOS 40
Figure 11 Code snippet from upload loop 41
Figure 12 Example data upload 41
Figure 13 Uploading from iPhone and Apple Watch 43
Figure 14 User interface in iOS 44
Figure 15 60 days with records from the mSpider application 45
Figure 16 24-hour overview of scopes on 19 of June 46
Figure 17 10-day view on scope 46

List of tables

Table 1 Summary of included papers 14
Table 2 Functional Requirements 22
Table 3 Use cases. 27
Table 4 HealthKit object names 39
Table 5 Count of records over 60 days 47
Table 6 Estimated data size in bytes 47

Abbreviations

API	Application Programming Interface
BPM	Beats Per Minute
ECG	Electrocardiography
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
mSpider	Motivating continuous Sharing of Physical activity using non-Intrusive Data Extraction methods Retro- and prospectively.
HIPAA	Health Insurance Portability and Accountability Act
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JWT	JSON Web Token
ORM	Object Relational Mapping
OS	Operating System
SDK	Software Development Kit
TOS	Terms Of Service
XML	Extensible Markup Language

1 Introduction

1.1 Background

Collecting health data from consumer-based wearables can be an avenue for researchers to gather valuable information on what their participants do without too much of an inconvenience for the participants. The ability to get the health data delivered in a way that both helps the researchers and does not interfere with the privacy or security of the participant. Mobile crowdsensing [1] is the name of the data collection method, and there are two main methods of crowdsensing, participatory sensing [2] and opportunistic sensing [3].

Participatory sensing is when the participants are aware and have agreed to the data collection and opportunistic sensing is when data is collected without the implicit approval or knowledge from the data originator.

The Apple Watch (Apple Inc, California (CA), US) and Samsung Galaxy Watch (Samsung Electronics Co, South Korea) were selected to the be devices that is focused on. The main reason for selecting the Apple Watch and Samsung Galaxy Watch is that these are the most popular on the market and has the most users both in Norway and worldwide. Both watches have been on the market for a few iterations and have gotten most of their flaws sorted out. Market research done by Counterpoint Research states that Apple has 43 percent[1] of the smart watch market globally, and that Samsung has 8 percent of the smart watch market.

The second reason for selecting these two watches is that when it comes to integration with a data gathering platform, these watches with their accompanying phones do not offer an integrated way for third parties to collect the stored information using Application Programming Interface (API).

An API is often a webservice hosted by the manufacturers of products that allows developers and end users to interact with the product offered in a programmatic way. This allows third parties to develop software that will extend the functionalities of the original product offered by the first party.

[1] https://www.counterpointresearch.com/global-smartwatch-shipments-market-share/

Since neither Apple Health[2] nor Samsung Health (Samsung Electronics Co, South Korea) are providing a convenient way for accessing externally stored data using an API, a third-party developer must use a service running locally on the device in question itself to be able to access and utilize physical activity and other related health data. To do this in a programmatic way, a third-party developer must use a Software Development Kit (SDK).

SDKs are sets of programming libraries that are provided by the product manufacturer and are used to interface with the software running locally on an individual user's device. The SDKs allow third party developers to request permission from the user to change the phone's settings, read, and write data from and to applications and the phone itself.

Neither Apple nor Samsung offers an SDK capable of uploading data directly from the users' phone or watch. A third-party application is needed to read data using the SDK and then process and upload it to a service run by third-party developers.

1.2 Problem statement

In this project, the primary objective is to simplify the prosses of sharing data from smart watches and phones with researchers. The following problem statement will outline the issue and propose possible solutions.

How can health data be collected from smart watch vendors that do not support backend API data retrieval?

[2] https://www.apple.com/ios/health/

Sub-problems

SP1: *How can health data be automatically and continuously collected from wearables?*

Collecting activity data automatically is important since users cannot be expected to interact with the application daily. Continuously collecting data in the background also reduces the burden placed on the user while increasing the amount of data collected.

SP2: *How can data be collected while preserving the privacy and security of the participants?*

For the users of the application to trust that their data is not mismanaged, it is important to consider privacy and security when developing health and research applications, particularly for health data when this is potentially sensitive in nature.

SP3: *How can collected data be prevented from being sent or stored on vendor services?*

Preventing upload to the vendors services where possible makes sure that the shared data is only uploaded to specified locations.

1.3 Limitations

The initial goal of this master project was to develop a health data gathering application that would work on both Apple iPhone and Samsung Galaxy phones. However, due to time constraints, the scope was adjusted to only focus on making a solution that collects physical activity and health data from the Apple iPhone and the Apple Watch.

Another reason to only focus on Apple devices is that Samsung recently changed their system for collecting data from individual users' phones and the new system was still in development at the time of writing.

1.4 Thesis structure

Chapter 1

The introduction chapter explains the needed background and context. This chapter also covers the limitations and explains the problem statement.

Chapter 2

The second chapter introduces the theoretical framework and related work (both commercial and open source). The chapter additionally covers the literature review and the results from it.

Chapter 3

The third chapter explains the methods used for designing and implementing the application and when writing and researching for the thesis.

Chapter 4

The fourth chapter contains use cases and the functional and non-functional requirements of the application.

Chapter 5

The fifth chapter contains the design decisions during the development of the mobile application and web application.

Chapter 6

The sixth chapter explains the implementation of the mobile application.

Chapter 7

The seventh chapter shows the results and analysis of the data collection as well as a small breakdown of the collected data and other results from the application development.

Chapter 8

The eighth chapter discusses the solution and the results from the seventh chapter and suggests possible future work for the application.

Chapter 9

The ninth chapter concludes the findings of the report and adds final remarks.

2 Theoretical Framework

This chapter explains the concepts needed to understand the next chapters of the thesis.

2.1 Physical Activity

The World Health Organization (WHO) defines physical activity as

> *"Any bodily movement by skeletal muscles that requires energy expenditure" [4]*

Physical activity is when the user is not sedentary, when a person is physically active, they could be doing a set of training exercises, or they could just be doing their daily routine. There are various levels of physical activity, they range from sedentary behavior to standing and the three different intensities. Sedentary behavior [5] is any behavior consistent with having less than 1.5 metabolic equivalents (METs) [6] of energy expenditure.

- Light Physical Activity (LPA).
- Moderate Physical Activity (MPA).
- Vigorous Physical Activity (VPA).

The above list shows the different types of physical activity.

2.2 Apple iPhone

The Apple iPhone (Apple Inc, CA, US) is a line of both premium and budget phones from Apple. The first generation of iPhone was announced in 2007 [7], by the late Steve Jobs (Apple's CEO at the time). The iPhone was one of the first commercially available smartphones that made an appeal to the masses. The iPhone has become a household name in the consumer phone market, with a market share of over 15 %[3]. As of May 2023, the newest revision of the iPhone is the line of 14th generation iPhones.

Apple iOS (Apple Inc, CA, US) is the operating system (OS) that powers the iPhones. Launched at the same time as the iPhone in 2007, but under the longer name of iPhone OS, the iOS name was first used for the third version of the OS. The latest version of iOS is 16, and there is a new major version of iOS launched each year in the fall.

[3] https://www.statista.com/statistics/299153/apple-smartphone-shipments-worldwide/

Figure 1 Apple Watch series 8 on the author's hand

2.3 Apple Watch

The Apple Watch (Apple Inc, CA, US) is made by Apple and has millions of users worldwide. An Apple Watch series 8 is shown in Figure 1 on the author's hand.

The Apple Watch has had 8 generations since the introduction to the market in 2015. The Apple Watch is a smart watch with a lot of the same features of the standard iPhone but in a smaller form factor. There are versions of Apple Watch that have cellular connection and allow regular phone calls. The Apple Watch also features multiple sensors and other health related features along with the usual clock features. The Apple Watch runs watchOS (Apple Inc, CA, US) as the operating system.

The Apple Watch collects a plethora of information from the user. All health-related information is recorded and stored in Apple Health (Apple Inc, CA, US). Heart rate monitoring and activity tracking, and sleep tracking are valuable information sources for researchers as they can provide information about the current health of the population.

2.4 Apple Health

Apple Health is a standard application where Apple stores health data from both the iPhone and the Apple Watch. Apple Health is installed on every iPhone sold today, and it is enabled by default. Apple Health contains both sensitive and less sensitive data and therefore the information stored is encrypted at rest. While the iPhone is unlocked, third party applications can read and write to the Apple Health database, using Apple HealthKit (Apple Inc, CA, US). Third-party access is restricted to what the user allows the applications to read and write.

Health records stored in Apple Health

Health data records stored in Apple Health have standard properties like when the data were recorded, the type of activity, and when it was added to Apple Health.

Steps are recorded both by the iPhone itself and the Apple Watch. These records are combined by Apple Health to avoid duplication when stored. Steps also store the start and stop time of the activity.

Beats Per Minute[4] (BPM) are stored in Apple Health, but the records are divided into subcategories, BPM, average BPM when walking, resting BPM, and BPM variation. The Heart rate records are saved at regular intervals. The intervals change depending on if the user is physically active or if they have activated a workout mode. When none of the conditions are present, the Apple Watch will store recorded BPM at an average of once per 10 minutes.

Active Energy Burned and Resting Energy are two records where one builds on the other. Active energy is what the body burns on top of the resting energy when the body is not sedentary, while resting energy is what is being burned when the body is sedentary. Both active energy and resting energy are stored in kcal.

Both devices (iPhone and Watch) record and store distance traveled in kilometers. The devices also store flights climbed. Cycling distance traveled is an additional record, but only stored by the Apple Watch and only when activated by the user or the user has been cycling for longer time and Apple Watch has detected it. These records have a start and stop time attached. Oxygen in the blood is another statistic that the Apple Watch stores at regular intervals, when the user is sedentary and has their hands laying down on either a table or another flat surface.

Sleep Records

Apple Health can collect data about the users' sleep if the user has enabled the sleep plan in the settings of Health. The sleep tracking feature is not enabled by default, but once enabled, the Apple Watch will collect sleep data about the user. The sleep records shows when the user is in the various stages of sleep and store these with a start and stop time.

---

[4] https://developer.apple.com/documentation/healthkit/hkquantitytypeidentifier/1615138-heartrate

The Apple Watch collects how long the user sleeps, how much time the user spends in bed before going to sleep and the heart rate while in bed and sleeping.

Apple Fitness

Apple Fitness (Apple Inc, CA, US) is an application developed by Apple to help iOS and watchOS users keep track of their exercise sessions. Apple has a paid subscription program (Apple Fitness+) that users can subscribe to for additional guides and videos on several types of exercises. Apple Fitness also connects to Apple Health and allows developers to read data from the exercise sessions. When an exercise session is active the Apple Watch will record heart rate and other metrics from the user continuously. If the user has enabled the GPS feature in the privacy settings of their iPhone, Apple Fitness will store the user's workout routes[5] and tracks, so that the users can look at it later or review it after they are finished with the session. The sessions can also be shared with friends and family.

2.5 Apple ResearchKit

Apple ResearchKit[6] (Apple Inc, CA, US) is an open-source system, developed by Apple for their products to use in research projects. ResearchKit offers ways of gaining consent from the participants. Another feature is the ability to make surveys, present them to the user, and collect the answers from the surveys. ResearchKit can be implemented with HealthKit to allow for interaction with between them and allow health data from HealthKit to be used.

2.6 Samsung Health

Samsung Health is the equivalent of Apple Health but for Samsung phones and watches. This application is also installed on every Samsung Galaxy phone sold today. This application is the hub for storing physical activity records and other health related data collected from the Galaxy Watch and the Galaxy suite of phones.

Samsung[7] has a new program for third-party developers called Privileged Access. Developers can use this program to request access to the Samsung Health Privileged Access SDK, and that gives the developers access to physical activity data, and other health related data from

[5] https://support.apple.com/en-us/HT210385

[6] https://www.apple.com/lae/researchkit/

[7] https://developer.samsung.com/health/privileged

Samsung Health. This SDK is compatible with the fourth generation of Samsung Galaxy Watches running on Wear OS (Google, CA, US).

2.7 Security

The world is a connected place, our lives are lived both online and offline. With the recent surge in usage of smart watches. Making applications that utilize them as secure as possible from the ground up is important.

Security is more involved than respecting the privacy of the users of any given service provided. It is making sure data about the user is stored securely. It is also about how the collected data is transferred between the end user's device and servers owned by device manufacturers and service providers, who have access to that data. There are laws and regulations service providers must follow to be able to handle user and patient data.

Health Insurance Portability and Accountability Act (HIPAA) and General Data Protection Regulation (GDPR) [8] are two sets of regulations that protects the privacy of individuals living in the respective countries.

GDPR is a European Union (EU) regulation that became effective in 2018, and the purpose of it is to protect the users of digital mediums in Europe. GDPR is also effective in Norway because of the European Economic Area (EEA). The EEA grants Norway access to the single market of EU but also requires that Norway follow relevant regulation in return. The GDPR regulation states that any service provider outside of EU that renders services to EU and EEA citizens must follow the regulations set by it and that the data that European citizens generate is stored in the EU or in a jurisdiction with similar regulations. The right to be forgotten, the right to have data amended in case of error and the right to see the stored data stored are the main rights GDPR grants EU and EEA citizens.

HIPAA is a US federal law that prohibits health care personnel from disclosing information to non-authorized parties than allowed by the patient themselves or their next of kind, it also has other sections but the one in focus here is the handling of personal medical records and information.

2.8 Related work

WeFitter Connections (WeFitter, Netherlands)[8] is a commercial offering that aims to be one API, that developers can use to connect to magnitude of smart devices. They offer support for the Apple Watch and Google Fit (Google LLC, CA, US), Samsung Health (Samsung Electronics, Seoul, South Korea) and Garmin.

WeFitter also offers a gamification API, that will motivate users to exercise more and use the system more frequently. Gamification is trying to use the mechanics of video games to influence the users' choices.

Terra API (Terra, London, UK) offers an API[9] akin WeFitter's API, in the regard that it offers a way for developers to integrate with the other service providers services instead of doing all the prerequisites themselves, this API will do the interfacing to the third-party services (Apple, Samsung and the like) but at a cost since they are a commercial service. Terra is funded in part with venture capital from Samsung Next[10].

2.8.1 mSpider

mSpider [9] is a data collection platform, developed in-house at UIT. The mSpider project is led by Henriksen (PhD). The mSpider platform is designed to be easy to expand when a new product is launched, or when a new project has requirements that are not yet supported by platform. The mSpider platform is also designed to support future master students and PhD candidates that want to do a variation of health data collection. There are two versions of mSpider. The first version had support for the most common smart watches and devices. The second version has been in development at UIT since the end of 2022, this version is designed to better support expandability and scalability. The mSpider mobile application is developed for the newest version. The first version of mSpider also had a mobile application, but this version was never used or published, only tested internally.

mSpider (Motivating continuous Sharing of Physical activity using non-Intrusive Data Extraction methods Retro- and prospectively).

---

[8] https://www.wefitter.com/en-us/features/connections/

[9] https://tryterra.co/products/api

[10] https://www.samsungnext.com/blog/why-we-invested-in-terra-an-api-for-fitness-and-health

Other related work

CardinalKit (Stanford Byers Center for Biodesign, CA, US) [11] is an iOS application and a webservice with Firebase (Google, CA US) database. CardinalKit offers template applications and a framework, that are programmed using Swift (Apple, CA, US) instead of using a framework like React Native (Meta Inc, CA, US), as has been done with the mSpider smart phone application.

2.9 State of the art

The literature review was undertaken in October and November 2022. The literature review started on PubMed before branching out to the other search engines. Institute of Electrical and Electronics Engineers (IEEE) and Association of Computer Machinery (ACM) were the other databases where the search also was undertaken.

After finding papers on all three databases, the next phase of the screening started. This phase involves reading the titles and selecting the ones that should be reviewed further. Once all the results had been reviewed. The next phase started, and this phase involved reading the individual abstracts for each of the remaining articles.

After all the abstracts had been reviewed, the ones were still relevant were imported into the literature reference manager EndNote for further evaluation. In EndNote, the full text versions of the article were downloaded. The remaining relevant papers were read and included in the review further down.

2.9.1 Methods

Databases

The following databases were used when performing the literature review:

- IEEE (23), selected (1)
- PubMed (151), selected (9)
- ACM (459), selected (4)

[11] https://cardinalkit.sites.stanford.edu/

Inclusion criteria and exclusion criteria

The first inclusion criterion is that the article must written and available in English to be included in the review. The second criterion is that the article must be accessible and retrievable using either open-access, or access through the university library at UIT. The third criterion is that the article must mention how the authors extracted the data from the participants.

(“Apple Watch” OR “Samsung Galaxy Watch”) AND (“physical activity” OR “steps” OR “kcal” OR "energy expenditure" OR “ecg” OR “heart rate”)

The research query used to provide the results displayed further down is divided into two categories.

The first block is the products and their manufacturers.

Apple Watch, Samsung Galaxy Watch

The secund block is the type of activity and measurements of heart rate and energy expenditure.

physical activity, steps, kcal, energy expenditure, ECG, heart rate.

2.9.2 Results

The Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) **[10]** is a guideline that will help literature reviewers explain what they have done in a transparent and open way.

The initial search resulted in 633 articles. After removing duplicates, 631 remained. After screening 631 records, 599 were removed because of the title or the abstract was not interesting enough to be included further. 33 papers were left after abstract and title screening and in the end 14 papers were included in the review.

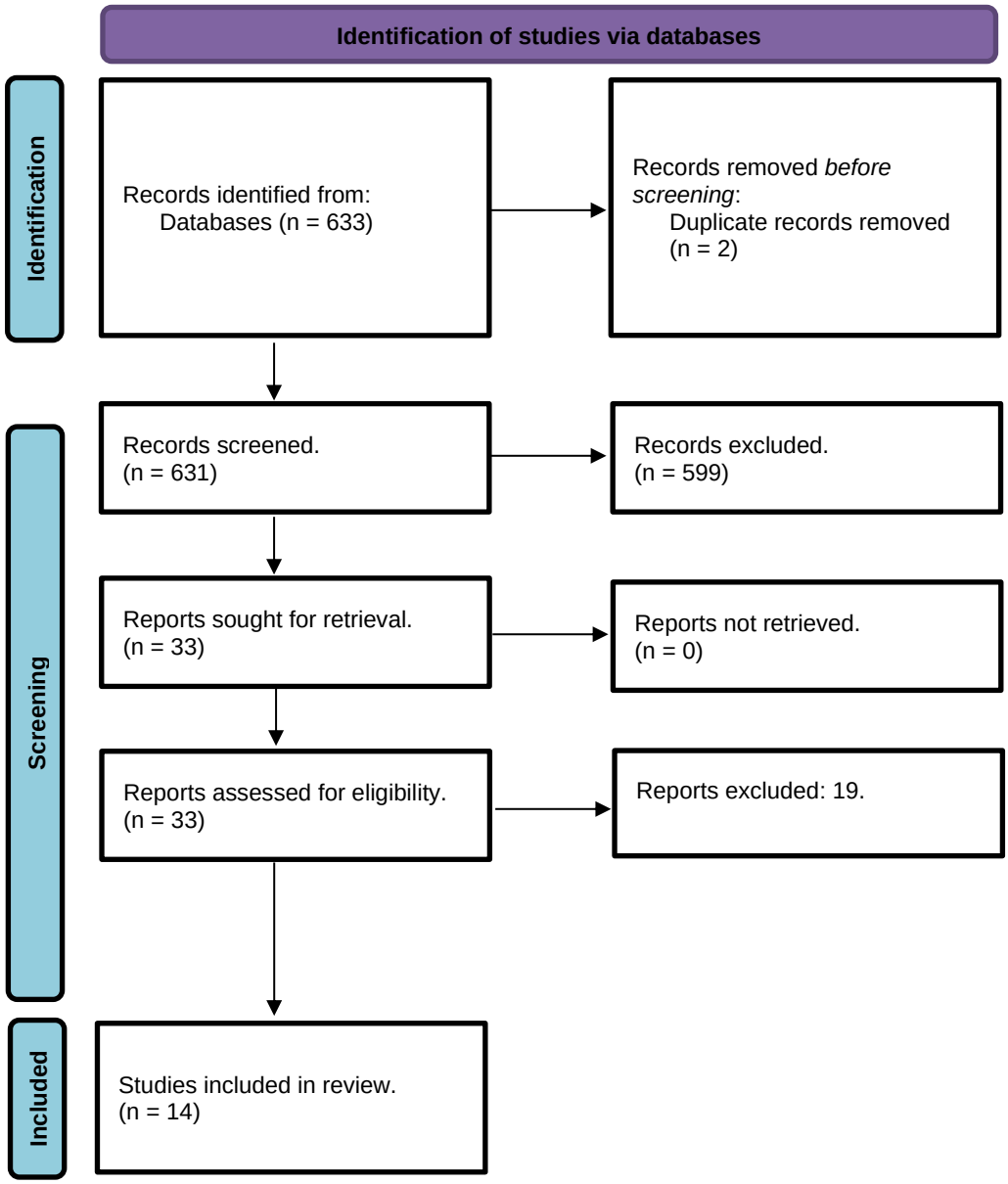

Figure 2 Prisma diagram

Article summary

Table 1 Summary of included papers

Author	Data Extraction method	Main results
Bartschke, et al. [11]	FHIR	The authors used an application to parse the raw data from the Apple Watch and then convert it to a FHIR observation and from there send it to a server.
Curtis, et al. [12]	All types of sensors, GPS, call log, text log,	Full scale system that uses Amazon Web Services for scalability and MongoDB as the database. The researchers can input the specifics about their study and then the system will adapt to fit it. The need for software engineering and coding skills is minimal.
Dandapani, et al. [13]	Apple HealthKit and ResearchKit with their own application in real time.	A system for conducting research on human falls, and other health related data collected by the Apple Watch. Their system is explained in detail. The Redcap system (with its API) is also explained and lastly, they explain the shortcomings of the technology.

Fuller, et al. [14]	Apple HealthKit with their own application.	Authors made their own unpublished application.
Goldberg and Ho [15]	Apple HealthKit and ResearchKit.	An application that facilitates the collection of heart rate data through Apple HealthKit. They have also made a model to detect exercise during free living conditions. The authors tried to use React Native to manage one code base, but after further research, they found out that it was not equipped to process data extraction from smartwatches.
Hänsel, et al. [16]	Data is stored on the watch itself and then transmitted to the iPhone for later use.	A bachelor project with an application for both the iPhone and the Apple Watch, with this, they can read the raw heart rate data.
Kunchay and Abdullah [17]	Aware framework and Apple Health using Apple HealthKit.	The iPhone and Apple Watch are used to collect data about substance abuse. The Apple Watch asks the users small questions with a system called micro interruptions. Further it outlined what happens when the devices are offline. The system also collected location using GPS

		and health data from Apple Health.
Lima, et al. [18]	No Data collection.	The authors discuss how users of smart watches use them, and how the market has grown over the years.
Marinescu [19]	Apple HealthKit with self-developed app.	iOS application called SOS Heart. SOS Heart collects real time data from the Apple Watch and detects abnormalities. SOS Heart will then alert the user's doctor. The doctor will then assess if the patient needs emergency care. SOS Heart provides doctors with valuable information that can save time.
McManus, et al. [20]	Apple ResearchKit and their own app called eFHS.	Authors made their own application. This is part of a larger study called Framingham Heart Study. Participants had a weekly blood pressure measurement and daily data from Apple Watch. The study ran on a 3-month cycle, and then the participants were called back in for follow-up sessions.
Ding, et al. [21]	Apple Health kit and mydatahelps.	Updated version of the Framingham Heart Study, using mydatahelps to simplify management.

Shcherbina, et al. [22]	Manual data extraction using XML from Apple Watch and Apple Health.	Larger study of many major smart watch brands, but no automatic data collection.
Turki, et al. [23]	Application called FITIV, and then downloaded to the computer by the researcher manually.	The accuracy of the Apple Watch is discussed. The article explains how they extracted the data to analyze it.
Walch, et al. [24]	Raw data collection from Apple HealthKit and Apple Watch.	Authors developed their own application with raw XYZ data from the accelerometer of the Apple Watch. The collected data was used to conduct sleep studies. The application also collected heart rate and then they ran the collected data through different algorithms.

FHIR: Fast Healthcare Interoperability Resources, API: Application Programing Interface, XML: Extensible Markup Language, GPS: Global Positioning System.

The system described in [12] is the most interesting system found in the literature review, but this system also collects information about call logs and location information, which could present challenges when it comes to privacy.

The literature review shows that there has been research done in this field before, but it also shows that most of the research done before uses semi-automated systems.

Chapter summary

This chapter explained the theorical framework used for the thesis, the mSpider system the application is made for and covered the literature found and the other related work.

3 Method

3.1 Software and tools

Prior to starting development, both *Google* (Google LLC, CA, US) and *GitHub* (GitHub Inc, CA, US) were used heavily to find what software that could aid the mSpider mobile application.

Git[12] (Git, Open-source projects) is a version control software made by Linus Torvalds to manage the Linux kernel. In the mSpider project, Git has been used to keep track of the changes made by the author.

GitHub[13] is a code sharing platform. GitHub was used to find third-party packages that could help the mSpider application reach the goals set for it.

GitLab[14] (GitLab Inc, CA, US) was used to store the git repository for the source code and to run and compile the code using a GitLab runner. A GitLab Runner is a continuous integration runner that will look for changes in the git repository and when there are changes, it will fetch the changes and build them without the need for manual intervention. The GitLab runner ran on a Mac mini (Apple Inc, CA, US) and uploaded the compiled software package to GitLab so it could be downloaded and installed on the mobile device.

Visual Studio Code[15] (Microsoft, WA, US) was used to edit the source files for the project and interact with the server running the application and database.

Xcode[16] (Apple Inc, CA, US) is an integrated development environment (IDE) for Apple products. Xcode was used to compile and test the mSpider iOS application.

Docker[17] (Docker Inc, CA, US) was used when the application was containerized. The application, server and database ran in Docker during the project's development phase.

---

[12] https://git-scm.com/

[13] https://github.com/

[14] https://about.gitlab.com/

[15] https://code.visualstudio.com/

[16] https://developer.apple.com/xcode/

[17] https://www.docker.com/

PostgreSQL[18] (PostgreSQL Global Development Group, CA, US) is a database server that is used to store uploaded records. PostgreSQL is an open-source relational database built on Structured Query Language (SQL) and uses SQL combined with other features.

Go[19] (Google Inc, CA, US) is a programing language used for the mSpider web API, made by The Go Authors in 2009 at Google.

Gorm[20] is a database Object–relational mapping (ORM) for Go. ORM 's are software packages that help developers use databases more efficiently as the package provides a standard way for interacting with one or more databases. Gorm was used to program the database schema for PostgreSQL database.

NodeJS[21] (OpenJS Foundation, CA, US) is a web server that runs on the JavaScript runtime. NodeJS makes developers able to run JavaScript outside the web browser and in a server environment.

Yarn[22] (Meta Open Source, CA, US) is a package manager that lets developers install dependencies and libraries to their NodeJS server environments. Yarn was used to add libraries, start, and stop the development environment of the mSpider React Native application.

Axios[23] (Matt Zabriskie, et al) is a NodeJS library that handles HTTP requests. Axios supports all the HTTP request methods. The request methods that are primarily used for mSpider mobile application are POST and GET. Axios was used to handle the web requests that are made from the mSpider application.

[18] https://www.postgresql.org/about/

[19] https://go.dev/

[20] https://gorm.io/

[21] https://nodejs.org/en

[22] https://yarnpkg.com/

[23] https://axios-http.com/

Grafana[24] (Grafana Labs, NY, US) is a web tool for making graphs and metrics. Grafana was used to make some of the figures that are in this report. Grafana was also used to query the data that the author collected during the testing period.

Swift (Apple Inc, CA, US) is a programing language that powers Apple devices. The language has been in development for 10 years and it is planned to be the replacement of Objective-C. Objective-C is the former language that Apple used on their devices. Swift is derived from Objective C, but Swift is modern and safe by design.

3.2 Reach Native

React[25] (Meta Open Source, CA, US) was chosen to be the framework that powers the application. React is an open-source JavaScript framework made by Facebook (Meta Inc, CA, US), that makes it easier to make single page applications. The framework is client side, but it can also be used with server-side rendering.

React Native [26] (Meta Open Source, CA, US) is an open-source framework made by Facebook which makes it easier to develop applications for both Android and iOS. React Natives cross platform features development easier by using the same code base, written in either JavaScript or TypeScript (Microsoft, WA, US) and embedding a website in the application.

React Native is the backbone of the mSpider iOS application, it handles everything from the Graphical User Interface (GUI) to the background data fetching from Apple HealthKit.

React Native HealthKit

React Native HealthKit[27] (Kingstinct AB, Sweden) was found using GitHub. React Native HealthKit is an open source React Native package that lets third party developers access data from Apple HealthKit using TypeScript Swift bindings.

[24] https://grafana.com/

[25] https://react.dev/

[26] https://reactnative.dev/

[27] https://github.com/Kingstinct/react-native-healthkit

Using Swift bindings, React Native HealthKit can interact with Apple HealthKit in a similar fashion that a developer using Swift directly can. The documentation[28] made available from Apple at their developer sites is applicable to what a developer can utilize React Native HealthKit for.

Hardware

The iOS version of the mSpider application was developed and tested on an iPhone XR (with iOS 16.4a) with an Apple Watch series 8 connected to it. A Mac mini M1(2020) with Xcode (version 14.3) was used to compile and install the application on the iPhone, the iOS version was targeted at iOS 16 as that is what the iPhone runs. An older Apple Watch series 2 was also used to test the application with another iPhone that ran iOS 16.4. During development multiple new versions of both iOS and Mac OS were released, these were installed as they became available, and the application was tested with them.

3.3 Testing

The mSpider iOS application was tested with an Apple Watch series 8 that the author wore on their left arm for the duration of three months (start of April to the end of June). On 9th of May, the development of the mSpider iOS application was paused, and the focus shifted to testing the application instead. The Apple Watch was connected to an iPhone XR running iOS 16, this iPhone was not connected to the cellular network, and had flight mode activated, meaning that uploads only happen on wireless networks at the UIT campus or at the author's apartment.

From 12th of May and until 21st of May, the sleep tracking capability of the Apple Watch was tested. During this 10-day period, the author wore the watch continuously throughout most days and nights, except for when the Apple Watch was charging, or the author was showering. The author has also tested the hand washing detection capability of the Apple Watch.

[28] https://developer.apple.com/documentation/healthkit

4 Requirement specifications

The Volere Requirements [25] template was used to help create functional requirements shown in
Table 2. The Volere requirements template is used in software engineering to help developers understand their projects deeper. The number in the table states the identifier for the requirement and will be used in dependencies and when referred to in the rest of the report.

The Unified Modeling Language (UML) [26] has been used to create the sequence diagram and the use-case diagram. UML is a standard from the International Organization for Standardization for creating diagrams in software engineering.

Functional requirements and non-functional requirements are distinct types of requirements. both are important for the functionality of the application. Functional requirements refer to the functionality of the application the participant interacts with, while non-functional requirements refer to the functionality that the participant does not interact with directly.

4.1 Functional requirements

Table 2 Functional Requirements

#	**Requirement**	**Rationale**	**Fit criteria**	**Priority**	**Dependencies**
#1	The application must start when opened by the participant.	For the participant to be able to utilize the functions of the application, it must start.	The participant can reach the login page of the application.	High	
#2	The participant must be able to sign in with a one-time unique invite code.	To maintain the privacy of the participant, while also identifying which study the health data should get stored to.	The participant is enrolled into the correct research project when signing in.	High	#1

#3	The participant must be asked what data they wish to share with the researcher(s).	The participant's privacy should be maintained, they should decide what is shared with the researcher(s).	The collected records shall only contain health data collected with the consent of the participant.	High	# 1, # 2
#4	Consensual retrieval of health data from Apple Health using Apple HealthKit.	Health data collection from Apple Health will help researchers understand the population whom the data was collected from.	The application only uploads shared health data.	High	
#5	Health data must be sent securely to the mSpider Apple provider.	Health regulations and privacy laws require that health data is encrypted in transit.	Data uploads shall happen over an encrypted connection to the mSpider server.	High	# 4
#6	The application must be able to run in the background after initial setup.	For it to be useful for a longer period, background upload is necessary.	The application can receive data from Apple Health and upload it is in in the background.	High	

#7	The application must be able to continue after the participant has restarted their phone.	Avoid unnecessary reliance on the participant.	The application continues to operate, requiring no action after restarting the phone.	Medium	# 2
#8	The application must function autonomously after initial setup by the participant.	Avoiding interference with the daily routine of the study participants to get data.	The application functions normally when in the different states (active, background).	Medium	# 2
#9	The application must use atomic uploads.	A record of the uploaded data will keep data consistent in the event of an upload failure.	All shared health data is uploaded to the mSpider server.	High	
#10	The application must alert the participant if an error occurs.	To give the participant the ability to fix smaller issues, should they arise.	The application alerts the participant if an error occurs.	Low	
#11	The application must upload health data at regular intervals.	To minimize risk of losing data, it should be uploaded regularly.	Health data received is uploaded to mSpider at regular intervals.	Medium	

#12	The application should check when it last uploaded to the server.	Only uploading changed data is more efficient use of resources.	The application can store the state of the upload using an anchor.	Low	
#13	The application should ask the participant if they want to upload historical data.	To be able to see past activity and compare it to newer data.	The participant is presented with an option for including historical data.	Low	
#14	The application must have an option to log out and stop the data collection.	The participant may wish to resign from the studies.	The participant can sign out of the application and have it stop uploading health data.	Low	#1 #2
#15	The application should only upload data on Wi-Fi.	To not add any data costs to the participant, the mSpider application should only upload on Wi-Fi networks.	The application pauses uploads until a Wi-Fi network is available.	Low	

4.2 Non-functional requirements

Application Security and privacy

The mSpider application must adhere to strict privacy regulations as it collects health data from Apple Health. Health data must never be exposed while in transit.

Performance and uptime

The mSpider application and server-side software must minimize downtime to a minimum.

Expandability and reusability

To allow for future research projects with different data type requirements to use the same application with minor changes. The mSpider iOS application and server-side provider must be easily expandable to add support for new Apple Health datatypes.

4.3 Actors

An actor is a term from UML that describes a user that has a role in a system they interact with. The actor can be a physical user, or it can be a system interacting with another system.

iPhone user

The iPhone user has health data from their iPhone and Apple Watch and wishes to share it with researchers to provide up to date information about their health and physical activity.

Samsung user

The Samsung user has health data from their Samsung Galaxy and Galaxy Watch and wishes to share it with researchers to provide up to date information about their health and physical activity.

Apple Watch

The Apple Watch is responsible for the collection of various health related metrics about the participant, including heart rate, steps taken, and sending this to Apple Health.

Samsung Galaxy Watch

The Galaxy watch is responsible for the collection of various health related metrics about the participant, including heart rate, steps taken, and sending this to Samsung Health.

Apple Health

Apple Health is responsible for saving health data from the iPhone and the Apple Watch and then making it available to third-party applications.

Samsung Health

Samsung Health is responsible for saving health data from Samsung Galaxy and the Galaxy watch and then making it available to third-party applications.

mSpider

The mSpider server is used to retrieve/receive the health data sent from the iOS application on the users iPhone.

mSpider application

The mSpider application is responsible for the transfer and upload of health data from Apple Health/Samsung Health to the mSpider server.

4.4 Use-cases

The term *use case* is from UML, and it is used to describe the interaction between either the user of a system or the system itself takes to achieve a goal.

Table 3 Use cases.

Use-case	**Goal**	**Actor and requirement**	**Flow**
Log in	A participant must be able login to the mSpider application.	iPhone user with an Apple Watch #1 and #2	The participant opens the mSpider application and logs in with the provided invite code.

Approve read permissions from Apple Health	A participant must be to approve the permissions request from Apple Health.	iPhone user with an Apple Watch. #3	The participant opens the mSpider application, the application will ask them to approve permissions from Apple Health.
Upload data.	A participant must be able to share health and physical activity data with researchers.	iPhone user with an Apple Watch #4, #5, #6, #7, #8 and #10, # 11	The participant lets mSpider run in the background on their phone and the application uploads data.
Log out.	A participant must be able log out of the mSpider application and stop the data collection.	iPhone user with an Apple Watch #14	The participant opens the mSpider application and finds the log out button, and press it, and the application will then log them out and stop uploading new health data from Apple Health.
View data.	An administrator must be able to view the collected data.	Admin with database or mSpider web portal access	Admin logs in to the database or front end of mSpider web portal and looks at the collected data.

The use case diagram is shown in Figure 3 shows that the participants can log in to the mSpider application, they can also log out of the system. Once logged in, the participants will get a screen where they will be asked to approve the required permissions that the study, they have enrolled in has set.

Use-case diagram

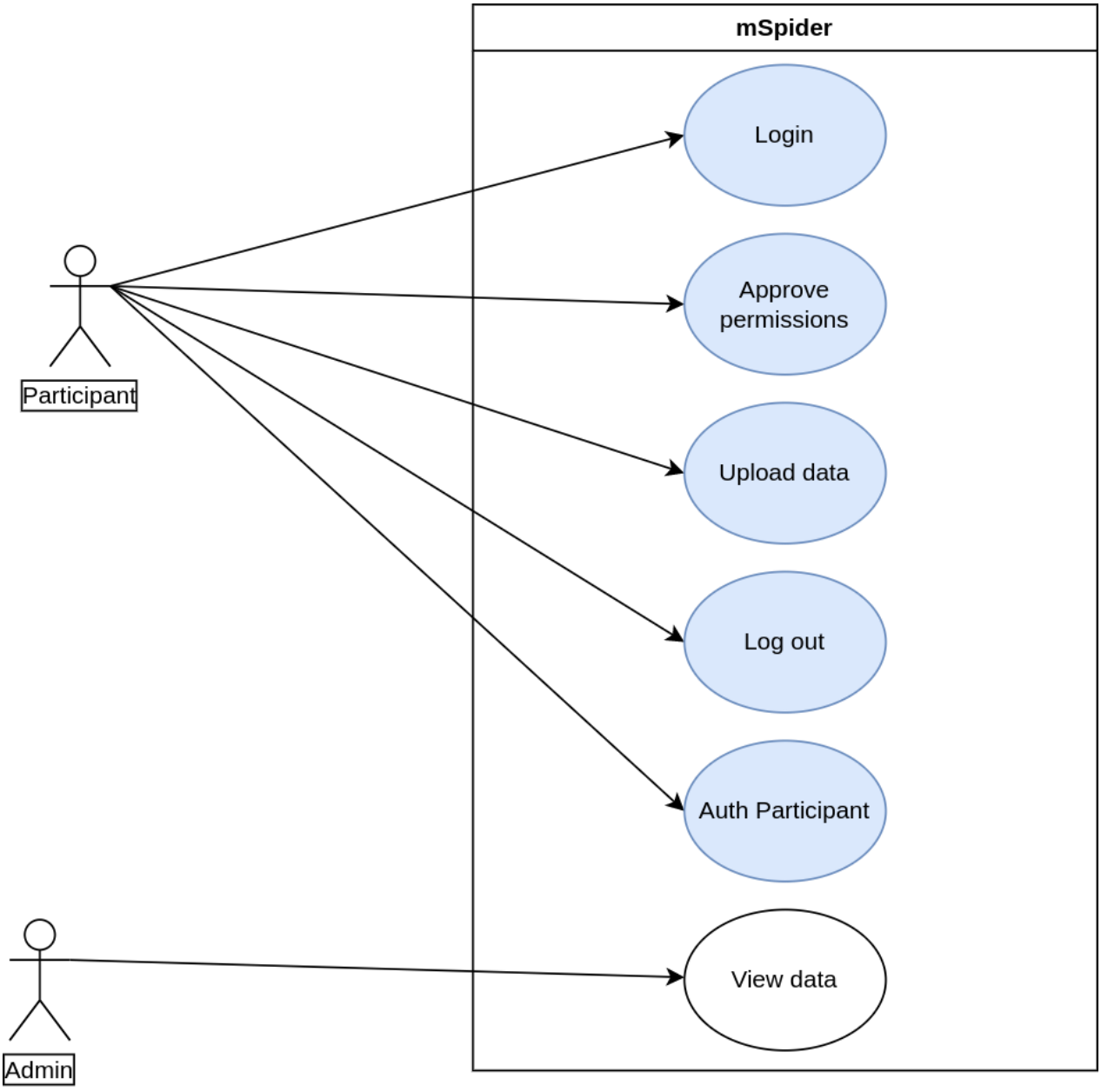

Figure 3 Use case diagram.

4.5 Personas and usage scenarios

A persona is a mockup of a user that might use a service or a product. The persona is there to showcase who might use the service or product offered. They are often in the target group of the product. Figure 4 and Figure 5 are pictures of two potential users, they are both smart watch users but have different technical skill levels.

The images were generated using Stable Diffusion [27]. Stable Diffusion is a way to generate images from natural text. The personas of both Mike and Robert were posted in the text field and images were generated using that.

Personas

Mike Smith

Mike works at a marketing firm in a large city. He enjoys walking to and from work, sometimes he uses his bicycle, and when it is raining, he takes the bus. Mike does not exercise other than the physical activity he gets from walking and cycling while doing his daily routine. Mike's daily routine includes going to work every day, sometimes after work he goes shopping for groceries. Mike lives alone with his cat.

Figure 4 Persona 1 Mike Smith

Mike likes to travel on his vacations, visit various places and explore their cultures. Mike is an average iPhone user, he has recently acquired an Apple Watch, and uses it every day. Mike enjoys using his watch to control his music, check notifications and the weather.

Mike saw a poster about a research project that seeks participants to conduct research about physical activity in the population. Mike has applied to join the research survey as a participant. Mike downloads the application and logs in and approves the permissions required.

Figure 5 Persona 2 Robert

Robert Pit

Robert is a man who does many things, he works at an office, with many other office workers. He likes to ride his motorcycle to work. He has a wife that likes cats, and together they live a peaceful life in the city.

Robert uses a Samsung Galaxy with a Galaxy Watch connected.

Usage scenarios

The participant is allowed to enter a study that uses mSpider iOS application. Firstly, the participant downloads the application from the Apple App Store. The participant then checks their email to find the invite code, sent by mSpider. Secondly, the participant opens the application, then the participant logs in to the application using the provided invite code. Thirdly the participant approves the application's request to read health and physical activity data from Apple Health. Lastly the participant closes the application.

Chapter Summary

An explanation about the requirements and use cases has been provided. Additionally, the chapter provides some examples of personas using the application.

5 Design

The design chapter will explain the decisions during the design phase of the project. The decisions explained in this chapter will impact both platforms, Android and iOS.

The mSpider login system and token storage work on Android and iOS as shown in Figure 6 and Figure 7. The screenshots shown in the Figure 6 is taken from a physical iPhone while the one in Figure 7 shows the application running on an Android emulator.

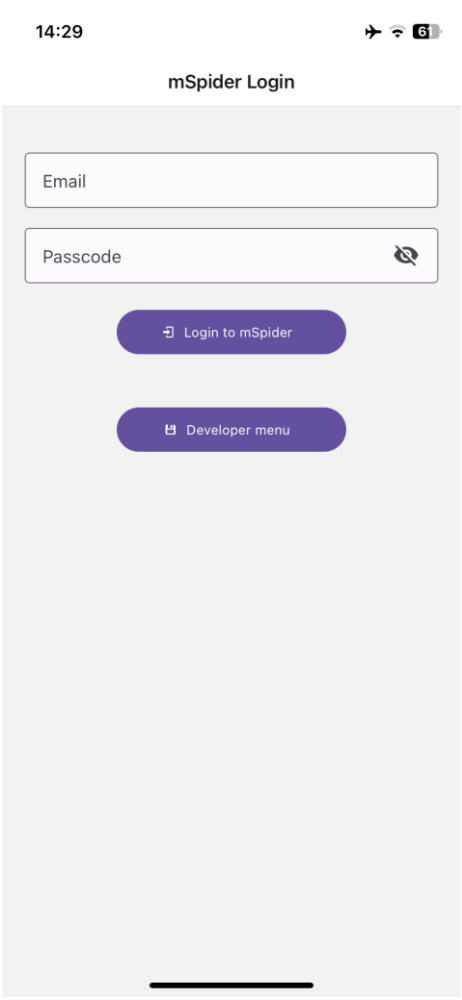

Figure 6 iOS

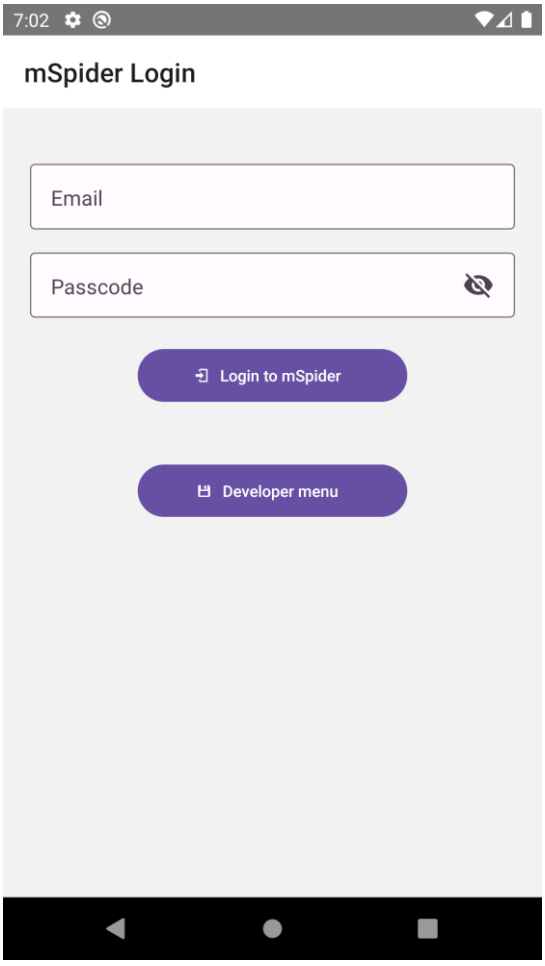

Figure 7 Android

The mSpider webserver has different providers that are in charge of fetching the health data from the different APIs and service providers that are supported by the platform. These providers are used to collect health data using Open Authorization (OAuth) tokens.

OAuth2 [28] is an internet standard for securely interacting with applications or API on behalf of a user. The mSpider mobile application does not use OAuth2, but the scope terminology is still used for the application.

The participant will upon registration be given an invite link and with that link they will be able to authorize data they want the mSpider platform to collect from their accounts at Google Fit or Garmin or Fitbit (Google LLC, CA, US) using a OAuth2 connection. The scopes are what data, the mSpider platform and its mobile applications has been given permission from the participants to collect and store into the database. The term scope is also used to describe the permissions given by a participant using the mobile application of mSpider.

Error! Reference source not found. shows two database tables, *apple_data* and participants. The *ID* field in *apple_data* is the primary key and generated by PostgreSQL automatically. The *apple_data* also contains the id of the participant(*participant_id*) which is connected to the participants table using a foreign key. The *date_time_reference* field contains an int collected from each record uploaded from the phone, the int is an epoch. An epoch timestamp is the number of seconds elapsed that have elapsed since the first of January 1970.

The *data_type* field contains information about what metric the value from the data field is. The scope field contains information about the object name used to call the object from the HealthKit SDK. The *data* field contains raw JSON (JavaScript Object Notation) of the uploaded data.

The participant database table contains four rows. The first-row *participant_id* is randomly generated by PostgreSQL and used to link the participants up to the uploaded records. The second row is *email*, and this may change in the production version of mSpider as data should not contain identifying parameters. Third row is the *username* of the participant and that may also change in the production version. The fourth row is the *apple_auth_token* that the participant uses to login to the application with.

The token table is not in use currently, but before the mSpider application goes into production, the token table should be used instead of *apple_auth_token* in *participants* table.

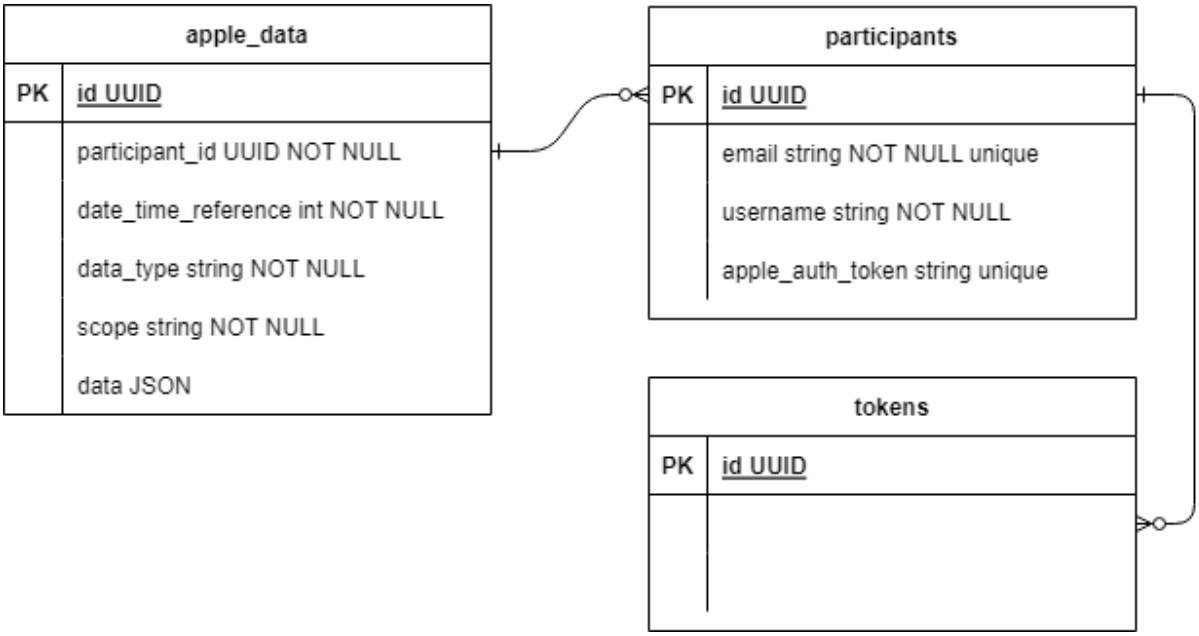

Figure 8 Database diagram

Figure 9 shows the participant entering the email and their invite code to the mSpider application. When the participant has pressed the sign in button in the application, the following will happen, the application will send a post web request to the mSpider API. This request is asking for the server to validate the existence of a record with the specific invite code that the participant provided.

If the server can validate the request, the mSpider API will send back a JWT token to the application and the application will save the token in a secure keystore on the phone. When this is done, the participant is presented with the home screen of the application and that the data uploads will start shortly and continue until either the study is over, or the participant signs out of the application and uninstalls it from their phone.

The data uploads will fetch the token from the secure storage of the phone and use that to upload health data when there is new data registered in Apple Health. The uploading prosses does not require further action from the user after initial setup. However, there is one requirement to uploading prosses, the participant needs to have background fetching enabled on their phone.

Every week the application will send a request with the JWT token to the mSpider API and request a new valid token to upload. This will continue until the token either is invalid or the participant stops using the application and logs out.

Authentication sequence for a participant with an iPhone.

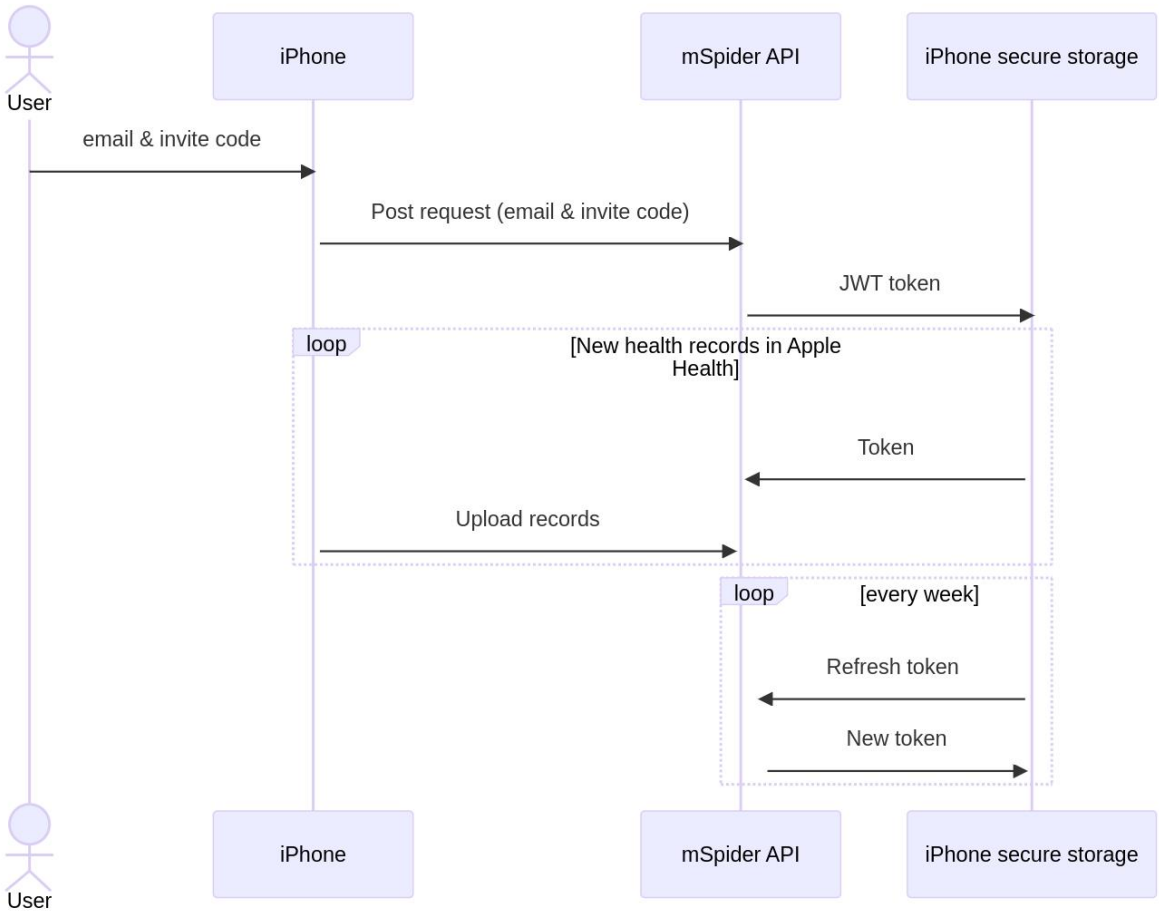

Figure 9 Sequence diagram of token use

- HTTPS Web request to the mSpider web API with email and passcode embed.
- JWT token is returned if passcode is accepted.
- JWT is stored on the participants' phone in a secure location and used in further and future communication with the server.
- JWT expires after the set time of a week.
    - The mSpider application must then use a refresh token to get a new token and save that to the secure storage on the phone.

Chapter summary

This chapter explains the database and how the systems (mSpider mobile application and mSpider web API) play together.

6 Implementation

This chapter will contain the specific details of how the mSpider mobile application was implemented and how the implementation is different on the different platforms (Android and iOS).

The initial plan was to implement mSpider as either an application on the watches themselves or as a cross-platform application on phone that communicates with the watches. After conducting research and finding out that watch application would require separate applications with platform specific code bases, this would increase the complexity of the development and future maintenance and was scrapped.

In the end, the option that was chosen was to only implement a data gathering solution for the Apple products (Apple Watch and iPhone). Samsung Health was considered not feasible in the current timeframe since it requires extensive implementation time to make it usable for the user and could be a separate master thesis.

RN-fitness-tracker[29] supports both Android and Apple iOS but not Samsung Health. This library supports Android using Google Fit. Google Fit is already supported with an API in the web application of mSpider.

Running services and applications in the background on iOS and iPhone is much more restricted than it is on Android as when the user navigates away from application, the application either gets suspended or terminated depending on the settings of the phone. The restriction on such services makes it more difficult to have an application that runs in the background and uploads data on a regular basis.

Manual upload is a possible solution and was done in Bartschke, et al. [11] and in Shcherbina, et al. [22] where upload without an application was performed. However, there are down sides with manual upload and manual extraction as it requires that the user must open the application and let it upload data, but this method is relying on that the participants remembers and have time to interact with the application on regular basis.

[29] https://github.com/kilohealth/rn-fitness-tracker

To utilize background processes with the restriction in mind, the application uses background fetch that is initialized by Apple Health. This is a feature in HealthKit that fires an event when there is new data saved.

The data retrieval frequencies are defined by Apple in their developer documentation and selected by the developer. Apple Health will send an internal notification using HealthKit to the mSpider application, and this allows the mobile application to wake up, and fetch new data from the Apple Health database.

During the first phase of the development, the author tested three different libraries for collecting health data from Apple Health and found two open-source libraries (AE Studio 's React-Native-Health[30] and Kingstinct's React-Native-HealthKit[31]). These libraries were easy to set up and test on an Apple Mac mini and get results quickly. Both libraries also support background delivery of health data from Apple Health to mSpider. The support of background delivery is crucial to the mobile application.

AE Studio's React-Native Health was the first iOS only React Native library that the author tested on a physical iPhone. After initial testing of the library, it became clear that the implementation of it was outdated as AE's library uses Objective-C as a backbone for interfacing with Apple HealthKit. Since the introduction of Swift in 2014, Objective-C has become less popular to use in new applications, as Apple themselves uses Swift in their new applications.

The AE's library uses React Native event emitters to listen to background updates from Apple Health. Event emitters are special types of listeners that listen for an event to happen and when the event has happened, they notify the subscribers of the listener to update themselves with the new state of the event that fired.

[30] https://github.com/agencyenterprise/react-native-health

[31] https://github.com/Kingstinct/react-native-healthkit

The AE's background observers need four separate event listeners (setup success, setup failure, new listener, and retrieval failure) for each data type. A listener for each state is required the mSpider application using the AE library had to subscribe to four emitters simultaneously to access six different data types, each with their own method to call and get a response from.

The AE library does not offer a way to iterate over the results from the different data types requested from Apple Health, each data type had to be called separately and their return value had to be handled separately and then processed for uploading to the mSpider API.

The AE library has a limited set of datatypes to query from, and this set is further reduced if the application needs to access them in the background, and it is this limitation that led the author to look for a different library to access data from Apple Health using HealthKit.

Kingstinct's React-Native-HealthKit is the second React Native library the author tried to use with a physical iPhone. The library uses Swift and TypeScript as a backend for connecting to Apple Health. Furthermore, the same object and function names as found in Apples documentation are used.

The outputs returned from calls are similar to what a native Swift developer would get from HealthKit. React-Native-HealthKit is easier to maintain because of the large connection it has with the native Apple documentation.

Background delivery works differently in React Native HealthKit than in the AE library. To setup background delivery from React Native-HealthKit to the mSpider application, the author made a list of permissions at the top of the JavaScript file, and this list was passed to the permissions request method and to the method that handles the querying of information from Apple Health. This implementation method makes for highly modular code, that can easily be called upon and reused when needed.

The different quantity data types from HealthKit are called *HKQuantityType*[32], and these types store all the information about each individual sample stored in Apple Health. Each of the data types in Apple HealthKit has one, one example is *distanceWalkingRunning* and this type is called *HKQuantityTypeIdentifier.distanceWalkingRunning* when called from the code.

The *HKQuantityType* types are called both during the initial permissions request from the participant to read them and during the process of uploading them the mSpider server after retrieval from Apple Health. React Native HealthKit does not use a separate listener for each state.

The mSpider application supports six types of HealthKit quantity records. Table 4 shows a list over HealthKit type identifiers and their friendly names that is used in Apple Health.

Table 4 HealthKit object names

HealthKit name	Apple Health name
HKQuantityTypeIdentifierDistanceWalkingRunning	Walking + Running Distance
HKQuantityTypeIdentifierActiveEnergyBurned	Active Energy Burned
HKQuantityTypeIdentifierDistanceCycling	Cycling Distance
HKQuantityTypeIdentifierFlightsClimbed	Flights Climbed
HKQuantityTypeIdentifierStepCount	Steps
HKQuantityTypeIdentifierHeartRate	Heart Rate

The only factor that changes during each iteration is the input parameter of what *HKQuantityType* is next in the queue to be uploaded. This implementation makes for modular code, and since the same methods are used for all the data types.

[32] https://developer.apple.com/documentation/healthkit/hkquantitytype

React Native HealthKit uses the *HKQuantityType* to keep what data belongs to which sample and it makes it easier to add new data types in the future. This allows the application to be expanded to support both currently unsupported types and new types that Apple adds in the future.

React Native HealthKit supports other types of samples (Category types and Workout Activity types) so these could be added in a future update to the mSpider application. This will make the mSpider application able to collect nearly any data that is stored in Apple Health and that could be of interest for a future researcher. Third party applications can also push records to Apple Health, making them available for the mSpider application as well. This means that any application that supports writing to Apple Health could be supported by the mSpider application.

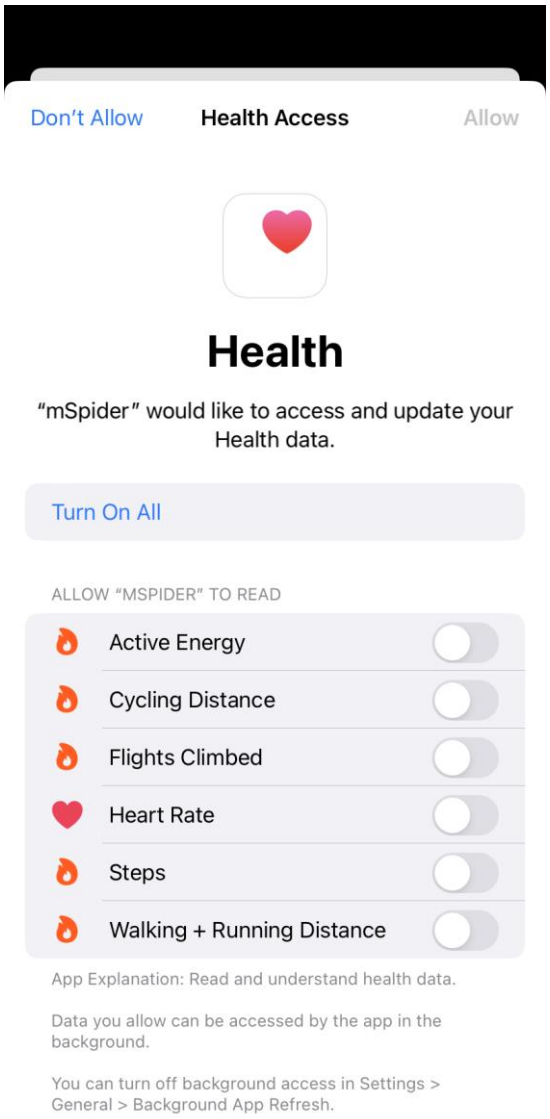

Figure 10 Permissions request on iOS

Figure 10 shows the permissions request from the mSpider application.

The mSpider application makes use of a list of permissions and the list is the same *HKQuantityType* as the data collection loop uses. The participants will be asked to allow the mSpider application to read from the different scopes in Apple Health when they log in the first time. When a participant declines mSpider to read from a specific permission, the application skips over it as it was not there in the first place.

The mSpider iOS application uses a HealthKit anchor to be able to store what Apple Health samples, the application has read and uploaded and what samples it has not uploaded. During the upload loop, the application will only mark a sample as uploaded if the mSpider API server sent a HTTP status code of 201 Created in return after records have been uploaded.

Figure 11 shows a code snippet of the uploading loop and the anchor mechanism.

```
        type.forEach(permission => {
            HealthKit.useSubscribeToChanges(
                permission,
                async () => {
                    const savedAnchor = await Keychain.getGenericPassword({ service: "anchor" });
                    let anchor = savedAnchor.password;

                    //const data = await HealthKit.getMostRecentQuantitySample(permission);

                    const data = await HealthKit.queryQuantitySamples(permission, {
                        // limit: 10,
                        anchor: anchor,
                        to: new Date(),
                        from: addHours(new Date(), -168)
                    });

                    if (!data) {
                        return;
                    }

                    const response =  await uploadDataBatch(data.samples, jwtToken);

                    if (response == null) {
                        return;
                    }

                    if(response.status == HttpStatusCode.Created) {
                        await Keychain.setGenericPassword("anchor", data.newAnchor, { service: "anchor" });
                    }

                    if (response.status == HttpStatusCode.Unauthorized) {
                        await handleRefresh(jwtToken)
                    }

```

Figure 11 Code snippet from upload loop

The code snippet showed in Figure 11 starts with a for each loop that will loop trough every *HKQuantityType* that has been set in the *Type* list. After the list has been read, the application will try to load the anchor from the phone storage. Once the anchor has been loaded, the code execution continues with querying the new samples since the last upload, and it queries seven days back and until the current time. After the samples have been loaded from Apple Health and they are not empty. The samples will then be sent to the mSpider API in batches for each scope using the JWT-token loaded from the phone's storage. If the result after sending is 201 Created, the sample will be marked as uploaded in the anchor, and the loop continues until there are no more samples left that have not been uploaded and marked as such.

Data upload

Figure 12 shows a record from the PostgreSQL database uploaded from the iPhone.

```
0de4d9a5-f6f3-4e8a-88e6-97a7bc7bc6c3 | 9d1d43fc-f14a-4e23-886f-
27ef62a09db2 | 1686247823000 | kcal |
HKQuantityTypeIdentifierActiveEnergyBurned| 0.21599999999999997
```

Figure 12 Example data upload

The record is explained as follows. The first row is the unique row id generated by the PostgreSQL database system. The second row is the *id* of the participant who generated the row. The third row is the epoch timestamp of the record and that converts to Thursday 8th June 2023 at 20:10:23. The fourth row is the data type, and this data type is kilocalorie(kcal).

The fifth row is the identifier of the object, named from the Apple Developer documentation *HKQuantityTypeIdentifierActiveEnergyBurned*[33]. The sixth row is the actual data that was uploaded. This record was collected from an 8th generation Apple Watch that the author uses to test the implementation and uploaded using an iPhone XR.

The figure starts at the participant (Participant, 1). The participant is wearing an Apple Watch, this Apple Watch is connected to the iPhone as shown in 2. Apple Health (3) is the local data store for health data. Once the data from the Apple Watch is saved in Apple Health, and the phone is connected to the internet and has iCloud backup enabled, it will get transferred to the iCloud backup system, as shown in 4. The transfer of data from Apple Health to the mSpider mobile application happens with a connection between Apple HealthKit (5), React Native HealthKit (6) and mSpider iOS (7). The health data from the participant (1) is uploaded to (8) and stored in the mSpider database (9).

The ability to disable iCloud backup (4) is an optional feature, but this is important to the process as depicts, the option to have a closed loop system. When the backup feature is disabled, the health data from the participant is only uploaded to the mSpider system and not to Apple's own systems.

[33] https://developer.apple.com/documentation/healthkit/hkquantitytypeidentifier/1615771-activeenergyburned

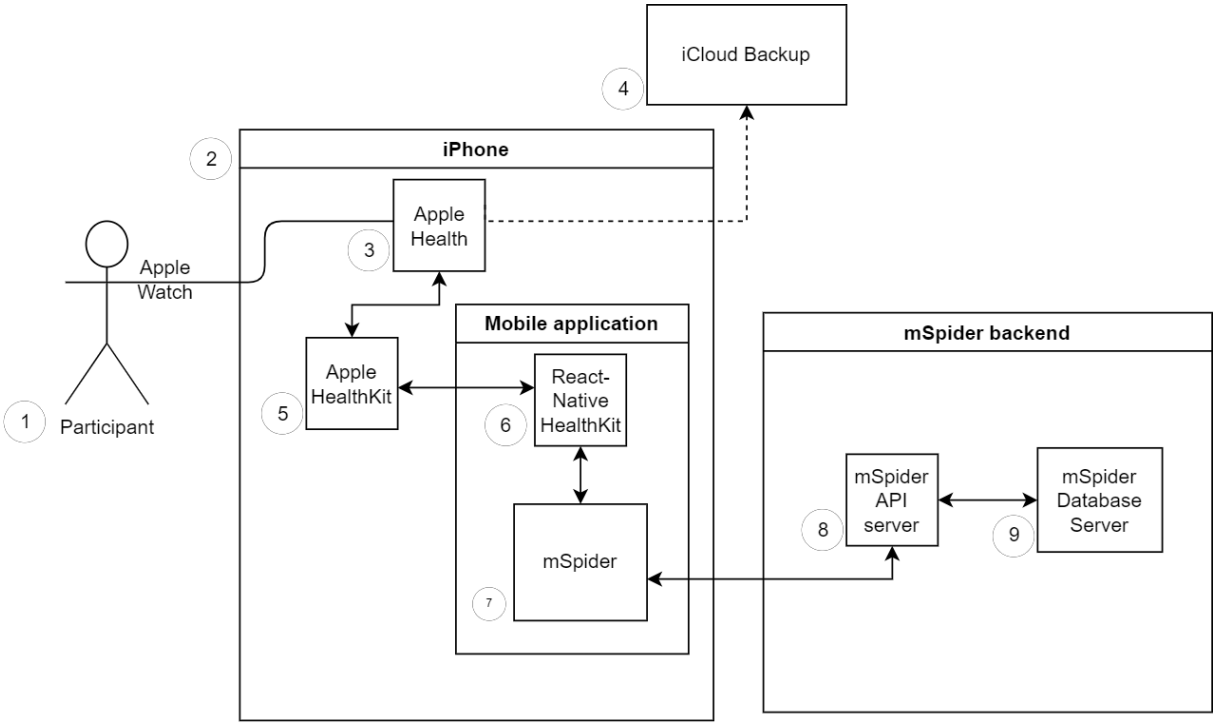

Figure 13 Uploading from iPhone and Apple Watch

- Stage 1 depicts the actor/participant wearing the Apple Watch.
- Stage 2 depicts the data being transmitted between the iPhone and the Apple Watch.
- Stage 3 depicts the health data being stored locally in Apple Health.
- Stage 4 depicts the flow of data between Apple Health and the optional backup feature iCloud, provided the user has not disabled backup of Apple Health data to iCloud.
- Stage 5 depicts the transfer of data from 5 to 7 using Apple HealthKit and React Native HealthKit.
- Stage 6 depicts that React Native HealthKit is used to transfer health data from Apple Health and to mSpider.
- Stage 7 depicts the mSpider iOS application transferring data from the iPhone and to the mSpider cloud server.
- Stage 8 depicts the mSpider API receiving health data from the iOS application.
- Stage 9 depicts the data received by the API being saved into the mSpider database server.

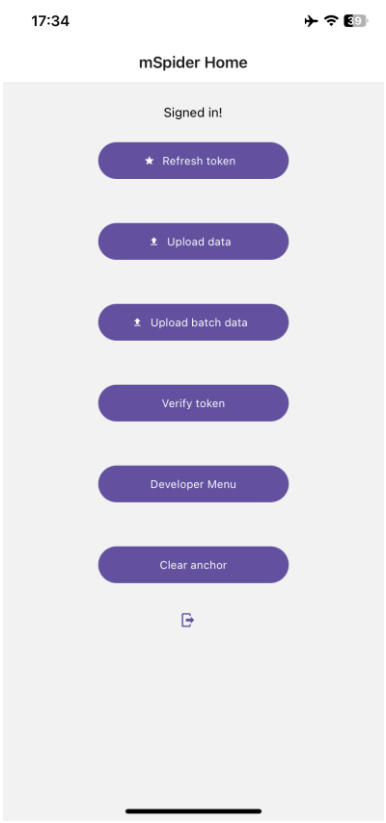

Figure 14 User interface in iOS

Figure 14 shows the user interface in iOS that used during development, the user interface in the mSpider application uses React Native Paper and React Native Navigation. These libraries are cross-compatible, making the interface and other core functions the same on both platforms (iOS and Android). This will allow for the future implementation of the data collection for Samsung, while relying on other features that are cross-compatible and already present in the iOS version of the application.

Other cross platform features include the login and uploader features that make use of the JavaScript library *Axios*. The React Native Library *React Native-Keychain* handles the storage of the JSON Web Token (JWT) and the anchor from Apple HealthKit. The JSON Web Token is a system and a proposed internet standard [29] for handling access tokens.

The mSpider application is meant to run in the background on the participants phone, so the user interface has not been a top priority, and therefore the user interface has a lot of potential in terms of future work. Currently, the user interface offers the ability to log in and out to the application, and the rest of the functionality is automatic. Uploading happens automatically when Apple Health has new data from either the iPhone or the Apple Watch.

For future work, it could be useful to look in to if it is possible to have a dynamic list that is requested from the mSpider server, and based on that it will customize the list in the application, so that the researcher can also customize what kind of data they want to collect, making the collection more specialized and therefore the researcher will not get data they do not intend to use or wish to use, and at the same time making it easier to analyze data after the data collection is completed.

Chapter summary

This chapter described the proof-concept application that was implemented for iOS in React Native using React Native HealthKit as data source.

7 Results

This chapter will explain the results the author has come to before, during and after the development of mSpider application. The figures shown in this chapter are graphed with Grafana. Figure 15 shows an overview of the 60 days of data collection the author did during the period. The data was collected from 1sth of May and until the end of 28th of June 2023.

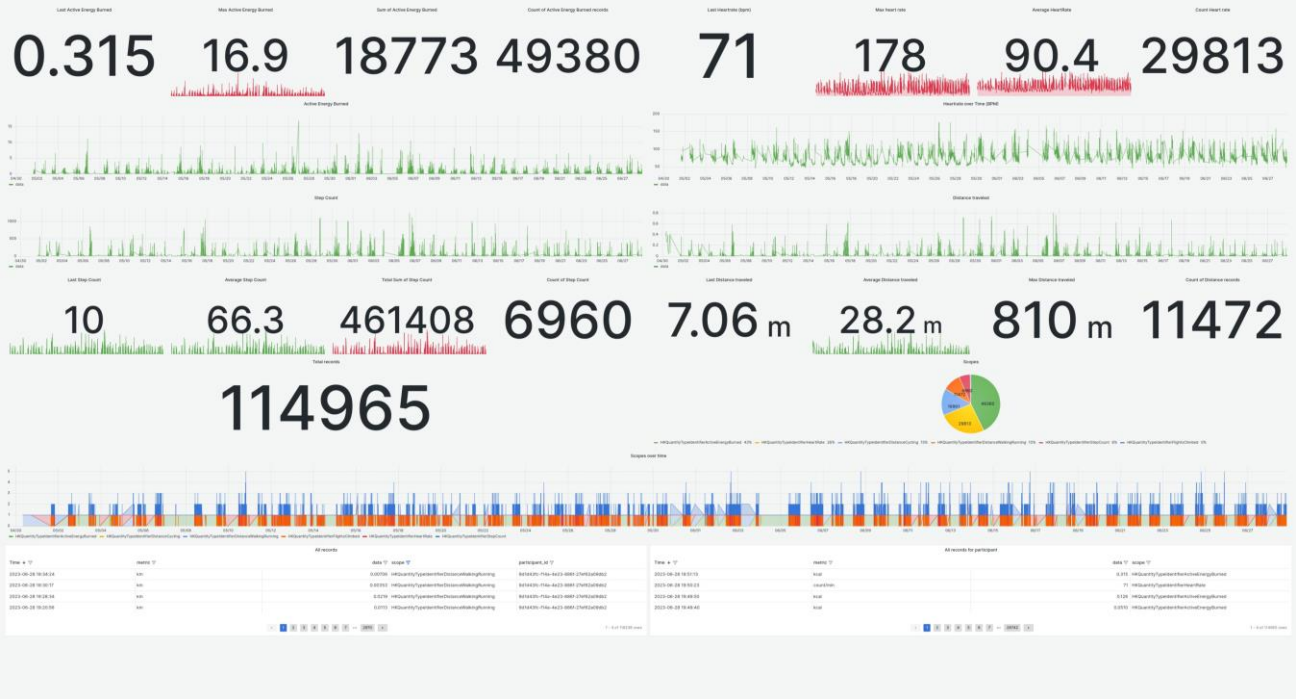

Figure 15 60 days with records from the mSpider application

7.1 Data collected and battery usage

164 upload requests were sent to the server during a 24-hour period on the 19th of June 2023. These upload requests resulted that 3905 records were stored in the PostgreSQL database. The mSpider application used 11 percent of battery on an iPhone on the 19th of June. The battery usage is collected from the battery usage screen in the settings menu of iOS.

Figure 16 shows the records uploaded on 19th of June grouped by scope.

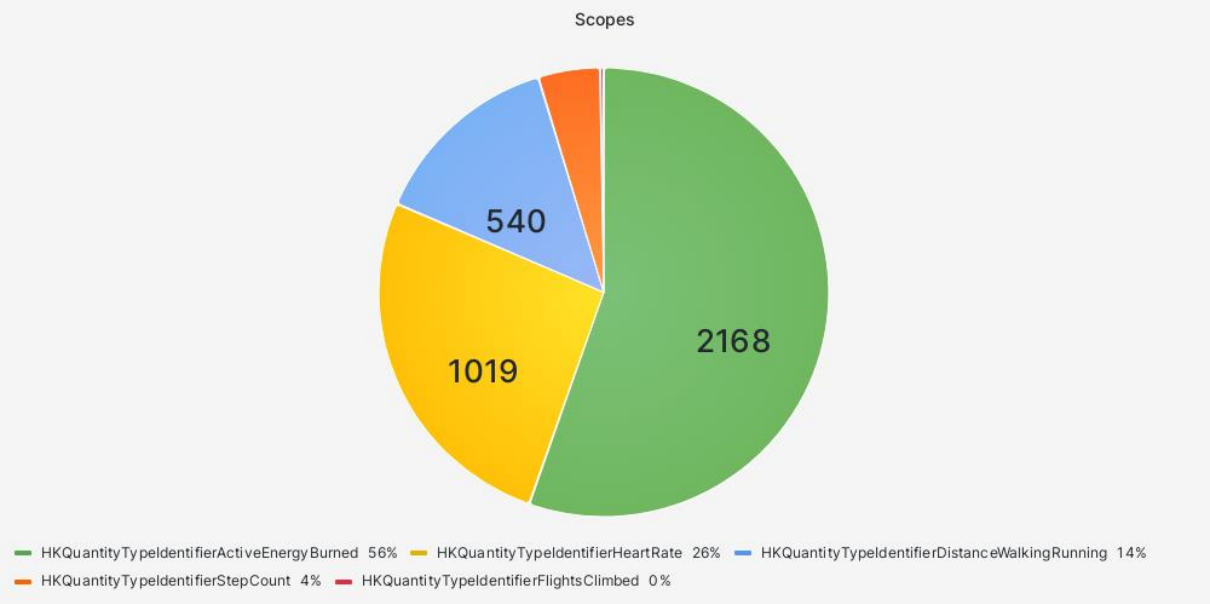

Figure 16 24-hour overview of scopes on 19 of June

The figure shows that 56 % percent of the uploaded records were active energy burned. This is expected as this is the record that the Apple Watch saves the most.

In the days since the 19th of June and until 28th of June, the mSpider application has used six percent of battery, and there were 26820 records added to the PostgreSQL database. As shown in Figure 17.

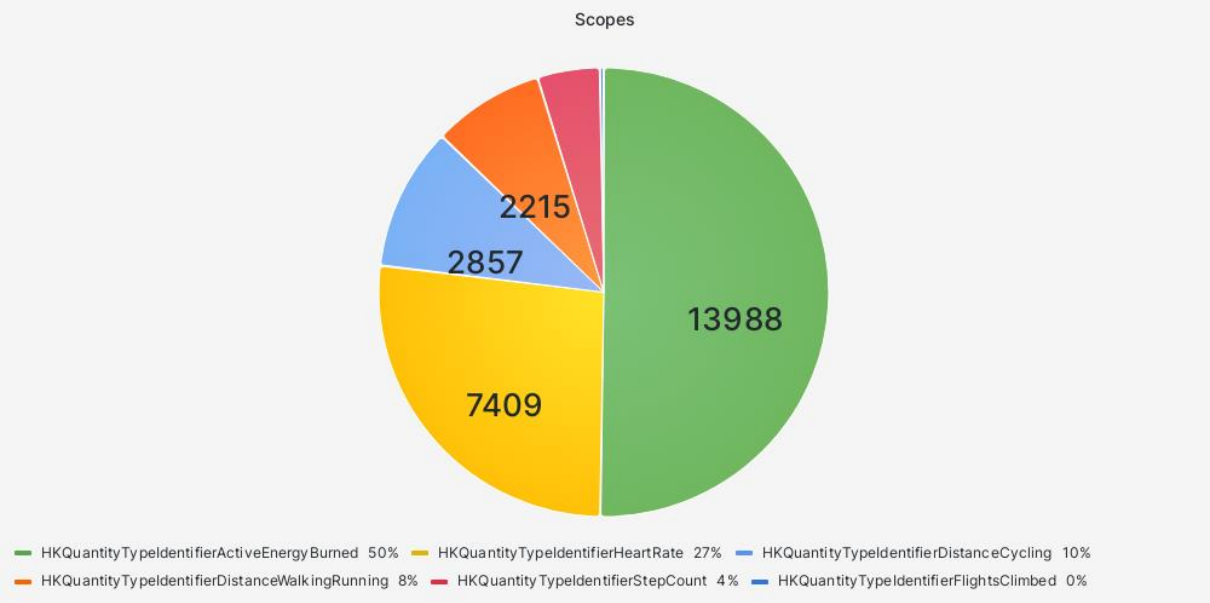

Figure 17 10-day view on scope

The figure shows that again the scope with the most records uploaded is active energy burned.

Table 5 shows the records in numbers from Figure 15

Table 5 Count of records over 60 days

Record name	Count
Active energy burned	49380
Heart Rate	29813
Steps	6960
Distance traveled	11472
Total	114965

Table 6 shows the estimated size of one data record uploaded.

Table 6 Estimated data size in bytes

Name	Data	Bytes per record
Object ID	0de4d9a5-f6f3-4e8a-88e6-97a7bc7bc6c3	37
Participant ID	9d1d43fc-f14a-4e23-886f-27ef62a09db2	37
Timestamp	1686247823000	8
Metric	kcal	16
Scope	HKQuantityTypeIdentifierActiveEnergyBurned	35
JSON Data	0.21	50
	183 bytes multiplied by 114965 records =	21 megabytes

If the results from Table 5 are multiplied with the data from Table 6 the result is 21.

21 megabytes is the total megabytes used for all the data that the author collected on themselves during the two-month collection period. When the data is extracted from the PostgreSQL database the file is also 21 megabytes in size.

7.2 User testing

The mSpider iOS user interface was shown to other students during the development, to get feedback on the user interface. The feedback showed that the application lacks an informative screen to inform the participants about the study they are enrolled in. Other students have not been used in the testing of data gathering from the Apple Watch.

User testing with data collection from private devices has not been conducted as that would have been a challenge with Apple devices as they require that the developer either publishes the application using TestFlight and invites the users through this system or publishes the application to the App Store.

7.3 Security

7.3.1 Apple

Apple has requirements that every developer that wishes to publish to the App Store (Apple Inc, CA, US) needs to follow. These requirements include that the developer needs to own a Mac capable of running the Xcode for the development and compilation and testing of their software packages. Developers also need to pay a fee to Apple to be granted access to the Apple Developer Program[34] (Apple Inc, CA, US). The Developer Program has different levels of access depending on how large the business is.

A single developer needs to pay 99 USD to enter the program, while a larger enterprise with 100 or more employees needs to pay 299 USD to enter the Apple Developer Enterprise Program[35] (Apple Inc, CA, US). The fee is a subscription as the developers need to pay it each year.

There are benefits to requiring developers to pay a fee to be allowed to publish to the App Store. One large reason for the fee is to increase the barrier and to hinder the publication of spam and malicious applications. The App Store also has strict regulations for what Apple deems an application worthy of being allowed onto the App Store as outlined in their App Store Review Guidelines[36] and polices, the fee serves as first line of defense.

---

[34] https://developer.apple.com/programs/

[35] https://developer.apple.com/programs/enterprise/

[36] https://developer.apple.com/app-store/review/guidelines/

HealthKit background fetching is one of the features a developer enrolled into the Apple Developer Program gets, this feature is used heavily in the mSpider iOS application. To test the background fetching on a physical iPhone, the developer needs to be a member of the program. This requirement does complicate the process for a small-time student developer.

If the user declines a permission request for access to health data from Apple Health, the application requesting the permission will get no notice of the permission being declined, and that is how it is designed by Apple. Apple states that the reason that applications do not know what permissions are declined is to protect the user's privacy[37].

Apple Health offline data storage

Apple does not act as an intermediary[3839] when an iPhone user allows a third-party application access to their data stored in Apple Health. The shared health data is transferred directly from Apple Health and to the third-party application. If the user wishes to keep the data stored only on their iPhone, they can do so by disabling the iCloud backup feature (the iCloud backup is enabled by default), while this is not required, it is a nice option that will allow for closed loop data gathering.

Closed loop data gathering solutions opens new opportunities for research data gathering that has previously been harder to accomplish. There are researchers that would prefer to remain in complete control of the collection of data, and this option will appeal to them. Closed loop systems also make risk analysis easier as there are fewer parties to account for in terms of who has access to the data. The findings from this satisfies the third sub-problem to some degree.

7.3.2 Samsung and Google

Samsung has an SDK (Samsung Privileged Health) similar to what Apple offers with HealthKit, but it is harder to gain access to it, as developers need to request access to the software kit using a Samsung account and then Samsung needs time to go through the request and then approve it before you can start testing your application with physical hardware.

---

[37] https://developer.apple.com/documentation/healthkit/authorizing_access_to_health_data

[38] https://support.apple.com/en-us/HT209519

[39] https://www.apple.com/legal/privacy/data/en/health-app/

Samsung has also restricted the system so developers cannot test the SDK features of the Privileged Health program in an emulator, making it harder for developers to test their applications quickly. Entering the program is not an easy task according to a developer relation spokesperson Ron at Samsung[40].

Developers publishing an application to the Google Play Store (Google LLC, CA, US). The developers would also be subjected to similar restrictions and guidelines from Googles as outlined in the Google Play Developer Distribution Agreement[41]. Google charges a 25-dollar one-time fee to be able to upload applications to the Play Store. The fee charged by Google is smaller than what Apple charges.

7.4 Limitations with using React Native.

Platform dependent dependencies, and that means, that each platform requires their own packages, and these packages change based on if the device is an Android or an iPhone, an example is the HealthKit package is used in the application. The package uses Swift native code and therefore only supports the iOS platform.

Another limitation with React Native is the dependencies that depend on other dependencies, and to utilize them the developer must install large amounts of packages. Loss of support from developer of a package is a further limitation that can lead to the application being unsupported on newer phones, and then requiring an update to swap out the deprecated packages / outdated packages, making the maintenance harder.

[40] https://forum.developer.samsung.com/t/i-need-to-get-data-from-samsung-health/19625/4

[41] https://play.google.com/intl/en_us/about/developer-distribution-agreement.html

8 Discussion

8.1 Thesis summary

Collecting data from smart watches is an interesting way of gaining insights about the population. This thesis consists of an explanation about Apple Watch, and Samsung Galaxy Watch and how to collect data from them using an automated system. The thesis also covers the theoretical framework needed to explain the core concepts, a literature review, and a requirement specification. The thesis further explains the design and implementation of the mSpider mobile application, and results from the data collection and results from battery consumption and the results of the other findings.

8.2 Data collection

During the phase after development had been paused on the mobile application, the author wore the Apple Watch for an extended period, this allowed the author to test the application for a longer time, and the amount of data is surprisingly large for one person to have generated in just two months, while the storage required for the records is low, this is good news for the participants as that means the application will have low impact on their cellular data if they choose to allow mSpider to use mobile data.

The application also has a low impact on the battery of the participants. The results from the battery usage are an estimate as the phone used was put in flight mode as it did not have a sim card activated. The phone was not used as a primary phone during the period as the author used their regular phone as well during the two-month collection period.

The data generated by the author and shown in 7.1, gives an estimate of how the application would have performed in a real production environment. This data set is an estimate based on one person usage of the system and this person (the author) was more enthusiastic than a regular participant might have been, as the author did make an effort to wear the Apple Watch whenever possible, while a regular participant might forget to use the watch and not record data to the phone, causing a lower amount of data to be collected.

8.3 Privacy and security

Once the uploaded data has reached the mSpider server and database. If the data gets anonymized before it is stored on disk, it would be easier when it comes to privacy as with no identifiable information, no privacy or trust can be broken.

While if the data were to be stored with identifying information left in it, it could help researchers gain a larger understanding about the population on a more individual level. To keep data with identifying information in it, the researchers would need to apply to the Norwegian Research Ethics board (SIKT) for approval.

The author has not put in an application with SIKT for this thesis or the mSpider application. There are three reasons for not sending a request to SIKT. The first one is that the data that has been collected has only came from the author, and the second reason is that the collected data contains no identifiable information about the author. The third reason is that the mSpider mobile application is still under development and not ready for use with real participants as currently the application is still in a proof-of-concept stage.

8.4 User testing

As user testing was not done on actual participants or other students. There are fewer elements to discuss. While the chance was there to test on actual participants, this was not done due to time constraints.

In 7.2, feedback was shared with the author about the user interface, that the interface should give information about the studies and the author agrees with the feedback given but has not had time to implement it.

8.5 Strengths and limitations

There are strengths and limitations with this project as any other project.

High barrier to entry to access studies, as the participant must purchase at least a high-end smart phone to be eligible for the study. The cost of entry further increases when the participant needs either purchase or own the Apple Watch or a Samsung Galaxy Watch for most of the records the be saved.

Large portions of the population do own these devices from before, but the ones that do not own them, will not be able to take part in the studies offered through the mSpider mobile application. These participants can however gain entry to the studies by using other less costly devices (Fitbit and the like).

The Apple Watch has fitness motivational features that could impact the data collected through Apple Health, which could make the results different from a person that is not wearing an Apple Watch and therefore not necessarily representable of the general population while these features are enabled.

There are multiple ways a participant can trick the Apple Watch. The Apple Watch is susceptible to the participant shaking the Apple Watch or using other ways to trick the Apple Watch into detecting exercise.

Activity data can also be manipulated if the user starts a workout session on the watch and then proceeds to just shake the watch repeatedly. Data manipulation normally only affects the user. If the user were to cheat while part of a research project that could change the results. This could however be mitigated using motion data[30].

In this article they have participants trying to trick a fitness tracker installed on a cell phone and when the participant successfully cheats the system, the authors retrain the model until the participants were not able to trick the detection.

In Apple Health, the user can add manual records to Apple Health, thus making it seem like they were more active than they were. This can also be mitigated by not uploading manual records.

8.6 Problem statements

The introduction outlined the statements and potential solutions, and those solutions were further detailed in the design chapter and the implementation chapter.

SP1: How can health data be collected from smart watch vendors that do not support backend API data retrieval?

This sub-problem was resolved, as the solution described in the sections 5 and 6 have shown that is possible.

SP2: How can data be collected while preserving the privacy and security of the participants?

This sub-problem was not completely resolved, as this project only has developed a proof-of-concept system. Security and privacy concerns described in sections 2.7 and 8.3 should be evaluated further before the system is used for any data collection that involves any personal data or health data covered by GDPR or HIPAA. An application to SIKT is required as well.

SP3: *How can collected data be prevented from being sent or stored on vendor services?*

This sub-problem was resolved for the iOS application. As described in 7.3.1, if the user disables the backup feature there is no data that will leave the phone unless the user has accepted it.

8.7 Contributions

Automatic data collection from Apple Watch with a background process, that does not require manual management after initial setup. This process allows for complete control over the data collection and storage. The mSpider mobile upload system has a low cost as it does not require researchers pay a third party for access to their data as would have been necessary if a similar but commercial system like WeFitter or Terra was chosen to be the platform for data collection.

Login system for mSpider with support for both Android and iPhone, so that another master student in the future can add data uploader support for Android.

8.8 Future work

General improvements

- Develop a nice-looking user interface for mSpider.
- The mSpider application should only upload data on Wi-Fi networks.
    - Give the participant options where they can select what they want.
        - A choice for the participant to enable data upload over cellular data / 4G / 5G.
- Check that there is data to collect, to avoid looking after no existent data.
- Show last uploaded data to let the user know when data was last uploaded, and that the application is working as normal.
- Fetch and show information about the current user / study in the main page of the application.

Android version improvements

- Develop the Android version of mSpider to have more features and get it up to par with the iOS version.

iOS improvements

- Check for duplicated data from either iPhone or Watch.
    - Add the device source of the uploaded data.
- Apple ResearchKit is not used in the iOS version of mSpider data collector. A future version of mSpider could implement ResearchKit as it grants access to more metrics and usage information that is not available otherwise.

As mentioned in chapter 6, a feature that would have been beneficial for both the participants and the researchers analyzing the collected data is that mSpider application would get a list of permissions or scopes from the mSpider API at login. The mSpider application would then base the permissions request on the list of permissions gotten from the server, giving researchers customizability in the data they collect. The customizability would both be helpful and provide more privacy to the participant.

9 Conclusion

The findings presented in this report and the literature review show that it is possible to collect data automatically while still preserving privacy and security of the participant.

The thesis contributes to the existing knowledge by providing an implementation and discussion of a proof-of-concept system for collecting health data from wearables that do not support online data extraction while preserving the privacy and security of the data originators using a third-party application.

As discussed in section 8.6, resolutions to the problem statements have been found through an automated smartphone application for health data collection where the participant is only required to provide login credentials and accept permissions. Not all the problem statements have been fully resolved, but it shows that it is possible but further research is required to comply with international regulations.

References

[1] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32-39, 2011, doi: 10.1109/mcom.2011.6069707.

[2] J. Burke, "Participatory sensing," presented at the ACM Sensys 2006, Colorado, USA, 2006.

[3] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140-150, 2010, doi: 10.1109/mcom.2010.5560598.

[4] *Physical activity fact sheet*, World Health Organization, Geneva, 2021. [Online]. Available: https://apps.who.int/iris/handle/10665/346252

[5] D. Thivel, A. Tremblay, P. M. Genin, S. Panahi, D. Riviere, and M. Duclos, "Physical Activity, Inactivity, and Sedentary Behaviors: Definitions and Implications in Occupational Health," *Front Public Health*, vol. 6, p. 288, 2018, doi: 10.3389/fpubh.2018.00288.

[6] M. S. Tremblay *et al.*, "Sedentary Behavior Research Network (SBRN) - Terminology Consensus Project process and outcome," *Int J Behav Nutr Phys Act*, vol. 14, no. 1, p. 75, Jun 10 2017, doi: 10.1186/s12966-017-0525-8.

[7] Apple, "Apple reinvents the Phone with the iPhone," Apple Newsroom, January 9, 2007. [Online]. Available: https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/.

[8] *General Data Protection Regulation*, European Union 2016/679, 2016.

[9] A. Henriksen, E. Johannessen, G. Hartvigsen, S. Grimsgaard, and L. A. Hopstock, "Consumer-Based Activity Trackers as a Tool for Physical Activity Monitoring in Epidemiological Studies During the COVID-19 Pandemic: Development and Usability Study," *JMIR Public Health Surveill*, vol. 7, no. 4, p. e23806, Apr 23 2021, doi: 10.2196/23806.

[10] M. J. Page *et al.*, "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, vol. 372, p. n71, Mar 29 2021, doi: 10.1136/bmj.n71.

[11] A. Bartschke, Y. Borner, and S. Thun, "Accessing the ECG Data of the Apple Watch and Accomplishing Interoperability Through FHIR," (in eng), *Stud Health Technol Inform*, vol. 278, pp. 245-250, May 24 2021, doi: 10.3233/SHTI210076.

[12] A. Curtis, A. Pai, J. Cao, N. Moukaddam, and A. Sabharwal, "HealthSense: Software-defined Mobile-based Clinical Trials," presented at the The 25th Annual International Conference on Mobile Computing and Networking, Los Cabos, Mexico, 2019. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3300061.3345433.

[13] H. G. Dandapani *et al.*, "Leveraging Mobile-Based Sensors for Clinical Research to Obtain Activity and Health Measures for Disease Monitoring, Prevention, and Treatment," (in eng), *Front Digit Health*, vol. 4, p. 893070, 2022, doi: 10.3389/fdgth.2022.893070.

[14] D. Fuller *et al.*, "Predicting lying, sitting, walking and running using Apple Watch and Fitbit data," (in eng), *BMJ Open Sport Exerc Med*, vol. 7, no. 1, p. e001004, 2021, doi: 10.1136/bmjsem-2020-001004.

[15] A. Goldberg and J. W. K. Ho, "Hactive: a smartphone application for heart rate profiling," (in eng), *Biophys Rev*, vol. 12, no. 4, pp. 777-779, Aug 2020, doi: 10.1007/s12551-020-00731-3.

[16] K. Hänsel, H. Haddadi, and A. Alomainy, "Demo: AWSense: A Framework for Collecting Sensing Data from the Apple Watch," presented at the Proceedings of the

15th Annual International Conference on Mobile Systems, Applications, and Services, Niagara Falls, New York, USA, 2017. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3081333.3089333.

[17] S. Kunchay and S. Abdullah, "WatchOver: using Apple watches to assess and predict substance co-use in young adults," presented at the Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, Virtual Event, Mexico, 2020. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3410530.3414373.

[18] F. V. Lima *et al.*, "At the Crossroads! Time to Start Taking Smartwatches Seriously," (in eng), *Am J Cardiol,* vol. 179, pp. 96-101, Sep 15 2022, doi: 10.1016/j.amjcard.2022.06.020.

[19] R. a. N. A. Marinescu, "Smartphone application for heart rate monitoring," presented at the 2017 E-Health and Bioengineering Conference (EHB), June ,, 2017.

[20] D. D. McManus *et al.*, "Design and Preliminary Findings From a New Electronic Cohort Embedded in the Framingham Heart Study," (in eng), *J Med Internet Res,* vol. 21, no. 3, p. e12143, Mar 1 2019, doi: 10.2196/12143.

[21] E. Y. Ding *et al.*, "Design, deployment, and usability of a mobile system for cardiovascular health monitoring within the electronic Framingham Heart Study," (in eng), *Cardiovasc Digit Health J,* vol. 2, no. 3, pp. 171-178, Jun 2021, doi: 10.1016/j.cvdhj.2021.04.001.

[22] A. Shcherbina *et al.*, "Accuracy in Wrist-Worn, Sensor-Based Measurements of Heart Rate and Energy Expenditure in a Diverse Cohort," (in eng), *J Pers Med,* vol. 7, no. 2, May 24 2017, doi: 10.3390/jpm7020003.

[23] A. Turki, K. Behbehani, K. Ding, R. Zhang, M. Li, and K. Bell, "Estimation of Heart Rate Variability Measures Using Apple Watch and Evaluating Their Accuracy: Estimation of Heart Rate Variability Measures Using Apple Watch," presented at the The 14th PErvasive Technologies Related to Assistive Environments Conference, Corfu, Greece, 2021. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3453892.3462647.

[24] O. Walch, Y. Huang, D. Forger, and C. Goldstein, "Sleep stage prediction with raw acceleration and photoplethysmography heart rate data derived from a consumer wearable device," (in eng), *Sleep,* vol. 42, no. 12, Dec 24 2019, doi: 10.1093/sleep/zsz180.

[25] S. R. James Robertson, *Volere Requirements Specification Template*. Atlantic Systems Guild, 2016, p. 90.

[26] *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure*, ISO/IEC 19505-1:2012, Object Managment Group, 2012. [Online]. Available: https://www.iso.org/standard/32624.html

[27] A. B. Robin Rombach, Dominik Lorenz, Patrick Esser, Björn Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," 2021, doi: 2112.10752.

[28] *The OAuth 2.0 Authorization Framework*, RFC 6749, The Internet Engineering Task Force, 2012. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749

[29] *JSON Web Token (JWT)*, RFC 7519, The Internet Engineering Task Force, 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

[30] S. Saeb, K. Kording, and D. C. Mohr, "Making Activity Recognition Robust against Deceptive Behavior," *PLoS One,* vol. 10, no. 12, p. e0144795, 2015, doi: 10.1371/journal.pone.0144795.